Symmetric Constrained Optimal Control: Theory, Algorithms, and Applications

by

Claus Robert Danielson

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2014

Abstract

Symmetric Constrained Optimal Control: Theory, Algorithms, and Applications

by

Claus Robert Danielson

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

This dissertation develops the theory of symmetry for constrained linear systems. We use symmetry to design model predictive controllers for constrained linear systems with reduced complexity.

The dissertation is divided into three parts. In the first part we review the relevant results from model predictive control and group theory. In particular we present algorithms and data structures from computational group theory used to efficiently search groups.

In the second part we develop the theory of symmetry for constrained linear systems and model predictive control problems. Symmetries of constrained linear systems are linear transformations that preserve the dynamics and constraints. A symmetry of a model predictive control problem is a symmetry of the underlying constrained system that also preserves the cost. We use a group theoretic formalism to derive properties of symmetric constrained linear systems and symmetric model predictive control problems. We prove conditions under which the model predictive controller is symmetric and present a procedure for efficiently computing the symmetries of constrained linear systems and model predictive control problems. Our method transforms the problem of finding generators for the symmetry group into a graph automorphism problem. These symmetries are used to design model predictive control algorithms with reduced complexity.

We also present two novel explicit model predictive control designs. Both reduce memory requirements by discarding symmetrically redundant pieces of the control-law. The control-law in the eliminated pieces can be reconstructed online using symmetry. We show that storing the symmetries of the problem requires less memory than storing the controller pieces.

In the third part of this dissertation we apply our symmetry theory to the battery balancing problem. We use symmetry to reduce the memory requirements for explicit model predictive controllers for seven battery-balancing hardware designs proposed in the literature. This application demonstrates that our symmetric controller designs can significantly reduce the memory requirements of explicit model predictive control. In particular for four

out of seven of the designs in our numerical study, the number of pieces in the symmetric controller did not increase as the battery pack-size was increased.

Contents

List of Figures

Chapter 1

Introduction

The chief forms of beauty are order and symmetry and definiteness, which the mathematical sciences demonstrate in a special degree.

-Aristotle

This dissertation develops the theory of symmetry for constrained linear systems. We use symmetry to design model predictive controllers with reduced complexity for constrained linear systems.

1.1 Motivation

Piecewise affine controllers are commonly used to control constrained linear systems. These controllers use different feedback and feedforward gains in different regions of the state-space. For instance, the controller may have more aggressive gains near the boundary of the state constraints and less aggressive gains near the equilibrium.

One method for generating a piecewise affine control-law is explicit model predictive control. Model predictive control is a popular technique for controlling constrained systems. In model predictive control, the control input is obtained by solving a constrained finite-time optimal control problem. If the constrained finite-time optimal control problem is solved online then the controller is called implicit. In explicit model predictive control the constrained finite-time optimal control problem is solved offline and stored as a look-up table. For linear systems subject to polytopic constraints with linear or quadratic cost functions, the explicit model predictive controller is a piecewise affine state-feedback control-law. This controller divides the state-space into polytopic regions, each of which has a different feedback and feedforward gain.

Explicit model predictive control provides a systematic method for constructing a piecewise affine control-law for a constrained linear system. Through proper design of the constrained finite-time optimal control problem, we can guarantee that this piecewise affine controller is stabilizing and satisfies the constraints on the system states and inputs. The disadvantage of using explicit model predictive control is that the piecewise affine control-law can have an excessive number of pieces. The number of controller pieces can grow exponentially with the number of constraints on the system. Often an embedded micro-controller will have insufficient memory to store the full explicit controller. Extensive research has been conducted on reducing the memory required for explicit model predictive controllers either by compressing the controller [1, 56, 78] or generating a sub-optimal explicit controller [84, 52, 58]

In this dissertation we use a property of the system, called symmetry, to compress the explicit model predictive controller without loss of performance. For a piecewise affine control-law symmetries are state-space and input-space transformations that relate controller pieces. Using symmetry we can discard some of the pieces of our controller. These discarded pieces can be reconstructed online using symmetry. Thus we can reduce the complexity of our controller and save memory without sacrificing performance. The amount of reduction in complexity depends on the number of symmetries possessed by the system. For systems with large symmetry groups the techniques presented in this dissertation can significantly reduce the complexity of the piecewise affine control-law produced using explicit model predictive control.

1.2 Literature Review

Symmetry has been used extensively in numerous fields to analyze and simplify complex problems. Symmetry is a standard analysis tool in physics [3] and chemistry [21]. In recent years, symmetry has been applied to optimization. In optimization theory symmetries are transformations of the decision space that do not change the cost or feasible region. See surveys [62, 35] for details. In [12] the authors showed that the optimal solution to a symmetric linear program lies in a lower-dimensional subspace called the fixed-space of the group. This result was extended to semi-definite programs and integer programs in [68] and [13] respectively. In [42] symmetric semi-definite programming was used to simplify the sum-of-squares decomposition of polynomials. In [4] symmetric semi-definite programming was applied to the optimal design of truss structures.

In this dissertation we use the concept of fundamental domains [38] to simplify the design of explicit model predictive controllers. In [40] the author used fundamental domains to simplify the linear programming relaxation of integer linear programs. In [54] the authors applied the related concept of orbitopes to the packing and partitioning integer programs.

Symmetry has also been used to analyze and simplify control design [79]. In control theory symmetries are transformations of the state-space, input-space, and output-space that map the state-space matrices to themselves. Many systems tend to exhibit large symmetry groups. For instance apartment and office buildings have many rooms laid out in regular patterns [55]. Complex machines are built from similar components arranged into specific patterns [82]. Multi-agent systems tend to be composed of identical agents or a small number of different types of agents [44].

For unconstrained symmetric linear systems, symmetry can be used to block-diagonalize the state space equations. Using group representation theory [75, 38] a basis for the state-space, input-space, and output-space can be found that decomposes the system into smaller decoupled subsystems. This basis is called the *symmetry adapted basis*. In [37, 47] it was shown that properties of the system such as controllability and stability can be determined by checking the smaller decoupled subsystems. In [25] symmetry adapted basis was used to simplify the design of $\mathcal{H}_2$ and $\mathcal{H}_\infty$ controllers. Using symmetry adapted basis, the authors designed $\mathcal{H}_2$ and $\mathcal{H}_\infty$ controllers for each of the decoupled subsystems. The $\mathcal{H}_2$ norm of the full system is the sum of the $\mathcal{H}_2$ norms of the decoupled subsystems. And the $\mathcal{H}_\infty$ norm of the full system is the max of the $\mathcal{H}_\infty$ norm of the decoupled subsystems. In [15] symmetry was used to simplify the process of finding the transition probabilities of Markov-Chains that produce the fastest convergence. The authors used symmetry adapted basis to decompose the fastest-mixing Markov chain problem into smaller semi-definite programs.

In [5] the authors exploit spatial invariance for controlling distributed systems. The authors show how to decompose an infinite dimensional semi-definite program into an infinite family of finite problems by using spatial Fourier transforms. In [73] the authors use these semi-definite programming formulations to design distributed controllers for spatial distributed systems. The interconnection pattern of the subsystems is the Cayley graph of an Abelian group. The authors use this group structure to simplify their semi-definite programs. In the subsequent paper [26] the authors extend their results to interconnections that are Cayley graphs of non-Abelian groups.

Unfortunately these approaches are not applicable for constrained systems. All previous methods use symmetry to find invariant subspaces that decompose the system into smaller subsystem. Each of these decoupled subsystems can be controlled independently. This simplifies the control design since these subsystems are much smaller than the original system. However, when applied to constrained systems, these transformations do not decompose the state and input constraints. Therefore the constraints recouple the subsystem. This negates the advantage of being able to design the controllers for the decouple subsystems independently.

In this dissertation we adopt a novel approach for exploiting symmetry to simplify control design for symmetric constrained linear systems. We use symmetry to relate the pieces of the piecewise controller. Thus we can eliminate redundant controller pieces and simplify the controller. This concept has not appeared in the literature since previous work has focused on linear controllers which have only a single piece.

1.3 Contributions and Structure

This dissertation is divided into three parts. The first part is a review of the relevant results from model predictive control and group theory. In particular, we present algorithms and data-structures from computational group theory used to efficiently search groups.

In the second part we develop the theory of symmetry for constrained linear systems and model predictive control problems. In Chapter 4 we define symmetry for constrained linear systems, piecewise affine on polytope controllers, and model predictive control problems. We discuss how these symmetry groups are related. We prove that symmetric model predictive control problems produce symmetric model predictive controllers. In particular we show that for linear model predictive control problems with linear or quadratic cost the symmetries permute the pieces of the piecewise affine on polytopes control-law. This result is used in Chapter 7 to reduce the memory requirements for explicit model predictive controllers. The main results of Chapter 4 have been previously published by the author in [27, 28, 29, 24].

In Chapter 5 we present a method for finding the symmetries of a constrained linear system. We call the problem of finding the symmetry group symmetry identification. We show how to transform this problem into a graph automorphism problem. We then extend this result to find the symmetries of linear model predictive control problems with linear and quadratic cost functions. The main results of Chapter 5 have been previously published by the author in [28, 29].

In Chapter 6 we define the concept of a fundamental domain. Intuitively, a fundamental domain is a type of subset that contains all the ‘information’ of the original set without the ‘redundancy’ due to symmetry. We provide a polynomial time algorithm for constructing a minimal polytopic fundamental domain of a set with respect to a group of symmetries. Next we present an algorithm for quickly mapping a point outside of the fundamental domain into the fundamental domain. This algorithm is necessary to implement the fundamental domain controller defined in Chapter 7. We prove our algorithms have polynomial complexity.

Finally in Chapter 7 we propose two explicit model predictive control designs, called the orbit controller and fundamental domain controller, that exploit symmetry to reduce controller complexity and save memory. The orbit controller uses symmetry to organize the controller pieces into equivalence classes called orbits. The orbit controller stores one representative controller piece from each orbit. The other controller pieces can be reconstructed online using symmetry. We analyze the computational and memory complexity of the orbit controller. We show that the orbit controller will never require more memory than the standard explicit controller. We show that the orbit controller offers significant memory savings when the number of group generators is small compared to the group size.

The fundamental domain controller solves the model predictive control problem on a subset of the state-space called a fundamental domain. A fundamental domain is a subset that contains at least one representative from each point orbit of a set. Since the fundamental domain is a subset of the state-space it intersects fewer critical regions of the controller. Therefore the fundamental domain controller has fewer pieces than the full explicit controller and requires less memory. We analyze the computational complexity of the fundamental domain controller. The main results of Chapter 7 have been previously published by the author in [27, 28, 29].

In the final part we apply our theory of symmetric model predictive control to the battery balancing problem. In Chapter 8 we define the battery balancing problem. We propose a control algorithm for solving the battery balancing problem and show that our controller is persistently feasible and asymptotically stable. We describe seven battery balancing hardware designs proposed in the literature. Finally we present simulation results for our control design. The main results of Chapter 8 have been previously published by the author in [30, 31, 70].

In Chapter 9 we use our symmetry theory to reduce the memory of explicit model predictive controllers for the battery balancing problem. We apply our symmetry theory to the seven battery balancing hardware designs presented in the previous section. Through a numerical study we demonstrate that the orbit controller and fundamental domain controller can significantly reduce the memory needed to store the explicit controller. We explain the intuition of how symmetry reduces the controller memory complexity in these examples.

Part I

Model Predictive Control and Symmetry

Chapter 2

Model Predictive Control

In this chapter we present the relevant material from model predictive control that will be used in this dissertation. We provide definitions and results for sets and polytopes, constrained systems, and model predictive controllers.

2.1 Sets and Polytopes

In this section, we present several topics from set theory. We define polytopes, p-collections, Minkowski functions, piecewise affine on polytope and p-collection functions, and set-valued functions. We present the relevant results from these topics used in this dissertation. The definitions and results in this section are taken from [14, 85, 45, 11, 39, 16].

An affine combination of a finite set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ is a point $\lambda_1 x_1 + \cdots + \lambda_k x_k \in \mathbb{R}^n$ where $\lambda_1 + \cdots + \lambda_k = 1$. For a set $\mathcal{K} \subseteq \mathbb{R}^n$ its affine hull is the set of all affine combinations of points in $\mathcal{K}$

$$\text{aff}(\mathcal{K}) = \Big\{ \textstyle\sum_{i=1}^k \lambda_i x_i : x_i \in \mathcal{K}, \sum_{i=1}^k \lambda_i = 1, \text{ for some } k \Big\}. \tag{2.1}$$

The affine hull $\text{aff}(\mathcal{K})$ is the translation of a linear subspace. The dimension of the affine hull is the dimension of that subspace.

A convex combination of a finite set of points $x_1, \ldots, x_k \in \mathbb{R}^n$ is a point $\lambda_1 x_1 + \cdots + \lambda_k x_k \in \mathbb{R}^n$ where $\lambda_1 + \cdots + \lambda_k = 1$ and $\lambda_i \geq 0$ for $i = 1, \ldots, k$. For a set $\mathcal{K} \subseteq \mathbb{R}^n$ its convex hull is the set of all convex combinations of points in $\mathcal{K}$

$$\text{conv}(\mathcal{K}) = \Big\{ \textstyle\sum_{i=1}^k \lambda_i x_i : x_i \in \mathcal{K}, \lambda_i \geq 0, \sum_{i=1}^k \lambda_i = 1, \text{ for some } k \Big\}. \tag{2.2}$$

A cone spanned by a finite set of points $\mathcal{K} = \{x_1, \ldots, x_k\}$ is the set

$$\text{cone}(\mathcal{K}) = \Big\{ \textstyle\sum_{i=1}^k \lambda_i x_i : x_i \in \mathcal{K}, \lambda_i \geq 0 \Big\}. \tag{2.3}$$

A closed half-space is the set

$$\mathcal{H}(h,k) = \big\{x \in \mathbb{R}^n : h^T x \leq k\big\}. \tag{2.4}$$

The Minkowski sum of two sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^n$ is defined as

$$\mathcal{X} \oplus \mathcal{Y} = \big\{x + y : x \in \mathcal{X}, y \in \mathcal{Y}\big\}. \tag{2.5}$$

An $\mathcal{H}$-polytope $\mathcal{X} \subseteq \mathbb{R}^n$ is the intersection of a finite number of closed half-spaces

$$\mathcal{X} = \big\{x \in \mathbb{R}^n : h_i^T x \leq k_i, i = 1, \dots, c\big\} = \big\{x \in \mathbb{R}^n : Hx \leq K\big\} \tag{2.6}$$

where $H \in \mathbb{R}^{c \times n}$ is a matrix with rows h_i^T and $K \in \mathbb{R}^c$ is a vector with elements k_i for $i = 1, \dots, c$. A half-space $\mathcal{H}(h_i, k_i)$ is non-redundant if removing it from the definition of the polytope $\mathcal{X}$ does not change the set. For a non-redundant half-space $\mathcal{H}(h_i, k_i)$ the set $\mathcal{H}(h_i, k_i) \cap \mathcal{X}$ is called a facet of $\mathcal{X}$.

If the polytope $\mathcal{X}$ contains the origin $0 \in \mathbb{R}^n$ in its interior then the half-space vectors can be normalized such that

$$\mathcal{X} = \big\{x \in \mathbb{R}^n : h_i^T x \leq 1, i = 1, \dots, c\big\} = \big\{x \in \mathbb{R}^n : Hx \leq \mathbf{1}\big\} \tag{2.7}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones.

If the polytope $\mathcal{X}$ is bounded then its half-space vectors h_i for $i = 1, \dots, c$ contain an affinely independent subset. In this case the matrix $H^T H$ is positive definite. Therefore we can define the transformation $T = (H^T H)^{-1/2}$ of the space $\mathbb{R}^n$ containing $\mathcal{X}$ such that the polytope $\mathcal{X}$ is given by

$$\mathcal{X} = \big\{x \in \mathbb{R}^n : \bar{H}x \leq \mathbf{1}\big\} \tag{2.8}$$

where $\bar{H} = HT$ and $\bar{H}^T \bar{H} = I$. This basis is called the regular basis for the polytope $\mathcal{X}$. We will use this basis to simplify our analysis of polytope symmetries.

Example 1. *Consider the triangle $\mathcal{X}$ shown in Figure 2.1a where*

$$\mathcal{X} = \left\{ x \in \mathbb{R}^2 : \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} x \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}. \tag{2.9}$$

Under the change of basis $T = (H^T H)^{-1/2}$ this triangle gets mapped to the triangle $\bar{\mathcal{X}}$ shown in Figure 2.1b. The triangle $\bar{\mathcal{X}}$ is an equilateral triangle and therefore more "symmetric" than the triangle shown in Figure 2.1a. This basis will simplify our analysis in later chapters when we define symmetry for polytopes. Specifically we will show that under this basis our symmetries are orthogonal matrices.

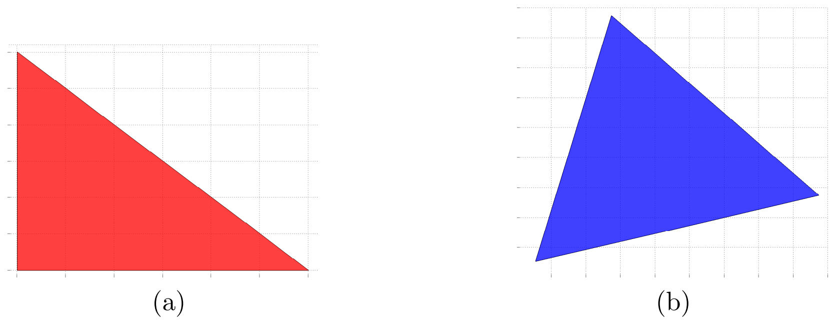


Figure 2.1: Example of regular basis. (a) Unregularized triangle $\mathcal{X}$. (b) Regularized triangle $\bar{\mathcal{X}}$.

A $\mathcal{V}$-polytope $\mathcal{X} \subseteq \mathbb{R}^n$ is the Minkowski sum of the convex hull of a finite set of points $\mathcal{V} = \{v_1, \dots, v_k\} \subset \mathbb{R}^n$ and the cone of a finite set of vectors $\mathcal{Y} = \{y_1, \dots, y_{k'}\} \subset \mathbb{R}^n$

$$\mathcal{X} = \text{conv}(\mathcal{V}) \oplus \text{cone}(\mathcal{Y}). \tag{2.10}$$

The following theorem relates $\mathcal{H}$ and $\mathcal{V}$ polytopes.

Theorem 1 (Farkas-Minkowski-Weyl). *All $\mathcal{V}$-polytopes are $\mathcal{H}$-polytopes and all $\mathcal{H}$-polytopes are $\mathcal{V}$-polytopes.*

The set of points $\mathcal{V}$ are called the extreme points of $\mathcal{X}$. The set of vectors $\mathcal{Y}$ are called the extreme-rays of $\mathcal{X}$. An extreme-point $v \in \mathcal{V}$ is non-redundant if removing it from $\mathcal{V}$ does not change the set $\mathcal{X} = \text{conv}(\mathcal{V})$. A non-redundant extreme-point is called a vertex.

If the polytope $\mathcal{X}$ is bounded then it is the convex hull of a finite number of points $\mathcal{X} = \text{conv}(\mathcal{V})$. The dimension $\dim(\mathcal{X})$ of a polytope $\mathcal{X}$ is the dimension of its affine hull. The polytope $\mathcal{X} \subseteq \mathbb{R}^n$ is called full-dimensional if $\dim(\mathcal{X}) = n$ and lower-dimensional otherwise. If a bounded polytope $\mathcal{X}$ is full-dimensional its vertices form an affinely independent set.

Let $\mathcal{X}$ be a polytope and $\mathcal{P} = \{p_i, \dots, p_r\} \subset \mathcal{X}$ be a finite set of points. The Voronoi cell $\mathcal{F}_{\mathcal{P}}(p_i)$ is defined as the set of points $x \in \mathcal{X}$ at least as close to p_i as any other point in $\mathcal{P}$

$$\mathcal{F}_{\mathcal{P}}(p_i) = \Big\{ x \in \mathcal{X} : \|x - p_i\| \leq \|x - p\|, \forall p \in \mathcal{P} \Big\}. \tag{2.11}$$

When distance is measured using the standard Euclidean norm the Voronoi cell $\mathcal{F}_{\mathcal{P}}(p_i)$ is a polytope. The Voronoi diagram of $\mathcal{P}$ is the collection of Voronoi cells $\mathcal{F}_{\mathcal{P}}(p_i)$ for each point $p_i \in \mathcal{P}$

$$\mathcal{F}_{\mathcal{P}}(\mathcal{P}) = \Big\{ \mathcal{F}_{\mathcal{P}}(p_1), \dots, \mathcal{F}_{\mathcal{P}}(p_r) \Big\}. \tag{2.12}$$

Functions and Sets

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a set (not necessarily a polytope) and $\Theta \in \mathbb{R}^{m \times n}$ a matrix. The set $\Theta\mathcal{X}$ is the image of $\mathcal{X}$ through the linear operator Θ

$$\Theta\mathcal{X} = \Big\{ \Theta x : x \in \mathcal{X} \Big\}. \tag{2.13}$$

Thus the set $\lambda\mathcal{X}$ for a scalar $\lambda \in \mathbb{R}$ is the image of $\mathcal{X}$ through the matrix $\lambda I \in \mathbb{R}^{n\times n}$. The Minkowski function $\Phi_{\mathcal{X}}(x)$ for a set $\mathcal{X}$ is defined as

$$\Phi_{\mathcal{X}}(x) = \inf\left\{\lambda \in \mathbb{R}_+ : x \in \lambda\mathcal{X}\right\}. \tag{2.14}$$

The Minkowski function has the following properties

1. If $\mathcal{X}$ contains the origin then $\Phi_{\mathcal{X}}(0) = 0$.
2. If $\mathcal{X}$ is bounded and contains the origin then $\Phi_{\mathcal{X}}(x)$ is positive definite i.e. $\Phi_{\mathcal{X}}(x) \geq 0$ for all $x \in \mathbb{R}^n$ and $\Phi_{\mathcal{X}}(x) = 0$ if and only if $x = 0$.
3. For any $\alpha > 0$ the Minkowski function satisfies $\Phi_{\mathcal{X}}(\alpha x) = \alpha\Phi_{\mathcal{X}}(x)$.
4. If the set $\mathcal{X}$ is balanced ($\mathcal{X} = -\mathcal{X}$) then $\Phi_{\mathcal{X}}(\alpha x) = \alpha\Phi_{\mathcal{X}}(x)$ for any $\alpha \in \mathbb{R}$.
5. If the set $\mathcal{X}$ is convex then the Minkowski function satisfies $\Phi_{\mathcal{X}}(x+y) \leq \Phi_{\mathcal{X}}(x)+\Phi_{\mathcal{X}}(y)$.
6. If the set $\mathcal{X}$ is convex, bounded, balanced, and contains the origin then the Minkowski function $\Phi_{\mathcal{X}}(x)$ is a norm on $\mathbb{R}^n$.
7. If the set $\mathcal{X}$ is a cross-polytope $\mathcal{X} = \text{conv}\{\pm e_1, \dots, \pm e_n\}$ where $e_i \in \mathbb{R}^n$ are the standard basis vectors then the Minkowski function is the 1-norm $\Phi_{\mathcal{X}}(x) = \|x\|_1$.
8. If the set $\mathcal{X}$ is a hypersphere $\mathcal{X} = \{x \in \mathbb{R}^n : x^T x \leq 1\}$ then Minkowski function is the 2-norm $\Phi_{\mathcal{X}}(x) = \|x\|_2$.
9. If the set $\mathcal{X}$ is a hypercube $\mathcal{X} = \{x \in \mathbb{R}^n : -\mathbf{1} \leq x \leq \mathbf{1}\}$ then Minkowski function is the ∞-norm $\Phi_{\mathcal{X}}(x) = \|x\|_\infty$.
10. If the set $\mathcal{X} = \{x : Hx \leq \mathbf{1}\}$ is a polytope then the Minkowski function is the solution to the linear program

$$\Phi_{\mathcal{X}}(x) = \underset{x,\lambda\geq 0}{\text{minimize}} \quad \lambda \tag{2.15a}$$

$$\text{subject to } Hx \leq \lambda\mathbf{1}. \tag{2.15b}$$

A p-collection is the union of a finite number of polytopes. A collection of sets $\{\mathcal{X}_i\}_{i=1}^p$ is a partition of $\mathcal{X}$ if it satisfies $\bigcup_{i=1}^N \mathcal{X}_i = \mathcal{X}$ and $\text{int}(\mathcal{X}_i) \cap \text{int}(\mathcal{X}_j) = \varnothing$ for all $i \neq j$. The partition $\{\mathcal{X}_i\}_{i=1}^p$ is called a polytope partition if its elements $\mathcal{X}_i$ are polytopes. The partition $\{\mathcal{X}_i\}_{i=1}^p$ is called a p-collection partition if its elements $\mathcal{X}_i$ are p-collections.

A continuous function $f : \mathcal{X} \to \mathcal{Y}$ is called piecewise affine if there exists a partition $\{\mathcal{X}_i\}_{i=1}^p$ of $\mathcal{X}$ such that

$$f(x) = \begin{cases} A_1 x + b_1 & \text{for } x \in \mathcal{X}_1 \\ \quad\vdots & \\ A_p x + b_p & \text{for } x \in \mathcal{X}_p \end{cases}. \tag{2.16}$$

where p is the partition $\{\mathcal{X}_i\}_{i=1}^p$ size. The function is called piecewise affine on polytopes if the partition $\{\mathcal{X}_i\}_{i=1}^p$ is a polytope partition. It is called piecewise affine on p-collections if the partition $\{\mathcal{X}_i\}_{i=1}^p$ is a p-collection partition. If the affine parameters $A_i = A_j$ and $b_i = b_j$ are the same on the p-collection regions $\mathcal{X}_i$ and $\mathcal{X}_j$ then we can merge $\mathcal{X}_i \cup \mathcal{X}_j$ these regions to obtain a new expression for f with a smaller p-collection partition $\{\mathcal{X}_i\}_{i=1}^p$. We call a p-collection partition $\{\mathcal{X}_i\}_{i=1}^N$ minimal if there does not exist another p-collection partition for the function f with fewer pieces p. The minimal p-collection partition is unique.

Proposition 1. *Let $f : \mathcal{X} \to \mathcal{Y}$ be a piecewise affine on p-collection function. Then the minimal p-collection partition $\{\mathcal{X}_i\}_{i=1}^N$ for f is unique.*

Proof. Suppose $\{\mathcal{Z}_i\}_{i=1}^N$ is a different minimal p-collection partition for f. Then for some region $\mathcal{Z}_i \in \{\mathcal{Z}_i\}_{i=1}^N$ we have $\mathcal{Z}_i \neq \mathcal{X}_j$ for all $j = 1, \dots, N$. Without loss of generality we assume that $\mathcal{Z}_i \not\subset \mathcal{X}_j$ for any $j = 1, \dots, N$ (otherwise switch $\mathcal{Z}_i$ and $\mathcal{X}_j$ in the following argument). Then there exists at least two set $\mathcal{X}_j$ and $\mathcal{X}_k$ in the partition $\{\mathcal{X}_i\}_{i=1}^N$ such that $\mathcal{Z}_i \cap \mathcal{X}_j$ and $\mathcal{Z}_i \cap \mathcal{X}_k$are full-dimensional since $\mathcal{Z}_i \subseteq \mathcal{X}$ and $\{\mathcal{X}_i\}_{i=1}^N$ partitions $\mathcal{X}$. However this implies $f(x) = A_i x + b_i = A_j x + b_j = A_k x + b_k$ on the affinely independent set of points contained in $\mathcal{Z}_i \cap \mathcal{X}_j$ and $\mathcal{Z}_i \cap \mathcal{X}_k$. Thus $A_j = A_k$ and $b_j = b_k$. Therefore we can merge regions $\mathcal{X}_j$ and $\mathcal{X}_k$ to produce a partition with $N-1$ pieces which contradicts the minimality of the partition $\{\mathcal{X}_i\}_{i=1}^N$. □

This proposition will allow us to prove properties of the function f and partition $\{\mathcal{X}_i\}_{i=1}^N$ using uniqueness arguments.

Let $2^{\mathcal{X}}$ denote the power set of $\mathcal{X}$. A set valued function $f : \mathcal{X} \to 2^{\mathcal{Y}}$ is a function from $\mathcal{X}$ to the power-set $2^{\mathcal{Y}}$ of $\mathcal{Y}$ i.e. $f(x) \subseteq \mathcal{Y}$. A set valued function $f : \mathcal{X} \to 2^{\mathcal{Y}}$ is upper-semicontinuous at the point $x \in \mathcal{X}$ if for any open neighborhood $\mathcal{V} \subset \mathcal{Y}$ of $f(x) \subset \mathcal{V}$ there exists an open neighborhood $\mathcal{U}$ of $x \in \mathcal{X}$ such that for all $z \in \mathcal{U}$ we have $f(z) \subseteq \mathcal{V}$. Semicontinuity is also called hemicontinuity in some texts [39]. The following theorem gives a condition for checking upper-semicontinuity.

Theorem 2 (Closed Graph Theorem)**.** *Let $f : \mathcal{X} \to 2^{\mathcal{Y}}$ be a set-valued function with $f(x)$ closed for all $x \in \mathcal{X}$ with closed domain $\mathcal{X}$, and compact range $\mathcal{Y}$. Then the function f is upper-semicontinuous if and only if its graph*

$$\operatorname{graph}(f) = \big\{(x, y) \in \mathcal{X} \times \mathcal{Y} : y \in f(x)\big\} \tag{2.17}$$

is closed in $\mathcal{X} \times \mathcal{Y}$.

2.2 Constrained Systems

The definitions and results in this section are taken from [14, 11].

In this section we consider two types of discrete-time systems: autonomous systems

$$x(t+1) = f_a\big(x(t)\big) \tag{2.18}$$

and systems with controlled inputs

$$x(t+1) = f\big(x(t), u(t)\big). \tag{2.19}$$

Both types of systems are subject to constraints on the state and input

$$x(t) \in \mathcal{X} \tag{2.20a}$$
$$u(t) \in \mathcal{U}. \tag{2.20b}$$

For the autonomous system (2.18) we define $\text{Pre}(\mathcal{S})$ as the set of states x that will reach the target set $\mathcal{S}$ in one time-step

$$\text{Pre}(\mathcal{S}) = \big\{x \in \mathbb{R}^n : f_a(x) \in \mathcal{S}\big\}. \tag{2.21}$$

For the controlled system (2.19) we define $\text{Pre}(\mathcal{S})$ as the set of states x for which there exists a feasible control input $u \in \mathcal{U}$ that will drive the system to the target set $\mathcal{S}$ in one time-step

$$\text{Pre}(\mathcal{S}) = \big\{x \in \mathbb{R}^n : \exists u \in \mathcal{U}, \text{ such that } f(x,u) \in \mathcal{S}\big\}. \tag{2.22}$$

These sets are also called the backward reachable sets.

For the autonomous system (2.18) we define $\text{Reach}(\mathcal{S})$ as the set of states x that the system could reach in one time-step starting from some state inside the set $\mathcal{S}$

$$\text{Reach}(\mathcal{S}) = \big\{y \in \mathbb{R}^n : \exists x \in \mathcal{S} \text{ such that } y = f_a(x)\big\}. \tag{2.23}$$

For the controlled system (2.19) we define $\text{Reach}(\mathcal{S})$ as the set of states x that the system can reach in one time-step starting from some state inside the set $\mathcal{S}$

$$\text{Reach}(\mathcal{S}) = \big\{y \in \mathbb{R}^n : \exists x \in \mathcal{S}, \exists u \in \mathcal{U}, \text{ such that } y = f(x,u)\big\}. \tag{2.24}$$

A set $\mathcal{O} \subseteq \mathcal{X}$ is positive invariant for the autonomous system (2.18) subject to the constraints (2.20) if $x \in \mathcal{O}$ implies $f_a(x) \in \mathcal{O}$. This means that if the state $x(t)$ starts in the positive invariant set $x(0) \in \mathcal{O}$ it remains in the positive invariant set $x(t) \in \mathcal{O}$ for all time $t \in \mathbb{N}$. The set $\mathcal{O}_\infty$ is the maximal positive invariant for the autonomous system (2.18) subject to the constraints (2.20) if $\mathcal{O}_\infty$ is positive invariant and $\mathcal{O}_\infty$ contains all positive invariant sets contained in $\mathcal{X}$.

A set $\mathcal{C} \subseteq \mathcal{X}$ is control invariant for the autonomous system (2.18) subject to the constraints (2.20) if $x \in \mathcal{C}$ implies there exists $u \in \mathcal{U}$ such that $f(x,u) \in \mathcal{C}$. This means that if the state $x(t)$ is inside the control invariant set $x(t) \in \mathcal{C}$ there exists a feasible control input $u(t) \in \mathcal{U}$ that keeps the state in the control invariant set $x(t+1) \in \mathcal{C}$ for all time $t \in \mathbb{N}$. The set $\mathcal{C}_\infty$ is the maximal control invariant for the controlled system (2.19) subject to the constraints (2.20) if $\mathcal{C}_\infty$ is control invariant and $\mathcal{C}_\infty$ contains all control invariant sets contained in $\mathcal{X}$.

2.3 Model Predictive Control

The definitions and results in this section are taken from [14, 72].

Model Predictive Control is described in Algorithm 1. At each sample-time $t \in \mathbb{N}$ the control input $u(t)$ is obtained by solving problem (2.25) where $x_0 = x(t)$ is the current state of the system, N is the prediction horizon, $q(x,u)$ is the stage cost, $p(x)$ is the terminal cost, and $\mathcal{X}_N$ is the terminal constraint set. Solving (2.25) produces an optimal open-loop control input sequence $u_0^\star, \dots, u_{N-1}^\star$ over the horizon N. The first element of this sequence is implemented $u(t) = u_0^\star$. The process repeats at time $t+1$ with the updated state $x(t+1) = f\big(x(t), u(t)\big)$.

Algorithm 1 Model Predictive Control Algorithm

```
1: Measure (or estimate) the current state $x_0 = x(t)$
2: Solve the constrained finite-time-optimal control problem (2.25) to obtain an optimal
   open-loop control sequence $u_0^\star, \dots, u_{N-1}^\star$.
```

$$
\begin{aligned}
J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & p(x_N) + \textstyle\sum_{k=0}^{N-1} q(x_k, u_k) && \text{(2.25a)}\\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k) && \text{(2.25b)}\\
& x_k \in \mathcal{X}, u_k \in \mathcal{U} && \text{(2.25c)}\\
& x_N \in \mathcal{X}_N && \text{(2.25d)}\\
& x_0 = x(t) && \text{(2.25e)}
\end{aligned}
$$

```
3: Implement $u(t) = u_0^\star$.
```

Designing a model predictive controller involves selecting a cost function

$$
p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) \tag{2.26}
$$

and terminal constraint set $\mathcal{X}_N$ to produce the desired closed-loop system behavior. Three of the issues that must be considered when designing a model predictive controller are persistent feasibility, stability, and real-time implementation. We will briefly discuss these issues in the next three subsections.

Persistent Feasibility

At each sample-time $t \in \mathbb{N}$ the model predictive controller solves the optimization problem (2.25) which depends on the current state $x(t)$. For certain states $x(t)$ the model predictive control problem (2.25) is infeasible. Indeed even when the model predictive control problem (2.25) is feasible it may produce a control input $u(t) = u_0^\star$ that drives the system to a state $x(t+1) = f(x(t), u(t))$ where (2.25) is infeasible. Therefore we would like to guarantee that the closed-loop system is persistently feasible.

Definition 1 (Persistent Feasibility)**.** Let $\mathcal{X}_0 \subseteq \mathcal{X}$ be the set of states x for (2.25) is feasible. Then the system (2.19) in closed-loop with the model predictive controller in Algorithm 1 is persistently feasible if $f(x(t), u(t)) \in \mathcal{X}_0$ for all $x(t) \in \mathcal{X}_0$ where $u(t) = u_0^\star(x(t))$ is the model predictive control input.

This definition says that the model predictive controller is persistently feasible if the feasible set $\mathcal{X}_0$ of (2.25) is a positive invariant set for the system (2.19) subject to the constraints (2.20) in closed-loop with the model predictive controller $u(t) = u_0^\star(x(t))$. The following theorem gives a condition for persistent feasibility.

Theorem 3. *If the terminal set $\mathcal{X}_N$ is control invariant for the system* (2.19) *subject to the constraints* (2.20) *then the system* (2.19) *in closed-loop with the model predictive controller in Algorithm 1 is persistently feasible.*

Therefore we can guarantee the closed-loop system is persistently feasible by choosing a control invariant set for the terminal constraint set $\mathcal{X}_N$.

Stability

An equilibrium of the discrete-time autonomous system (2.18) is a state x_e that is a fixed-point of the dynamics $f_a(x_e) = x_e$. An equilibrium x_e is stable if for states $x(t)$ that starts near the equilibrium $\|x(0) - x_e\| < \delta$ the state stays near the equilibrium $\|x(t) - x_e\| < \epsilon$. The equilibrium is asymptotically stable if in addition to being stable the state $x(t)$ converges to the equilibrium. The formal definition is given below.

Definition 2 (Lyapunov Stability)**.** The equilibrium x_e is stable if for every $\epsilon > 0$ there exists a $\delta > 0$ such that if $\|x(0) - x_e\| < \delta$ then $\|x(t) - x_e\| < \epsilon$ for all $t \in \mathbb{N}$. The equilibrium x_e is asymptotically stable if it is stable and there exists a neighborhood Ω of x_e such that if $x(0) \in \Omega$ then $\lim_{t\to\infty} x(t) = x_e$.

We will assume $x_e = 0$ is an equilibrium of the system (2.18). The following theorem is used to prove the stability of this equilibrium.

Theorem 4. *Consider the equilibrium point $x_e = 0$ of the discrete-time autonomous system* (2.18)*. Let $\Omega \subset \mathbb{R}^n$ be a compact set containing the origin. Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuous function such that*

1. *$V(0) = 0$ and $V(x) > 0$ for all $x \in \Omega \setminus \{0\}$*
2. *$V\big(f_a(x)\big) - V(x) \leq -\alpha(x)$ for all $x \in \Omega \setminus \{0\}$.*

where α is a continuous positive definite function. Then $x = 0$ is asymptotically stable on Ω.

The function $V(x)$ in Theorem 4 is called a Lyapunov function. The set Ω is called the domain of attraction.

The following theorem gives a condition for the stability of the origin $x_e = 0$ for the controlled system (2.19) in closed-loop with the model predictive controller Algorithm 1.

Theorem 5. *Consider the system* (2.19) *in closed-loop with the model predictive controller in Algorithm 1. Let $p(x)$ and $q(x,u)$ be positive definition functions. Let the sets $\mathcal{X}$, $\mathcal{X}_N$, and $\mathcal{U}$ be closed and contain the origin in their interiors. Let $\mathcal{X}_N \subseteq \mathcal{X}$ be a control invariant set. Suppose that for all $x \in \mathcal{X}_N$ there exists $u \in \mathcal{U}$ such that $f(x,u) \in \mathcal{X}_N$ and*

$$p(f(x,u)) - p(x) \leq -q(x,u). \tag{2.27}$$

Then the origin is asymptotically stable with domain of attraction $\mathcal{X}_0$.

This theorem says that we can stabilize the origin by choosing the terminal cost $p(x)$ to be a Lyapunov function for the system.

Real-time Implementation

The model predictive controller described in Algorithm 1 requires solving an optimization problem at each sample-time $t \in \mathbb{N}$. This can be challenging on embedded hardware or when the controller is running at a high sample-rate. One method for reducing the computational load is explicit model predictive control where the solution to the model predictive control problem (2.25) is precomputed as a piecewise affine function of the state vector. In this section we briefly describe explicit model predictive control.

Problem (2.25) can be written as a multi-parametric program [6]

$$\underset{U}{\text{minimize}}\ J(U, x_0) \tag{2.28a}$$

$$\text{subject to } U \in \mathcal{T}(x_0) \tag{2.28b}$$

$$x_0 \in \mathcal{X} \tag{2.28c}$$

where $U = [u_0^T, \ldots, u_{N-1}^T]^T$ is the input trajectory over the horizon N, $J(U, x_0)$ is the cost function, and $\mathcal{T}(x_0)$ is the set of feasible input trajectories U that satisfy state and input constraints (2.25c) and terminal constraint (2.25d) under the dynamics (2.25b).

Consider the case where the dynamics (2.25b) are linear, and the constraints on the state and input (2.25c) and terminal state (2.25d) are polytopic. If the cost function (2.25a) is quadratic then (2.28) is a multi-parametric quadratic program. If the cost function (2.25a) is linear then (2.28) is a multi-parametric linear program.

In [6] it was shown that if (2.28) is a multi-parametric linear or quadratic program then it has a continuous piecewise affine solution

$$U^\star(x_0) = \begin{cases} \mathbf{F}_1 x_0 + \mathbf{G}_1 & \text{for } x_0 \in \mathcal{CR}_1 \\ \quad\vdots & \\ \mathbf{F}_p x_0 + \mathbf{G}_p & \text{for } x_0 \in \mathcal{CR}_p \end{cases} \tag{2.29}$$

where $\mathcal{CR}_i \subseteq \mathcal{X}$ are polytopes called critical regions, and $\mathbf{F}_i \in \mathbb{R}^{Nm\times n}$ and $\mathbf{G}_i \in \mathbb{R}^{Nm}$ are constant matrices. The set of initial conditions x_0 for which problem (2.28) is feasible is denoted by $\mathcal{X}_0$. The critical regions $\{\mathcal{CR}_i\}_{i=1}^p$ partition the set $\mathcal{X}_0$, i.e. $\cup_{i=1}^p \mathcal{CR}_i = \mathcal{X}_0$ and $\text{int}(\mathcal{CR}_i) \cap \text{int}(\mathcal{CR}_j) = \varnothing$ for all $i \neq j$.

The optimal state-feedback piecewise affine model predictive control-law can be obtained by extracting the first m terms of the optimal input trajectory $U^\star(x_0)$ and setting $x_0 = x$

$$u_0^\star(x) = \kappa(x) = \begin{cases} F_1 x + G_1 & \text{for } x \in \mathcal{R}_1 \\ \vdots & \\ F_r x + G_r & \text{for } x \in \mathcal{R}_r \end{cases} \tag{2.30}$$

where $F_i \in \mathbb{R}^{m\times n}$ and $G_i \in \mathbb{R}^m$ are the optimal feedback and feedforward gains. The regions $\mathcal{R}_i \subseteq \mathcal{X}$ are unions of critical region $\mathcal{CR}_j$ where the first m terms of $U^\star(x)$ are identical. We refer to the triple $(F_i, G_i, \mathcal{R}_i)$ as the i-th piece of the controller $\kappa(x)$ and $\mathcal{I} = \{1, \ldots, r\}$ is the index set of the controller pieces.

It is important to make a distinction between the optimal control-law $u_0^\star(x)$ and its piecewise affine expression $\kappa(x)$ defined in (2.30). The optimal control-law $u_0^\star : \mathcal{X}_0 \to \mathcal{U}$ is a function from the set of feasible states $\mathcal{X}_0$ to the set of feasible control inputs $\mathcal{U}$. The piecewise affine controller $\kappa(x)$ is a representation of this function. Even when the optimal control-law $u_0^\star(x)$ is unique the piecewise affine expression (2.30) is not unique.

A simple implementation of controller (7.2) is shown in Algorithm 2. First the algorithm determines the region $\mathcal{R}_i$ that contains the measured state $x \in \mathcal{R}_i$. This step is called the point location problem. Next the optimal control $u_0^\star(x) = F_i x + G_i$ is calculated. The computational complexity of this algorithm is dominated by the point location problem which requires $O(|\mathcal{I}|)$ set membership tests $x \in \mathcal{R}_i$ of complexity $O(nc_i)$ where c_i is the number of constraints that defines region $\mathcal{R}_i$. Many improvements of this basic implementation can be found in the literature [80, 50, 49, 23, 59].

Algorithm 2 Standard Implementation of Explicit Model Predictive Controller

```
Input: Measured State $x$
Output: Optimal Control $u$
1: for $i \in \mathcal{I}$ do
2:     if $x \in \mathcal{R}_i$ then return $i$
3:     end if
4: end for
5: $u_0^\star = F_i x + G_i$
```

Chapter 3

Group Theory

In this chapter we define groups and their basic properties. We also introduce important algorithms and data-structures used to manipulate groups of matrices or permutations.

3.1 Definitions and Basic Results

In this section we provide the relevant terminology and results from group theory used in this dissertation. The definitions and results in this section are taken from [60, 48, 74, 34].

A group $(\mathcal{G}, \circ)$ is a set $\mathcal{G}$ along with an associative binary operator $\circ$ that satisfies the following axioms

1. Closure: For every $g, h \in \mathcal{G}$ we have $h \circ g \in \mathcal{G}$.
2. Identity: There exists a unique identity element $e \in \mathcal{G}$ such that for every $g \in \mathcal{G}$ we have $e \circ g = g \circ e = g$.
3. Inverse: For every $g \in \mathcal{G}$ there exist an element g^{-1} such that $g \circ g^{-1} = g^{-1} \circ g = e$.

For notational simplicity we will drop the $\circ$ and write gh for $g \circ h$. A group $\mathcal{G}$ that contains only the identity element $\mathcal{G} = \{e\}$ is called trivial.

This dissertation deals with groups of matrices and permutations. In matrix groups the operator $\circ$ is matrix multiplication and the identity element is the identity matrix I. In permutation groups the operator $\circ$ is function composition and the identity element is the identity permutation e. Permutations will be described using the Cayley notation. For instance the permutation π such that $\pi(1) = 2$, $\pi(2) = 3$, and $\pi(3) = 1$ will be denoted as

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}. \tag{3.1}$$

A permutation matrix $\Pi \in \mathbb{R}^{n \times n}$ is a square binary matrix that has exactly one 1 in each row and column. Each permutation π on $\{1, \ldots, n\}$ corresponds to a permutation matrix

$\Pi \in \mathbb{R}^{n \times n}$ given by

$$\Pi_{ij} = \begin{cases} 1 & \text{if } \pi(i) = j \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

A signed permutation matrix $\Sigma \in \mathbb{R}^{n \times n}$ is a square binary matrix that has exactly one non-zero element in each row and column where the non-zero element is ± 1. Each signed permutation matrix can be written as the product of a permutation matrix and a diagonal matrix with ± 1 on the diagonals.

A group $\mathcal{G}$ acts on a set $\mathcal{X}$ if its elements $g \in \mathcal{G}$ are functions $g : \mathcal{X} \to \mathcal{X}$ on the set $\mathcal{X}$. The orbit $\mathcal{G}x$ of a point $x \in \mathcal{X}$ is the image $g(x)$ of the point $x \in \mathcal{X}$ under every function $g \in \mathcal{G}$ in the group $\mathcal{G}$

$$\mathcal{G}x = \{g(x) \in \mathcal{X} : g \in \mathcal{G}\} \subseteq \mathcal{X}. \tag{3.3}$$

One set that the group $\mathcal{G}$ acts on is itself $\mathcal{X} = \mathcal{G}$ with action $g(h) = gh$ where $h \in \mathcal{X} = \mathcal{G}$. Under this action the orbit $\mathcal{G}h$ of a point $h \in \mathcal{X} = \mathcal{G}$ is the group $\mathcal{G}h = \mathcal{G}$. In the next section we will use this result to show how to reconstruct the group $\mathcal{G} = \langle \mathcal{S} \rangle$ from its generators $\mathcal{S}$ using an orbit search algorithm.

The stabilizer subgroup $\mathcal{G}_x$ of a point $x \in \mathcal{X}$ is the largest subset of $\mathcal{G}$ that fixes the point $x \in \mathcal{X}$

$$\mathcal{G}_x = \{g \in \mathcal{G} : g(x) = x\} \subseteq \mathcal{G}. \tag{3.4}$$

The stabilizer $\mathcal{G}_x$ is a group. For a finite set $\mathcal{B} \subseteq \mathcal{X}$ the point-wise stabilizer subgroup $\mathcal{G}_\mathcal{B}$ is the subgroup that fixes $\mathcal{B}$ point-wise

$$\mathcal{G}_\mathcal{B} = \{g \in \mathcal{G} : g(x) = x, x \in \mathcal{B}\}. \tag{3.5}$$

This size of the orbit $\mathcal{G}x$ and stabilizer $\mathcal{G}_x$ are related by the orbit-stabilizer theorem.

Theorem 6 (Orbit-Stabilizer)**.** $|\mathcal{G}| = |\mathcal{G}x||\mathcal{G}_x|$.

A set $\mathcal{S} \subseteq \mathcal{G}$ is a called a generating set if every element of $\mathcal{G}$ can be expressed as a product of the elements of $\mathcal{S}$ and their inverses. If the group $\mathcal{G}$ is finite then its elements can be expressed as products of the elements of $\mathcal{S}$. The group generated by the set $\mathcal{S}$ is denoted by $\langle \mathcal{S} \rangle$. Equivalently $\mathcal{G} = \langle \mathcal{S} \rangle$ is the smallest group that contains $\mathcal{S}$. The trivial group is generated by the empty set $\langle \varnothing \rangle$. Computational group theory is primarily concerned with efficiently manipulating groups using only their generators.

A group homomorphism is a function $f : \mathcal{G} \to \mathcal{H}$ between groups that preserves the group structure $f(g)f(h) = f(gh)$ for all $g, h \in \mathcal{G}$. A bijective group homomorphism f is called a group isomorphism and the groups $\mathcal{G}$ and $\mathcal{H}$ are called isomorphic. If $\mathcal{G}$ and $\mathcal{H}$ are isomorphic groups and $\mathcal{S} \subseteq \mathcal{G}$ generates $\mathcal{G} = \langle \mathcal{S} \rangle$ then $f(\mathcal{S})$ generates $\mathcal{H} = \langle f(\mathcal{S}) \rangle$. The kernel $\mathrm{Ker}(f) = \{g \in \mathcal{G} : f(g) = e\}$ of a group homomorphism $f : \mathcal{G} \to \mathcal{H}$ is the subgroup $\mathrm{Ker}(f) \subseteq \mathcal{G}$ of elements of $\mathcal{G}$ that are mapped to the identity element $e \in \mathcal{H}$.

Let $\mathcal{H} \subseteq \mathcal{G}$ be a subgroup of $\mathcal{G}$. The sets

$$g\mathcal{H} = \{gh : h \in \mathcal{H}\} \tag{3.6}$$

are called the cosets of $\mathcal{H}$ in $\mathcal{G}$. The set of cosets is denoted by $\mathcal{G}/\mathcal{H} = \{g\mathcal{H} : g \in \mathcal{G}\}$. The cosets $g\mathcal{H}$ partition the group $\mathcal{G}$ into disjoint subsets

$$\mathcal{G} = \bigsqcup_{\mathcal{G}/\mathcal{H}} g\mathcal{H}. \tag{3.7}$$

There are $|\mathcal{G}/\mathcal{H}| = |\mathcal{G}|/|\mathcal{H}|$ unique cosets $g\mathcal{H}$. Intuitively this mean that group $\mathcal{G}$ contains $|\mathcal{G}|/|\mathcal{H}|$ copies of the subgroup $\mathcal{H}$. This repetitive structure in the group $\mathcal{G}$ is exploited in many computational group theory algorithms. With abuse of notation we will use $\mathcal{G}/\mathcal{H}$ to denote a set of coset representatives $\mathcal{G}/\mathcal{H} = \{u_1, \ldots, u_m\}$ such that $\mathcal{G} = \bigsqcup_{i=1}^{m} u_i\mathcal{H}$.

Equation 3.7 implies the following important property about groups.

Proposition 2. *Let $\mathcal{H} \subseteq \mathcal{G}$ be a subgroup of $\mathcal{G}$. Then every element $g \in \mathcal{G}$ of the group $\mathcal{G}$ can be decomposed into the product $g = uh$ of a coset representative $u \in \mathcal{G}/\mathcal{H}$ and an element of the subgroup $h \in \mathcal{H}$.*

Proposition 2 is instrumental in allowing us to quickly and efficiently search groups.

Common Groups

In this section we introduce several common groups that will appear in this dissertation.

- General Linear Group $\mathrm{GL}(n)$: The general linear group $\mathrm{GL}(n)$ is the set of all $n \times n$ real invertible matrices $\Theta \in \mathbb{R}^{n\times n}$. This group has infinite order.

- Orthogonal Group $\mathrm{O}(n)$: The orthogonal group $\mathrm{O}(n) \subset \mathrm{GL}(n)$ is the set of matrices $\Theta \in \mathrm{GL}(n) \subset \mathbb{R}^{n\times n}$ that preserve the euclidean norm $\|\Theta x\|_2 = \|x\|_2$. The elements $\Theta \in \mathrm{O}(n)$ have the property that $\Theta^{-1} = \Theta^T$. Equivalently $\Theta\Theta^T = \Theta^T\Theta = I$. This group has infinite order.

- Special orthogonal group $\mathrm{SO}(n)$: The special orthogonal group $\mathrm{SO}(n) \subset \mathrm{O}(n)$ is the subgroup of orthogonal matrices $\Theta \in \mathrm{O}(n) \subset \mathbb{R}^{n\times n}$ with determinant $\det(\Theta) = +1$. In $n = 2$ and $n = 3$ dimensions the special orthogonal matrices $\Theta \in \mathrm{SO}(n) \subset \mathbb{R}^{n\times n}$ are proper rotations. This group has infinite order.

- Hyperoctahedral Group $\mathfrak{B}_n$: The hyperoctahedral group $\mathfrak{B}_n \subset \mathrm{GL}(n)$ is the set of matrices $\Theta \in \mathrm{GL}(n) \subset \mathbb{R}^{n\times n}$ that preserve the 1-norm $\|\Theta x\|_1 = \|x\|_1$ and ∞-norm $\|\Theta x\|_\infty = \|x\|_\infty$. The hyperoctahedral group $\mathfrak{B}_n \subset \mathrm{GL}(n)$ is the set of all signed permutation matrices. This group has finite order $|\mathfrak{B}_n| = 2^n n!$.

- Symmetric Group $\mathfrak{S}_n$: The symmetric group $\mathfrak{S}_n$ is the set of all permutations π on the set $\{1, \dots, n\}$. Equivalently $\mathfrak{S}_n \subset \mathfrak{B}_n \subset \mathrm{GL}(n)$ is the set of all permutation matrices $\Pi \in \mathbb{R}^{n \times n}$. For a finite set $\mathcal{X}$ we will denote the set of all permutations on this set $\mathcal{X}$ by $\mathfrak{S}_{\mathcal{X}}$. The group $\mathfrak{S}_n$ has finite order $|\mathfrak{S}_n| = n!$.
- Cyclic group $\mathcal{C}_n$: The cyclic-n group $\mathcal{C}_n$ is an abstract group generated by a single element and has order $|\mathcal{C}_n| = n$. In $\mathbb{R}^n$ the representations of the cyclic group are rotation matrices $\Theta \in \mathrm{SO}(n)$ by a fixed increment $2\pi/n$.
- Dihedral group $\mathcal{D}_n$: The dihedral-n group $\mathcal{D}_n$ is an abstract group with two generators; a rotation and reflection. In two dimensions the representations of the dihedral-n group are the symmetries of a regular n-sided polygon.

3.2 Computational Group Theory

Computational group theory is the study of groups represented on computers. One of the major advantages of groups is that they can be represented compactly using their generators. However representing groups in this way requires algorithms to manipulate the group using only its generators. In this section we present algorithms and data-structures from computational group theory used to manipulate groups through their generators. We focus on how to search a group efficiently for a group element with a specific property.

Graph Theory

The algorithms used to efficiently search groups are based on graph searches. In this section we present basic results from graph theory. The graph theory definitions in this section are taken from [43, 10, 36].

An undirected graph $\Gamma = (\mathcal{V}, \mathcal{E})$ is a set of vertices $\mathcal{V} \subset \mathbb{N}$ together with a set of unordered pairs $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ called edges. If the edges are ordered then the graph is called directed. Vertices $i, j \in \mathcal{V}$ are called adjacent if $(i, j) \in \mathcal{E}$ or $(j, i) \in \mathcal{E}$ is an edge. The set of all vertices adjacent to $i \in \mathcal{V}$ will be denoted by $\mathcal{V}_i$. A path is a sequence of adjacent vertices. A cycle is a path that begins and ends at the same vertex. A graph is acyclic if it contains no cycles. A graph is connected if every pair of vertices is connected by a path.

A graph coloring $c : \mathcal{V} \to \mathcal{C}$ assigns a color $c(i) \in \mathcal{C}$ to each vertex $i \in \mathcal{V}$ of the graph. An edge labeling $l : \mathcal{E} \to \mathcal{L}$ assigns a set of labels $l(e) \subseteq \mathcal{L}$ to each edge $e = (i, j) \in \mathcal{E}$ of the graph. A graph symmetry $\pi : \mathcal{V} \to \mathcal{V}$ is a permutation of the vertices $\mathcal{V}$ that preserves the edge list $\pi(\mathcal{E}) = \mathcal{E}$ and vertex coloring $C(\pi(i)) = C(i)$ for all $v \in \mathcal{V}$.

A subgraph of a graph Γ is a graph whose vertex set is a subset of that of Γ and whose edge set is a subset of that of Γ. A subgraph is spanning if its vertex set is the same as Γ. A tree is a connected acyclic graph. A spanning tree of a graph Γ is a spanning subgraph that is a tree.

A contraction of a set of vertices $\mathcal{U} \subset \mathcal{V}$ merges all the vertices $\mathcal{U}$ into a single vertex u where u is adjacent to the all the vertices that were adjacent to some vertex in $\mathcal{U}$. Formally let f be the function that maps each vertex in $\mathcal{V} \setminus \mathcal{U}$ to itself and the vertices in $\mathcal{U}$ to u. Then the contraction of $\mathcal{U}$ is the graph $\Gamma = \big(f(\mathcal{V}), f(\mathcal{E})\big)$ with self-loops removed from $f(\mathcal{E})$.

A graph search is the problem of searching a graph for a particular node. Algorithm 3 is an example of a graph search algorithm. In the first part, Algorithm 3 uses a breadth-first search strategy to visit each node $\mathcal{V}$ of the graph while searching for a desired node $d \in \mathcal{V}$. As it searches the graph, Algorithm 3 constructs a search-tree data-structure `parent` that records which node $u = \texttt{parent}(v)$ precedes node v in the search-tree. In the second part, Algorithm 3 uses this search-tree to reconstruct a path from the root-node $r \in \mathcal{V}$ to the desired node $d \in \mathcal{V}$. Algorithm 3 has computational complexity $O(|\mathcal{E}|)$ where $|\mathcal{E}| \geq |\mathcal{V}|$ [34].

Algorithm 3 Breadth-first Search of graph $\Gamma = (\mathcal{V}, \mathcal{E})$ for desired node d

Input: Graph $\Gamma = (\mathcal{V}, \mathcal{E})$ and desired node $d \in \mathcal{V}$
Output: Path from root node r to desired node d

```
 1: // Search graph $\Gamma$ for desired node $d \in \mathcal{V}$
 2: enqueue root node $r \in \mathcal{V}$ in $\mathcal{Q}$
 3: set parent$(r) = \varnothing$
 4: while unexplored nodes in $\mathcal{Q}$ do
 5:     dequeue first unexplored element $v$ in $\mathcal{Q}$
 6:     if $v \in \mathcal{V}$ is the desired node $d \in \mathcal{V}$ then
 7:         return $v$
 8:     else
 9:         for each node $u \in \mathcal{V}_v$ adjacent to $v \in \mathcal{V}$ do
10:             if $u \notin \mathcal{Q}$ then
11:                 enqueue $u$ in $\mathcal{Q} = \mathcal{Q} \cup \{u\}$
12:                 set parent$(u) = v$
13:             end if
14:         end for
15:     end if
16:     mark node $v \in \mathcal{V}$ as explored
17: end while

18: // Reconstruct path from desired node $d \in \mathcal{V}$ to root node $r \in \mathcal{V}$
19: initialize path $\mathcal{P} = (d)$ and set $v = d$
20: while not at root node $v \neq r$ do
21:     update path $\mathcal{P} = \mathcal{P} + v$ and node $v =$ parent$(v)$
22: end while
```

Group Search Algorithms

In this section we discuss how to search a group $\mathcal{G} = \langle \mathcal{S} \rangle$ using only its generators $\mathcal{S}$. The algorithms for searching groups are based on searching specific graphs associated with the group. In this section we define three graphs associated with groups and discuss how to search these graphs. This section focuses on matrix groups although the algorithms can be used for permutation groups as well.

The first graph we introduce is the orbit graph. The orbit graph $\Gamma(\mathcal{S}, x) = (\mathcal{V}, \mathcal{E})$ is a graph that represents the orbit of a point $x \in \mathcal{X}$ under the group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S}$. The nodes $\mathcal{V} = \mathcal{G}x$ of the orbit graph $\Gamma(\mathcal{S}, x)$ are elements of the orbit $\mathcal{G}x = \langle \mathcal{S} \rangle x$. Two nodes $y, z \in \mathcal{G}x$ are connected by a directed edge (y, z) with label $S \in \mathcal{S}$ if $z = Sy$. A path $S_1, \dots, S_p$ from node $x \in \mathcal{X}$ to $y \in \mathcal{X}$ corresponds to a sequence of generators $S_1, \dots, S_p \in \mathcal{S}$ such that $y = S_1 \cdots S_p x = \Theta x$ where $\Theta = S_1 \cdots S_p$.

The orbit $\mathcal{G}x$ of a point $x \in \mathcal{X}$ under the group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S}$ can be searched using Algorithm 4. This algorithm is a modification of the breadth-first search in Algorithm 3. In the first part, Algorithm 4 searches the orbit $\mathcal{G}x$ of the point $x \in \mathcal{X}$ under the group $\mathcal{G} = \langle \mathcal{S} \rangle$ for the desired point $y \in \mathcal{X}$. The symbol $\mathcal{O}$ denotes the partially constructed orbit $\mathcal{G}x$. The algorithm initializes the partial constructed orbit as $\mathcal{O} = \{x\} \subseteq \mathcal{G}x$. During each while-loop, Algorithm 4 uses a point $z \in \mathcal{O} \subseteq \mathcal{G}x$ in the orbit $\mathcal{G}x$ and the generators $\mathcal{S}$ of the group $\mathcal{G} = \langle \mathcal{S} \rangle$ to find more points Sz in the orbit $\mathcal{G}x$. The algorithm uses the data-structure `generators` to keep track of its path through the orbit graph $\Gamma(\mathcal{S}, x)$. In computational group theory the data-structure `generators` is called a Schreier vector.

In the second part, Algorithm 4 constructs a sequence of generators $S_1, \dots, S_p$ such that $y = S_1 \cdots S_p x$. This is done by backtracking through the orbit graph $\Gamma(\mathcal{S}, x)$ of $\mathcal{G}x$ from the node $y \in \mathcal{G}x$ to the root-node $x \in \mathcal{G}x$ using the data-structure `generators`. In each while-loop the algorithm adds generator $S = \texttt{generator}(z) \in \mathcal{S}$ to the generator sequence $\mathcal{P}$ and moves to a node $S^{-1}z \in \mathcal{G}x$ closer to the root-node $x \in \mathcal{G}x$. The algorithm terminates when it reaches the root-node $x \in \mathcal{G}x$. This algorithm has worst-case complexity $O(n^2|\mathcal{S}||\mathcal{G}x|)$ for matrix groups and $O(|\mathcal{S}||\mathcal{G}x|)$ for permutation groups [74, 48].

Example 2. *Consider the symmetric group $\mathfrak{S}_n$. This group can be generated by $\mathcal{S} = \{S_1, \dots, S_{n-1}\}$ where S_i is the permutation matrix that transposes row i and row n for $i = 1, \dots, n-1$.*

Consider the points $x = e_1 \in \mathcal{X} = \mathbb{R}^n$ and $y = e_2 \in \mathcal{X} = \mathbb{R}^n$ where $e_1, \dots, e_n$ are the standard basis vectors for $\mathbb{R}^n$. We can use Algorithm 4 to find a sequence of generators that maps $x = e_1$ to $y = e_2$.

The algorithm initializes the orbit $\mathcal{O} = \{e_1\}$. During the first while-loop the algorithm selects $z = e_1$ and computes $S_i e_1$ for each generator $S_i \in \mathcal{S}$. For $S_1 \in \mathcal{S}$ this produces a new point $S_1 e_1 = e_n$ which is added to the orbit $\mathcal{O} = \{e_1, e_n\}$. The algorithm sets `generator`$(e_n) = S_1$. *Since this is the only new point created by the generators $S_i e_1$ the algorithm marks $z = e_1$ as explored.*

Algorithm 4 Search orbit $\mathcal{G}x$ of a point $x \in \mathcal{X}$ under the group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S}$ for the point $y \in \mathcal{X}$.

