Stabilization of Polytopes for Fully Actuated Euler-Lagrange Systems

by

Eugene Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2014

Abstract

Given an Euler-Lagrange system and a convex polytope in its output space, we design a switched feedback controller that drives the output to the polytope. On the polytope, the system output tracks assigned trajectories or follows assigned paths. The study of this problem is motivated by industrial applications such as robotic painting, welding and three dimensional printing. Many engineering systems, such as robotic manipulators, can be modelled with Euler-Lagrange equations, and many engineered surfaces, designed using software, are naturally modelled as convex polytopes. We use feedback linearization to decompose the design problem into two subproblems; stabilizing the polytope surface, and controlling its motion along the surface.

The first subproblem, known as the design of the transversal controller, leverages the fact that a polytope can be represented as a finite union of facets. The controller determines the closest facet to the system output and stabilizes that facet by stabilizing its corresponding hyperplane via feedback linearization. The transversal dynamics can be stabilized using linear controllers. At the boundary of a facet, we propose a switching law that ensures weak invariance of the polytope for the closed-loop system.

The second subproblem, known as the design of the tangential controller, enforces desired dynamics while the system output is restricted to the polytope. We investigate control specifications such as following a predefined path on the surface and tracking a trajectory that moves along the surface. The separation of the transversal and tangential control design phases is possible because feedback linearization decouples the transversal and tangential dynamic subsystems.

This approach to control design is demonstrated experimentally on a four degree-of-freedom robotic manipulator. The experimental implementation is made robust to modelling uncertainty via Lyapunov re-design methods.

Acknowledgements

I would like to thank my supervisor Christopher Nielsen for all his guidance and the opportunity to work together these past few years. I am very grateful for the supportive environment that he has provided, and all the knowledge that I have gained.

I am also grateful for the help I've received from those who I have worked with throughout my time. I am especially thankful for Rajan Gill for all of his assistance through the experimental process.

I would like to thank Esmeralda Siddarta for helping me edit my thesis, and in general for putting up with me.

I am most thankful to my friends and family, who have been with me through this process.

Dedication

I would like to dedicate this thesis to everyone who helped me throughout my graduate career. To those who helped teach me the knowledge I have today. To those that helped me become a better teacher. Most importantly, to those who helped me when times were tough. I certainly could not have become to person I am today without you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter formally introduces the problem studied in this thesis and provides a literature review of relevant research.

1.1 Motivation

This thesis is motivated by the use of robotic manipulators in the manufacturing industry. For instance, robotic manipulators painting the fuselage of the Boeing 777 [19]. In this application, a team of robotic manipulators, naturally modelled using Euler-Lagrange equations, work together to have their end effectors approach the surface of the fuselage. The fuselage can be modelled, using computer aided design (C.A.D.) software, as a polytope. Once the surface has been stabilized, tasks along the surface, such as washing, applying solvent, rinsing and spraying, take place. These tasks require the robot's end effector to stay close to the surface, while the motion along the surface is kept smooth to accomplish a clean finish.

Another motivating example is surface stabilization required in three-dimensional printing. The printer tool tip, which once again can be modelled using Euler-Lagrange equations, must be driven to the surface of a polytope modelled in C.A.D. software. Once on the "surface" of the polytope, the tool tip must move smoothly to properly dispense the printing material. If the tool tip moves too far away from the polytope surface, or has unpredictable motion along the surface, the resulting printed model will be of poor quality.

1.2 Problem formulation

Partially motivated by the above scenarios and as well as previous research [49], we introduce the problem considered in this thesis.

1.2.1 Class of systems

Consider an Euler-Lagrange system of the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)\tau, \tag{1.1}$$

where $q \in \mathbb{R}^N$ is the vector of generalized coordinates, N is the number of degrees-of-freedom and $\dot{q} \in \mathbb{R}^N$ is the vector of generalized velocities. We assume that there are m control inputs, $\tau \in \mathbb{R}^m$. The $N \times N$ inertia matrix $M(q)$ is a smooth function of q and positive definite for all $q \in \mathbb{R}^N$. The matrix $C(q,\dot{q}) \in \mathbb{R}^{N\times N}$ represents the centripetal and Coriolis terms while the vector $G(q) = \nabla P_q$ is the gradient of the system's potential energy $P : \mathbb{R}^N \to \mathbb{R}$ and represents gravitation effects. Finally, $B : \mathbb{R}^N \to \mathbb{R}^{N\times m}$ is assumed to be smooth and full rank for all $q \in \mathbb{R}^N$ [47].

The output of the system is given by

$$y = h(q), \qquad h : \mathbb{R}^N \to \mathbb{R}^p, \tag{1.2}$$

where p is the dimension of the output space and y represents the variables we wish to control. In robotics applications, the function $h(q)$ can be viewed as the forward kinematics of the system. For example, the end effector position of a robotic manipulator.

System (1.1) can be re-written in state space form by selecting the state variable $x \coloneqq (x_c, x_v) \coloneqq (q, \dot{q})$. With this choice system (1.1) can be expressed as a smooth control affine system

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i \eqqcolon f(x) + g(x)u. \tag{1.3}$$

where

$$\begin{aligned} f(x) &\coloneqq \begin{bmatrix} x_v \\ f_v(x) \end{bmatrix} = \begin{bmatrix} x_v \\ M^{-1}(q)\left(-C(q,\dot{q}) - G(q)\right) \end{bmatrix}, \\ g(x) &\coloneqq \begin{bmatrix} 0 \\ g_v(x_c) \end{bmatrix} = \begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix}. \end{aligned}$$

The control input $u = (u_1, \ldots, u_m)$ is obtained from τ in (1.1) via

$$u \coloneqq B(q)\tau. \tag{1.4}$$

1.2.2 Class of polytopes

We assume that we are given a polytope P in the output space $\mathbb{R}^p$ of system (1.1). We assume that the polytope lies within a region of the output space where the Jacobian of the output map $h(q)$ has full rank. To make this assumption precise, we first define a functional workspace.

Definition 1.2.1. A **functional workspace** of (1.1) with output (1.2) is an open and connected set $W \subseteq \mathbb{R}^p$ with the property that, for any $x_c \in W$, the Jacobian $\mathrm{d}h_{x_c}$ is full rank.

An example of a functional workspace space is the dexterous workspace of a robotic manipulator.

Assumption 1. *The set P to be stabilized is a compact full-dimensional polytope contained in a functional workspace $W \subseteq \mathbb{R}^p$ of* (1.1) *with output* (1.2).

Definitions of the terms employed in Assumption 1 are found in Section 2.3.

1.2.3 Problem statement

Problem 1. Given a control system of the form (1.1), with output (1.2), and a polytope P satisfying Assumption (1), find, if possible, a finite number of smooth feedback controllers $\mathcal{U} = \{\mathrm{u}_1, \ldots, \mathrm{u}_M\}$, $\mathrm{u}_i : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^m$, $i \in \{1, \ldots, M\}$ and a state-dependent switching law

$$\mathcal{I} : \mathbb{R}^N \times \mathbb{R}^N \to \{1, \ldots, M\} \tag{1.5}$$

such that the closed-loop system

$$\begin{aligned} \dot{x} &= f(x) + g(x)\mathrm{u}_{\mathcal{I}(x)} \\ y &= h(x_c) \end{aligned}$$

satisfies the following:

(a) The surface of the polytope P is practically asymptotically stable with respect to the output trajectory $y(t) = h(x_c(t))$.[1]

[1]Practical asymptotic stability is defined in Definition 2.5.1.

(b) The surface of the polytope P is output weakly invariant.[2]

(c) The output $y(t) = h(x_c(t))$, restricted to the surface of the polytope P, either (i) follows paths defined on the surface of P or (ii) otherwise has application specific desired dynamics. •

Problem 1 is conceptually similar to the problem considered in [49]. The problem considered in this thesis differs in two fundamental ways (i) the target set P is not a smooth manifold and hence cannot be represented as the zero level set of a smooth function (ii) we seek to track trajectories and follow paths on the surface of the polytope P. In [49] the motion on the surface of the target set corresponded solely to damped motion.

1.3 Literature review

The following literature review discusses current research related to Problem 1. It has been organized by the major fields relevant to solving Problem 1.

1.3.1 Set stability and stabilization

In the field of dynamics, Barbashin [5] and Zubov [55] initiated a thorough study of the stability of sets. Zubov considers closed, not necessarily compact, invariant sets and gives necessary and sufficient conditions for the stability of the sets. His conditions are in terms of qualitative properties of the trajectories of the dynamical system and in terms of a scalar functional, a Lyapunov-like function. Bhatia [8] arrives at similar results. In 1995 Lin, Sontag and Wang [36] study parameterized families of systems of the form

$$\dot{x} = f(x, \mu) \tag{1.6}$$

with state $x \in \mathbb{R}^n$ and parameter $\mu \in \mathbb{R}^\ell$. The authors introduce a Lyapunov-theoretic necessary and sufficient condition for (1.6) to be uniformly, globally asymptotically stable with respect to a set $\mathcal{A}$. The set of interest for robust stability of (1.6) is $\mathcal{A} \times \mathbb{R}^\ell$ with the additional state equation $\dot{\mu} = 0$. The authors provide a converse Lyapunov theorem for this class of systems to be robustly stable with respect to a set.

[2]Output weak invariance of a set is defined in Definition 2.5.3.

Despite the voluminous work on set stability in the dynamics community, the problem of set stabilization has, comparatively, less work. The authors in [11] solve a set stabilization problem using, as in this thesis, not necessarily smooth controls for systems of the form

$$\dot{x} = Ax + u$$

where $u \in K$ where K is a convex closed cone of the state space $\mathbb{R}^n$ and the set to be stabilized is an invariant, convex closed cone $M \subseteq \mathbb{R}^n$.

In 1995, Banaszuk and Hauser publish a pair of papers [3, 4] similar in spirit to the work of Nijmeijer and Campion [41]. There the authors consider single input control-affine systems and attempt to stabilize an invariant periodic orbit, Γ, of the unforced system. Closely related to the set stabilization problem is output stabilization. Isidori [29] and Nijmeijer and van der Schaft [42] provide algorithms for control-affine systems which find the largest controlled invariant submanifolds on which the system output is zero.

Albertini and Sontag [1] extended the work of Lin *et al.* [36] to time-varying systems with control. In [40] the results of Banaszuk and Hauser were extended to arbitrary embedded submanifolds of the state space and multi-input systems. In [39] a similar problem is solved for single-input systems under the more restrictive assumption that only an output, and not the entire state, is available for feedback. Passivity based approaches in [16] have also been investigated.

Polytope sets have been studied in control problems since the 1970's [9]. Polytopes are often a natural choice in many applications, as their flexible nature allows for the expression of many physical constraints, allowing for more accurate representations when dealing with dynamic systems. However, the trade off of this flexibility is that the polytope representation is often times complex. This complex representation, and general non-smoothness, contrasts the smooth manifolds that set stabilization techniques are typically applied to.

1.3.2 Polytopes and hybrid systems

Polytopes have been widely used in the control literature in the field of hybrid systems. In many of these applications the system is being driven to, or through, a particular facet of a polytope, often at a predetermined speed and direction. The application areas for these problems include motion planning [6], [7], [21], and control under safety constraints [24], [25]. These algorithms can restrict the state of a system to a polytope in its state space [37].

In most of the above papers, the system, often linear affine, to be controlled is defined to evolve on a polytope. As a result, the polytope is intrinsically invariant. In this thesis the polytope exists as a subset of the state space. As a result, invariance is not guaranteed and must be enforced via feedback.