Input: Points $x, y \in \mathcal{X}$ and group generators $\mathcal{S}$

Output: If $y \in \mathcal{G}x$ returns sequence of generators $S_1, \ldots, S_p$ such that $y = S_1 \cdots S_p x$. Otherwise returns $y \notin \mathcal{G}x$.

```
1: // Search orbit $\mathcal{G}x$ for point $y \in \mathcal{X}$.
2: initialize $y \notin \mathcal{G}x$
3: enqueue root point $x \in \mathcal{X}$ in orbit $\mathcal{O} = \{x\}$
4: set parent$(x) = \varnothing$ and generator$(x) = \varnothing$
5: while unexplored points in orbit $\mathcal{O}$ do
6:     select an unexplored point $z \in \mathcal{O}$ in the orbit $\mathcal{O}$
7:     if desired point found $z = y$ then
8:         return $y \in \mathcal{G}x$
9:     else
10:        for each generator $S \in \mathcal{S}$ do
11:            if $Sz \notin \mathcal{O}$ is not in the orbit $\mathcal{O}$ then
12:                add $Sz$ to the orbit $\mathcal{O} = \mathcal{O} \cup \{Sz\}$
13:                set generator$(Sz) = S$
14:            end if
15:        end for
16:    end if
17:    mark point $z \in \mathcal{O}$ as explored
18: end while
19: // Reconstruct group element $\Theta \in \mathcal{G}$ such that $y = \Theta x$.
20: if point $y \in \mathcal{G}x$ is in the orbit $\mathcal{G}x$ then
21:    initialize generator sequence $\mathcal{P} = (I)$ and set $z = y$
22:    while $z$ not the root point $x \neq z$ do
23:        update generator sequence $\mathcal{P} = \mathcal{P} + S$ where $S =$ generator$(z)$
24:        update point $z = S^{-1}z$
25:    end while
26: end if
```

On the next while-loop the algorithm selects $z = e_n$ from $\mathcal{O}$. For the generator $S_2 \in \mathcal{S}$ this produces the point $S_2 e_n = e_2$. The algorithm sets `generator`$(e_2) = S_2$*. Since $e_2 = y$ is the desired point the algorithm moves on to the second part.*

In the second part, Algorithm 4 backtracks from node $y = e_2$ to the root-node $x = e_1$. The algorithm initializes the path $\mathcal{P} = (I)$ and sets $z = y = e_2$. In the first while-loop the algorithm adds $S_2 =$ `generator`$(z = e_2)$ *to the path $\mathcal{P} = (I, S_2)$ and updates the node $z = S_2^{-1} e_2 = e_n$. On the next loop the algorithm adds $S_1 =$* `generator`$(z = e_n)$ *to the path $\mathcal{P} = (I, S_2, S_1)$ and updates the node $z = S_1^{-1} e_n = e_1$. Since $e_1 = x$ is the root-node the algorithm terminates. The sequence $\mathcal{P} = (I, S_2, S_1)$ can be used to produce a matrix $\Theta = I S_2 S_1$ that maps the point $x = e_1$ to $y = e_2 = \Theta x$.*

If the set $\mathcal{X}$ is finite then we can extend the Schreier vector `generator` to the entire set $\mathcal{X}$ by defining

$$\texttt{generators}(y) = \begin{cases} I & y = x \\ S^{-1} & y \in \mathcal{G}x \setminus \{x\} \\ \varnothing & y \notin \mathcal{G}x \end{cases} . \tag{3.8}$$

As before, for a point $y \in \mathcal{G}x$ in the orbit $\mathcal{G}x$, the Schreier vector `generators` returns the generator $S \in \mathcal{S}$ corresponding to the edge entering node $y \in \mathcal{X}$ in the search-tree of the orbit graph $\Gamma(\mathcal{S}, x)$. If the point $y \in \mathcal{X}$ is not an element of the orbit $\mathcal{G}x$ then the Schreier vector returns the null element `generators`$(y) = \varnothing$ indicating that it is not possible to map x to y. For the root-node $y = x$ the Schreier vector returns the identity element I which is the only element of $\mathcal{G}/\mathcal{G}_x$ that maps x to itself. Using this Schreier vector `generators` we can reconstruct the orbit $\mathcal{G}x$ by reading the non-null elements of `generators`. Algorithm 5 can be used to find a group element $U \in \mathcal{G}$ that maps $x \in \mathcal{G}x$ to $y \in \mathcal{G}x$. Algorithm 5 has complexity $O(n^2|\mathcal{G}x|)$ for matrix groups and $O(|\mathcal{G}x|)$ for permutation groups [74, 48].

Algorithm 5 Use Schreier vector `generators` to find $U \in \mathcal{G}$ such that $y = Ux$.

```
Input: Point $y \in \mathcal{G}x$ and Schreier vector generators for orbit $\mathcal{G}x$.
Output: Group element $U \in \mathcal{G}$ such that $y = Ux$.
1: initialize group element $U = I$ and set $z = y$
2: while not at root-node generators($z$) $\neq I$ do
3:     read generator $S =$ generators($z$)
4:     update group element $U = SU$ and point $z = Sz$
5: end while
```

Often we are only interested in calculating the orbit $\mathcal{G}x$ of a point $x \in \mathcal{X}$. In this case we can use Algorithm 6 which is a simplified version of Algorithm 4.

The next graph we introduce is the Cayley graph. The Cayley graph $\Gamma(\mathcal{S}) = (\mathcal{V}, \mathcal{E})$ is a graph that describes how the group $\mathcal{G} = \langle \mathcal{S} \rangle$ is constructed from its generators $\mathcal{S}$. The vertices $\mathcal{V} = \mathcal{G}$ of the Cayley graph $\Gamma(\mathcal{S})$ are the elements $\Theta \in \mathcal{G}$ of the group $\mathcal{G}$. Two group

Algorithm 6 Orbit Construction

```
Input: Point $x \in \mathcal{X}$ and generators $\mathcal{S}$ of the group $\mathcal{G} = \langle \mathcal{S} \rangle$
Output: Orbit $\mathcal{O} = \mathcal{G}x = \langle \mathcal{S} \rangle x$
1: Initialize orbit $\mathcal{O} = \{x\}$
2: while Orbit $\mathcal{O}$ still growing do
3:     for each $g \in \mathcal{S}$ do
4:         Expand orbit $\mathcal{O} = \mathcal{O} \cup g\mathcal{O}$
5:     end for
6: end while
```

elements $\Theta_g, \Theta_h \in \mathcal{G}$ are connected by a directed edge $(\Theta_g, \Theta_h) \in \mathcal{E}$ with label $S \in \mathcal{S}$ if $\Theta_h = S\Theta_g$.

Since the group $\mathcal{G}$ acts on itself $\mathcal{X} = \mathcal{G}$, the Cayley graph is a special type of orbit graph. For any group element $\Theta \in \mathcal{G}$ we can find a sequence of generators $S_1, \ldots, S_p \in \mathcal{S}$ that generate $\Theta = S_1 \cdots S_p$ by using Algorithm 4 to find a path through the Cayley graph from the identity element $x = I \in \mathbb{R}^{n \times n}$ to the desired element $y = \Theta \in \mathcal{G}$. The generators $S_1, \ldots, S_p \in \mathcal{S}$ are the labels of the edges of the path from I to Θ.

Example 3. *Consider the symmetric group $\mathfrak{S}_n$ with the generators $\mathcal{S}$ defined in Example 2. Suppose we want to find a sequence of generators for the permutation matrix $\Pi \in \mathfrak{S}_n$. This can be done using Algorithm 4 with $x = I$ and $y = \Pi$. However this will have worst-case complexity $O(n^2|\mathcal{S}||\mathfrak{S}_n I|) = O(n^3 n!)$ since $|\mathcal{S}| = n - 1$ and $|\mathfrak{S}_n I| = |\mathfrak{S}_n| = n!$.*

Next we introduce the Schreier graph. Recall that if $\mathcal{H} \subseteq \mathcal{G}$ is a subgroup of $\mathcal{G}$ then its cosets $U\mathcal{H}$ decompose the group $\mathcal{G}$ into disjoint subsets. If the generators $\mathcal{S}$ have the property that $\mathcal{H} = \langle \mathcal{S} \cap \mathcal{H} \rangle$ then the Cayley graph $\Gamma(\mathcal{S})$ of $\mathcal{G} = \langle \mathcal{S} \rangle$ will contain $|\mathcal{G}|/|\mathcal{H}|$ copies of the Cayley graph $\Gamma(\mathcal{S} \cap \mathcal{H})$ of the subgroup $\mathcal{H} = \langle \mathcal{S} \cap \mathcal{H} \rangle$. Thus the Cayley graph $\Gamma(\mathcal{S})$ of $\mathcal{G} = \langle \mathcal{S} \rangle$ has the same repetitive structure as the group $\mathcal{G}$ it represents.

If we contract each copy of the graph $\Gamma(\mathcal{S} \cap \mathcal{H})$ into a single node, then the resulting graph is called the Schreier coset graph $\Gamma_{\mathcal{H}}(\mathcal{S})$. This graph describes how the copies of the graph $\Gamma(\mathcal{S} \cap \mathcal{H})$ are connected together to form the original Cayley graph $\Gamma(\mathcal{S})$. We can search the Cayley graph $\Gamma(\mathcal{S})$ of $\mathcal{G} = \langle \mathcal{S} \rangle$ by searching the Schreier graph $\Gamma_{\mathcal{H}}(\mathcal{S})$ for the copy of the graph $\Gamma(\mathcal{S} \cap \mathcal{H})$ that contains the node we are interested in and then searching that copy $\Gamma(\mathcal{S} \cap \mathcal{H})$. This requires visiting $|\mathcal{G}|/|\mathcal{H}| + |\mathcal{H}|$ graph nodes instead of the full $|\mathcal{G}| = |\mathcal{G}|/|\mathcal{H}| \times |\mathcal{H}|$ vertices required to search the Cayley graph $\Gamma(\mathcal{S})$.

If the subgroup $\mathcal{H} = \mathcal{G}_x$ is the stabilizer subgroup $\mathcal{G}_x$ of a point $x \in \mathcal{X}$ then the Schreier graph $\Gamma_{\mathcal{H}}(\mathcal{S})$ of the cosets $U\mathcal{G}_x$ of $\mathcal{G}_x$ in $\mathcal{G}$ is also the orbit graph $\Gamma(\mathcal{S}, x)$ of the point $x \in \mathcal{X}$. This means we can use Algorithm 4 to search the Schreier graph $\Gamma_{\mathcal{H}}(\mathcal{S})$. This decomposition is exploited in bases and strong generating sets defined in the next section.

Bases and Strong Generating Sets

In this section we introduce bases and strong generating sets which are an important data-structure in computational group theory. All groups in this section are finite $|\mathcal{G}| < \infty$.

To motivate bases and strong generating sets we consider the following problem: Given a set of generators $\mathcal{S}$ for the group $\mathcal{G} = \langle \mathcal{S} \rangle$ how do we determine if the matrix $\Theta_0 \in \mathbb{R}^{n \times n}$ is an element $\Theta_0 \in \mathcal{G}$ of the group $\mathcal{G} = \langle \mathcal{S} \rangle$? One method is to use Algorithm 4 to search the Cayley graph $\Gamma(\mathcal{S})$ for the matrix Θ_0. However this will typically have exponential complexity.

Instead we can use the following methods to determine whether Θ_0 is *not* an element of the group $\mathcal{G} = \langle \mathcal{S} \rangle$. Consider a point $x_1 \in \mathcal{X} = \mathbb{R}^n$. If Θ_0 is an element of the group $\mathcal{G} = \langle \mathcal{S} \rangle$ then the point $\Theta_0 x_1$ should be in the orbit $\mathcal{G}x_1$ of the point $x_1 \in \mathcal{X}$ under the group $\mathcal{G} = \langle \mathcal{S} \rangle$. If $\Theta_0 x_1 \notin \mathcal{G}x_1$ is not in the orbit $\mathcal{G}x_1$ then Θ is not an element of the group $\mathcal{G} = \langle \mathcal{S} \rangle$.

What if $\Theta_0 x_1 \in \mathcal{G}x_1$ is an element of the orbit $\mathcal{G}x_1$? Then by Proposition 2 we known that every element of the group $\mathcal{G}$ is the product $\Theta_0 = U_1 \Theta_1$ of a matrix $\Theta_1 \in \mathcal{G}_1 = \mathcal{G}_{x_1}$ that fixes the point x_1 and a matrix $U_1 \in \mathcal{G}/\mathcal{G}_1$ such that $U_1 x_1 = \Theta_0 x_1$. Using Algorithm 4 we have already found a matrix $U_1 \in \mathcal{G}/\mathcal{G}_1$ such that $U_1 x_1 = \Theta_0 x_1$. All that remains is to search the group $\mathcal{G}_1 = \{\Theta \in \mathcal{G} : \Theta x_1 = x_1\}$ for the matrix $\Theta_1 = U_1^{-1}\Theta_0$. This can be accomplished by selecting a point $x_2 \in \mathcal{X} = \mathbb{R}^n$ and testing whether $\Theta_1 x_2 \in \mathcal{G}_1 x_2$ is an element of the orbit $\mathcal{G}_1 x_2$ of the point x_2 under the group $\mathcal{G}_1$ that fixes x_1. We can continue this process until the group

$$\mathcal{G}_b = \big\{\Theta \in \mathcal{G} : \Theta x_i = x_i, i = 1, \dots, b\big\} = \big\{I\big\} \tag{3.9}$$

contains only the identity matrix $I \in \mathbb{R}^{n \times n}$. In this case we have produced a sequence $U_1, \dots, U_b$ of matrices such that $\Theta_0 = U_1 \cdots U_b$.

The set of points $x_1, \dots, x_b$ that we have defined is called a base. The formal definition is given below.

Definition 3. A base $\mathcal{B} = \{x_1, \dots, x_b\}$ is an ordered set of points whose point-wise stabilizer group $\mathcal{G}_\mathcal{B}$ is trivial

$$\mathcal{G}_\mathcal{B} = \big\{\Theta \in \mathcal{G} : \Theta x = x, \forall x \in \mathcal{B}\big\} = \langle \varnothing \rangle. \tag{3.10}$$

Definition 3 says that the every element of the group, other than the identity matrix I, transforms at least one point $x \in \mathcal{B}$ in the base $\mathcal{B}$. We can uniquely identify the elements $\Theta \in \mathcal{G}$ of the group $\mathcal{G}$ by determining how they transform the base points $\mathcal{B}$.

The base $\mathcal{B}$ produces a chain of stabilizer subgroups

$$\mathcal{G} = \mathcal{G}_0 \supseteq \mathcal{G}_1 \supseteq \cdots \supseteq \mathcal{G}_{\mathcal{B}-1} \supseteq \mathcal{G}_\mathcal{B} = \langle \varnothing \rangle \tag{3.11}$$

where

$$\mathcal{G}_i = \big\{\Theta \in \mathcal{G} : \Theta x_j = x_j, j = 1, \dots, i\big\} = \big\{\Theta \in \mathcal{G}_{i-1} : \Theta x_i = x_i\big\} \tag{3.12}$$

is the subgroup that fixes bases points $x_1, \ldots, x_i$ and $\mathcal{G}_0 = \mathcal{G}$ is the original group. The base $\mathcal{B}$ is called non-redundant if $\mathcal{G}_{i+1} \subset \mathcal{G}_i$ is a strict subset of the group $\mathcal{G}_i$. In other words the base $\mathcal{B}$ is non-redundant if each base point x_{i+1} strictly decreases the size of the group $|\mathcal{G}_{i+1}| < |\mathcal{G}_i|$.

In order to use the base $\mathcal{B}$ to test group membership $\Theta_0 \in \mathcal{G}_0 = \mathcal{G}$ we need to be able to search the orbit $\mathcal{G}_{i-1}x_i$ of the base point x_i under the subgroup $\mathcal{G}_{i-1} = \{\Theta \in \mathcal{G} : \Theta x_j = x_j, j = 1, \ldots, i-1\}$ that fixes the first $i-1$ base points $x_1, \ldots, x_{i-1}$. We can search these orbits using Algorithm 4 provided we have a set of generators $\mathcal{S}_i$ for the group $\mathcal{G}_i = \langle \mathcal{S}_i \rangle$. One might expect that we can find generators of $\mathcal{G}_i$ by selecting the subset $\mathcal{S}_i$ of generators $\mathcal{S}$ for $\mathcal{G} = \langle \mathcal{S} \rangle$ that fix $x_1, \ldots, x_i$

$$\mathcal{S}_i = \{S \in \mathcal{S} : Sx_j = x_j, j = 1, \ldots, i\}. \tag{3.13}$$

However in general $\mathcal{S}_i$ does not generate $\mathcal{G}_i \supseteq \langle \mathcal{S}_i \rangle$ as shown in the following example.

Example 4. *The dihedral-4 group in $\mathbb{R}^2$ can be generated by $\mathcal{S} = \{S_1, S_2\}$ where*

$$S_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{ and } S_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{3.14}$$

Consider the point $x_1 = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$. The stabilizer subgroup $\mathcal{G}_1$ of this point is the set

$$\mathcal{G}_1 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \right\}. \tag{3.15}$$

However the subset $\mathcal{S}_1 = \{S \in \mathcal{S} : Sx_1 = x_1\}$ of generators $\mathcal{S}$ that fix x_1 is empty $\mathcal{S}_1 = \varnothing$. Therefore $\mathcal{G}_1 \neq \langle \mathcal{S}_1 \rangle$.

A set of generators $\mathcal{S}$ that has the property $\langle \mathcal{S}_i \rangle = \mathcal{G}_i$ for $i = 1, \ldots, \mathcal{B}$ where $\mathcal{S}_i = \{S \in \mathcal{S} : Sx_j = x_j, j = 1, \ldots, i\}$ and $\mathcal{G}_i = \{\Theta \in \mathcal{G} : \Theta x_j = x_j, j = 1, \ldots, i\}$ is called a strong generating set of the base $\mathcal{B}$. A formal definition is given below.

Definition 4. The generating set $\mathcal{S}$ for the group $\mathcal{G}$ is called strong with respect to the base $\mathcal{B}$ if $\mathcal{S}_i = \mathcal{S} \cap \mathcal{G}_i$ is a generating set for $\mathcal{G}_i = \langle \mathcal{S}_i \rangle$.

The pair $(\mathcal{B}, \mathcal{S})$ is called a base and strong generating set. A base and strong generating set $(\mathcal{B}, \mathcal{S})$ can be calculated using the Schreier-Sims algorithm [48, 74].

Example 5. *Consider the symmetric group $\mathfrak{S}_n$. A base for this group is the set $\mathcal{B} = \{e_1, \ldots, e_{n-1}\}$. A strong generating set $\mathcal{S} = \{S_1, \ldots, S_{n-1}\}$ for this base is the set of generators $\mathcal{S} = \{S_1, \ldots, S_{n-1}\}$ where S_i is the permutation matrix that transposes row i and row n for $i = 1, \ldots, n-1$.*

Suppose we want to find a sequence of generators for the permutation matrix $\Pi \in \mathfrak{S}_n$. This can be done by first finding a matrix $U_1 \in \mathfrak{S}_n$ such that $U_1 e_1 = \Pi e_1$. This can be done *using Algorithm 4 for $x = e_1$ and $y = \Pi e_1$ with generators $\mathcal{S}$. This search has complexity $O(n^2|\mathcal{S}||\mathfrak{S}_n e_1|) = O(n^4)$.*

Next we need to find a matrix $U_2 \in \mathfrak{S}_{n-1}$ such that $U_2 e_2 = U_1^{-1}\Pi e_2$ where $\mathfrak{S}_{n-1}$ is the subgroup of $\mathfrak{S}_n$ that fixes $x_1 = e_1$. Since $\mathcal{S}$ is a strong generating set, the subgroup $\mathfrak{S}_{n-1}$ is generated by the set $\mathcal{S}_1 = \{S \in \mathcal{S} : Sx_1 = x_1\}$. In this case we discard S_1 from the generators $\mathcal{S} = \{S_1, \ldots, S_{n-1}\}$ to produce $\mathcal{S}_1 = \{S_2, \ldots, S_{n-1}\}$. Next we call Algorithm 4 with points $x = e_2$ and $y = \Pi e_2$, and generating set $\mathcal{S}_1 = \{S_2, \ldots, S_{n-1}\}$. This returns a matrix $U_2 \in \mathfrak{S}_{n-1}$ such that $U_2 e_2 = U_1^{-1}\Pi e_2$. This step had complexity $O(n^4)$.

Continuing this process we can find a set of generators for the matrix $\Pi \in \mathfrak{S}_n$ with worse-case complexity $O(n^5) \ll O(n^3 n!)$. By exploiting that the elements of $\mathfrak{S}_n$ and its subgroups are permutation matrices this complexity can be reduced to $O(n^3)$.

Part II

Symmetry in Constrained Linear Control

Chapter 4

Definitions of Symmetry

In this chapter we define symmetry for dynamic systems and controllers. In particular we define symmetry for constrained linear systems and piecewise affine on polytope controllers. We prove basic results about reachability for symmetric constrained systems.

This chapter is primarily focused on symmetry in model predictive control. We show that model predictive control problems for symmetric constrained systems with symmetric cost functions produce symmetric controllers. In particular for linear model predictive control problems with quadratic or linear costs the symmetries of the model predictive control problem permute the pieces of a symmetric controller.

4.1 Symmetries of Dynamic Systems

Consider the discrete-time system

$$x(t+1) = f(x(t), u(t)) \tag{4.1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. A symmetry of the system (4.1) is a state-space $\Theta \in \mathbb{R}^{n \times n}$ and input-space $\Omega \in \mathbb{R}^{m \times m}$ transformation that preserves the dynamics.

Definition 5. The pair of invertible matrices (Θ, Ω) is a symmetry of the system (4.1) if they satisfy

$$f(\Theta x, \Omega u) = \Theta f(x, u) \tag{4.2}$$

for all $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$.

For an autonomous system, Definition 5 says that the vector field $f(x) = f(x, 0)$ is invariant under the state-space transformation Θ i.e. $f(x) = \Theta^{-1} f(\Theta x)$. For a non-autonomous system this definition says that if the state and input trajectories $\{x(t)\}_{t=0}^{\infty}$ and $\{u(t)\}_{t=0}^{\infty}$ satisfy the dynamics (4.1) then so does the state and input trajectories $\{\Theta x(t)\}_{t=0}^{\infty}$ and $\{\Omega u(t)\}_{t=0}^{\infty}$.

The set of all invertible matrices (Θ, Ω) that satisfy (4.2) is a group denoted by $\mathrm{Aut}(f)$.

Proposition 3. *The set* $\mathrm{Aut}(f)$ *of all invertible matrices* (Θ, Ω) *that satisfy* (4.2) *is a group under pairwise matrix multiplication.*

Proof. We will show that $\mathrm{Aut}(f)$ with pairwise matrix multiplication satisfies the axioms of group theory. Note that matrix multiplication is associative.

1. Identity: The identity $(I_n, I_m) \in \mathrm{Aut}(f)$ is an element of the set $\mathrm{Aut}(f)$ since
$$f(I_n x, I_m u) = I_n f(x, u). \tag{4.3}$$

2. Inverse: By the definition of $\mathrm{Aut}(f)$ for each symmetry $(\Theta, \Omega) \in \mathrm{Aut}(f)$ the inverse $(\Theta^{-1}, \Omega^{-1})$ exists. By (4.2) the inverse $(\Theta^{-1}, \Omega^{-1}) \in \mathrm{Aut}(f)$ is an element of the set $\mathrm{Aut}(f)$ since
$$\Theta^{-1} f(y, v) = f(\Theta^{-1} y, \Omega^{-1} v) \tag{4.4}$$
where $y = \Theta x$ and $v = \Theta u$.

3. Closure: Let $(\Theta_1, \Omega_1), (\Theta_2, \Omega_2) \in \mathrm{Aut}(f)$ then $(\Theta_1\Theta_2, \Omega_1\Omega_2) \in \mathrm{Aut}(f)$ since
$$f(\Theta_1\Theta_2 x, \Omega_1\Omega_2 u) = \Theta_1 f(\Theta_2 x, \Omega_2 u) = \Theta_1\Theta_2 f(x, u) \tag{4.5}$$

□

The following example demonstrates this concept of symmetry.

Example 6. *This example is taken from [22]. Consider the autonomous nonlinear system*
$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} -x_1(t)x_2(t)^2 + 2x_2(t)^3 \\ -x_2(t)^3 \end{bmatrix} \Delta\tau \tag{4.6}$$
where $\Delta\tau \ll 1$ *is the sample-time. The vector-field is shown in Figure 4.1a. In [22] it was shown that the symmetry group* $\mathrm{Aut}(f)$ *of this system is the infinite set*
$$\mathrm{Aut}(f) = \left\{ \begin{bmatrix} 1 & \theta \\ 0 & 1 \end{bmatrix} : \theta \in \mathbb{R} \right\}. \tag{4.7}$$
Any $\Theta \in \mathrm{Aut}(f)$ *will map the vector-field* $f(x)$ *to itself* $f(x) = \Theta^{-1} f(\Theta x)$*. Consider the symmetry*
$$\Theta = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \in \mathrm{Aut}(f). \tag{4.8}$$
The image $\Theta^{-1} f(\Theta x)$ *of the vector-field is shown in Figure 4.1b. It can be readily seen that* Θ *maps the vector-field to itself.*

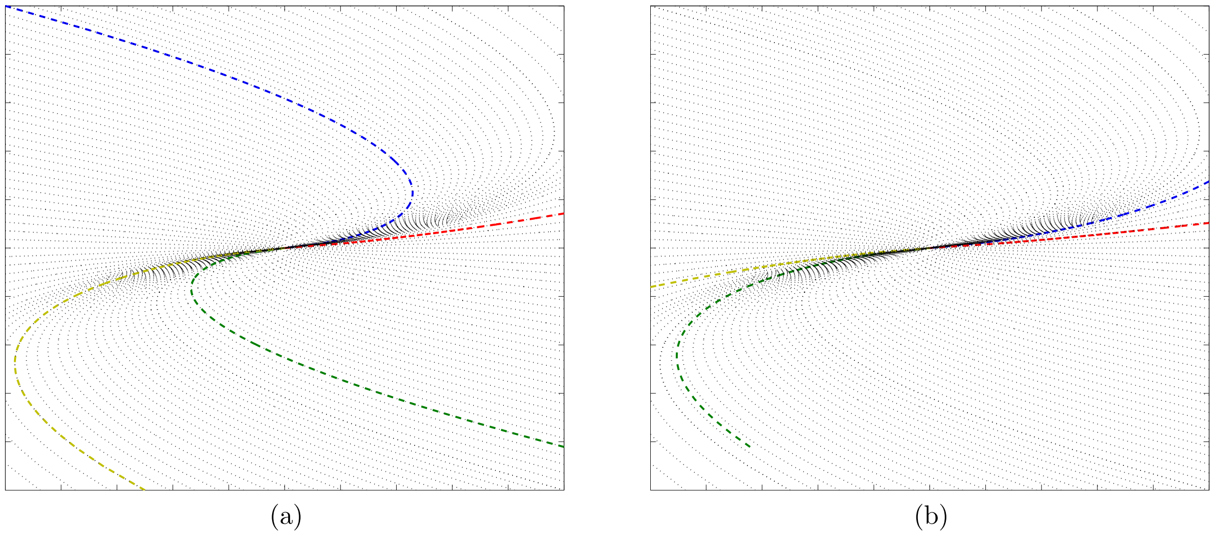


Figure 4.1: Example of symmetric non-linear system. (a) Vector-field $f(x)$. (b) Vector-field $\Theta^{-1}f(\Theta x)$. Four state-trajectories have been color-coded to illustrate how Θ transforms the vector field.

Symmetries of Linear Systems

We are particularly interested in studying the symmetry groups for linear systems

$$x(t+1) = Ax(t) + Bu(t) \tag{4.9}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$.

For the linear system (4.9) we will often denote its symmetry group by $\text{Aut}(A,B) = \text{Aut}(f)$ where $f(x,u) = Ax + Bu$. This group $\text{Aut}(A,B)$ is always infinite. For $\theta \neq 0 \in \mathbb{R}$ the pair $(\Theta,\Omega) = (\theta I_n, \theta I_m) \in \text{Aut}(A,B)$ is a symmetry. In fact the closure of the group $\text{Aut}(A,B)$ is a vector-space.

Proposition 4. *The closure of the group* $\text{Aut}(A,B)$ *is a finite-dimensional vector-space.*

Proof. First we show that the set $\text{Vec}(A,B)$ of all matrices (not necessarily invertible) that satisfy (4.2) is a vector-space. Associativity, commutativity, and scalar multiplication follow from the properties of matrix algebra. The set $\text{Vec}(A,B)$ contains the additive identity $(0_n, 0_m) \in \text{Vec}(A,B)$ and the additive inverse $(-\Theta,-\Omega) \in \text{Vec}(A,B)$ for all $(\Theta,\Omega) \in \text{Vec}(A,B)$. Distributivity follows from (4.2) and the linearity of the vector-field $f(x,u) = Ax + Bu$

$$\begin{aligned} f\big((\alpha\Theta_1 + \beta\Theta_2)x, (\alpha\Omega_1 + \beta\Omega_2)u\big) &= \alpha f(\Theta_1 x, \Omega_1 u) + \beta f(\Theta_2 x, \Omega_2 u) \\ &= (\alpha\Theta_1 + \beta\Theta_2) f(x,u). \end{aligned} \tag{4.10}$$

Thus $(\alpha\Theta_1 + \beta\Theta_2, \alpha\Omega_1 + \beta\Omega_2) \in \mathrm{Vec}(A,B)$ for any $\alpha, \beta \in \mathbb{R}$ and $(\Theta_1,\Omega_1),(\Theta_2,\Omega_2) \in \mathrm{Vec}(A,B)$. Thus $\mathrm{Vec}(A,B)$ is a vector-space. In particular it is a finite-dimensional vector-space $\dim(\mathrm{Vec}(A,B)) \leq n^2m^2$ and therefore complete [64].

Next we show $\mathrm{Vec}(A,B)$ is the closure of $\mathrm{Aut}(A,B)$. Since $\mathrm{Aut}(A,B) \subset \mathrm{Vec}(A,B)$ and $\mathrm{Vec}(A,B)$ is complete, any limit point of $\mathrm{Aut}(A,B)$ is contained in $\mathrm{Vec}(A,B)$. Conversely every point in $\mathrm{Vec}(A,B)$ is a limit point of $\mathrm{Aut}(A,B)$. For any $(\Theta,\Omega) \in \mathrm{Vec}(A,B)$ we can construct a sequence $(\alpha I_n + \Theta, \alpha I_m + \Omega) \in \mathrm{Aut}(A,B)$ of invertible matrices that limit to (Θ,Ω) as $\alpha \to 0$. □

This proposition means that we can use standard linear algebra techniques when working with the group $\mathrm{Aut}(A,B)$. This makes $\mathrm{Aut}(A,B)$ easier to work with than the other symmetry groups we will introduce in this chapter.

4.2 Symmetries of Constrained Systems

We now consider the system (4.1) subject to constraints on the state and input

$$x(t) \in \mathcal{X} \tag{4.11a}$$
$$u(t) \in \mathcal{U} \tag{4.11b}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are full-dimensional.

A symmetry of the constraints (4.11) is a state-space Θ and input-space Ω transformation that maps the constraint sets to themselves $\Theta\mathcal{X} = \mathcal{X}$ and $\Omega\mathcal{U} = \mathcal{U}$.

Definition 6. The pair of matrices (Θ,Ω) is a symmetry of the constraints (4.11) if they satisfy

$$\Theta\mathcal{X} = \mathcal{X} \tag{4.12a}$$
$$\Omega\mathcal{U} = \mathcal{U}. \tag{4.12b}$$

The sets $\mathrm{Aut}(\mathcal{X})$ and $\mathrm{Aut}(\mathcal{U})$ of all matrices Θ and Ω that satisfy (4.12a) and (4.12b) respectively are groups. Since we can choose Θ and Ω independently, the set $\mathrm{Aut}(\mathcal{X},\mathcal{U}) = \mathrm{Aut}(\mathcal{X}) \times \mathrm{Aut}(\mathcal{U})$ of all matrix pairs (Θ,Ω) that satisfy (4.12) is the cartesian product of the groups $\mathrm{Aut}(\mathcal{X})$ and $\mathrm{Aut}(\mathcal{U})$.

Proposition 5. *The set* $\mathrm{Aut}(\mathcal{X})$ *of all matrices* Θ *that satisfy* (4.12a) *and the set* $\mathrm{Aut}(\mathcal{U})$ *of all matrices* Ω *that satisfy* (4.12b) *are groups under pairwise matrix multiplication.*

Proof. The proof follows the lines of the proof of Proposition 3. In particular the matrices $\Theta \in \mathrm{Aut}(\mathcal{X})$ and $\Omega \in \mathrm{Aut}(\mathcal{U})$ are invertible since otherwise $\Theta\mathcal{X}$ and $\Omega\mathcal{U}$ would be lower dimensional and thus $\Theta\mathcal{X} \neq \mathcal{X}$ and $\Omega\mathcal{U} \neq \mathcal{U}$. □

The symmetry group of the system (4.1) subject to the constraints (4.11) is a group denoted by $\text{Aut}(\Sigma) = \text{Aut}(f) \cap \text{Aut}(\mathcal{X}, \mathcal{U})$. Next we use this group to prove some basic results about reachability for symmetric constrained systems. The following proposition states that the maximal control invariant set of the symmetric constrained system is symmetric.

Proposition 6. *Let $\mathcal{C}_\infty$ be the maximal control invariant set for the system* (4.1) *subject to the constraints* (4.11) *is symmetry with respect to the group* $\text{Aut}(\Sigma)$

$$\mathcal{C}_\infty = \Theta \mathcal{C}_\infty \tag{4.13}$$

for all $\Theta \in \text{Aut}(\Sigma) = \text{Aut}(f) \cap \text{Aut}(\mathcal{X}, \mathcal{U})$

Proof. Consider the set

$$\mathcal{C} = \bigcup\nolimits_{\Theta \in \mathcal{G}} \Theta \mathcal{C}_\infty \tag{4.14}$$

where $\mathcal{G} = \text{Aut}(\Sigma)$. This set is symmetric by construction $\mathcal{C} = \Theta \mathcal{C}$ for all $\Theta \in \text{Aut}(\Sigma)$. We will show $\mathcal{C} = \mathcal{C}_\infty$.

First we will prove $\mathcal{C}$ is a control invariant set. Let $x \in \mathcal{C}$ then there exists $\Theta \in \text{Aut}(\Sigma)$ such that $\Theta^{-1} x \in \mathcal{C}_\infty$. By the definition of control invariance there exists a feasible control input $v \in \mathcal{U}$ such that

$$f(\Theta^{-1} x, v) \in \mathcal{C}_\infty. \tag{4.15}$$

Using the properties of the symmetry group $\text{Aut}(\Sigma) = \text{Aut}(f) \cap \text{Aut}(\mathcal{X}, \mathcal{U})$ we have

$$f(x, \Omega v) \in \Theta \mathcal{C}_\infty \subseteq \mathcal{C}. \tag{4.16}$$

Thus for any $x \in \mathcal{C}$ there exists $u = \Omega v \in \Omega \mathcal{U} = \mathcal{U}$ such that $f(x, u) \in \mathcal{C}$. Therefore $\mathcal{C}$ is a control invariant set.

Since $\mathcal{C}_\infty$ is the maximal control invariant set we have $\mathcal{C} \subseteq \mathcal{C}_\infty$. However by construction $\mathcal{C}_\infty \subseteq \mathcal{C}$. Therefore $\mathcal{C} = \mathcal{C}_\infty$ is symmetric with respect to the group $\text{Aut}(\Sigma)$. □

The following proposition shows that the set of states x that are driven into a symmetric set $\mathcal{S}$ is symmetric. Likewise the set of states x that can be reached from the symmetric set $\mathcal{S}$ is symmetric.

Proposition 7. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a symmetric set $\Theta \mathcal{S} = \mathcal{S}$ for all $\Theta \in \text{Aut}(\Sigma)$. Then the backward* $\text{Pre}(\mathcal{S})$ *and forward* $\text{Reach}(\mathcal{S})$ *reachable sets for the system* (4.1) *subject to the constraints* (4.11) *are symmetric*

$$\Theta\, \text{Pre}(\mathcal{S}) = \text{Pre}(\mathcal{S}) \tag{4.17a}$$
$$\Theta\, \text{Reach}(\mathcal{S}) = \text{Reach}(\mathcal{S}) \tag{4.17b}$$

for all $\Theta \in \text{Aut}(\Sigma)$.

Proof. Let $y \in \operatorname{Reach}(\mathcal{S})$ be forward reachable. Then there exists $x \in \mathcal{S}$ and $u \in \mathcal{U}$ such that $y = f(x,u)$. Thus $\Theta y \in \operatorname{Reach}(\mathcal{S})$ since

$$\Theta y = \Theta f(x,u) = f(\Theta x, \Omega u) \tag{4.18}$$

where $\Theta x \in \Theta\mathcal{S} = \mathcal{S}$ and $\Omega u \in \Omega\mathcal{U} = \mathcal{U}$. Therefore $\Theta \operatorname{Reach}(\mathcal{S}) \subseteq \operatorname{Reach}(\mathcal{S})$ for all $\Theta \in \operatorname{Aut}(\Sigma)$. Since $\operatorname{Aut}(\Sigma)$ is a group, for any $\Theta \in \operatorname{Aut}(\Sigma)$ we have $\Theta^{-1} \in \operatorname{Aut}(\Sigma)$. Therefore $\Theta \operatorname{Reach}(\mathcal{S}) = \operatorname{Reach}(\mathcal{S})$.

Let $x \in \operatorname{Pre}(\mathcal{S})$ be backward reachable. Then there exists $u \in \mathcal{U}$ such that $f(x,u) \in \mathcal{S}$. Thus $\Theta x \in \operatorname{Pre}(\mathcal{S})$ since

$$f(\Theta x, \Omega u) = \Theta f(x,u) \in \Theta\mathcal{S} = \mathcal{S} \tag{4.19}$$

where $\Omega u \in \Omega\mathcal{U} = \mathcal{U}$. Therefore $\Theta \operatorname{Pre}(\mathcal{S}) \subseteq \operatorname{Pre}(\mathcal{S})$ for all $\Theta \in \operatorname{Aut}(\Sigma)$. Since $\operatorname{Aut}(\Sigma)$ is a group, for any $\Theta \in \operatorname{Aut}(\Sigma)$ we have $\Theta^{-1} \in \operatorname{Aut}(\Sigma)$ therefore $\Theta \operatorname{Pre}(\mathcal{S}) = \operatorname{Pre}(\mathcal{S})$. □

Symmetries of Constrained Linear Systems

We are particularly interested in studying the symmetry of linear systems (4.9) subject to polytopic constraints. For bounded, full-dimensional polytopic sets $\mathcal{X}$ and $\mathcal{U}$ the symmetry group $\operatorname{Aut}(\mathcal{X},\mathcal{U})$ is finite.

Proposition 8. *Let $\mathcal{X}$ and $\mathcal{U}$ be bounded, full-dimensional polytopes. Then the set $\operatorname{Aut}(\mathcal{X},\mathcal{U})$ is a finite group.*

Proof. Each matrix $\Theta \in \operatorname{Aut}(\mathcal{X})$ permutes the vertices of $\mathcal{X}$ since $\Theta\mathcal{X} = \mathcal{X}$. Since the set $\mathcal{X}$ is bounded and full-dimensional its vertices contain an affinely independent subset. Thus Θ is uniquely defined by how it permutes the vertices. Since there are a finite number of vertices, there are a finite number of permutations of these vertices. Thus $\operatorname{Aut}(\mathcal{X})$ is finite. Likewise $\operatorname{Aut}(\mathcal{U})$ is finite. Therefore $\operatorname{Aut}(\mathcal{X},\mathcal{U}) = \operatorname{Aut}(\mathcal{X}) \times \operatorname{Aut}(\mathcal{U})$ is finite. □

The following examples show that if the polytope $\mathcal{X}$ is unbounded or lower-dimensional its symmetry group can be infinite.

Example 7. *Consider the positive quadrant $\mathcal{X} = \{x \in \mathbb{R}^2 : x \geq 0\}$ shown in Figure 4.2a. This set $\mathcal{X}$ is unbounded. Its symmetry group $\operatorname{Aut}(\mathcal{X})$ is the infinite set*

$$\operatorname{Aut}(\mathcal{X}) = \left\{ \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}, \begin{bmatrix} 0 & \alpha \\ \beta & 0 \end{bmatrix} : \alpha, \beta > 0 \right\}. \tag{4.20}$$

This symmetry group $\operatorname{Aut}(\mathcal{X})$ says we can positively scale the space $\mathbb{R}^2$ and permute the basis vectors x_1 and x_2 without changing the set $\mathcal{X}$.

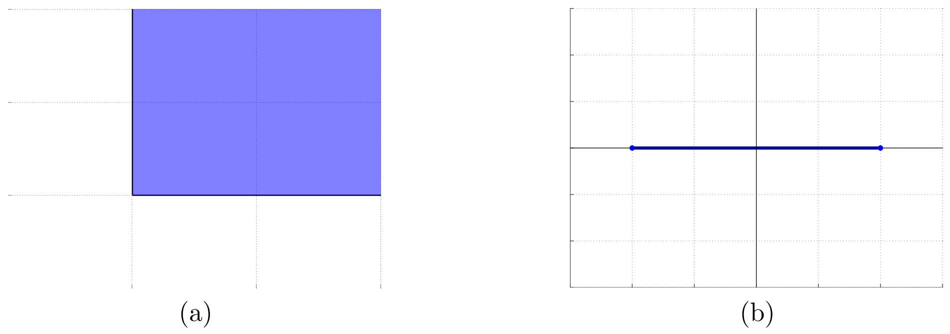


Figure 4.2: Examples of polytopes with infinite symmetry groups. (a) Unbounded polytope $\mathcal{X} = \{x \in \mathbb{R}^2 : x \geq 0\}$ with infinite symmetry group $\text{Aut}(\mathcal{X})$. (b) Lower-dimensional polytope with infinite symmetry group $\text{Aut}(\mathcal{X})$.

Example 8. *Consider the lower-dimensional set*

$$\mathcal{X} = \text{conv}\left\{\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}-1\\0\end{bmatrix}\right\} \tag{4.21}$$

shown in Figure 4.2b. The symmetry group $\text{Aut}(\mathcal{X})$ *of this set* $\mathcal{X}$ *is the infinite set*

$$\text{Aut}(\mathcal{X}) = \left\{\begin{bmatrix}1 & 0\\0 & \theta\end{bmatrix}, \begin{bmatrix}-1 & 0\\0 & \theta\end{bmatrix} : \theta \in \mathbb{R} \setminus \{0\}\right\}. \tag{4.22}$$

This symmetry group $\text{Aut}(\mathcal{X})$ *says that we can arbitrarily scale the* x_2 *dimension without changing the set* $\mathcal{X}$*. In the* x_1 *dimension we can only permute the vertices* $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ *and* $\left[\begin{smallmatrix}-1\\0\end{smallmatrix}\right]$

Since $\text{Aut}(\mathcal{X},\mathcal{U})$ is finite for bounded full-dimensional polytopes $\mathcal{X}$ and $\mathcal{U}$ this implies that $\text{Aut}(\Sigma) = \text{Aut}(A, B) \cap \text{Aut}(\mathcal{X},\mathcal{U})$ is finite. Therefore we will need to use techniques from discrete mathematics to analyze this group.

The following example demonstrates symmetry for linear system with polytopic constraints.

Example 9. *Consider the constrained linear autonomous system*

$$x(t+1) = Ax(t), x(t) \in \mathcal{X} \tag{4.23}$$

where $\mathcal{X}$ *is the octagon shown in Figure 4.3a and*

$$A = \begin{bmatrix}0.2 & 0.5\\-0.5 & 0.2\end{bmatrix}. \tag{4.24}$$

The set $\mathcal{X}$ *is symmetric with respect to the dihedral-8 group* $\text{Aut}(\mathcal{X}) = \mathcal{D}_8$ *which includes planar rotations by* 45 *degrees. The matrix* A *is invariant under all rotations* $\mathcal{SO}(2) \subset$

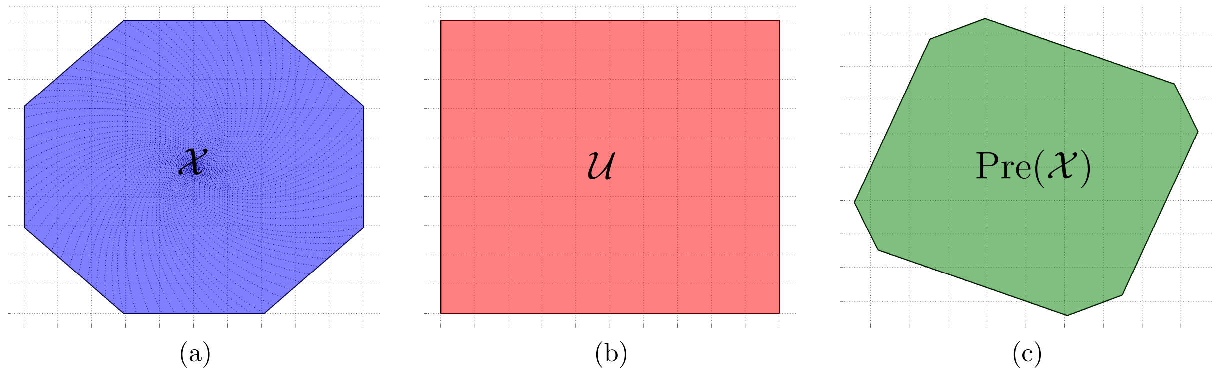


Figure 4.3: Example of symmetric of constrained linear system. (a) State constraint set $\mathcal{X}$ with vector-field $f(x) = Ax$. (b) Input constraint set $\mathcal{U}$. (c) Set $\text{Pre}(\mathcal{X})$ of states that can be driven into $\mathcal{X}$.

$\text{Aut}(A)$. *Therefore the symmetry group* $\text{Aut}(\Sigma)$ *of the autonomous system is the cyclic-8 group* $\text{Aut}(\Sigma) = \mathcal{C}_8$ *of rotations by* 45 *degrees.*

Now we add control inputs to the system

$$x(t+1) = Ax(t) + Bu(t) \tag{4.25}$$

$$x(t) \in \mathcal{X}, u(t) \in \mathcal{U} \tag{4.26}$$

where A *and* $\mathcal{X}$ *are the same as before, and* $B = I \in \mathbb{R}^{2\times 2}$ *is the identity matrix and* $\mathcal{U}$ *is the square shown in Figure 4.3b. The symmetry group* $\text{Aut}(\mathcal{U})$ *of the square* $\mathcal{U}$ *is the dihedral-4 group* $\text{Aut}(\mathcal{U}) = \mathcal{D}_4$*. Therefore* $\text{Aut}(\mathcal{X}, \mathcal{U}) = \mathcal{D}_8 \times \mathcal{D}_4$ *is the set of pairs* (Θ, Ω) *such that* $\Theta \in \mathcal{D}_8$ *and* $\Omega \in \mathcal{D}_4$*.*

The dynamics couple the state and input spaces. For each $\Theta \in \text{Aut}(A)$ *the matrix* $\Omega = \Theta$ *commutes* $\Theta B = B\Omega$ *with the matrix* $B = I$*. Therefore we are no longer able to choose* Θ *and* Ω *independently. Thus the symmetry group* $\text{Aut}(A, B)$ *of the dynamics is*

$$\text{Aut}(A, B) = \big\{(\Theta, \Omega) : \Theta \in \text{Aut}(A), \Omega = \Theta\big\}. \tag{4.27}$$

Thus the symmetry group $\text{Aut}(\Sigma) = \text{Aut}(A, B) \cap \text{Aut}(\Sigma)$ *is the cyclic-4 group* $\text{Aut}(\Sigma) = \mathcal{C}_4$*. This can be seen in the pre-set* $\text{Pre}(\mathcal{X})$ *of* $\mathcal{X}$ *shown in Figure 4.3c. The set* $\text{Pre}(\mathcal{X})$ *is not symmetric under rotations of* 45 *degrees, only rotations by* 90 *degrees.*

Finally we provide a corollary of Proposition 6 that allows the construction of symmetric control invariant sets for constrained linear systems.

Corollary 1. *Let* $\mathcal{C}$ *be a control invariant set for the system* (4.9) *subject to polytopic constraints* (4.11)*. Then the set* $\bar{\mathcal{C}} = \text{conv}\{\Theta\mathcal{C} : \Theta \in \text{Aut}(\Sigma)\}$ *is a symmetric control invariant set.*

Proof. The set $\bar{\mathcal{C}}$ is symmetric since

$$\hat{\Theta}\bar{\mathcal{C}} = \mathrm{conv}\{\hat{\Theta}\Theta\mathcal{C} : \Theta \in \mathcal{G}\} = \mathrm{conv}\{\Theta\mathcal{C} : \Theta \in \mathcal{G}\} = \bar{\mathcal{C}} \tag{4.28}$$

for all $\hat{\Theta} \in \mathcal{G} = \mathrm{Aut}(\Sigma)$.

Next we show this set is control invariant. By the definition of $\bar{\mathcal{C}}$ for any $x \in \bar{\mathcal{C}}$ there exists $x_i \in \Theta_i\mathcal{C}$ for $i = 1, \dots, |\mathcal{G}|$ such that x is the convex combination

$$x = \sum_{i=1}^{|\mathcal{G}|} \lambda_i x_i. \tag{4.29}$$

By the definition of control invariance for each $\Theta_i^{-1} x_i \in \mathcal{C}$ there exists $v_i \in \mathcal{U}$ such that

$$A\Theta_i^{-1} x + Bv_i \in \mathcal{C}. \tag{4.30}$$

Thus $u_i = \Omega v_i \in \Omega\mathcal{U} = \mathcal{U}$ satisfies $Ax_i + Bu_i \in \Theta_i\mathcal{C}$. Therefore we have shown there exists $u = \sum_{i=1}^{|\mathcal{G}|} \lambda_i u_i \in \mathcal{U}$ such that

$$\begin{aligned} Ax + Bu = A\Big(\textstyle\sum_{i=1}^{|\mathcal{G}|}\lambda_i x_i\Big) + B\Big(\textstyle\sum_{i=1}^{|\mathcal{G}|}\lambda_i u_i\Big) &= \textstyle\sum_{i=1}^{|\mathcal{G}|}\lambda_i\big(Ax_i + Bu_i\big) \\ \in \mathrm{conv}\big\{\Theta\mathcal{C} : \Theta \in \mathcal{G}\big\} &= \bar{\mathcal{C}}. \end{aligned} \tag{4.31}$$

□

This corollary is useful when we cannot calculate the maximal control invariant set $\mathcal{C}_\infty$ for a constrained linear system but we have a non-symmetric control invariant set $\mathcal{C} \subseteq \mathcal{C}_\infty$. This corollary allows us to build a symmetric control invariant set $\bar{\mathcal{C}} \supseteq \mathcal{C}$ from $\mathcal{C}$.

4.3 Symmetries of Controllers

A symmetry of the control-law $u = \kappa(x)$ is a state-space Θ and input-space Ω transformation that preserves the control-law.

Definition 7. The pair of invertible matrices (Θ, Ω) is a symmetry of the control-law $u = \kappa(x)$ if

$$\Omega\kappa(x) = \kappa(\Theta x) \tag{4.32}$$

for all $x \in \mathcal{X}$

Definition 7 says that the control actions at point x and $y = \Theta x$ are related by a linear transformation Ω. For a linear controller $u = Fx$, Definition 7 says that a symmetry is a pair of invertible matrices Θ and Ω that commute $\Omega F = F\Theta$ with the feedback gain matrix F.

The set of all symmetries of the control-law $u = \kappa(x)$ is a group denoted by $\mathrm{Aut}(\kappa)$.

Proposition 9. *The set* $\mathrm{Aut}(\kappa)$ *of all invertible matrices* (Θ, Ω) *that satisfy* (4.32) *is a group under pairwise matrix multiplication.*

Next we use the symmetry groups $\mathrm{Aut}(\Sigma)$ and $\mathrm{Aut}(\kappa)$ to prove some basic results about reachability for symmetric constrained systems in closed-loop with symmetric controllers $\kappa(x)$. The following proposition shows that the maximal positive invariant set $\mathcal{O}_\infty$ is symmetric.

Proposition 10. *The maximal positive invariant set* $\mathcal{O}_\infty$ *of the system* (4.1) *subject to the constraints* (4.11) *in closed-loop with the controller* $\kappa(x)$ *is symmetric* $\Theta\mathcal{O}_\infty = \mathcal{O}_\infty$ *with respect to the symmetry group* $\mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$.

Proof. Consider the set

$$\mathcal{O} = \bigcup_{\Theta \in \mathcal{G}} \Theta \mathcal{O}_\infty \tag{4.33}$$

where $\mathcal{G} = \mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$. This set is symmetric by construction $\mathcal{O} = \Theta\mathcal{O}$ for all $\Theta \in \mathcal{G} = \mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$. We will show $\mathcal{O} = \mathcal{O}_\infty$.

First we will prove $\mathcal{O}$ is positive invariant. Let $x \in \mathcal{O}$ then there exists $\Theta \in \mathcal{G}$ such that $\Theta^{-1}x \in \mathcal{O}_\infty$. By the definition of positive invariant

$$f\big(\Theta^{-1}x, \kappa(\Theta^{-1}x)\big) \in \mathcal{O}_\infty \tag{4.34}$$

where $\kappa(\Theta^{-1}x) = \Omega^{-1}\kappa(x) \in \Omega^{-1}\mathcal{U} = \mathcal{U}$ is a feasible input. By the symmetry of the dynamics $\mathrm{Aut}(\Sigma)$ and controller $\mathrm{Aut}(\kappa)$ we have

$$f\big(x, \kappa(x)\big) \in \Theta\mathcal{O}_\infty \subseteq \mathcal{O}. \tag{4.35}$$

Thus for any $x \in \mathcal{O}$ we have $f\big(x, \kappa(x)\big) \in \mathcal{O}$. Therefore $\mathcal{O}$ is a positive invariant set.

Since $\mathcal{O}_\infty$ is the maximal positive invariant set we have $\mathcal{O} \subseteq \mathcal{O}_\infty$. However by construction $\mathcal{O}_\infty \subseteq \mathcal{O}$. Therefore $\mathcal{O} = \mathcal{O}_\infty$ is symmetric with respect to the group $\mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$. □

The following proposition shows that the set of states x that the controller $\kappa(x)$ drives into a symmetric set $\mathcal{S}$ is symmetric. Likewise the set of states x that the controller $\kappa(x)$ can reach from the symmetric set $\mathcal{S}$ is symmetric.

Proposition 11. *Let* $\mathcal{S} \subseteq \mathbb{R}^n$ *be a symmetric set* $\Theta\mathcal{S} = \mathcal{S}$ *for all* $\Theta \in \mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$. *Then the backward* $\mathrm{Pre}(\mathcal{S})$ *and forward* $\mathrm{Reach}(\mathcal{S})$ *reachable sets for the system* (4.1) *in closed-loop with the controller* $\kappa(x)$ *are symmetric*

$$\Theta\,\mathrm{Pre}(\mathcal{S}) = \mathrm{Pre}(\mathcal{S}) \tag{4.36a}$$

$$\Theta\,\mathrm{Reach}(\mathcal{S}) = \mathrm{Reach}(\mathcal{S}) \tag{4.36b}$$

for all $\Theta \in \mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(\kappa)$.

Proof. Let $y \in \text{Reach}(\mathcal{S})$ be forward reachable. Then there exists $x \in \mathcal{S}$ such that $y = f(x, \kappa(x))$. Thus $\Theta y \in \text{Reach}(\mathcal{S})$ since

$$\Theta y = \Theta f\big(x, \kappa(x)\big) = f\big(\Theta x, \kappa(\Theta x)\big) \tag{4.37}$$

where $\Theta x \in \Theta\mathcal{S} = \mathcal{S}$ and $\kappa(\Theta x) = \Omega\kappa(x) \in \Omega\mathcal{U} = \mathcal{U}$. Therefore $\Theta\,\text{Reach}(\mathcal{S}) \subseteq \text{Reach}(\mathcal{S})$. Since $\text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$ is a group, for any $\Theta \in \text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$ we have $\Theta^{-1} \in \text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$. Therefore $\Theta\,\text{Reach}(\mathcal{S}) = \text{Reach}(\mathcal{S})$.

Let $x \in \text{Pre}(\mathcal{S})$ be backward reachable. Then $f(x, \kappa(x)) \in \mathcal{S}$. Thus $\Theta x \in \text{Pre}(\mathcal{S})$ since

$$f(\Theta x, \kappa(\Theta x)) = \Theta f(x, u) \in \Theta\mathcal{S} = \mathcal{S} \tag{4.38}$$

where $\kappa(\Theta x) = \Omega\kappa(x) \in \Omega\mathcal{U} = \mathcal{U}$. Therefore $\Theta\,\text{Pre}(\mathcal{S}) \subseteq \text{Pre}(\mathcal{S})$. Since $\text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$ is a group, for any $\Theta \in \text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$ we have $\Theta^{-1} \in \text{Aut}(\Sigma) \cap \text{Aut}(\kappa)$ therefore $\Theta\,\text{Pre}(\mathcal{S}) = \text{Pre}(\mathcal{S})$. □

Symmetries of Piecewise Affine Controllers

We are particularly interested in symmetries of piecewise affine on polytopes control-laws

$$\kappa(x) = \begin{cases} F_1 x + G_1 & x \in \mathcal{R}_1 \\ \quad\vdots & \\ F_p x + G_p & x \in \mathcal{R}_p \end{cases} \tag{4.39}$$

where $F_i \in \mathbb{R}^{m \times n}$ and $G_i \in \mathbb{R}^m$ are the feedback and feedforward gains, and $\mathcal{R}_i$ are polytopes. The regions $\mathcal{R}_i$ for $i \in \mathcal{I}$ partition the domain $\mathcal{X}$ of the controller $\kappa(x)$.

We are interested in symmetries $(\Theta, \Omega) \in \text{Aut}(\kappa)$ of the controller $\kappa(x)$ that permute the controller pieces $(F_i, G_i, \mathcal{R}_i)$

$$\Omega F_i = F_j \Theta \tag{4.40a}$$
$$\Omega G_i = G_j \tag{4.40b}$$
$$\Theta \mathcal{R}_i = \mathcal{R}_j. \tag{4.40c}$$

Equation (4.40) says that the state-space transformation Θ maps the i-th region $\mathcal{R}_i \subseteq \mathcal{X}$ to the j-th region $\mathcal{R}_j = \Theta\mathcal{R}_i$. Furthermore the symmetry (Θ, Ω) maps the control-law in region $\mathcal{R}_i$ to the control-law in region $\mathcal{R}_j$. The control-law for $x \in \mathcal{R}_j = \Theta\mathcal{R}_i$ is

$$\begin{aligned} \kappa(x) &= F_j x + G_j \\ &= \Omega F_i \Theta^{-1} x + \Omega G_i \end{aligned} \tag{4.41}$$

where F_i and G_i are the feedback and feedforward gains respectively for region $\mathcal{R}_i$. We will denote the subgroup of symmetries that satisfy (4.40) by $\text{Aut}(\mathcal{I})$.

Clearly symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$ that permute the pieces $\mathcal{I}$ of the controller $\kappa(x)$ are symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ of the controller $\kappa(x)$. However symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ of the control-law $\kappa(x)$ do not necessarily permute the controller pieces $\mathcal{I}$ i.e. $\mathrm{Aut}(\mathcal{I}) \subseteq \mathrm{Aut}(\kappa)$. The symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$ that permute the controller pieces $\mathcal{I}$ depend on how the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is chosen. For instance we can break symmetry by splitting region $\mathcal{R}_i$ but not its image $\mathcal{R}_j = \Theta\mathcal{R}_i$. The following theorem shows that when $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is the minimal p-collection partition for the control-law $\kappa(x)$ then the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ of the controller $\kappa(x)$ permute the pieces $\mathcal{I}$ i.e. $\mathrm{Aut}(\mathcal{I}) = \mathrm{Aut}(\kappa)$.

Theorem 7. *Let $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ be the minimal p-collection partition of the domain of κ. Then $\mathrm{Aut}(\mathcal{I}) = \mathrm{Aut}(\kappa)$.*

Proof. Any symmetry $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ of the control-law $\kappa(x)$ is a symmetry of its piecewise affine expression (4.39). Thus $\mathrm{Aut}(\mathcal{I}) \subseteq \mathrm{Aut}(\kappa)$. Next we show the converse.

Suppose $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ does not permute the controller pieces $(\Theta, \Omega) \notin \mathrm{Aut}(\mathcal{I})$. Then there exists $i \in \mathcal{I}$ such that (4.40) does not hold for any $j \in \mathcal{I}$.

Consider the set $\Theta\mathcal{R}_i \subset \mathcal{X}$. There exists a minimal finite covering $\mathcal{J} \subseteq \mathcal{I}$ of this set

$$\mathrm{int}\left(\Theta\mathcal{R}_i\right) \subseteq \bigcup\nolimits_{j\in\mathcal{J}} \mathcal{R}_j \tag{4.42}$$

since $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is a partition of the domain $\mathcal{X}$ of the controller $\kappa(x)$ and $\Theta\mathcal{R}_i \subseteq \mathcal{X}$ is a subset of the domain $\mathcal{X} = \Theta\mathcal{X}$. The sets

$$\mathcal{P}_j = (\Theta\mathcal{R}_i) \cap \mathcal{R}_j \tag{4.43}$$

for $j \in \mathcal{J}$ are full-dimensional since $\Theta\mathcal{R}_i$ is full-dimensional and $\mathcal{J}$ is a minimal cover. Therefore each $\mathcal{P}_j$ contains a set of affinely independent points. Since $\Omega\kappa(x) = \kappa(\Theta x)$ at each of these affinely independent points we have

$$\Omega F_i = F_j \Theta \tag{4.44a}$$

$$\Omega G_i = G_j \tag{4.44b}$$

for each $j \in \mathcal{J}$. Thus $(\Omega F_i \Theta^{-1}, \Omega G_i, \Theta\mathcal{R}_i)$ and $(F_j, G_j, \mathcal{R}_j)$ differ only by the partition $\Theta\mathcal{R}_i \neq \mathcal{R}_j$.

Suppose $|\mathcal{J}| = 1$ then $\Theta\mathcal{R}_i \subset \mathcal{R}_j$ since $\Theta\mathcal{R}_i \neq \mathcal{R}_j$. In this case we can reverse the roles of i and j and consider a cover $\mathcal{K}$ of $\Theta^{-1}\mathcal{R}_j$ where $|\mathcal{K}| > 1$. Since $\mathrm{Aut}(\mathcal{I})$ is a group $(\Theta^{-1}, \Omega^{-1}) \in \mathrm{Aut}(\mathcal{I})$ implies $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$. Thus we can assume without loss of generality that $|\mathcal{J}| > 1$.

For $|\mathcal{J}| > 1$ we can merge the regions $\mathcal{R}_j$ for $j \in \mathcal{J}$. However this contradicts the assumption that $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is the minimal partition. Therefore we conclude that $(\Theta, \Omega) \in \mathrm{Aut}(\kappa)$ permutes the controller pieces $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$. $\square$

The following example shows that this result does not hold when $\mathcal{R}_i$ are not p-collections.

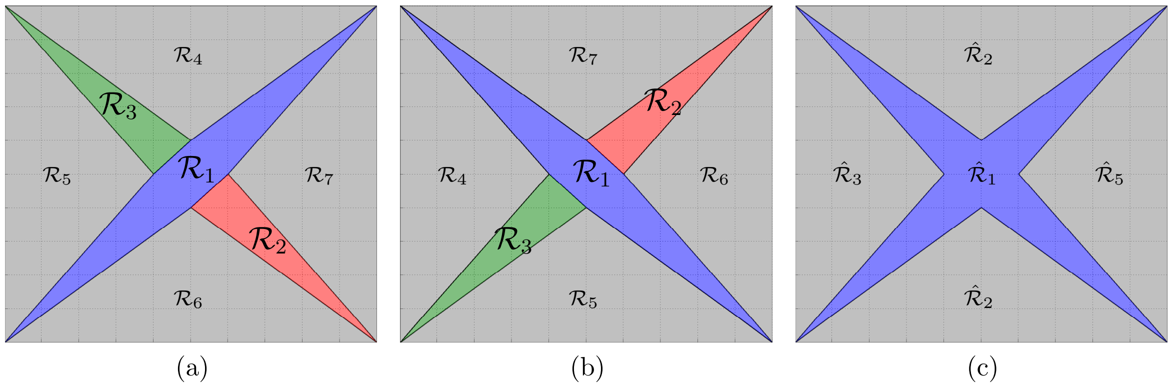


Figure 4.4: Example of symmetry under the minimal p-collection partition. (a) Convex partition $\{\mathcal{R}_i\}_{i=1}^7$ for the symmetric controller $\kappa(x)$. (b) The image $\{\Theta\mathcal{R}_i\}_{i=1}^7$ of the partition $\{\mathcal{R}_i\}_{i=1}^7$ under the 90 degree counter-clockwise rotation matrix Θ. (c) The unique minimal p-collection partition $\{\hat{\mathcal{R}}_i\}_{i=1}^5$ for the symmetric controller $\kappa(x)$.

Example 10. *Let $\kappa(x)$ be a piecewise affine on p-collection controller that is symmetric with respect to the dihedral-4 group $\text{Aut}(\kappa) = \mathcal{D}_4$ with the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ shown in Figure 4.4a.*

For the partition shown in Figure 4.4a the symmetries $\Theta \in \text{Aut}(\kappa)$ do not permute $\text{Aut}(\kappa) \neq \text{Aut}(\mathcal{I})$ the controller pieces $\mathcal{I} = \{1,2,3,4,5,6,7\}$. For instance a 90 degree rotation Θ will map the partition in Figure 4.4a to the partition in Figure 4.4b. This is because the partition is not the minimal p-collection partition.

By symmetry of the controller $\kappa(x)$ we know the feedback F_i and feedforward gains G_i in regions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ must be the same. Therefore we can merge these regions as shown in Figure 4.4c. This symmetric partition is the minimal p-collection partition. Under this partition symmetries $\Theta \in \text{Aut}(\kappa)$ of the controller $\kappa(x)$ permute the regions $\hat{\mathcal{R}}_i$. Therefore $\text{Aut}(\kappa) = \text{Aut}(\mathcal{I})$.

The group $\text{Aut}(\mathcal{I})$ is the set of controller symmetries $(\Theta, \Omega) \in \text{Aut}(\kappa)$ that permute the controller pieces $(F_i, G_i, \mathcal{R}_i)$ for $i \in \mathcal{I}$. Each symmetry $(\Theta, \Omega) \in \text{Aut}(\mathcal{I})$ corresponds to a permutation $\pi : \mathcal{I} \to \mathcal{I}$ of the controller pieces $\mathcal{I}$ where $j = \pi(i)$ for i and j satisfying (4.40). It will be convenient to define a function $\mathbf{\Pi}$ that maps transformations $(\Theta, \Omega) \in \text{Aut}(\mathcal{I})$ to the corresponding permutation $\pi = \mathbf{\Pi}(\Theta, \Omega)$ of the controller pieces $\pi : \mathcal{I} \to \mathcal{I}$. This function is a group homomorphism from $\text{Aut}(\kappa)$ to $\mathfrak{S}_{\mathcal{I}}$.

Proposition 12. *The function $\mathbf{\Pi} : \text{Aut}(\mathcal{I}) \to \mathfrak{S}_{\mathcal{I}}$ is a group homomorphism. The subgroup $\text{Aut}(\kappa)/\,\text{Ker}(\mathbf{\Pi})$ is finite.*

Proof. Suppose $(\Theta_1, \Omega_1) \in \text{Aut}(\kappa)$ corresponds to the permutation $\pi_1 = \mathbf{\Pi}(\Theta_1, \Omega_1)$ and $(\Theta_2, \Omega_2) \in \text{Aut}(\kappa)$ corresponds to the permutation $\pi_2 = \mathbf{\Pi}(\Theta_2, \Omega_2)$. Then clearly $(\Theta_1\Theta_2, \Omega_1\Omega_2) \in \text{Aut}(\kappa)$ corresponds to the permutation $\pi_1\pi_2 \in \mathbf{\Pi}(\text{Aut}(\kappa))$. Thus $\mathbf{\Pi}$ is a group homomorphism.

Since $\mathbf{\Pi}$ is a group homomorphism, the subgroup $\mathrm{Aut}(\kappa)/\operatorname{Ker}(\mathbf{\Pi})$ isomorphic to the group $\mathbf{\Pi}(\mathrm{Aut}(\kappa))$ which if finite since it is a permutation group on a finite set $\mathcal{I}$. □

The function $\mathbf{\Pi}$ is not necessarily an isomorphism. There may be symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$ that permute the controller pieces but simply send every controller pieces $i \in \mathcal{I}$ to itself. These are the elements of the subgroup $\operatorname{Ker}(\mathbf{\Pi}) = \{(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I} : \mathbf{\Pi}(\Theta, \Omega) = e\}$ where $e(i) = i$ is the identity permutation. In Chapter 7 we will discuss how symmetry can be used to reduce the complexity of piecewise affine on polytopes control-laws. Symmetries $(\Theta, \Omega) \in \operatorname{Ker}(\mathbf{\Pi})$ in the set $\operatorname{Ker}(\mathbf{\Pi})$ cannot be used to compress the controller. The subgroup $\operatorname{Ker}(\mathbf{\Pi})$ can potentially be infinite as shown in the following example.

Example 11. *For an open-loop stable system we can use the controller $u = \kappa(x) = 0$ for all $x \in \mathcal{X}$. This controller has one piece $\kappa(x) = 0x + 0$ for $x \in \mathcal{R} = \mathcal{X}$. Therefore there is only one permute of the controller pieces $|\mathbf{\Pi}(\mathrm{Aut}(\kappa))| = 1$. However the symmetry group $\mathrm{Aut}(\kappa)$ of the controller is infinite since any invertible matrix $\Omega \in \mathbb{R}^{m \times m}$ is a symmetry $\Omega 0 = 0$. Therefore the set of useless symmetries $\operatorname{Ker}(\mathbf{\Pi})$ is infinite.*

4.4 Symmetries of Model Predictive Controllers

Consider the following model predictive control problem

$$
\begin{aligned}
J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k) && (4.45a)\\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k) && (4.45b)\\
& x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} && (4.45c)\\
& x_N \in \mathcal{X}_N && (4.45d)\\
& x_0 = x && (4.45e)
\end{aligned}
$$

where $x \in \mathbb{R}^n$ is the measured state, x_k is the predicted state under the control action $u_k \in \mathbb{R}^m$ over the horizon N, and $\mathcal{X}_N$ is the terminal constraint set.

A symmetry of the cost function (4.45a) is a state-space Θ and input-space Ω transformation that preserves the cost function $J(X, U) = p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$.

Definition 8. The pair of invertible matrices (Θ, Ω) is a symmetry of the cost function (4.45a) if it satisfies

$$
\begin{aligned}
p(\Theta x) &= p(x) && (4.46a)\\
q(\Theta x, \Omega u) &= q(x, u). && (4.46b)
\end{aligned}
$$

Definition 8 says that the cost (4.45a) of the state trajectory $\{x(t)\}_{t=0}^{N}$ and input trajectory $\{u(t)\}_{t=0}^{N-1}$ is the same as the cost (4.45a) of the state trajectory $\{\Theta x(t)\}_{t=0}^{N}$ and input trajectory $\{\Omega u(t)\}_{t=0}^{N-1}$. The set of all symmetries of the cost function (4.45a) is a group denoted by $\mathrm{Aut}(J)$.

Proposition 13. *The set* Aut(J) *of all invertible matrices* (Θ, Ω) *that satisfy* (4.46) *is a group under pairwise matrix multiplication.*

The symmetry group $\mathrm{Aut}(MPC) = \mathrm{Aut}(J) \cap \mathrm{Aut}(\mathcal{X}_N) \cap \mathrm{Aut}(\Sigma)$ of the model predictive control problem (4.45) is the intersection of the symmetry groups of the cost function Aut(J), the terminal set Aut$(\mathcal{X}_N)$, and the constrained system $\mathrm{Aut}(\Sigma) = \mathrm{Aut}(f) \cap \mathrm{Aut}(\mathcal{X}, \mathcal{U})$.

Typically the terminal set $\mathcal{X}_N$ is chosen to be a control invariant set to guarantee persistent feasibility. In Proposition 6 we showed the maximal control invariant set $\mathcal{C}_\infty$ is symmetric with respect to the symmetry group Aut(Σ) of the constrained system. Furthermore in Corollary 1 we showed how to construct a symmetric control invariant set $\bar{\mathcal{C}}$ for a constrained linear system. Thus we will assume the terminal set $\mathcal{X}_N$ is symmetric with respect to the symmetry group $\mathrm{Aut}(\Sigma) \subseteq \mathrm{Aut}(\mathcal{X}_N)$ of the constrained system. Therefore the symmetry group $\mathrm{Aut}(MPC) = \mathrm{Aut}(J) \cap \mathrm{Aut}(\Sigma)$ of the model predictive control problem (4.45) is the intersection of the symmetry groups of the cost function Aut(J) and the constrained system $\mathrm{Aut}(\Sigma) = \mathrm{Aut}(f) \cap \mathrm{Aut}(\mathcal{X}, \mathcal{U})$.

We will use the group Aut(MPC) to study the symmetry of model predictive controllers. Our analysis will be based on the dynamic programming formulation of the model predictive control problem (4.45). We define the terminal cost-to-go as

$$J_N^\star(x_N) = p(x_N) \tag{4.47}$$

where $p(x_N)$ is the terminal cost of the model predictive control problem (4.45a). This cost $J_N^\star(x_N)$ is defined on the terminal set $\mathcal{X}_N$. For each stage $k = N-1, \dots, 0$ the cost-to-go is the solution to the one-step optimization problem

$$\begin{aligned} J_k^\star(x) = &\underset{u \in \mathcal{U}}{\text{minimize}} \quad q(x,u) + J_{k+1}^\star\big(f(x,u)\big) && (4.48\text{a}) \\ &\text{subject to } f(x,u) \in \mathcal{X}_{k+1}. && (4.48\text{b}) \end{aligned}$$

The cost function $J_k^\star(x)$ is defined on the feasible set

$$\begin{aligned} \mathcal{X}_k &= \big\{ x \in \mathcal{X} : \exists u \in \mathcal{U}, f(x,u) \in \mathcal{X}_{k+1} \big\} \\ &= \mathrm{Pre}(\mathcal{X}_{k+1}) \cap \mathcal{X}. \end{aligned} \tag{4.49}$$

For each stage $k = N-1, \dots, 0$ the optimal control-law $u_k^\star(x)$ is an optimizer $u_k^\star : \mathcal{X}_k \to \mathcal{U}$ of the one-step optimization (4.48). A model predictive controller $u_0^\star(x)$ is an optimizer at stage $k = 0$. The following theorem shows that the cost-to-go $J_k^\star(x)$ and feasible region $\mathcal{X}_k$ are symmetric at each stage $k = 0, \dots, N$. Furthermore the one-step optimization problem has a symmetry solution $u_k^\star(x)$ at each stage $k = 0, \dots, N$.

Theorem 8. *Let the model predictive control problem* (4.45) *be convex. Then the cost-to-go* $J_k^\star(x)$ *and feasible region* $\mathcal{X}_k$ *are symmetric*

$$\begin{aligned} J_k^\star(\Theta x) &= J_k^\star(x) && (4.50\text{a}) \\ \Theta \mathcal{X}_k &= \mathcal{X}_k && (4.50\text{b}) \end{aligned}$$

for all $\Theta \in \mathrm{Aut}(MPC)$ *and* $k = 0, \dots, N$*. The one-step optimization problem* (4.48) *has a symmetric solution* $\bar{u}_0^\star(x)$

$$\bar{u}_k^\star(\Theta x) = \Omega \bar{u}_k^\star(x) \tag{4.51}$$

for all $\Theta \in \mathrm{Aut}(MPC)$ *and* $k = 0, \dots, N-1$*.*

Proof. First we will prove the symmetry of $J_k^\star(x)$ and $\mathcal{X}_k$ by induction on k. Note that $J_N^\star(x) = p(x)$ is symmetric since $p(\Theta x) = p(x)$ by definition of $\mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(J)$. Likewise $\mathcal{X}_N$ is symmetric since $\Theta\mathcal{X}_N = \mathcal{X}_N$ by the assumption on the terminal set $\mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(\Sigma) \subseteq \mathrm{Aut}(\mathcal{X}_N)$.

Assume $J_{k+1}^\star(x)$ and $\mathcal{X}_{k+1}$ are symmetric with respect to $\mathrm{Aut}(MPC)$. Then we will show $J_k^\star(x)$ and $\mathcal{X}_k$ are symmetric. By Proposition 7 the set $\mathrm{Pre}(\mathcal{X}_{k+1})$ is symmetric with respect to $\mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(\Sigma)$. Thus $\mathcal{X}_k = \mathrm{Pre}(\mathcal{X}_{k+1}) \cap \mathcal{X}$ is symmetric since it is the intersection of symmetric sets.

Next we will prove the symmetry of $J_k^\star(x)$. Let $u_k^\star(x)$ be an optimal solution to (4.48) at the state x. Then $\Omega u_k^\star(x)$ is a feasible solution to (4.48) at the state Θx since $\Omega u_k^\star(x) \in \Omega\mathcal{U} = \mathcal{U}$ and $f(\Theta x, \Omega u_k^\star(x)) = \Theta f(x, u_k^\star(x)) \in \Theta\mathcal{X}_{k+1} = \mathcal{X}_{k+1}$. Using this input $\Omega u_k^\star(x)$ we obtain an upper-bound on the cost-to-go $J_k^\star(\Theta x)$ at Θx

$$\begin{aligned} J_k^\star(\Theta x) &\le q(\Theta x, \Omega u_k^\star(x)) + J_{k+1}^\star\big(f(\Theta x, \Omega u_k^\star(x))\big) && (4.52)\\ &= q(x, u_k^\star(x)) + J_{k+1}^\star\big(\Theta f(x, u_k^\star(x))\big)\\ &= q(x, u_k^\star(x)) + J_{k+1}^\star\big(f(x, u_k^\star(x))\big)\\ &= J_k^\star(x). && (4.53) \end{aligned}$$

Thus $J_k^\star(\Theta x) \le J_k^\star(x)$ for all $\Theta \in \mathrm{Aut}(MPC)$ and $x \in \mathcal{X}_{k+1}$. Since $\mathrm{Aut}(MPC)$ is a group for every $\Theta \in \mathrm{Aut}(MPC)$ we have $\Theta^{-1} \in \mathrm{Aut}(MPC)$. Therefore $J_k^\star(\Theta x) = J_k^\star(x)$.

Finally we prove (4.48) has a symmetric optimal solution $\bar{u}_k^\star(x)$. Let $u_k^\star(x)$ be an optimal solution to (4.48). Define

$$\bar{u}_k^\star(x) = \frac{1}{|\mathcal{G}|} \sum_{(\Theta,\Omega)\in\mathcal{G}} \Omega^{-1} u_k^\star(\Theta x) \tag{4.54}$$

where $\mathcal{G} = \mathrm{Aut}(MPC)$. By construction $\bar{u}_k^\star(x)$ is symmetric $\Omega\bar{u}_k^\star(x) = \bar{u}_k^\star(\Theta x)$ for all $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$.

We will show that the function $\bar{u}_k^\star(x)$ is a convex combination of optimal solutions $\Omega^{-1}u_k^\star(\Theta x)$ to (4.48). First we note that $\Omega^{-1}u_k^\star(\Theta x)$ is a feasible solution to (4.48) since $\Omega^{-1}u_k^\star(\Theta x) \in \Omega^{-1}\mathcal{U} = \mathcal{U}$ and $f\big(x, \Omega^{-1}u(\Theta x)\big) = \Theta^{-1} f\big(\Theta x, u(\Theta x)\big) \in \Theta^{-1}\mathcal{X}_{k+1} = \mathcal{X}_{k+1}$ where $\Theta x \in \Theta\mathcal{X}_k = \mathcal{X}_k$. Thus $\bar{u}_k^\star(x)$ is a feasible solution since it is a convex combination of feasible solutions and the feasible set of (4.48) is convex.

Next we show $\Omega^{-1}u_k^\star(\Theta x)$ is an optimal solution to (4.48). Note

$$\begin{aligned} q\big(x, \Omega^{-1}u_k^\star(\Theta x)\big) + J_{k+1}^\star\big(f(x, \Omega^{-1}u_k^\star(\Theta x))\big) &= q(\Theta x, u_k^\star(\Theta x) + J_{k+1}^\star\big(\Theta^{-1} f(\Theta x, u_k^\star(\Theta x))\big)\\ &= q(\Theta x, u_k^\star(\Theta x) + J_{k+1}^\star\big(f(\Theta x, u_k^\star(\Theta x))\big)\\ &= J_k^\star(\Theta x) = J_k^\star(x). && (4.55) \end{aligned}$$

Thus $\bar{u}_k^\star(x)$ is an optimal solution to (4.48) since it is a convex combination of optimal solutions and (4.48) is convex

$$\begin{aligned} q(x, \bar{u}_k^\star(x)) + J_{k+1}^\star\big(f(x, \bar{u}_k^\star(x))\big) \\ &= q\Big(x, \frac{1}{|\mathcal{G}|}\sum\nolimits_{\mathcal{G}} \Omega^{-1} u_k^\star(\Theta x)\Big) + J_{k+1}^\star\Big(f\big(x, \frac{1}{|\mathcal{G}|}\sum\nolimits_{\mathcal{G}} \Omega^{-1} u_k^\star(\Theta x)\big)\Big) \\ &\leq \frac{1}{|\mathcal{G}|}\sum\nolimits_{\mathcal{G}} q\big(x, \Omega^{-1} u_k^\star(\Theta x)\big) + J_k^\star\big(f(x, \Omega^{-1} u_k^\star(\Theta x))\big) \\ &= \frac{1}{|\mathcal{G}|}\sum\nolimits_{\mathcal{G}} J_k^\star(x) = J_k^\star(x) \end{aligned} \tag{4.56}$$

where $\mathcal{G} = \mathrm{Aut}(MPC)$. Therefore $\bar{u}_k^\star(x)$ is a symmetric optimal solution to the one-step optimization problem (4.48). □

This theorem says that every convex symmetric model predictive control problem has at least one symmetric solution $\bar{u}_0^\star(x)$ where $\mathrm{Aut}(\bar{u}_0^\star) \supseteq \mathrm{Aut}(MPC)$. However if there are multiple solutions then some of these solutions can be non-symmetric. The following corollary states that if the model predictive control problem (4.45) is strictly convex then the model predictive controller $u_0^\star(x)$ is symmetric.

Corollary 2. *Suppose* (4.45) *is strictly convex. Then the model predictive controller* $u_0^\star(x)$ *is symmetric* $\mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(u_0^\star)$.

Proof. Since (4.45) is strictly convex $u_0^\star(x)$ is its unique solution. Thus by Theorem 8 it is symmetric $\Omega u_0^\star(x) = u_0^\star(\Theta x)$ for all $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$. □

The following example shows that a symmetric model predictive controller $\bar{u}_0^\star(x)$ can have symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\bar{u}_0^\star)$ that do not appear in the model predictive control problem $(\Theta, \Omega) \notin \mathrm{Aut}(MPC)$. Thus the symmetry group of the model predictive control problem can be a strict subset $\mathrm{Aut}(MPC) \subset \mathrm{Aut}(\bar{u}_0^\star)$ of the symmetry group $\mathrm{Aut}(\bar{u}_0^\star)$ of a symmetric model predictive controller $\bar{u}_0^\star(x)$.

Example 12. *Consider the one-step model predictive control problem*

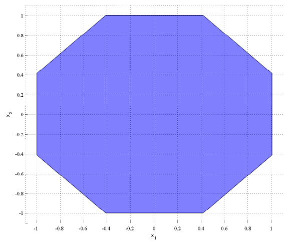
$$\underset{u_0 \in \mathcal{U}}{\text{minimize}} \quad \|Ax + Bu_0\|_2^2 + \|x\|_2^2 + \rho\|u_0\|_2^2 \tag{4.57a}$$

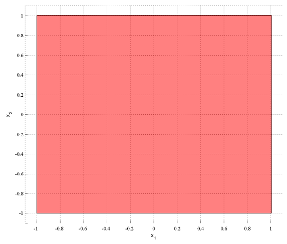
$$\text{subject to} \quad Ax + Bu_0 \in \mathcal{X} \tag{4.57b}$$

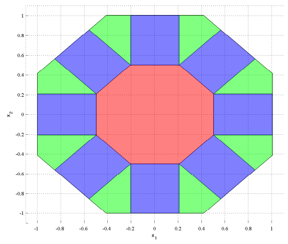
where $\rho = 5 \times 10^5$ *and the dynamics matrices are*

$$A = \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ -\sqrt{2} & \sqrt{2} \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.58}$$

The state constraints are the octagon shown in Figure 4.5a and the input constraints are the box shown in Figure 4.5b.


(a)


(b)


(c)

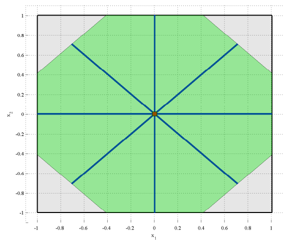

(d)

Figure 4.5: Counter-example showing the symmetry group $\mathrm{Aut}(u_0^\star)$ of the model predictive controller $u_0^\star(x)$ can be larger than the symmetry group $\mathrm{Aut}(MPC)$ of the model predictive control problem.

The model predictive control problem is symmetric with respect to rotations of the state-space and input-space by 90 *degree increments* $\mathrm{Aut}(MPC) = \mathcal{C}_4$*. However the controller* $\kappa(x)$ *is symmetric with respect to rotation of the state-space* Θ *and input-space* Ω *by* 45 *degree increments* $\mathrm{Aut}(u_0^\star) = \mathcal{C}_8$*. This symmetry can be seen in Figure 4.5c which shows the controller partition* $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ *of the piecewise affine solution* $u_0^\star(x)$ *to this model predictive control problem.*

The limiting factor in the symmetry group $\mathrm{Aut}(MPC)$ *of the model predictive control problem in this example is the input constraint set* $\mathcal{U}$*. However the model predictive controller* $u_0^\star(x)$ *does not use every possible control input* $u \in \mathcal{U}$*. The image* $u_0^\star(\mathcal{X})$ *of the optimal control for each* $x \in \mathcal{X}$ *is shown in Figure 4.5d. It can be seen that the model predictive controller only uses an octagonal subset of* $\mathcal{U}$*. Therefore the controller* $u_0^\star(x)$ *can have a larger symmetry group* $\mathrm{Aut}(u_0^\star)$ *than the model predictive control problem* $\mathrm{Aut}(MPC)$*. Likewise the symmetry group* $\mathrm{Aut}(u_0^\star)$ *could be strictly larger than* $\mathrm{Aut}(MPC)$ *if the feasible state-space* $\mathcal{X}_0 \subset \mathcal{X}$ *has more symmetries than* $\mathcal{X}$*.*

If the model predictive control problem (4.45) can be posed as a linear or quadratic program then the model predictive controllers $u_0^\star(x)$ are piecewise affine on polytope functions. According to Theorems 7 and 8 the symmetries $(\Theta,\Omega) \in \mathrm{Aut}(MPC)$ of the model predictive control problem permute the pieces $(F_i, G_i, \mathcal{R}_i)$ of the controller $u_0^\star(x)$ when $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ is the minimal p-collection partition. In the next two subsections we will show this holds when $\{\mathcal{R}_i\}_{i\in\mathcal{I}} = \{\mathcal{CR}_i\}_{i\in\mathcal{I}}$ is the critical region partition. This stronger result will be used in Chapter 7 to simplify the explicit model predictive controller.

Linear Model Predictive Control with Quadratic Cost

Consider the following model predictive control problem for a constrained linear time-invariant system with quadratic cost function and polytopic constraints

$$
\begin{aligned}
J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & x_N^T P x_N + \sum\nolimits_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k && \text{(4.59a)}\\
\text{subject to} \quad & x_{k+1} = Ax_k + Bu_k && \text{(4.59b)}\\
& x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} && \text{(4.59c)}\\
& x_N \in \mathcal{X}_N && \text{(4.59d)}\\
& x_0 = x && \text{(4.59e)}
\end{aligned}
$$

where $x \in \mathbb{R}^n$ is the measured state, x_k is the predicted state under the control action $u_k \in \mathbb{R}^m$ over the horizon N, $\mathcal{X}, \mathcal{U}$, and $\mathcal{X}_N$ are polytopic sets, and $P = P^T \succeq 0$, $Q = Q^T \succeq 0$ and $R = R^T \succ 0$. The optimal solution $u_0^\star(x)$ to this model predictive control problem is a piecewise affine on polytopes function.

According to Definition 8 the symmetries $(\Theta, \Omega) \in \text{Aut}(J)$ of the quadratic cost function (4.59a) must satisfy

$$
\begin{aligned}
\Theta^T P \Theta &= P && \text{(4.60a)}\\
\Theta^T Q \Theta &= Q && \text{(4.60b)}\\
\Omega^T R \Omega &= R. && \text{(4.60c)}
\end{aligned}
$$

Often the terminal cost $p(x) = x^T P x$ is chosen to be the cost-to-go $J_\infty^\star(x) = x^T P_\infty x$ for the infinite-horizon linear quadratic regulator problem. In this case the terminal cost $p(x)$ is symmetric with respect to the symmetries of the dynamics $\text{Aut}(A, B)$ and cost matrices Q and R.

Proposition 14. *Let (A, B) be controllable and $(Q^{1/2}, A)$ be observable. Let P_∞ be the solution to the discrete algebraic Riccati equation*

$$
P = Q + A^T P A - A^T P B (R + B^T P B)^{-1} B^T P A. \tag{4.61}
$$

Let $\text{Aut}(Q, R)$ *be the set of all matrices (Θ, Ω) that satisfy $\Theta^T Q \Theta = Q$ and $\Omega^T R \Omega = R$. Then P_∞ is symmetric $\Theta^T P_\infty \Theta = P_\infty$ for all $\Theta \in$* $\text{Aut}(A, B) \cap \text{Aut}(Q, R)$.

Proof. Consider the positive definite matrix $\bar{P} = \Theta^T P_\infty \Theta$ for some $\Theta \in \text{Aut}(A, B) \cap$

$\mathrm{Aut}(Q, R)$. Note that this matrix satisfies the discrete algebraic Riccati equation

$$\begin{aligned} Q + A^T\bar{P}A - A^T\bar{P}B\big(R + B^T\bar{P}B\big)^{-1}B^T\bar{P}A &\qquad (4.62)\\ = Q + A^T\Theta^T P\Theta A - A^T\Theta^T P\Theta B\big(R + B^T\Theta^T P\Theta B\big)^{-1}B^T\Theta^T P\Theta A &\\ = \Theta^T Q\Theta + \Theta^T A^T PA\Theta - \Theta^T A^T PB\Omega\big(\Omega^T R\Omega + \Omega^T B^T PB\Omega\big)^{-1}\Omega B^T PA\Theta &\\ = \Theta^T\Big(Q + A^T PA - A^T PB\big(R + B^T PB\big)^{-1}B^T PA\Big)\Theta &\\ = \Theta^T P\Theta &\\ = \bar{P} & \end{aligned}$$

where $P = P_\infty$. However since (A, B) is controllable and $(Q^{1/2}, A)$ is observable, $P = P_\infty$ is the unique positive definite solution to (4.61). Thus P_∞ is symmetric since $P_\infty = \bar{P} = \Theta^T P_\infty\Theta$ for any $\Theta \in \mathrm{Aut}(A, B) \cap \mathrm{Aut}(Q, R)$. □

This proposition allows us to simplify the definition of the symmetries of the quadratic cost function (4.59a). With this simplification, the symmetry group $\mathrm{Aut}(MPC) = \mathrm{Aut}(J) \cap \mathrm{Aut}(\Sigma)$ of the model predictive control problem (4.59) is the set of all pairs of matrices (Θ, Ω) that satisfy

$$\Theta A = A\Theta \qquad (4.63a)$$
$$\Theta B = B\Omega \qquad (4.63b)$$
$$\Theta\mathcal{X} = \mathcal{X} \qquad (4.63c)$$
$$\Omega\mathcal{U} = \mathcal{U} \qquad (4.63d)$$
$$\Theta^T Q\Theta = Q \qquad (4.63e)$$
$$\Omega^T R\Omega = R \qquad (4.63f)$$

where $\mathcal{X}_N$ is a symmetric set $\mathrm{Aut}(\mathcal{X}_N) \supseteq \mathrm{Aut}(\Sigma)$.

According to Theorem 8 the solution $u_0^\star(x)$ to the model predictive control problem (4.59) is symmetric with respect to this group. The following theorem shows a stronger result: when $\{\mathcal{R}_i\}_{i\in\mathcal{I}} = \{\mathcal{CR}_i\}_{i\in\mathcal{I}}$ is the critical region partition, the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$ permute the pieces $(F_i, G_i, \mathcal{R}_i)$ of the controller $u_0^\star(x)$.

Theorem 9. *Let $u_0^\star(x)$ be the optimal piecewise affine controller law for the model predictive control problem* (4.59) *where $\{\mathcal{R}_i\}_{i\in\mathcal{I}} = \{\mathcal{CR}_i\}_{i\in\mathcal{I}}$ is the critical region partition. Then* $\mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(\mathcal{I})$.

Proof. For each $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$ and critical region $\mathcal{CR}_i$ there exists a critical region $\mathcal{CR}_j$ such that the set $\mathcal{P} = \mathcal{CR}_i \cap \Theta\mathcal{CR}_j$ is full-dimensional since $\{\Theta\mathcal{CR}_i\}_{i\in\mathcal{I}}$ covers $\Theta\mathcal{X}_0 = \mathcal{X}_0$. This set $\mathcal{P}$ contains an affinely independent set of points. Thus the feedback gains of $u_0^\star(x)$ satisfy $F_i\Theta = \Omega F_j$ and the feedforward gains satisfy $G_i = \Omega G_j$ since $u_0^\star(\Theta x) = \Omega u_0^\star(x)$ by Theorem 8.

Next we show $\mathcal{CR}_i = \Theta\mathcal{CR}_j$. The critical region $\mathcal{CR}_i$ is defined as

$$\mathcal{CR}_i = \left\{x : \mathbf{F}_i x + \mathbf{G}_i \in \mathcal{U}^N, (\mathbf{A} + \mathbf{B}\mathbf{F}_i)x + \mathbf{B}\mathbf{G}_j \in \mathcal{X}^N\right\} \tag{4.64}$$

where $\mathbf{A} \in \mathbb{R}^{Nn\times n}$ and $\mathbf{B} \in \mathbb{R}^{Nn\times Nm}$ come from the N-step batch dynamics

$$\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ \vdots \\ A^N \end{bmatrix}}_{\mathbf{A}} x_0 + \underbrace{\begin{bmatrix} B & & \\ \vdots & \ddots & \\ A^{N-1}B & \cdots & B \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{U} \tag{4.65}$$

and $\mathbf{F}_i \in \mathbb{R}^{Nm\times n}$ and $\mathbf{G}_i \in \mathbb{R}^{Nm}$ come from the optimal solution $U^\star(x_0) = \mathbf{F}_i x_0 + \mathbf{G}_i$ of the multi-parametric program (2.28) formulation of the model predictive control problem (4.59).

Since $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma) = \mathrm{Aut}(A, B) \cap \mathrm{Aut}(\mathcal{X}, \mathcal{U})$, the critical region $\mathcal{CR}_j$ satisfies

$$\begin{aligned} \Theta^{-1}\mathcal{CR}_j &= \left\{x : \mathbf{F}_j\Theta x + \mathbf{G}_j \in \mathcal{U}^N, (\mathbf{A} + \mathbf{B}\mathbf{F}_j)\Theta x + \mathbf{B}\mathbf{G}_j \in \mathcal{X}^N\right\} \\ &= \left\{x : (I \otimes \Omega)\mathbf{F}_i x + \mathbf{G}_j \in \mathcal{U}^N, (I \otimes \Theta)(\mathbf{A} + \mathbf{B}\mathbf{F}_i)x + \mathbf{B}\mathbf{G}_j \in \mathcal{X}^N\right\} \\ &= \left\{x : \mathbf{F}_i x + \mathbf{G}_i \in (I \otimes \Omega)^{-1}\mathcal{U}^N, (\mathbf{A} + \mathbf{B}\mathbf{F}_i)x + \mathbf{B}\mathbf{G}_i \in (I \otimes \Theta)^{-1}\mathcal{X}^N\right\} \\ &= \mathcal{CR}_i \end{aligned}$$

where $\otimes$ is the Kronecker product.

Thus each $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$ permutes the controller pieces $(F_i, G_i, \mathcal{R}_i)$ for $i \in \mathcal{I}$ of the controller $u_0^\star(x)$. □

This theorem means that we do not need to redefine the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the controller $u_0^\star(x)$ in order to use symmetry to permute the controller pieces $(F_i, G_i, \mathcal{R}_i)$. This result will be important in Chapter 7 when we use symmetry to reduce the complexity of explicit model predictive controllers.

Linear Model Predictive Control with Linear Cost

Consider the following model predictive control problem for a constrained linear time-invariant system with linear cost function and polytopic constraints

$$\begin{aligned} J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & \Phi_{\mathcal{P}}(x_N) + \sum_{k=0}^{N-1} \Phi_{\mathcal{Q}}(x_k) + \Phi_{\mathcal{R}}(u_k) && (4.66a) \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k && (4.66b) \\ & x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} && (4.66c) \\ & x_0 = x && (4.66d) \end{aligned}$$

where $x \in \mathbb{R}^n$ is the measured state, x_k is the predicted state under the control action $u_k \in \mathbb{R}^m$ over the horizon N, $\mathcal{X}$, $\mathcal{U}$, $\mathcal{X}_N$, $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ are polytopic sets, and $\Phi_{\mathcal{P}}(x)$, $\Phi_{\mathcal{Q}}(x)$, and $\Phi_{\mathcal{R}}(u)$ are the Minkowski functions for the sets $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ respectively. Recall that the Minkowski function for a set $\mathcal{X}$ is defined as $\Phi_{\mathcal{X}}(x) = \inf\{\lambda \in \mathbb{R}_+ : x \in \lambda\mathcal{X}\}$.

According to Definition 8 the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(J)$ of the linear cost function (4.66a) must satisfy

$$\Theta\mathcal{P} = \mathcal{P} \tag{4.67a}$$
$$\Theta\mathcal{Q} = \mathcal{Q} \tag{4.67b}$$
$$\Omega\mathcal{R} = \mathcal{R}. \tag{4.67c}$$

For polytopic sets $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$, the model predictive control problem (4.66) can be posed as a multi-parametric linear program. The optimal solutions $u_0^\star(x)$ to the model predictive control problem (4.66) are piecewise affine on polytopes functions. The value function $J^\star(x)$ is piecewise affine

$$J^\star(x) = \max_{i \in \mathcal{I}} c_i^T x + d_i \tag{4.68}$$

where the critical regions $\mathcal{CR}_i$ are given by [83, 51]

$$\mathcal{CR}_i = \{x \in \mathcal{X}_0 : c_i^T x + d_i \geq c_j^T x + d_j, \forall j \in \mathcal{I}\}. \tag{4.69}$$

The symmetries $\Theta \in \mathrm{Aut}(MPC)$ of the model predictive control problem (4.66) permutes the value function pieces.

Lemma 1. *For each $\Theta \in \mathrm{Aut}(MPC)$ and $i \in \mathcal{I}$ there exists $j \in \mathcal{I}$ such that $\Theta^T c_i = c_j$ and $d_i = d_j$.*

Proof. For each $\Theta \in \mathrm{Aut}(MPC)$ and critical region $\mathcal{CR}_i$ there exists a critical region $\mathcal{CR}_j$ such that $\mathcal{CR}_i \cap \Theta\mathcal{CR}_j$ is full-dimensional since the partition $\{\Theta\mathcal{CR}_i\}_{i \in \mathcal{I}}$ covers $\mathcal{X}_0$ and $\mathcal{CR}_i \subseteq \mathcal{X}_0$. In the critical region $\mathcal{CR}_i$ the value function is $J^\star(x) = c_i^T x + d_i$ and in critical region $\mathcal{CR}_j$ the value function is $J^\star(x) = c_j^T x + d_j$. By Theorem 8 we have

$$J^\star(x) = c_i^T x + d_i = c_j^T \Theta x + d_j = J^\star(\Theta x) \tag{4.70}$$

over the set $\mathcal{CR}_i \cap \Theta\mathcal{CR}_j$. Since $\mathcal{CR}_i \cap \Theta\mathcal{CR}_j$ is full-dimensional it contains an affinely independent set of points x. Thus $\Theta^T c_i = c_j$ and $d_i = d_j$. □

One consequence of this result is that there exists a symmetric piecewise affine on critical regions solution $\bar{u}_0^\star(x)$ to the model predictive control problem (4.66).

Theorem 10. *The model predictive control problem* (4.66) *has a symmetric piecewise affine control-law defined on the critical regions*

$$\bar{u}_0^\star(x) = \begin{cases} \bar{F}_1 x + \bar{G}_1 & x \in \mathcal{CR}_1 \\ \quad\vdots & \\ \bar{F}_1 x + \bar{G}_p & x \in \mathcal{CR}_p. \end{cases} \tag{4.71}$$

Proof. Let $u_0^\star(x)$ be a continuous piecewise affine on polytopes solution to (4.66). Define

$$\bar{u}_0^\star(x) = \frac{1}{|\mathcal{G}|}\sum\nolimits_{(\Theta,\Omega)\in\mathcal{G}} \Omega^{-1} u_0^\star(\Theta x) \tag{4.72}$$

where $\mathcal{G} = \mathrm{Aut}(MPC)$. By Theorem 8, $\bar{u}_0^\star(x)$ is a solution to (4.66). Furthermore it is piecewise affine on polytopes since it is the finite sum of piecewise affine on polytope functions. The partitions of $\bar{u}_0^\star(x)$ are sets of the form $\bigcap_{j=1}^{|\mathcal{G}|} \Theta_j \mathcal{CR}_{i_j}$. Since $\bar{u}_0^\star(x)$ is continuous we are only interested in full-dimensional partitions. By Lemma 1 the sets $\bigcap_{j=1}^{|\mathcal{G}|} \Theta_j \mathcal{CR}_{i_j}$ are only full-dimensional if $i_j = i_1$ for all $j = 1, \dots, |\mathcal{G}|$. Thus the partition of $\bar{u}_0^\star(x)$ is the critical region partition $\{\mathcal{CR}_i\}_{i\in\mathcal{I}}$. $\square$

The importance of this theorem is that it shows there exists a symmetric controller $\bar{u}_0^\star(x)$ with the same number of pieces $|\mathcal{I}|$ as the non-symmetric controllers $u_0^\star(x)$. This means storing the symmetric controller $\bar{u}_0^\star(x)$ requires the same amount of memory as storing a non-symmetric controller $u_0^\star(x)$.

Chapter 5

Identification of Symmetry Groups

In this chapter we consider the problem of finding the symmetry groups of linear system $\text{Aut}(A, B)$, constrained linear system $\text{Aut}(\Sigma)$, and linear model predictive control problems $\text{Aut}(MPC)$ with linear and quadratic costs. This problem is called symmetry identification. For constrained systems and model predictive control problems with bounded full-dimensional sets $\mathcal{X}$ and $\mathcal{U}$ the symmetry groups $\text{Aut}(\Sigma)$ and $\text{Aut}(MPC)$ are finite. Identifying these groups requires techniques from discrete mathematics. We transform the problem of finding the symmetries of a constrained linear system into the problem of finding the symmetries of a vertex colored graph. The symmetries of a vertex colored graph can be found efficiently using graph automorphism software. In the last section we demonstrate our symmetry identification procedure on a quadrotor system.

5.1 Symmetry Identification for Linear Systems

In this section we address the problem of finding the set $\text{Aut}(A, B)$ of all state-space Θ and input-space Ω transformations that preserve the dynamics $\Theta A = A\Theta$ and $\Theta B = B\Omega$. This problem is called symmetry identification for linear systems.

This problem is straight-forward since the closure of the group $\text{Aut}(A, B)$ is a vector-space. The closure of the group $\text{Aut}(A, B)$ is the set of all matrices Θ and Ω that satisfy

$$\begin{bmatrix} \Theta & 0 \\ 0 & \Omega \end{bmatrix} \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Theta & 0 \\ 0 & \Omega \end{bmatrix}. \tag{5.1}$$

The group $\text{Aut}(A, B)$ is the invertible elements of this space.

5.2 Symmetry Identification for Constrained Linear Systems

In this section we address the problem of finding the set $\text{Aut}(\Sigma)$ of all state-space Θ and input-space Ω transformations that preserve the dynamics (4.2) and constraints (4.12).

Recall that a set $\mathcal{S}$ is a generating set for the finite group $\mathcal{G} = \langle \mathcal{S} \rangle$ if every element $\Theta \in \mathcal{G}$ of the group $\mathcal{G}$ can be written as a product $\Theta = S_1 S_2 \cdots$ of elements $S_1, S_2, \ldots. \in \mathcal{S}$ of the set $\mathcal{S}$. The set $\mathcal{S}$ is analogous to a basis for a vector-space. Thus we can describe the group $\mathrm{Aut}(\Sigma)$ by finding a set $\mathcal{S}$ that generates the group $\mathrm{Aut}(\Sigma) = \langle \mathcal{S} \rangle$. We call this problem symmetry identification for constrained linear systems.

Problem 1 (Symmetry Identification for Constrained Linear Systems)**.** Find a set $\mathcal{S}$ of state-space and input-space transformations $(\Theta, \Omega) \in \mathcal{S}$ that generate the symmetry group $\mathrm{Aut}(\Sigma) = \langle \mathcal{S} \rangle$ of the system (4.9) subject to the constraints (4.11).

For bounded full-dimensional polytopes $\mathcal{X}$ and $\mathcal{U}$ the symmetry group $\mathrm{Aut}(\Sigma) \subseteq \mathrm{Aut}(\mathcal{X}, \mathcal{U})$ is finite. Therefore we must use techniques from discrete-mathematics to identify $\mathrm{Aut}(\Sigma)$. In the next section we will describe how to find the symmetries $\mathrm{Aut}(\mathcal{X}, \mathcal{U})$ of polytopes $\mathcal{X}$ and $\mathcal{U}$. We will extend this method to find the symmetry group $\mathrm{Aut}(\Sigma)$ of constrained linear systems.

In order to simplify our analysis we will assume the dynamics matrices A and B, and the constraint sets $\mathcal{X}$ and $\mathcal{U}$ are expressed in the regular basis. Recall that we can transform the state-space and input-space into the regular basis using the state-space transformation $T_x = (\bar{H}_x^T \bar{H}_x)^{-1/2}$ and input-space transformation $T_u = (\bar{H}_u^T \bar{H}_u)^{-1/2}$ where $\bar{H}_x$ and $\bar{H}_u$ are the non-redundant half-space matrices of the constraint sets $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$ in the original basis. Under this regular basis we have $H_x = \bar{H}_x (\bar{H}_x^T \bar{H}_x)^{-1/2}$ and $H_u = \bar{H}_u (\bar{H}_u^T \bar{H}_u)^{-1/2}$ satisfy $H_x^T H_x = I_n$ and $H_x^T H_x = I_m$. The dynamics matrices are

$$A = (\bar{H}_x^T \bar{H}_x)^{1/2} \bar{A} (\bar{H}_x^T \bar{H}_x)^{-1/2} \tag{5.2a}$$

$$B = (\bar{H}_x^T \bar{H}_x)^{1/2} \bar{B} (\bar{H}_u^T \bar{H}_u)^{-1/2} \tag{5.2b}$$

where $\bar{A}$ and $\bar{B}$ are the dynamics matrices in the original basis.

We are searching for the set $\mathrm{Aut}(\Sigma)$ of matrices Θ and Ω that preserve the dynamics

$$\Theta A = A \Theta \tag{5.3a}$$

$$\Theta B = B \Omega \tag{5.3b}$$

and the constraints

$$\Theta \mathcal{X} = \mathcal{X} \tag{5.4a}$$

$$\Omega \mathcal{U} = \mathcal{U}. \tag{5.4b}$$

These matrix pairs $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ will be represented in the regular basis and must be converted back to the original basis using the transformations

$$\bar{\Theta} = (H_x^T H_x)^{-1/2} \Theta (H_x^T H_x)^{1/2} \tag{5.5a}$$

$$\bar{\Omega} = (H_x^T H_x)^{-1/2} \Omega (H_x^T H_x)^{1/2}. \tag{5.5b}$$

Identifying the Symmetries of Polytopes

In this section we describe how to find the symmetry group $\mathrm{Aut}(\mathcal{X},\mathcal{U}) = \mathrm{Aut}(\mathcal{X}) \times \mathrm{Aut}(\mathcal{U})$ of polytopes $\mathcal{X}$ and $\mathcal{U}$. We assume the polytopes $\mathcal{X}$ and $\mathcal{U}$ are bounded, full-dimensional, and contain the origin in their interiors.

For bounded, full-dimensional sets $\mathcal{X}$ and $\mathcal{U}$ the symmetry group $\mathrm{Aut}(\mathcal{X},\mathcal{U}) = \mathrm{Aut}(\mathcal{X}) \times \mathrm{Aut}(\mathcal{U})$ is isomorphic to a permutation group acting on the facets of $\mathcal{X}$ and $\mathcal{U}$. We will denote this group by $\mathrm{Perm}(\mathcal{X},\mathcal{U})$. The symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{X},\mathcal{U})$ and the permutation matrix $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\mathcal{X},\mathcal{U})$ are related by the expression

$$\begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix} \begin{bmatrix} H_x & 0 \\ 0 & H_u \end{bmatrix} = \begin{bmatrix} H_x & 0 \\ 0 & H_u \end{bmatrix} \begin{bmatrix} \Theta & 0 \\ 0 & \Omega \end{bmatrix} \tag{5.6}$$

where $\Pi_x \in \mathbb{R}^{c_x \times c_x}$ and $\Pi_u \in \mathbb{R}^{c_u \times c_u}$ are permutation matrices. Taking the pseudo-inverse of H_x and H_u we can express Θ and Ω in terms of Π_x and Π_u by

$$\Theta = H_x^T \Pi_x H_x \tag{5.7a}$$

$$\Omega = H_u^T \Pi_u H_u \tag{5.7b}$$

where $H_x^T H_x = I_n$ and $H_u^T H_u = I_m$ under the regular basis. In [18] it was shown that (5.7) is a group isomorphism from $\mathrm{Perm}(\mathcal{X},\mathcal{U})$ to $\mathrm{Aut}(\mathcal{X},\mathcal{U})$. Therefore the problem of identify the symmetry group $\mathrm{Aut}(\mathcal{X},\mathcal{U})$ is equivalent to finding the group of permutation matrices $\mathrm{Perm}(\mathcal{X},\mathcal{U})$.

In [18] the permutation group $\mathrm{Perm}(\mathcal{X},\mathcal{U})$ was characterized as the set of all permutation matrices Π_x and Π_u that satisfy the commutative relation

$$\begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix} \begin{bmatrix} H_x H_x^T & 0 \\ 0 & H_u H_u^T \end{bmatrix} = \begin{bmatrix} H_x H_x^T & 0 \\ 0 & H_u H_u^T \end{bmatrix} \begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix}. \tag{5.8}$$

In Section 5.4 we will show how to convert the problem of finding all permutation matrices Π_x and Π_u that satisfy (5.8) into a graph automorphism problem.

Identifying the Constrained Linear Systems

In this section we extend the method presented in the previous section for identifying the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{X},\mathcal{U})$ of polytopes $\mathcal{X}$ and $\mathcal{U}$ to the problem of identifying the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of constrained linear systems.

For the matrices $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{X},\mathcal{U})$ to be elements of the symmetry group $\mathrm{Aut}(\Sigma) \subseteq \mathrm{Aut}(\mathcal{X},\mathcal{U})$ they must preserve (5.3) the dynamics matrices A and B. This suggests that we use Algorithm 7 for computing the symmetry group $\mathrm{Aut}(\Sigma)$.

Unfortunately Algorithm 7 is impractical since the group $\mathrm{Aut}(\mathcal{X},\mathcal{U})$ can be very large $|\mathrm{Aut}(\mathcal{X},\mathcal{U})| = O(c_x^n c_u^m)$. In order to be computationally tractable we need to design an algorithm that will operate on the generators $\mathcal{S}$ of the group $\mathrm{Aut}(\mathcal{X},\mathcal{U}) = \langle \mathcal{S} \rangle$. The generators are bounded in size by $|\mathcal{S}| \leq (c_x - 1)(c_u - 1)$. Unfortunately in general Algorithm 7 does not

Algorithm 7 Naive calculation of Aut(Σ)

```
1: Calculate $\text{Aut}(\mathcal{X},\mathcal{U})$
2: for each $(\Theta,\Omega) \in \text{Aut}(\mathcal{X},\mathcal{U})$ do
3:     if $\Theta\mathbf{A} = \mathbf{A}\Theta$ and $\Theta\mathbf{B} = \mathbf{B}\Omega$ then
4:         Add $(\Theta,\Omega)$ to $\text{Aut}(\Sigma)$.
5:     end if
6: end for
```

produce the full symmetry group $\text{Aut}(\Sigma)$ when the for-loop is restricted to the generators $\mathcal{S}$ of $\text{Aut}(\mathcal{X},\mathcal{U}) = \langle \mathcal{S} \rangle$. This is demonstrated by the following example.

Example 13. *Consider the constrained autonomous system*

$$x(t+1) = Ax(t) \tag{5.9a}$$

$$x(t) \in \mathcal{X} \tag{5.9b}$$

where $\mathcal{X}$ is the square $\mathcal{X} = \{x \in \mathbb{R}^2 : Hx \leq 1\}$ with

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{ and } H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \tag{5.10}$$

where $H^T H = I$ since we are in the regular basis.

The symmetry group $\text{Aut}(\mathcal{X})$ *of the square $\mathcal{X}$ is the dihedral-4 group; the eight element set of all rotations by increments of* 90 *degrees and the reflections about the horizontal, vertical, and both diagonal axes. The group* $\text{Aut}(\mathcal{X})$ *can be generated by the horizontal and diagonal reflections*

$$\Theta_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \Theta_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{5.11}$$

The symmetry group $\text{Aut}(\Sigma)$ *contains four elements; the rotations by increments of* 90 *degrees. However executing Algorithm 7 using the generators $\mathcal{S} = \{\Theta_1, \Theta_2\}$ will not return* $\text{Aut}(\Sigma)$ *since neither of the generators Θ_1 or Θ_2 commute with the matrix A.*

Our method for finding $\text{Aut}(\Sigma)$ is similar to the method for finding $\text{Aut}(\mathcal{X},\mathcal{U})$. We find a permutation group $\text{Perm}(\Sigma)$ isomorphic to $\text{Aut}(\Sigma)$. The generators of $\text{Aut}(\Sigma)$ are then the image of the generators of $\text{Perm}(\Sigma)$ mapped through the isomorphism (5.7).

Since $\text{Aut}(\Sigma) \subseteq \text{Aut}(\mathcal{X},\mathcal{U})$ is a subgroup of $\text{Aut}(\mathcal{X},\mathcal{U})$ it is also isomorphic to a permutation group $\text{Perm}(\Sigma) \subseteq \text{Perm}(\mathcal{X},\mathcal{U})$. The permutations $(\Pi_x, \Pi_u) \in \text{Perm}(\Sigma)$ must not only permute the facets of $\mathcal{X}$ and $\mathcal{U}$ but also the facets of the pre-set $\mathrm{Pre}(\mathcal{X})$ of the set $\mathcal{X}$ under the linear dynamics (4.9) subject to the constraints (4.11). Therefore the permutations $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ and linear transformations $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ must satisfy

$$\begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix} \begin{bmatrix} H_x A & H_x B \\ 0 & H_u \end{bmatrix} = \begin{bmatrix} H_x A & H_x B \\ 0 & H_u \end{bmatrix} \begin{bmatrix} \Theta & 0 \\ 0 & \Omega \end{bmatrix}. \tag{5.12}$$

This expression allows us to characterize $\mathrm{Perm}(\Sigma) \subseteq \mathrm{Perm}(\mathcal{X},\mathcal{U})$ as the set of all permutation matrices $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\mathcal{X},\mathcal{U})$ from the permutation group $\mathrm{Perm}(\mathcal{X},\mathcal{U})$ that satisfy the commutative relation

$$\begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix} \begin{bmatrix} H_x A H_x^T & H_x B H_u^T \\ 0 & H_u^T H_u \end{bmatrix} = \begin{bmatrix} H_x A H_x^T & H_x B H_u^T \\ 0 & H_u^T H_u \end{bmatrix} \begin{bmatrix} \Pi_x & 0 \\ 0 & \Pi_u \end{bmatrix} \tag{5.13}$$

where $H_x^T H_x = I_n$ and $H_u^T H_u = I_m$ under the regular basis. The following theorem shows that (5.13) ensures the linear transformations $\Theta = H_x^T \Pi_x H_x$ and $\Omega = H_u^T \Pi_u H_u$ preserve the dynamics (5.3) and constraints (5.4).

Theorem 11. *Let* $\mathrm{Perm}(\Sigma)$ *be the set of all permutation matrices* Π_x *and* Π_u *that satisfy* (5.8) *and* (5.13). *Then* $\mathrm{Perm}(\Sigma)$ *is isomorphic to* $\mathrm{Aut}(\Sigma)$ *with isomorphism (5.7).*

Proof. In order to prove this theorem we must prove three claims: the matrices Θ and Ω given by (5.7) for $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ are symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of the linear system (4.1) subject to the constraints (4.11), the function (5.7) is a group homomorphism, and it is bijective.

First we prove that the transformations Θ and Ω produced by (5.7) are symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of the constrained linear system. First we show Θ and Ω satisfy (5.3). By (5.13) the permutation matrices $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ satisfy

$$\Pi_x H_x B H_u^T = H_x B H_u^T \Pi_u. \tag{5.14}$$

We pre-multiply this expression by H_x^T and post-multiply by H_u to produce

$$H_x^T \Pi_x H_x B H_u^T H_u = H_x^T H_x B H_u^T \Pi_u H_u. \tag{5.15}$$

Since $H_x^T H_x = I_n$ and $H_u^T H_u = I_m$ under the regular basis, we have

$$\Theta B = B \Omega \tag{5.16}$$

where $\Theta = H_x^T \Pi_x H_x$ and $\Omega = H_u^T \Pi_u H_u$. Likewise we can show $\Theta A = A \Theta$. Thus $(\Theta, \Omega) \in \mathrm{Aut}(A, B) \supseteq \mathrm{Aut}(\Sigma)$.

Next we show that Θ and Ω satisfy (5.4). By (5.8) we have

$$\begin{aligned} H_x \Theta &= H_x H_x^T \Pi_x H_x \\ &= \Pi_x H_x H_x^T H_x \\ &= \Pi_x H_x \end{aligned} \tag{5.17}$$

where $H_x^T H_x = I_n$. Thus Θ permutes the rows of H_x. Since H_x is non-redundant this means Θ permutes the facets of the set $\mathcal{X} = \{x : H_x x \leq 1\}$ which preserves the set $\mathcal{X} = \Theta\mathcal{X}$. Likewise we can show Ω preserves the set $\mathcal{U} = \Omega\mathcal{U}$. Thus $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma) = \mathrm{Aut}(A, B) \cap \mathrm{Aut}(\mathcal{X}, \mathcal{U})$.

Now we show that (5.7) is a group homomorphism from $\mathrm{Perm}(\Sigma)$ to $\mathrm{Aut}(\Sigma)$. This follows from the expression

$$\begin{aligned}\Theta(\Pi_x^1)\Theta(\Pi_x^2) &= H_x^T \Pi_x^1 H_x H_x^T \Pi_x^2 H_x \\ &= H_x^T \Pi_x^1 \Pi_x^2 H_x H_x^T H_x \\ &= H_x^T \Pi_x^1 \Pi_x^2 H_x \\ &= \Theta(\Pi_x^1 \Pi_x^2)\end{aligned} \tag{5.18}$$

where $H_x^T H_x = I_n$ and the second equality follows from (5.8). Likewise Ω is a group homomorphism.

Finally we show that (5.7) is bijective. Since the sets $\mathcal{X}$ and $\mathcal{U}$ are bounded and full-dimensional the matrices $H_x \in \mathbb{R}^{c_x \times n}$ and $H_u \in \mathbb{R}^{c_u \times m}$ are full row-rank. Therefore the pseudo-inverse expressions $\Theta = H_x^T \Pi_x H_x$ and $\Omega = H_u^T \Pi_u H_u$ are the unique solutions to $H_x \Theta = \Pi_x H_x$ and $H_u \Omega = \Pi_u H_u$ where $H_x^T H_x = I_n$ and $H_u^T H_u = I_m$. Therefore (5.7) is injective.

To prove that (5.7) is surjective we need to show that for every $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ there exists a $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ such that $\Theta = H_x^T \Pi_x H_x$ and $\Omega = H_u^T \Pi_u H_u$. By definition of $\mathrm{Aut}(\Sigma)$ there exists permutation matrices Π_x and Π_u such that $H_x \Theta = \Pi_x H_x$ and $H_u \Omega = \Pi_u H_u$ since $\Theta\mathcal{X} = \mathcal{X}$ and $\Omega\mathcal{U} = \mathcal{U}$, and H_x and H_u are non-redundant. We need to show Π_x and Π_u satisfy (5.8) and (5.13) i.e. $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$. Note

$$\begin{aligned}\Pi_x H_x B H_u^T \Pi_u^T &= H_x \Theta B \Omega^T H_u^T \\ &= H_x B \Omega \Omega^T H_u^T \\ &= H_x B \Omega (H_u^T H_u)^{-1} \Omega^T H_u^T \\ &= H_x B (\Omega^{-T} H_u^T H_u \Omega^{-1})^{-1} H_u^T\end{aligned} \tag{5.19}$$

where $(H_u^T H_u)^{-1} = I_m$ and $\Theta B = B\Omega$ since $\Omega \in \mathrm{Aut}(\Sigma)$. Since $\Omega^{-1} \in \mathrm{Aut}(\Sigma)$ there exists a permutation matrix Q such that $H_u \Omega^{-1} = Q H_u$. Thus

$$\begin{aligned}\Pi_x H_x B H_u^T \Pi_u^T &= H_x B (\Omega^{-T} H_u^T H_u \Omega^{-1})^{-1} H_u^T \\ &= H_x B (H_u^T Q^T Q H_u)^{-1} H_u^T \\ &= H_x B (H_u^T H_u)^{-1} H_u^T \\ &= H_x B H_u^T\end{aligned} \tag{5.20}$$

where $Q^T Q = I_{c_u}$ by the orthogonality of permutation matrices. Therefore Π_x and Π_u satisfy $\Pi_x H_x B H_u^T = H_x B H_u^T \Pi_u$. Similarly we can show Π_x and Π_u satisfy each of the commutative relations in (5.8) and (5.13). Therefore $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$.

We conclude that $\mathrm{Aut}(\Sigma)$ and $\mathrm{Perm}(\Sigma)$ are isomorphic with isomorphism (5.7). $\square$

Remark 1. *In [18] the authors proved this results for* $\mathrm{Aut}(\mathcal{X},\mathcal{U})$ *and* $\mathrm{Perm}(\mathcal{X},\mathcal{U})$ *using graph theory. In Theorem 11 we have re-proven their result using linear algebra and extended it to constrained linear systems.*

Our procedure for identifying the symmetry group $\mathrm{Aut}(\Sigma)$ can be summarized by Procedure 1. In the Section 5.4 we will describe the procedure for finding the group $\mathrm{Perm}(\Sigma)$.

Procedure 1 Calculation of symmetry group $\mathrm{Aut}(\Sigma)$ of constrained linear system Σ

1: Regularize the constrained linear system using the state-space and input-space transformations $T_x = (\bar{H}_x^T \bar{H}_x)^{-1/2}$ and $T_u = (\bar{H}_u^T \bar{H}_u)^{-1/2}$

$$\begin{array}{c} A = T_x^{-1}\bar{A}T_x \\ B = T_x^{-1}\bar{B}T_u \end{array} \quad \text{and} \quad \begin{array}{c} H_x = \bar{H}_x T_x \\ H_u = \bar{H}_u T_u \end{array}$$

where $\bar{A}$ and $\bar{B}$ are the dynamics matrices and $\bar{H}_x$ and $\bar{H}_u$ are the half-space matrices under the original basis.

2: Compute the matrices M_1 and M_2 from (5.8) and (5.13)

$$M_1 = \begin{bmatrix} H_x H_x^T & 0 \\ 0 & H_u H_u^T \end{bmatrix} \text{ and } M_1 = \begin{bmatrix} H_x A H_x^T & H_x B H_u^T \\ 0 & H_u H_u^T \end{bmatrix}.$$

3: Find a set of generators $\mathcal{P}$ for the group $\mathrm{Perm}(\Sigma) = \langle \mathcal{P} \rangle$ of permutation matrices $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ that satisfy (5.8) and (5.13) i.e. commute with M_1 and M_2.

4: Calculate the generators $\mathcal{S}$ for the symmetry group $\mathrm{Aut}(\Sigma) = \langle \mathcal{S} \rangle$ from $\mathcal{P}$ using the isomorphism (5.7)

$$\begin{aligned} \Theta &= H_x^T \Pi_x H_x \\ \Omega &= H_u^T \Pi_u H_u. \end{aligned}$$

5: Convert Θ and Ω to the original basis using the transformations $T_x^{-1} = (H_x^T H_x)^{1/2}$ and $T_u^{-1} = (H_u^T H_u)^{1/2}$

$$\begin{aligned} \bar{\Theta} &= T_x \Theta T_x^{-1} \\ \bar{\Omega} &= T_u \Omega T_u^{-1}. \end{aligned}$$

Before proceeding we prove one useful corollary to Theorem 11.

Corollary 3. *Under the regular basis, the symmetries* $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ *of the linear system* (4.9) *subject to the constraints* (4.11) *are orthogonal* $\Theta^{-1} = \Theta^T$ *and* $\Omega^{-1} = \Omega^T$.

Proof. By isomorphism (5.7) we have

$$\begin{aligned}\Theta^T\Theta &= H_x^T\Pi_x H_x H_x^T \Pi_x^T H_x \\ &= H_x^T H_x H_x^T \Pi_x \Pi_x^T H_x = I\end{aligned} \tag{5.21}$$

since $\Pi_x H_x H_x^T = H_x H_x^T \Pi_x$ by (5.8), $\Pi_x\Pi_x^T = I$, and $H_x^T H_x = I$ under the regular basis. Likewise we can show $\Omega^T\Omega = I$. □

5.3 Symmetry Identification for Model Predictive Control Problems

The symmetry group $\text{Aut}(MPC) = \text{Aut}(\Sigma) \cap \text{Aut}(J) \cap \text{Aut}(\mathcal{X}_N)$ is the intersection of the symmetry groups of the cost $\text{Aut}(J)$, the constrained linear system $\text{Aut}(\Sigma)$, and terminal set $\mathcal{X}_N$. We will assume the terminal set $\mathcal{X}_N$ is symmetric with respect to the symmetry group $\text{Aut}(\Sigma)$ of the constrained linear system. In the previous section we showed how to find generators $\mathcal{S}$ of the symmetry group $\text{Aut}(\Sigma) = \langle \mathcal{S} \rangle$ of a constrained linear system. There are two ways we can address the symmetry of the cost function: we can identify the symmetries $\text{Aut}(J)$ of the cost function J or we can modify the cost function J so that it is symmetric with respect to the symmetry group $\text{Aut}(\Sigma)$ of the dynamics and constraints.

Consider the quadratic cost function

$$J(X,U) = x_N^T P x_N + \sum\nolimits_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \tag{5.22}$$

where $U = \{u_k\}_{k=0}^{N-1}$ is the input trajectory and $X = \{x_k\}_{k=0}^{N}$ is the state trajectory. Recall that (Θ, Ω) is a symmetry of (5.22) if they satisfy

$$\Theta^T P \Theta = P \tag{5.23a}$$

$$\Theta^T Q \Theta = Q \tag{5.23b}$$

$$\Omega^T R \Omega = R. \tag{5.23c}$$

The symmetry group $\text{Aut(MPC)} = \text{Aut}(\Sigma) \cap \text{Aut}(J)$ can be identified by finding the set of all permutations $(\Pi_x, \Pi_u) \in \text{Perm}(\Sigma)$ in the symmetry group $\text{Perm}(\Sigma)$ that satisfy the commutative relations

$$\begin{bmatrix}\Pi_x & 0 \\ 0 & \Pi_u\end{bmatrix}\begin{bmatrix}H_xQH_x^T & 0 \\ 0 & H_uRH_u^T\end{bmatrix} = \begin{bmatrix}H_xQH_x^T & 0 \\ 0 & H_uRH_u^T\end{bmatrix}\begin{bmatrix}\Pi_x & 0 \\ 0 & \Pi_u\end{bmatrix} \tag{5.24}$$

and

$$\begin{bmatrix}\Pi_x & 0 \\ 0 & \Pi_u\end{bmatrix}\begin{bmatrix}H_xPH_x^T & 0 \\ 0 & 0\end{bmatrix} = \begin{bmatrix}H_xPH_x^T & 0 \\ 0 & 0\end{bmatrix}\begin{bmatrix}\Pi_x & 0 \\ 0 & \Pi_u\end{bmatrix}. \tag{5.25}$$

Corollary 4. *Let* $\mathrm{Perm}(MPC)$ *be the set of all permutation matrices* $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ *in* $\mathrm{Perm}(\Sigma)$ *that satisfy* (5.24) *and* (5.25). *Then* $\mathrm{Perm}(MPC)$ *is isomorphic to* $\mathrm{Aut}(MPC)$ *with isomorphism* (5.7).