These algorithms achieve their objectives by defining admissible regions of the state space, often based on the desired trajectory or velocity of the system output. Once these admissible regions have been defined, a standard form for the controllers is substituted into the closed loop equation. The coefficients of the controller are then determined to force the system to operate in the admissible region. Through backwards recursion algorithms, a series of piecewise-affine controllers can be computed to control the motion of the hybrid system.

The ideas from hybrid systems and reach control are complimentary to the work presented in this thesis. Once the polytope P has been rendered weakly invariant, we can then use the aforementioned results to achieve objective (c) in Problem 1.

1.3.3 Polytope descriptions in system output space

Polytopes have been used in the field of robotics to model the surroundings of mobile robots for path planning and navigation purposes [7], [13], [12]. In these applications, large maps of the surroundings are modelled and often stored in memory for future use. By modelling the surroundings in this way, complex tasks such as motion and trajectory planning, demonstrated in [38], [7], and [33], can be accomplished. Furthermore, the modelling of the surroundings, and computation of trajectories can be completed in real time on top of other control tasks. The use of polytopes in these complex situations demonstrates their applicability to other control tasks, such as surface stabilization.

1.3.4 Trajectory tracking along surfaces

Once on the surface, we are interested in having the system output track a trajectory or follow paths defined on the surface. In [48], [46], [53] and [23], a sliding mode controller is implemented to track a trajectory while stabilizing the surface of a fully actuated mechanical system. In these implementations, the target surface is modelled, either with a set or a gradient, and then the sliding mode controller used to drive the system towards the surface through its forward dynamics. This approach allows for precise control of the system, and makes use of the information available from the geometry of the mechanical system. However in these applications, the surface stabilization problem is not decoupled from the tangential control problem. This causes the overall controller design to be complex, and limits the flexibility to accomplish tangential design objectives.

1.4 Organization and contribution

This thesis is organized as follows, Chapter 2 provides preliminary material and mathematical background, Chapter 3 outlines the details of the surface stabilization problem, Chapter 4 provides possible solutions to the tangential control problem, Chapter 5 provides experimental and simulation results.

The theoretical contributions of this thesis are primarily the formulation of the polytope stabilization problem and the algorithm presented to solve the problem.

The surface stabilization algorithm presented in this thesis can be broken down into the following steps.

1. Determine the closest facet to the system output
2. Stabilize the hyperplane spanning the closest facet
3. If possible, move along the stabilized surface as desired
4. Switch and stabilize new facet if necessary
5. Repeat steps 3 and 4

The main experimental contribution of this thesis is the verification, on a four degree-of-freedom robotic manipulator, of the aforementioned algorithm.

Chapter 2

Mathematical Preliminaries

The material presented in this chapter is drawn from several disciplines and fields of study including [26], [34], [32], [51], [44], [54].

2.1 Notation

We represent a vector $x \in \mathbb{R}^n$ as a column vector or as an n-tuple $x = (x_1, \ldots, x_n)$. Given vectors $x, y \in \mathbb{R}^n$, $\langle x,\ y\rangle$ denotes the Euclidean inner product and $\|x\|$ denotes the induced Euclidean norm. We denote by I_n the $n \times n$ identity matrix and $0_{m\times n}$ is an $m \times n$ zero matrix. Given two subspaces $\mathcal{X}_1$, $\mathcal{X}_2$ of a finite dimensional vector space $\mathcal{X}$, the symbol $\mathcal{X}_1 \oplus \mathcal{X}_2$ is used to indicate the subspace $\mathcal{X}_1 + \mathcal{X}_2$ when $\mathcal{X}_1$ and $\mathcal{X}_2$ are independent, i.e., $\mathcal{X}_1 \cap \mathcal{X}_2 = \{0\}$. Given a matrix $A \in \mathbb{R}^{m\times n}$ with rank m, its Moore-Penrose pseudoinverse is $A^+ = A^\top \left(AA^\top\right)^{-1}$. The matrix A^+ is a right inverse, i.e., $AA^+ = I_m$.

Given a set $S \subseteq \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$, the point-to-set distance from x to S is

$$\operatorname{dist}(x, S) \coloneqq \inf_{s\in\mathcal{S}} \|x - s\|.$$

The boundary of a set S is written ∂S. For $p \in \mathbb{R}^n$ and $\delta > 0$, the set $B_\delta(p) \coloneqq \{x \in \mathbb{R}^n : \|x - p\| < \delta\}$ is called an open ball of radius δ centred at p or a neighbourhood of p.

Given a function $f : A \to B$ and a subset $\Omega \subseteq A$, the image of Ω under f is the set

$$f(\Omega) = \{y \in B : (\exists\, x \in \Omega)\, y = f(x)\}.$$

The set $f(A)$ is the image of f. When $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ are open and connected sets and f is differentiable, for each $x \in A$ the $n \times m$ Jacobian of f is denoted by $\mathrm{d}f_x$. Let $f,\, g : \mathbb{R}^n \to \mathbb{R}^n$ be smooth vector fields and $\phi : \mathbb{R}^n \to \mathbb{R}^m$ a smooth map. We use the standard notation for iterated Lie derivatives

$$\begin{aligned}
&L_f^0\phi := \phi,\\
&L_f^k\phi := L_f(L_f^{k-1}\phi) = \langle \mathrm{d}L_f^{k-1}\phi_x, f(x)\rangle,\\
&L_gL_f\phi := L_g(L_f\phi) = \langle \mathrm{d}L_f\phi_x, g(x)\rangle.
\end{aligned}$$

2.2 Linear functions

A linear function between vector spaces can be represented as a matrix. In this thesis we do not distinguish between a linear map and its matrix representation.

Definition 2.2.1. The **rank** of a linear function $A : \mathcal{X} \to \mathcal{Y}$ between finite-dimensional vector spaces is $\dim(\mathrm{Im}(A))$.

Definition 2.2.2. The **kernel** of a linear function $A : \mathcal{X} \to \mathcal{Y}$ between finite-dimensional vector spaces is

$$\mathrm{Ker}(A) := \{x \in \mathbb{R}^n : Ax = 0\}.$$

The following two linear functions are conceptually important in this thesis.

Definition 2.2.3. Let $\mathcal{X}$ be a real n-dimensional vector space and let $\mathcal{V}$ and $\mathcal{W}$ be subspaces such that $\mathcal{X} = \mathcal{V} \oplus \mathcal{W}$. The **projection on $\mathcal{V}$ along $\mathcal{W}$** is a map $\tilde{Q} : \mathcal{X} \to \mathcal{X}$ that maps $x = v + w$ ($v \in \mathcal{V}$, $w \in \mathcal{W}$) to v.

Definition 2.2.4. Let $\mathcal{X}$ be a real n-dimensional vector space and let $\mathcal{V}$ and $\mathcal{W}$ be subspaces such that $\mathcal{X} = \mathcal{V} \oplus \mathcal{W}$. The **natural projection on $\mathcal{V}$ along $\mathcal{W}$** is a map $Q : \mathcal{X} \to \mathcal{V}$ that maps $x = v + w$ ($v \in \mathcal{V}$) to v.

The difference between the two projections above lies in their co-domains. In terms of matrix representations, let $\mathcal{X} = \operatorname{span}\{x_1, \ldots, x_n\}$ where the first $v \in \mathbb{N}$, $v < n$ vectors in the list span $\mathcal{V}$ and the last $n - v$ vectors span $\mathcal{W}$ then

$$\begin{aligned} \tilde{Q} &= \begin{bmatrix} I_v & 0_{v\times(n-v)} \\ 0_{(n-v)\times v} & 0_{(n-v)\times(n-v)} \end{bmatrix} \\ Q &= \begin{bmatrix} I_v & 0_{v\times(n-v)} \end{bmatrix}. \end{aligned}$$

The sets of interest in this thesis are the facets of polytopes. A facet of a polytope can be viewed as an affine subspace.

Definition 2.2.5. A subset $\mathcal{A}$ of a vector space $\mathcal{Y}$ is an **affine subspace** of $\mathcal{Y}$ if there exists a $b \in \mathcal{Y}$ and a subspace $\mathcal{V}$ of $\mathcal{Y}$ such that

$$\mathcal{A} = \{y \in \mathcal{Y} : y = v + b,\ v \in \mathcal{V}\}.$$

The **dimension** of $\mathcal{A}$ is the dimension of $\mathcal{V}$.

An affine subspace can be represented as the zero-level set of a linear affine function. Consider

$$\begin{aligned} s : \mathcal{Y} &\to \mathcal{Z} \\ y &\mapsto Ay + b \end{aligned} \tag{2.1}$$

with A onto, i.e., $\operatorname{rank}(A) = \dim(\mathcal{Z})$. Then the set

$$\mathcal{A} = \{y \in \mathcal{Y} : s(y) = Ay + b = 0\} \tag{2.2}$$

is an affine subspace with dimension $\dim(\operatorname{Ker}(A))$. To see this note that, since A is full rank,

$$\begin{aligned} \mathcal{A} &= \{y \in \mathcal{Y} : Ay + b = 0\} \\ &= \left\{y \in \mathcal{Y} : y = -A^\top (AA^\top)^{-1} b + v,\ v \in \operatorname{Ker}(A)\right\}. \end{aligned}$$

which is the definition of an affine subspace given previously.

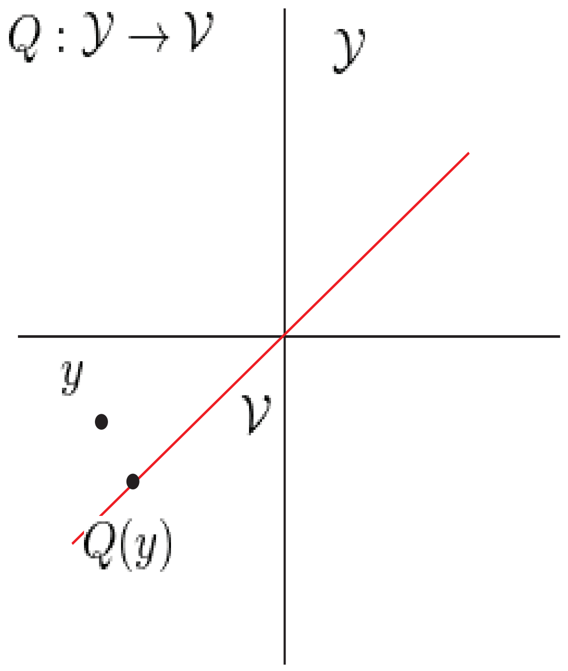
We now modify the definition of the natural projection from Definition 2.2.4 for affine subspaces.

Definition 2.2.6. Let

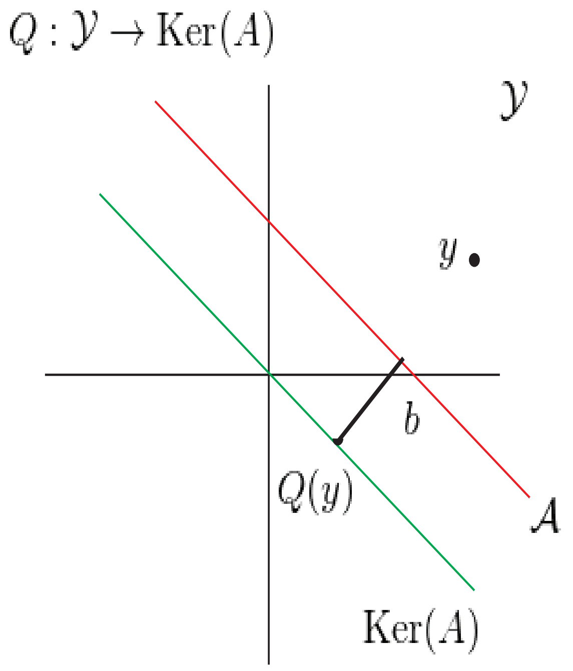
$$\mathcal{A} = \{y \in \mathcal{Y} : y = v + b,\ v \in \mathcal{V}\}$$

be an affine subspace of $\mathcal{Y}$. Let $\mathcal{W}$ be a subspace such that $\mathcal{Y} = \mathcal{V} \oplus \mathcal{W}$. The **natural projection on $\mathcal{A}$ along $\mathcal{W}$** is the natural projection on $\mathcal{V}$ along $\mathcal{W}$.