Proof. Let $\mathrm{Perm}(J)$ be the set of permutation matrices Π_x and Π_u that satisfy (5.24) and (5.25). It can be easily verified that $\mathrm{Perm}(J)$ is a group. Therefore $\mathrm{Perm}(MPC) = \mathrm{Perm}(\Sigma) \cap \mathrm{Perm}(J)$ is a group.

Next we prove that for each $(\Pi_x, \Pi_u) \in \mathrm{Perm}(MPC)$ the matrices $\Theta = H_x^T \Pi_x H_x$ and $\Omega = H_u^T \Pi_u H_u$ satisfy (5.23). Note

$$\begin{aligned}\Theta^T P \Theta &= H_x^T \Pi_x^T H_x P H_x^T \Pi_x H_x \\ &= H_x^T H_x P H_x^T \Pi_x^T \Pi_x H_x \\ &= P. \end{aligned} \tag{5.26}$$

Likewise for Q and R. Therefore (5.7) is an injection from $\mathrm{Perm}(MPC) = \mathrm{Perm}(\Sigma) \cap \mathrm{Perm}(J)$ to $\mathrm{Aut}(MPC) = \mathrm{Aut}(\Sigma) \cap \mathrm{Aut}(J)$.

Finally we show (5.7) is surjective. Let $(\Theta, \Omega) \in \mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(\Sigma)$ and $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ be the corresponding permutation matrices. Then

$$\begin{aligned}\Pi_x H_x P H_x^T \Pi_x^T &= H_x^T \Theta^T P \Theta H_x^T \\ &= H_x^T P H_x \end{aligned} \tag{5.27}$$

where $\Pi H_x = H_x \Theta$ and $\Theta^T P \Theta = P$ since $\Theta \in \mathrm{Aut}(MPC)$. Thus Π_x and Π_u satisfy (5.25). Likewise we can show Π_x and Π_u satisfy (5.24). Thus $(\Pi_x, \Pi_u) \in \mathrm{Perm}(MPC) = \mathrm{Perm}(\Sigma) \cap \mathrm{Perm}(J)$. □

Alternatively we can modify the cost function of the model predictive control problem so that it shares the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of the system. The quadratic cost function (5.22) can be made symmetric with respect to $\mathrm{Aut}(\Sigma)$ by taking its barycenter

$$\bar{J}(X, U) = x_N^T \bar{P} x_N + \sum\nolimits_{k=0}^{N-1} x_k^T \bar{Q} x_k + u_k^T \bar{R} u_k \tag{5.28}$$

where

$$\bar{P} = \frac{1}{|\mathcal{G}|} \sum_{\Theta \in \mathcal{G}} \Theta^T P \Theta \tag{5.29a}$$

$$\bar{Q} = \frac{1}{|\mathcal{G}|} \sum_{\Theta \in \mathcal{G}} \Theta^T Q \Theta \tag{5.29b}$$

$$\bar{R} = \frac{1}{|\mathcal{G}|} \sum_{\Omega \in \mathcal{G}} \Omega^T R \Omega \tag{5.29c}$$

and $\mathcal{G} = \mathrm{Aut}(\Sigma)$.

Proposition 15. *The cost function* (5.29) *is symmetric with respect to the symmetries* $\mathrm{Aut}(\Sigma)$ *of the dynamics and constraints.*

Proof. From the closure of groups we have $\Theta^T \bar{P} \Theta = \bar{P}$, $\Theta^T \bar{Q} \Theta = \bar{Q}$, and $\Omega^T \bar{R} \Omega = \bar{R}$ for all $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$. Therefore $\bar{J}\big((I \otimes \Theta)X, (I \otimes \Omega)U\big) = \bar{J}(X, U)$. □

Now consider the linear cost function

$$J(X,U) = \Phi_{\mathcal{P}}(x_N) + \sum\nolimits_{k=0}^{N-1} \Phi_{\mathcal{Q}}(x_k) + \Phi_{\mathcal{R}}(u_k) \tag{5.30}$$

where $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ are bounded, full-dimensional polytopes containing the origin in their interior. Recall that the pair (Θ, Ω) is a symmetry of the cost (5.30) if it satisfies

$$\Theta\mathcal{P} = \mathcal{P} \tag{5.31a}$$
$$\Theta\mathcal{Q} = \mathcal{Q} \tag{5.31b}$$
$$\Omega\mathcal{R} = \mathcal{R}. \tag{5.31c}$$

The symmetry group $\mathrm{Aut}(J)$ can be identified by finding permutation matrices Π_p, Π_q, and Π_r that satisfy the commutative relationship

$$\begin{aligned} &\begin{bmatrix} 0 & 0 & H_x H_p^T & H_x H_q^T & 0 \\ 0 & 0 & 0 & 0 & H_u H_r^T \\ H_p H_x^T & 0 & 0 & 0 & 0 \\ H_q H_x^T & 0 & 0 & 0 & 0 \\ 0 & H_r H_u^T & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Pi_x & 0 & 0 & 0 & 0 \\ 0 & \Pi_u & 0 & 0 & 0 \\ 0 & 0 & \Pi_p & 0 & 0 \\ 0 & 0 & 0 & \Pi_q & 0 \\ 0 & 0 & 0 & 0 & \Pi_r \end{bmatrix} \\ &\quad = \begin{bmatrix} \Pi_x & 0 & 0 & 0 & 0 \\ 0 & \Pi_u & 0 & 0 & 0 \\ 0 & 0 & \Pi_p & 0 & 0 \\ 0 & 0 & 0 & \Pi_q & 0 \\ 0 & 0 & 0 & 0 & \Pi_r \end{bmatrix} \begin{bmatrix} 0 & 0 & H_x H_p^T & H_x H_q^T & 0 \\ 0 & 0 & 0 & 0 & H_u H_r^T \\ H_p H_x^T & 0 & 0 & 0 & 0 \\ H_q H_x^T & 0 & 0 & 0 & 0 \\ 0 & H_r H_u^T & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{5.32}$$

where H_p, H_q, and H_r are the non-redundant half-space matrices of $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$ respectively.

Corollary 5. *Let* $\mathrm{Perm}(MPC)$ *be the set of all permutation matrices* $(\Pi_x, \Pi_u, \Pi_p, \Pi_q, \Pi_r)$ *that satisfy* (5.32) *with* $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$*. Then* $\mathrm{Perm}(MPC)$ *is isomorphic to* $\mathrm{Aut}(MPC)$ *with isomorphism* (5.7)*.*

Proof. First we show that each $(\Pi_x, \Pi_u, \Pi_p, \Pi_q, \Pi_r) \in \mathrm{Perm}(MPC)$ produces a matrix pair $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$. Since $(\Pi_x, \Pi_u) \in \mathrm{Perm}(\Sigma)$ we have $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$. We need to show that $(\Theta, \Omega) \in \mathrm{Aut}(J)$ and thus $(\Theta, \Omega) \in \mathrm{Aut}(MPC) = \mathrm{Aut}(J) \cap \mathrm{Aut}(\Sigma)$.

The matrices Π_p and Π_x satisfy

$$\Pi_q H_p H_x^T = H_p H_x^T \Pi_x. \tag{5.33}$$

Post-multiplying this expression by H_x produces $\Pi_p H_p = H_p \Theta$. Likewise we have $\Pi_q H_q = H_q \Theta$ and $\Pi_r H_r = H_r \Omega$. Thus Θ and Ω satisfy $\Theta\mathcal{P} = \mathcal{P}$, $\Theta\mathcal{Q} = \mathcal{Q}$, and $\Omega\mathcal{R} = \mathcal{R}$. Therefore $(\Theta, \Omega) \in \mathrm{Aut}(J)$.

Next we show that each $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$ corresponds to a set of permutation matrices $(\Pi_x, \Pi_u, \Pi_p, \Pi_q, \Pi_r) \in \mathrm{Perm}(MPC)$. Since $(\Pi_x, \Pi_u) \in \mathrm{Aut}(\Sigma)$ we only need to show $(\Pi_x, \Pi_u, \Pi_p, \Pi_q, \Pi_r) \in \mathrm{Perm}(J)$. Since $\Theta \in \mathrm{Aut}(\Sigma) \supseteq \mathrm{Aut}(MPC)$ there exists Π_x such that $\Theta = H_x^T \Pi_x H_x$. Since $\Theta \in \mathrm{Aut}(J) \supseteq \mathrm{Aut}(MPC)$ there exists a permutation matrix Π_p that describes the permutation the facets of $\mathcal{P} = \{x : H_p x \leq \mathbf{1}\}$ by Θ. Thus

$$\Pi_p H_p = H_p \Theta = H_p H_x^T \Pi_x H_x \tag{5.34}$$

since H_p is non-redundant. Post-multiplying this expression by H_x^T we obtain

$$\Pi_p H_p H_x^T = H_p H_x^T \Pi_x \tag{5.35}$$

since $H_x^T \Pi_x H_x H_x^T = H_x^T H_x H_x^T \Pi_x$ by (5.8) and $H_x^T H_x = I$. Likewise we can show Π_x, Π_u, Π_p, Π_q, Π_r satisfies all the commutative relationships in (5.32). Therefore $(\Pi_x, \Pi_u, \Pi_p, \Pi_q, \Pi_r)$ is an element of $\mathrm{Perm}(MPC)$. □

Alternatively we can modify the cost function of the model predictive control problem so that is shares the symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of the system. The linear cost function (5.30) can be made symmetric with respect to $\mathrm{Aut}(\Sigma)$ by taking the barycenter of the sets $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{R}$

$$\bar{\mathcal{P}} = \bigcap\nolimits_{\Theta \in \mathcal{G}} \Theta\mathcal{P} \tag{5.36a}$$

$$\bar{\mathcal{Q}} = \bigcap\nolimits_{\Theta \in \mathcal{G}} \Theta\mathcal{Q} \tag{5.36b}$$

$$\bar{\mathcal{R}} = \bigcap\nolimits_{\Omega \in \mathcal{G}} \Theta\mathcal{R} \tag{5.36c}$$

where $\mathcal{G} = \mathrm{Aut}(\Sigma)$. The following linear cost is symmetric with respect to $\mathrm{Aut}(\Sigma)$

$$\bar{J}(X, U) = \Phi_{\bar{\mathcal{P}}}(x_N) + \sum\nolimits_{k=0}^{N-1} \Phi_{\bar{\mathcal{Q}}}(x_k) + \Phi_{\bar{\mathcal{R}}}(u_k). \tag{5.37}$$

Proposition 16. *The cost function* (5.36) *is symmetric with respect to the symmetries* $\mathrm{Aut}(\Sigma)$ *of the dynamics and constraints.*

Proof. From the closure of groups we have $\Theta\bar{\mathcal{P}} = \bar{\mathcal{P}}$, $\Theta\bar{\mathcal{Q}} = \bar{\mathcal{Q}}$, and $\Theta\bar{\mathcal{R}} = \bar{\mathcal{R}}$ for all $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$. Therefore $\bar{J}\big((I \otimes \Theta)X, (I \otimes \Omega)U\big) = \bar{J}(X, U)$. □

5.4 Matrix Permutation Groups

In the previous sections we transformed the symmetry identification problems into the problem of finding the group of permutation matrices that commute with a set of matrices $\{M_1, \ldots, M_r\}$. In this section we reformulate this problem as a graph automorphism problem. Graph automorphism groups can be identified efficiently using software packages such as *Nauty* [63], *Saucy* [33], *Bliss* [53], and *GAP* [41].

Permutations that Commute with a Single Matrix

We begin by presenting the procedure for finding the set $\mathrm{Perm}(M)$ of all permutation matrices $\Pi \in \mathbb{R}^{n\times n}$ that commute with a single matrix $M \in \mathbb{R}^{n\times n}$

$$\Pi M = M\Pi. \tag{5.38}$$

This problem was first addressed in [63]. We identify the group $\mathrm{Perm}(M)$ by constructing a graph $\Gamma(M)$ whose symmetry group $\mathrm{Aut}(\Gamma(M))$ is isomorphic to the permutation group $\mathrm{Perm}(M)$. The generators of $\mathrm{Aut}(\Gamma(M))$ can then be mapped to generators for $\mathrm{Perm}(M)$. Recall that a graph symmetry $\pi : \mathcal{V} \to \mathcal{V}$ is a permutation of the vertex set $\mathcal{V}$ of the graph $\Gamma = (\mathcal{V}, \mathcal{E})$ that preserves the edges list $\pi(\mathcal{E}) = \mathcal{E}$ and vertex coloring $C(\pi(v)) = C(v)$ for all $v \in \mathcal{V}$. The group $\mathrm{Aut}(\Gamma)$ is the set of all symmetries π of the graph Γ.

Procedure 2 produces a vertex colored graph $\Gamma(M)$ whose symmetry group $\mathrm{Aut}(\Gamma(M))$ is isomorphic to $\mathrm{Perm}(M)$. The generators of the group $\mathrm{Aut}(\Gamma(M))$ are found using graph automorphism software [63, 46, 33, 53, 41].

Procedure 2 Create graph $\Gamma(M)$ whose symmetry group $\mathrm{Aut}(\Gamma(M))$ is isomorphic to the permutation group $\mathrm{Perm}(M)$

.

```
Input: Square matrix $M \in \mathbb{R}^{n\times n}$
Output: Vertex colored graph $\Gamma(M) = (\mathcal{V}, \mathcal{E})$
1: Construct the vertex set $\mathcal{V}$:
        add a vertex $r_i$ for $i = 1, \dots, n$ of color $r$ for each row
        add a vertex $c_j$ for $j = 1, \dots, n$ of color $c$ for each column
        add a vertex $e_{ij}$ for $i, j = 1, \dots, n$ of color $e$ for each element
        add a vertex $s$ for each unique value $M_{ij}$
2: Construct the edge set $\mathcal{E}$:
        connect each row vertex $r_i$ with the matrix element vertex $e_{ij}$
        connect each column vertex $c_j$ with the matrix element vertex $e_{ij}$
        connect each matrix element vertex $e_{ij}$ with the unique matrix value $s_k$
```

The graph $\Gamma(M)$ contains a vertex r_i for each row $i = 1, \dots, n$, a vertex c_j for each column $j = 1, \dots, n$, and a vertex e_{ij} for each element $i, j = 1, \dots, n$ of the matrix $M \in \mathbb{R}^{n\times n}$. For each unique value $s \in \{M_{ij} : i, j = 1, \dots, n\}$ of the matrix we create a vertex s. The rows, columns, and element vertices are different colors. Each value vertex $s \in \{M_{ij} : i, j = 1, \dots, n\}$ has a different color. The graph $\Gamma(M)$ contains edges that connect each element vertex e_{ij} to the row r_i and column c_j vertices and the vertex for the value $s = M_{ij}$. If $M_{ij} = M_{kl}$ then the element vertices e_{ij} and e_{kl} are both connected to the same value vertex $s \in \{M_{ij} : i, j = 1, \dots, n\}$.

The graph $\Gamma(M) = (\mathcal{V}, \mathcal{E})$ has $|\mathcal{V}| = n^2 + 2n + n_s$ vertices where $n_s = |\{M_{ij} : i, j = 1, \ldots, n\}|$ is the number of unique values in the matrix M. There are $3 + n_s$ vertex colors. The graph $\Gamma(M)$ has $|\mathcal{E}| = 3n^2$ edges; each element vertex e_{ij} is neighbors with the row vertex r_i, column vertex c_j, and the value $s = M_{ij}$ vertex.

The graph automorphism software searches for permutations $\pi \in \text{Aut}(\Gamma(M))$ of the vertex set $\mathcal{V}$ that preserves the edge list $\mathcal{E}$ and vertex coloring. The graph automorphism software produces a set $\mathcal{S}$, of at most $n-1$ permutations, that generates the symmetry group $\text{Aut}(\Gamma(M)) = \langle \mathcal{S} \rangle$.

The graph $\Gamma(M)$ is constructed so that the symmetries $\pi \in \text{Aut}(\Gamma(M))$ have a particular structure. The graph automorphism software will not find permutations π that transpose row vertices with non-row vertices since they have different colors. Therefore each $\pi \in \text{Aut}(\Gamma(M))$ will map row vertices r_i to row vertices. Likewise each $\pi \in \text{Aut}(\Gamma(M))$ maps column vertices c_j to column vertices and element vertices e_{ij} to element vertices. The value vertices $s \in \{M_{ij} : i, j = 1, \ldots, n\}$ can only be mapped to themselves since each has a different color.

Let π_n denote the restriction of the permutations $\pi \in \text{Aut}(\Gamma(M))$ to the set of row vertices $\mathcal{V}_r = \{r_1, \ldots, r_n\} \subset \mathcal{V}$. The edges of the graph $\Gamma(M)$ are constructed such that π_n is also the restriction of $\pi \in \text{Aut}(\Gamma(M))$ to the column vertices. The element vertices e_{ij} are permuted according to the expression $\pi(e_{ij}) = e_{\pi_n(i)\pi_n(j)}$. The value vertices $s \in \{M_{ij} : i, j = 1, \ldots, n\}$ are not permuted by any $\pi \in \text{Aut}(\Gamma(M))$. Therefore each row permutation π_n uniquely defines the permutation $\pi \in \text{Aut}(\Gamma(M))$. Thus the set of restricted permutations

$$\big\{\pi_n : \pi \in \text{Aut}(\Gamma(M))\big\} \tag{5.39}$$

is a group isomorphic to $\text{Aut}(\Gamma(M))$. With abuse of notation, we will denote this group by $\text{Aut}(\Gamma(M))$. Thus we can define the following isomorphism from $\text{Aut}(\Gamma(M))$ to $\text{Perm}(M)$ by

$$\Pi_{ij} = \begin{cases} 1 & \text{for } r_j = \pi_n(r_i) \\ 0 & \text{otherwise} \end{cases} \tag{5.40}$$

where $\Pi \in \text{Perm}(M) \subset \mathbb{R}^{n\times n}$ and $\pi_n \in \text{Aut}(\Gamma(M))$. This result is formally stated in the following theorem.

Theorem 12. *The symmetry group* $\text{Aut}(\Gamma(M))$ *of the vertex colored graph* $\Gamma(M)$ *is isomorphic to the permutation group* $\text{Perm}(M)$ *of the square matrix* $M \in \mathbb{R}^{n\times n}$ *with isomorphism* (5.40)*.*

Proof. See [13]. □

This procedure is demonstrated by the following example.

Example 14. *The graph* $\Gamma(I)$ *for the* 2×2 *identity matrix* $I \in \mathbb{R}^{2\times 2}$ *is shown in Figure 5.1. The green diamonds represent the row vertices* r_1 *and* r_2 *and the blue squares represent*

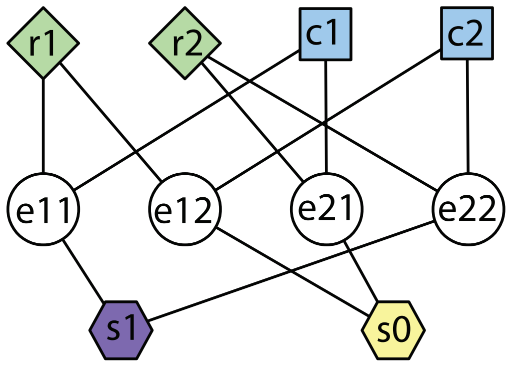


Figure 5.1: Graph $\Gamma(I)$ for the identity matrix $I \in \mathbb{R}^{2\times 2}$. Green diamonds represent row vertices r_i, blue squares represent column vertices c_j, white circles represent element vertices e_{ij}, and the hexagons represent the unique values $s1 = 1$ and $s0 = 0$.

the column vertices c_1 and c_2. The white circles represent the matrix element vertices e_{ij}. The purple and yellow hexagons represent the unique matrix values $I_{1,1} = I_{2,2} = 1$ and $I_{1,2} = I_{2,1} = 0$ respectively.

The graph automorphism software will search for permutations π of the vertex set $\mathcal{V}$ that preserve the edge list $\mathcal{E}$ and vertex coloring. For instance the the permutation

$$\pi = \begin{pmatrix} r_1 & r_2 & c_1 & c_2 & e_{11} & e_{12} & e_{21} & e_{22} & s_1 & s_0 \\ c_1 & r_2 & r_1 & c_2 & e_{11} & e_{12} & e_{21} & e_{22} & s_1 & s_0 \end{pmatrix} \tag{5.41}$$

is not a symmetry of $\Gamma(M)$ since it transposes vertices of different colors: row vertex r_1 and column vertex c_1. The permutation

$$\pi = \begin{pmatrix} r_1 & r_2 & c_1 & c_2 & e_{11} & e_{12} & e_{21} & e_{22} & s_1 & s_0 \\ r_1 & r_2 & c_2 & c_1 & e_{11} & e_{12} & e_{21} & e_{22} & s_1 & s_0 \end{pmatrix} \tag{5.42}$$

is not a symmetry of $\Gamma(M)$ since it changes the edge list: $\{c_2, e_{22}\} \in \mathcal{E}$ but $\pi(\{c_2, e_{22}\}) = \{c_1, e_{22}\} \notin \mathcal{E}$. However the permutation

$$\pi = \begin{pmatrix} r_1 & r_2 & c_1 & c_2 & e_{11} & e_{12} & e_{21} & e_{22} & s_1 & s_0 \\ r_2 & r_1 & c_2 & c_1 & e_{22} & e_{21} & e_{12} & e_{11} & s_1 & s_0 \end{pmatrix} \tag{5.43}$$

and the identity permutation are elements of $\mathrm{Aut}(\Gamma(I))$ since they preserve the edge list $\mathcal{E}$ and vertex coloring. In fact these are the only permutations, out of the 10! possible permutations of the vertex set $\mathcal{V}$, that preserve the edge list $\mathcal{E}$ and vertex colors. Using the isomorphism (5.40) we find that the permutation group of the matrix $M = I$ is

$$\mathrm{Perm}(I) = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}. \tag{5.44}$$

Alternative graph constructions $\Gamma(M)$ can be found in the literature [17, 63]. Using our graph construction $\Gamma(M)$ in Procedure 2 we can find the permutation group $\mathrm{Perm}(M)$ for matrices $M \in \mathbb{R}^{n\times n}$ in thousand of dimensions $O(n) = 1000$.

Permutations that Commute with Multiple Matrices

In this section we present the procedure for finding the set of all permutation matrices that commute with multiple matrices $M_1, \ldots, M_r$. The obvious way to calculate the set of all permutation matrices that commute with multiple matrices $M_1, \ldots, M_r$ is to calculate the groups $\mathrm{Perm}(M_1), \ldots, \mathrm{Perm}(M_r)$ and then compute their intersection. However this method is impractical since the groups $\mathrm{Perm}(M_i)$ can have exponential size in n.

Calculating the intersection of groups using the generators is non-trivial since the intersection of the group generators $\mathcal{S} = \bigcap_{i=1}^r \mathcal{S}_i$ does not necessarily generate the intersection of the groups $\bigcap_{i=1}^r \mathrm{Perm}(M_i) \neq \langle \mathcal{S} \rangle$. This is demonstrated by the following example.

Example 15. *Consider the matrices*

$$M_1 = \begin{bmatrix} I_{n-1} & 0 \\ 0 & 2 \end{bmatrix} \text{ and } M_2 = \begin{bmatrix} 2 & 0 \\ 0 & I_{n-1} \end{bmatrix}. \tag{5.45}$$

The size of the permutation groups for these matrices are $|\mathrm{Perm}(M_1)| = |\mathrm{Perm}(M_2)| = (n-1)!$. *Therefore directly computing the intersection* $\mathrm{Perm}(M_1) \cap \mathrm{Perm}(M_2)$ *is impractical.*

In general we cannot calculate the intersection $\mathrm{Perm}(M_1) \cap \mathrm{Perm}(M_2)$ *using the group generators* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *of* $\mathrm{Perm}(M_1) = \langle \mathcal{S}_1 \rangle$ *and* $\mathrm{Perm}(M_2) = \langle \mathcal{S}_2 \rangle$ *respectively. A set of generators for the group* $\mathrm{Perm}(M_1)$ *are the permutation matrices that transpose row* 1 *with row* j *for* $j = 2, \ldots, n-1$. *A set of generators for the group* $\mathrm{Perm}(M_2)$ *are the permutation matrices that transpose row* n *with row* j *for* $j = 2, \ldots, n-1$. *Therefore the intersection of these generators is the empty set. However the intersection of the groups* $\mathrm{Perm}(M_1)$ *and* $\mathrm{Perm}(M_2)$ *is non-empty. It has size* $|\mathrm{Perm}(M_1) \cap \mathrm{Perm}(M_2)| = (n-2)!$.

We identify the permutation group $\mathrm{Perm}(\{M_k\}_{k=1}^r)$ by creating a single matrix M that captures the structure of the set of matrices $\{M_k\}_{k=1}^r$. The matrix M has the property $M_{ij} = M_{hl}$ if and only if $M_{ij,k} = M_{hl,k}$ for $k = 1, \ldots, r$. Thus a permutation matrix Π will commute with M if and only if it will commute with each M_k. Therefore $\mathrm{Perm}(M) = \mathrm{Perm}(\{M_k\}_{k=1}^r)$. The permutation group $\mathrm{Perm}(M)$ is identified using Procedure 2 from the previous section. This method is demonstrated in the following examples.

Example 16. *Consider the matrices* $M_1 \in \mathbb{R}^{n \times n}$ *and* $M_2 \in \mathbb{R}^{n \times n}$ *from Example 15. The matrix*

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2I_{n-1} & 0 \\ 0 & 0 & 3 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{5.46}$$

has the property that $M_{ij} = M_{hl}$ *if and only if* $M_{ij,1} = M_{hl,1}$ *and* $M_{ij,2} = M_{hl,2}$. *From inspection we can see that the symmetry group* $\mathrm{Perm}(M)$ *of this matrix* M *is the set of permutation matrices* Π *that permute rows* $2, \ldots, n-1$. *This is the permutation group* $\mathrm{Perm}(M_1, M_2) = \mathrm{Perm}(M)$.

Example 17. *Consider the autonomous system* (5.9) *from Example 13. In order to find the symmetry group* $\mathrm{Aut}(\Sigma)$ *we must find the group* $\mathrm{Perm}(\Sigma)$ *of all permutation matrices* Π *that commute with the matrices* $M_1 = HH^T$ *and* $M_2 = HAH^T$ *where*

$$M_1 = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \text{ and } M_2 = \frac{1}{2}\begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \end{bmatrix}. \tag{5.47a}$$

The pair of matrices M_1 *and* M_2 *has four unique values*

$$(M_{ij,1}, M_{ij,2}) \in \left\{(\tfrac{1}{2},0),(0,\text{-}\tfrac{1}{2}),(\text{-}\tfrac{1}{2},0),(0,\tfrac{1}{2})\right\}. \tag{5.48}$$

Arbitrarily labeling these value $a = (\frac{1}{2},0)$*,* $b = (0,\text{-}\frac{1}{2})$*,* $c = (\text{-}\frac{1}{2},0)$*, and* $d = (0,\frac{1}{2})$ *we can construct the following matrix*

$$M = \begin{bmatrix} a & b & c & d \\ d & a & b & c \\ c & d & a & b \\ b & c & d & a \end{bmatrix} \tag{5.49}$$

which has the property that $M_{ij} = M_{hl}$ *if and only if* $M_{ij,1} = M_{hl,1}$ *and* $M_{ij,2} = M_{hl,2}$.

Using Procedure 2 we can construct a graph $\Gamma(M) = (\mathcal{V},\mathcal{E})$ *whose automorphism group* $\mathrm{Aut}(\Gamma(M))$ *is isomorphic to* $\mathrm{Perm}(M) = \mathrm{Perm}(M_1) \cap \mathrm{Perm}(M_1)$*. The graph has* $|\mathcal{V}| = 28$ *vertices and* $|\mathcal{E}| = 48$ *edges. The generators of the symmetry group* $\mathrm{Aut}(\Gamma(M))$ *of the graph* $\Gamma(M)$ *are found using graph automorphism software. The group* $\mathrm{Aut}(\Gamma(M)) = \langle \pi_n \rangle$ *has one generator*

$$\pi_n = \begin{pmatrix} r_1 & r_2 & r_3 & r_4 \\ r_4 & r_1 & r_2 & r_3 \end{pmatrix}. \tag{5.50}$$

Using isomorphism (5.40) we find that $\mathrm{Perm}(M) = \langle \Pi \rangle$ *is generated by the cyclic matrix*

$$\Pi = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{5.51}$$

Using isomorphism (5.7) for $\mathrm{Perm}(\Sigma)$ *to* $\mathrm{Aut}(\Sigma)$ *we find that* $\mathrm{Aut}(\Sigma) = \langle \Theta \rangle$ *is generated by the* 90 *degree rotation matrix*

$$\Theta = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{5.52}$$

Thus the symmetry group $\mathrm{Aut}(\Sigma) = \langle \Theta \rangle$ *of the autonomous system* (5.9) *is the set of all rotations of the state-space by increments of* 90 *degrees.*

Block Structured Permutation Matrices

In this section we show how to modify Procedure 2 to restrict our search for permutation matrices $\Pi \in \mathbb{R}^{n\times n}$ to permutations matrices with a block structure

$$\Pi = \begin{bmatrix} \Pi_1 & & \\ & \ddots & \\ & & \Pi_q \end{bmatrix}. \tag{5.53}$$

We can encode this desired block structure into the graph $\Gamma(M)$. Let the blocks $\Pi_k \in \mathbb{R}^{n_k\times n_k}$ have size n_k such that $n = \sum_{k=1}^{q} n_k$. For $i \in [n_k, n_{k+1})$ we give the row vertex r_i color k for $k = 1,\dots,q$. Since the row vertices have different colors, the graph automorphism software will only return permutations π_n that correspond to the block structure (5.53). This is demonstrated by the following example.

Example 18. *Consider the double integrator system*

$$x(t+1) = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{A} x(t) + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u(t) \tag{5.54}$$

subject to the constraints $x(t) \in \mathcal{X}$ *and* $u(t) \in \mathcal{U}$ *where*

$$\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \le x_i \le 1, i = 1,2\} \tag{5.55a}$$
$$\mathcal{U} = \{u \in \mathbb{R} : -1 \le u \le 1\}. \tag{5.55b}$$

Under the regular basis we have

$$A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ and } B = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5.56}$$

In order to find the symmetry group $\mathrm{Aut}(\Sigma)$ *of this constrained linear system we must find the group* $\mathrm{Perm}(\Sigma)$ *of all permutation matrices* $\Pi_x \in \mathbb{R}^{4\times4}$ *and* $\Pi_u \in \mathbb{R}^{2\times2}$ *that commute with the matrices*

$$M_1 = \begin{bmatrix} H_xH_x^T & 0 \\ 0 & H_uH_u^T \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{5.57a}$$

$$M_2 = \begin{bmatrix} H_xAH_x^T & H_xBH_u^T \\ 0 & H_uH_u^T \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}. \tag{5.57b}$$

The pair of matrices M_1 and M_2 have seven unique values. We can construct a matrix $M \in \mathbb{R}^{6\times 6}$ with the property $M_{ij} = M_{hl}$ if and only if $M_{ij,1} = M_{hl,1}$ and $M_{ij,2} = M_{hl,2}$.

Using Procedure 2 we can construct a graph$\Gamma(M) = (\mathcal{V}, \mathcal{E})$ that captures the symmetry of the constrained double integrator system. This graph has $|\mathcal{V}| = 55$ vertices and $|\mathcal{E}| = 108$ edges. The generators of the symmetry group $\mathrm{Aut}(\Gamma(M))$ of the graph $\Gamma(M)$ are found using graph automorphism software. The group $\mathrm{Aut}(\Gamma(M)) = \langle \pi_n \rangle$ has one generator

$$\pi_n = \begin{pmatrix} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \\ r_3 & r_4 & r_1 & r_2 & r_6 & r_5 \end{pmatrix}. \tag{5.58}$$

The vertices $\{r_1, r_2, r_3, r_4\}$ correspond to the facets of the set $\mathcal{X} \subset \mathbb{R}^2$ and the vertices $\{r_5, r_6\}$ correspond to facets of the polytope $\mathcal{U} \subset \mathbb{R}$. Parsing the permutation π_n we obtain the permutation matrices

$$\Pi_x = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \text{ and } \Pi_u \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{5.59}$$

Using isomorphism (5.7) for $\mathrm{Perm}(\Sigma)$ to $\mathrm{Aut}(\Sigma)$ we find that $\mathrm{Aut}(\Sigma) = \langle (\Theta, \Omega) \rangle$ is generated by the pair of matrices

$$\Theta = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } \Omega = -1. \tag{5.60}$$

Since $\Theta^2 = I$ and $\Omega^2 = 1$ this is the only non-trivial symmetry $(\Theta, \Omega) \in \mathrm{Aut}(\Sigma)$ of the double integrator system.

5.5 Example: Quadrotor

In this section we apply Procedure 1 to the quadrotor system described in [19, 7] and shown in Figure 5.2. The quadrotor has $n = 13$ states: 6 cartesian position and velocity states, 6 angular position and velocity states, and 1 integrator state to counteract gravity. The quadrotor has $m = 4$ inputs corresponding to the voltage applied to the four motors. The continuous-time non-linear dynamics were derived using a Newton-Euler model

$$\ddot{x} = -\frac{1}{m}\Big(\beta\dot{x} + \sin(\theta_x)\sum\nolimits_{i=1}^{4} Fu_i\Big) \tag{5.61a}$$
$$\ddot{y} = -\frac{1}{m}\Big(\beta\dot{y} + \cos(\theta_x)\sin(\theta_y)\sum\nolimits_{i=1}^{4} Fu_i\Big) \tag{5.61b}$$
$$\ddot{z} = -\frac{1}{m}\Big(\beta\dot{z} + \cos(\theta_x)\cos(\theta_y)\sum\nolimits_{i=1}^{4} Fu_i\Big) - g \tag{5.61c}$$
$$\ddot{\theta}_x = -\frac{l}{I_{xx}}\Big(Fu_2 - Fu_4\Big) \tag{5.61d}$$
$$\ddot{\theta}_y = -\frac{l}{I_{yy}}\Big(Fu_3 - Fu_1\Big) \tag{5.61e}$$
$$\ddot{\theta}_z = -\frac{1}{I_{zz}}\Big(-Tu_1 + Tu_2 - Tu_3 + Tu_4\Big) \tag{5.61f}$$
$$\dot{e} = z \tag{5.61g}$$

where x, y, z describe the cartesian position, θ_x, θ_y, θ_z describe the angular orientation, e is the integrator state, F is the gain from the motor voltage u_i to thrust, T is the gain from the motor voltage u_i to torque, l is the length of the arms holding the motors, g is the gravitational acceleration, m is the mass of the quadrotor, I_{xx}, I_{yy}, and I_{zz} are the moments of inertia of the quadrotor about the x, y, and z axes respectively, and β is the drag coefficient. The parameters for this model were taken from [19, 7]. The dynamics were linearized about the equilibrium $[x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta_x, \theta_y, \theta_z, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z]^T = 0$. The dynamics discretize using the zero-order hold method.

The constraints on the states and inputs are simple upper and lower bounds

$$\begin{array}{rcl} -p \leq & x, y, z & \leq p \\ -v \leq & \dot{x}, \dot{y}, \dot{z} & \leq v \\ -\alpha \leq & \theta_x, \theta_y, \theta_z, & \leq \alpha \\ -\omega \leq & \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z, & \leq \omega \\ -\bar{u} \leq & u_1, u_2, u_3, u_4 & \leq \bar{u} \end{array} \tag{5.62}$$

where p, v, α, and ω are the maximum deviation of the cartesian position, cartesian velocity, angular position, and angular velocity from 0 respectively, and $\bar{u}$ maximum deviation of the motor voltages u_i from the nominal voltage. The values for p, v, α, and ω were taken from [19, 7].

Using Procedure 1 we found $|\operatorname{Aut}(\Sigma)| = 16$ symmetries with $|\mathcal{S}| = 3$ generators. The

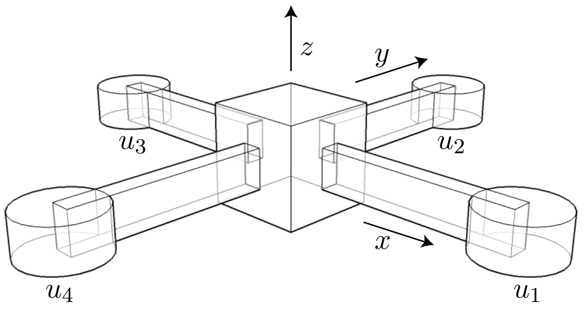


Figure 5.2: Quadrotor layout.

generators of the state-space transformations are

$$\Theta_1 = \text{blkdiag}\left\{ I_2 \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, I_2 \otimes \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 1 \right\} \tag{5.63a}$$

$$\Theta_2 = \text{blkdiag}\left\{ I_2 \otimes \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, I_2 \otimes \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, 1 \right\} \tag{5.63b}$$

$$\Theta_3 = \text{blkdiag}\left\{ I_2 \otimes \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, I_2 \otimes \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, -1 \right\} \tag{5.63c}$$

where $\otimes$ is the Kronecker product and blkdiag represents a block-diagonal matrix. The generators of the input-space transformations are

$$\Omega_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \Omega_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \Omega_3 = -\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.64}$$

The generator pair (Θ_1, Ω_1) mirrors the quadrotor through the xz-plane. The input-space transformation Ω_1 swaps motors 2 and 4. The state-space transformation Θ_1 reverses the y-axis. Rotations about the x-axis are reversed since motors 2 and 4 have been swapped.

The generator pair (Θ_2, Ω_2) rotates the quadrotor 90 degrees about the z-axis. The input-space transformation Ω_2 maps motor 1 to 2, motor 2 to 3, motor 3 to 4, and motor 4 to 1. The state-space transformation Θ_2 maps the x-axis to the y-axis and the y-axis to the negative x-axis. Rotations about the x- and y axis are swapped and rotations about the x-axis are reversed.

The generator pair (Θ_3, Ω_3) flips the gravity vector. The input-space transformation $\Omega_3 = -I$ reverses the thrust on all the motors. The state-space transformation reverses each of the x, y, and z axes and rotations about these axes.

From a controls perspective, these symmetries make intuitive sense. Designing an input sequence $\{u(t)\}_{t=0}^{\infty}$ that makes the quadrotor fly along the positive x-axis is equivalent to designing an input sequence $\{\Omega u(t)\}_{t=0}^{\infty}$ that makes it fly along the negative x-axis or along either direction along the y-axis. With the integrator state counteracting gravity, an input sequence $\{u(t)\}_{t=0}^{\infty}$ that increases altitude is equivalent to one $\{\Omega u(t)\}_{t=0}^{\infty}$ that decreases altitude. Symmetry codifies this intuition into linear transformations.

Chapter 6

Fundamental Domains

In this section we define fundamental domains and present algorithms for solving two common problems associated with fundamental domains. A fundamental domain $\hat{\mathcal{X}}$ is a type of subset of a symmetric set $\mathcal{X}$. Intuitively a fundamental domain contains all the 'information' of the original set but without the 'redundancy' due to symmetry. Later in this dissertation we will use fundamental domains to simplify symmetric controllers by eliminating part of the controller that is symmetrically redundant.

6.1 Definition of Fundamental Domain

In this section we define minimal polytopic fundamental domains and present two problems associated with fundamental domains.

Recall that the symmetry group $\mathrm{Aut}(\mathcal{X})$ of a polytope $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of all matrices $\Theta \in \mathbb{R}^{n \times n}$ that map the set to itself $\Theta\mathcal{X} = \mathcal{X}$. Two points $x, y \in \mathcal{X}$ are equivalent under a group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ if they are in the same orbit i.e. $y = \Theta x$ for some $\Theta \in \mathcal{G}$. Thus the group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ partitions the set $\mathcal{X}$ into disjoint orbits $\mathcal{G}x = \{\Theta x : \Theta \in \mathcal{G}\}$ where $x \in \mathcal{X}$. A fundamental domain $\hat{\mathcal{X}}$ is subset of $\mathcal{X}$ that contains at least one representative $y \in \mathcal{G}x$ from each orbit $\mathcal{G}x$ for $x \in \mathcal{X}$.

We would like our fundamental domains to have two additional properties. First we want the fundamental domain $\hat{\mathcal{X}} \subseteq \mathcal{X}$ to be a polytope. This property is desirable since in our applications we will be solving optimization problems on $\hat{\mathcal{X}}$. Second we want the fundamental domain $\hat{\mathcal{X}}$ to be minimal in the sense that any closed subset of $\hat{\mathcal{X}}$ will not be a fundamental domain. We call such a set a minimal polytopic fundamental domain.

Definition 9. A polytope $\hat{\mathcal{X}}$ is a fundamental domain of $\mathcal{X}$ with respect to the group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ if for each $x \in \mathcal{X}$ there exists $\Theta \in \mathcal{G}$ such that $\Theta x \in \hat{\mathcal{X}}$. The fundamental domain $\hat{\mathcal{X}}$ is minimal if each closed polytopic subset $\hat{\mathcal{Y}} \subset \hat{\mathcal{X}}$ is not a fundamental domain.

The following theorem characterizes fundamental domains with these properties.

Theorem 13. *A polytope $\hat{\mathcal{X}} \subseteq \mathcal{X}$ is a polytopic fundamental domain if it satisfies*

$$\bigcup_{\Theta \in \mathcal{G}} \Theta \hat{\mathcal{X}} = \mathcal{X}. \tag{6.1}$$

The fundamental domain $\hat{\mathcal{X}}$ is minimal if it satisfies

$$\operatorname{int}(\hat{\mathcal{X}} \cap \Theta \hat{\mathcal{X}}) = \varnothing \tag{6.2}$$

for all $\Theta \neq I \in \mathcal{G}$.

Proof. By the hypothesis of the theorem the set $\hat{\mathcal{X}}$ is a polytope.

Next we show by contraposition that (6.1) implies $\hat{\mathcal{X}}$ is a fundamental domain. Suppose there exists an orbit $\mathcal{G}x = \{\Theta x : \Theta \in \mathcal{G}\}$ for some $x \in \mathcal{X}$ that does not have a representative in $\hat{\mathcal{X}}$. Then $\bigcup_{\Theta \in \mathcal{G}} \Theta \hat{\mathcal{X}} \neq \mathcal{X}$ since it does not contain x. By the contrapositive, this means (6.1) implies that $\hat{\mathcal{X}}$ contains a representative from every orbit.

Finally we prove (6.2) implies $\hat{\mathcal{X}}$ is a minimal fundamental domain. Suppose there exists a closed fundamental domain $\hat{\mathcal{Y}} \subset \hat{\mathcal{X}}$. Then the set $\hat{\mathcal{X}} \setminus \hat{\mathcal{Y}}$ has a non-empty interior since $\hat{\mathcal{Y}}$ is a closed strict subset of $\hat{\mathcal{X}} \subseteq \mathbb{R}^n$. Consider an open ball $\mathcal{B} \subset \hat{\mathcal{X}} \setminus \hat{\mathcal{Y}}$. By (6.2) we have

$$\operatorname{int}(\mathcal{B} \cap \Theta \hat{\mathcal{X}}) \subset \operatorname{int}(\hat{\mathcal{X}} \cap \Theta \hat{\mathcal{X}}) = \varnothing. \tag{6.3}$$

Thus $\operatorname{int}(\mathcal{B} \cap \Theta \hat{\mathcal{Y}}) = \varnothing$ since $\Theta \hat{\mathcal{Y}} \subset \Theta \hat{\mathcal{X}}$. Additionally $\mathcal{B} \cap \hat{\mathcal{Y}} = \varnothing$ by definition of $\mathcal{B} \subset \hat{\mathcal{X}} \setminus \hat{\mathcal{Y}}$. Thus $\hat{\mathcal{Y}}$ fails to satisfy (6.1) since it does not cover $\mathcal{B} \subset \mathcal{X}$. □

We consider two problems related to fundamental domains. The first problem is how to construct a minimal polytopic fundamental domain.

Problem 2 (Fundamental Domain Construction)**.** Given a polytope $\mathcal{X}$ and finite matrix group $\mathcal{G} \subseteq \operatorname{Aut}(\mathcal{X})$ calculate a minimal polytopic fundamental domain $\hat{\mathcal{X}}$.

The second problem is how to search the group $\mathcal{G}$ for a symmetry that maps a point $x \in \mathcal{X}$ into the set $\hat{\mathcal{X}}$.

Problem 3 (Fundamental Domain Search)**.** Given a point $x \in \mathcal{X}$ find a group element $\Theta \in \mathcal{G}$ that maps the point x to its orbit representative $y = \Theta x \in \mathcal{G}x$ in the fundamental domain $\Theta x \in \hat{\mathcal{X}}$.

We assume the polytope $\mathcal{X}$ is bounded, full-dimensional, and contains the origin in its interior. In this case under the regular basis the half-space matrix H of the polytope $\mathcal{X} = \{x : Hx \leq \mathbf{1}\}$ satisfies $H^T H = I$.

For a bounded, full-dimensional polytope $\mathcal{X}$ a linear symmetry $\Theta \in \operatorname{Aut}(\mathcal{X})$ will permute its facets. Thus $\operatorname{Aut}(\mathcal{X})$ can be represented by a permutation group acting on the set of non-redundant half-space indices $\{1, \ldots, c\}$. We restate the following theorem that relates permutations π and linear transformations Θ.

Theorem 14. *The linear symmetry group* $\mathrm{Aut}(\mathcal{X})$ *of a polytope* $\mathcal{X}$ *is isomorphic to a permutation group acting on the facets of* $\mathcal{X}$. *A permutation* $\pi \in \mathrm{Aut}(\mathcal{X})$ *and matrix* $\Theta \in \mathrm{Aut}(\mathcal{X})$ *are related by*

$$\Theta = H^T \Pi H \tag{6.4}$$

where Π *is the permutation matrix of the permutation* π *and* H *is the half-space matrix of* $\mathcal{X}$ *in the regular basis.*

6.2 Fundamental Domain Construction

In this section we present an algorithm for constructing a minimal fundamental domain $\hat{\mathcal{X}}$ of a polytope $\mathcal{X}$ with respect to a group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$. First we present an existing method for computing minimal fundamental domains. Then we present a modification of this method that can construct minimal fundamental domains in polynomial time.

Existing Method

In [2] the authors presented a method for constructing minimal polytopic fundamental domains. Their method is based on calculating the Voronoi diagram of the orbit of a point $z \in \mathcal{X}$ not fixed $\Theta z \neq z$ by any non-identity element $\Theta \neq I$ of the group $\mathcal{G}$. The following theorem summarizes and generalizes the results of [2].

Theorem 15. *Let* $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ *be an orthogonal matrix group. The Voronoi cell*

$$\mathcal{F}_{\mathcal{G}z}(z) = \big\{ x \in \mathcal{X} : \|z - x\|_2 \leq \|\Theta z - x\|_2, \ \forall \Theta \in \mathcal{G} \big\} \tag{6.5}$$

of a point z *and its orbit* $\mathcal{G}z$ *has the following properties*

1. $\mathcal{F}_{\mathcal{G}z}(z) = \{ x \in \mathcal{X} : z^T x \geq z^T \Theta x, \ \forall \Theta \in \mathcal{G} \}$
2. $\bigcup_{\Theta \in \mathcal{G}} \Theta \mathcal{F}_{\mathcal{G}z}(z) = \mathcal{X}$
3. $\mathrm{int}\big(\mathcal{F}_{\mathcal{G}z}(z) \cap \Theta \mathcal{F}_{\mathcal{G}z}(z)\big) = \varnothing$ *for all* $\Theta \in \mathcal{G}/\mathcal{G}_z$.

Proof.

1. Follows directly from the orthogonality of the matrices $\Theta \in \mathcal{G}$ and definition of the Euclidean norm.

2. For any $x \in \mathcal{X}$ we can define the finite set of real-numbers

$$\big\{ z^T \Theta x : \Theta \in \mathcal{G} \big\} \subset \mathbb{R}. \tag{6.6}$$

This set has a maximum element $z^T \bar{\Theta} x \geq z^T \Theta x$ for all $\Theta \in \mathcal{G}$. By the closure of the group $\mathcal{G}$ this is equivalent to $z^T \bar{\Theta} x \geq z^T \Theta \bar{\Theta} x$ for all $\Theta \in \mathcal{G}$. Thus for all $x \in \mathcal{X}$ there exists $\bar{\Theta}$ such that $\bar{\Theta} x \in \mathcal{F}_{\mathcal{G}z}(z)$.

3. The set $\mathcal{F}_{\mathcal{G}z}(z)$ is contained in the half-space $\{x : (z - \Theta z)^T x \geq 0\}$ for $\Theta \in \mathcal{G}$. The set $\Theta\mathcal{F}_{\mathcal{G}z}(z)$ is contained in the half-spaces $\{x : (\Theta z - \Theta\hat{\Theta} z)^T x \geq 0\}$ for every $\hat{\Theta} \in \mathcal{G}$. In particular for $\hat{\Theta} = \Theta^{-1} \in \mathcal{G}$ we see the set $\Theta\mathcal{F}_{\mathcal{G}z}(z)$ is contained in the half-space $\{x : (z - \Theta z)^T x \leq 0\}$. Thus $\mathcal{F}_{\mathcal{G}z}(z) \cap \Theta\mathcal{F}_{\mathcal{G}z}(z) \subseteq \{x : (z - \Theta z)^T x = 0\}$. If $\Theta \notin \mathcal{G}_z$ is not in the stabilizer subgroup $\mathcal{G}_z$ of z then $\Theta z \neq z$ and therefore $\{x : (z - \Theta z)^T x = 0\}$ has no interior.

□

An immediate corollary of this theorem, given below, allows us to construct minimal fundamental domains.

Corollary 6. *If the point z is not fixed $\Theta z \neq z$ by any non-identity element $\Theta \neq I$ of $\mathcal{G}$ then the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ is a minimal fundamental domain.*

Proof. Since the point z is not fixed by any non-identity element of $\mathcal{G}$, its stabilizer subgroup $\mathcal{G}_z = \langle \varnothing \rangle$ is trivial and therefore $\mathcal{G}/\mathcal{G}_z = \mathcal{G}$. Thus the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ satisfies (6.1) and (6.2). □

This corollary says that we can construct a fundamental domain using the Voronoi cell of a non-fixed point z. This suggests Procedure 3 as a method for constructing minimal polytopic fundamental domains. First we find a non-symmetric point z such that $\Theta z \neq z$ for all $\Theta \in \mathcal{G}$. Then we calculate the image Θz of this point z under every symmetry $\Theta \in \mathcal{G}$. Finally we use a Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ to separate the point z from its orbit $\mathcal{G}z$. This procedure is demonstrated in the following example.

Procedure 3 Fundamental Domain Construction

```
1: Find a point $z \in \mathcal{X}$ not fixed by $\mathcal{G}$.
2: Calculate the orbit of $\mathcal{G}z$
3: Construct the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$
```

Example 19. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group* $\mathrm{Aut}(\mathcal{X}) = \mathcal{D}_4$ *generated by $\mathcal{S} = \{S_1, S_2\}$ where S_1 is the 90 degree counter-clockwise rotation and S_2 is the vertical reflection*

$$S_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{ and } S_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{6.7}$$

The point $z = \left[\begin{smallmatrix} 1 \\ 0.5 \end{smallmatrix}\right] \in \mathcal{X}$ is not fixed by any element $\Theta \in \mathcal{G} = \mathrm{Aut}(\mathcal{X})$ of the group $\mathrm{Aut}(\mathcal{X}) = \mathcal{D}_4$. The orbit $\mathcal{G}z$ of this point z is the set

$$\mathcal{G}z = \left\{ \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}, \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0.5 \end{bmatrix}, \begin{bmatrix} -1 \\ -0.5 \end{bmatrix}, \begin{bmatrix} -0.5 \\ -1 \end{bmatrix}, \begin{bmatrix} 0.5 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -0.5 \end{bmatrix} \right\} \tag{6.8}$$

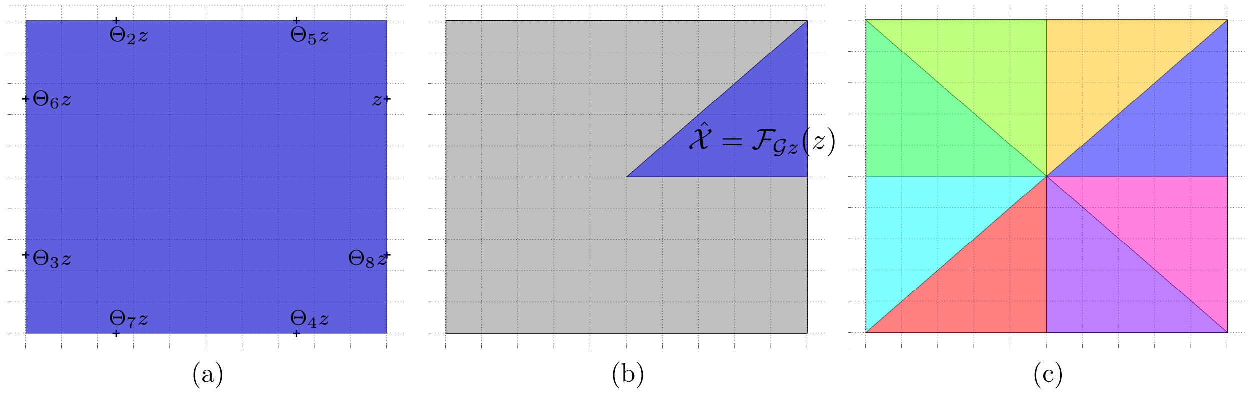


Figure 6.1: Example of constructing a fundamental domain using a Voronoi cell. (a) Square $\mathcal{X}$ with orbit $\mathcal{G}z$ of non-fixed point z. (b) Fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_{\mathcal{G}z}(z)$ constructed using Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ of non-fixed point z.

where $\mathcal{G} = \text{Aut}(\mathcal{X}) = \langle S_1, S_2 \rangle$. These points are shown in Figure 6.1a. The Voronoi cell

$$\mathcal{F}_{\mathcal{G}z}(z) = \left\{ x \in \mathcal{X} : \begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix} x \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} \tag{6.9}$$

shown in Figure 6.1b is a fundamental domain for the square $\mathcal{X}$ under the dihedral-4 group $\mathcal{D}_4$. This can be seen in Figure 6.1c. The images $\Theta\hat{\mathcal{X}}$ of the set $\hat{\mathcal{X}} = \mathcal{F}_{\mathcal{G}z}(z)$ under the group $\mathcal{D}_4$ cover the square $\mathcal{X}$. Furthermore these images $\Theta\hat{\mathcal{X}}$ only overlap on the boundary. The fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_{\mathcal{G}z}(z)$ is the set of points $x \in \mathcal{X}$ closer to z than any other point $\Theta z \in \mathcal{G}z$ in the orbit $\mathcal{G}z$ of z.

There are two drawbacks to this procedure. First we do not have a systematic method for finding a point z that has no symmetry $\Theta z \neq z$ for all $\Theta \in \mathcal{G}$. The second drawback is the computational complexity of computing the orbit $\mathcal{G}z$ of the non-fixed point z. Algorithm 6 which may be used for computing orbits has complexity $O(|\mathcal{S}||\mathcal{G}z|)$ where $\mathcal{S}$ is a set of generators for the group $\mathcal{G}$. Since z is not-fixed $\Theta z \neq z$ by any non-identity $\Theta \neq I$ element of the group $\mathcal{G}$, the number of points $|\mathcal{G}z|$ in the orbit $\mathcal{G}z$ is the same as the group $|\mathcal{G}z| = |\mathcal{G}|$ by the Orbit-Stabilizer Theorem. The group size $|\mathcal{G}|$ is potentially exponential $O(|\mathcal{G}|) = c^n$ in the number c of facets and dimension n of the set $\mathcal{X} \subseteq \mathbb{R}^n$.

In the next section we present a systematic algorithm for constructing minimal polytopic fundamental domains in polynomial time.

Proposed Method

Consider the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ when the point z is fixed $\Theta z = z$ by some elements $\Theta \in \mathcal{G}$ of the group $\mathcal{G}$. By Theorem 15 the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ is still a fundamental domain. However it is not necessarily a minimal fundamental domain since $\text{int}(\mathcal{F}_{\mathcal{G}z}(z) \cap \Theta\mathcal{F}_{\mathcal{G}z}(z))$ may not be empty for $\Theta \in \mathcal{G}_z$. In fact it turns out $\mathcal{F}_{\mathcal{G}z}(z) \cap \Theta\mathcal{F}_{\mathcal{G}z}(z) = \mathcal{F}_{\mathcal{G}z}(z)$ since $\mathcal{F}_{\mathcal{G}z}(z) = \Theta\mathcal{F}_{\mathcal{G}z}(z)$ for all $\Theta \in \mathcal{G}_z$.

Lemma 2. *$\Theta\mathcal{F}_{\mathcal{G}z}(z) = \mathcal{F}_{\mathcal{G}z}(z)$ for all $\Theta \in \mathcal{G}_z$ in the stabilize subgroup $\mathcal{G}_z$ of z.*

Proof. For $\Theta \in \mathcal{G}_z = \{\Theta \in \mathcal{G} : \Theta z = z\}$ we have

$$\begin{aligned}
\Theta\mathcal{F}_{\mathcal{G}z}(z) &= \{x : z^T\Theta x \geq z^T\hat{\Theta}\Theta x,\ \forall\hat{\Theta} \in \mathcal{G}\} \\
&= \{x : z^T x \geq z^T\hat{\Theta}\Theta x,\ \forall\hat{\Theta} \in \mathcal{G}\} \\
&= \{x : z^T x \geq z^T\bar{\Theta} x,\ \forall\bar{\Theta} \in \Theta^{-1}\mathcal{G}\} \\
&= \{x : z^T x \geq z^T\bar{\Theta} x,\ \forall\bar{\Theta} \in \mathcal{G}\} \\
&= \mathcal{F}_{\mathcal{G}z}(z)
\end{aligned} \tag{6.10}$$

where $\bar{\Theta} = \hat{\Theta}\Theta$ and $\Theta z = z$ since $\Theta \in \mathcal{G}_z$. □

This lemma says symmetries $\Theta \in \mathcal{G}$ that fix the point $z = \Theta z$ also fix its Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z) = \Theta\mathcal{F}_{\mathcal{G}z}(z)$. In other words, the Voronoi cell $\mathcal{F}_{\mathcal{G}z}(z)$ is a symmetric set with respect to the stabilizer subgroup $\mathcal{G}_z = \{\Theta \in \mathcal{G} : \Theta z = z\}$ of the point z.

To create a minimal fundamental domain we need to eliminate the left-over symmetry $\mathcal{G}_z$ of the set $\hat{\mathcal{X}}_1 = \mathcal{F}_{\mathcal{G}z}(z)$. This can be done by constructing a fundamental domain $\hat{\mathcal{X}}_2$ of the set $\hat{\mathcal{X}}_1 = \mathcal{F}_{\mathcal{G}z}(z)$ with respect to its symmetry group $\mathcal{G}_z$. This set $\hat{\mathcal{X}}_2$ will be fundamental domains of the original set $\mathcal{X}$ since $\bigcup_{\Theta\in\mathcal{G}_z} \Theta\hat{\mathcal{X}}_2$ covers $\mathcal{F}_{\mathcal{G}z}(z)$ and $\bigcup_{\Theta\in\mathcal{G}/\mathcal{G}_z} \Theta\mathcal{F}_{\mathcal{G}z}(z)$. We can repeat this process until the resulting fundamental domain is minimal. This is the principle behind our proposed algorithm for constructing minimal fundamental domains.

Algorithm 8 Fundamental Domain Construction

Input: Base and strong generating set $(\mathcal{B}, \mathcal{S})$ for group $\mathcal{G}$
Output: Minimal fundamental domain $\hat{\mathcal{X}} = \bar{\mathcal{X}}_{|\mathcal{B}|}$ of $\mathcal{X}$ with respect to $\mathcal{G}$.

```
1: Initialize fundamental domain $\hat{\mathcal{X}}_0 = \mathcal{X}$
2: for $i = 1$ to $|\mathcal{B}|$ do
3:     Calculate the orbit $\mathcal{G}_{i-1}z_i = \langle\mathcal{S}_{i-1}\rangle z_i$.
4:     Calculate the Voronoi cell $\mathcal{F}_{i-1}(z_i)$
5:     Calculate the intersect $\hat{\mathcal{X}}_i = \hat{\mathcal{X}}_{i-1} \cap \mathcal{F}_{i-1}(z_i)$
6: end for
```

Algorithm 8 describes our method for constructing minimal fundamental domains. The algorithm requires as an input a non-redundant base $\mathcal{B}$ and strong generating set $\mathcal{S}$ for the group $\mathcal{G}$. A non-redundant bases and strong generating set can be systematically generated using the Schreier-Sims algorithm [48, 74]. Thus our algorithm does not have the previous issue of finding a non-fixed point.

The algorithm is initialized with the non-minimal fundamental domain $\hat{\mathcal{X}}_0 = \mathcal{X}$ which has the symmetry group $\mathcal{G}_0 = \mathcal{G}$. During iteration i, the algorithm computes a new fundamental domain $\hat{\mathcal{X}}_i = \hat{\mathcal{X}}_{i-1} \cap \mathcal{F}_{i-1}(z_i)$ by taking the intersection of the previous fundamental domain $\hat{\mathcal{X}}_{i-1}$ and the Voronoi cell

$$\mathcal{F}_{i-1}(z_i) = \left\{x \in \mathcal{X} : z_i^T x \geq y^T x, \forall y \in \mathcal{G}_{i-1} z_i\right\} \tag{6.11}$$

of the base point $z_i \in \mathcal{B}$ and its orbit $\mathcal{G}_{i-1} z_i$ under the subgroup $\mathcal{G}_{i-1}$ which stabilizes the previous $i-1$ base points

$$\mathcal{G}_{i-1} = \{\Theta \in \mathcal{G} : \Theta z_j = z_j, j = 1, \ldots, i-1\}. \tag{6.12}$$

Since the base $\mathcal{B}$ is non-redundant the symmetry group $\mathcal{G}_i \subset \mathcal{G}_{i-1}$ of the fundamental domain $\hat{\mathcal{X}}_i$ is strictly smaller than the symmetry group $\mathcal{G}_{i-1}$ of the previous fundamental domain $\hat{\mathcal{X}}_{i-1}$. The final fundamental domain $\hat{\mathcal{X}}_{|\mathcal{B}|}$ is minimal since its symmetry group $\mathcal{G}_{|\mathcal{B}|} = \langle \varnothing \rangle$ is trivial by the definition of a base.

Since $\mathcal{S}$ is a strong generating set, the orbit $\mathcal{G}_{i-1} z_i$ of base point $z_i \in \mathcal{B}$ under the subgroup $\mathcal{G}_{i-1} = \langle \mathcal{S}_{i-1} \rangle$ can be calculated using Algorithm 6 with generators

$$\mathcal{S}_{i-1} = \{\Theta \in \mathcal{S} : \Theta z_j = z_j, j = 1, \ldots, i-1\}. \tag{6.13}$$

The following theorem shows that the final set $\hat{\mathcal{X}}_{|\mathcal{B}|}$ is a minimal fundamental domain.

Theorem 16. *Algorithm 8 will produce a minimal polytopic fundamental domain $\hat{\mathcal{X}} = \hat{\mathcal{X}}_{|\mathcal{B}|}$ of the set $\mathcal{X}$ with respect to the group $\mathcal{G}$.*

Proof. We will prove by induction that each set

$$\hat{\mathcal{Y}}_{i-1} = \bigcap_{j=i}^{|\mathcal{B}|} \mathcal{F}_{j-1}(z_j) \tag{6.14}$$

for $i = |\mathcal{B}|, \ldots, 1$ is a minimal polytopic fundamental domain of $\mathcal{X}$ for the subgroup $\mathcal{G}_{i-1}$. Thus $\hat{\mathcal{Y}}_0 = \hat{\mathcal{X}}_{|\mathcal{B}|}$ is a minimal polytopic fundamental domain of $\mathcal{X}$ for the group $\mathcal{G} = \mathcal{G}_0$.

First we show that $\hat{\mathcal{Y}}_{|\mathcal{B}|-1} = \mathcal{F}_{|\mathcal{B}|-1}(z_{|\mathcal{B}|})$ is a minimal polytopic fundamental domain of $\mathcal{X}$ under the group $\mathcal{G}_{|\mathcal{B}|-1}$. This follows from Corollary 6. The base point $z_{|\mathcal{B}|}$ is not fixed by any non-identity element of $\mathcal{G}_{|\mathcal{B}|-1}$ since $(\mathcal{G}_{|\mathcal{B}|-1})_{b_{|\mathcal{B}|}} = \mathcal{G}_{|\mathcal{B}|} = \langle \varnothing \rangle$ is trivial by the definition of a base $\mathcal{B}$.