When $\mathcal{A}$ is represented as the zero level set of a function (2.2), then the natural projection onto $\mathcal{A}$ is just the natural projection onto the kernel of A, i.e., $\mathcal{V} = \mathrm{Ker}\,(A)$ in Definition 2.2.6. The conceptual difference between a natural projection onto a subspace and the natural projection onto an affine subspace is illustrated in Figure 2.1.



(a) Projection onto a subspace



(b) Projection onto an affine subspace

Figure 2.1: Natural projections.

Example 2.2.1. Consider a linear affine function $\mathbb{R}^3 \to \mathbb{R}$ where, using the natural bases for $\mathbb{R}^3$ and $\mathbb{R}$ we have

$$A = \begin{bmatrix} 3 & 1 & -2 \end{bmatrix}, \qquad b = 2.$$

Then

$$\operatorname{Ker}(A) = \operatorname{span}\left\{ \begin{bmatrix} 1 \\ -3 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \right\}.$$

and $\mathcal{W} := \operatorname{span}\{\operatorname{col}(1,0,0)\}$ is such that $\mathbb{R}^3 = \operatorname{Ker}(A) \oplus \mathcal{W}$. With these choices for the bases of $\mathbb{R}^3$, $\operatorname{Ker}(A)$ and $\mathcal{W}$ we can compute the projection and natural projection on $\operatorname{Ker}(A)$ along $\mathcal{W}$. For any point $y = y_1e_1 + y_2e_2 + y_3e_3 \in \mathbb{R}^3$ we have that

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -3 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \beta_1 \\ \beta_2 \end{bmatrix}.$$

The projection on $\operatorname{Ker}(A)$ along $\mathcal{W}$ maps y to $(0, \beta_1, \beta_2)$ while the natural projection on $\operatorname{Ker}(A)$ along $\mathcal{W}$ maps y to (β_1, β_2). Therefore

$$\tilde{Q} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} 0 & -\frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix}.$$

△

Example 2.2.1 shows that the choice of complementary subspace $\mathcal{W}$ changes the matrix representations of $\tilde{Q}$, Q. In this thesis we'll typically assume that the vector space is equipped with an inner product and take $\mathcal{W}$ to be the orthogonal complement of $\operatorname{Ker}(A)$.

Lemma 2.2.7. *Let $A : \mathcal{Y} \to \mathcal{Z}$ be onto and let $Q : \mathcal{Y} \to \operatorname{Ker}(A)$ be the natural projection on $\operatorname{Ker}(A)$ along a subspace $\mathcal{W}$. Then the linear function*

$$\begin{bmatrix} A \\ Q \end{bmatrix} : \mathcal{Y} \to \mathcal{Y}$$

is bijective.

Proof. The map is one-to-one since

$$\begin{aligned} \operatorname{Ker}\begin{bmatrix} A \\ Q \end{bmatrix} &= \operatorname{Ker}(A) \cap \operatorname{Ker}(Q) \\ &= \operatorname{Ker}(A) \cap \mathcal{W} \\ &= 0 \end{aligned}$$

where the last equality follows from the definition of Q. Therefore we have that

$$\dim(\mathcal{Y}) = \operatorname{rank}\left(\begin{bmatrix} A \\ Q \end{bmatrix}\right) + \dim\left(\operatorname{Ker}\begin{bmatrix} A \\ Q \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} A \\ Q \end{bmatrix}\right).$$

This shows that the map is onto which completes the proof. □

Finally we present basic results about the composition of linear maps. Let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ be three finite dimensional vector spaces. Let $A : \mathcal{Y} \to \mathcal{Z}$ and $B : \mathcal{X} \to \mathcal{Y}$ be linear transformations and define $M := AB : \mathcal{X} \to \mathcal{Y}$.

Proposition 2.2.8. *If $M : \mathcal{X} \to \mathcal{Z}$ is onto, then $A : \mathcal{Y} \to \mathcal{Z}$ is onto*

Proof. Since

$$\operatorname{Im}(M) = \operatorname{Im}(AB) \subseteq \operatorname{Im}(A)$$

and since by hypothesis

$$\operatorname{Im}(M) = \mathcal{Z}$$

we have that

$$\mathcal{Z} \subseteq \operatorname{Im}(A) \subseteq \mathcal{Z}$$

which shows that A is onto. □

Proposition 2.2.9. *Let $A : \mathcal{Y} \to \mathcal{Z}$ and let $\mathcal{Y}_1$ be an independent complement of* $\operatorname{Ker}(A)$ *such that*

$$\mathcal{Y} = \mathcal{Y}_1 \oplus \operatorname{Ker}(A).$$

Then A restricted to $\mathcal{Y}_1$ is one-to-one.

Proof. Suppose, by way of contradiction, that there exists $y_1, y_2 \in \mathcal{Y}_1$, where $y_1 \neq y_2$ and $Ay_1 = Ay_2$. Then

$$A(y_1 - y_2) = Ay_1 - Ay_2 = 0.$$

This implies that the non-zero vector $y_1 - y_2$ is in $\operatorname{Ker}(A)$ which contradicts the fact that y_1 and y_2 are elements of $\mathcal{Y}_1$. □

Proposition 2.2.10. *If $M : \mathcal{X} \to \mathcal{Z}$ is onto, then*

$$\operatorname{Ker}(A) + \operatorname{Im}(B) = \mathcal{Y}. \tag{2.3}$$

Proof. Let $\mathcal{Y}_1$ be an independent complement of $\operatorname{Ker}(A)$ such that

$$\mathcal{Y} = \mathcal{Y}_1 \oplus \operatorname{Ker}(A). \tag{2.4}$$

In order to prove this claim, it is sufficient to show that

$$\mathcal{Y}_1 \subseteq \operatorname{Im}(B). \tag{2.5}$$

Consider the pre-image of $\mathcal{Y}_1$ under B

$$B^{-1}\mathcal{Y}_1 = \left\{x \in \mathcal{X} : Bx \in \mathcal{Y}_1\right\}.$$

By [51, Section 0.4], there exists a, not necessarily unique, subspace $\mathcal{X}_1 \subseteq \mathcal{X}$ such that

$$\dim(\mathcal{X}_1) = \dim(\mathcal{Y}_1 \cap \operatorname{Im}(B)), \qquad B^{-1}\mathcal{Y}_1 = \mathcal{X}_1 \oplus \operatorname{Ker}(B).$$

Let $\mathcal{X}_2 \subseteq \mathcal{X}$ be an independent complement so that

$$\mathcal{X} = \mathcal{X}_1 \oplus \mathcal{X}_2 \oplus \operatorname{Ker}(B).$$

With this decomposition we have that[1] $B(\mathcal{X}_1 \cap \mathcal{X}_2) = B\mathcal{X}_1 \cap B\mathcal{X}_2 = \{0\}$ and $B\mathcal{X}_1 \subseteq \mathcal{Y}_1$.

We can now show that (2.5) holds. Let $y_1 \in \mathcal{Y}_1$ be arbitrary. Let $z := Ay_1$. Since M is onto, there exists an $x \in \mathcal{X}$, such that $z = ABx$. The vector x can be uniquely written as $x = x_1 + x_2 + x_k$ with $x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2$ and $x_k \in \operatorname{Ker}(B)$. We have that

$$\begin{aligned} Bx &= B(x_1 + x_2 + x_k) \\ &= Bx_1 + Bx_2. \end{aligned}$$

Since $B(\mathcal{X}_2) \cap \mathcal{Y}_1 = \{0\}$ and $z = ABx = Ay_1$, we have that $Bx_1 = y_1$, i.e., $y_1 \in \operatorname{Im}(B)$. Since y_1 was arbitrary, we conclude that (2.5) holds.

□

[1]For general subspaces and linear transformations it is only true that $B(\mathcal{X}_1 \cap \mathcal{X}_2) \subseteq B\mathcal{X}_1 \cap B\mathcal{X}_2$. However since $(\mathcal{X}_1 + \mathcal{X}_2) \cap \operatorname{Ker}(B) = \mathcal{X}_1 \cap \operatorname{Ker}(B) + \mathcal{X}_2 \cap \operatorname{Ker}(B)$ we have equality.

Proposition 2.2.11. *If $A : \mathcal{Y} \to \mathcal{Z}$ is onto and*

$$\operatorname{Ker}(A) + \operatorname{Im}(B) = \mathcal{Y} \tag{2.6}$$

then $M : \mathcal{X} \to \mathcal{Z}$ is onto.

Proof. Let $z \in \mathcal{Z}$ be arbitrary and let $y \in \mathcal{Y}$ be such that $z = Ay$. Since A is onto, such a y is guaranteed to exist. By (2.6), y can be written as $y = y_1 + y_k$ with $y_1 \in \operatorname{Im}(B)$ and $y_k \in \ker(A)$. Therefore, there exists some $x \in \mathcal{X}$ such that $y_1 = Bx$ and hence $Mx = ABx = Ay_1 = z$. $\square$

2.3 Convex analysis

We now define some concepts from convex analysis that are used frequently throughout this thesis. The main references for this section are [26], [44], [54].

Definition 2.3.1. A set of $m + 1$ points $v_0, \dots, v_m \in \mathbb{R}^n$ are **affinely independent** if $\{v_1 - v_0, \dots, v_m - v_0\}$ are linearly independent.

Given a subset $S \subseteq \mathbb{R}^n$, the affine hull of S, denoted $\operatorname{aff}(S)$, is the unique smallest affine subspace containing S. Given a set of affinely independent points $v_0, \dots, v_m \in \mathbb{R}^n$, the set $\operatorname{aff}\{v_0, \dots, v_m\} = \{x \in \mathbb{R}^n : x = v + v_0,\ v \in \mathcal{V}\}$ where $\mathcal{V} = \operatorname{span}\{v_1 - v_0, \dots, v_m - v_0\}$. The dimension of $\operatorname{aff}\{v_0, \dots, v_m\}$ is the dimension of $\mathcal{V}$, see Definition 2.2.5.

Definition 2.3.2. A subset set $S \subseteq \mathbb{R}^n$ is **convex** if

$$(\forall x, y \in S)\,(\forall \lambda \in [0, 1])\,\lambda x + (1 - \lambda)y \in S.$$

In general, the dimension of a convex set S equals the dimension of its affine hull $\operatorname{aff}(S)$.

Theorem 2.3.3 ([44])**.** *A subset S of $\mathbb{R}^n$ is convex if and only if it contains all the convex combinations of its elements.*

All affine subspaces are convex. Given a subset $S \subseteq \mathbb{R}^n$, the convex hull of S, denoted $\operatorname{conv}(S)$, is the unique smallest convex set containing S.