Next we assume $\hat{\mathcal{Y}}_i$ is a minimal polytopic fundamental domain and we will show that

$$\hat{\mathcal{Y}}_{i-1} = \hat{\mathcal{Y}}_i \cap \mathcal{F}_{i-1}(z_i) \tag{6.15}$$

is a minimal polytopic fundamental domain. First note that $\hat{\mathcal{Y}}_{i-1}$ is a polytope since it is the intersection of polytopes. Next we show

$$\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta \hat{\mathcal{Y}}_{i-1} = \mathcal{X}. \tag{6.16}$$

By the properties of union and intersection we have

$$\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta\big(\hat{\mathcal{Y}}_i \cap \mathcal{F}_{i-1}(z_i)\big) \subseteq \mathcal{X} \tag{6.17}$$

since $\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta\hat{\mathcal{Y}}_i = \mathcal{X}$ by the induction hypothesis and $\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta\mathcal{F}_{i-1}(z_i) = \mathcal{X}$ by Theorem 15. We need to show that for all $x \in \mathcal{X}$ there exists $\Theta \in \mathcal{G}_{i-1}$ such that $\Theta x \in \hat{\mathcal{Y}}_i$ and $\Theta x \in \mathcal{F}_{i-1}(z_i)$.

By Theorem 15 for each $x \in \mathcal{X}$ there exists $\Theta_{i-1} \in \mathcal{G}_{i-1}$ such $\Theta_{i-1}x \in \mathcal{F}_{i-1}(z_i)$. Since $\hat{\mathcal{Y}}_i$ is a fundamental domain, for any $\Theta_{i-1}x \in \mathcal{X}$ there exists $\Theta_i \in \mathcal{G}_i$ such that $\Theta_i\Theta_{i-1}x \in \hat{\mathcal{Y}}_i$. But by Lemma 2 we have $\Theta_i\Theta_{i-1}x \in \mathcal{F}_{i-1}(z_i)$ since $\Theta_i \in \mathcal{G}_i$ fixes $\mathcal{F}_{i-1}(z_i)$. Thus $\Theta_i\Theta_{i-1}x \in \hat{\mathcal{X}}_i \cap \mathcal{F}_{i-1}(b_i)$ and

$$\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta\big(\hat{\mathcal{Y}}_i \cap \mathcal{F}_{i-1}(b_i)\big) \supseteq \mathcal{X}. \tag{6.18}$$

Therefore $\bigcup_{\Theta \in \mathcal{G}_{i-1}} \Theta\hat{\mathcal{Y}}_{i-1} = \mathcal{X}$.

Next we show

$$\operatorname{int}\big(\hat{\mathcal{Y}}_{i-1} \cap \Theta\hat{\mathcal{Y}}_{i-1}\big) = \varnothing \tag{6.19}$$

for all $\Theta \neq I \in \mathcal{G}_{i-1}$. By Proposition 2 we can decompose the matrix $\Theta = U_i\Theta_i$ where $\Theta_i \in \mathcal{G}_i \subset \mathcal{G}_{i-1}$ and $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$. Thus we can write the intersection $\hat{\mathcal{Y}}_{i-1} \cap \Theta\hat{\mathcal{Y}}_{i-1}$ as

$$\hat{\mathcal{Y}}_i \cap U_i\Theta_i\hat{\mathcal{Y}}_i \cap \mathcal{F}_{i-1}(z_i) \cap U_i\mathcal{F}_{i-1}(z_i) \tag{6.20}$$

where $\Theta_i\mathcal{F}_{i-1}(z_i) = \mathcal{F}_{i-1}(z_i)$ by Lemma 2 since $\Theta_i \in \mathcal{G}_i$. If $U_i \neq I \in \mathcal{G}_{i-1}/\mathcal{G}_i$ then (6.20) has an empty interior since $\operatorname{int}(\mathcal{F}_{i-1}(z_i) \cap U_i\mathcal{F}_{i-1}(z_i)) = \varnothing$ by Theorem 15. If $U_i = I$ then $\Theta_i = \Theta \neq I \in \mathcal{G}_{i-1}$. Thus (6.20) has an empty interior since $\operatorname{int}(\hat{\mathcal{Y}}_i \cap U_i\Theta_i\hat{\mathcal{Y}}_i) = \varnothing$ by the induction hypothesis that $\hat{\mathcal{Y}}_i$ is a minimal fundamental domain for the group $\mathcal{G}_{i-1}$.

Thus $\hat{\mathcal{Y}}_{i-1}$ is a minimal polytopic fundamental domain for the group $\mathcal{G}_{i-1}$ since it satisfies the conditions of Theorem 13. Therefore $\hat{\mathcal{Y}}_0 = \hat{\mathcal{X}}_{|\mathcal{B}|}$ is a minimal polytopic fundamental domain of the set $\mathcal{X}$ with respect to the group $\mathcal{G} = \mathcal{G}_0$. □

The complexity of Algorithm 8 depends on the length of the base $|\mathcal{B}|$ and the lengths of the orbits $|\mathcal{G}_{i-1}z_i|$. The length of a non-redundant base $|\mathcal{B}| \leq n$ is bounded by the dimension n of the set $\mathcal{X} \subseteq \mathbb{R}^n$ since the only matrix $\Theta \in \mathbb{R}^{n\times n}$ that fixes n linearly independent points $z_i \in \mathcal{B}$ is the identity matrix $\Theta = I$. The length of the orbit $|\mathcal{G}_{i-1}z_i| \leq c$ can be bounded by choosing the base points $z_i = h_i$ to be the normal vectors of facets of the polytope $\mathcal{X} = \{x : h_i^T x \leq 1, i = 1, \dots, c\}$. Since $\mathcal{G} \subseteq \operatorname{Aut}(\mathcal{X})$ permutes the facets of $\mathcal{X}$, each orbit $\mathcal{G}_{i-1}z_i$ of a facet normal vector $z_i = h_i$ will be a subset of facets. Therefore the orbit $\mathcal{G}_{i-1}z_i$ cannot be larger than the total number c of facets. Thus Algorithm 8 computes at most n orbits of length c while Procedure 3 potentially requires computing one orbit of length c^n. The following example demonstrates Algorithm 8.

Example 20. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S} = \{S_1, S_2\}$ defined in* (6.7)*.*

The set $\mathcal{S} = \{S_1, S_2\}$ is a strong generating set for the base $\mathcal{B} = (h_1, h_2)$ where $h_1 = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $h_2 = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ are normal vectors of facets 1 and 2 of the square $\mathcal{X}$ shown in Figure 6.1a. The subgroup $\mathcal{G}_1 = \mathcal{G}_{h_1}$ which fixes facet 1 is generated by the vertical reflection S_2 and the subgroup $\mathcal{G}_2 = (\mathcal{G}_1)_{h_2}$ which fixes facets 1 and 2 is trivial $\mathcal{G}_2 = \langle \varnothing \rangle$. Thus we have the stabilizer subgroup chain

$$\mathcal{D}_4 = \mathcal{G}_0 = \langle \mathbf{S}_1, \mathbf{S}_2 \rangle \supset \mathcal{G}_1 = \langle \mathbf{S}_2 \rangle \supset \mathcal{G}_2 = \langle \varnothing \rangle. \tag{6.21}$$

For each base point $\mathcal{B} = \{h_1, h_2\}$ the algorithm will generate a Voronoi cell $\mathcal{F}_{i-1}(h_i)$ of the point h_i and its orbit $\mathcal{G}_{i-1}h_i$. In iteration $i = 1$, the algorithm calculates the orbit $\mathcal{G}_0 h_1$ of the normal vector h_1 of facet 1 under the group $\mathcal{G}_0 = \langle S_1, S_2 \rangle$

$$\mathcal{G}_0 h_1 = \{h_1, h_2, h_3, h_4\} = \left\{ \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}\text{-}1\\0\end{bmatrix}, \begin{bmatrix}0\\\text{-}1\end{bmatrix} \right\}. \tag{6.22}$$

The Voronoi cell $\mathcal{F}_0(h_1)$ that separates h_1 from its orbit $\mathcal{G}_0 h_1$ is shown in Figure 6.2a. The Voronoi cell $\mathcal{F}_0(h_1)$ is a fundamental domain. However it is not a minimal fundamental domain. The Voronoi cell $\mathcal{F}_0(h_1)$ is symmetric with respect to the stabilizer subgroup $\mathcal{G}_1 = \langle S_2 \rangle$ which is generated by the vertical reflection S_2.

In the next iteration $i = 2$, the algorithm generates a new Voronoi cell $\mathcal{F}_1(h_2)$ that eliminates the symmetry $\mathcal{G}_1 = \langle S_2 \rangle$ of the Voronoi cell $\mathcal{F}_0(h_1)$. The orbit $\mathcal{G}_1 h_2$ of the normal vector h_2 of facet 2 under the subgroup $\mathcal{G}_1 = \langle S_2 \rangle$ is

$$\mathcal{G}_1 h_2 = \{h_2, h_4\} = \left\{ \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\\text{-}1\end{bmatrix} \right\}. \tag{6.23}$$

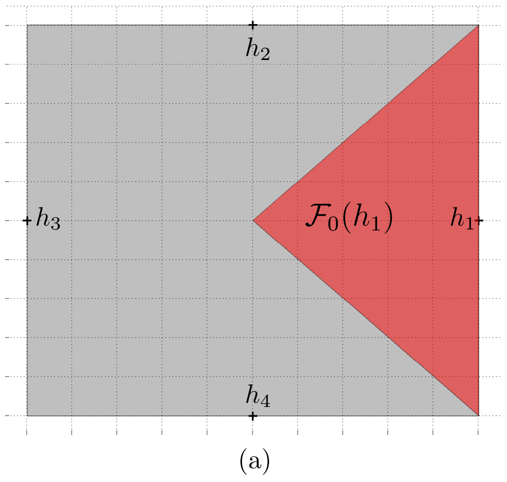
The Voronoi cell $\mathcal{F}_1(h_2)$ that separates h_2 form its orbit $\mathcal{G}_1 h_2$ is shown in Figure 6.2b.

Since $|\mathcal{B}| = 2$ the algorithm is now complete. The set $\hat{\mathcal{X}}_{|\mathcal{B}|} = \mathcal{F}_1(h_2) \cap \mathcal{F}_0(h_1)$ is a minimal fundamental domain of $\mathcal{X}$ with respect to the group $\mathcal{G}$. The minimal fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_1 h_2 \cap \mathcal{F}_0 h_1$ shown in Figure 6.1b is the same fundamental domain produces by Procedure 3 in the previous example.

Using the facet normal vectors h_i as base points has another advantage. The matrices $\Theta \in \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ permute the facets $\Theta h_i = h_{\pi(i)}$ of $\mathcal{X}$. This means we can use the permutation representation of the group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ to compute the orbits of the bases points $z_i = h_i$. Therefore Algorithm 8 never needs to perform expensive matrix computations.

The following example demonstrates how to use the permutation representation to implement Algorithm 8.

Example 21. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S} = \{S_1, S_2\}$ defined in* (6.7)*.*



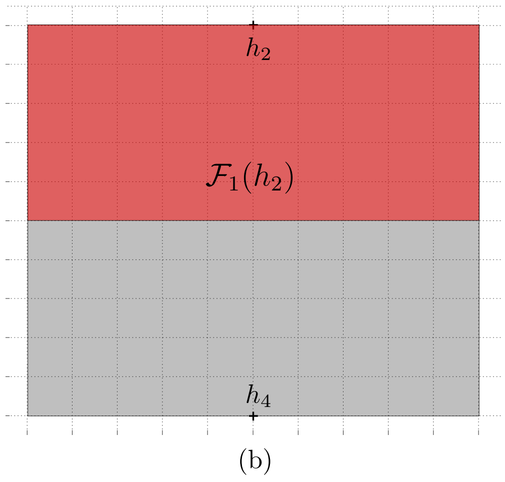


Figure 6.2: Example showing the construction of fundamental domain from Voronoi cells. (a) Voronoi cell $\mathcal{F}_0(h_1)$ of the normal vector h_1 of facets 1 and its orbit $\mathcal{G}_0 h_1 = \{h_1, h_2, h_3, h_4\}$ under the group $\mathcal{G}_0 = \langle S_1, S_2 \rangle$ generated by the rotation matrix S_1 and vertical reflection matrix S_2. (b) Voronoi cell $\mathcal{F}_1(h_2)$ of the normal vector h_2 of facets 2 and its orbit $\mathcal{G}_1 h_2 = \{h_2, h_4\}$ under the group $\mathcal{G}_1 = \langle S_2 \rangle$ vertical reflection matrix S_2.

The symmetry group Aut$(\mathcal{X})$ *can be represented by a permutation group on the set of the facets* $\{1,2,3,4\}$ *of the square* $\mathcal{X}$. *This group is generated by the permutations*

$$s_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix} \tag{6.24a}$$

$$s_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix} \tag{6.24b}$$

where the permutation s_1 *is isomorphic to the* 90 *degree counter-clockwise rotation matrix* S_1 *in (6.7) and the permutation* s_2 *is isomorphic to the vertical reflection matrix* S_2 *in (6.7).*

The bases points $h_1 = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ *and* $h_2 = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ *are the normal vectors of the facets* 1 *and* 2 *of the square* $\mathcal{X}$. *The orbit* $\mathcal{G}_0 h_1$ *of the base point* h_1 *under the matrix group* $\mathcal{G}_0 = \langle S_1, S_2 \rangle$ *corresponds to the orbit* $\mathcal{G}_0(1)$ *of facet* 1 *under the permutation representation group* $\mathcal{G}_0 = \langle s_1, s_2 \rangle$. *Likewise the orbit* $\mathcal{G}_1 h_2$ *of the base point* h_2 *under the matrix group* $\mathcal{G}_1 = \langle S_2 \rangle$ *corresponds to the orbit* $\mathcal{G}_1(2)$ *of facet* 1 *under the permutation representation group* $\mathcal{G}_1 = \langle s_2 \rangle$.

The Voronoi cell $\mathcal{F}_0(h_1)$ *is the set of points* $x \in \mathcal{X}$ *closer to facet* 1 *than the other facets in its orbit* $\mathcal{G}_0(1) = \{1,2,3,4\}$. *Thus constructing the Voronoi cell* $\mathcal{F}_0(h_1)$ *only requires three vector subtractions*

$$\mathcal{F}_0(h_1) = \left\{ x \in \mathcal{X} : (h_1 - h_2)^T x \geq 0, (h_1 - h_3)^T x \geq 0, (h_1 - h_4)^T x \geq 0 \right\}. \tag{6.25}$$

The Voronoi cell $\mathcal{F}_1(h_2)$ *is the set of points* $x \in \mathcal{X}$ *closer to facet* h_2 *than the other facets in the orbit* $\mathcal{G}_1 h_2 = \{2,4\}$. *Constructing the Voronoi cell* $\mathcal{F}_1(h_2)$ *only requires one vector* *subtraction*

$$\mathcal{F}_0(h_1) = \{x \in \mathcal{X} : (h_2 - h_4)^T x \geq 0\}. \tag{6.26}$$

The fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_0(h_1) \cap \mathcal{F}_1(h_2)$ is the intersection of these sets.

The following theorem shows that Algorithm 8 has polynomial computational complexity.

Theorem 17. *Algorithm 8 has computation complexity $O(nc^2)$ where c is the number of facets and n is the dimension of $\mathcal{X}$.*

Proof. The orbit $\mathcal{G}_{i-1}z_i$ is calculated using Algorithm 6. Using permutations $\pi \in \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$, this algorithm has complexity $O\big(|\mathcal{S}_{i-1}||\mathcal{G}_{i-1}z_i|\big)$ where $|\mathcal{S}_{i-1}| \leq c-1$ is the number of generators of $\mathcal{G}_{i-1}$ and $|\mathcal{G}_{i-1}z_i| \leq c$ is the number of points in the orbit. Thus the complexity of computing the orbit is $O(c^2)$.

Computing the Voronoi cell $\mathcal{F}_{i-1}(z_i)$ requires subtracting at most c vectors of n dimensions and therefore has complexity $O(nc)$. Since $\mathcal{X}$ is bounded the number of inequalities is greater than the dimension $c \geq n+1$. Thus calculating the Voronoi cell $\mathcal{F}_{i-1}(z_i)$ has worst-case complexity $O(c^2)$.

Computing the intersection has trivial complexity. It simply requires concatenating the inequalities of $\hat{\mathcal{X}}_i$ and $\mathcal{F}_{i-1}(z_i)$.

Algorithm 8 loops $|\mathcal{B}|$ times. The size of a non-redundant base is bounded $|\mathcal{B}| \leq n$ since the only matrix to fix n linearly independent vectors is the identity matrix $I \in \mathbb{R}^{n \times n}$. Therefore we conclude that Algorithm 8 has complexity $O(nc^2)$. □

6.3 Fundamental Domain Search

In this section we present an algorithm for searching a group $\mathcal{G}$ for an element $\Theta \in \mathcal{G}$ that maps a point $x \in \mathcal{X}$ to its representative $\Theta x \in \hat{\mathcal{X}}$ in the fundamental domain $\hat{\mathcal{X}}$. First we present a naive algorithm for solving this problem and detail the flaws of this method. Using properties of groups and fundamental domain, we present an algorithm that solves this problem in polynomial time.

Existing Method

A straight forward solution to Problem 3 is the exhaustive search procedure presented in Algorithm 9. An obvious drawback to this Algorithm is its poor computation complexity. In the worst-case, Algorithm 9 requires searching every element of the group $\mathcal{G} = \langle \mathcal{S} \rangle$ which can have exponential size $O(|\mathcal{G}|) = c^n$ in the number c of facets and dimension n of the set $\mathcal{X} \subseteq \mathbb{R}^n$.

However there is another more subtle issue with Algorithm 9, we do not have an exhaustive list of the elements of $\mathcal{G}$. Instead we have a list of generators $\mathcal{S}$ from which the elements of $\mathcal{G} = \langle \mathcal{S} \rangle$ can be obtained by multiplying sequences of generators $S \in \mathcal{S}$ of arbitrary length.

Algorithm 9 Fundamental Domain Search - Exhaustive Search

```
1: for all $\Theta \in \mathcal{G} = \langle \mathcal{S} \rangle$ do
2:     if $\Theta x \in \hat{\mathcal{X}}$ then return $\Theta$
3:     end if
4: end for
```

Even for a finite group $\mathcal{G}$ every element $\Theta \in \mathcal{G}$ can have an infinite number of sequences of generators that will generate it.

The problem of enumerating the elements of $\mathcal{G}$ using a set of generators $\mathcal{S}$ requires searching the Cayley graph $\Gamma(\mathcal{S})$ of the group $\mathcal{G} = \langle \mathcal{S} \rangle$. Recall that the nodes of the Cayley graph $\Gamma(\mathcal{S})$ are the elements of $\mathcal{G}$ and two group elements Θ_g and Θ_h are connected by a directed edge (Θ_g, Θ_h) with label $S \in \mathcal{S}$ if $\Theta_h = S\Theta_g$. For small groups a spanning-tree can be created using the orbit graph search given by Algorithm 4. An example of the construction of a spanning-tree is given in the following example.

Example 22. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S} = \{S_1, S_2\}$ defined in* (6.7)*.*

The Cayley graph $\Gamma(\mathcal{S})$ of the group Aut$(\mathcal{X})$ *with generators $\mathcal{S} = \{S_1, S_2\}$ is shown in Figure 6.3b. The $|\mathcal{G}| = 8$ nodes of this graph correspond to elements of the group $\Theta \in \mathcal{G} = \mathcal{D}_4$. The edges are created by the* 90 *degree rotation matrix S_1 and vertical reflection matrix S_2.*

Any path from the root node $\Theta_1 = I$ to the node $\Theta \in \mathcal{G}$ corresponds to a sequence of generators $S_1, S_2 \in \mathcal{S}$ that will produce the matrix $\Theta \in \mathcal{G}$. For instance the matrix $\Theta_7 \in \mathcal{G}$ can be generated by the sequence $\Theta_7 = S_2 S_1$ or $\Theta_7 = S_1^3 S_2$ which correspond to two different paths through the Cayley graph $\Gamma(\mathcal{S})$. Since the graph $\Gamma(\mathcal{S})$ contains loops there are an infinite number of paths to each node $\Theta \in \mathcal{G}$. Therefore there are an infinite number of sequences of generators $\mathcal{S}$ that will produce the matrix $\Theta \in \mathcal{G}$.

Figure 6.3a shows the image of the fundamental domain $\hat{\mathcal{X}}$ under each element $\Theta \in \mathcal{G}$. This figure helps to understand how the nodes of the Cayley graph $\Gamma(\mathcal{S})$ operate on the set $\mathcal{X}$. Each polytope in Figure 6.3a corresponds to a node in the Cayley graph. The fundamental domain $\hat{\mathcal{X}}$ corresponds to the root node $\Theta = I$ of the Cayley graph. If we wished to map a state in the image $\Theta_6\hat{\mathcal{X}}$ to the fundamental domain $\hat{\mathcal{X}}$ we could apply the vertical reflection S_2 and then the counter-clockwise rotation S_1 twice or we could apply the rotation S_1 twice and then the reflection S_2.

Figure 6.3c shows a spanning tree of the Cayley graph $\Gamma(\mathcal{S})$ produced using breath-first search. In this search-tree there is a unique path from the root node $\Theta = I$ to each matrix $\Theta \in \mathcal{G}$. Therefore the elements of $\mathcal{G}$ can be enumerated without repetition using these search-trees.

We now demonstrate the exhaustive search in Algorithm 9 for the breadth-first search-tree shown in Figure 6.3c. Suppose $x = \left[\begin{smallmatrix} -0.75 \\ 0.5 \end{smallmatrix}\right]$ as shown in Figure 6.3a. We exhaustively test each node $\Theta \in \mathcal{G}$ of the search-tree to determine whether $\Theta x \in \hat{\mathcal{X}}$. In the breadth-first

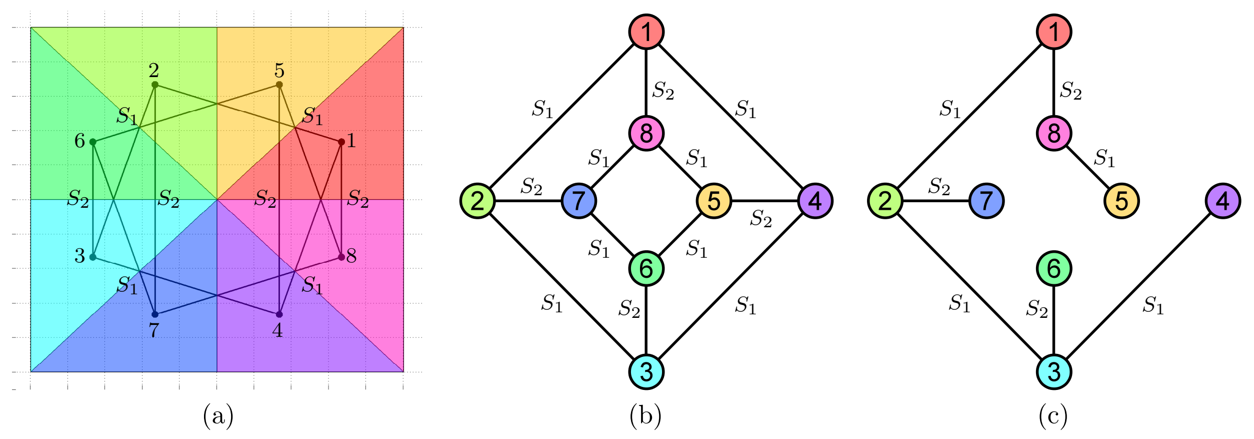


Figure 6.3: Example of exhaustive search using Cayley graph. (a) Fundamental domain $\hat{\mathcal{X}}$ and its images $\Theta\hat{\mathcal{X}}$ under each element $\Theta \in \mathcal{G}$. Cayley graph $\Gamma(\mathcal{S})$ shows how these images $\Theta\hat{\mathcal{X}}$ are related by the generators $S_1, S_2 \in \mathcal{S}$. (b) Cayley graph $\Gamma(\mathcal{S})$ of the dihedral-4 group and generators $\mathcal{S} = \{S_1, S_2\}$ where S_1 is the 90 degree rotation matrix and S_2 is the vertical reflection matrix. (c) Depth-first search-tree of the Cayley graph $\Gamma(\mathcal{S})$.

search-tree the nodes will be tested in the order Θ_1, Θ_2, Θ_8, Θ_3, Θ_7, Θ_5, Θ_4, *and* Θ_6. *Finally we find the matrix*

$$\Theta_6 = S_2 S_1^2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.27}$$

which maps the state $x = \left[\begin{smallmatrix} -0.75 \\ 0.5 \end{smallmatrix}\right]$ *into the fundamental domain* $\hat{\mathcal{X}}$. *Using the breadth-first search-tree shown in Figure 6.3c this search required visiting all* $|\mathcal{D}_4| = 8$ *nodes.*

Unfortunately Algorithm 9 is impractical for large groups. The computational complexity of using Algorithm 4 to search the Cayley graph $\Gamma(\mathcal{S})$ is $O(n^2|\mathcal{S}||\mathcal{G}|)$. As discussed previously the group $\mathcal{G}$ can have exponential size $O(|\mathcal{G}|) = c^n$ in the number c of facets and dimension n of the set $\mathcal{X} \subseteq \mathbb{R}^n$. In the next section we present a polynomial algorithm for solving Problem 3.

Proposed Method

Algorithm 10 describes our method for searching the group $\mathcal{G}$ for a symmetry $\Theta \in \mathcal{G}$ that maps a point $x \in \mathcal{X}$ to its representative in $\Theta x \in \hat{\mathcal{X}}$ in the fundamental domain $\hat{\mathcal{X}}$. This algorithm uses the nested sequence of fundamental domains $\hat{\mathcal{X}}_i = \bigcap_{i=0}^{|\mathcal{B}|} \mathcal{F}_{i-1}(z_i)$ produced by Algorithm 8 to search the group efficiently. The algorithm begins with a point $x \in \mathcal{X}$ in the initial fundamental domain $\hat{\mathcal{X}}_0 = \mathcal{X}$. During each iteration i, the algorithm searches the subgroup $\mathcal{G}_{i-1}$ for the symmetry $U_i \in \mathcal{G}_{i-1}$ that maps the point $\Theta x \in \mathcal{X}_{i-1}$ into the next fundamental domain $\hat{\mathcal{X}}_i = \hat{\mathcal{X}}_{i-1} \cap \mathcal{F}_{i-1}(z_i)$. Since $\mathcal{G}_{i-1}$ is the symmetry group of the non-minimal fundamental domain $\hat{\mathcal{X}}_{i-1} = U_i\hat{\mathcal{X}}_{i-1}$ we only need to find $U_i \in \mathcal{G}_{i-1}$ that maps Θx into the Voronoi cell $\mathcal{F}_{i-1}(z_i)$. Then we have $U_i\Theta x \in \hat{\mathcal{X}}_i = \hat{\mathcal{X}}_{i-1} \cap \mathcal{F}_{i-1}(z_i)$. Furthermore we do not need to search the entire group $\mathcal{G}_{i-1}$ since elements $U_i \in \mathcal{G}_i$ of the subgroup $\mathcal{G}_i \subset \mathcal{G}_{i-1}$ are symmetries of the Voronoi cell $\mathcal{F}_{i-1}(z_i) = U_i\mathcal{F}_{i-1}(z_i)$. Instead we search the non-redundant set $\mathcal{G}_{i-1}/\mathcal{G}_i$ of group elements $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that move the Voronoi cell $\mathcal{F}_{i-1}(z_i) \neq U_i\mathcal{F}_{i-1}(z_i)$.

Algorithm 10 Fundamental Domain Search

```
Input: Point $x \in \mathcal{X}$, Voronoi cells $\mathcal{F}_{i-1}(z_i)$, and group $\mathcal{G}$
Output: Group element $\Theta \in \mathcal{G}$ such that $\Theta x \in \hat{\mathcal{X}}$
1: Initialize matrix $\Theta = I$
2: for $i = 1$ to $|\mathcal{B}|$ do
3:     Find $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ such that $U_i\Theta x \in \mathcal{F}_{i-1}(z_i)$
4:     Update matrix $\Theta = U_i\Theta$
5: end for
```

In terms of group theory, this algorithm breaks up the search of the group $\mathcal{G}$ into a sequence of searches of the cosets $U_i\mathcal{G}_i \subset \mathcal{G}_{i-1}$ of the subgroup $\mathcal{G}_i$ in $\mathcal{G}_{i-1}$ over the chain of subgroups

$$\mathcal{G} = \mathcal{G}_0 \supset \mathcal{G}_1 \supset \cdots \supset \mathcal{G}_{|\mathcal{B}|-1} \supset \mathcal{G}_{|\mathcal{B}|} = \langle \varnothing \rangle. \tag{6.28}$$

During each iteration i, the algorithm searches the coset representatives $\mathcal{G}_{i-1}/\mathcal{G}_i$ for the coset $U_i\mathcal{G}_i$ that contains the matrix $\Theta \in U_i\mathcal{G}_i$ that maps x into the fundamental domain $\hat{\mathcal{X}}$. Since the cosets $U_i\mathcal{G}_i$ divide the group $\mathcal{G}_{i-1}$ into disjoint subsets $\mathcal{G}_{i-1} = \bigsqcup_{U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i} U_i\mathcal{G}_i$ we know that $\Theta \in U_i\mathcal{G}_i$ is contained in exactly one of these cosets. The Voronoi cell $\mathcal{F}_{i-1}(b_i)$ is used to determine when we have found the appropriate coset representative $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$. At termination the algorithm has constructed a symmetry $\Theta = U_{|\mathcal{B}|}\cdots U_1$ that maps the point $x \in \mathcal{X}$ into the fundamental domain $\Theta x \in \hat{\mathcal{X}} = \bigcap_{i=1}^{|\mathcal{B}|} \mathcal{F}_{i-1}(z_i)$.

Algorithm 10 is more efficient that the exhaustive search in Algorithm 9 since it avoids the redundancy of searching the entire subgroup $\mathcal{G}_{i-1}$ rather than its cosets $\mathcal{G}_{i-1}/\mathcal{G}_i$. In the worst-case, this Algorithm 10 requires

$$\sum_{i=1}^{|\mathcal{B}|} |\mathcal{G}_{i-1}|/|\mathcal{G}_i| = cn \tag{6.29}$$

point-in-set tests $U_i\Theta x \in \mathcal{F}_{i-1}(z_i)$ while Algorithm 9 requires

$$|\mathcal{G}| = \prod_{i=1}^{|\mathcal{B}|} |\mathcal{G}_{i-1}|/|\mathcal{G}_i| = c^n \tag{6.30}$$

point-in-set test $\Theta x \in \hat{\mathcal{X}}$.

The following example demonstrate Algorithm 10.

Example 23. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S} = \{S_1, S_2\}$ defined in* (6.7). *A minimal fundamental domain $\hat{\mathcal{X}}$ for $\mathcal{X}$ is shown in Figure 6.1b.*

Consider the point $x = \left[\begin{smallmatrix} -0.75 \\ 0.5 \end{smallmatrix}\right]$. Algorithm 10 maps this point x into the fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_0(h_1) \cap \mathcal{F}_1(h_2)$ by first mapping it into the Voronoi cell $\mathcal{F}_0(h_1)$ and then into the Voronoi cell $\mathcal{F}_1(h_2)$. The Voronoi cells $\mathcal{F}_0(h_1)$ and $\mathcal{F}_1(h_2)$ are shown in Figures 6.2a and 6.2b respectively.

First Algorithm 10 searches the coset representative $U_1\mathcal{G}_1$ of $\mathcal{G}_1$ in $\mathcal{G}_0$ for the symmetry $U_1 \in \mathcal{G}_0/\mathcal{G}_1$ that maps x it into the Voronoi cell $\mathcal{F}_0(h_1)$. In this case $U_1 = S_1^2$. Next Algorithm 10 searches the coset representative $U_2\mathcal{G}_2$ of $\mathcal{G}_2$ in $\mathcal{G}_1$ for the symmetry $U_2 \in \mathcal{G}_1/\mathcal{G}_2$ that maps U_1x into the Voronoi cell $\mathcal{F}_1(h_2)$. In this case $U_2 = S_2$. Thus the symmetry $\Theta = U_2U_1 \in \mathcal{G}$ maps the point $x = \left[\begin{smallmatrix} -0.75 \\ 0.5 \end{smallmatrix}\right]$ into the fundamental domain.

Next we discuss the details of how Algorithm 10 searches the coset representative $\mathcal{G}_{i-1}/\mathcal{G}_i$ for a symmetry $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps Θx into the Voronoi cell $\mathcal{F}_{i-1}(z_i)$. Like Algorithm 9, our algorithm uses graph search techniques to search the group $\mathcal{G}$. However Algorithm 10 replaces the Cayley graph $\Gamma(\mathcal{S})$ with a sequence of Schreier graphs $\Gamma_i(\mathcal{S})$ for $i = 1, \dots, |\mathcal{B}|$ of the cosets of $\mathcal{G}_i$ in the group $\mathcal{G}_{i-1}$. In each iteration i, Algorithm 10 uses a search tree of the i-th Schreier graphs $\Gamma_i(\mathcal{S})$ to find the node $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ such that $U_i\Theta x \in \mathcal{F}_{i-1}(z_i)$.

Searching the Schreier graphs $\Gamma_i(\mathcal{S})$ is equivalent to searching the Cayley graph $\Gamma(\mathcal{S})$ since the Schreier graphs $\Gamma_i(\mathcal{S})$ can be combined to form a spanning subgraph of the Cayley graph $\Gamma(\mathcal{S})$. The i-th Schreier graph $\Gamma_i(\mathcal{S})$ is appended by the root to each node in the $(i-1)$-th Schreier graph Γ_{i-1} for each $i = 1, \dots, |\mathcal{B}|$. The resulting graph is connected and spans the nodes of the Cayley graph $\Gamma(\mathcal{S})$. Thus a spanning search-tree of the sequence of Schreier graphs $\Gamma_i(\mathcal{S})$ is also a spanning search-tree of the Cayley graph $\Gamma(\mathcal{S})$.

Searching a sequence of Schreier graphs $\Gamma_i(\mathcal{S})$ is more efficient than searching the Cayley graph $\Gamma(\mathcal{S})$ since the Schreier graphs are significantly smaller and can be searched independently in sequence. Furthermore the number of node in the each Schreier graph is bounded. Since $\mathcal{G}_i$ is the subgroup of $\mathcal{G}_{i-1}$ that stabilizes the i-th base point $z_i \in \mathcal{B}$, the Schreier graph $\Gamma_i(\mathcal{S})$ is also the graph of the orbit $\mathcal{G}_{i-1}z_i$ of $z_i \in \mathcal{B}$. Since the base points $\mathcal{B}$ are facets of $\mathcal{X}$ their orbits are bounded by the number of facets c.

The following example demonstrates the Schreier graph and Algorithm 10.

Example 24. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$ shown in Figure 6.1a. The square $\mathcal{X}$ is symmetric with respect to the dihedral-4 group $\mathcal{G} = \langle \mathcal{S} \rangle$ generated by $\mathcal{S} = \{S_1, S_2\}$ defined in* (6.7).

The set $\mathcal{S} = \{S_1, S_2\}$ is a strong generating set for the base $\mathcal{B} = \{h_1, h_2\}$ where $h_1 = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ and $h_2 = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ are the normal vectors of facets 1 *and* 2 *respectively. This base produces the chain of stabilizer subgroups*

$$\mathcal{D}_4 = \mathcal{G}_0 = \langle \mathbf{S}_1, \mathbf{S}_2 \rangle \supset \mathcal{G}_1 = \langle \mathbf{S}_2 \rangle \supset \mathcal{G}_2 = \langle \varnothing \rangle. \tag{6.31}$$

The Schreier graph $\Gamma_1(\mathcal{S})$ of the cosets $U\mathcal{G}_1$ of $\mathcal{G}_1$ in $\mathcal{G}_0$ is shown in Figure 6.4a. Since $\mathcal{G}_1$ is the subgroup of $\mathcal{G}_0 = \mathcal{G}$ that stabilizes $h_1 = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$, this graph corresponds to the graph of *the orbit $\mathcal{G}_0 h_1$ of the base point $h_1 = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ under the group $\mathcal{G}_0 = \langle S_1, S_2\rangle$. Searching this graph corresponds to searching for the coset representative $U \in \mathcal{G}_0/\mathcal{G}_1$ that contains the matrix $\Theta \in U\mathcal{G}_1 \subset \mathcal{G}_0$ such that $\Theta x \in \hat{\mathcal{X}}$.*

The Schreier graph $\Gamma_2(\mathcal{S})$ of the cosets $U\mathcal{G}_2$ of $\mathcal{G}_2$ in $\mathcal{G}_1$ is shown in Figure 6.4b. Since $\mathcal{G}_2 = \langle\varnothing\rangle$ is trivial this is also the Cayley graph of the subgroup $\mathcal{G}_1$ that stabilizes the base point $h_2 = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$.

The Schreier graph $\Gamma_2(\mathcal{S})$ can be appended to each node of the Schreier graph $\Gamma_1(\mathcal{S})$ to form the graph shown in Figure 6.4c. This graph is a spanning subgraph of the Cayley graph $\Gamma(\mathcal{S})$ shown in Figure 6.3b. Clearly any spanning search-tree of the graph in Figure 6.4c will also be a spanning search-tree of the Cayley graph $\Gamma(\mathcal{S})$ shown in Figure 6.3b. Therefore searching the Schreier graphs $\Gamma_i(\mathcal{S})$ for $i = 1, \dots, |\mathcal{B}|$ is equivalent to searching the Cayley graph $\Gamma(\mathcal{S})$.

We now demonstrate Algorithm 10 for the point $x = \left[\begin{smallmatrix}-0.75\\0.5\end{smallmatrix}\right]$. First we search the coset representatives $\mathcal{G}_0/\mathcal{G}_1$ for a matrix U_1 that maps $x \in \mathcal{X}$ into the Voronoi cell $\mathcal{F}_0(h_1)$ shown in Figure 6.4a. This is equivalent to performing a depth-first search on the Schreier graph $\Gamma_1(\mathcal{S})$ in Figure 6.4a. We obtain the matrix $U_1 = S_1^2$ which maps $x = \left[\begin{smallmatrix}-0.75\\0.5\end{smallmatrix}\right]$ to $U_1 x = \left[\begin{smallmatrix}0.75\\-0.5\end{smallmatrix}\right] \in \mathcal{F}_0(b_1)$.

In the next iteration we search for a matrix $U_2 \in \mathcal{G}_1/\mathcal{G}_2$ that maps $U_1 x = S_1^2 x$ into the set $\mathcal{F}_1(b_2)$ shown in Figure 6.4b. This is equivalent to searching the Schreier graph $\Gamma_2(\mathcal{S})$ shown in Figure 6.4b. We obtain the matrix $U_2 = S_2$ which maps $U_1 x = \left[\begin{smallmatrix}0.75\\-0.5\end{smallmatrix}\right]$ to $U_2 U_1 x = \left[\begin{smallmatrix}0.75\\0.5\end{smallmatrix}\right] \in \mathcal{F}_1(h_2)$. Since $U_2 \in \mathcal{G}_1/\mathcal{G}_2 \subseteq \mathcal{G}_1$ it keeps $U_1 x \in \mathcal{F}_0(h_1)$ inside the set $\mathcal{F}_0(h_1) = U_2\mathcal{F}_0(h_1)$. Thus we have found the matrix $\Theta = S_2 S_1^2$ which maps the point $x = \left[\begin{smallmatrix}-0.75\\0.5\end{smallmatrix}\right]$ into the fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_1(h_2) \cap \mathcal{F}_0(h_1)$. This is the same matrix found using the exhaustive search in the previous example.

In this small example searching the Schreier graphs $\Gamma_i(\mathcal{S})$ is only slightly more efficient than exhaustively searching the Cayley graph $\Gamma(\mathcal{S})$. In the worst-case, searching the Schreier graphs requires visiting $|\mathcal{G}_0|/|\mathcal{G}_1| + |\mathcal{G}_1|/|\mathcal{G}_2| = 4 + 2 = 6$ nodes rather than the $(|\mathcal{G}_0|/|\mathcal{G}_1|) \times (|\mathcal{G}_1|/|\mathcal{G}_2|) = 4 \times 2 = 8$ nodes that are visited in Cayley graph search. For larger problems the computational savings becomes more significant. We will discuss the computational complexity of this algorithm in detail in the next section.

The following theorem shows that Algorithm 10 maps x into the fundamental domain $\hat{\mathcal{X}}$.

Theorem 18. *Algorithm 10 will produce a matrix $\Theta \in \mathcal{G}$ that maps the point $x \in \mathcal{X}$ into the fundamental domain $\hat{\mathcal{X}}$.*

Proof. First we show that at each iteration i the point Θx is mapped into $\mathcal{F}_{i-1}(z_i)$ for $i = 1, \dots, |\mathcal{B}|$. The matrix $\Theta = U_{i-1} \cdots U_1$ is an element of $\mathcal{G}_0/\mathcal{G}_i \subseteq \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$. Thus $\Theta x \in \mathcal{X}$. By Theorem 15 there exists a matrix $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ such that $U_i \Theta x \in \mathcal{F}_{i-1}(z_i)$.

Next we show that after iteration i the point $\Theta x = U_i \cdots U_1 x$ does not leave $\mathcal{F}_{i-1}(z_i)$. At iteration $j > i$ the matrix $U_j \cdots U_{i+1}$ is an element of $\mathcal{G}_{i+1}/\mathcal{G}_j \subset \mathcal{G}_i$. Thus by Lemma 2 we have $U_j \cdots U_{i+1} U_i \cdots U_1 x \in \mathcal{F}_{i-1}(z_i)$.

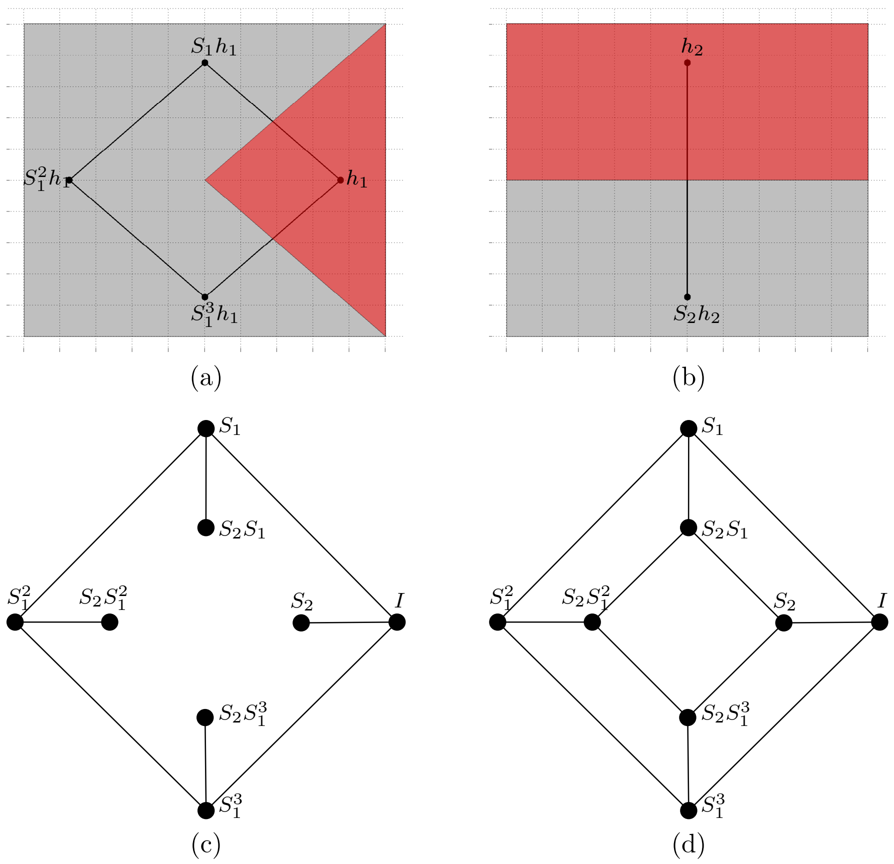


Figure 6.4: Example of group search using Schreier graphs. (a) Schreier graph $\Gamma_1(\mathcal{S})$ of the orbit $\mathcal{G}_0h_1$ of the normal vector h_1 of facet 1 under the dihedral-4 group $\mathcal{G} = \mathcal{G}_0 = \langle S_1, S_2\rangle$. This is also the graph of the cosets $U_1\mathcal{G}_1$ of the subgroup $\mathcal{G}_1$ in $\mathcal{G}_0$. (b) Schreier graph $\Gamma_2(\mathcal{S})$ of the orbit $\mathcal{G}_1h_2$ of the normal vector h_2 of facet 2 under the stabilizer subgroup $\mathcal{G}_1 = \langle S_2\rangle$. This is also the graph of the cosets $U_2\mathcal{G}_2$ of the trivial subgroup $\mathcal{G}_2 = \langle\varnothing\rangle$ in $\mathcal{G}_1$. (c) Spanning subgraph of the Cayley graph $\Gamma(\mathcal{S})$ formed by appending the Schreier graph $\Gamma_2(\mathcal{S})$ by the root to each node of the Schreier graph $\Gamma_1(\mathcal{S})$. This graph is a spanning-subgraph of the Cayley graph $\Gamma(\mathcal{S})$. (d) Cayley graph $\Gamma(\mathcal{S})$ of the dihedral-4 group.

Since $\hat{\mathcal{X}} = \bigcap_{i=1}^{|\mathcal{B}|} \mathcal{F}_{i-1}(b_i)$, we conclude that Algorithm 10 maps $x \in \mathcal{X}$ to the fundamental domain $\hat{\mathcal{X}}$. $\square$

Permutation Search

In this section we present a variant of Algorithm 10 that uses the permutation representation of the group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ to improve computational complexity. Algorithm 11 uses Schreier vectors and the permutation representation of the group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ to efficiently compute a matrix $\Theta \in \mathcal{G}$ that maps the point x into the fundamental domain. This algorithm has computational complexity $O(nc^2)$ and memory complexity $O(c^2)$.

Algorithm 11 exploits the structure of the Voronoi cells $\mathcal{F}_{i-1}(h_i)$ given in Theorem 15. The Voronoi cell $\mathcal{F}_{i-1}(h_i)$ is the set of points x such that $h_i^T x \geq h_i^T \Theta x$ for all $\Theta \in \mathcal{G}$. Or equivalently the $1 - h_i^T x \leq 1 - h_j^T x$ for any $h_j \in \mathcal{G}_{i-1}h_i$ in the orbit of h_i. In other words, the point $x \in \mathcal{X}$ is in the Voronoi cell $\mathcal{F}_{i-1}(h_i)$ if it is closer $1 - h_i^T x \leq 1 - h_j^T x$ to the facet i than any other facets j in the orbit $\mathcal{G}_{i-1}h_i$ of facet i. Thus we can determine which cell j of the Voronoi diagram of the orbit $\mathcal{G}_{i-1}h_i$ contains the point $x \in \mathcal{X}$ by finding the closest facet $j = \mathrm{argmin}_k\{1 - h_k^T x : h_k \in \mathcal{G}_{i-1}h_i\}$ to the point $x \in \mathcal{X}$. The orbit

$$\mathcal{G}_{i-1}h_i = \{h_j : \texttt{generator}_i(j) \neq \varnothing\} \tag{6.32}$$

of the i-th facet can be determined by reading the non-null elements of the Schreier vector $\texttt{generator}_i$ for the orbit $\mathcal{G}_{i-1}h_i$.

Once we know which cell of the Voronoi diagram contains the point $x \in \mathcal{X}$ we can construct a matrix $U \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps $x \in \mathcal{X}$ into the Voronoi cell $\mathcal{F}_{i-1}(h_i)$ using a Schreier vector. By Theorem 14 each matrix $U \in \mathcal{G}_{i-1}/\mathcal{G}_i$ permutes the facets of $\mathcal{X}$ and therefore permutes the cells of the Voronoi diagram of the facet orbit $\mathcal{G}_{i-1}h_i$. Thus if $x \in \mathcal{X}$ is closest to facet j then we can map $x \in \mathcal{X}$ into the Voronoi cell $\mathcal{F}_{i-1}(h_i)$ by finding a matrix $U \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps facets j to facet i. This is equivalent to searching the permutation representation of $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$ for a permutation $u \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps j to i. This can be done efficiently using Schreier vectors and Algorithm 5.

Recall that a Schreier vector is a data-structure that represents a search-tree in a Schreier graph. The Schreier vector $\texttt{generators}_i$ tells us how to combine generators $s \in \mathcal{S}$ to produce a permutation $u = s_1 s_2 \cdots \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps j to $i = u(j)$. Each entry $\texttt{generators}_i(j)$ of the Schreier vector contains the generator $s \in \mathcal{S}$ that is the label of the edge entering node j in the Schreier graph. The Schreier vector allows us to backtrack through the graph $\Gamma_i(\mathcal{S})$ of the orbit $\mathcal{G}_{i-1}h_i$ from node $j \in \mathcal{G}_{i-1}h_i$ to the root node i. Each base point $h_i \in \mathcal{B}$ has a Schreier vector $\texttt{generators}_i$. The Schreier vectors can be created using Algorithm 4.

Finally Algorithm 11 improves on Algorithm 10 by not calculating the matrix $U \in \mathcal{G}_{i-1}/\mathcal{G}_i$ at each iteration i. Instead Algorithm 11 calculates the final matrix $\Theta = U_{|\mathcal{B}|} \cdots U_1$ using corresponding permutation $\theta = u_{|\mathcal{B}|} \cdots u_1$ and the isomorphism (6.4). This avoids having to perform expensive matrix computations at each iteration.

Implementing Algorithm 11 requires storing the half-spaces h_i for $i = 1, \dots, c$ of the set $\mathcal{X}$, the generators $\mathcal{S}$ of $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{X})$, and the Schreier vectors $\texttt{generators}_i$ for $i =$

$1, \dots, |\mathcal{B}|$. Storing the half-spaces requires $O(nc)$ memory. Storing the generators requires $O(c^2)$ memory since each generator $s \in \mathcal{S}$ is a permutation of length c and there are at most $c-1$ generators. Storing the Schreier vectors requires $O(nc)$ memory since each Schreier vector is of length c and there are at most $|\mathcal{B}| \leq n$ Schreier vectors. Therefore Algorithm 11 has memory complexity $O(c^2)$.

Algorithm 11

```
Input: Point $x \in \mathcal{X}$, group generators $\mathcal{S}$, Schreier vectors $\texttt{generators}_i$, facets $h_i$
Output: Group element $\Theta \in \mathcal{G}$ such that $\Theta x \in \hat{\mathcal{X}}$
1: calculate initial distances to facets $d_j = 1 - h_j^T x$
2: initialize permutation $\theta = e$
3: for $i = 1$ to $|\mathcal{B}|$ do
4:     read the orbit $\mathcal{G}_{i-1} h_i = \{h_j : \texttt{generators}_i(j) \neq \varnothing\}$ of facet $h_i \in \mathcal{B}$
5:     calculate the closest facet $j^\star = \text{argmin}_{h_j \in \mathcal{G}_{i-1} h_i} d_j$ to the point $\Theta x \in \mathcal{X}$
6:     use $\texttt{generators}_i$ to find permutation $u_i \in \mathcal{G}_{i-1}/\mathcal{G}_i$ that maps facet $j^\star$ to facet $i$
7:     update distances $d_j^+ = d_{u(j)}$ and permutation $\theta^+ = u_i \theta$
8: end for
9: calculate $\Theta x = H^T(1-d)$ or calculate $\Theta = H^T \Pi_\theta H$.
```

The following example illustrates the use of Algorithm 11.

Example 25. *Consider the square $\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x \leq 1\}$. This set is symmetric with respect to the dihedral-4 group. The permutation representation of the dihedral-4 group is generated by the permutations*

$$s_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \end{pmatrix} \tag{6.33a}$$

$$s_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}. \tag{6.33b}$$

The permutation s_1 corresponds to a 90 degree counter-clockwise rotation of the square and the permutation s_2 corresponds to a vertical reflection of the square. The generators $\mathcal{S} = \{s_1, s_2\}$ are strong generators for the base $\mathcal{B} = \{1, 2\} \subset \{1, 2, 3, 4\}$.

The Schreier vectors $\texttt{generators}_1$ *and* $\texttt{generators}_2$ *for orbit $\mathcal{G}_0 h_1$ of facet h_1 under the group $\mathcal{G}_0 = \langle s_1, s_2 \rangle$ and the orbit $\mathcal{G}_1 h_2$ of the facet h_2 under the group $\mathcal{G}_1 = \langle s_2 \rangle$ respectively* *are*

$$\texttt{generators}_1 = \begin{cases} e & \textit{for } y = 1 \\ s_1 & \textit{for } y = 2 \\ s_1 & \textit{for } y = 3 \\ s_1 & \textit{for } y = 4 \end{cases} \tag{6.34a}$$

$$\texttt{generators}_2 = \begin{cases} \varnothing & \textit{for } y = 1 \\ e & \textit{for } y = 2 \\ \varnothing & \textit{for } y = 3 \\ s_2 & \textit{for } y = 4 \end{cases}. \tag{6.34b}$$

These Schreier vectors represent trees of the graphs shown in Figures 6.4a and 6.4b respectively.

We now demonstrate Algorithm 11 for the point $x = \left[\begin{smallmatrix}-0.75\\0.5\end{smallmatrix}\right]$*. First we calculate the distances from* $x \in \mathcal{X}$ *to each facet* $d = 1 - Hx = [1.75, 0.5, 0.25, 1.5]^T$*. The permutation* θ *is initialized as the identity permutation*

$$\theta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}. \tag{6.35}$$

In iteration $i = 1$ *we are trying to map the point* $x \in \mathcal{X}$ *into the Voronoi cell* $\mathcal{F}_0(h_1)$ *shown in Figure 6.4a. The Voronoi cell* $\mathcal{F}_0(h_1) = \{x : h_1^T x \geq h_j^T x, j = 2, 3, 4\}$ *is the set of points* $x \in \mathcal{X}$ *closer to facet* h_1 *than facets* $\mathcal{G}_0 h_1 \setminus \{h_1\} = \{h_2, h_3, h_4\}$*. The orbit* $\mathcal{G}_0 h_1$ *can be read from the Schreier vector* $\texttt{generators}_1$ *which has no null elements.*

The point $x \in \mathcal{X}$ *is not in the Voronoi cell* $\mathcal{F}_0(h_1)$ *since it is closest* $d_3 \leq d_1, d_2, d_4$ *to facet* $h_3 \in \mathcal{G}_0 h_1$*. Therefore we need a permutation* $u \in \mathcal{G}_0/\mathcal{G}_1$ *that maps facet 3 to facet 1. This can be constructed using the Schreier vector* $\texttt{generators}_1$*.*

Using Algorithm 5 with $\texttt{generators}_1$ *and* $y = 3$ *we obtain the permutation*

$$u_1 = s_1 s_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix} \in \mathcal{G}_0/\mathcal{G}_1 \tag{6.36}$$

which maps facet 3 to facet 1. This permutation corresponds to the 180 degrees rotation matrix

$$U_1 = H^T \Pi_{u_1} H = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \in \mathcal{G}_0/\mathcal{G}_1. \tag{6.37}$$

Applying the transformation $\Theta = U_1$ *maps the point* $x \in \mathcal{X}$ *into the Voronoi cell* $\mathcal{F}_0(h_1)$*. The updated distances* $d_j^+ = 1 - h_j^T \Theta x_0 = 1 - h_j^T U_1 x_0$ *from* $\Theta x_0 = U_1 x_0 \in \mathcal{X}$ *to each of the facets* h_j *can be calculated using the permutation* u

$$d^+ = \Pi_{u_1} d = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1.75 \\ 0.5 \\ 0.25 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 1.5 \\ 1.75 \\ 0.5 \end{bmatrix} \tag{6.38}$$

where Π_{u_1} is the permutation matrix for the facet permutation u_1. Finally we update the permutation θ corresponding to transformation $\Theta = U_1$

$$\theta^+ = u_1\theta = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}. \tag{6.39}$$

In iteration $i = 2$ we are trying to map the point $\Theta x = U_1 x \in \mathcal{X}$ into the Voronoi cell $\mathcal{F}_1(h_2)$ shown in Figure 6.4b. The Voronoi cell $\mathcal{F}_1(h_2) = \{x \in \mathcal{X} : h_2^T x \geq h_4^T x\}$ is the set of points $x \in \mathcal{X}$ closer to facet h_2 than facet h_4 where $\mathcal{G}_1 h_2 = \{h_2, h_4\}$. The orbit $\mathcal{G}_1 h_2$ can be read from the Schreier vector `generators`$_2$ *which has elements 2 and 4 as non-null.*

The point $U_1 x_0 \in \mathcal{X}$ is not in the Voronoi cell $\mathcal{F}_1(h_2)$ since it is closest $d_4 \leq d_2$ to facet $h_4 \in \mathcal{G}_1 h_2$. Therefore we need a permutation $u \in \mathcal{G}_1/\mathcal{G}_2$ that maps facets 4 to facet 2. This can be constructed using the Schreier vector `generators`$_2$.

Using Algorithm 5 with `generators`$_2$ *and $y = 4$ we obtain the permutation*

$$u_2 = s_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix} \in \mathcal{G}_1/\mathcal{G}_2 \tag{6.40}$$

which maps facet 4 to facet 2. This permutation corresponds to the vertical reflection matrix

$$U_2 = H^T \Pi_{u_2} H = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \in \mathcal{G}_1/\mathcal{G}_2. \tag{6.41}$$

Applying the transformation U_2 maps the point $\Theta x = U_1 x \in \mathcal{X}$ into the Voronoi cell $\mathcal{F}_1(h_2)$. Since $U_2 \in \mathcal{G}_1/\mathcal{G}_2$ fixes the Voronoi cell $\mathcal{F}_0(h_1) = \Theta\mathcal{F}_1(h_2)$ for all $\Theta \in \mathcal{G}_1$ we have that $U_2 U_1 x$ is in the fundamental domain

$$U_2 U_1 x \in \hat{\mathcal{X}} = \mathcal{F}_1(h_2) \cap \mathcal{F}_0(h_1). \tag{6.42}$$

In Algorithm 11 the matrix $\Theta = U_2 U_1$ is computed last from the permutation

$$\theta = u_2 u_1 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 4 \end{pmatrix} \tag{6.43}$$

which produces the matrix $\Theta = H^T \Pi_\theta H = H^T \Pi_{u_2} \Pi_{u_1} H$

$$\Theta = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{6.44}$$

This matrix $\Theta \in \mathcal{G}$ is the horizontal reflection which maps the point $x = \begin{bmatrix} -0.75, 0.5 \end{bmatrix}$ into the fundamental domain $\hat{\mathcal{X}} = \{x \in \mathcal{X} : x_1, x_2 \geq 0, x_1 \geq x_2\} = \mathcal{F}_1(h_2) \cap \mathcal{F}_0(h_1)$.

The correctness of Algorithm 11 is proven by the following theorem.

Theorem 19. *Algorithm 11 is equivalent to Algorithm 10.*

Proof. First we will show that if the distances d_j from Θx to each facet j are correct $d_j = 1 - h_j^T \Theta x$ then $U = H^T \Pi_u H$ maps Θx into the Voronoi cell $\mathcal{F}_{i-1}(h_i)$ where $u \in \mathcal{G}_{i-1}/\mathcal{G}_i$ is a representative function that maps i to j. By definition of $d_{j^\star} = \operatorname{argmin}_{h_j \in \mathcal{G}_{i-1} h_i} 1 - h_j^T \Theta x$, we have

$$d_{j^\star} = 1 - h_{j^\star}^T \Theta x = 1 - h_{u(i)}^T \Theta x = 1 - h_i^T U \Theta x \leq 1 - h_j^T \Theta x = d_j \tag{6.45}$$

for all $h_j \in \mathcal{G}_{i-1} h_i$ where $h_{u(i)} = h_i^T U$ since $\Pi_u H = HU$ by Theorem 14. Since $U \in \mathcal{G}_{i-1}/\mathcal{G}_i \subseteq \mathcal{G}_{i-1}$ we have $U\mathcal{G}_{i-1} = \mathcal{G}_{i-1}$ and thus

$$h_i^T U \Theta x \geq h_j^T U \Theta x \tag{6.46}$$

for all $h_j \in \mathcal{G}_{i-1} h_i$. By Theorem 15 this means U maps Θx into the Voronoi cell $\mathcal{F}_{i-1}(h_i) = \{x : h_i^T x \geq h_j^T x, \forall h_j \in \mathcal{G}_{i-1} h_i\}$.

Next we will prove by induction that at iteration i the updated distances $d_j = d_{u(j)}$ are correct $d = 1 - H\Theta x$ where $\Theta = U_{i-1} \cdots U_1$. This is clearly true in the base case $i = 1$ since $\Theta = I$ and the distances d_j are initialized as $d_j = 1 - h_j^T x$. For iteration $i > 1$ the updated distances d satisfy

$$d_{u(j)} = 1 - h_{u(j)}^T \Theta x = 1 - h_j^T U \Theta x. \tag{6.47}$$

Thus $d = 1 - H\Theta x = 1 - HU_i \cdots U_1 x$ at the end of each iteration $i = 1, \ldots, |\mathcal{B}|$.

Finally we note $\Theta x = H^T(1 - d) = H^T H \Theta x$ since the polytope $\mathcal{X}$ is regularized $H^T H = I$. □

The computational of Algorithm 11 is given by the following proposition.

Proposition 17. *Calculating $\Theta x \in \hat{\mathcal{X}}$ has complexity $O(nc)$ and calculating Θ has complexity $O(nc^2)$ where c is the number of facets and n is the dimension of the set $\mathcal{X} \subseteq \mathbb{R}^n$.*

Proof. Calculating the initial distances $d_j = 1 - h_j^T x$ has complexity $O(nc)$. Calculating the orbit $\mathcal{G}_{i-1} h_i$ using the Schreier vector $\texttt{generators}_i$ has trivial complexity $O(1)$. Calculating the minimum distance $d_{j^\star}$ has complexity $O(c)$. Calculating the representative function has complexity $O(|\mathcal{G}_{i-1} h_i|) \leq c$. And updating the distances d_j has trivial complexity $O(1)$. Thus the for-loop of Algorithm 11 has complexity $O(nc)$.

Finally calculating $\Theta x = H^T(1 - d)$ has complexity $O(nc)$. Thus finding Θx has complexity $O(nc)$. On the other hand calculating $\Theta = H^T \Pi_\theta H$ has $O(cn^2)$ complexity. □

6.4 Sorting using Fundamental Domains

In this section we will show that the set of sorted states is a fundamental domain that can be constructed using Algorithm 8. Furthermore Algorithm 10 can be used as a sorting algorithm. In this context Algorithm 10 is equivalent to bubble search [34].

Consider the set $\mathcal{X} = \rho[-1,1]^n$ for large radius $\rho \to \infty$. A symmetry group $\mathcal{G} \subseteq \text{Aut}(\mathcal{X})$ of this set is the symmetric group $\mathfrak{S}_n \subseteq \text{Aut}(\mathcal{X})$ which is the set of all permutation matrices $\Pi \in \mathbb{R}^{n\times n}$. We will use Algorithm 8 to show the set of sorted states is a fundamental domain of this set.

Consider the base $\mathcal{B} = \{e_1, e_2, \ldots, e_{n-1}\}$ where e_i is the i-th standard basis vector. This base produces the stabilizer chain

$$\mathcal{G} = \mathfrak{S}_n \supset \mathfrak{S}_{n-1} \supset \cdots \supset \mathfrak{S}_2 \supset \mathfrak{S}_1 = \langle \varnothing \rangle \tag{6.48}$$

where $\mathfrak{S}_{n-i}$ is the subgroup of permutation matrices $\Pi \in \mathbb{R}^{n\times n}$ that fix $\Pi e_j = e_j$ the first $i \geq j$ basis vectors $e_1, \ldots, e_i$. Note that the indexing of these stabilizer subgroups is the reverse of our standard notation $\mathcal{G}_i = \mathcal{S}_{n-i}$.

We will execute Algorithm 8 with this base. The orbit $\mathcal{G}_{i-1}e_i$ of the base point e_i under the subgroup $\mathcal{G}_{i-1} = \mathfrak{S}_{n-i+1}$ is the set $\mathcal{G}_{i-1}e_i = \{e_i, \ldots, e_n\}$. The Voronoi cell $\mathcal{F}_{i-1}(e_i)$ that separates e_i from its orbit $\mathcal{G}_{i-1}e_i = \{e_i, \ldots, e_n\}$ is the set

$$\begin{aligned}\mathcal{F}_{i-1}(e_i) &= \{x \in \mathcal{X} : (e_i - e_j)^T x \geq 0, \text{ for } j = i, \ldots, n\} \\ &= \big\{x \in \mathcal{X} : x_i \geq x_j, \text{ for } j = i, \ldots, n\big\}.\end{aligned} \tag{6.49}$$

This set says that x_i is larger than $x_{i+1}, \ldots, x_n$. According to Algorithm 8 the set $\hat{\mathcal{X}} = \bigcap_{i=1}^{\mathcal{B}} \mathcal{F}_{i-1}(e_i)$ is a fundamental domain

$$\begin{aligned}\hat{\mathcal{X}} &= \bigcap_{i=1}^{n} \mathcal{F}_{i-1}(e_i) = \bigcap_{i=1}^{n} \big\{x : x_i \geq x_j, \text{ for } j = i, \ldots, n\big\} \\ &= \big\{x \in \mathcal{X} : x_1 \geq x_2 \geq \cdots \geq x_n\big\}.\end{aligned} \tag{6.50}$$

This is the set of sorted states $\hat{\mathcal{X}} = \big\{x \in \mathcal{X} : x_1 \geq x_2 \geq \cdots \geq x_n\big\}$.