Corollary 2.3.4 ([44])**.** *The convex hull of a set points $v_0, \ldots, v_m \in \mathbb{R}^n$ consists of all vectors of the form*

$$\lambda_0 v_0 + \cdots \lambda_m v_m, \qquad \sum_{i=0}^{m} \lambda_i = 1, \qquad \lambda_i \geq 0.$$

Definition 2.3.5 ([26], [44])**.** A set which is the convex hull of a finitely many points is called a **polytope**. If $v_0, \ldots, v_m \in \mathbb{R}^n$ are affinely independent then $\operatorname{conv}\{v_0, \ldots, v_m\}$ is called an **m-dimensional simplex** and $v_0, \ldots, v_m$ are called the **vertices** of the simplex. The simplex is called **full dimensional** if $m = n$.

By Definition 2.3.5, an m-dimensional simplex is a particular type of polytope. Furthermore, in light of Corollary 2.3.4, an m-dimensional simplex is completely characterized by its vertices. When $m = 0, 1, 2$, or 3, the simplex is called a point, (closed) line segment, triangle or tetrahedron, respectively.

Half-spaces are important examples of convex sets. For any $b \in \mathbb{R}$, $a \in \mathbb{R}^n$, the sets

$$\{x \in \mathbb{R}^n : \langle a,\ x \rangle \leq b\}, \qquad \{x \in \mathbb{R}^n : \langle a,\ x \rangle \geq b\}$$

are called **closed half-spaces**. A set that can be expressed as the intersection of finitely many closed half-spaces of $\mathbb{R}^n$ is called a **polyhedral convex set**. Polyhedral convex sets are better behaved than general convex sets because of their lack of "curvature". Polytopes, and hence simplicies, are examples of polyhedral convex sets. Therefore, given a polytope $\operatorname{conv}\{v_0, \ldots, v_m\}$, there exists an integer k, non-zero vectors $a_1, \ldots, a_k \in \mathbb{R}^n$ and scalars $b_1, \ldots, b_k \in \mathbb{R}$ such that

$$P \coloneqq \operatorname{conv}\{v_0, \ldots, v_m\} = \bigcap_{i=1}^{k} \{x \in \mathbb{R}^n : \langle a_i,\ x \rangle + b_i \leq 0\}. \tag{2.7}$$

The representation (2.7) of a polytope as the finite intersection of closed half-spaces is called an implicit representation. A simplex can thus be described entirely either by its implicit representation, or through a convex hull of a finitely many points.

Example 2.3.1. A cube in $\mathbb{R}^3$ is an example of a three dimensional polytope. The set

$$\begin{aligned} P &:= \operatorname{conv}\{v_0, \dots, v_7\}, \\ &= \operatorname{conv}\left\{ \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}2\\1\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\end{bmatrix}, \begin{bmatrix}2\\2\\1\end{bmatrix}, \begin{bmatrix}1\\1\\2\end{bmatrix}, \begin{bmatrix}2\\1\\2\end{bmatrix}, \begin{bmatrix}1\\2\\2\end{bmatrix}, \begin{bmatrix}2\\2\\2\end{bmatrix} \right\}, \end{aligned}$$

is a cube centered at $(0.5, 0.5, 0.5)$. It has implicit representation

$$P := \bigcap_{i=1}^{6} \left\{ x \in \mathbb{R}^3 : \langle a_i,\ x\rangle + 1 \leq 0 \right\},$$

where $a_i^\top$ is the ith row of

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

It is instructive to note that $\{v_0, \dots, v_7\}$ are not affinely independent and hence they are not vertices and the cube is not a simplex.

△

Definition 2.3.6 ([26]). The intersection of an m-dimensional polytope (2.7) with one of its supporting hyperplanes

$$\mathcal{F}_i := \{x \in \mathbb{R}^n : \langle a_i,\ x\rangle + b_i = 0\} \cap P, \tag{2.8}$$

is called a **facet** of P.

A facet is itself a polytope of dimension $m - 1$.

Example 2.3.2. A facet of the cube from Example 2.3.1 is

$$\mathcal{F}_1 := \left\{ x \in \mathbb{R}^3 : \langle a_1,\ x\rangle + 1 = 0 \right\} \cap P$$

where $a_1 = (0, 0, -1)$. The facet $\mathcal{F}_1$ is shown in Figure 2.2.

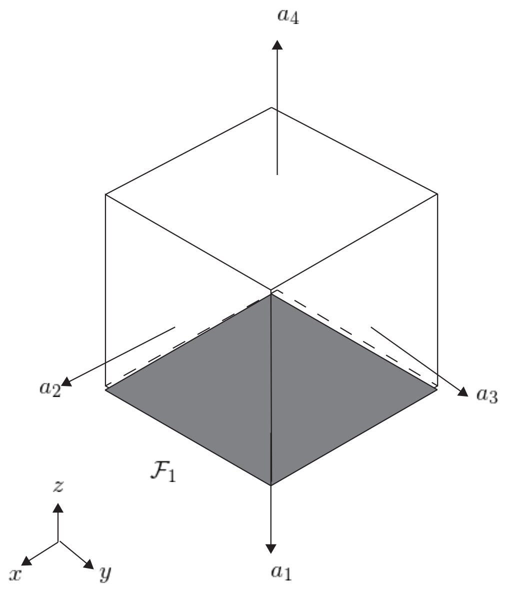


Figure 2.2: A facet of a three dimensional polytope.

△

Definition 2.3.7 ([54])**.** The non-empty intersection of two different facets of an m-dimensional polytope P is called the (i, j)**th ridge** of P

$$\mathcal{R}_{i,j} \coloneqq \mathcal{F}_i \cap \mathcal{F}_j = \{x \in \mathbb{R}^n : \langle a_i,\ x\rangle + b_i = \langle a_j,\ x\rangle + b_j = 0, j \neq i\} \cap P. \tag{2.9}$$

A ridge is itself an $m-2$ dimensional polytope.

Example 2.3.3. The ridges of a cube are simply the line segments between two of its vertices. For example, continuing Example 2.3.1, the $(2,3)$-ridge is

$$\mathcal{R}_{2,3} = \mathcal{F}_2 \cap \mathcal{F}_3 = \left\{x \in \mathbb{R}^3 : \left\langle \begin{bmatrix} 0 & 1 & 0 \end{bmatrix},\ x \right\rangle + 1 = \left\langle \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},\ x \right\rangle + 1 = 0 \right\}$$

The $(2,3)$-ridge is illustrated in Figure 2.3.

△

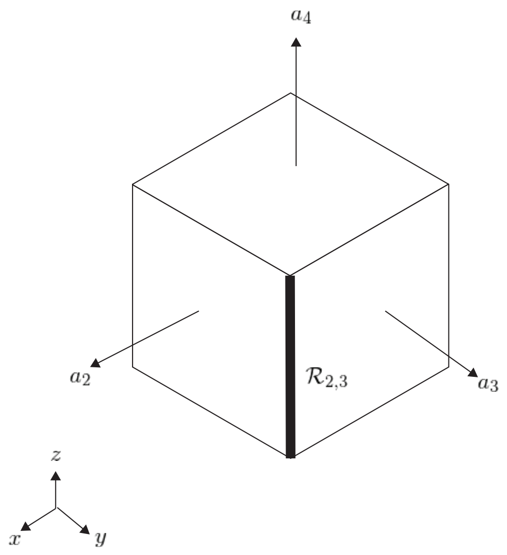


Figure 2.3: Ridge for a Cube

A convenient representation of a polytope P is an undirected, unweighted graph. Given an m-dimensional polytope (2.7), let $\mathscr{G}$ denote its associated graph. The graph is constructed as follows. Take the facets of P as the vertices of $\mathscr{G}$, i.e., $\mathsf{V}(\mathscr{G}) = \{\mathcal{F}_1, \ldots, \mathcal{F}_N\}$. The edges of the graph correspond to the ridges of P, i.e., $\mathsf{E}(\mathscr{G}) \subseteq \mathsf{V}(\mathscr{G}) \times \mathsf{V}(\mathscr{G})$ where $e_{ij} = (\mathcal{F}_i, \mathcal{F}_j)$ if $\mathcal{R}_{i,j} \neq \varnothing$.

The graph $\mathscr{G}$ can be visualized graphically or it can be represented algebraically as an adjacency matrix. The adjacency matrix of $\mathscr{G}$, denoted $A(\mathscr{G})$, is constructed element wise as

$$A(\mathscr{G})_{i,j} = \begin{cases} 1 & \text{if } (\mathcal{F}_i, \mathcal{F}_j) \in \mathsf{E}(\mathscr{G}) \\ 0 & \text{otherwise} \end{cases}$$

Example 2.3.4. A pictorial representation for a graph of a cube is shown in Figure 2.4. Its associated adjacency matrix is given by

$$A(\mathscr{G}) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

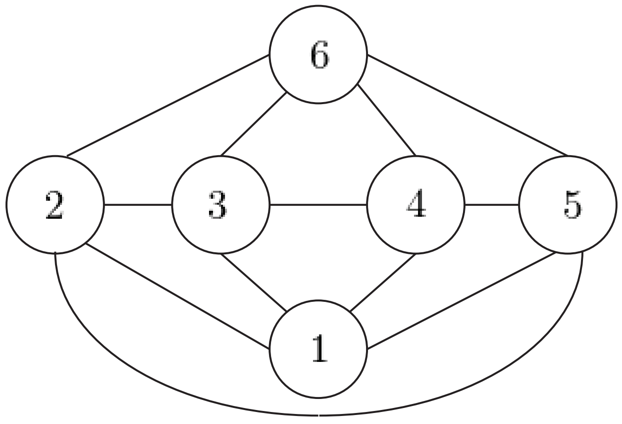


Figure 2.4: Graphical representation of the graph associated to a cube

$\triangle$

2.4 Robust control

In this thesis we use two different robust control techniques; one is based on backstepping, the other is based on Lyapunov redesign.

2.4.1 Backstepping without uncertainty

To understand robust backstepping, we start by describing the case without model uncertainty. Consider a smooth, time-invariant, control-affine, nonlinear control system of the form[2]

$$\dot{x}_1 = f(x_1) + g(x_1)x_2, \tag{2.10}$$
$$\dot{x}_2 = u, \tag{2.11}$$

where $x_1 \in \mathbb{R}^n$, $x_2 \in \mathbb{R}$, $u \in \mathbb{R}$. The smooth vector fields $f, g : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ are defined on an open and connected set D containing the origin with $f(0) = 0$. These vector fields known. The block diagram of this system is shown in Figure 2.5. The key idea in integrator backstepping is to treat the state x_2 as a virtual input to the "slow dynamics",

[2]There are more general versions of backstepping. In this section we focus on the basic case.

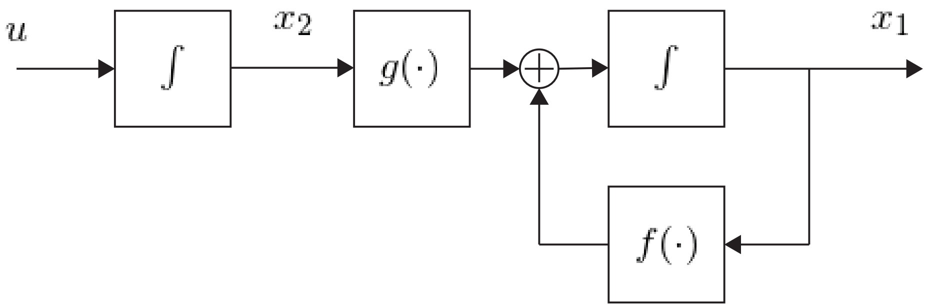


Figure 2.5: Backstepping Figure 1

i.e., the x_1-subsystem, design a feedback controller for the virtual input x_2, and bring in the integrator afterwards to suggest the stabilizing control law for u.

Assume there exists a smooth feedback $k : D \to \mathbb{R}$, $k(0) = 0$, that asymptotically stabilizes the origin of the slow dynamics. That is $x_1 = 0$ is an asymptotically stable equilibrium of the system $\dot{x}_1 = f(x_1) + g(x_1)k(x_1)$. Let $f_{CL}(x_1) \coloneqq f(x_1) + g(x_1)k(x_1)$. By the converse Lyapunov theorems [32, Theorem 4.16], there exists a continuously differentiable function $V : D \to \mathbb{R}$ and three class-$\mathcal{K}$ functions α_1, α_2, α_3 defined on an open subset $D_0 \subseteq D$ such that

$$\begin{aligned}&(\forall x_1 \in D_0)\ \alpha_1(\|x_1\|) \leq V(x_1) \leq \alpha_2(\|x_1\|)\\ &(\forall x_1 \in D_0)\ L_{f_{CL}}V \leq -\alpha_3(\|x_1\|).\end{aligned}$$

Since $x_2 = k(x_1)$ is a good choice for the virtual control to the slow dynamics, it is natural to define an error coordinate $e \coloneqq x_2 - k(x_1)$. In (x_1, e)-coordinates system (2.10), (2.11), reads

$$\begin{aligned}\dot{x}_1 &= f(x_1) + g(x_1)(k(x_1) + e) = f_{CL}(x_1) + g(x_1)e\\ \dot{e} &= u - \dot{k}(x_1) = u - \frac{\partial k}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right).\end{aligned}$$

We now construct a control Lyapunov function $W(x_1, e)$ for this system using $V(x_1)$ that will yield the stabilizing controller for the entire system. Let

$$W(x_1, e) \coloneqq V(x_1) + \frac{1}{2}e^2 \tag{2.12}$$

and note that W is positive definite. Then, in a sufficiently small neighbourhood of $(x_1, e) =$

$(0, 0)$,

$$\begin{aligned}\frac{\mathrm{d}W}{\mathrm{d}t} &= \frac{\partial V}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right) + e\left(u - \frac{\partial k}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right)\right) \\ &= L_{F_{CL}}V + e\left(\frac{\partial V}{\partial x_1}g(x_1) + u - \frac{\partial k}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right)\right) \\ &\le -\alpha_3(\|x_1\|) + e\left(\frac{\partial V}{\partial x_1}g(x_1) + u - \frac{\partial k}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right)\right).\end{aligned}$$

We now select the control u to cancel out the sign indefinite terms and yield a negative definite derivative for W. To this end select

$$u = \frac{\partial k}{\partial x_1}\left(f_{CL}(x_1) + g(x_1)e\right) - \frac{\partial V}{\partial x_1}g(x_1) - K_e e, \qquad K_e > 0. \tag{2.13}$$

With this choice we have that

$$\frac{\mathrm{d}W}{\mathrm{d}t} \le -\alpha_3(\|x_1\|) - K_e e^2.$$

Hence, by Lyapunov's direct method [32, Theorem 4.1], $(x_1, e) = (0, 0)$ is an asymptotically stable equilibrium. Since $(x_1, e) = (0, 0)$ if and only if $(x_1, x_2) = (0, 0)$, the controller (2.13) asymptotically stabilizes the origin of (2.10), (2.11).

2.4.2 Backstepping with uncertainty

The control law (2.13) relies on perfect knowledge of the vector fields $f(x_1)$, $g(x_1)$ and the virtual feedback $k(x_1)$ in order to cancel the sign indefinite terms. In [34] the authors discuss how to robustify the controller. First we assume that the model uncertainty in (2.10) appears as

$$\dot{x}_1 = f(x_1) + g(x_1)\left(x_2 + \Phi(x_1, x_2)^\top \delta(x_1, x_2, u)\right) \tag{2.14}$$

where $\Phi : D \times \mathbb{R} \to \mathbb{R}^n$ is a known smooth function and $\delta(x_1, x_2, u) \in \mathbb{R}^n$ is a vector of unknown functions that are uniformly bounded, with known bounds, for all values of x_1, x_2, u.

If the uncertainty satisfies the above assumptions, then the robust controller is

$$u = u_{\text{nom}}(x_1, e) - K_u \frac{\partial V}{\partial x_1} g(x_1)\|\Phi(x_1, x_2)\|^2, \quad K_u > 0 \tag{2.15}$$

where $u_{\mathrm{nom}}(x_1, e)$ is the nominal controller (2.13). The gain K_u is a tuning parameter used to dominate the effect of the disturbances. The controller (2.15), renders the system (2.14), (2.11) input-to-state stable with respect to the uncertainty $\delta(x_1, x_2, u)$ [34, Lemma 2.26], and, when $\Phi(x_1, x_2) = 0$, i.e., there is no uncertainty, the controller reduces to u_{nom} and asymptotically stabilizes the origin.

2.4.3 Lyapunov redesign

Consider a nonlinear control system system of the form

$$\dot{x} = f(x) + g(x)\left(u + \delta(x, u)\right), \tag{2.16}$$

where $u \in \mathbb{R}$, and $f, g : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ are defined on an open and connected set D containing the origin with $f(0) = 0$. The term $\delta : D \times \mathbb{R} \to \mathbb{R}$ in (2.16) represents modelling uncertainty.

Assume that there exists a feedback control $u = \psi(x)$ that renders the origin asymptotically stable for the nominal closed-loop system

$$\dot{x} = f(x) + g(x)\psi(x) =: f_{CL}(x).$$

Once again, by the converse Lyapunov theorems [32, Theorem 4.16], there exists a continuously differentiable function $V : D \to \mathbb{R}$ and three class-$\mathcal{K}$ functions α_1, α_2, α_3 defined on an open subset $D_0 \subseteq D$ such that

$$\begin{aligned} &(\forall x_1 \in D_0) \;\; \alpha_1(\|x_1\|) \leq V(x_1) \leq \alpha_2(\|x_1\|) \\ &(\forall x_1 \in D_0) \;\; L_{f_{CL}} V \leq -\alpha_3(\|x_1\|). \end{aligned}$$

Now select the control law

$$u = \psi(x) + v, \tag{2.17}$$

where $v \in \mathbb{R}$ is an auxiliary controller to be designed. Now consider (2.16) with controller (2.17). The closed-loop system is

$$\dot{x} = f_{CL}(x) + g(x)\left(v + \delta(x, \psi(x) + v)\right). \tag{2.18}$$

Therefore

$$\begin{aligned} \frac{\mathrm{d}V}{\mathrm{d}t} &= L_{f_{CL}} V(x) + \frac{\partial V}{\partial x} g(x)\left(v + \delta(x, \psi(x) + v)\right) \\ &\leq -\alpha_3(\|x\|) + \frac{\partial V}{\partial x} g(x)\left(v + \delta(x, \psi(x) + v)\right). \end{aligned}$$

In the Lyapunov redesign approach, under suitable Lipschitz-like assumptions on the uncertainty term δ, we select v so that the sign indefinite terms above are dominated by $-\alpha_3(\|x\|)$. For this thesis, the design of v follows the methodology outlined in [17] and [32, Chapter 14].

2.5 Weak invariance and practical stability

Two important concepts that appear in the problem of Chapter 1 are the weak invariance of a set and practical stability. Consider a smooth, autonomous, unforced system

$$\dot{x} = f(x), \qquad f : \mathbb{R}^n \to \mathbb{R}^n, \qquad f(0) = 0. \tag{2.19}$$

Definition 2.5.1 ([35],[52])**.** The equilibrium $x = 0$ of (2.19) is **practically stable** if, for all (λ, α) with $0 < \lambda < \alpha$ we have that

$$(\forall x_0 \in B_\lambda(0)) \left(\exists T \in \mathbb{R}^+\right) (\forall t \geq T)\, x(t) \in B_\alpha(0).$$

Unlike Lyapunov stability, Definition 2.5.1 states that if the initial condition starts inside a ball of radius λ, the state $x(t)$ *eventually* enters a ball of radius α and stays there. Practical stability allows for a transient response that may take the solution of (2.19) far from the equilibrium before eventually returning.

Definition 2.5.2 ([35],[52])**.** The equilibrium $x = 0$ of (2.19) is **practically asymptotically stable** if it is practically stable and

$$(\forall \epsilon, \lambda > 0)\, (\exists T > 0)\, (\forall x_0 \in B_\lambda(0))\, (\forall t \geq T)\, x(t) \in B_\epsilon(0).$$

Definition 2.5.2 says that, not only does the state eventually enter the ball of radius ϵ, but that ϵ can be chosen as small as desired. Lastly, we define weak invariance of sets for switched systems. A switched system is a finite family of vector fields expressed as

$$\dot{x} = f(x), \qquad f \in \mathscr{F} = \{f_1(x), \ldots, f_N(x)\} \tag{2.20}$$

where each vector field $f_i : \mathbb{R}^n \to \mathbb{R}^n$ in the set $\mathcal{F}$ is smooth [2].

A solution for (2.20) is any continuous, not necessarily smooth, curve $x(t)$, with the property that there exists, a possibly divergent, sequence of times $\tau_0 = 0 < \tau_1 < \tau_2 < \dots$ called switching times, and a series of indices $i_0, i_1, i_2, \dots$ with $i_k \in \{1, \dots, N\}$, such that $x(t)$ is an integral curve of the vector field f_{i_k} for $t \in (\tau_k, \tau_{k+1})$.

Definition 2.5.3 ([2])**.** A set $P \subset \mathbb{R}^n$ is **weakly invariant** under the dynamics (2.20) if, for all $x_0 \in P$, there exists $i \in \{1, \dots, N\}$, a real number $b > 0$ such that the solution $x(t)$ to

$$\dot{x} = f_i(x), \qquad x(0) = x_0$$

belongs to P for all t in, either, $[-b, 0]$ or $[0, b]$.

Chapter 3

Polytope stabilization

In this chapter we propose a switched feedback control law that, under suitable assumptions, practically asymptotically stabilizes a polytope set. We start by finding a controller that stabilizes a single affine subspace. This controller is extended to the polytope case by adding affine subspace selection and switching algorithms. The resulting controllers make the polytope weakly invariant for the closed-loop system. Finally, issues with modelling uncertainty and robustness of the solution are addressed.

3.1 Stabilization of an affine subspace

Consider the Euler-Lagrange system (1.1), equivalently (1.3), with output (1.2). Let

$$\mathcal{A} = \{y \in \mathbb{R}^p : Ay + b = 0\},$$

with $A : \mathbb{R}^p \to \mathbb{R}^q$ onto, be a given affine subspace. Let $Q : \mathbb{R}^p \to \operatorname{Ker}(A)$ be the natural projection on $\operatorname{Ker}(A)$ along any complementary subspace $\mathcal{W} \subset \mathbb{R}^p$. Define the function

$$\begin{aligned} S :& \mathbb{R}^p \to \mathbb{R}^p \\ y \mapsto& \begin{bmatrix} Ay + b \\ Qy \end{bmatrix}. \end{aligned} \tag{3.1}$$

The following preliminary result is conceptually similar to [28, Theorem 3.2].