Using this base $\mathcal{B} = \{e_1, e_2, \ldots, e_{n-1}\}$ we can find a symmetry $\Theta = \Pi \in \mathfrak{S}_n = \mathcal{G}$ that sorts the elements $x \in \mathcal{X}$ (i.e. $\Theta x \in \hat{\mathcal{X}}$) using Algorithm 10.

In iteration i the Algorithm 10 finds a symmetry $U_i \in \mathcal{G}_{i-1}/\mathcal{G}_i = \mathfrak{S}_{n-i+1}/\mathfrak{S}_{n-i}$ that maps $\Theta x = U_{i-1} \cdots U_0 x$ into the Voronoi cell $\mathcal{F}_{i-1}(e_i) = \{x \in \mathcal{X} : x_i \geq x_{i+1}, \ldots, x_n\}$. The set $\mathcal{G}_{i-1}/\mathcal{G}_i = \mathfrak{S}_{n-i+1}/\mathfrak{S}_{n-i}$ is the set of permutation matrices $\Pi \in \mathbb{R}^{n\times n}$ that permute the base vectors $\{e_i, \ldots, e_n\}$ modulo the permutations $\mathcal{S}_{n-i}$ that fix base vector e_i. This set has size $|\mathfrak{S}_{n-i+1}/\mathfrak{S}_{n-i}| = (n-i+1)!/(n-i)! = n-i+1 \leq n$. The Voronoi cell $\mathcal{F}_{i-1}(e_i)$ is the set where x_i is the largest element of the subset $\{x_i, \ldots, x_n\}$. Thus after iteration i Algorithm 10 has mapped the point x into the non-minimal fundamental domain $\hat{\mathcal{X}}_i = \bigcap_{j=1}^{i} \mathcal{F}_{j-1}(e_j)$ where the first i elements $x_1, \ldots, x_i$ of x are sorted $x_1 \geq \cdots \geq x_i$. At termination $i = |\mathcal{B}| = n$ the algorithm has sorted all n elements of the vector $x \in \mathbb{R}^n$. This sorting algorithm is equivalent to the bubble search [34]. It has computational complexity $O(n^2)$.

Chapter 7

Symmetric Explicit Model Predictive Controllers

In this chapter we present two novel explicit model predictive control designs that exploit symmetry to reduce memory complexity. The orbit controller reduces memory by eliminating pieces of the controller that are symmetrically redundant. The fundamental domain controller reduces memory by eliminating a portion of the state-space that is symmetrically redundant. We describe how to synthesize and implement these controllers.

In this chapter we consider the following model predictive control problem for a constrained linear time-invariant system with polytopic constraints, and a linear or quadratic cost function

$$
\begin{aligned}
J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & p(x_N) + \sum\nolimits_{k=0}^{N-1} q(x_k, u_k) && \text{(7.1a)}\\
\text{subject to} \quad & x_{k+1} = Ax_k + Bu_k && \text{(7.1b)}\\
& x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} && \text{(7.1c)}\\
& x_N \in \mathcal{X}_N && \text{(7.1d)}\\
& x_0 = x \in \mathcal{X} && \text{(7.1e)}
\end{aligned}
$$

where $x \in \mathbb{R}^n$ is the measured state, x_k is the predicted state under the control action $u_k \in \mathbb{R}^m$ over the horizon N, and $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{X}_N$ are polytopic sets.

Problem (7.1) has a state-feedback piecewise affine on polytopes control-law

$$
u_0^\star(x) = \kappa(x) \begin{cases} F_1 x + G_1 & \text{if } x \in \mathcal{R}_1 \\ \quad \vdots & \\ F_p x + G_p & \text{if } x \in \mathcal{R}_p \end{cases} \tag{7.2}
$$

where $F_i \in \mathbb{R}^{m\times n}$ and $G_i \in \mathbb{R}^m$ are the optimal feedback and feedforward gains. The regions $\mathcal{R}_i$ are polytopes.

7.1 Orbit Controller

In this section we define the orbit controller $\kappa^o(x)$ which uses symmetry to reduce the complexity of storing the optimal control-law $u_0^\star(x) = \kappa(x)$. The results of this section hold for any subgroup $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$ of controller symmetries.

Consider an explicit model predictive controller $u_0^\star = \kappa(x)$ of the form (7.2). Symmetry tells us that pieces $i, j \in \mathcal{I}$ of the controller are related by state-space Θ and input-space Ω transformations. Thus we need only store one of these controller pieces and the other piece can be recovered using the symmetry transformations. This intuition is formalized and generalized using the concept of a controller orbit.

Two controller pieces $i, j \in \mathcal{I}$ are symmetrically equivalent if there exists a state-space and input-space transformation pair $(\Theta, \Omega) \in \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$ that maps controller piece i to piece $j = \pi(i)$ for $\pi \in \mathbf{\Pi}(\mathcal{G})$. The set of all controller pieces equivalent to the i-th piece is called a controller orbit

$$\mathcal{G}(i) = \big\{\pi(i) : \pi \in \mathbf{\Pi}(\mathcal{G})\big\} \tag{7.3}$$

where $\mathbf{\Pi}$ maps linear symmetries $(\Theta, \Omega) \in \mathrm{Aut}(\mathcal{I})$ to the corresponding permutation $\pi = \mathbf{\Pi}(\Theta, \Omega)$ of the controller pieces $\mathcal{I}$. The set of controller orbits is denoted by $\mathcal{I}/\mathcal{G} = \{\mathcal{G}(i_1), \dots, \mathcal{G}(i_r)\}$, read as $\mathcal{I}$ modulo $\mathcal{G}$, where $\{i_1, \dots, i_r\}$ is a set that contains one representative controller piece i_j from each orbit $\mathcal{G}(i_j)$. With abuse of notation we will equate the set of controller orbits $\mathcal{I}/\mathcal{G}$ with sets of representative controller pieces $\mathcal{I}/\mathcal{G} = \{i_1, \dots, i_r\}$. The controller orbits $\mathcal{G}(i)$ for $i \in \mathcal{I}/\mathcal{G}$ partition the controller pieces $\mathcal{I}$ into disjoint equivalence classes $\mathcal{I} = \bigsqcup_{i \in \mathcal{I}/\mathcal{G}} \mathcal{G}(i)$.

Using orbits the memory requirements for the explicit model predictive controller $\kappa(x)$ can be reduced by storing only one representative $i \in \mathcal{G}(i)$ from each controller orbit $\mathcal{G}(i)$. The other controller pieces $j \in \mathcal{G}(i)$ can be recovered using the state-space and input-space transformations $(\Theta, \Omega) \in \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$ and the symmetry relation

$$u_0^\star(x) = F_j x + G_j = \Omega F_i \Theta^{-1} x + \Omega G_i. \tag{7.4}$$

In terms of the controller orbits, the optimal control-law $u_0^\star(x) = \kappa(x)$ can be written as

$$u_0^\star(x) = \begin{cases} \Omega F_{i_1} \Theta^{-1} x + \Omega G_{i_r} & \text{if } x \in \Theta \mathcal{R}_{i_1} \text{ for some } (\Theta, \Omega) \in \mathcal{G} \\ \vdots & \\ \Omega F_{i_r} \Theta^{-1} x + \Omega G_{i_r} & \text{if } x \in \Theta \mathcal{R}_{i_r} \text{ for some } (\Theta, \Omega) \in \mathcal{G} \end{cases} \tag{7.5}$$

where $\{i_1, \dots, i_r\} = \mathcal{I}/\mathcal{G}$ is a set of representative controller pieces. We call this representation of the optimal control-law $u_0^\star(x) = \kappa(x)$ the orbit controller $\kappa^o(x)$.

The orbit controller $\kappa^o(x)$ neglects part of the feasible state-space $\mathcal{X}_0$ where the optimal control-law $u_0^\star(x)$ is symmetrically redundant. The orbit controller $\kappa^o(x)$ only explicitly stores the control-law $u_0^\star(x)$ on a subset $\hat{\mathcal{R}}$ of the feasible region $\mathcal{X}_0$

$$\hat{\mathcal{R}} = \bigcup\nolimits_{i \in \mathcal{I}/\mathcal{G}} \mathcal{R}_i \subset \mathcal{X}_0. \tag{7.6}$$

We can visualize the orbit controller in terms of the orbit graph $\Gamma(\mathcal{S}) = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}$ of the orbit graph $\Gamma(\mathcal{S})$ is the index set $\mathcal{V} = \mathcal{I}$ of the controller pieces $\{(F_i, G_i, \mathcal{R}_i)\}_{i\in\mathcal{I}}$. By definition the symmetries $(\Theta, \Omega) \in \text{Aut}(\mathcal{I})$ induce sets of edges

$$\mathcal{E}_{(\Theta,\Omega)} = \Big\{(i, \pi(i)) \in \mathcal{I} \times \mathcal{I} : \pi = \mathbf{\Pi}(\Theta, \Omega), i \neq \pi(i)\Big\} \tag{7.7}$$

that related controller pieces $\{(F_i, G_i, \mathcal{R}_i)\}_{i\in\mathcal{I}}$. For a subset of symmetries $\mathcal{S} \subseteq \text{Aut}(\mathcal{I})$ we can define the graph $\Gamma(\mathcal{S}) = (\mathcal{V}, \mathcal{E}_\mathcal{S})$ whose node set is the set of controller pieces $\mathcal{V} = \mathcal{I}$ and whose edge set $\mathcal{E}_\mathcal{S} = \bigcup_{(\Theta,\Omega)\in\mathcal{S}} \mathcal{E}_{(\Theta,\Omega)}$ is the union of edge sets induced by the symmetries $(\Theta, \Omega) \in \mathcal{S}$.

Each connected component of the graph $\Gamma(\mathcal{S})$ corresponds to a controller orbit $\mathcal{G}(i) = \{\pi(i) \in \mathcal{I} : \pi \in \mathbf{\Pi}(\mathcal{G})\}$ where $\mathcal{G} = \langle \mathcal{S} \rangle$ is the symmetry group generated by $\mathcal{S}$. The orbit graph $\Gamma(\mathcal{S})$ has $|\mathcal{I}/\mathcal{G}|$ connected components. An orbit controller $\kappa^o(x)$ can be created by selecting one representative controller piece $(F_i, G_i, \mathcal{R}_i)$ from each connected component of the orbit graph $\Gamma(\mathcal{S})$. The symmetries $(\Theta, \Omega) \in \mathcal{S}$ allow us to move along the edges of the orbit graph $\Gamma(\mathcal{S})$ between controller pieces $(F_i, G_i, \mathcal{R}_i)$ for $i \in \mathcal{I}$ to reconstruct the full explicit controller $\kappa(x)$. Obviously the orbit controller $\kappa^o(x)$ is not unique since the choice of controller pieces from each component is arbitrary.

The following example demonstrates the concept of an orbit controller.

Example 26. *Consider the model predictive control problem*

$$\begin{aligned} J^\star(x) = \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & x_N^T x_N + \sum\nolimits_{k=0}^{N-1} x_k^T x_k + u_k^T u_k && (7.8a)\\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k && (7.8b)\\ & x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} && (7.8c)\\ & x_0 = x && (7.8d) \end{aligned}$$

where $N = 5$, the dynamics matrices are

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{7.9}$$

and $\mathcal{X}$ and $\mathcal{U}$ are box constraints on the state and input

$$\mathcal{X} = \{x \in \mathbb{R}^2 : -1 \leq x_i \leq 1, i = 1, 2\} \tag{7.10}$$
$$\mathcal{U} = \{u \in \mathbb{R}^2 : -1 \leq u_i \leq 1, i = 1, 2\}. \tag{7.11}$$

This model predictive control problem has a piecewise affine on polytope control-law $u_0^\star(x) = \kappa(x)$. The partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of this controller is shown in Figure 7.1a and the vector field $u = \kappa(x)$ of the controller $\kappa(x)$ is shown in Figure 7.1b.

The symmetry group Aut$(\mathcal{I})$ *of the piecewise affine on polytope controller $\kappa(x)$ is the dihedral-4 group $\mathcal{D}_4$ which consists of the planar rotations by* 90 *degree increments and the* *reflections of the plane about the horizontal, vertical, and both diagonal axis. This group* $\mathrm{Aut}(\mathcal{I}) = \mathcal{D}_4$ *can be generated by the* 90 *degree planar rotation and the reflection about the vertical axis in the state-space and the* 90 *degree planar rotation and the diagonal reflection in the input-space*

$$(\Theta_r, \Omega_r) = \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \tag{7.12a}$$

$$(\Theta_s, \Omega_s) = \left(\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right) \tag{7.12b}$$

where (Θ_r, Ω_r) *are the* 90 *degree rotation matrices and* (Θ_s, Ω_s) *are the reflections about the vertical and diagonal axis respectively.*

The symmetry $\mathrm{Aut}(\kappa)$ *of the explicit controller* $\kappa(x)$ *can be seen in the controller partition* $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ *shown in Figure 7.1a. Rotating the partition* $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ *by increments of* 90 *degrees does not change the pattern of the partition. Likewise reflecting the partition* $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ *about the horizontal, vertical, or either diagonal axis does not change the partition.*

The symmetry can also be seen in the vector field of the optimal control-law $u_0^\star(x)$ *shown in Figure 7.1b. Rotating the state-space by* 90 *degrees is equivalent to rotating the vector field* $u_0^\star(x)$ *by* 90 *degrees* $\Omega_r u_0^\star(x) = u_0^\star(\Theta_r x)$*. The points* $x, y \in \mathcal{X}_0$ *shown in Figure 7.1b are related* $y = \Theta_s x$ *by the vertical reflection* Θ_s*. The vector field at these points is related by the diagonal reflection* Ω_s

$$u_0^\star(y) = u_0^\star(\Theta_s x) = \Omega_s u_0^\star(x). \tag{7.13}$$

Symmetry organizes the $|\mathcal{I}| = 33$ *pieces of the explicit controller* $\kappa(x)$ *into* $|\mathcal{I}/\mathcal{G}| = 8$ *orbits: one orbit of size* $|\mathcal{G}(i)| = 1$*, six orbits of size* $|\mathcal{G}(i)| = 4$ *and one orbit of size* $|\mathcal{G}(i)| = 8$*. Each of these orbits correspond to connected component of the orbit graph* $\Gamma(\mathcal{S})$ *shown in Figure 7.2.*

Figure 7.2 shows the connected components of the orbit graph $\Gamma(\mathcal{S}) = (\mathcal{V}, \mathcal{E}_\mathcal{S})$ *for the explicit controller* $\kappa(x)$ *and the generating set* $\mathcal{S} = \{(\Theta_r, \Omega_r), (\Theta_s, \Omega_s)\}$*. The nodes* $i \in \mathcal{I} = \mathcal{V}$ *of these graph components are controller pieces* $(F_i, G_i, \mathcal{R}_i)$ *with the node drawn inside the region* $\mathcal{R}_i$*. The edges are induced by the* 90 *degree rotation matrix* Θ_r *and vertical reflection* Θ_s*.*

The orbit shown in Figure 7.2a is a singleton orbit $|\mathcal{G}(i)| = 1$*. The orbit shown in 7.2e contains* $|\mathcal{G}(i)| = 8$ *controller pieces* $(F_i, G_i, \mathcal{R}_i)$*. The regions* $\mathcal{R}_i$ *for* $i \in \mathcal{G}(i)$ *are related by the reflections and rotations of the symmetry group* $\mathrm{Aut}(\mathcal{I}) = \langle \mathcal{S} \rangle$*. The feedback* F_i *and feedforward* G_i *gains in these regions are related by symmetry* $\Omega F_i = F_j \Theta$ *and* $\Omega G_i = G_i$*. This can be seen in the vector field of the optimal control-law* $u_0^\star = \kappa(x)$ *shown in Figure 7.1b.*

An orbit controller $\kappa^o(x)$ *can be created by selecting one representative controller piece from each orbit. The partition* $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ *of an orbit controller* $\kappa^o(x)$ *is shown in Figure 7.1c. The colored regions represent the pieces of the controller explicitly stored by* $\kappa^o(x)$*. The control-law in the gray regions is recovered using symmetry. This controller contains one*

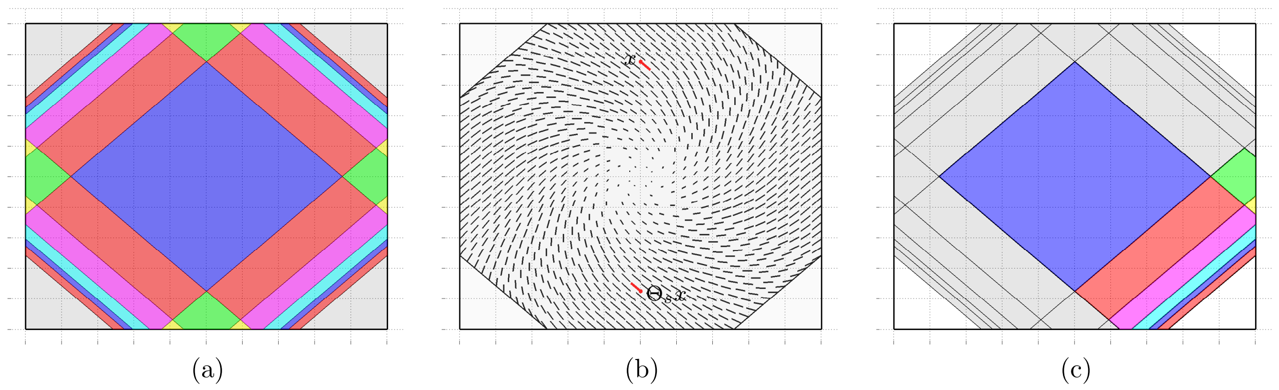


Figure 7.1: Example of orbit controller. (a) Partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the explicit model predictive controller $\kappa(x)$. (b) Vector field $u = \kappa(x)$ of the explicit model predictive controller $\kappa(x)$. (c) Partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ of an orbit controller $\kappa^o(x)$. The control-law is explicitly stored in the colored regions and reconstructed from symmetry in the gray regions.

node from each connected component of the orbit graph $\Gamma(\mathcal{S})$*. The choice of the pieces to store is arbitrary.*

Synthesis of the Orbit Controller

In this section we present a simple procedure for synthesizing the orbit controller $\kappa^o(x)$.

The orbit controller can be synthesized using the steps in Procedure 4. The first step of this procedure is to generate the full explicit controller $\kappa(x)$. This can be done using the Multi-Parametric Toolbox (MPT) [57]. Next a symmetry group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$ of the controller $\kappa(x)$ is identified. This can be done using the techniques presented in Chapter 5 to find $\mathcal{G} = \mathrm{Aut}(MPC) \subseteq \mathrm{Aut}(\mathcal{I})$. Next the orbits $\mathcal{G}(i)$ of the controller pieces $\mathcal{I}$ are calculated using Algorithm 12. Finally the orbit controller $\kappa^o(x)$ is constructed by selecting one controller piece $(F_i, G_i, \mathcal{R}_i)$ from each orbit $\mathcal{G}(i)$.

Procedure 4 Synthesis of Orbit Controller $\kappa^o(x)$

```
1: Generate the full explicit controller $\kappa(x)$.
2: Identify a symmetry group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$ for explicit controller $\kappa(x)$
3: Calculate the orbits $\mathcal{G}(i)$ of the controller pieces $\mathcal{I}$
4: Select a representative controller piece $(F_i, G_i, \mathcal{R}_i)$ from each orbit $\mathcal{G}(i)$.
```

This procedure can be interpreted in terms of the orbit graph $\Gamma(\mathcal{S})$. In the first step we construct the node set $\mathcal{I}$ of the orbit graph $\Gamma(\mathcal{S})$ by constructing the explicit controller $\kappa(x)$. In the second step we find the edge set $\mathcal{E}_{\mathcal{S}}$ of the orbit graph $\Gamma(\mathcal{S})$. In the third step we use the orbit algorithm to find the connected components of the orbit graph $\Gamma(\mathcal{S})$. Finally we

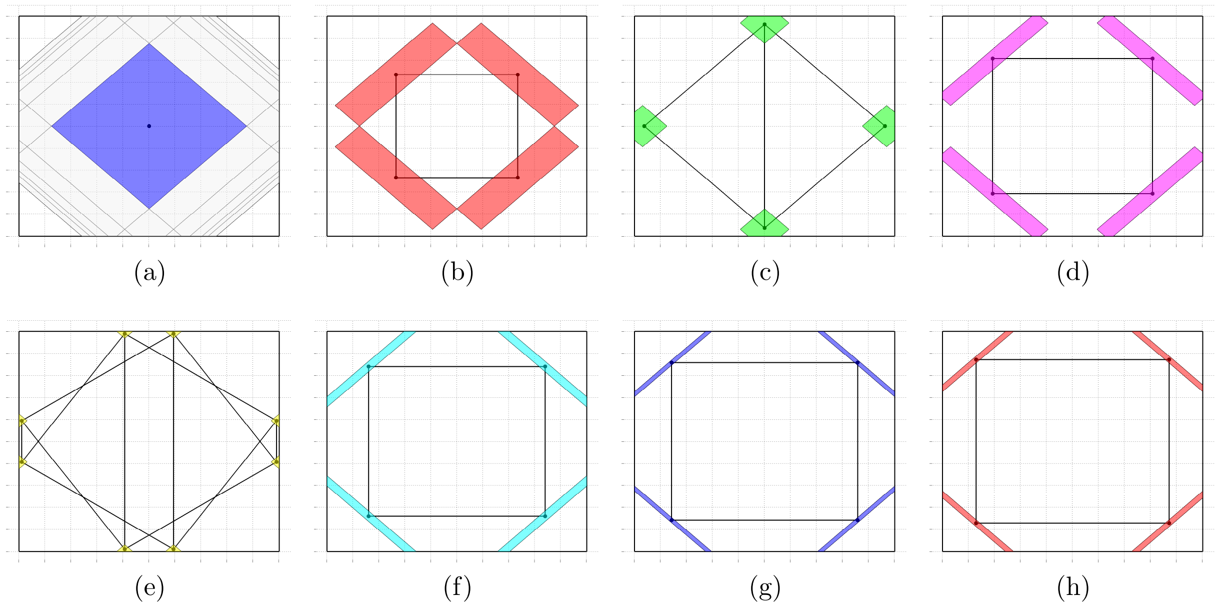


Figure 7.2: Connected components of the orbit graph $\Gamma(\mathcal{S})$ for the explicit controller $\kappa(x)$. The dots represent the graph nodes $\mathcal{V} = \mathcal{I}$ and the lines represent the graph edges $\mathcal{E}_{\mathcal{S}}$. The graph nodes $i \in \mathcal{I}$ are drawn inside the region $\mathcal{R}_i$ of the controller piece $(F_i, G_i, \mathcal{R}_i)$ they represent.

chose one representative from each connected component of the orbit graph $\Gamma(\mathcal{S})$ to produce an orbit controller $\kappa^o(x)$.

The orbits $\mathcal{G}(i)$ of the controller pieces $i \in \mathcal{I}$ can be identified using Algorithm 12. Algorithm 12 loops until every controller piece $i \in \mathcal{I}$ has been assigned to an orbit $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$. During each loop, the algorithm begins by selecting an unassigned controller piece $i \in \mathcal{J}$. The algorithm finds a point $z \in \text{int}(\mathcal{R}_i)$ in the interior of the region $\mathcal{R}_i$. This can be accomplished by calculating the Chebyshev ball $\mathcal{B}(c, r) \subset \mathcal{R}_i$ and letting $z = c$ be its center. The orbit $\mathcal{G}z$ of this point z will be used to construct the orbit $\mathcal{G}(i)$ of the i-th controller piece. From the definition of the symmetry group $\mathcal{G} \subseteq \text{Aut}(\mathcal{I})$, we know that if a region $\mathcal{R}_j \in \{\mathcal{R}_i\}_{i\in\mathcal{I}}$ contains the point Θz then $\mathcal{R}_j = \Theta\mathcal{R}_i$ since the interiors of the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ are mutually disjoint $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset$. Thus the orbit $\mathcal{G}(i)$ contains the indices of each region $\mathcal{R}_j$ that contain a point Θz from the orbit $\mathcal{G}z$. If $\Theta z \in \mathcal{R}_j$ then we add $j \in \mathcal{I}$ to the orbit $\mathcal{G}(i)$ and remove j from the set $\mathcal{J}$ of unassigned controller pieces. The algorithm terminates when all the controller pieces $\mathcal{I}$ have been assigned to an orbit $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$.

The following example demonstrates the use of Procedure 4 and Algorithm 12 to synthesize the orbit controller.

Example 27. *In this example we demonstrate the synthesis of an orbit controller for the*

Algorithm 12 Calculation of orbits $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$ of the controller pieces $\mathcal{I}$

```
Input: controller partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ and symmetry group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$.
Output: controller orbits $\mathcal{I}/\mathcal{G} = \{\mathcal{G}(i)_1, \dots, \mathcal{G}(i)_r\}$
 1: set all controller pieces to unassigned $\mathcal{J} = \mathcal{I}$
 2: while unassigned pieces $\mathcal{J} \neq \varnothing$ do
 3:     select an unassigned controller piece $i \in \mathcal{J}$
 4:     find point $z \in \mathrm{int}(\mathcal{R}_i)$ in the interior of $\mathcal{R}_i$
 5:     calculate orbit $\mathcal{G}z$ of point $z$
 6:     for each $y = \Theta z \in \mathcal{G}z$ do
 7:         find $\mathcal{R}_j \in \{\mathcal{R}_i\}_{i\in\mathcal{I}}$ containing $y \in \mathcal{R}_j$
 8:         add $j$ to orbit $\mathcal{G}(i)$ and remove $j$ from unassigned set $\mathcal{J}$
 9:     end for
10: end while
```

model predictive control problem described in Example 26.

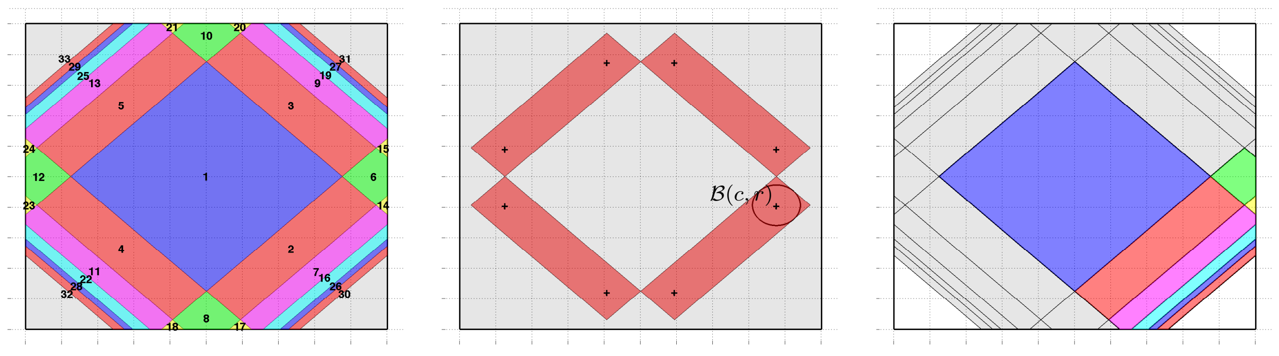
The partition of the explicit model predictive controller $\kappa(x)$ is shown in Figure 7.3a. The symmetry group $\mathcal{G} = \mathrm{Aut}(\mathcal{I})$ is the dihedral-4 group identified previously. Next we need to find the orbits of the controller pieces using Algorithm 12.

The algorithm begins by setting all of the controller pieces $\mathcal{I} = \mathcal{J} = \{1, 2, \dots, 33\}$ as unassigned to orbits. The algorithm selects an unassigned controller pieces $i = 1 \in \mathcal{J}$ and finds a point $z = 0 \in \mathrm{int}(\mathcal{R}_1)$ in the interior of region $\mathcal{R}_1$. The orbit of this point $z = 0$ is the singleton set $\mathcal{G}z = \{0\}$. Thus the orbit of controller piece $(F_1, G_1, \mathcal{R}_1)$ is the singleton orbit $\mathcal{G}(1) = \{1\}$. Controller piece $i = 1$ is removed from the list $\mathcal{J}$ of unassigned controller pieces.

On the next loop, the algorithm selects a new unassigned controller piece $i = 2 \in \mathcal{J}$. For the region $\mathcal{R}_2$ the Chebyshev ball is not unique. Suppose the algorithm chose the Chebyshev ball shown in Figure 7.3b. Then the orbit of its center z is the set of eight points shown in Figure 7.3b. Algorithm 12 will find that these eight points are contained in the four sets $\mathcal{R}_2$, $\mathcal{R}_3$, $\mathcal{R}_4$, and $\mathcal{R}_5$. Thus the orbit of controller piece $(F_2, G_2, \mathcal{R}_2)$ is $\mathcal{G}(2) = \{2, 3, 4, 5\}$. These pieces $\{2, 3, 4, 5\}$ are removed from the list of unassigned controller pieces $\mathcal{J} = \{6, 7, \dots, 33\}$.

Algorithm 12 continues until each controller piece $\mathcal{I} = \{1, 2, \dots, 33\}$ has been assigned to an orbit. At termination, the algorithm has organized the controller pieces into $|\mathcal{I}/\mathcal{G}| = 8$ orbits. The final step of Procedure 4 is to select one representative piece from each orbit $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$. The partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ of the resulting controller is shown in Figure 7.3c.

Remark 2. *Instead of using the Chebyshev ball to find a point $z \in \mathrm{int}(\mathcal{R}_i)$ in the interior of region $\mathcal{R}_i$ we could use the analytic center of $\mathcal{R}_i$. In this case there is a one-to-one correspondence between the orbit $\mathcal{G}z$ of the analytic center z and the orbit $\mathcal{G}(i)$ of the controller piece. This reduces the complexity of computing the orbits $\mathcal{G}z$. However calculating the analytic center is much more computationally expensive than computing a Chebyshev ball. In*



(a) (b) (c)

Figure 7.3: Example of the synthesis of an orbit controller. (a) The partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the controller pieces $\mathcal{I} = \{1, 2, \ldots, 33\}$. (b) A Chebyshev ball for region $\mathcal{R}_2$ and the orbit $\mathcal{G}z$ of its center $z \in \text{int}(\mathcal{R}_2)$ under the dihedral-4 group. (c) The partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ of the orbit controller which includes one piece i from each orbit $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$ of controller pieces.

practice we have found that using a Chebyshev ball in Algorithm 12 is more computationally efficient than using the analytic center.

Implementation of the Orbit Controller

In this section we discuss the implementation of the orbit controller $\kappa^o(x)$.

Conceptually the orbit controller $\kappa^o(x)$ is implemented using Algorithm 13. The implementation of the orbit controller $\kappa^o(x)$ is similar to the standard implementation of the explicit controller $\kappa(x)$ in Algorithm 2. The point location problem still requires searching the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the feasible region $\mathcal{X}_0$. However this step is divided into two nested loops. In the outer-loop the algorithm searches the representative regions $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$. In the inner-loop the algorithm searches the orbit $\{\mathcal{R}_j\}_{j\in\mathcal{G}(i)} = \{\Theta\mathcal{R}_i\}_{\Theta\in\mathcal{G}}$ of the representative region $\mathcal{R}_i$ for the set $\mathcal{R}_j = \Theta\mathcal{R}_i$ which contains the measured state $x \in \Theta\mathcal{R}_i$.

Once the region $\mathcal{R}_j = \Theta\mathcal{R}_i$ containing the measured state $x \in \Theta\mathcal{R}_i$ has been found the appropriate control action $u_0^\star = \kappa(x)$ is calculated using the symmetry relation

$$u_0^\star(x) = \Omega F_i \Theta^{-1} x + \Omega G_i \tag{7.14}$$

where $\Theta^{-1}x \in \mathcal{R}_i$.

This search can be interpreted in terms of the orbit graph $\Gamma(\mathcal{S})$. In the outer-loop we search among the connected components of the orbit graph $\Gamma(\mathcal{S})$. The inner-loop searches the components.

The computational complexity of Algorithm 13 depends on how it searches the orbits $\{\Theta\mathcal{R}_i\}_{\Theta\in\mathcal{G}} = \{\mathcal{R}_j\}_{j\in\mathcal{G}(i)}$. The orbits $\{\Theta\mathcal{R}_i\}_{\Theta\in\mathcal{G}}$ can be efficiently searched by creating a search-tree for each connected component of the orbit graph $\Gamma(\mathcal{S})$. The search-trees are stored using a directed acyclic graph data-structure. The nodes $j \in \mathcal{G}(i) \subseteq \mathcal{I}$ of the search-tree correspond to regions $\mathcal{R}_j = \Theta\mathcal{R}_i$ in the orbit $\{\Theta\mathcal{R}_i\}_{\Theta\mathcal{G}} = \{\mathcal{R}_j\}_{j\in\mathcal{G}(i)}$ of the region $\mathcal{R}_i$.

Algorithm 13 Conceptual Implementation of Orbit Controller $\kappa^o(x)$

```
Input: Measured State $x$
Output: Optimal Control $u_0^\star = \kappa(x)$
1: for each $\mathcal{R}_i \in \{\mathcal{R}_j\}_{j \in \mathcal{I}/\mathcal{G}}$ do
2:     for each $\mathcal{R}_j \in \{\Theta \mathcal{R}_i\}_{\Theta \in \mathcal{G}}$ do
3:         if $x \in \mathcal{R}_j = \Theta \mathcal{R}_i$ then
4:             $u_0^\star = \Omega F_i \Theta^{-1} x + \Omega G_i$ return
5:         end if
6:     end for
7: end for
```

To each node $j \in \mathcal{G}(i)$ we assign a list $\texttt{child}(j)$ of children nodes $k \in \texttt{child}(j)$ that succeed node j in the search-tree. If $j \in \mathcal{G}(i)$ is a leaf-node then its child list is empty $\texttt{child}(j) = \emptyset$. We also assign a generator $(\Theta_j, \Omega_j) \in \mathcal{S}$ to each node $j \in \mathcal{G}(i)$ which corresponds to the edge entering node $j \in \mathcal{G}(i)$ in the search-tree. If $j = i \in \mathcal{G}(i)$ is the root-node then we chose the identity element $(\Theta_j, \Omega_j) = (I_n, I_m)$.

Algorithm 14 is a practical implementation of Algorithm 13 that uses search-trees to efficiently search the orbits $\{\Theta \mathcal{R}_i\}_{\Theta \in \mathcal{G}}$ of the representative controller pieces $i \in \mathcal{I}/\mathcal{G}$. Algorithm 14 calls the function `SearchPiece` which recursively searches the orbit $\mathcal{G}(i)$ using a search-tree. As the function `SearchPiece` proceeds deeper into the search-tree it transforms the state $z = \Theta_{j_d}^{-1} \cdots \Theta_{j_1}^{-1} x = \Theta^{-1} x$ and tests the condition $z = \Theta^{-1} x \in \mathcal{R}_i$. This is equivalent to testing $\Theta z = x \in \Theta \mathcal{R}_i = \mathcal{R}_{j_d}$ where $\Theta = \Theta_{j_1} \cdots \Theta_{j_d}$. This implementation requires only matrix-vector multiplication $\Theta^{-1} x$ rather than matrix-set multiplication $\Theta \mathcal{R}_i$ where the inverses Θ^{-1} of the generators $\Theta \in \mathcal{S}$ are stored explicitly.

If the region $\mathcal{R}_i$ contains the transformed state $z = \Theta_{j_d}^{-1} \cdots \Theta_{j_1}^{-1} x \in \mathcal{R}_i$ then the control is calculated by

$$u = \Omega_{j_d} F_i z + \Omega_{j_d} G_i. \tag{7.15}$$

The algorithm then backs out of the search-tree applying the input-space transformations $\Omega_{j_{d-1}}, \ldots, \Omega_{j_1}$ to the control input u. The orbit search returns the control input

$$u = \Omega_{j_1} \cdots \Omega_{j_d} F_i \Theta_{j_d}^{-1} \cdots \Theta_{j_1}^{-1} x + \Omega_{j_1} \cdots \Omega_{j_d} G_i. \tag{7.16}$$

If the region $\mathcal{R}_i$ does not contain the transformed state $z = \Theta_{j_d}^{-1} \cdots \Theta_{j_1}^{-1} x \notin \mathcal{R}_i$ then the algorithm searches the child nodes $k \in \texttt{child}(j)$ of node $j \in \mathcal{G}(i)$ to check whether they contain the state $x \in \mathcal{R}_k$. If $j \in \mathcal{G}(i)$ is a leaf-node then the algorithm returns to the parent node.

The computational complexity of Algorithm 14 is the same as the standard implementation described in Algorithm 2.

Proposition 18. *Let the number of states n and inputs m satisfy $m \leq n$. Then the computational complexity of Algorithm 13 using the orbit search Algorithm 14 is $O\big(\sum_{i \in \mathcal{I}} n c_i\big)$.*

Algorithm 14 Practical Implementation of Orbit Controller $\kappa^o(x)$

```
Input: measured state $x$ and search-tree for orbit graph $\Gamma(\mathcal{S})$
Output: optimal control $u = \kappa(x)$
  for each representative controller piece $i \in \mathcal{I}/\mathcal{G}$ do
      SearchPiece($i$,$x$,$F_i$,$G_i$,$\mathcal{R}_i$)
      if found then
          return $u$
      end if
  end for
  function SearchPiece($j$,$z$,$F$,$G$,$\mathcal{R}$)
      if $\Theta_j^{-1} z \in \mathcal{R}$ then
          calculate control:
              $$u = \Omega_j F \Theta_j^{-1} z + \Omega_j G$$
          set found = true
          return $u$ and found
      else
          for each $k \in$ child($j$) do
              SearchPiece($k$,$\Theta_j^{-1} z$,$F$,$G$,$\mathcal{R}$)
              if found then
                  transform input:
                      $$u = \Omega_j u$$
                  return $u$ and found
              end if
          end for
          found = false
          return found
      end if
  end function
```

Proof. The most computationally expensive steps in Algorithm 14 are the transformation of the state $\Theta_j^{-1}z$ and input $\Omega_j u$, and the point-in-set test $z \in \mathcal{R}_i$ which have computational complexity $O(n^2)$, $O(m^2)$, and $O(nc_i)$ respectively where c_i is the number of constraints that define $\mathcal{R}_i$. Since the region $\mathcal{R}_i$ is bounded we have $c_i \geq n+1$. Thus the cost of visiting each node $j \in \mathcal{G}(i)$ is $O(nc_i)$.

In the worst-case, Algorithm 13 will visit every orbit $\mathcal{G}(i) \in \mathcal{I}/\mathcal{G}$ and every node in the orbit $j \in \mathcal{G}(i)$ once. Thus the complexity is

$$\sum_{\mathcal{G}(i)\in\mathcal{I}/\mathcal{G}} \sum_{j\in\mathcal{G}(i)} O(nc_i) = \sum_{i\in\mathcal{I}} O(nc_i). \tag{7.17}$$

□

The following example demonstrates the implementation of the orbit controller.

Example 28. *In this example we demonstrate the implementation of the orbit controller from Example 27.*

Consider the measured state $x(t)$ shown in Figure 7.4a. Since the orbit controller $\kappa^o(x)$ does not explicit store the optimal control-law $u_0^\star(x)$ at this state it must be reconstructed using symmetry. This can be accomplished using Algorithm 13.

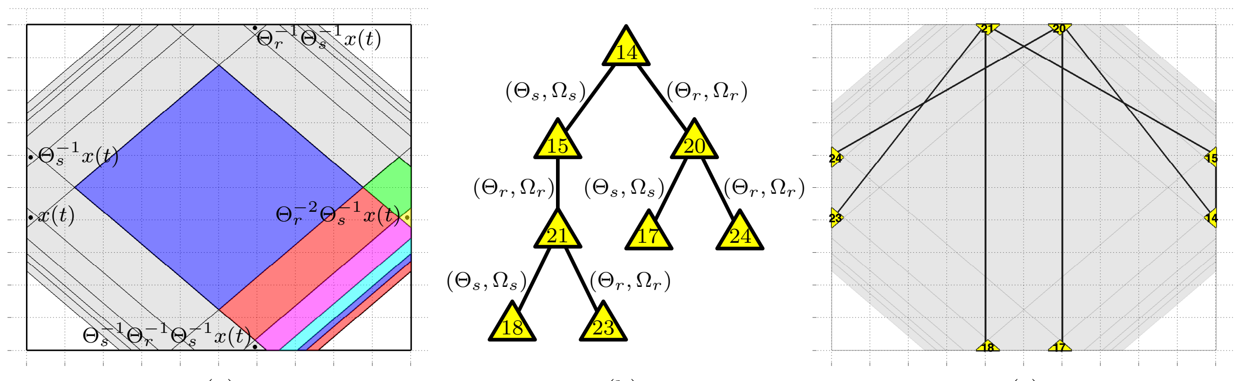
The outer-loop of Algorithm 13 searches $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ for the orbit $\{\mathcal{R}_j\}_{i\in\mathcal{G}(i)} = \{\Theta\mathcal{R}_i\}_{\Theta\in\mathcal{G}}$ of the region $\mathcal{R}_j = \Theta\mathcal{R}_i$ that contains the measured state $x \in \mathcal{R}_j$. This corresponds to searching over the connected components of the orbit graph $\Gamma(\mathcal{S})$ shown in Figure 7.2. After searching the connected components of the orbit graph $\Gamma(\mathcal{S})$ shown in Figures 7.2a-7.2d the Algorithm will search the orbit shown in Figure 7.2e which contains the region $\mathcal{R}_{23}$ that contains the measured state $x \in \mathcal{R}_{23}$.

This orbit $\{\Theta\mathcal{R}_{14}\}_{\Theta\in\mathcal{G}} = \big\{\mathcal{R}_{14}, \mathcal{R}_{15}, \mathcal{R}_{17}, \mathcal{R}_{18}, \mathcal{R}_{20}, \mathcal{R}_{21}, \mathcal{R}_{23}, \mathcal{R}_{24}\big\}$ can be searched using the search-tree shown in Figure 7.4b. Figure 7.4c shows how the nodes in this search-tree correspond to the regions $\mathcal{R}_j$ in the orbit $\{\Theta\mathcal{R}_{14}\}$ of region $\mathcal{R}_{14}$.

The orbit search begins at the root node $i = 14$. Since the region $\mathcal{R}_{14}$ does not contain the state $x \notin \mathcal{R}_{14}$ the algorithm proceeds to search the children $\texttt{child}(14) = \{15, 20\}$ of node $j = 14$. It begins with node $j = 15$ where region $\mathcal{R}_{15} = \Theta_s\mathcal{R}_{14}$ is the image of region $\mathcal{R}_{14}$ under the vertical reflection Θ_s. This node does not contain the state $\Theta_s^{-1}x \notin \mathcal{R}_{14}$. Thus the algorithm proceeds to the single child node $21 \in \texttt{child}(15)$ of node $j = 15$. This node does not contain state $\Theta_r^{-1}\Theta_s^{-1}x \notin \mathcal{R}_{14} = \Theta_r^{-1}\Theta_s^{-1}\mathcal{R}_{21}$. Next the algorithm tests the leaf-node $j = 18 \in \texttt{child}(21)$. Since this node does not contain the state $\Theta_s^{-1}\Theta_r^{-1}\Theta_s^{-1}x \notin \mathcal{R}_{14}$ the algorithm backtracks to node $j = 21$ and tests its next child node $23 \in \texttt{child}(21)$. This node does contain the state $\Theta_r^{-1}\Theta_r^{-1}\Theta_s^{-1}x \in \mathcal{R}_{14}$ (i.e. $x \in \Theta_s\Theta_r^2\mathcal{R}_{14} = \mathcal{R}_{23}$). Therefore the algorithm calculates the control

$$u = \Omega_r F_{14}\Theta_r^{-1}\Theta_r^{-1}\Theta_s^{-1}x + \Omega_r G_{14}. \tag{7.18}$$

and returns to node $j = 21$. The algorithm proceeds to backtrack to the root-node $i = 14$. As it returns to node 15 from node 21 and to node 14 from node 15 it applies the input-space



(a) (b) (c)

Figure 7.4: Example of the implementation of an orbit controller. (a) Partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}/\mathcal{G}}$ of the orbit controller $\kappa^0(x)$ with measured state $x(t)$. (b) Search-tree of controller piece $i = 14$. Θ_r is the counter-clockwise 90 degree rotation matrix and Θ_s is the vertical reflection matrix. (c) Search-tree of controller piece $i = 14$ drawn on the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the controller $\kappa(x)$.

transformations Ω_r and Ω_s respectively. The final control is

$$u = \Omega_s\Omega_r\Omega_r F_{14}\Theta_r^{-1}\Theta_r^{-1}\Theta_s^{-1}x + \Omega_s\Omega_r\Omega_r G_{14} \tag{7.19}$$
$$= F_{15}x + G_{15}.$$

Memory Reduction

In this section we discuss the memory advantages of using the orbit controller $\kappa^o(x)$. The group structure $\mathrm{Aut}(\kappa)$ is instrumental in reducing the memory needed to store $\kappa^o(x)$.

The orbit controller $\kappa^o(x)$ saves memory by replacing controller pieces $\{(F_i, G_i, \mathcal{R}_i)\}_{i\in\mathcal{I}}$ with symmetries $(\Theta, \Omega) \in \mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})$. We observe that symmetries $(\Theta, \Omega) \in \mathcal{G}$ are less expensive to store than controller pieces $(F_i, G_i, \mathcal{R}_i)$. Each symmetry $(\Theta, \Omega) \in \mathcal{G}$ requires $O(n^2)$ memory assuming the number of states n is greater than the number of inputs $m \leq n$. The i-th controller piece $(F_i, G_i, \mathcal{R}_i)$ requires $O(nc_i)$ memory where c_i is the number of half-spaces defining region $\mathcal{R}_i$. Since the regions $\mathcal{R}_i$ are bounded $c_i \geq n + 1$, we have a reduction in memory $O(n^2) \leq O(nc_i)$. However for our analysis we will conservatively assume $O(nc_i) = O(n^2)$.

In practice the main memory savings comes from the fact that a small number generators $\mathcal{S}$ can produces a large number of symmetries $\mathcal{G} = \langle\mathcal{S}\rangle$ that related many controller pieces. The orbit controller requires

$$O(n^2)\Big(|\mathcal{S}| + |\mathcal{I}/\mathcal{G}|\Big) \tag{7.20}$$

memory: $O(n^2)|\mathcal{I}/\mathcal{G}|$ memory to store the representative controller pieces $\{(F_i, G_i, \mathcal{R}_i)\}_{i\in\mathcal{I}/\mathcal{G}}$ and $O(n^2)|\mathcal{S}|$ memory to store the generators of the group $\mathcal{G} = \langle\mathcal{S}\rangle \subseteq \mathrm{Aut}(\mathcal{I})$. On the other hand the full explicit controller $\kappa(x)$ requires $O(n^2)|\mathcal{I}|$ memory to store each piece $\mathcal{I}$ of the controller individually.

For any group $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})/\mathrm{Ker}(\mathbf{\Pi})$ the memory requirements of storing the orbit controller $\kappa^o(x)$ are no more than storing the full explicit controller $\kappa(x)$ since $|\mathcal{S}| + |\mathcal{I}/\mathcal{G}| \leq |\mathcal{I}|$.

Proposition 19. *Let $\mathcal{G} \subseteq \mathrm{Aut}(\mathcal{I})/\mathrm{Ker}(\mathbf{\Pi})$. Then there exists a generating set $\mathcal{S}$ such that $|\mathcal{S}| + |\mathcal{I}/\mathcal{G}| \leq |\mathcal{I}|$.*

Proof. Consider the stabilizer chain

$$\mathcal{G} = \mathcal{G}_0 \supseteq \cdots \supseteq \mathcal{G}_{\mathcal{I}/\mathcal{G}} = \mathrm{Ker}(\mathbf{\Pi}(\mathcal{G})) = \langle\varnothing\rangle. \tag{7.21}$$

Let $\mathcal{S}_i = (\mathcal{G}_{i-1}/\mathcal{G}_i) \setminus \{e\}$ be a set of coset representatives $\mathcal{G}_{i-1}/\mathcal{G}_i$ minus the identity element $\{e\}$. By construction $\mathcal{S}_i \cup \mathcal{G}_i$ generates $\mathcal{G}_{i-1} = \langle\mathcal{S}_i \cup \mathcal{G}_i\rangle$. Thus $\mathcal{S} = \cup_{i=1}^{\mathcal{I}/\mathcal{G}} \mathcal{S}_i$ generates $\mathcal{G} = \langle\mathcal{S}\rangle$.

By the orbit-stabilizer theorem $|\mathcal{S}_i| = |\mathcal{G}_{i-1}i| - 1 \leq |\mathcal{G}i| - 1$ since $\mathcal{G}_{i-1} \subseteq \mathcal{G}$. Thus

$$|\mathcal{S}| \leq \sum_{i\in\mathcal{I}/\mathcal{G}} |\mathcal{G}i| - 1 = |\mathcal{I}| - |\mathcal{I}/\mathcal{G}|. \tag{7.22}$$

□

This proof says that in the worst-case we need to add one generator (Θ, Ω) to the generating set $\mathcal{S}$ for each controller piece $(F_i, G_i, \mathcal{R}_i)$ we eliminate. However typically the number of generators $\mathcal{S} \subseteq \mathcal{G}$ will be small compared to the number of redundant controller pieces $|\mathcal{S}| \ll |\mathcal{I}| - |\mathcal{I}/\mathcal{G}|$ eliminated. For instance if $\mathcal{G} = \mathcal{C}_N$ is a cyclic group then it can be arbitrarily large $|\mathcal{C}_N| = N \in \mathbb{N}$ while requiring only a single generator $\mathcal{S} = \{(\Theta, \Omega)\}$. If $\mathcal{G}$ is an abelian group then it requires at most k generators where $|\mathcal{G}| = p_1^{n_1} \cdots p_k^{n_k}$ is the unique prime-factor decomposition of its order [60]. If $\mathcal{G}$ is a non-abelian finite simple group (no non-trivial subgroup $\mathcal{H} \subset \mathcal{G}$ satisfies $g\mathcal{H} = \mathcal{H}g\ \forall g \in \mathcal{G}$) then it can be generated by at most 2 generators [65]. Unfortunately quantifying the exact memory savings provided by orbit controller $\kappa(x)$ is non-trivial.

The following example shows how the orbit controller $\kappa^o(x)$ reduces the memory requirements of storing the optimal control-law $u_0^\star = \kappa(x)$.

Example 29. *In this example we examine the memory reduction offered by the orbit controller $\kappa^o(x)$ defined in Example 26.*

The explicit controller $\kappa(x)$ requires 786 kilobytes to store the $|\mathcal{I}| = 33$ controller pieces $(F_i, G_i, \mathcal{R}_i)$. The orbit controller $\kappa^o(x)$ requires 207 kilobytes to store the $|\mathcal{I}/\mathcal{G}| = 8$ representative controller pieces $(F_i, G_i, \mathcal{R}_i)$ and 1.2 kilobytes to store the $|\mathcal{S}| = 2$ generators of the symmetry group $\mathrm{Aut}(\mathcal{I}) = \langle\mathcal{S}\rangle$. Thus the orbit controller $\kappa^o(x)$ achieves nearly a one-quarter reduction in memory.

Example: Quadrotor

In this section we apply the concept of an orbit controller to the quadrotor system described in Section 5.5 and shown in Figure 5.2. Recall that our model of the quadrotor has $m = 4$ inputs corresponding to the voltages applied to the four motors and $n = 13$ states: 6 cartesian position and velocity states, 6 angular position and velocity states, and 1 integrator state. The continuous-time nonlinear dynamics are given by (5.61). The dynamics were linearized and discretized using standard techniques. The states and inputs are subject to constraints that upper-bound and lower-bound their values given by (5.62).

Our model predictive controller uses a quadratic cost function

$$J(X,U) = x_N^T P x_N + \sum\nolimits_{k=0}^{N-1} x_k^T Q x_k + \rho u_k^T u_k \tag{7.23}$$

where $\rho = 0.1$. The diagonal matrix Q has the form

$$Q = \begin{bmatrix} q_1 I_3 & & & & \\ & q_2 I_3 & & & \\ & & q_3 I_3 & & \\ & & & q_4 I_3 & \\ & & & & q_5 \end{bmatrix} \tag{7.24}$$

where $q_1 = q_3 = 10$ is the penalty on the cartesian and angular positions, $q_2 = q_4 = 1$ is the penalty on the cartesian and angular velocities, and $q_5 = 10$ is the penalty of the integrator state. The matrix P is the solution to the infinite-horizon linear quadratic regulator problem.

The symmetries of the quadrotor were identified in Section 5.5. We found $|\operatorname{Aut}(\Sigma)| = 16$ symmetries with $|\mathcal{S}| = 3$ generators. The state-space generators Θ are given by (5.63) and the input-space generators Ω are given by (5.64).

We generated an explicit model predictive controller $\kappa(x)$ for the model predictive control problem described in [19, 7]. For a horizon of $N = 2$ the explicit controller $\kappa(x)$ has $|\mathcal{I}| = 10173$ regions and requires 53.7 megabytes of memory.

From the full explicit controller $\kappa(x)$ we constructed an orbit controller $\kappa^o(x)$ by discarding pieces $(F_i, G_i, \mathcal{R}_i)$ of the full explicit controller $\kappa(x)$ that are redundant under the symmetry group $\mathcal{G} = \operatorname{Aut}(MPC)$. The orbits $\mathcal{G}(i) \subset \mathcal{I}$ of the explicit controller $\kappa(x)$ are classified by their size $|\mathcal{G}(i)|$ in Table 7.1. The total number of controller pieces in the orbit controller $\kappa^o(x)$ is $|\mathcal{I}/\mathcal{G}| = 772$ which requires 4.2 megabytes of memory. Thus the $|\operatorname{Aut}(\Sigma)| = 16$ symmetries of the quadrotor reduced memory footprint of the explicit controller by a factor of approximately 12.8.

7.2 Fundamental Domain Controller

In this section we define the fundamental domain controller $\hat{\kappa}(x)$ which uses symmetry to reduce the memory complexity of storing the optimal control-law $u_0^\star(x) = \kappa(x)$.

Orbit Size $\lvert\mathcal{G}(i)\rvert$	Number of Orbits
1	1
2	6
4	30
8	215
16	520
Total Number of Orbits $\lvert\mathcal{I}/\mathcal{G}\rvert$:	772

Table 7.1: Classification of the orbits $\mathcal{G}(i)$ for $i \in \mathcal{I}$ of controller pieces for the quadrotor explicit model predictive controller $\kappa(x)$

By the definition of controller symmetry, the optimal control-law $u_0^\star(x)$ at two feasible points $x, y \in \mathcal{X}_0$ is equivalent if there exists a state-space transformation $\Theta \in \mathrm{Aut}(u_0^\star)$ such that $y = \Theta x$. Thus a group $\mathcal{G} \subseteq \mathrm{Aut}(u_0^\star)$ will partition the feasible state-space $\mathcal{X}_0$ into disjoint point orbits $\mathcal{G}x = \{\Theta x : \Theta \in \mathcal{G}\}$. The feasible region $\mathcal{X}_0$ is partitioned into an uncountable number of point orbits $\mathcal{X}_0/\mathcal{G}$. Recall that a fundamental domain $\hat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$ of the feasible state-space $\mathcal{X}_0$ is a subset of $\mathcal{X}_0$ that contains at least one representative from each orbit $\mathcal{G}x$. A polytopic fundamental domain $\hat{\mathcal{X}}_0$ is minimal if $\mathrm{int}(\hat{\mathcal{X}}_0 \cap \Theta\hat{\mathcal{X}}_0) = \varnothing$ for all $\Theta \neq I \in \mathcal{G}$.

The set $\hat{\mathcal{R}}$ in (7.6) is a fundamental domain of $\mathcal{X}_0$. Each point $x \in \mathcal{X}_0$ is contained in some region $\mathcal{R}_j$ of the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ and each region $\mathcal{R}_j$ is contained in a controller orbit $\mathcal{R}_j \in \{\Theta\mathcal{R}_i\}_{\Theta\in\mathcal{G}}$. Thus $\hat{\mathcal{R}}$ contains at least one representative x from each point orbit $\mathcal{G}x$. However working with the set $\hat{\mathcal{R}}$ is difficult since it is generally not convex nor connected. This means we must use brute-force techniques in the synthesis and implementation of the orbit controller.

To avoid this complication we are interested in using fundamental domains that are compact polytopes. Using the results of Chapter 6 we will construct an explicit model predictive controller that is defined on a polytopic fundamental domain.

Let us consider the model predictive control problem (7.1) restricted to a fundamental domain $\hat{\mathcal{X}}$ of the state-space $\mathcal{X}$ with respect to a subgroup of problem symmetries $\mathcal{G} \subseteq \mathrm{Aut}(MPC)$

$$J^\star(x) = \underset{u_0,\ldots,u_{N-1}}{\text{minimize}} \quad p(x_N) + \sum\nolimits_{k=0}^{N-1} q(x_k, u_k) \tag{7.25a}$$

$$\text{subject to} \quad x_{k+1} = Ax_k + Bu_k \tag{7.25b}$$

$$x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U} \tag{7.25c}$$

$$x_N \in \mathcal{X}_N \tag{7.25d}$$

$$x_0 = x \in \hat{\mathcal{X}} \tag{7.25e}$$

where $p(x_N) + \sum_{k=0}^{N-1} q(x_k, u_k)$ is a linear or quadratic cost function. This problem is identical to the original model predictive problem (7.1) except that the initial state x_0 is restricted to the fundamental domain $x_0 \in \hat{\mathcal{X}}$. The feasible state-space $\hat{\mathcal{X}}_0$ for this problem is a fundamental domain of the feasible set $\mathcal{X}_0$ of the original model predictive control problem (7.1).

Lemma 3. *The set of initial conditions for which (7.25) is feasible $\hat{\mathcal{X}}_0 = \mathcal{X}_0 \cap \hat{\mathcal{X}}$ is a minimal fundamental domain of $\mathcal{X}_0$.*

Proof. If $x \in \hat{\mathcal{X}}$ then problems (7.1) and (7.25) are identical. Thus $x \in \hat{\mathcal{X}}$ is feasible for (7.25) if and only if it is feasible for (7.1) $x \in \mathcal{X}_0 \cap \hat{\mathcal{X}}$. Therefore $\hat{\mathcal{X}}_0 = \mathcal{X}_0 \cap \hat{\mathcal{X}}$ is the set of feasible initial conditions for (7.25).

Next we prove $\hat{\mathcal{X}}_0 = \mathcal{X}_0 \cap \hat{\mathcal{X}}$ is a fundamental domain. Since $\hat{\mathcal{X}}$ is a minimal fundamental domain and $\mathcal{X}_0$ is symmetric $\mathcal{X}_0 = \Theta\mathcal{X}_0$, the set $\hat{\mathcal{X}}_0 = \hat{\mathcal{X}} \cap \mathcal{X}_0$ satisfies

$$\begin{aligned} \bigcup_{\Theta\in\mathcal{G}} \Theta\hat{\mathcal{X}}_0 &= \bigcup_{\Theta\in\mathcal{G}} \Theta\mathcal{X}_0 \cap \Theta\hat{\mathcal{X}} \\ &= \bigcup_{\Theta\in\mathcal{G}} \mathcal{X}_0 \cap \Theta\hat{\mathcal{X}} \;\; = \mathcal{X}_0 \end{aligned} \tag{7.26}$$

and

$$\begin{aligned} \hat{\mathcal{X}}_0 \cap \Theta\hat{\mathcal{X}}_0 &= \mathcal{X}_0 \cap \hat{\mathcal{X}} \cap \Theta\mathcal{X}_0 \cap \Theta\hat{\mathcal{X}} \\ &= \mathcal{X}_0 \cap \left(\hat{\mathcal{X}} \cap \Theta\hat{\mathcal{X}}\right) \end{aligned} \tag{7.27}$$

where $\hat{\mathcal{X}} \cap \Theta\hat{\mathcal{X}}$ has no interior for $\Theta \neq I \in \mathcal{G} \subseteq \mathrm{Aut}(MPC)$. Thus $\hat{\mathcal{X}}_0$ is a minimal fundamental domain of $\mathcal{X}_0$ for the group $\mathcal{G} \subseteq \mathrm{Aut}(MPC)$. □

Problem (7.25) is a multi-parametric linear or quadratic program and therefore has a piecewise affine explicit solution

$$\hat{\kappa}(x) = \begin{cases} \hat{F}_1 x + \hat{G}_1 & \text{for } x \in \hat{\mathcal{R}}_1 \\ \quad\vdots & \\ \hat{F}_r x + \hat{G}_r & \text{for } x \in \hat{\mathcal{R}}_{\hat{r}} \end{cases} \tag{7.28}$$

where $\hat{\mathcal{I}} = \{1, \ldots, \hat{r}\}$ is the index set of the controller pieces. We call $\hat{\kappa}(x)$ the fundamental domain controller. We can extend this controller from $\hat{\mathcal{X}}_0$ to the domain $\mathcal{X}_0$ of the full model predictive controller $\kappa(x)$ using symmetry

$$\bar{\kappa}(x) = \begin{cases} \Omega_1 \hat{\kappa}(\Theta_1^{-1} x) & \text{for } x \in \Theta_1\hat{\mathcal{X}} \\ \quad\vdots & \\ \Omega_{|\mathcal{G}|} \hat{\kappa}(\Theta_{|\mathcal{G}|}^{-1} x) & \text{for } x \in \Theta_{|\mathcal{G}|}\hat{\mathcal{X}}. \end{cases} \tag{7.29}$$

This solution covers $\mathcal{X}_0$ since $\{\Theta\mathcal{X}_0\}_{\Theta\in\mathcal{G}}$ is a partition of $\mathcal{X}_0$. The following theorem shows that $\bar{\kappa}(x)$ is a solution to the model predictive control problem (7.1).

Theorem 20. *The controller $\bar{\kappa}(x)$ in* (7.29) *is a solution to the model predictive control problem* (7.1).

Proof. For $x \in \hat{\mathcal{X}}_0$ the controller $\bar{\kappa}(x) = \hat{\kappa}(x)$ is feasible and optimal since $\hat{\kappa}(x)$ is the restriction of $\kappa(x)$ to $\hat{\mathcal{X}}_0$ for some optimal solution $u_0^\star(x) = \kappa(x)$ to 7.1.

Suppose $x \in \mathcal{X}_0$ but $x \notin \hat{\mathcal{X}}_0$. Then there exists $\Theta \in \mathcal{G}$ such that $\Theta^{-1}x \in \hat{\mathcal{X}}_0$ since $\hat{\mathcal{X}}_0$ is a fundamental domain of $\mathcal{X}_0$. Thus $\hat{\kappa}(\Theta^{-1}x)$ is defined.

By Theorem 8 the controller $\bar{\kappa}(x) = \Omega\hat{\kappa}(\Theta^{-1}x)$ is feasible since $\bar{\kappa}(x) = \Omega\hat{\kappa}(\Theta^{-1}x) \in \Omega\mathcal{U} = \mathcal{U}$ and

$$\begin{aligned} f\big(x, \bar{\kappa}(x)\big) &= f\big(x, \Omega\hat{\kappa}(\Theta^{-1}x)\big) \\ &= \Theta f\big(\Theta^{-1}x, \hat{\kappa}(\Theta^{-1}x)\big) \in \Theta\mathcal{X}_1 = \mathcal{X}_1 \end{aligned} \tag{7.30}$$

where $\Theta^{-1}x \in \hat{\mathcal{X}}_0 \subseteq \mathcal{X}_0$.

Furthermore by Theorem 8 the controller $\bar{\kappa}(x) = \Omega\hat{\kappa}(\Theta^{-1}x)$ is optimal since

$$\begin{aligned} q(x, \Omega\hat{\kappa}(\Theta^{-1}x)) + J_1^\star\big(f(x, \Omega\hat{\kappa}(\Theta^{-1}x)\big) &= q(\Theta^{-1}x, \hat{\kappa}(\Theta^{-1}x)) + J_1^\star(f(\Theta^{-1}x, \hat{\kappa}(\Theta^{-1}x)) \\ &= J_0^\star(\Theta^{-1}x) = J_0^\star(x). \end{aligned} \tag{7.31}$$

□

The concept of a fundamental domain controller is illustrated in the following example.