Theorem 3.1.1. *Consider the Euler-Lagrange system (1.1), equivalently (1.3), with $m \geq p$ inputs and with output*

$$\hat{y} = S \circ h(x_c) \tag{3.2}$$

where $h : \mathbb{R}^N \to \mathbb{R}^p$ is given by (1.2) *and $S : \mathbb{R}^p \to \mathbb{R}^p$ is given by* (3.1). *Let* $x^\star = \operatorname{col}(x_c^\star, x_v^\star) \in \mathbb{R}^N \times \mathbb{R}^N$ *be such that $h(x_c^\star) \in \mathcal{A} \cap W$ where W is a functional configuration space. The system yields a vector relative degree of $\{2, \ldots, 2\}$ at $x^\star$ if and only if*

$$\operatorname{Im}\left(\mathrm{d}h_{x_c^\star} g_v(x_c^\star)\right) + \operatorname{Ker}\left(\mathrm{d}S_{h(x_c^\star)}\right) \simeq \mathbb{R}^p. \tag{3.3}$$

Proof. Assume that (3.3) holds at $x^\star$. We begin by showing that the virtual output, $\hat{y}$, yields a well defined relative degree at $x^\star$ by showing the decoupling matrix is full rank at $x^\star$. We have

$$\begin{aligned}\frac{\mathrm{d}\hat{y}}{\mathrm{d}t} &= \begin{bmatrix} \frac{\partial S \circ h(x_c)}{\partial x_c} & 0_{1\times N}\end{bmatrix} \left(\begin{bmatrix} x_v \\ f_v(x) \end{bmatrix} + \begin{bmatrix} 0_{N\times m} \\ g_v(x_c) \end{bmatrix} u \right) \\ &= \frac{\partial S \circ h(x_c)}{\partial x_c} x_v =: L_f(S\circ h)(x).\end{aligned}$$

The control input u does not appear in this expression, $L_g(S \circ h)(x) \equiv 0$, so we continue differentiating. We obtain

$$\frac{\mathrm{d}^2\hat{y}}{\mathrm{d}t^2} = L_f^2(S\circ h)(x) + L_g L_f (S\circ h) u$$

where, using the chain rule,

$$\begin{aligned} L_g L_f(S\circ h) &= \left.\frac{\partial S}{\partial y}\right|_{y=h(x_c)} \frac{\partial h}{\partial x_c} g_v(x_c) && (3.4)\\ &= \begin{bmatrix} A \\ Q \end{bmatrix} \frac{\partial h}{\partial x_c} g_v(x_c). && (3.5)\end{aligned}$$

By Lemma 2.2.7 and since $h(x^\star)$ belongs to a functional workspace, the Jacobian of (3.2) is non-singular. This, together with (3.3), means that the hypothesis of Proposition 2.2.11 hold and that the decoupling matrix is full-rank.

Conversely, assume that the virtual output yields a well-defined relative degree of $\{2, \ldots, 2\}$ at $x^\star \in \mathbb{R}^n$. By definition, this means that (3.5) is onto. Hence, by Proposition 2.2.10 with $A = \mathrm{d}S_{h(x_c^\star)}$ and $B = \mathrm{d}h_{x_c^\star} g_v(x_c^\star)$, we conclude that (3.3) holds. □

Theorem 3.1.1 can be used to find a coordinate and feedback transformation that converts the Euler-Lagrange system (1.3) into a form that makes stabilizing $\mathcal{A}$ and controlling the motion along $\mathcal{A}$ easier.

Let $x^\star$ be a point at which the conditions of Theorem 3.1.1 hold. Then, in a neighbourhood $U_c \times U_v \subseteq \mathbb{R}^N \times \mathbb{R}^N$ of $x^\star$ there exists a smooth coordinate transformation (diffeomorphism onto its image) of the form

$$\begin{aligned} T : & U_c \times U_v \to T(U_c \times U_v) \subseteq \mathbb{R}^N \times \mathbb{R}^N \\ & x \mapsto \begin{bmatrix} \xi_1 \\ \xi_2 \\ \eta_1 \\ \eta_2 \\ \zeta \end{bmatrix} = \begin{bmatrix} Ah(x_c) + b \\ A\mathrm{d}h_{x_c} x_v \\ Qh(x_c) \\ Q\mathrm{d}h_{x_c} x_v \\ \phi(x) \end{bmatrix}. \end{aligned} \tag{3.6}$$

Here $\xi_1, \xi_2 \in \mathbb{R}^{p-q}$ represent the dynamics transversal to the target affine subspace. The $\eta_1, \eta_2 \in \mathbb{R}^q$ states represent the dynamics which result in motion tangential to the affine subspace. The $\zeta \in \mathbb{R}^{2N-2p}$ states represent any redundant degrees-of-freedom. For example, in a robotic manipulator, they could represent the orientation of the end-effector which does not determine whether or not the output is on the set $\mathcal{A}$. By [32, Theorem 13.1], the function $\phi : \mathbb{R}^n \to \mathbb{R}^{2N-2p}$ can be selected so that T is a diffeomorphism when restricted to a sufficiently small neighbourhood of $x^\star$.

Let $N(x)$ represent a basis for the kernel of the decoupling matrix

$$\begin{bmatrix} L_g L_f (Ah(x_c) + b) \\ L_g L_f \left(Qh(x_c)\right) \end{bmatrix} = \begin{bmatrix} A\frac{\partial h}{\partial x_c} \\ Q\frac{\partial h}{\partial x_c} \end{bmatrix}$$

and consider the feedback

$$u = \left(\begin{bmatrix} \begin{bmatrix} A\frac{\partial h}{\partial q} \\ Q\frac{\partial h}{\partial q} \end{bmatrix}^+ & N(x) \end{bmatrix} \right) \left(- \begin{bmatrix} L_f^2(Ah(x_c) + b) \\ L_f^2(Qh(x_c)) \\ 0 \end{bmatrix} + \begin{bmatrix} v^\pitchfork \\ v^\parallel \\ v^\zeta \end{bmatrix} \right). \tag{3.7}$$

By Theorem 3.1.1, the feedback (3.7) is well-defined in a neighbourhood of $x^\star$, without loss of generality, $U_c \times U_v$. Under this feedback transformation, the system in (ξ, η, ζ) -coordinates reads

$$\begin{aligned} \dot{\xi}_1 &= \xi_2 \\ \dot{\xi}_2 &= v^\pitchfork \\ \dot{\eta}_1 &= \eta_2 \\ \dot{\eta}_2 &= v^\parallel \\ \dot{\zeta} &= f_\zeta(\xi, \eta, \zeta) + g^\pitchfork(\xi, \eta, \zeta) v^\pitchfork + g^\parallel(\xi, \eta, \zeta) v^\parallel + g_\zeta(\xi, \eta, \zeta) v^\zeta \end{aligned} \tag{3.8}$$

where $v^{\pitchfork} \in \mathbb{R}^{p-q}$, $v^{\parallel} \in \mathbb{R}^{p}$ and $v^{\zeta} \in \mathbb{R}^{m-p}$. In transformed coordinates the target affine subspace is given by

$$T(h^{-1}(\mathcal{A} \cap W)) = \{(\xi, \eta, \zeta) : \xi_1 = 0\}. \tag{3.9}$$

Remark 3.1.2. The choice of $\phi(x)$ to complete the coordinate transformation T can often be done using physical intuition. The redundant dynamics of the system are given by

$$\dot{\zeta} = f_{\zeta}(0, 0, \zeta) + g_{\zeta}(0, 0, \zeta)v^{\zeta}.$$

It may be possible, if $m > p$, to use the extra control inputs v^{ζ} to smartly control these dynamics. In general, these dynamics have no special structure and the design of $v^{\pitchfork}$ may be challenging.

We call the input $v^{\pitchfork}$ the transversal input because it affects the behaviour of the transversal states $\xi = (\xi_1, \xi_2)$. Since driving the transversal states to zero corresponds to, assuming bounded trajectories, driving the output to $\mathcal{A} \cap W$, we seek a stabilizing controller. It is readily checked that the transversal dynamics are controllable so we use a linear feedback

$$v^{\pitchfork} = -K_1\xi_1 - K_2\xi_2, \tag{3.10}$$

with the gains $K_1, K_2 \in \mathbb{R}^{(p-q)\times(p-q)}$ selected so that the eigenvalues of

$$\begin{bmatrix} 0_{(p-q)\times(p-q)} & I_{(p-q)} \\ -K_1 & -K_2 \end{bmatrix} \tag{3.11}$$

are in the open left-half complex plane. The gains can be selected using, for example, linear quadratic regulation (LQR) optimization. The above transversal controller renders the origin of the transversal dynamics exponentially stable.

Proposition 3.1.3. *Suppose that the conditions of Theorem 3.1.1 hold at $x^{\star}$. Let the feedback control be given by (3.7) and (3.10) so that (3.11) is Hurwitz. Let $U_c \times U_v \subseteq \mathbb{R}^N \times \mathbb{R}^N$ be the open set where (3.6) is a diffeomorphism. Let $v^{\parallel}$ and v^{ζ} be such that, for all $(x_c(0), x_v(0)) \in U_c \times U_v$, and all $t \geq 0$, the solution $(x_c(t), x_v(t)) \in U_c \times U_v$ and is bounded. Then $\xi = 0$ is exponentially stable and $y(t) = h(x_c(t))$ approaches $\mathcal{A} \cap W$ as $t \to \infty$.*

Proof. The hypothesis of the proposition guarantee that, for all $(x_c(0), x_v(0)) \in U_c \times U_v$, the Euler-Lagrange system is feedback equivalent to (3.8) for all $t \geq 0$. Since (3.11) is Hurwitz, the origin of the transversal dynamics is exponentially stable. Finally, since the trajectories are bounded, we have that $\xi_1 \to 0 \Leftrightarrow y \to \mathcal{A} \cap W$. □

We end this section by showing that the set $\mathcal{A}$ is output invariant.

Proposition 3.1.4. *Suppose that the conditions of Theorem 3.1.1 hold at $x^\star$. Let the feedback control be given by (3.7) and (3.10) so that (3.11) is Hurwitz. Then the set*

$$\mathcal{A}^\star := \left\{ x \in h^{-1}(W) \cap \mathbb{R}^{2N} : Ah(x_c) + b = A\mathrm{d}h_{x_c} x_v = 0 \right\} \tag{3.12}$$

is controlled invariant.

Proposition 3.1.4 says that if the output is initialized on $\mathcal{A}$ with the rate change of the output tangent to $\mathcal{A}$, then the output remains on $\mathcal{A}$ as long as the feedback transformation remains well-defined.

Proof. Let $x_0 \in \mathcal{A}^\star$. By the definition of the coordinate transformation (3.6) we have that $\xi_1(x_0) = \xi_2(x_0) = 0$. Since $\xi = 0$ is an equilibrium is for the closed-loop system, the result follows immediately. $\square$

3.2 Polytope surface stabilization

The surface of a polytope can be viewed, as discussed in Section 2.3, as the union of its facets. Let $\{\mathcal{F}_1, \ldots, \mathcal{F}_M\}$ be the facets of P. Let

$$\mathcal{F} := \bigcup_{i}^{M} \mathcal{F}_i. \tag{3.13}$$

We refer to $\mathcal{F}$ as the surface of the polytope P. To stabilize the surface, our approach is to select a single facet $\mathcal{F}_i$ of the polytope, stabilize that facet using the controllers from Section 3.1, and then switch the target facet when needed. The surface stabilization algorithm is encapsulated in Figure 3.1.

The selection of a facet is accomplished with a switching function

$$\mathcal{I} : \mathbb{R}^N \times \mathbb{R}^N \to \{1, \ldots, M\}\,.$$

Algorithm 3.2.1. *Given the state x of (1.3), the last switch time t_{last}, the current time t and the current facet i_{last}, select the index i corresponding to facet $\mathcal{F}_i$ of polytope P for stabilization*

```
1: if ( then $t - t_{last} > t_d$)
2:     for $j = 1$ to $M$ do
```

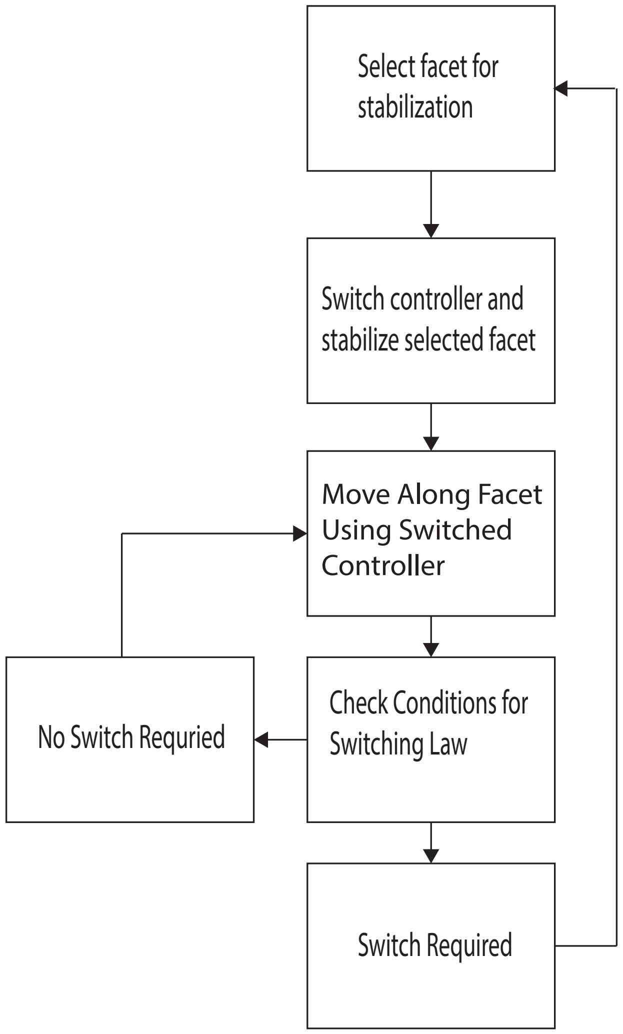

Select facet for stabilization
Switch controller and stabilize selected facet
Move Along Facet Using Switched Controller
No Switch Requried
Check Conditions for Switching Law
Switch Required

Figure 3.1: Polytope Stabilization Algorithm

```
3:      Calculate coarse distance to each facet using $\xi_1$
4:      ($d_{min}$ = Shortest distance to facet)
5:   end for
6:   if $d_{min} > \epsilon$ then
7:      Output is far away from polytope
8:      Invoke Algorithm B.1.1 or Algorithm B.2.1
9:       $i$ = results of Algorithm B.1.1 or Algorithm B.2.1
10:  else
```

```
11:         Output is close to surface of polytope
12:         Invoke Algorithm 3.2.3
13:         $i$ = results of Algorithm 3.2.3
14:     end if
15: else
16:     Facet is unchanged $i = i_{last}$
17: end if
```

The parameter ϵ is a user defined variable, that allows for some measure of tuning based on the geometry of the polytope. Furthermore, the parameter t_d, represents a user defined dwell time. This choice of dwell time is further discussed in Section 3.2.2. The various subroutines in Algorithm 3.2.1 will be discussed shortly.

We now turn our attention to the controllers used to stabilize the facet $\mathcal{F}_i$ of P. Extending the controller outlined in (3.7), we modify the virtual output to the following

$$\begin{aligned} S_i :& \mathbb{R}^p \to \mathbb{R}^p \\ y \mapsto & \begin{bmatrix} A_i y + b_i \\ Q_i y \end{bmatrix} \end{aligned} \tag{3.14}$$

where $i \in \{1, \ldots, M\}$, A_i, b_i and Q_i each correspond to the ith facet. The auxiliary control inputs $v_i^{\pitchfork}, v_i^{\parallel}, v_i^{\zeta}$ are also defined on the ith facet, and $N_i(x)$ is the basis for the kernel of $L_g L_f \left(A_i h(x_c) + b_i\right)$.

We define a finite set of controllers $\mathcal{U} = \{\mathrm{u}_1, \ldots, \mathrm{u}_M\}$ of the form

$$\mathrm{u}_i = \left(\begin{bmatrix} \begin{bmatrix} A_i \frac{\partial h}{\partial q} \\ Q_i \frac{\partial h}{\partial q} \end{bmatrix}^+ & N(x)_i \end{bmatrix}\right) \left(- \begin{bmatrix} L_f^2(A_i h(x_c) + b_i) \\ L_f^2(Q_i h(x_c)) \\ 0 \end{bmatrix} + \begin{bmatrix} v_i^{\pitchfork} \\ v_i^{\parallel} \\ v_i^{\zeta} \end{bmatrix} \right) \tag{3.15}$$
$$i \in \{1, \ldots, M\}.$$

Each controller u_i in the set $\mathcal{U}$ stabilizes, under the assumptions of Proposition 3.1.3, the defining the affine subspace

$$\mathcal{A}_i = \{y \in \mathbb{R}^p : A_i y + b_i = 0\}.$$

3.2.1 Affine subspace selection algorithm

A key component of Algorithm 3.2.1 is to compute the distance of the system output y to the surface $\mathcal{F}$ of the polytope P. To avoid heavy computations, we initially derive a coarse estimate of the distance from y to the polytope by evaluating ξ_1. This calculation is inaccurate, as it does not take the ridges of the facets into account. However, it does provide for a gross estimate of the distance to determine which subspace selection algorithm to use. To obtain an accurate measure of the distance we must invoke more advanced algorithms. We now discuss two different approaches for implementing this calculation. The first method is conceptually simple, but is not suited to higher dimensions. The second is well-suited to higher dimensions but requires faster hardware.

Closest distance by projection

An algorithm for determining the closest distance from a point to a polytope in $\mathbb{R}^p$ is outlined in [22]. By recursively applying this algorithm to each individual facet of the polytope, we are able to determine the distance to the surface of the polytope. From that minimum distance, we are able to select the closest facet to stabilize.

Although the algorithm outlined in [22] is a capable of dealing with surfaces in $\mathbb{R}^p$, there exist many practical simplifications that can be made by restricting ourselves to surfaces in $\mathbb{R}^3$. These simplifications allow for dramatic decreases in computation time, and a reduction in overall complexity.

A simplified algorithm for plane selection comes from [30]. In [30], an algorithm based on well known vector properties in $\mathbb{R}^3$ is applied to triangles. However, this algorithm can be extended to more complex polytopes in $\mathbb{R}^3$. It should be noted that although we generalize this result to a larger class of polytopes, we are still bound to $\mathbb{R}^3$ due to the use of the cross product in the algorithm. This algorithm and an illustrative example can be found in Appendix B.1.

The algorithm presented in Appendix B.1 for $\mathbb{R}^3$ is substantially less computationally intensive in comparison to the algorithm presented in [22]. As outlined in [22], the order of the presented algorithm for $\mathbb{R}^p$ is $O(n^4)$, which would then need to be repeated k times. However, the algorithm for $\mathbb{R}^3$ is $O(n\log(n))$, as it is dominated by the calculations required to determine the orientation of the point relative to the polytope, and is also repeated k times.

Closest distance by convex optimization

Since polytopes are convex sets, computing the distance from y to $\mathcal{F}$ can be naturally cast as a convex optimization problem. There are many pre-built convex optimization solvers that have been refined to provide optimal computational efficiency, that can be easily implemented. However, the projection methods above have been presented for instances where there may be limitations preventing the implementation of a convex optimization method. The implementation for determining the distance to a polytope via convex optimization can be found in Appendix B.2.

3.2.2 Switching conditions along surface

The subspace selection algorithm outlined in Section 3.2.1 provides a clear choice of closest facet when the distance between the system output and the polytope is large. However, the switching conditions become less clear when the output is closer to the surface ($d_{min} \leq \epsilon$). For example, if the system output lies on the intersection of two or more facets, the distance to each facet is close to zero, and the distance condition alone cannot be used to select a particular facet. In this section we detail the operation of Algorithm 3.2.1 when $d_{min} \leq \epsilon$, i.e., the ***else*** state of Algorithm 3.2.1.

The idea behind this aspect of the algorithm is simple : when the distance from the output to facet i and facet j is less than ϵ, select the facet based on all of the transversal states, rather than the distance alone.

To implement our switching law, we need the notion of a vector being restricted to a polytope. A vector is restricted to a polytope if its motion does not force the system to leave the boundary of the polytope. Adopting the approach shown in [45], we introduce the following lemma to check this condition.

Lemma 3.2.2. *Given a vector field $\frac{\partial h(x)}{\partial x_v} x_v$ at a point v_i along ridge $\mathcal{R}_{ij}$ of a polytope, a vector is restricted to the polytope if*

$$(\forall j \in \mathcal{J}) \left\langle A_j, \ \frac{\partial h(x)}{\partial x_c} x_v \right\rangle \leq 0$$

where A_j comes from the zero level set representation of the jth facet, and $\mathcal{J}$ represents the set of all facets intersecting ridge $\mathcal{R}_{ij}$.

With this lemma to check if a vector is restricted to a polytope, our switching law can determine which of several subcases the system may be in.

Let our system be at a point v_i on ridge $\mathcal{R}_{ij}$ of polytope P. This point is the intersection of several facets and the indicies of these facets are contained in the set $\mathcal{J}$. If the system output $h(x)$ is restricted to the polytope (i.e. its motion would drive it into the polytope), then we seek to choose the facet where the transversal states are the smallest. This choice ensures the smoothest motion along the surface, as the system is likely already weakly invariant at this point. By choosing the facet with the smallest transversal states, the weak invariance is more likely preserved, as the system is not being driven away from the polytope. If instead the motion would drive the output away from the polytope, then we choose the facet where the traversal states are largest. This choice comes from the fact that the motion of the system is being directed towards another facet, indicating that a switch is necessary. In this situation, $h(x)$ will not be invariant to any particular facet for an instance of time, but given sufficient conditions, can be within a distance away from each facet.

Algorithm 3.2.3. *Given state x of system (1.3), determine index i of facet $\mathcal{F}_i$ to stabilize, if the point to set distance $\|y\|_P$ between the system output $h(x)$ and polytope P is less than ϵ. Let $\mathcal{J}$ be the set of all indices of facets where the point to set distance between $h(x)$ and P is less than ϵ.*

```
 1: if $|\mathcal{J}| = 1$ then
 2:     $i = \mathcal{J}$
 3: else
 4:     if $\left(\frac{\partial h(x)}{\partial x_v} x_v \text{is restricted to the polytope}\right)$ then
 5:         $\mathcal{T} = \arg\min_j \left\| \begin{bmatrix} Q_j \frac{\partial h(x)}{\partial x_v} x_v \\ A_j h(x) + b_j \end{bmatrix} \right\|, j \in \mathcal{J}$
 6:         $i = \min \mathcal{T}$
 7:     else
 8:         $i = \arg\max_j \left\| \begin{bmatrix} Q_j \frac{\partial h(x)}{\partial x_v} x_v \\ A_j h(x) + b_j \end{bmatrix} \right\|, j \in \mathcal{J}$
 9:     end if
10: end if
```

The above switching scheme may result in infinite switching along the edge. To avoid this, a small dwell time is implemented between updates of the selected plane. A systematic approach for selecting the dwell time is the subject of future research. For the purposes of this thesis, it was selected in an ad-hoc manner by selecting an initial estimate and iterating the dwell time to improve performance.

3.3 Weak invariance of surface set

We now show that our choice of control law (3.15), and the switching law $\mathcal{I}(x)$ defined by Algorithm 3.2.1, render the surface (3.13) of P weakly invariant, see Definition 2.5.3.