Example 30. *In this example we construct a fundamental domain controller $\hat{\kappa}(x)$ for the model predictive control problem described in Example 26.*

We construct a fundamental domain controller $\hat{\kappa}(x)$ by solving the model predictive control problem on the fundamental domain shown in Figure 7.5a. The partition $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ and vector field $u(x) = \hat{\kappa}(x)$ of the resulting fundamental domain controller $\hat{\kappa}(x)$ are shown in Figure 7.5b and 7.5c respectively. Comparing Figures 7.1a and 7.5b it is clear that $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ is the intersection of the partition $\{\mathcal{R}_i\}_{i\in\mathcal{I}}$ of the full explicit controller $\kappa(x)$ with the fundamental domain $\hat{\mathcal{X}}$.

The blue region $\hat{\mathcal{X}}_0$ in Figure 7.5c is the domain of the fundamental domain controller $\hat{\kappa}(x)$. From this figure it is clear that $\hat{\mathcal{X}}_0 = \hat{\mathcal{X}} \cap \mathcal{X}_0$ is a fundamental domain of the domain $\mathcal{X}_0$ of the original controller $\kappa(x)$. Thus through symmetry, the domain $\hat{\mathcal{X}}_0$ of the fundamental domain controller $\hat{\kappa}(x)$ covers the entire feasible state-space $\mathcal{X}_0$.

Synthesis of the Fundamental Domain Controller

In this section we describe how to synthesize the fundamental domain controller.

The synthesis of the fundamental domain controller $\hat{\kappa}(x)$ is described by Procedure 5. First we identify the symmetries $\mathrm{Aut}(MPC)$ of the model predictive control problem (7.1) using the techniques presented in Chapter 5. Next we construct a fundamental domain $\hat{\mathcal{X}}$

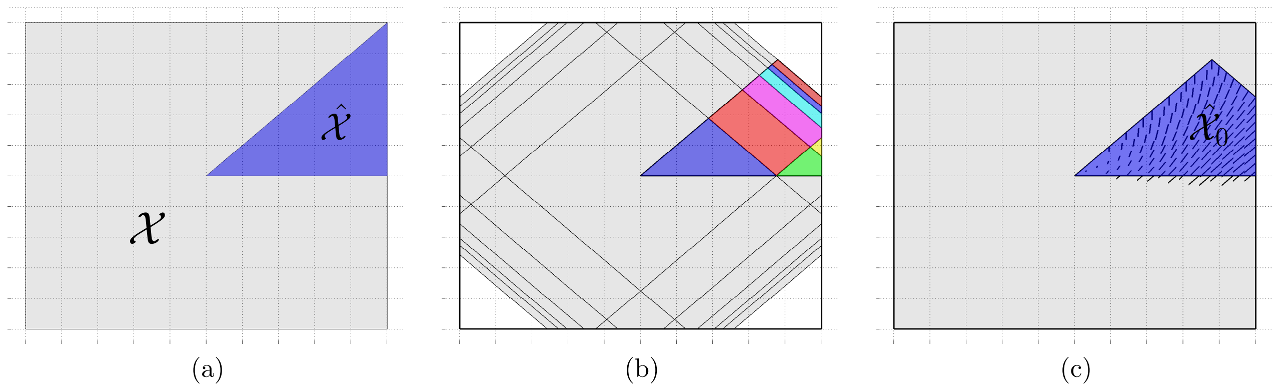


Figure 7.5: Example of fundamental domain controller synthesis. (a) Fundamental-domain $\hat{\mathcal{X}}$ of the set $\mathcal{X}$ for the dihedral-4 group $\mathcal{G} = \mathrm{Aut}(MPC) = \mathcal{D}_4$. (b) The partition $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ of the fundamental domain controller $\hat{\kappa}(x)$. (c) Vector-field $u(x) = \hat{\kappa}(x)$ of the fundamental domain controller $\hat{\kappa}(x)$ defined on the set $\hat{\mathcal{X}}_0$.

of the state-space $\mathcal{X}$ for $\mathcal{G} \subseteq \mathrm{Aut}(MPC)$. This can be done using Algorithm 8 presented in Chapter 6. Finally we solve the model predictive control problem (7.25) using Multi-Parametric Toolbox [57] for MATLAB. The solution piecewise affine solution $\hat{\kappa}(x)$ to (7.25) completely defines the control-law $u_0^\star(x) = \bar{\kappa}(x)$ for all $x \in \mathcal{X}_0$.

Procedure 5 Synthesis of Fundamental Controller $\hat{\kappa}(x)$

```
1: Identify the symmetries $\mathrm{Aut}(MPC)$ of the model predictive control problem (7.1).
2: Construct a fundamental domain $\hat{\mathcal{X}}$ of $\mathcal{X}$ under $\mathcal{G} = \mathrm{Aut}(MPC)$ using Algorithm 8.
3: Solve the model predictive control problem (7.25) for the fundamental domain controller
   $\hat{\kappa}(x)$.
```

Implementation of the Fundamental Domain Controller

In this section we describe the implementation of the fundamental domain controller $\hat{\kappa}(x)$.

The fundamental domain controller $\hat{\kappa}(x)$ is implemented using Algorithm 15. The point location problem has two phases. In the first phase the algorithm searches the symmetry group $\mathcal{G} \subseteq \mathrm{Aut}(MPC)$ for a state-space transformation $\Theta \in \mathcal{G}$ that maps the measured state $x \in \mathcal{X}_0$ into the fundamental domain $\hat{\mathcal{X}}_0$. This can be accomplished using Algorithm 10 in Chapter 6. In the second phase of the point location problem, the algorithm searches the partition $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ of the piecewise affine controller $\hat{\kappa}(x)$ for the region that contains the state $\Theta^{-1}x \in \hat{\mathcal{R}}_i$. This search can be implemented using any of techniques proposed in the literature [80, 50, 23, 59].

Finally the optimal control is calculated using the symmetry of the optimal control-law

$$u_0^\star = \Omega \hat{F}_i \Theta^{-1} x + \Omega \hat{G}_i \tag{7.32}$$

where $(\Theta, \Omega) \in \mathcal{G}$ is the symmetry found in the point location phase, and $(\hat{F}_i, \hat{G}_i)$ are the optimal feedback and feedforward gains for region $\hat{\mathcal{R}}_i$.

Algorithm 15 Implementation of Fundamental Controller $\bar{\kappa}(x)$

```
Input: Measured State $x \in \mathcal{X}_0$
Output: Optimal Control $u_0^\star(t) \in \mathcal{U}$
1: Find $(\Theta, \Omega) \in \mathcal{G}$ such that $\Theta x \in \hat{\mathcal{X}}_0$
2: Find $\hat{\mathcal{R}}_j \in \{\hat{\mathcal{R}}_i\}_{i \in \hat{\mathcal{I}}}$ such that $\Theta x \in \hat{\mathcal{R}}_j \subseteq \hat{\mathcal{X}}_0$
3: Calculate optimal control $u_0^\star(t) = \Omega^{-1}\hat{F}_i \Theta x + \Omega^{-1}\hat{G}_i$
```

The computational complexity of Algorithm 15 is dominated by the point location problem. Finding a state-space symmetry $\Theta \in \mathcal{G}$ that maps the measured-state x into the fundamental domain $x \in \hat{\mathcal{X}}$ requires $O(nc_x)$ time where c_x is the number of non-redundant half-spaces defining the state constraint set $\mathcal{X}$. The computational complexity of searching the partition $\{\hat{\mathcal{R}}\}_{i\in\hat{\mathcal{I}}}$ of $\hat{\kappa}(x)$ is $O(n \log |\hat{\mathcal{I}}|)$ [50].

Searching the group $\mathcal{G}$ and the controller partition $\mathcal{I}$ have comparable computational complexities $O(nc_x) = O(n \log |\mathcal{I}|)$ since the number of critical region $|\mathcal{I}|$ grows exponentially with the number of constraints $O(|\mathcal{I}|) = 2^c$ where $c \geq c_x$ is the total number of constraints in the multi-parametric program (7.1). Thus the implementation of the fundamental domain controller $\hat{\kappa}(x)$ has the same complexity as the implementation of the full explicit controller $\kappa(x)$.

The following example illustrates the implementation of the fundamental domain controller.

Example 31. *In this example we demonstrate the implementation of the fundamental domain controller $\hat{\kappa}(x)$ from Example 30.*

Consider the measured state x shown in Figure 7.6a. Since this state lies outside the domain $\hat{\mathcal{X}}_0$ of the fundamental domain controller $\hat{\kappa}(x)$ the optimal control at this point must be reconstructed using symmetry. This can be accomplished using Algorithm 15.

In the first step Algorithm 15 searches the symmetry group $\mathcal{G} = \mathrm{Aut}(MPC)$ for a state-space transformation $\Theta \in \mathcal{G}$ that maps the measured state x into the fundamental domain $\hat{\mathcal{X}}$ shown in Figure 7.6b. This can be accomplished using Algorithm 10 in Chapter 6. Algorithm 10 maps the state $x \in \mathcal{X}$ into the fundamental domain $\hat{\mathcal{X}} = \mathcal{F}_0(z_1) \cap \mathcal{F}_1(z_2)$ by first mapping it into the set $\mathcal{F}_0(z_1)$ shown in Figure 7.6c and then into the set $\mathcal{F}_1(z_2)$ shown in Figure 7.6d. The state $x \in \mathcal{X}$ can be mapped into $\mathcal{F}_0(z_1)$ by applying the 90 counter-clockwise rotation matrix Θ_r twice. The state $\Theta_r^{-2}x$ can then by mapped into the set $\mathcal{F}_1(z_2)$ by applying the vertical rotation Θ_s. Thus $\Theta_s^{-1}\Theta_r^{-2}x \in \hat{\mathcal{X}} = \mathcal{F}_0(z_1) \cap \mathcal{F}_1(z_2)$.

Next Algorithm 15 searches the partition $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ of the fundamental controller $\hat{\kappa}(x)$ for the region $\hat{\mathcal{R}}_i$ containing the state $\Theta_s^{-1}\Theta_r^{-1}\Theta_r^{-1}x$. In this case the yellow region shown in Figure 7.5b contains $\Theta_s^{-1}\Theta_r^{-1}\Theta_r^{-1}x$. This provides us with the appropriate feedback F_i and feedforward G_i gains.

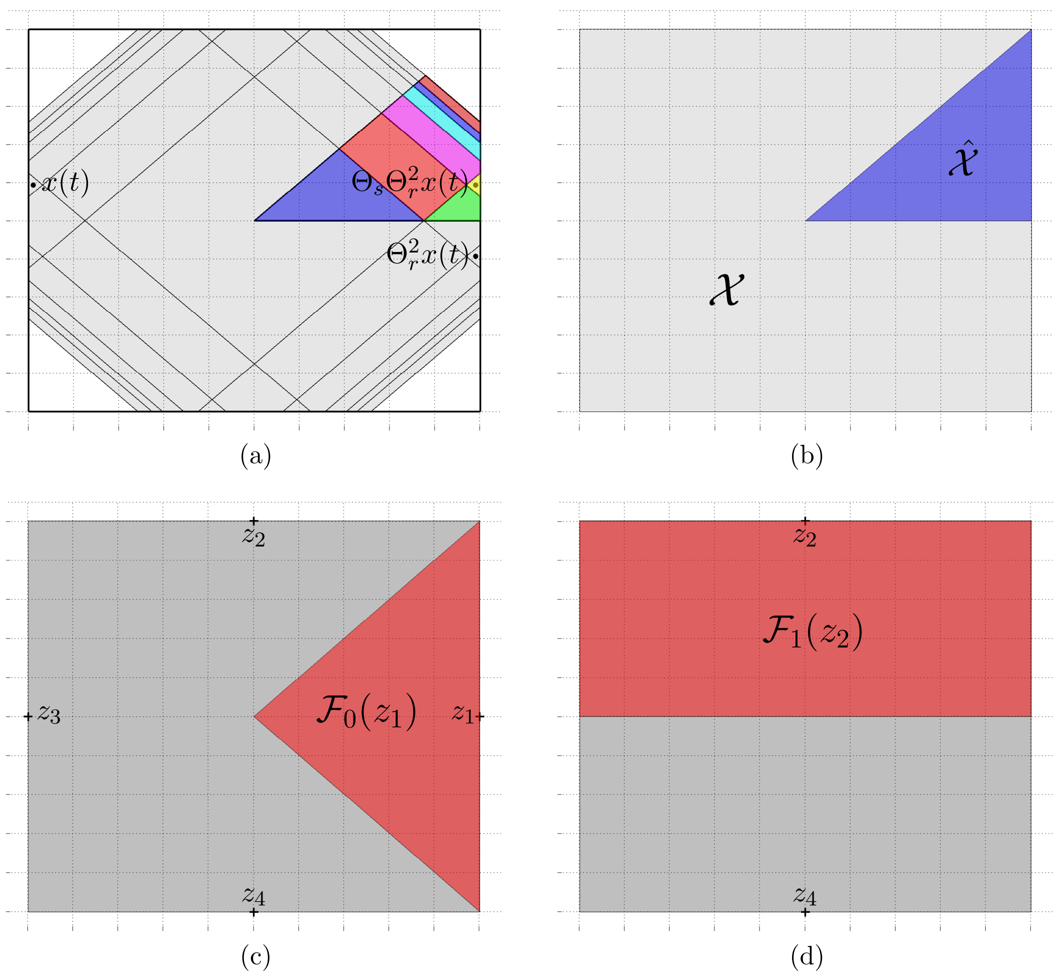


Figure 7.6: Example of the implementation of a fundamental domain controller. (a) Partition $\{\hat{\mathcal{R}}_i\}_{i\in\hat{\mathcal{I}}}$ of the fundamental domain controller $\hat{\kappa}(x)$. (b) Fundamental domain $\hat{\mathcal{X}}$ of the set $\mathcal{X}$ under the dihedral-4 group $\mathrm{Aut}(MPC) = \mathcal{D}_4$. (c) Voronoi cell $\mathcal{F}_0(z_1)$ of point z_1 under the group $\mathcal{G}_0 = \mathrm{Aut}(MPC)$. (d) Voronoi cell of facet z_2 under the subgroup $\mathcal{G}_1 = \langle\Theta_s\rangle$ which fixes z_1.

Finally the optimal control is given by

$$\begin{aligned} u_0^\star(t) &= \Omega F \Theta^{-1} x + \Omega G \\ &= \Omega_r\Omega_r\Omega_s F \Theta_s^{-1}\Theta_r^{-1}\Theta_r^{-1} x + \Omega_r\Omega_r\Omega_s G. \end{aligned} \tag{7.33}$$

Example: Quadrotor

In this section we apply the concept of a fundamental domain controller to the quadrotor system described in Section 5.5 and shown in Figure 5.2. Recall that our model of the quadrotor has $m = 4$ inputs corresponding to the voltages applied to the four motors and $n = 13$ states: 6 cartesian position and velocity states, 6 angular position and velocity states, and 1 integrator state. The continuous-time nonlinear dynamics are given by (5.61). The dynamics were linearized and discretized using standard techniques. The states and inputs are subject to constraints that upper-bound and lower-bound their values given by (5.62).

Our model predictive controller uses a quadratic cost function

$$J(X, U) = x_N^T P x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + \rho u_k^T u_k \tag{7.34}$$

where $\rho = 0.1$. The diagonal matrix Q has the form

$$Q = \begin{bmatrix} q_1 I_3 & & & & \\ & q_2 I_3 & & & \\ & & q_3 I_3 & & \\ & & & q_4 I_3 & \\ & & & & q_5 \end{bmatrix} \tag{7.35}$$

where $q_1 = q_3 = 10$ is the penalty on the cartesian and angular positions, $q_2 = q_4 = 1$ is the penalty on the cartesian and angular velocities, and $q_5 = 10$ is the penalty of the integrator state. The matrix P is the solution to the infinite-horizon linear quadratic regulator problem.

The symmetries of the quadrotor were identified in Section 5.5. We found $|\operatorname{Aut}(\Sigma)| = 16$ symmetries with $|\mathcal{S}| = 3$ generators. The state-space generators Θ are given by (5.63) and the input-space generators Ω are given by (5.64).

For a horizon of $N = 2$, we generated the full explicit model predictive controller $\kappa(x)$. Generating this controller required 2.1 hours. We were unable to generate the full explicit controller for a longer horizon. However using the fundamental domain controller we were able to extend the controller horizon to $N = 3$. This is the horizon of the implicit MPC used in [19, 7]. The fundamental domain controller had 6493 regions and required 0.8 hours to compute. Increasing the controller horizon will improve the closed-loop performance. This demonstrates one of the main advantages of our symmetric controllers: for a fixed memory size we can produce a controller with a longer horizons and therefore better closed-loop performance.

Part III

Application: Battery Balancing

Chapter 8

Battery Balancing

In this chapter we introduce the battery balancing problem which will be used as an example to demonstrate the theory of symmetric model predictive control developed in the previous chapters.

Cell-balancing is the process of balancing the state-of-charge of the cells in a battery pack. The state-of-charge of battery cells naturally tends to drift due to manufacturing differences, environmental factors, and aging. Over the course of several charge/discharge cycles the state-of-charge of the battery cells drifts apart. In older battery technologies, such as lead-acid, the imbalance is corrected using a controlled over-charge [20]. This process can cause permanent damage to lithium-based batteries. Instead lithium-based batteries employ a system called a *battery balancer* to balance the cells. There are two classes of battery balancers: dissipative and redistributive. In dissipative balancing, excess charge is drawn from the cells with the highest state-of-charge and dissipated through a resistor. In redistributive balancing, charge is exchanged between battery cells. Information about battery balancing hardware can be found in [20, 32, 66, 61].

Cell-balancing can increase the effective capacity of the battery pack and improve its durability. The active materials in lithium-based batteries can be damaged by overcharging or over-discharging. Since all the cells are charged simultaneously, charging must stop when one of the cells has reached the maximum state-of-charge. This leaves the other battery cells partially uncharged, reducing the effective capacity of the entire battery pack. Likewise, the battery system can only be discharged until one of the cells is depleted. This leaves the remaining charge in the other cells inaccessible and thereby reducing the effective capacity.

The time-scale of cell balancing is typically measured in minutes or hours. Therefore our model neglects the fast dynamics of the power electronics and internal battery chemistry. We model the battery cells using simple integrator dynamics. Higher fidelity mathematical battery models can be found in [77]. We consider seven hardware designs proposed in the literature [20, 32, 66, 61], although our balancing controller will work for general balancing systems. Throughout this chapter, we assume that the battery state-of-charge is known. Details on estimating the state-of-charge for battery cells can be found in [69, 76].

We begin this chapter by defining the battery balancing problem. We define the dynamics, constraints, and control objectives. In the next subsection we propose a controller for balancing the battery cells and prove its stability, persistent feasibility, and constraint satisfaction. Next we present seven battery balancing hardware designs described in the literature. In the next chapter we will exploit symmetry to reduce the complexity of our balancing controller for these seven hardware configurations. Finally we present simulation results for each of the battery balancing hardware designs in closed-loop with our battery balancing controller.

8.1 Battery System

The amount of charge stored in the battery cells is $Q_x x(t) \in \mathbb{R}^n_+$ where $x \in [0,1]^n \subseteq \mathbb{R}^n_+$ is the normalized state-of-charge of the cells and the diagonal matrix $Q_x \in \mathbb{R}^{n \times n}_+$ contains the charge capacities of the battery cells. State-of-charge is the amount of stored charge normalized by the total charge capacity of the cell. A state-of-charge of $x_i = 1$ corresponds to a full battery cell and $x_i = 0$ corresponds to an empty cell.

The state-of-charge of the battery cells is controlled through the balancing current $Q_u u(t) \in \mathbb{R}^m$ where $u(t) \in [-1,1]^m$ is the normalized balancing current and the diagonal matrix $Q_u \in \mathbb{R}^{m \times m}_+$ contains the current capacities of the balancing hardware.

The state-of-charge and balancing current are related by the integrator dynamics

$$Q_x \dot{x}_i(t) = T Q_u u(t) \tag{8.1}$$

where the topology matrix T relates the balancing currents and state-of-charge. The topology matrix T depends on the balancing hardware topology. In Section 8.3 we will define the balancing topology matrices T for the hardware topologies considered in this dissertation.

For notational simplicity we write the dynamics as

$$\dot{x}(t) = Bu(t) \tag{8.2}$$

where $B = Q_x^{-1} T Q_u \in \mathbb{R}^{n \times m}$.

In practice we use discrete-time controllers. The dynamics (8.2) in discrete-time are

$$x(t_{k+1}) = x(t_k) + Bu(t_k)\Delta t \tag{8.3}$$

where Δt is the sample-time, $t_k = t_0 + k\Delta t$ for $k \in \mathbb{N}$, and the input $u(t) = u(t_k)$ is held constant over the sample-period Δt.

The state-of-charge and normalized balancing current are constrained by

$$x(t) \in \mathcal{X} \tag{8.4a}$$

$$u(t) \in \mathcal{U} \tag{8.4b}$$

for all $t \in \mathbb{R}_+$ where $\mathcal{X} = [0,1]^n$ and the input constraints $\mathcal{U} \subseteq [-1,1]^m$ depend on the hardware topologies described in Section 8.3.

We consider two types of battery balancing systems: systems that redistribute charge between the battery cells and systems that dissipate charge from the battery cells. We make the following assumptions about the topology matrix $T \in \mathbb{R}^{n \times m}$ and the input constraint set $\mathcal{U} \subseteq \mathbb{R}^m$ for these two classes of systems.

Assumption 1 (Redistributive System)**.**

1. The topology matrix $T \in \mathbb{R}^{n \times m}$ satisfies $\text{range}(T) = \text{null}(1^T)$.
2. The input constraint set $\mathcal{U} \subset \mathbb{R}^m$ is full-dimensional and contains the origin in its interior.

The assumption that $\text{range}(T) = \text{null}(1^T)$ means that the battery balancing system can arbitrarily redistribute charge in the battery pack but cannot change the total amount of charge. In particular if charge is redistributed over a network, the condition $\text{range}(T) = \text{null}(1^T)$ ensures that the network is connected. The assumption that $\mathcal{U} \subset \mathbb{R}^m$ is full-dimensional means that bidirectional current flow is allowed on every link. The assumption that $\mathcal{U} \subseteq \mathbb{R}^m$ contains the origin $0 \in \mathcal{U}$ means that it is always possible to not move any charge between battery cells.

Assumption 2 (Dissipative System)**.**

1. The topology matrix is $T = -I \in \mathbb{R}^{n \times n}$.
2. The input constraint set $\mathcal{U} \subset \mathbb{R}^m_+$ is full-dimensional and contains the origin.

The assumption that $T = -I \in \mathbb{R}^{n \times n}$ means that every battery cells has its own circuit to dissipate charge. The assumption that $\mathcal{U}$ is full-dimensional means that every cell can dissipate charge since $\mathcal{U} \subset \mathbb{R}^m_+$. The assumption that $\mathcal{U} \subset \mathbb{R}^m_+$ contains the origin means that it is always possible to not dissipate any charge.

Reachable Sets

In this section we introduce the set of all states which can be reached from an initial state $x_0 = x(0)$ by choosing an appropriate control action. This set will be used in the next section to define the sets of equilibria to be controlled to and stabilized by feedback control.

Definition 10 (N-step Reachable Set)**.** The N-step reachable set from $x_0 \in \mathcal{X}$ for system (8.3) subject to the constraints (8.4) is defined recursively as

$$\mathcal{R}_{k+1}(x_0) = \text{Reach}(\mathcal{R}_k) \cap \mathcal{X} \tag{8.5}$$

for $k = 1, \dots, N-1$ and $\mathcal{R}_0(x_0) = \{x_0\}$.

Definition 11 (∞-step Reachable Set)**.** The ∞-step reachable set from $x_0 \in \mathcal{X}$ for system (8.3) subject to the constraints (8.4) is defined as

$$\mathcal{R}_\infty(x_0) = \lim_{N\to\infty} \mathcal{R}_N(x_0). \tag{8.6}$$

The following two propositions characterize the set of reachable states for the system (8.3) subject to the constraints (8.4) for redistributive and dissipative balancing system respectively.

Proposition 20. *Let T and $\mathcal{U}$ satisfy Assumption 1. Then the ∞-step reachable set for the system (8.2) subject to the constraints (8.4) is*

$$\mathcal{R}_\infty(x_0) = \Big\{ x \in \mathcal{X} : \mathbf{1}^T Q_x x = \mathbf{1}^T Q_x x_0 \Big\}. \tag{8.7}$$

Proof. First we show that reachable states $x_\infty \in \mathcal{R}_\infty$ must satisfy the conservation law

$$\mathbf{1}^T Q_x x_\infty = \mathbf{1}^T Q_x x_0 \tag{8.8}$$

where $\mathbf{1} \in \mathbb{R}^n$ is the vector of ones. By Assumption 1 we have $\mathbf{1}^T T = 0$ and thus from the dynamics (8.3) we have

$$\begin{aligned} \mathbf{1}^T Q_x x(t_{k+1}) &= \mathbf{1}^T Q_x x(t_k) + \mathbf{1}^T T Q_u \Delta t u(t_k) \\ &= \mathbf{1}^T Q_x x(t_k) \end{aligned} \tag{8.9}$$

Therefore charge is conserved.

Next we show every state $x_\infty \in \mathcal{X}$ that satisfies (8.8) is reachable $x_\infty \in \mathcal{R}_\infty$. The linear equation

$$Q_x x_\infty - Q_x x_0 = T Q_u \Delta t v \tag{8.10}$$

has a solution $v \in \mathbb{R}^m$ since $Q_x x_\infty - Q_x x_0$ is in the range-space of T by (8.8). Since $\mathcal{U}$ contains the origin in its interior there exists $\epsilon > 0$ such that the ball $\mathcal{B}(0,\epsilon) \subset \mathcal{U}$ is contained in $\mathcal{U}$. There exists $N \in \mathbb{N}$ such that $\frac{\|v\|}{N} \leq \epsilon$. Thus the input sequence

$$u(t_k) = \begin{cases} \frac{v}{N} & \text{for } k = 0, \ldots, N-1 \\ 0 & \text{for } k \geq N \end{cases} \tag{8.11}$$

is feasible $u(t_k) \in \mathcal{U}$ for all $k \in \mathbb{N}$ and reaches $x_\infty = x_0 + B\sum_{k=0}^{\infty} u(t_k)$. Every intermediate state $x(t_j) = x_0 + B\sum_{k=0}^{j} u(t_k)$ is feasible $x(t_j) \in \mathcal{X}$ by the convexity of $\mathcal{X}$. Thus $x_\infty \in \mathcal{R}_\infty$ is reachable. □

Proposition 21. *Let T and $\mathcal{U}$ satisfy Assumption 2. Then the ∞-step reachable set for the system (8.2) subject to the constraints (8.4) is*

$$\mathcal{R}_\infty(x_0) = \Big\{ x \in \mathcal{X} : x \leq x_0 \Big\}. \tag{8.12}$$

Proof. Clearly any state $x \in \mathcal{X}$ such that $x_i(t_k) > x_i(0)$ for some $i \in \{1, \dots, n\}$ is not reachable since we cannot add charge to the cells. We will show that any state $x_\infty \in \mathcal{X}$ such that $x_\infty \leq x_0$ is reachable $x_\infty \in \mathcal{R}_\infty$. Since $\mathcal{U}$ is full-dimensional, convex, and contains the origin there exists $N \in \mathbb{N}$ such that $\frac{v}{N} \in \mathcal{U}$ where $v = x_0 - x_\infty \in \mathbb{R}^m_+$. Thus the input sequence

$$u(t_k) = \begin{cases} \frac{v}{N} & \text{for } k = 0, \dots, N-1 \\ 0 & \text{for } k \geq N \end{cases} \tag{8.13}$$

is feasible $u(t_k) \in \mathcal{U}$ for all $k \in \mathbb{N}$ and reaches $x_\infty = x_0 + B\sum_{k=0}^{\infty} u(t_k)$. Every intermediate state $x(t_j) = x_0 + B\sum_{k=0}^{j} u(t_k)$ is feasible $x(t_j) \in \mathcal{X}$ by the convexity of $\mathcal{X}$. Thus $x_\infty \in \mathcal{R}_\infty$ is reachable. □

Battery Balancing Problem

The battery balancing problem involves finding a feasible balancing current trajectory that balances the battery cells. We define the set of balanced states as

$$\bar{\mathcal{X}} = \Big\{ x \in \mathbb{R}^n : \; x_i = x_j \; \forall i, j = 1, ..., n \Big\} \subseteq \mathcal{X} \tag{8.14}$$

In a balanced state, the amount of charge that can be removed from the battery pack and amount of charge that can be added to the battery pack are both maximized [30]. The battery balancing problem is formally defined below.

Problem 4 (Battery Balancing)**.** Find a terminal time τ and input trajectory $u(t)$ for $t \in [0, \tau]$ such that $x(t) \in \mathcal{X}$ and $u(t) \in \mathcal{U}$ for all $t \in [0, \tau]$, and $x(\tau) \in \bar{\mathcal{X}}$.

According to Propositions 20 and 21, the balanced set $\bar{\mathcal{X}}$ is reachable $\mathcal{R}_\infty(x_0) \cap \bar{\mathcal{X}} \neq \varnothing$ from any initial condition $x_0 \in \mathcal{X}$. Therefore Problem 4 always has a solution.

Performance Metrics

In this section we introduce two performance metrics; time to balance and energy dissipation during balancing. These performance metrics are used to evaluate the balancing trajectories $u(t)$ produced by our controller.

The time-to-balance $\tau(x, u)$ is the amount of time required to equalize the state-of-charge of the battery cells for a given initial imbalance $x \in \mathcal{X}$ and input trajectory $u(t) \in \mathcal{U}$ for $t \in [0, \infty)$

$$\tau(x, u) = \operatorname{argmin} \Big\{ \tau \in \mathbb{R}_+ : x(\tau) = x + B\int_0^\tau u(t)dt \in \hat{\mathcal{X}} \Big\}. \tag{8.15}$$

The energy-dissipated-to-balance $\varepsilon(u)$ is the amount of energy dissipated in the process of balancing the battery cells. Each charge transfer $u(t)$ dissipates a portion of the stored energy. The power dissipation is a non-linear function of the voltage and current [81, 71]. However in practice the power electronics operate about a single current set-point. Furthermore the voltage range of the battery cells is limited and can be upper-bounded. Therefore we assume the energy dissipation is directly proportional to the magnitude of the control action according to the relation

$$\varepsilon(u) = \int_0^\tau \|\eta u(t)\|_1 dt \tag{8.16}$$

where $\eta \in \mathbb{R}^{m \times m}$ is a diagonal matrix which defines the power dissipation of the links. Realistic efficiency ranges were obtained from [81, 71].

The ideal controller should be the solution $u^\star(t)$ to the infinite-horizon constrained optimal control problem

$$\begin{aligned} \underset{\tau, u(t)}{\text{minimize}} \quad & \tau + \rho \int_0^\infty \|\eta u(t)\|_1 dt && \text{(8.17a)} \\ \text{subject to} \quad & x + B \int_0^\tau u(t) dt \in \hat{\mathcal{X}} && \text{(8.17b)} \\ & u(t) \in \mathcal{U}, \; x(t) \in \mathcal{X} \quad \forall t \in \mathbb{R}_+ && \text{(8.17c)} \end{aligned}$$

where $x = x(t)$ is the measured state and the tuning parameter $\rho \in (0, \infty)$ weights the important of fast balancing verses energy efficient balancing. As $\rho \to 0$ equation (8.17) becomes the minimum-time control problem. As $\rho \to \infty$ equation (8.17) becomes the minimum-energy control problem.

8.2 Controller Design and Analysis

Our battery balancing controller is described in Algorithm 16. At each time instance t_k the controller measures or estimates the current state-of-charge $x(t_k)$ of the battery cells. The controller then solves a minimum-cost flow problem [8] to find the optimal total charge redistribution $v^\star$ needed to balanced the battery cells $x + Bv^\star \in \bar{\mathcal{X}}$. The balancing current $u(t_k)$ over the sample-period Δt is calculated by dividing the total charge redistribution $v^\star$ by a time-interval τ that ensures the current $u(t_k) = \frac{1}{\tau} v^\star \in \mathcal{U}$ is feasible. If the charge redistribution $v^\star$ can be accomplished in the sample-time Δt then we implement the current

$$u(t_k) = \frac{v^\star}{\Delta t} \in \mathcal{U}. \tag{8.18}$$

Otherwise we calculate the minimum amount of time $\tau = \Phi_{\mathcal{U}}(v^\star)$ a feasible current $u \in \mathcal{U}$ needs to be applied to accomplish the charge redistribution $v^\star$ where $\Phi_{\mathcal{U}}(v^\star) = \min\{\tau \in \mathbb{R}_+ : u \in \tau \mathcal{U}\}$ is the Minkowski function for the set $\mathcal{U}$. In this case we apply the saturated current

$$u(t_k) = \frac{v^\star}{\Phi_{\mathcal{U}}(v^\star)} \in \mathcal{U}. \tag{8.19}$$

Algorithm 16 Battery Balancing Controller

```
1: Measure (or estimate) the current state-of-charge $x = x(t_k)$
2: Solve the minimum-cost flow problem (8.20) to obtain the total charge redistribution $v^\star$
```

$$\begin{aligned} \underset{v \in \mathbb{R}^m}{\text{minimize}} \quad & \Phi_{\mathcal{U}}(v) + \rho \|\eta v\|_1 && (8.20a) \\ \text{subject to} \quad & x + Bv \in \bar{\mathcal{X}}. && (8.20b) \end{aligned}$$

```
3: Calculate the current $u(t_k)$
```

$$u(t_k) = \begin{cases} \frac{v^\star}{\Phi_{\mathcal{U}}(v^\star)} & \text{for } \Phi_{\mathcal{U}}(v^\star) > \Delta t \\ \frac{v^\star}{\Delta t} & \text{for } \Phi_{\mathcal{U}}(v^\star) \leq \Delta t \end{cases}. \tag{8.21}$$

In the next subsections we show that this control satisfies the state and input constraints, and is persistently feasible. We then prove that this controller stabilizes the equilibrium set $\hat{\mathcal{X}}$. In Chapter 9 we will use the symmetry to generate memory efficient explicit solutions to the one-step optimization problem (8.20).

Constraint Satisfaction and Feasibility

In this section we show that the controller described in Algorithm 16 satisfies the constraints and is persistently feasible. The following proposition shows that the control satisfies the state and input constraints.

Proposition 22. *Let $x(t_k) \in \mathcal{X}$ be feasible. Then the control input $u(t_k)$ produced by Algorithm 16 satisfies the state and input constraints*

$$\begin{aligned} u(t_k) &\in \mathcal{U} && (8.22a) \\ x(t_{k+1}) &\in \mathcal{X}. && (8.22b) \end{aligned}$$

Proof. First we show that $u(t_k)$ satisfies the input constraints $u(t_k) \in \mathcal{U}$. By the definition of the Minkowski function $\Phi_{\mathcal{U}}(u)$ the control input $u = u(t_k)$ is feasible $u \in \mathcal{U}$ if and only if $\Phi_{\mathcal{U}}(u) \leq 1$. By the properties of Minkowski functions we have

$$\Phi_{\mathcal{U}}\big(u(t_k)\big) \leq \Phi_{\mathcal{U}}\Big(\frac{v^\star}{\Phi_{\mathcal{U}}(v^\star)}\Big) = \frac{1}{\Phi_{\mathcal{U}}(v^\star)}\Phi_{\mathcal{U}}(v^\star) = 1. \tag{8.23}$$

Next we show that $x(t_{k+1})$ satisfies the state constraints $x(t_{k+1}) \in \mathcal{X}$. We can write $x(t_{k+1})$ as

$$x(t_{k+1}) = x(t_k) + Bu(t_k)\Delta t = \big(1 - \lambda\big)x(t_k) + \lambda\big(x(t_k)_B v^\star\big) \tag{8.24}$$

where $\lambda = \min\left\{\frac{\Delta t}{\Delta t}, \frac{\Delta t}{\Phi_{\mathcal{U}}(v^\star)}\right\} \in [0,1]$. By assumption $x(t_k) \in \mathcal{X}$ is feasible and by (8.20b) the point $x(t_k) + Bv^\star \in \bar{\mathcal{X}} \subset \mathcal{X}$ is feasible. Thus $x(t_{k+1})$ is the convex combination of feasible points. Since $\mathcal{X}$ is convex we have $x(t_{k+1})$ satisfies the state constraints $x(t_{k+1}) \in \mathcal{X}$. □

This proposition tacitly assumes that the one-step optimization problem (8.20) has a solution and therefore the control $u(t_k)$ is defined. The following proposition show that (8.20) is persistently feasible.

Proposition 23. *The system* (8.3) *subject to the constraints* (8.4) *in closed-loop with the controller in Algorithm 16 is persistently feasible for all* $x \in \mathcal{X}$*.*

Proof. By Propositions 20 and 21 for any $x \in \mathcal{X}$ there exists an input trajectory $\{u(t_k)\}_{k=0}^{\infty}$ such that

$$x + B\textstyle\sum_{k=0}^{\infty} u(t_k) \in \bar{\mathcal{X}}. \tag{8.25}$$

Since the set $\mathcal{X}$ and $\bar{\mathcal{X}}$ are compact there exists a subsequence $\{u(t_{k_j})\}_{j=0}^{\infty}$ such that the limit

$$\lim_{N\to\infty} x + B\textstyle\sum_{j=0}^{N} u(t_{k_j}) = \bar{x} \tag{8.26}$$

exists and $\bar{x} \in \bar{\mathcal{X}} \cap \mathcal{R}_\infty$ is balanced. By the properties of linear transformations there exists a finite vector $v \in \mathbb{R}^m$ such that $Bv = \bar{x} - x$. Thus we have shown there exists a vector $v \in \mathbb{R}^m$ that satisfies (8.20b) for every $x \in \mathcal{X}$.

Finally we need to show v is in the domain of the cost function, in particular the Minkowski function $\Phi_{\mathcal{U}}(v)$. For redistributive systems the domain of $\Phi_{\mathcal{U}}(v)$ is $\mathbb{R}^m$ since by Assumption 1 the set $\mathcal{U}$ is full-dimensional and contains the origin in its interior. For dissipative system we have $v = \bar{x} - x \in \mathbb{R}_+^m$ since $B = -T$ and $\bar{x} \in \mathcal{R}_\infty$. The domain of $\Phi_{\mathcal{U}}(v)$ is $\mathbb{R}_+^m$ since by Assumption 2 the set $\mathcal{U} \subset \mathbb{R}_+^m$ is full-dimensional and contains the origin.

Therefore we conclude (8.20) is feasible for all $x(t_k) \in \mathcal{X}$. □

Stability of Equilibria Set

In this section we show that the balancing controller described in Algorithm 16 stabilizes the set of balanced states $\bar{\mathcal{X}}$. We give the following definition of stability for a set of equilibria.

Definition 12 (Semistability [9])**.** The equilibrium set $\mathcal{X}^e$ is semistable if for all $x^e \in \mathcal{X}^e$ and for all $\epsilon > 0$ there exists $\delta > 0$ such that

$$\|x(0) - x^e\| < \delta \Rightarrow \|x(t) - x^e\| < \epsilon \tag{8.27}$$

for all $t \in \mathbb{N}$ and $\lim_{t\to\infty} x(t) = x_\infty \in \mathcal{X}^e$ for all $x(0) \in \mathcal{X}$.

For the battery systems, this definition says that if the state-of-charge x starts near $\|x(0)-\bar{x}\| < \delta$ a balanced state $\bar{x} \in \bar{\mathcal{X}}$ then it will stay near $\|x(t)-\bar{x}\| < \epsilon$ this state and converge to a state $x_\infty \in \bar{\mathcal{X}}$ in the set of balanced states $\bar{\mathcal{X}}$. This definition is important for balancing systems with dissipative balancing since there are multiple equilibria $x_\infty \in \bar{\mathcal{X}}$ to which the state can converge. The following theorem on set-valued Lyapunov functions provides a method for proving the stability of systems with sets of equilibrium points.

Theorem 21 (Set-Valued Lyapunov Functions[67]). *Let $\mathcal{X}^e$ be a set of equilibrium points and let $W : \mathcal{X} \to 2^{\mathcal{X}}$ be an upper-semicontinuous set-valued Lyapunov function satisfying*

1. $x \in W(x)\ \forall x \in \mathcal{X}$ *and* $W(x^e) = \{x^e\}\ \forall x^e \in \mathcal{X}^e$
2. $W(f(x)) \subseteq W(x)\ \forall x \in \mathcal{X}$
3. *There exists a function $\mu : \mathrm{Im}(W) \to \mathbb{R}_{\geq 0}$, bounded on bounded sets, such that*
$$\mu\big(W(f(x))\big) < \mu\big(W(x)\big) \tag{8.28}$$
for all $x \in \mathcal{X} \setminus \mathcal{X}^e$.

Then $\mathcal{X}^e$ is semistable and $x(t) \to x^e$ for some $x^e \in \mathcal{X}^e$ as $t \to \infty$.

Proof. See Theorem 4 in [67]. □

The following proposition uses this theorem to show the stability of the set of balanced states $\bar{\mathcal{X}}$ under our controller.

Proposition 24. *The set of balanced states $\bar{\mathcal{X}}$ is semistable for the system* (8.3) *in closed-loop with the controller described in Algorithm 16.*

Proof. Consider the candidate set-valued Lyapunov function
$$W(x) = \Big\{x + \lambda Bv : v \in V^\star(x), \lambda \in [0,1]\Big\} \tag{8.29}$$
where $V^\star(x)$ is the set of optimizers $v^\star$ of (8.20) at x. The set $W(x)$ is the convex-hull of the current state x and the set of equilibrium states $\bar{x} = x + Bv \in \bar{\mathcal{X}}$ that the controller wants to reach.

First we prove that $W(x)$ is upper-semicontinuous. Consider the graph
$$\text{graph} = \big\{(x,v) : x \in \mathcal{X}, v \in V^\star(x)\big\}. \tag{8.30}$$
This set is closed since it is the pre-image of the value function $J^\star(x)$ of (8.20) which is continuous. Thus by the closed-graph theorem [39] the set-valued function $F(x) = \{x + Bv : v \in V^\star(x)\}$ is upper-semicontinuous. We will use this fact to prove $W(x)$ is upper-semicontinuous.

Consider an open neighborhood $\mathcal{W} \supset W(z)$ of $W(x)$ at $x = z$. Let $\bar{\mathcal{W}} \subset \mathcal{W}$ be an open convex neighborhood of $W(x) \subset \bar{\mathcal{W}} \subseteq \mathcal{W}$ at $x = z$. Since $F(x)$ and the identity function $G(x) = x$ are upper-semicontinuous there exists an open neighborhood $\mathcal{Z} \subset \mathcal{X}$ of $z \in \mathcal{Z}$ such that $F(\mathcal{Z}) \subset \bar{\mathcal{W}}$ and $G(\mathcal{Z}) \subset \bar{\mathcal{W}}$. Since $\bar{\mathcal{W}}$ is convex we have $W(x) \subseteq \text{conv}(F(\mathcal{Z}), G(\mathcal{Z})) \subset \bar{\mathcal{W}} \subset \mathcal{W}$ for all $x \in \mathcal{Z}$. Thus we have shown that for every open neighborhood $\mathcal{W} \supset W(z)$ of $W(z)$ there exists a neighborhood $\mathcal{Z}$ of $z \in \mathcal{X}$ such that $W(x) \in \mathcal{W}$ for all $x \in \mathcal{Z}$. Therefore $W(x)$ is upper-semicontinuous.

Next we show $W(x)$ satisfies the conditions of the theorem.

1. For $\lambda = 0$ we see that $x \in W(x) = \{x + \lambda B v : v \in V^\star(x), \lambda \in [0,1]\}$. For a state $\bar{x} \in \bar{\mathcal{X}}$ in the equilibrium set $\bar{\mathcal{X}}$ we have $V^\star(\bar{x}) = \{0\}$. Thus $W(\bar{x}) = \{\bar{x}\}$.

2. Next we show $W(x + Bu\Delta t) \subseteq W(x)$. We will show that extreme-points of $W(x + Bu\Delta t)$ are contained in $W(x)$ thus by convexity $W(x + Bu\Delta t) \subseteq W(x)$.

   Consider the extreme-point $x + Bu\Delta t \in W(x + Bu\Delta t)$. According to Algorithm 16, we have $u\Delta t = \lambda \bar{v}$ for some $\bar{v} \in V^\star(x)$ and $\lambda = \min\{1, 1/\Phi(\bar{v})\} \in [0,1]$. Therefore $x + Bu\Delta t = x + \lambda B\bar{v} \in W(x)$.

   Now we consider the extreme-points $z + Bu\Delta t + Bv^+ \in W(z + Bu\Delta t)$ for some $v^+ \in V^\star(z + Bu\Delta t)$ where $x = z$. We will show that $v = u\Delta t + v^+ \in V^\star(z)$ and thus $z + B(u\Delta t + v^+) \in W(z)$. First note that $v = u\Delta t + v^+$ is a feasible solution to (8.20) for $x = z$ since $z + B(u\Delta t + v^+) \in \bar{\mathcal{X}}$ by the definition of $v^+ \in V^\star(z + Bu\Delta t)$. Next we show optimality.

   Since $u\Delta t = \lambda\bar{v}$ for some $\bar{v} \in V^\star(z)$ we have
   $$\Phi_{\mathcal{U}}(u\Delta t) + \rho\|\eta u\Delta t\|_1 \leq \Phi_{\mathcal{U}}(\lambda\bar{v}) + \rho\|\eta\lambda\bar{v}\|_1 = \lambda J^\star(z). \tag{8.31}$$

   Note that $\bar{v}^+ = (1-\lambda)\bar{v}$ is a feasible solution to (8.20) at $x = z + Bu\Delta t$ since $z + Bu\Delta t + B(1 - \lambda\bar{v}) = z + B\bar{v} \in \bar{\mathcal{X}}$. This provides an upper-bound on the cost $J^\star(x + Bu\Delta t)$ for any $v^+ \in V^\star(z + Bu\Delta t)$
   $$\begin{aligned}\Phi_{\mathcal{U}}(v^+) + \rho\|\eta v^+\|_1 &\leq (1-\lambda)\Phi_{\mathcal{U}}(\bar{v}) + (1-\lambda)\rho\|\eta\bar{v}\|_1 \\ &= (1-\lambda)J^\star(z).\end{aligned} \tag{8.32}$$

   Finally by the triangle inequality, we upper-bound the cost of $v = u + v^+$ for any $v^+ \in V^\star(x + Bu\Delta t)$
   $$\begin{aligned}\Phi_{\mathcal{U}}(u\Delta t + v^+) + &\rho\|\eta u\Delta t + v^+\|_1 \\ &\leq \lambda\Phi_{\mathcal{U}}(u\Delta t) + \lambda\rho\|\eta u\Delta t\|_1 + (1-\lambda)\Phi_{\mathcal{U}}(v^+) + (1-\lambda)\rho\|\eta v^+\|_1 \\ &= \lambda J^\star(z) + (1-\lambda)J^\star(z) = J^\star(z).\end{aligned} \tag{8.33}$$

   Thus $v = u\Delta t + v^+ \in V^\star(z)$ is an optimal solution to (8.20) at $x = z$ and therefore $z + Bu\Delta t + Bv^+ \in W(z)$.

   Finally since $W(x + Bu\Delta t)$ is the convex-hull of $z + Bu\Delta \in W(z)$ and $z + Bu\Delta t + Bv^+ \in W(z)$ for all $v^+ \in V^\star(z + Bu\Delta t)$ we conclude $W(z + Bu\Delta t) \subseteq W(z)$.

3. Finally we show $W(x)$ satisfies condition 3. Define the function

$$\mu(W(x)) = \sup_{z \in W(x)} d(z, \bar{\mathcal{X}}) \tag{8.34}$$

where $d(z, \bar{\mathcal{X}}) = \inf_{y \in \bar{\mathcal{X}}} \|z - y\|$ is the Hausdorff distance from $z \in W(x)$ to the equilibrium set $\bar{\mathcal{X}}$. Note that for any $z \in W(x)$ we have

$$\begin{aligned} d\big(z, \bar{\mathcal{X}}\big) &= d((1-\lambda)x + \lambda(x + Bv), \bar{\mathcal{X}}) \\ &\leq (1-\lambda)d(x, \bar{\mathcal{X}}) + \lambda d(x + Bv, \bar{\mathcal{X}}) \\ &= (1-\lambda)d(x, \bar{\mathcal{X}}) \\ &\leq d(x, \bar{\mathcal{X}}). \end{aligned} \tag{8.35}$$

Thus $\mu(W(x)) = d(x, \bar{\mathcal{X}})$ is the Hausdorff distance from the current state x to the equilibrium set $\bar{\mathcal{X}}$.

Next we show $d(x + Bu\Delta t, \bar{\mathcal{X}}) < d(x, \bar{\mathcal{X}})$. Since $\bar{\mathcal{X}}$ is a compact set there exists $\bar{x} \in \bar{\mathcal{X}}$ such that $d(x, \bar{\mathcal{X}}) = \|x - \bar{x}\|$. By convexity of $\bar{\mathcal{X}}$ we can define $y = (1-\lambda)\bar{x} + \lambda(x + Bv) \in \bar{\mathcal{X}}$. By the definition of Hausdorff distance we have

$$\begin{aligned} d(x + Bu\Delta t, \bar{\mathcal{X}}) &\leq \|x + Bu\Delta t - y\| \\ &= \|(1-\lambda)x + \lambda(x + Bu\Delta t) - (1-\lambda)\bar{x} + \lambda(x + Bv)\| \\ &= (1-\lambda)\|x + \bar{x}\| \\ &= (1-\lambda)d(x, \bar{\mathcal{X}}). \end{aligned} \tag{8.36}$$

For $x \notin \bar{\mathcal{X}}$ we have $\lambda > 0$ thus $d(x + Bu\Delta t, \bar{\mathcal{X}}) < d(x, \bar{\mathcal{X}})$.

Finally we conclude by Theorem 21 that the equilibrium set $\bar{\mathcal{X}}$ is semistable and $x(t) \to \bar{x} \in \bar{\mathcal{X}}$. □

8.3 Hardware Topologies

In this section we present seven models for battery balancing hardware designs described in the literature [20, 32, 66, 61]. We are concerned with modeling the behavior of the balancing hardware over long time-scales relative to the dynamics of the power electronics and battery chemistry. Each hardware topology is modeled using the generic model (8.2) and (8.4) where the topology matrix T and polytopic input constraint set $\mathcal{U}$ vary between hardware topologies. For a comparison of the performance of these battery balancing topologies see [70].

Dissipative Topology

The dissipative topology is shown in Figure 8.1. In the dissipative topology charge is removed from cells with high state-of-charge and dissipated as waste heat. Examples of dissipative topologies are the dissipative resistor and dissipative shunting topologies from [20, 32, 66, 61].

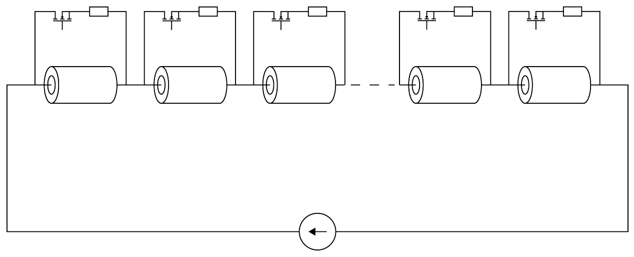


Figure 8.1: Dissipative topology. Charge from individual cells is dissipated through a resistor [70].

In the dissipative shunting topology each cell has an independently actuated shunt circuit. The topology matrix $T = -I$ is given by

$$T = \begin{bmatrix} -1 & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{8.37}$$

where n is the number of cell and $m = n$ is the number of dissipative links.

The input u is limited by peak current constraint of the links $|u_i| \leq 1$ and since the topology is dissipative we have $u_i \geq 0$ for each $i = 1, 2, \ldots, m$. Dissipating excess charge produces heat that can damage the battery cells. Therefore we limit the total dissipation current $\sum_{i=1}^{m} u_i \leq u_{\max}$ where $u_{\max}$ is the maximum current that can be dissipated without overheating. The input constraint set is

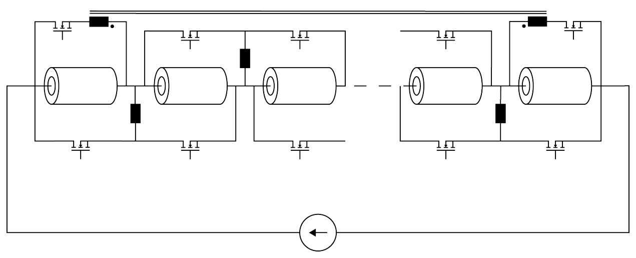
$$\mathcal{U} = \left\{ u \in \mathbb{R}^m \mid 0 \leq u_i \leq 1, \textstyle\sum_{i=1}^{m} u_i \leq u_{\max} \right\} \subseteq [0,1]^m. \tag{8.38}$$

Shunting Topologies

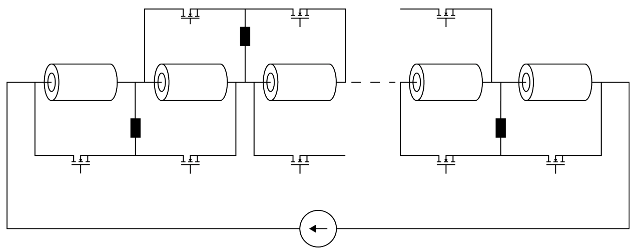
The shunting topologies are shown in Figure 8.2. In the line topology shown in Figure 8.2b charge is moved between neighboring cells in the series connection using buck-boost converters. In the ring topology shown in Figure 8.2a the first and last cells in the stack are connect to form a ring using a flyback converter. Examples of shunting topologies are the controlled shunting and switched capacitor topologies from [20, 32, 66, 61].

The topology matrix T for the line shunting topology is given by

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & & \vdots \\ 0 & -1 & \ddots & 0 \\ \vdots & & \ddots & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{8.39}$$



(a) Ring



(b) Line

Figure 8.2: Shunting topologies. Charge is moved between neighboring cells in the series connection [70].

where n is the number of battery cells and $m = n - 1$ is the number of links. The ring shunting topology has the topology matrix

$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ -1 & 1 & & \vdots & 0 \\ 0 & -1 & \ddots & 0 & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{8.40}$$

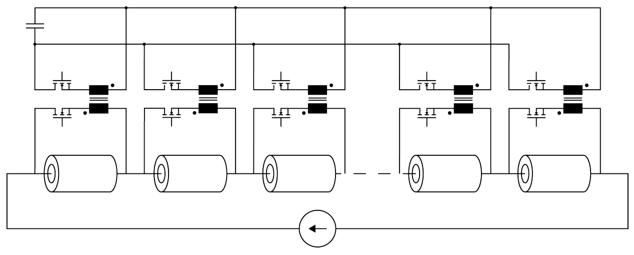
where n is the number of battery cells and $m = n$ is the number of links.

The input u is limited by peak current constraint of the links $|u_i| \leq 1$ for $i = 1, 2, \ldots, m$. Therefore the input constraint set is

$$\mathcal{U} = \left\{ u \in \mathbb{R}^m : -1 \leq u_i \leq 1 \right\} = [-1, 1]^m. \tag{8.41}$$

Storage Element Topologies

The storage element topologies are shown in Figure 8.3. In these topologies a passive storage element (capacitor or inductor) provides temporary storage for moving charge between cells. In Figure 8.3a the temporary storage element is a capacitor and the links a realized with flyback converters. In Figure 8.3b the temporary storage element is an inductor and we use the multiple-winding inductor converter. In both topologies the charge level of the storage


(a) Inductive

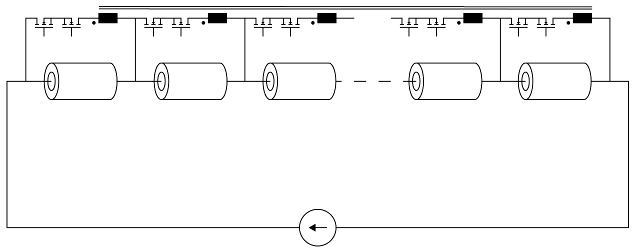

(b) Capacitive

Figure 8.3: Storage element topologies. A passive storage element is used to move charge between cells [70].

element remains constant. Examples are the single-switched capacitor topology and the multiple-winding inductor topology [20, 32, 66, 61].

The topology matrix $T = I$ for both the inductive and capacitive storage element topologies is given by

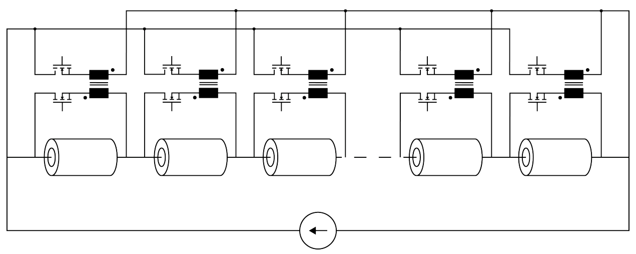
$$T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{8.42}$$

where n is the number of cells and $m = n$ is the number of links.

The input constraint sets are different for the capacitive and inductive storage element topologies. In both variants, the input u is limited by peak current constraint of the links $|u_i| \leq 1$ for $i = 1, 2, ..., m$. Since the storage element should neither be charged nor discharged over time, we have the addition constraint $\sum_{i=1}^{n} u_i = 0$. Therefore the input constraint set of the capacitive storage element topology is

$$\mathcal{U} = \Big\{ u \in \mathbb{R}^m : -1 \leq u_i \leq 1 \text{ and } \textstyle\sum_{i=1}^{m} u_i = 0 \Big\}. \tag{8.43}$$

In the inductive topology (Figure 8.3a) only one link can be active at each time instant. This constraint can be satisfied by ensuring that the total control action is less than one at each time instant $\|u\|_1 \leq 1$. Therefore the input constraint set of the inductive storage


(a) Individual

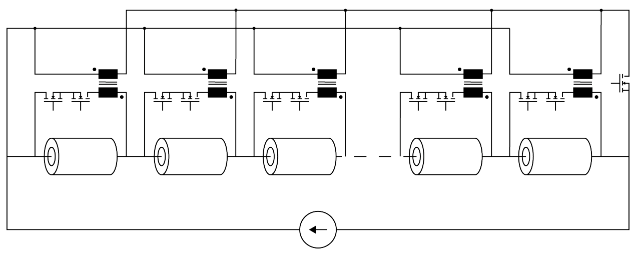

(b) Common

Figure 8.4: Cell-to-Stack topologies. Charge is removed from a cell and distributed equally among the stack or removed equally from the stack and added to a cell [70].

element topology is

$$\mathcal{U} = \left\{ u \in \mathbb{R}^m : -1 \le u_i \le 1,\ \textstyle\sum_{i=1}^{m} u_i = 0 \text{ and } \|u\|_1 \le 1 \right\}. \tag{8.44}$$

Cell-to-Stack Topologies

The cell-to-stack topologies are shown in Figure 8.4. In these topologies, charge is removed from one cell and distributed equally among all the cells in the stack. Likewise charge can be drawn equally from all the cells and added to a single cell. We consider two variants of the cell-to-stack topology; the common cell-to-stack topology shown in Figure 8.4b, which uses a converter with parallel mutual inductors, and the individual cell-to-stack topology shown in Figure 8.4a, which uses flyback converters. In the latter each cell has an individual link to the stack that can be operated independently from the other cells. This requires more high voltage switches but allows simultaneous movement of charge to and from multiple cells. Examples of the cell-to-stack topologies are the single and multiple transformer topologies from [20, 32, 66, 61].

The topology matrix $T = I - \frac{1}{n} 1 1^T$ for the cell-to-stack topologies is given by

$$T = \begin{bmatrix} 1 - \frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} \\ -\frac{1}{n} & 1 - \frac{1}{n} & & \vdots \\ \vdots & & \ddots & -\frac{1}{n} \\ -\frac{1}{n} & \cdots & -\frac{1}{n} & 1 - \frac{1}{n} \end{bmatrix} \in \mathbb{R}^{n \times m} \tag{8.45}$$

where n is the number of cells and $m = n$ is the number of links. We use the sign convention $u_i > 0$ to indicate charge flow from the stack to cell i and $u_j < 0$ to indicate charge flowing from cell j to the stack.

The input constraint sets are different for the individual and common cell-to-stack topologies. In both the input u is limited by peak current constraint of the links $|u_i| \leq 1$ for $i = 1, 2, ..., m$. Therefore the input constraint set of the individual cell-to-stack topology is

$$\mathcal{U} = \Big\{ u \in \mathbb{R}^n : -1 \leq u_i \leq 1 \Big\}. \tag{8.46}$$

In the common cell-to-stack topology only one link can be active at each time instant $\|u\|_1 \leq 1$. Therefore the input constraint set of the common cell-to-stack topology is

$$\mathcal{U} = \Big\{ u \in \mathbb{R}^n : -1 \leq u_i \leq 1 \text{ and } \|u\|_1 \leq 1 \Big\}. \tag{8.47}$$

The literature [20] describes unidirectional variants of the cell-to-stack topology. However we do not consider these variants since they cannot balance an arbitrary initial imbalance.

8.4 Simulation Results

In this section we demonstrate the controller in Algorithm 16 for each of the battery balancing hardware designs presented in Section 8.3. For this example the battery pack contains $n = 8$ cells. The charge capacity of the battery cells is $Q_x = 9.3$ amp-hours and the current capacity of the links is $Q_u = 5$ amps. The sample-time is $\Delta t = 60$ seconds. The maximum voltage for the battery cells is 3.9 volts. We assume the link efficiencies are 0% for the dissipative links and 95% for the non-dissipative links. Thus the power dissipation rates on the links are $\eta = 18.5$ watts for the dissipative topologies and $\eta = 0.975$ watts for the non-dissipative topologies.

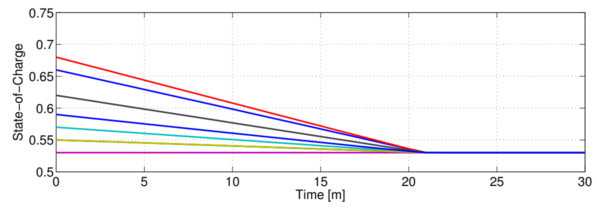
Our simulation model include ohmic losses on the current links and uncertainty about the charge capacities of the cells.

Dissipative Simulation Results

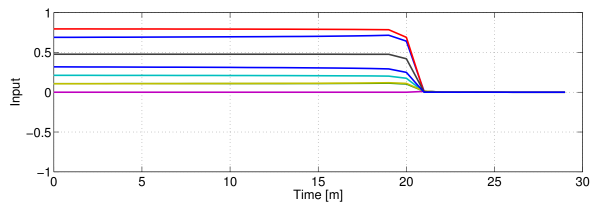
The simulation results for the dissipative battery balancing system are shown in Figure 8.5. Figure 8.5a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in closed-loop with the controller in Algorithm 16. Figure 8.5b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

Shunting Simulation Results

The simulation results for the ring shunting battery balancing system are shown in Figure 8.6. Figure 8.6a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in

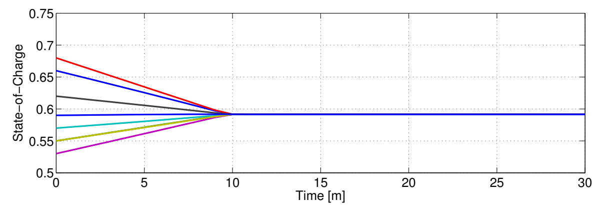


(a) State

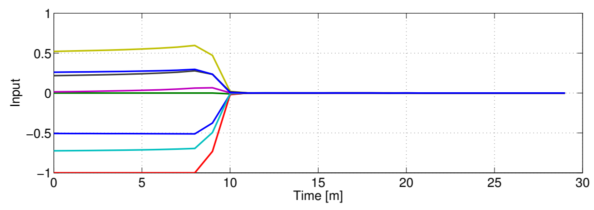


(b) Input

Figure 8.5: Dissipative topology simulation results. (a) Closed-loop trajectories $x(t)$ for the dissipative battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.

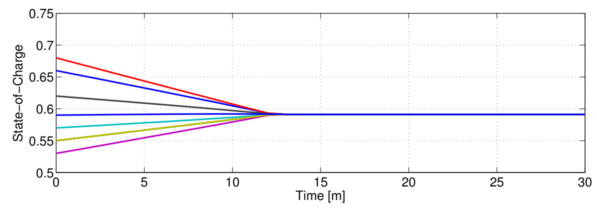


(a) State

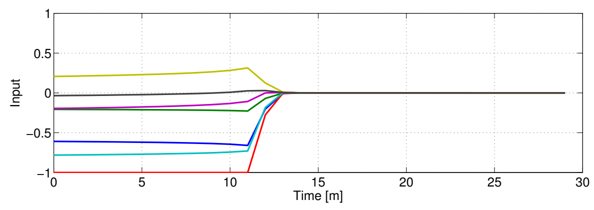


(b) Input

Figure 8.6: Ring shunting topology simulation results. (a) Closed-loop trajectories $x(t)$ for the ring shunting battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.



(a) State

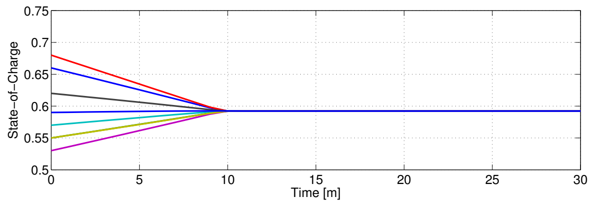


(b) Input

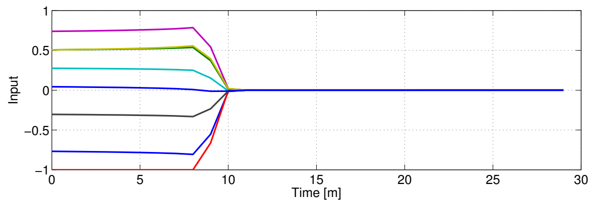
Figure 8.7: Line shunting topology simulation results. (a) Closed-loop trajectories $x(t)$ for the line shunting battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.

closed-loop with the controller in Algorithm 16. Figure 8.6b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

The simulation results for the line shunting battery balancing system are shown in Figure 8.7. Figure 8.7a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in closed-loop with the controller in Algorithm 16. Figure 8.7b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

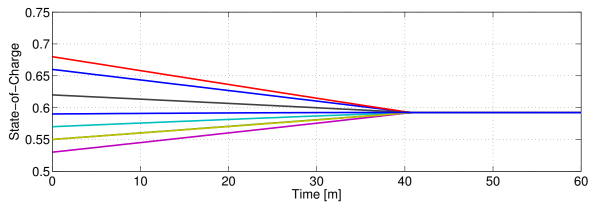


(a) State

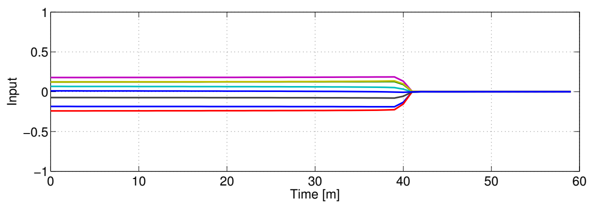


(b) Input

Figure 8.8: Capacitive storage-element topology simulation results: (a) Closed-loop trajectories $x(t)$ for the capacitive storage-element battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.



(a) State



(b) Input

Figure 8.9: Inductive storage-element topology simulation results. (a) Closed-loop trajectories $x(t)$ for the inductive storage-element battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.

Storage Element Simulation Results

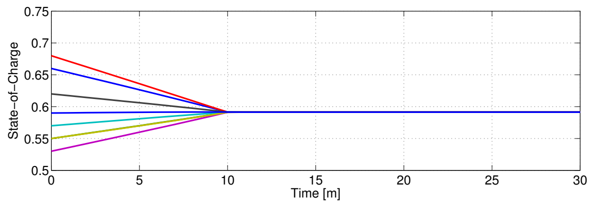
The simulation results for the capacitive storage element battery balancing system are shown in Figure 8.8. Figure 8.8a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in closed-loop with the controller in Algorithm 16. Figure 8.8b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

The simulation results for the inductor storage element battery balancing system are shown in Figure 8.9. Figure 8.7a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in closed-loop with the controller in Algorithm 16. Figure 8.9b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

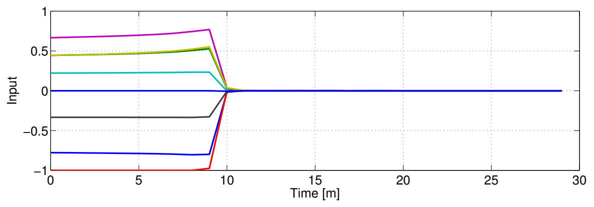
Cell to Stack Simulation Results

The simulation results for the individual cell-to-stack battery balancing system are shown in Figure 8.10. Figure 8.10a shows the state-of-charge trajectories $x(t)$ for the battery system (8.2) in closed-loop with the controller in Algorithm 16. Figure 8.10b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

The simulation results for the common cell-to-stack battery balancing system are shown in Figure 8.11. Figure 8.11a shows the state-of-charge trajectories $x(t)$ for the battery system

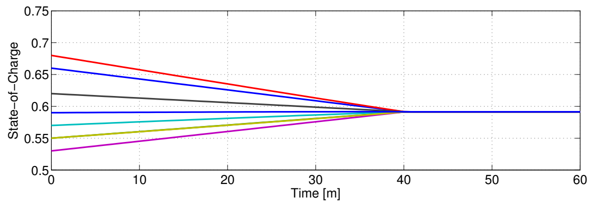


(a) State

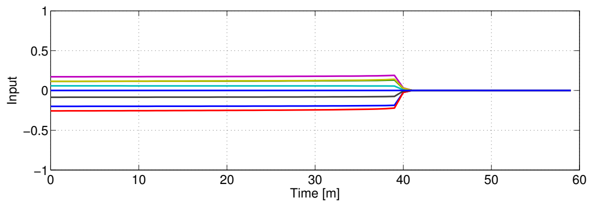


(b) Input

Figure 8.10: Individual cell-to-stack topology simulation results. (a) Closed-loop trajectories $x(t)$ for the individual cell-to-stack battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.



(a) State



(b) Input

Figure 8.11: Common cell-to-stack topology simulation results. (a) Closed-loop trajectories $x(t)$ for the common cell-to-stack battery balancing system. (b) Normalized balancing current $u(t)$ for the battery balancing controller.

(8.2) in closed-loop with the controller in Algorithm 16. Figure 8.11b shows the normalized balancing current $u(t)$ produced by the controller in Algorithm 16.

Chapter 9

Symmetric Battery Balancing Controllers

In this Chapter we apply the symmetric explicit model predictive control theory from Chapter 7 to the battery balancing problem. For each of the battery balancing systems described in Section 8.3 we study the memory savings offered by the orbit and fundamental domain controllers. We assume the batteries have identical charge capacities $Q_x = q_x I_n$ and the links have identical current capacities $Q_u = q_u I_m$. Therefore the symmetry groups of the battery balancing system depends on the hardware topology matrix $T \in \mathbb{R}^{n \times m}$ and constraint set $\mathcal{U} \subseteq \mathbb{R}^m$

In the following subsections we will derive the symmetry group $\mathrm{Aut}(MPC)$ for the model predictive control problem (8.20) for each of the battery balancing hardware designs. For a fixed battery pack size n, we can use the techniques described in Chapter 5 to identify $\mathrm{Aut}(MPC)$. However in this chapter we provide propositions that describe the symmetry group $\mathrm{Aut}(MPC)$ for each hardware configuration with an arbitrary pack size n. Each of these propositions is based on the following lemma which limits the class of matrices Θ and Ω we need to consider.

Lemma 4. *Let* $(\Theta, \Omega) \in \mathrm{Aut}(MPC)$ *be symmetries of the one-step optimization problem* (8.20)*. Then* $\Theta = \Pi_x$ *is a permutation matrix and* $\Omega = \Sigma_u$ *is a signed permutation matrix.*

Proof. Since the cost function $q(x, u)$, the parameter set $\mathcal{X}$, and the terminal set $\bar{\mathcal{X}}$ of (8.20) do not change between the different hardware designs, we use these to restrict the class of matrices to which Θ and Ω can belong.

The stage cost $q(x, u) = \Phi_{\mathcal{U}}(v) + \rho\|\eta v\|_1$ contains a 1-norm terms. The symmetry group of the 1-norm is the hyperoctahedral group $\mathfrak{B}_m$ which contains all signed permutation matrices $\Sigma \in \mathbb{R}^{m \times m}$. Suppose $\Omega \notin \mathfrak{B}_m$. Then Ω is not a symmetry of the cost function since

$$q(\Theta x, \Omega v) = \Phi_{\mathcal{U}}(\Omega v) + \rho\|\eta \Omega v\|_1 \neq \Phi_{\mathcal{U}}(v) + \rho\|\eta v\|_1 = q(x, v) \tag{9.1}$$

where $\Phi_{\mathcal{U}}(\Omega v) = \Phi_{\mathcal{U}}(v)$ since $\Omega\mathcal{U} = \mathcal{U}$ but $\|v\|_1 \neq \|\Omega v\|_1$ since $\Omega \in \mathfrak{B}_m$.

Next we show the state-space symmetries Θ must be permutation matrices $\Theta = \Pi$. The state constraint set $\mathcal{X} = [0,1]^n$ is intersection of a hypercube with the positive orthant. The symmetry group of a hypercube is the hyperoctahedral group $\mathfrak{B}_n$. Thus symmetries of $\mathcal{X}$ can only be signed permutation matrices $\Theta = \Sigma$. However since $\mathcal{X}$ is a subset of the positive orthant the matrix $\Theta = \Pi$ must be a permutation matrix.

Terminal set $\bar{\mathcal{X}}$ is the intersection of $\mathcal{X} = [0,1]^n$ with the range-space of the ones-vector $\mathbf{1} \in \mathbb{R}^n$. The permutation matrices $\Theta = \Pi$ preserve the ones-vector $\mathbf{1} \in \mathbb{R}^n$ and therefore its range-space. Thus $\mathrm{Aut}(\mathcal{X}) = \mathrm{Aut}(\bar{\mathcal{X}}) = \mathfrak{S}_n$ is the set of all permutation matrices $\Theta = \Pi$. □

9.1 Dissipative Battery Balancing System

The symmetry group $\mathrm{Aut}(MPC)$ of the dissipative battery balancing system is the symmetric group $\mathrm{Aut}(MPC) = \mathfrak{S}_n$. The group is generated by $\mathcal{S} = \{(\Theta_c, \Omega_c), (\Theta_t, \Omega_t)\}$ where $\Theta_c = \Omega_c = \Pi_c$ is the cyclic matrix Π_c corresponding to the cyclic permutation

$$\pi_c = \begin{pmatrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 3 & 4 & \dots & n & 1 \end{pmatrix} \tag{9.2}$$

and $\Theta_t = \Omega_t = \Pi_t$ is the transposition matrix corresponding to the transposition permutation

$$\pi_t = \begin{pmatrix} 1 & 2 & 3 & \dots & n-1 & n \\ 2 & 1 & 3 & \dots & n-1 & n \end{pmatrix}. \tag{9.3}$$

Proposition 25. *Let $T \in \mathbb{R}^{m \times n}$ be given by* (8.37) *and $\mathcal{U} \subseteq \mathbb{R}^m$ be given by* (8.38). *Then the symmetry group* $\mathrm{Aut}(MPC)$ *of the model predictive control problem* (8.20) *is the symmetric group $\mathfrak{S}_n$ which contains all pairs of permutation matrices* $(\Theta, \Omega) = (\Pi, \Pi)$.

Proof. First we show that if (Θ, Ω) are not permutation matrices then they are not symmetries $(\Theta, \Omega) \notin \mathrm{Aut}(MPC)$. By Lemma 4 the matrix Θ must be a permutation matrix. Since $B = -\frac{q_u}{q_x} I$ the symmetry condition $\Theta B = B\Omega$ implies $\Omega = \Theta$ is a permutation matrix.

Next we prove that every pair of symmetry matrices $(\Theta, \Omega) = (\Pi, \Pi)$ is a symmetry $(\Pi, \Pi) \in \mathrm{Aut}(MPC)$. By Lemma 4 we only need to verify that (Θ, Ω) satisfy $\Theta B = B\Omega$ and $\Omega\mathcal{U} = \mathcal{U}$. We have $\Theta B = B\Omega$ since $B = -\frac{q_u}{q_x} I$ and $\Theta = \Omega = \Pi$. We have $\Omega\mathcal{U} = \mathcal{U}$ since

$$\underbrace{\begin{bmatrix} -I_m \\ \mathbf{1}^T \end{bmatrix}}_{H_u} \Pi = \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} -I_m \\ \mathbf{1}^T \end{bmatrix}}_{H_u} \quad \text{and} \quad \underbrace{\begin{bmatrix} 0 \\ u_{\max} \end{bmatrix}}_{K_u} = \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} 0 \\ u_{\max} \end{bmatrix}}_{K_u} \tag{9.4}$$

where $\mathcal{U} = \{u : H_u u \leq K_u\}$. Thus $\Omega = \Pi$ simply permutes the constraints that define $\mathcal{U}$. □

Intuitively this symmetry group says that permuting the battery cells does not change the hardware topology shown in Figure 8.1. Likewise permuting the battery cells does not change the input constraint set $\mathcal{U} \subseteq \mathbb{R}^m$.

In Chapter 6.2 we showed that a fundamental domain $\hat{\mathcal{X}}$ for the set $\mathcal{X} = [0,1]^n$ with respect to the group $\text{Aut}(MPC) = \mathfrak{S}_n$ is the set of sorted states

$$\hat{\mathcal{X}} = \big\{x \in [0,1]^n : x_1 \geq x_2 \geq \cdots \geq x_n\big\}. \tag{9.5}$$

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the dissipative topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.1a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.1b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.1.

For battery packs with n cells, the full explicit controller $\kappa(x)$ has n pieces. On the other hand, the orbit controller $\kappa^o(x)$ and fundamental domain controller $\hat{\kappa}(x)$ have one piece regardless of pack-size n. The orbit controller $\kappa^o(x)$ and fundamental domain controller $\hat{\kappa}(x)$ use the single control-law

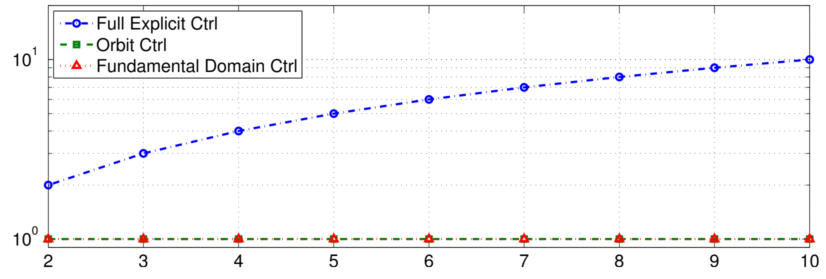
$$\tau^\star = \tfrac{q_x}{q_u u_{\max}}\big(\textstyle\sum_{i=1}^{n-1} x_i - x_n\big) \tag{9.6a}$$

$$v_i^\star = \tfrac{q_x}{q_u}(x_i - x_n) \tag{9.6b}$$

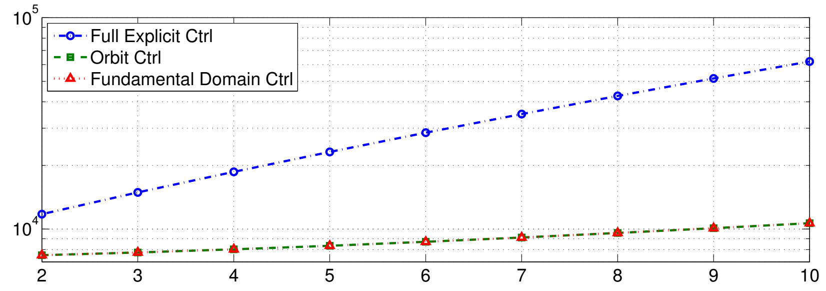
where $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$. Recall that in the fundamental domain, battery cell n has the lowest state-of-charge x_n. Therefore this control-law says that the amount of charge we should discharge $v_i^\star$ from cells $i = 1, \dots, n-1$ is proportional to the difference between the cells state-of-charge x_i and the state-of-charge x_n of the lowest cell n. The reason that the full explicit controller has n pieces is that without restricting ourselves to the fundamental domain any of the cells $1, \dots, n$ can have the lowest state-of-charge.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
2	2	11.7	1	7.5	1	7.5
3	3	14.9	1	7.7	1	7.7
4	4	18.6	1	8.0	1	8.0
5	5	23.2	1	8.4	1	8.4
6	6	28.6	1	8.7	1	8.7
7	7	35.0	1	9.1	1	9.1
8	8	42.6	1	10.1	1	10.1
9	9	51.6	1	10.6	1	10.6
10	10	62.0	1	11.2	1	11.2

Table 9.1: Dissipative topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory (kilobytes)

Figure 9.1: Dissipative topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

9.2 Shunting Battery Balancing Systems

The symmetry group $\text{Aut}(MPC)$ of the ring-shunting battery balancing system is the dihedral-n group $\text{Aut}(MPC) = \mathcal{D}_n$. The group is generated by $\mathcal{S} = \{(\Theta_c, \Omega_c), (\Theta_r, \Omega_r)\}$ where $\Theta_c = \Omega_c = \Pi_c$ is the cyclic matrix Π_c corresponding to the cyclic permutation (9.2) and $\Theta_r = -\Omega_r = \Pi_r$ is the reflection matrix corresponding to the reflection permutation

$$\pi_r = \begin{pmatrix} 1 & 2 & \dots & n-1 & n \\ n & n-1 & \dots & 2 & 1 \end{pmatrix}. \tag{9.7}$$

Proposition 26. *Let $T \in \mathbb{R}^{m\times n}$ be given by* (8.40) *and $\mathcal{U} \subseteq \mathbb{R}^m$ be given by* (8.41). *Then the symmetry group* $\text{Aut}(MPC)$ *of the model predictive control problem* (8.20) *is the dihedral-n group $\mathcal{D}_n$ generated by the matrix pairs* (Θ_c, Ω_c) *and* (Θ_r, Ω_r).

Proof. The matrix T is the incident matrix of a cycle graph. Therefore its symmetry group is the dihedral-n group generated by the matrix pairs $(\Theta_c, \Omega_c) = (\Pi_c, \Pi_c)$ and $(\Theta_r, \Omega_r) = (\Pi_r, -\Pi_r)$. The matrices Ω_c and Ω_r are also symmetries of the hypercube $\mathcal{U}$. □

Intuitively this symmetry group says that we can rotate or reflect the battery arrangement shown in Figure 8.2a with out changing the hardware topology T or constraint set $\mathcal{U}$.

Using Algorithm 10 we can construct a fundamental domain $\hat{\mathcal{X}}$ for the set $\mathcal{X} = [0,1]^n$ with respect the group $\mathrm{Aut}(MPC) = \mathcal{D}_n$. Regardless of the pack-size n we can use the base $\mathcal{B} = \{e_1, e_2\}$ where $e_1, \dots, e_n$ are the standard basis vectors in $\mathbb{R}^n$. A strong generating set $\mathcal{S} = \{S_1, S_2\}$ for this base $\mathcal{B}$ are the generators $S_1 = \Pi_c$ and $S_2 = \Pi_c^{-1}\Pi_r$. Using Algorithm 10 we obtain the fundamental domain

$$\hat{\mathcal{X}} = \big\{x \in [0,1]^n : x_1 \geq x_2, \dots, x_n \text{ and } x_2 \geq x_n\big\}. \tag{9.8}$$

This fundamental domain says that the first battery cell $i = 1$ has the highest state-of-charge $x_1 \geq x_2, \dots, x_n$ and that the state-of-charge x_2 of the second battery $i = 2$ is greater $x_2 \geq x_n$ than the state-of-charge x_n of the last battery $i = n$.

The symmetry group $\mathrm{Aut}(MPC)$ of the line-shunting battery balancing system is the reflection group $\mathrm{Aut}(MPC) = \mathfrak{S}_2$. This group $\mathrm{Aut}(MPC) = \mathfrak{S}_2$ has only one non-trivial element: the reflection matrix $\Theta = \Omega = \Pi_r$ corresponding to the reflection permutation (9.7).

Proposition 27. *Let $T \in \mathbb{R}^{m \times n}$ be given by* (8.40) *and $\mathcal{U} \subseteq \mathbb{R}^m$ be given by* (8.41)*. Then the symmetry group* $\mathrm{Aut}(MPC)$ *of the model predictive control problem* (8.20) *is the reflection group $\mathfrak{S}_2$ generated by the matrix pair* (Θ_r, Ω_r).

Proof. The matrix T is the incident matrix of a path graph. Therefore its symmetry group is generated by the single reflection $(\Theta_r, \Omega_r) = (\Pi_r, \Pi_r)$. The matrix Ω_r is also a symmetry of the hypercube $\mathcal{U}$. □

This group says that the only operations that do not change the hardware topology shown in Figure 8.2a are the identity and the reflection of the cells (9.7).

Using Algorithm 10 we can construct a fundamental domain $\hat{\mathcal{X}}$ for the set $\mathcal{X} = [0,1]^n$ with respect the group $\mathrm{Aut}(MPC) = \mathfrak{S}_2$. Regardless of the pack-szie n we can use the singleton base $\mathcal{B} = \{e_1\}$ where $e_1, \dots, e_n$ are the standard basis vectors in $\mathbb{R}^n$. By default $\mathcal{S} = \{\Pi_r\}$ is a strong generating set for this base. Using Algorithm 10 we obtain the fundamental domain

$$\hat{\mathcal{X}} = \big\{x \in [0,1]^n : x_1 \geq x_n\big\}. \tag{9.9}$$

This fundamental domain says that the state-of-charge x_1 of the first cell is higher than the state-of-charge x_n of the last cell.

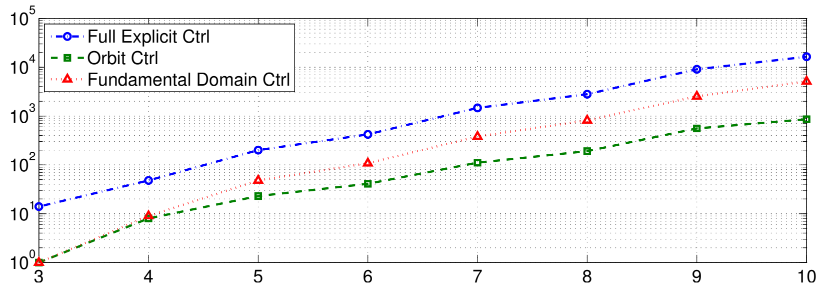
Ring Shunting Controllers

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the ring shunting topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.2a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.2b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.2.

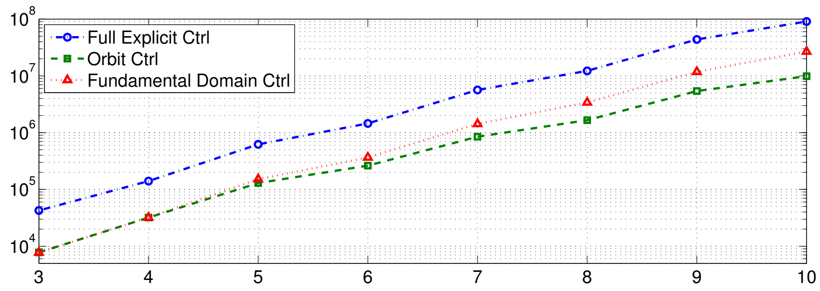
This hardware topology has a relatively small symmetry group $|\mathcal{D}_n| = 2n$. Therefore memory savings from using the orbit controllers $\kappa^o(x)$ and the fundamental domain $\hat{\kappa}(x)$ is minor. In this case, the orbit controller $\kappa^o(x)$ asymptotically reduces the memory by a factor of $2n$. However the number of controller pieces $|\mathcal{I}/\mathcal{G}|$ still grows exponentially with the number of cells n. The fundamental domain controller does not perform as well, in terms of complexity reduction, as the orbit controller in this case as shown in Figure 9.2a.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
3	14	42.6	1	7.7	1	7.7
4	48	140.2	8	22.4	9	31.9
5	200	621.9	23	66.8	48	151.9
6	420	1456.9	41	132.4	108	365.8
7	1470	5656.5	111	393.9	380	1426.4
8	2800	12254.2	191	780.3	816	3405.2
9	9072	43930.0	557	2502.7	2529	11810.0
10	16380	90904.0	859	4493.1	5114	26902.6

Table 9.2: Ring topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory (kilobytes)

Figure 9.2: Ring topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

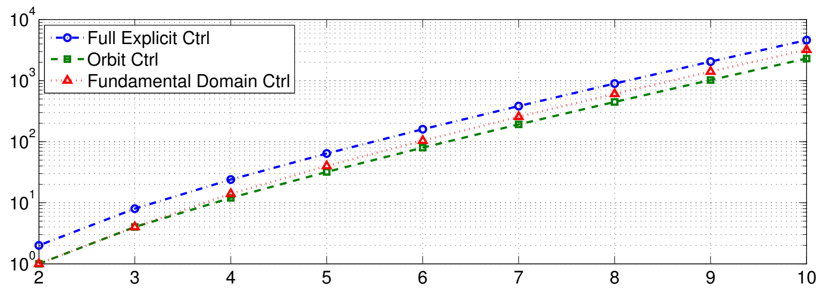
Line Shunting Controllers

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the line shunting topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.3a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.3b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.3.

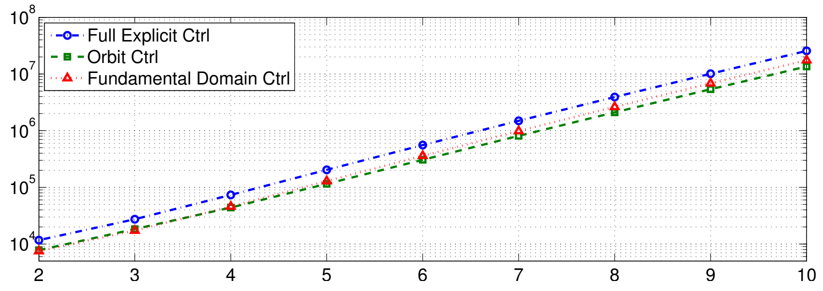
This hardware topology has the smallest symmetry group $|\mathfrak{S}_n| = 2$. Therefore memory savings from using the orbit controllers $\kappa^o(x)$ and the fundamental domain $\hat{\kappa}(x)$ is minimal. On average the orbit controller reduced memory by a factor of approximately 1.9 and the fundamental domain controller reduced the memory by a factor of approximately 1.5. However the number of pieces and therefore memory still grows exponentially with the number of cells n in the pack.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\|\mathcal{I}\|$	Memory	Pieces $\|\mathcal{I}/\mathcal{G}\|$	Memory	Pieces $\|\hat{\mathcal{I}}\|$	Memory
2	2	11.7	1	7.5	1	7.5
3	8	27.3	4	17.2	4	17.2
4	24	73.6	12	43.9	14	45.8
5	64	203.7	32	115.0	40	128.9
6	160	557.1	80	306.5	104	360.5
7	384	1492.8	192	809.4	256	984.9
8	896	3918.7	448	2104.1	608	2625.9
9	2048	10099.1	1024	5382.6	1408	6851.4
10	4608	25602.6	2304	13562.9	3200	17542.6

Table 9.3: Line topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory (kilobytes)

Figure 9.3: Line topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

9.3 Storage Element Battery Balancing Systems

The symmetry group $\mathrm{Aut}(MPC)$ of the storage element battery balancing systems is the symmetric group $\mathrm{Aut}(MPC) = \mathfrak{S}_n$. The group is generated by $\mathcal{S} = \{(\Theta_c, \Omega_c), (\Theta_t, \Omega_t)\}$ where $\Theta_c = \Omega_c = \Pi_c$ is the cyclic matrix Π_c corresponding to the cyclic permutation (9.2) and $\Theta_t = \Omega_t = \Pi_t$ is the transposition matrix corresponding to the transposition permutation (9.3).

Proposition 28. *Let $T \in \mathbb{R}^{m\times n}$ be given by* (8.42) *and $\mathcal{U} \subseteq \mathbb{R}^m$ be given by* (8.43) *or* (8.44). *Then the symmetry group* $\mathrm{Aut}(MPC)$ *of the model predictive control problem* (8.20) *is the set of all pair of permutation matrices* $(\Theta, \Omega) = (\Pi, \Pi)$.

Proof. First we show that if (Θ, Ω) are not permutation matrices then they are not symmetries $(\Theta, \Omega) \notin \mathrm{Aut}(MPC)$. By Lemma 4 the matrix Θ must be a permutation matrix. Since $B = \frac{q_u}{q_x} I$ the symmetry condition $\Theta B = B\Omega$ implies $\Omega = \Theta$ is a permutation matrix.

Next we prove that every pair of symmetry matrices $(\Theta, \Omega) = (\Pi, \Pi)$ is a symmetry $(\Pi, \Pi) \in \mathrm{Aut}(MPC)$. By Lemma 4 we only need to verify that (Θ, Ω) satisfy $\Theta B = B\Omega$ and $\Omega\mathcal{U} = \mathcal{U}$. We have $\Theta B = B\Omega$ since $B = -\frac{q_u}{q_x} I$ and $\Theta = \Omega = \Pi$.

The constraint set $\mathcal{U}$ given by (8.43) is the intersection of a hypercube with the null-space of the ones-vector $\mathbf{1}$. The hypercube is invariant under permutations. Likewise the ones-vector $\mathbf{1}$, and therefore its null-space, are invariant under permutations $\Pi\mathbf{1} = \mathbf{1}$.

The constraint set $\mathcal{U}$ given by (8.44) is the intersection of a cross-polytope with the null-space of the ones-vector $\mathbf{1}$. These sets are invariant under permutations. □

Intuitively this symmetry group says that permuting the battery cells does not change the hardware topology shown in Figure 8.3. Likewise permuting the battery cells does not change the input constraint sets $\mathcal{U} \subseteq \mathbb{R}^m$ for the capacitive and inductive cases.

In Section 6.2 we showed that a fundamental domain $\hat{\mathcal{X}}$ for the set $\mathcal{X} = [0, 1]^n$ with respect to the group $\mathrm{Aut}(MPC) = \mathfrak{S}_n$ is the set of sorted states (9.5).

Capacitive Storage Element Controllers

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the ring shunting topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.4a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.4b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.4.

In this numerical study, the number of pieces $|\mathcal{I}|$ in the full explicit controller $\kappa(x)$ grew exponentially with the number of battery cells n in the pack. However the orbit controller $\kappa^o(x)$ had $|\mathcal{I}/\mathcal{G}| = 2$ pieces for battery packs with more than $n = 3$ battery cells. The control-laws are

$$\tau^\star = \frac{q_x}{q_u}\begin{cases}\begin{bmatrix}(1-\frac{1}{n}) & \frac{1}{n} & \dots \frac{1}{n}\end{bmatrix} x & \text{for } x \in \mathcal{R}_1 \\ \begin{bmatrix}\frac{1}{n} & \dots \frac{1}{n} & (1-\frac{1}{n})\end{bmatrix} x & \text{for } x \in \mathcal{R}_2\end{cases} \tag{9.10a}$$

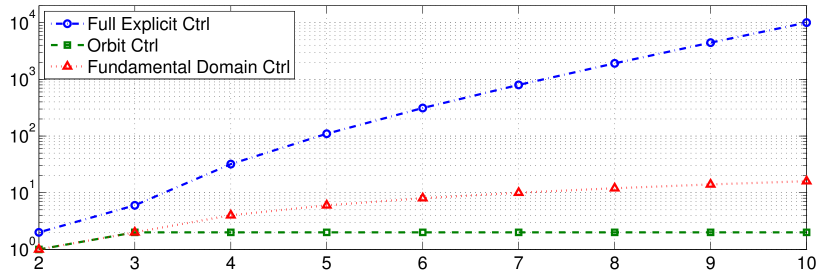
$$v_i^\star = -\frac{q_x}{q_u}\Big(x_i - \frac{1}{n}\sum\nolimits_{i=1}^{n} x_i\Big) \tag{9.10b}$$

where $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$ and the regions $\mathcal{R}_1$ and $\mathcal{R}_2$ polytopes. This control-law says that the optimal total charge redistribution $v_i^\star = -\frac{q_x}{q_u}(x_i - \frac{1}{n}\sum_{i=1}^n x_i)$ is always proportional to the deviation of the state-of-charge x_i of cell i from the average state-of-charge $\frac{1}{n}\sum_{i=1}^n x_i$. The difference between the two regions is the amount of time $\tau^\star$ needed to implement the charge redistribution $v^\star$. In the first case the time $\tau^\star$ depends on the difference between the highest state-of-charge x_1 and the average state of charge $\frac{1}{n}\sum_{i=1}^n x_i$. In the second case the time $\tau^\star$ depends on the difference between the lowest state-of-charge x_n and the average state of charge $\frac{1}{n}\sum_{i=1}^n x_i$.

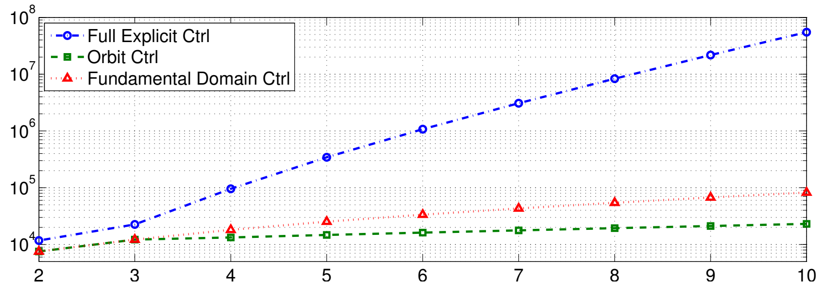
We conjecture that this result holds for arbitrary battery pack sizes n.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
2	2	11.8	1	7.5	1	7.5
3	6	22.6	2	12.2	2	12.2
4	32	95.7	2	13.3	4	18.1
5	110	344.4	2	14.7	6	25.3
6	312	1075.8	2	16.2	8	33.6
7	798	3079.1	2	17.8	10	43.2
8	1920	8331.4	2	19.4	12	54.5
9	4446	21706.5	2	21.2	14	67.4
10	10040	55049.7	2	23.1	16	82.4

Table 9.4: Capacitive topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory (kilobytes)

Figure 9.4: Capacitive topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

Inductive Storage Element Controllers

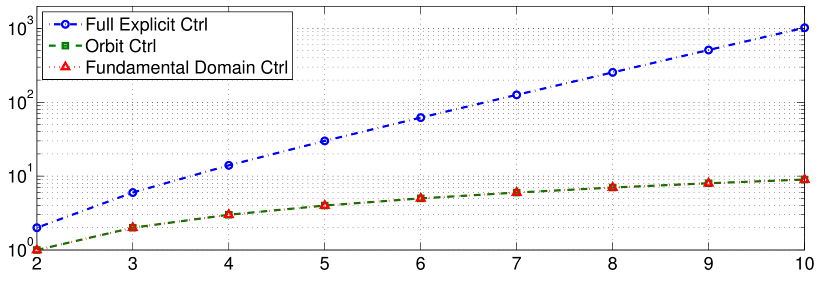
We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the ring shunting topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.5a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.5b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.5.

In this numerical study, the number of pieces $|\mathcal{I}|$ in the full explicit controller $\kappa(x)$ grew exponentially with the number of battery cells n in the pack. However the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$ grows linearly with the number of batteries n in the pack. For each case the optimal total charge redistribution $v_i^\star = -\frac{q_x}{q_u}(x_i - \frac{1}{n}\sum_{i=1}^n x_i)$ is always proportional to the deviation of the state-of-charge x_i of cell i from the average state-of-charge $\frac{1}{n}\sum_{i=1}^n x_i$. The difference between the $n-1$ control-laws is the amount of time $\tau^\star$ needed to implement the charge redistribution $v^\star$. There are $n-1$ different weighted averages of the state-of-charge of the cells used to determine the time $\tau^\star$. The regions $\mathcal{R}_1,\ldots,\mathcal{R}_{n-1}$ determine which of these weighted averages to use.

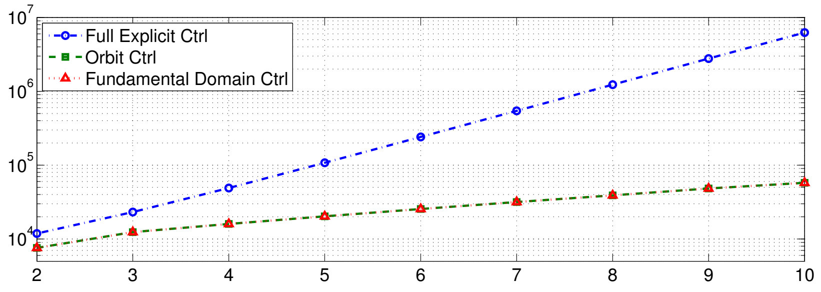
We conjecture that this result holds for arbitrary battery pack sizes n.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
2	2	11.9	1	7.6	1	7.6
3	6	23.2	2	12.4	2	12.4
4	14	49.0	3	16.0	3	16.0
5	30	107.8	4	20.3	4	20.3
6	62	241.4	5	25.5	5	25.5
7	126	544.5	6	31.6	6	31.6
8	254	1230.0	7	39.0	7	39.0
9	510	2775.9	8	48.3	8	48.3
10	1022	6249.3	9	57.7	9	57.7

Table 9.5: Inductive topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory

Figure 9.5: Inductive topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

9.4 Cell-to-Stack Battery Balancing Systems

The symmetry group $\mathrm{Aut}(MPC)$ of the cell-to-stack battery balancing systems is the symmetric group $\mathrm{Aut}(MPC) = \mathfrak{S}_n$. The group is generated by $\mathcal{S} = \{(\Theta_c, \Omega_c), (\Theta_t, \Omega_t)\}$ where $\Theta_c = \Omega_c = \Pi_c$ is the cyclic matrix Π_c corresponding to the cyclic permutation (9.2) and $\Theta_t = \Omega_t = \Pi_t$ is the transposition matrix corresponding to the transposition permutation (9.3).

Proposition 29. *Let $T \in \mathbb{R}^{m\times n}$ be given by* (8.45) *and $\mathcal{U} \subseteq \mathbb{R}^m$ be given by* (8.46) *or* (8.47). *Then the symmetry group* $\mathrm{Aut}(MPC)$ *of the model predictive control problem* (8.20) *is the set of all pair of permutation matrices* $(\Theta, \Omega) = (\Pi, \Pi)$.

Proof. First we show that if (Θ, Ω) are not permutation matrices then they are not symmetries $(\Theta, \Omega) \notin \mathrm{Aut}(MPC)$. By Lemma 4 the matrix Θ must be a permutation matrix. Since $T = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ the symmetry condition $\Theta B = B\Omega$ implies $\Omega = \Theta + \alpha\mathbf{1}\mathbf{1}^T$ for any $\alpha \in \mathbb{R}$. However for $\alpha \neq 0$ the matrix Ω is not a symmetry of the stage cost $q(x,u) = \|v\|_{\mathcal{U}} + \rho\eta\|v\|_1$. Thus $\Omega = \Theta$ is a permutation matrix.

Next we prove that every pair of permutation matrices $(\Theta, \Omega) = (\Pi, \Pi)$ is a symmetry $(\Pi, \Pi) \in \mathrm{Aut}(MPC)$. By Lemma 4 we only need to verify that (Θ, Ω) satisfy $\Theta B = B\Omega$ and $\Omega\mathcal{U} = \mathcal{U}$. We have $\Theta B = B\Omega$ since $B = -\frac{q_u}{q_x}(I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ and $\Theta = \Omega = \Pi$.

The constraint set $\mathcal{U}$ given by (8.46) is a hypercube and the constraint set $\mathcal{U}$ given by (8.47) is a cross-polytope. These sets are both symmetric with respect to permutation matrices $\Omega = \Pi$. □

Intuitively this symmetry group says that permuting the battery cells does not change the hardware topology shown in Figure 8.3. Likewise permuting the battery cells does not change the input constraint sets $\mathcal{U} \subseteq \mathbb{R}^m$ for the individual or common cases.

In Section 6.2 we showed that a fundamental domain $\hat{\mathcal{X}}$ for the set $\mathcal{X} = [0,1]^n$ with respect to the group $\mathrm{Aut}(MPC) = \mathfrak{S}_n$ is the set of sorted states (9.5).

Individual Cell-to-Stack Controllers

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the individual cell-to-stack topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.6a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.6b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.6.

In this numerical study, the number of pieces $|\mathcal{I}|$ in the full explicit controller $\kappa(x)$ grew exponentially with the number of battery cells n in the pack. On the other hand, the number of pieces for the orbit controller $\kappa^o(x)$ and fundamental domain $\hat{\kappa}(x)$ remained constant for odd or even battery pack sizes.

For a battery pack with an even-number of cells n, the orbit controller has three pieces

$$\tau^\star = \frac{q_x}{q_u} \begin{cases} \frac{1}{2}\big(x_1 - x_n\big) & \text{for } x \in \mathcal{R}_1 \\ x_1 - x_{\frac{n}{2}} & \text{for } x \in \mathcal{R}_2 \\ x_{\frac{n}{2}+1} - x_n & \text{for } x \in \mathcal{R}_3 \end{cases} \tag{9.11a}$$

$$v_i^\star = \frac{q_x}{q_u} \begin{cases} -x_i + \frac{1}{2}\big(x_1 + x_n\big) & \text{for } x \in \mathcal{R}_1 \\ -x_i + x_{\frac{n}{2}} & \text{for } x \in \mathcal{R}_2 \\ -x_i + x_{\frac{n}{2}+1} & \text{for } x \in \mathcal{R}_3 \end{cases} \tag{9.11b}$$

where $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$.

In the first case $x \in \mathcal{R}_1$, this control-law says that the optimal total charge redistribution $v_i^\star$ for cell i is proportional to the deviation of the state-of-charge x_i from the average state-of-charge $\frac{1}{2}(x_1 + x_2)$ of the highest and lowest cells. The time $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$ needed to implement this charge redistribution $v^\star$ is proportional to the difference in state-of-charge between the highest x_1 and lowest x_n cells.

In the second case $x \in \mathcal{R}_2$, this control-law says that the optimal total charge redistribution $v_i^\star$ for cell i is proportional to the deviation of the state-of-charge x_i from the upper-median state-of-charge $x_{\frac{n}{2}}$. The time $\tau^\star$ needed to implement this charge redistribution $v^\star$ is proportional to the difference between the highest state-of-charge x_1 and upper-median state-of-charge $x_{\frac{n}{2}}$.

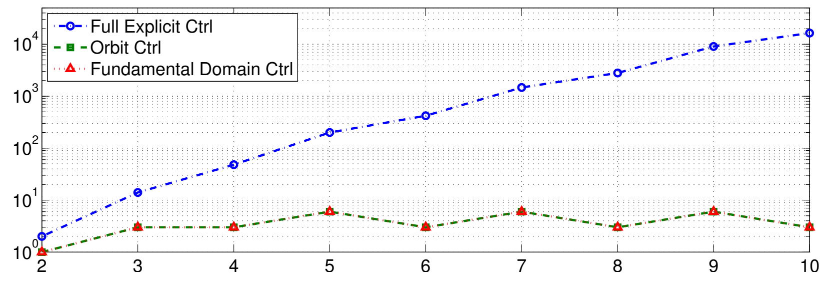
In the third case $x \in \mathcal{R}_3$, this control-law says that the optimal total charge redistribution $v_i^\star$ for cell i is proportional to the deviation of the state-of-charge x_i from the lower-median state-of-charge $x_{\frac{n}{2}+1}$. The time $\tau^\star$ needed to implement this charge redistribution $v^\star$ is proportional to the difference between lower-median state-of-charge $x_{\frac{n}{2}+1}$ and lowest state-of-charge x_n.

For a battery pack with an odd-number of cells $n > 3$, the orbit controller had six pieces. In this case the optimal total charge redistribution $v^\star = -\frac{q_x}{q_u}(x_i - x_m)$ is proportional to the deviation of the state-of-charge x_i of cells i and the state-of-charge x_m of the median cell $m = \lceil \frac{n}{2} \rceil$. The time $\tau^\star$ is a weighted average of the state-of-charge x that depends on which region $\mathcal{R}_1, \ldots, \mathcal{R}_6$ contains the state-of-charge x.

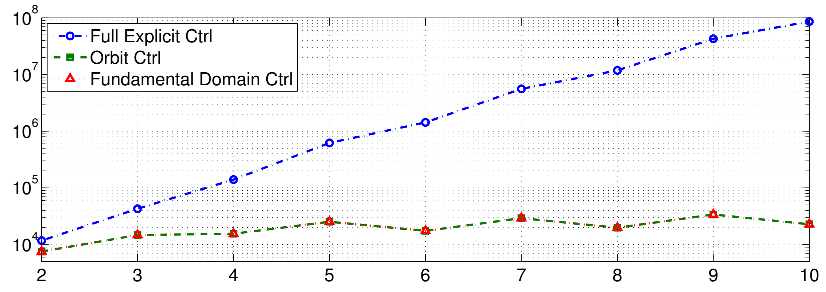
We conjecture that this result holds for arbitrary battery pack sizes n.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
2	2	11.8	1	7.5	1	7.5
3	14	42.8	3	14.7	3	14.7
4	48	140.3	3	15.5	3	15.5
5	200	621.0	6	25.3	6	25.3
6	420	1430.5	3	17.5	3	17.5
7	1470	5596.0	6	29.3	6	29.3
8	2800	11824.1	3	19.9	3	19.9
9	9072	42944.9	6	34.0	6	34.0
10	16380	86278.1	3	22.8	3	22.8

Table 9.6: Cell-to-stack individual topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory

Figure 9.6: Cell-to-stack individual topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

Common Cell-to-Stack Controllers

We generated a full explicit model predictive controller $\kappa(x)$, an orbit controller $\kappa^o(x)$, and a fundamental domain controller $\hat{\kappa}(x)$ for the common cell-to-stack topology for battery packs between $n = 2$ and $n = 10$ cells. Figure 9.7a shows the number of pieces $|\mathcal{I}|$ in the full controller $\kappa(x)$, the number of pieces $|\mathcal{I}/\mathcal{G}|$ in the orbit controller $\kappa^o(x)$, and the number of pieces $|\hat{\mathcal{I}}|$ in the fundamental domain controller $\hat{\kappa}(x)$. Figure 9.6b shows the memory requirements for the full controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$. This information is also summarized in Table 9.7.

In this numerical study, the number of pieces $|\mathcal{I}|$ in the full explicit controller $\kappa(x)$ grew exponentially with the number of battery cells n in the pack. On the other hand, the number of pieces for the orbit controller $\kappa^o(x)$ and fundamental domain $\hat{\kappa}(x)$ remained constant for odd or even battery pack sizes.

For a battery pack with an odd-number of cells n the orbit controller has one piece given by

$$\tau^\star = \tfrac{q_x}{q_u}\left(\textstyle\sum_{i=1}^{m-1} x_i - \sum_{i=m+1}^{n} x_i\right) \tag{9.12a}$$

$$v_i^\star = -\tfrac{q_x}{q_u}\left(x_i - x_m\right) \tag{9.12b}$$

where $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$ and m is the median value of the set $\{1, \ldots, n\}$. This control-law says that the amount of charge $v_i^\star$ that should be removed from cell x_i is proportional to the deviation of the cells state-of-charge x_i from the state-of-charge x_m of the median cell m. The amount of time $\tau^\star$ needed to implement this charge redistribution $v^\star$ is proportional to the difference between total amount of charge $q_x \sum_{i=1}^{m-1} x_i$ in the cells above the median m and total amount of charge $q_x \sum_{i=m+1}^{n} x_i$ in the cells below the median m.

For a battery pack with an even-number of cells n, there is ambiguity about which cells is the median. In this case the orbit controller switches between three rules based on the relative sizes of the state-of-charge $x_{n/2}$ of the upper-median cell $\frac{n}{2}$ and the state-of-charge $x_{n/2+1}$ of the lower-median cell $\frac{n}{2}+1$. The control-law is given by

$$\tau^\star = \frac{q_x}{q_u}\left(\textstyle\sum_{i=1}^{n/2} x_i - \sum_{i=n/2+1}^{n} x_i\right) \tag{9.13}$$

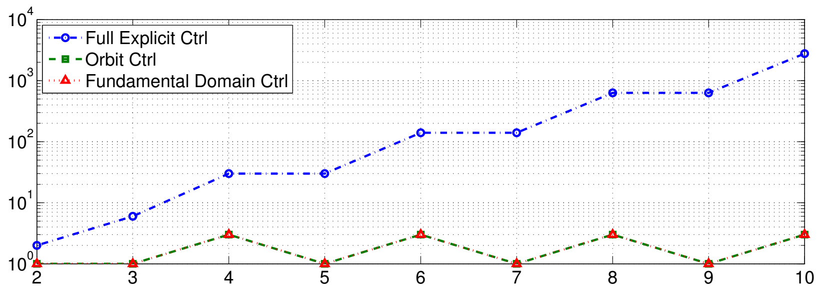
$$v_i^\star = \frac{q_x}{q_u}\begin{cases} -x_i + x_{n/2} & \text{for } x \in \mathcal{R}_1 \\ -x_i + x_{n/2+1} & \text{for } x \in \mathcal{R}_2 \\ -F_3 x & \text{for } x \in \mathcal{R}_3 \end{cases} \tag{9.14}$$

where $\tau^\star = \Phi_{\mathcal{U}}(v^\star)$ and F_3 is the feedback matrix for region $\mathcal{R}_3$. In region $\mathcal{R}_1$, this controller has the cells track the upper-median cell $x_{n/2}$. In region $\mathcal{R}_2$, this controller has the cells track the lower-median cell $x_{n/2+1}$. In region $\mathcal{R}_3$, the controller uses feedback $v^\star = F_3 x$.

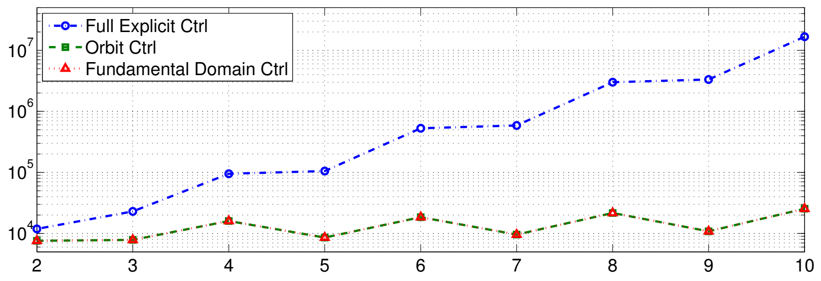
We conjecture that this result holds for arbitrary battery pack sizes n.

	$\kappa(x)$		$\kappa^o(x)$		$\hat{\kappa}(x)$	
# of Cells	Pieces $\lvert\mathcal{I}\rvert$	Memory	Pieces $\lvert\mathcal{I}/\mathcal{G}\rvert$	Memory	Pieces $\lvert\hat{\mathcal{I}}\rvert$	Memory
2	2	11.9	1	7.6	1	7.6
3	6	22.9	1	7.8	1	7.8
4	30	95.2	3	16.0	3	16.0
5	30	105.3	1	8.6	1	8.6
6	140	529.4	3	18.5	3	18.5
7	140	586.9	1	9.6	1	9.6
8	630	2999.3	3	21.6	3	21.6
9	630	3327.5	1	10.8	1	10.8
10	2772	16714.8	3	25.4	3	25.4

Table 9.7: Cell-to-stack common topology balancing controllers. Number of pieces and memory (kilobytes) for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller $\hat{\kappa}(x)$.



(a) Number of Pieces



(b) Memory

Figure 9.7: Cell-to-stack common topology battery balancing controllers. (a) Number of pieces $|\mathcal{I}|$ for the full explicit controller $\kappa(x)$, $|\mathcal{I}/\mathcal{G}|$ for the orbit controller $\kappa^o(x)$, and $|\hat{\mathcal{I}}|$ for the fundamental domain controller. (b) Memory requirements in bytes for the full explicit controller $\kappa(x)$, the orbit controller $\kappa^o(x)$, and the fundamental domain controller.

Chapter 10

Conclusions

In this dissertation we developed the theory of symmetry in model predictive control.

In Chapter 4 we defined symmetry for constrained linear systems and model predictive control problems. A symmetry of a model predictive control problem is a state-space and input-space transformation that preserves the dynamics, constraints, and cost function. We proved that the symmetries of the model predictive control problem are symmetries of the explicit model predictive controller. In particular for a linear system subject to polytopic constraints with a linear or quadratic cost, we showed that the symmetries relate the controller pieces.

In Chapter 5 we showed how to identify the symmetries of constrained linear systems. We showed how to transform this problem into a graph automorphism problem. We extended this result to find the symmetries of linear model predictive control problems with linear and quadratic cost functions.

In Chapter 7 we proposed two explicit model predictive control designs, the orbit controller and fundamental domain controller, that exploit symmetry to reduce the controller complexity and save memory. The orbit controller uses the symmetry relationships between controller pieces to eliminate pieces of the controller that are symmetrically redundant. This reduces the complexity of the controller and saves memory. We showed how the other controller pieces can be reconstructed online using symmetry. We showed that implementing the orbit controller has the same computational complexity as the standard implementation of an explicit model predictive controller. We analyzed the complexity reduction provided by the orbit controller. We showed that the orbit controller will never require more memory than the standard explicit controller and that the orbit controller offers significant memory savings when the number of group generators is small compared to the group size.

The fundamental domain controller reduces complexity by solving the constrained finite-time optimal control problem on a subset of the state-space called a fundamental domain. We defined a fundamental domain as a subset that contains at least one representative from each point of a set. In Chapter 6 we provided an algorithm for constructing fundamental domains. This algorithm was used to synthesize the fundamental domain controller. Implementing the fundamental domain controller requires mapping the state into the fundamental domain where the control-law is defined. In Chapter 6 we provided an algorithm for quickly mapping the state into the fundamental domain.

In the final part we applied our theory of symmetric model predictive control to the cell balancing problem. In Chapter 8 we define the battery balancing problem. Cell balancing is the problem of balancing the state-of-charge of the cells in a battery pack in order to increase the amount of charge that can be held in the pack. We proposed a control algorithm for solving the battery balancing problem and showed that our controller is persistently feasible and asymptotically stable. We described seven battery balancing hardware designs proposed in the literature. Finally we presented simulation results for our control design.

In Chapter 9 we used our symmetry theory to reduce the memory of explicit model predictive controllers for the battery balancing problem. We applied our symmetry theory to the seven battery balancing hardware designs presented in the previous section. Through a numerical study we demonstrate that the orbit controller and fundamental domain controller can significantly reduce the memory needed to store the explicit controller. In four of the seven battery balancing hardware designs to which we applied our methodology, the controller complexity did not grow with the battery pack size. We explained the intuition of how symmetry reduced controller complexity in these examples.

Bibliography

[1] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In *Nonlinear MPC*, Lecture Notes in Control and Information Sciences. Springer, 2009.

[2] F. Aurenhammer and R. Klein. *Voronoi Diagrams.* Elsevier, 1999.

[3] J. Avery, S. Rettrup, and J. Avery. *Symmetry-adapted Basis Sets.* World Scientific, 2012.

[4] Y. Bai, E. Klerk, D. Pasechnik, and R. Sotirov. Exploiting group symmetry in truss topology optimization. *Optimization and Engineering*, 2009.

[5] B. Bamieh, F. Paganini., and M. Dahleh. Distributed control of spatially invariant systems. *Transactions on Automatic Control*, 2002.

[6] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 2002.

[7] A. Bemporad and C. Rocchi. Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles. In *Conference on Decision and Control*, 2011.

[8] D. Bertsekas. *Network Optimization: Continuous and Discrete Models.* Athena Scientific, 1998.

[9] S. Bhat. and D. Bernstein. Nontangency-based lyapunov tests for convergence and stability in systems having a continuum of equilibria. *SIAM Journal on Control and Optimization*, 2003.

[10] N. Biggs. *Algebriac Graph Theory.* Cambridge University Press, 1974.

[11] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control.* Birkhäuser, 2008.

[12] R. Bodi, T. Grundhofer, and K. Herr. Symmetries of linear programs. *Note di Matematica*, 2010.

[13] R. Bodi, K. Herr, and M. Joswig. Algorithms for highly symmetric linear and integer programs. *Mathematical Programming*, 2011.

[14] F. Borrelli, A. Bemporad, and M. Morari. *Constrained Optimal Control and Predictive Control.* 2009.

[15] S. Boyd, P. Diaconis, P. Parrilo, and L. Xiao. Fastest mixing markov chain on graphs with symmetries. *Journal on Optimization*, 2009.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[17] D. Bremner, M. Dutour Sikiric, D. Pasechnik, T. Rehn, and A. Schuermann. Computing symmetry groups of polyhedra. *ArXiv e-prints*, 2012.

[18] D. Bremner, M. Dutour Sikiric, and A. Schrmann. Polyhedral representation conversion up to symmetries. In *Centre de Recherches Mathematiques*, 2009.

[19] T. Bresciani. Modeling, identification, and control of a quadrotor helicopter. Master's thesis, Lund University, 2008.

[20] J. Cao, N. Schofield, and A. Emadi. Battery balancing methods: A comprehensive review. In *Vehicle Power and Propulsion Conference*, 2008.

[21] R. Carter. *Molecular Symmetry and Group Theory.* Wiley, 1997.

[22] D. Cheng, G. Yang, and Z. Xi. Nonlinear systems possessing linear symmetry. *International Journal of Robust and Nonlinear Control*, 2007.

[23] F. Christophersen. Efficient evaluation of piecewise control laws defined over a large number of polyhedra. In *Opt Ctrl of Constrained PWA Systems*, Lecture Notes in Control and Information Sciences. Springer, 2007.

[24] F. Chuang, C. Danielson, and F. Borrelli. Optimality of certainty equivalence in expected value problems for uncertain linear systems. In *Submitted to Conference on Decision and Control*, 2014.

[25] R. Cogill, S. Lall, and P. Parrilo. Structured semidefinite programs for the control of symmetric systems. *Automatica*, 2008.

[26] R. D'Andrea and G. Dullerud. Distributed control design for spatially interconnected systems. *Transactions on Automatic Control*, 2003.

[27] C. Danielson and F. Borrelli. Symmetric explicit model predictive control. In *Nonlinear Model Predictive Control Conference*, 2012.

[28] C. Danielson and F. Borrelli. Identication of the symmetries of dynamics systems with polytopic constraints. In *American Control Conference*, 2014.

[29] C. Danielson and F. Borrelli. Symmetric linear model predictive control. *Accepted to Transactions on Automatic Control*, 2014.

[30] C. Danielson, F. Borrelli, D. Oliver, D. Anderson, M. Kuang, and T. Phillips. Balancing of battery networks via constrained optimal control. In *American Control Conference*, 2012.

[31] C. Danielson, F. Borrelli, Douglas Oliver, Dyche Anderson, and Tony Phillips. Constrained flow control in storage networks: Capacity maximization and balancing. *Automatica*, 2013.

[32] M. Daowd, N. Omar, P. Van Den Bossche, and J. Van Mierlo. Passive and active battery balancing comparison based on matlab simulation. In *Vehicle Power and Propulsion Conference*, 2011.

[33] P. Darga, K. Sakallah, and I. Markov. Faster symmetry discovery using sparsity of symmetries. In *Design Automation Conference*, 2008.

[34] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, 2006.

[35] E. de Klerk. Exploiting special structure in semidefinite programming: A survey of theory and applications. *European Journal of Operational Research*, 2010.

[36] R. Diestel. *Graph Theory*. Springer, 2005.

[37] F. Fagnani and J. Willems. Representations of symmetric linear dynamical systems. In *Conference on Decision and Control*, 1991.

[38] A. Fässler and E. Stiefel. *Group theoretical methods and their applications*. Birkhäuser, 1992.

[39] G. Folland. *Real analysis: modern techniques and their applications*. Wiley, 1999.

[40] E. Friedman. Fundamental domains for integer programs with symmetries. *Combinatorial Optimization and Applications*, 2007.

[41] The GAP Group. *GAP – Groups, Algorithms, and Programming*, 2014.

[42] K. Gatermann and P. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra*, 2004.

[43] C. Godsil and G. Royle. *Algebriac Graph Theory*. Springer, 2001.

[44] B. Goodwine and P. Antsaklis. Multiagent coordination exploiting system symmetries. In *American Control Conference*, 2010.

[45] B. Grünbaum and G. Ziegler. *Convex Polytopes*. Springer, 2003.

[46] S. Hartke and A. Radcliffe. Mckays canonical graph labeling algorithm. *Communicating Mathematics*, 2009.

[47] M. Hazewinkel and C. Martin. Symmetric linear systems: An application of algebraic systems theory. In *International Journal of Control*, 1983.

[48] D. Holt, B. Eick, and E. O'Brien. *Handbook of Computational Group Theory (Discrete Mathematics and Its Applications)*. Chapman and Hall/CRC, 2005.

[49] T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *Transactions on Automatic Control*, 2003.

[50] C. Jones, P. Grieder, and S.V. Rakovic. A logarithmic-time solution to the point location problem for parametric linear programming. *Automatica*, 2006.

[51] C. Jones, E. Kerrigan, and J. Maciejowski. On polyhedral projection and parametric programming. *Journal of Optimization Theory and Applications*, 2008.

[52] C. Jones and M. Morari. Polytopic approximation of explicit model predictive controllers. *Transactions on Automatic Control*, 2010.

[53] T. Junttila and P. Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In *Proc. of Analytic Algorithms and Combinatorics*, 2007.

[54] V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 2008.

[55] S. Koehler and F. Borrelli. Building temperature distributed control via explicit mpc and "trim and respond" methods. In *European Control Conference*, 2013.

[56] M. Kvasnica and M. Fikar. Clipping-based complexity reduction in explicit mpc. *Transactions on Automatic Control*, 2012.

[57] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

[58] M. Kvasnica, J. Löfberg, and M. Fikar. Stabilizing polynomial approximation of explicit mpc. *Automatica*, 2011.

[59] M. Kvasnica, I. Rauova, and M. Fikar. Automatic code generation for real-time implementation of model predictive control. In *Computer-Aided Control System Design*, 2010.

[60] S. Lang. *Algebra*. Springer, 2002.

[61] W. Chung Lee, D. Drury, and P. Mellor. Comparison of passive cell balancing and active cell balancing for automotive batteries. In *Vehicle Power and Propulsion Conference*, 2011.

[62] F. Margot. Symmetry in integer linear programming. In *50 years of linear programming 1958-2008*. Springer, 2010.

[63] B. McKay. Practical graph isomorphism. In *Manitoba Conference on Numerical Mathematics and Computing*, 1981.

[64] R. Megginson. *An introduction to Banach space theory.* Springer-Verlag, 1998.

[65] F. Menegazzo. The number of generators of a finite group, 2003.

[66] S. Moore and P. Schneider. A review of cell equalization methods for lithium ion and lithium polymer battery systems. In *Society of Automotive Engineers World Congress*, 2001.

[67] L. Moreau. Stability of multiagent systems with time-dependent communication links. *Transactions on Automatic Control*, 2005.

[68] P. Parrilo and S. Lall. Semidefinite programming relaxations and algebraic optimization in control. In *European Journal of Control*, 2003.

[69] G. Plett. Extended kalman filtering for battery management systems of lipb-based hev battery packs. *Journal of Power Sources*, 2004.

[70] M. Preindl, C. Danielson, and F. Borrelli. Performance evaluation of battery balancing hardware. In *European Control Conference*, 2013.

[71] M. Preindl, E. Schaltz, and P. Thogersen. Switching frequency reduction using model predictive direct current control for high-power voltage source inverters. *Transactions on Industrial Electronics*, 2011.

[72] J. Rawlings and D. Mayne. *Model Predictive Control: Theory and Design.* Nob Hill Pub, 2009.

[73] B. Recht and R. D'Andrea. Distributed control of systems over discrete groups. *Transactions on Automatic Control*, 2004.

[74] Á. Seress. *Permutation Group Algorithms.* Cambridge University Press, 2003.

[75] J.P. Serre. *Linear representations of finite groups.* Springer-Verlag, 1977.

[76] C. Speltino, D. Domenico, G. Fiengo, and A. Stefanopoulou. Experimental validation of a lithium-ion battery state of charge estimation with an extended kalman filter. In *European Control Conference*, 2009.

[77] C. Speltino, D. Di Domenico, G. Fiengo, and A. Stefanopoulou. Comparison of reduced order lithium-ion battery models for control applications. In *Conference on Decision and Control*, 2009.

[78] A. Szucs, M. Kvasnica, and M. Fikar. A memory-efficient representation of explicit mpc solutions. In *Conference on Decision and Control*, 2011.

[79] A. Tannenbaum. *Invariance and System Theory: Algebraic and Geometric Aspects*. Springer-Verlag, 1981.

[80] P. Tondel, T. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 2003.

[81] S. Waffler, M. Preindl, and J. W. Kolar. Multi-objective optimization and comparative evaluation of si soft-switche and sic hard-switched automotive dc-dc converters. In *Conference on Industrial Electronics*, 2009.

[82] Y. Wang and M. Morari. Structure of hierarchical linear systems with cyclic symmetry. *Systems and Control Letters*, 2009.

[83] H. Williams. Fourier's method of linear programming and its dual. *American Mathematical Monthly*, 1986.

[84] M. Zeilinger, C. Jones, and M. Morari. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. *Transactions on Automatic Control*, 2011.

[85] G. Ziegler. *Lectures on Polytopes*. Springer, 1995.