Using control laws (3.15), we have the following family of vector fields

$$\mathscr{F} = \{f_1(x), \ldots, f_M(x)\}, \ f_i(x) \coloneqq f(x) + g(x)\mathrm{u}_i. \tag{3.16}$$

In light of Proposition 3.1.4, let

$$\mathcal{A}_i^\star \coloneqq \left\{x \in h^{-1}(W) \cap \mathbb{R}^{2N} : A_i h(x_c) + b_i = A_i \mathrm{d}h_{x_c} x_v = 0\right\}, \ i \in \{1, \ldots, M\} \tag{3.17}$$

and define

$$\mathbf{A} \coloneqq \cup_{i=1}^M \mathcal{A}_i^\star. \tag{3.18}$$

Lemma 3.3.1. *The set* (3.18) *is weakly invariant with respect to* (3.16)

Proof. Given $x_0 \in \mathbf{A}$. We decompose the proof into two cases. The first case is where x_0 lies in the interior of a facet, the second where it lies on the intersection of two or more facets.

We begin by considering the first case, where

$$x_0 \in \mathrm{int}(\mathcal{A}_i^\star)$$

Let $x(t)$ denote the solution to $\dot{x} = f_i(x)$, with $x(0) = x_0$. By Proposition (3.1.4) we have that $\mathcal{A}_i^\star$ is invariant for this system which immediately shows that it is weakly invariant. Specifically, there exists a positive constant b such that

$$(\forall t \in [-b, b]) \ x(t) \in \mathcal{A}_i^\star.$$

Next we look at the case where x_0 lies on the intersection of two or more facets of the polytope.

$$x_0 \in \mathcal{A}_i^\star \cap \mathcal{A}_j^\star, \ i, j \in \{1, \ldots, M\}, i < j.$$

We now consider two subcases based on the initial velocity of the system. The initial velocity will either bring (3.16) into the ith facet or out of the ith facet. These cases are distinguished using Lemma 3.2.2. If a vector is restricted to the facet, then the system is entering the facet. If the system is not restricted to the facet, then it is exiting the facet.

We begin with the first subcase with the velocity bringing the system out of the ith facet

$$A_i \frac{\partial h(x_c)}{\partial x_c}\bigg|_{x=x_0} x_v(0) = 0$$
$$A_j \frac{\partial h(x_c)}{\partial x_c}\bigg|_{x=x_0} x_v(0) > 0.$$

Since the initial condition x_0 corresponds to leaving the ith facet, the solution to $\dot{x} = f_i(x)$, $x(0) = x_0$ must belong to $\mathcal{A}_i^\star$ for sufficiently small negative time. That is, there exists a constant b such that $x(t) \in \mathcal{A}_i^\star$ for $t \in [-b, 0]$.

The next subcase involves the velocity bringing the system into the ith facet.

$$A_i \frac{\partial h(x_c)}{\partial x_c}\bigg|_{x=x_0} x_v(0) = 0$$
$$A_j \frac{\partial h(x_c)}{\partial x_c}\bigg|_{x=x_0} x_v(0) \leq .$$

The above condition implies that the solution $x(t)$ to $\dot{x} = f_i(x)$, $x(0) = x_0$ evolves along the ith facet for forward time. By Proposition 3.1.4 there exists a real constant b for which $x(t) \in \mathcal{A}_i^\star$ for $t \in [0, b]$. □

3.4 Modelling uncertainty

The presented algorithms provide a solution when exact model knowledge is available, but issues arise in implementation due to modelling uncertainties. These uncertainties introduce inaccuracies in the feedback linearization of the system, resulting in imperfect cancellation of nonlinearities. One common source of uncertainty in the system is that of unmodelled friction. To compensate for this, we introduce a friction model that includes many of the typical sources of friction, and then present a robust control technique to deal with the modelling uncertainty. The robust controllers are implemented in the transformed coordinates of the system. As a motivating example, simulation results are shown on a four degree of freedom robotic manipulator of the form seen in Figure 3.2.

3.4.1 Friction model

We present a friction model outlined in [14]. This model incorporates friction effects such as hysteresis, spring-like characteristics of stiction, break away forces and the Stribeck

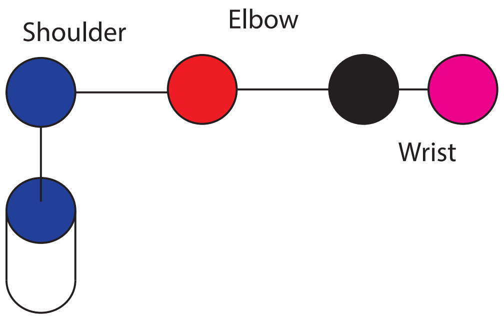


Figure 3.2: Arm Diagram

effect [15]. This model has been shown to accurately model actuator friction in robotic manipulators in [31]. Following the implementation method outlined in [31], we introduce the friction as an additional force in the Euler-Lagrange equations. This appears as follows

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = B(q)\left(\tau - \tau_f\right), \tag{3.19}$$

where τ_f represents the friction force. This friction force is modelled as a function of the joint velocity as follows

$$\dot{z} = \dot{q}_j - \frac{\sigma_0|\dot{q}_j|}{s(\dot{q}_j)} z \tag{3.20}$$

$$\tau_{F_j} = \sigma_1\dot{z} + \sigma_0 + \sigma_2\dot{q}_j, \tag{3.21}$$

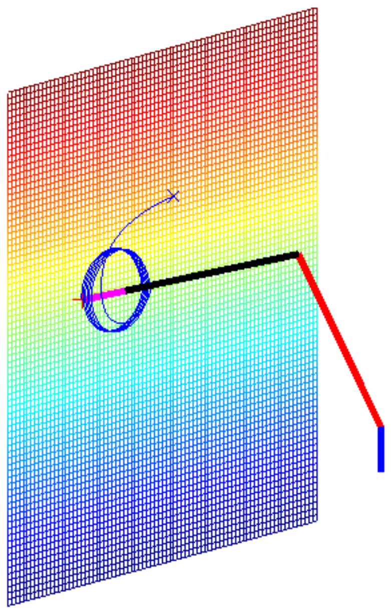
where j corresponds to the particular joint of interest, and $\sigma_0,\ \sigma_1$ are stiffness and damping coefficients, σ_2 viscous coefficient and z the average deflection of some fictitious bristles causing friction [31].

The scalar function $s(\dot{q}_j)$ can be defined as

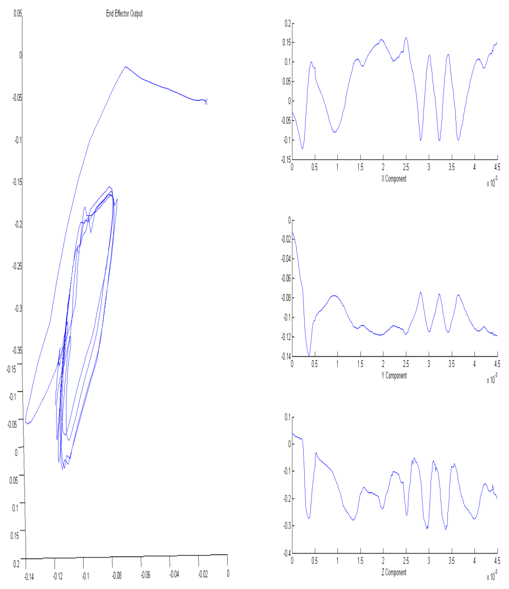
$$s(\dot{q}_j) = F_c + (F_s - F_c)e^{-\alpha|\dot{q}_j|}, \tag{3.22}$$

where F_c is the Coulomb force, F_s the stiction force and α determines the variation of $s(\dot{q}_j)$ between F_s and F_c [31]. Together, these equations form a representative model of friction.

The effects of friction are simulated and compared to experimental results. The simulation result reflects trends observed in similar experimental tests in Figure 3.3. Figure 3.3(a) shows a simulation with the friction model present, and Figure 3.3(b) shows an experiment with the same configuration. As the figures show, both in the simulation and the experiment, the end effector of the manipulator initially stabilizes the plane, but begins to drift away as time progresses, ultimately resulting in steady state error. Although the friction model was not tuned to identically match the simulation, it does provide some modeling of the friction effects.



(a) Simulation Results



(b) Experimental Results

Figure 3.3: Friction Effects on System output without Robust Control

3.4.2 Robust control techniques

To compensate for uncertainties such as friction or inaccuracy in friction identification, we introduce a robust control technique. See also Chapter 2.

Lyapunov redesign

Following the robust control algorithm presented in [20], we once again adapt our transformed coordinates to compensate for the unmodelled presence of friction and other factors.

In this implementation, we treat all the unknown disturbances as a single term $\Delta(\xi, \eta, u)$. If we simply use the transverse controller we designed for the nominal system the parameter uncertainties would appear as follows

$$\begin{aligned}\dot{\xi} &= A\xi + B\left(\alpha(\xi,\eta) + \Delta_\alpha(\xi,\eta) + \left(\beta(\xi,\eta) + \Delta_\beta(\xi,\eta)\right)u\right) \\ &=: A\xi + B\left(\alpha(\xi,\eta) + \beta(\xi,\eta)u\right) + \delta(\xi,\eta,u).\end{aligned}$$

As outlined in [20], we can thus use our control input to overcome the uncertainty term $\delta(\xi, \eta, u)$, by modifying the control input to become

$$u = (K + K_0)\xi + \begin{cases} K_1 \frac{\xi}{\|\xi\|} & : \|\xi\| \geq \mu > 0 \\ K_2\|\xi\|\xi & : \|\xi\| < \mu, \end{cases} \tag{3.23}$$

where $(A + BK)$ is Hurwitz, $K_0, K_1, K_2 \in \mathbb{R}^{n\times n}$ and $\mu \in \mathbb{R}$ is a constant that depends on bounding factors based on the uncertainty [20].

The robust version of this simulation can be seen in Figure 3.4. As this figure shows, the robust controller is able to overcome the effect of the modelling uncertainties, and track the defined path, with little modification to the existing control structure.

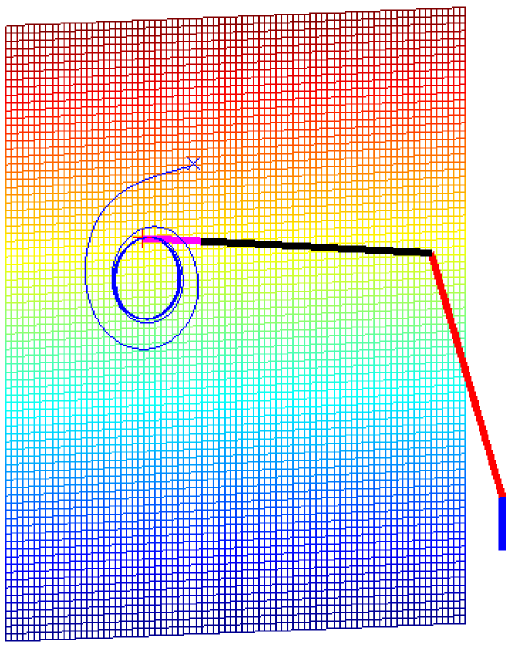


Figure 3.4: Friction Effects on System with Robust Control

Chapter 4

Control design on facets

This chapter introduces several methods to control the motion of a system along the facets of the target polytope. The techniques investigated include path following, damped dynamics along the surface and trajectory tracking. The solutions presented are derived for an arbitrary ith facet on the polytope P.

4.1 Tangential dynamics

Lemma 3.3.1 shows that, under the control set $\mathcal{U}$ given by (3.15) and the switching law $\mathcal{I}$ given by Algorithm 3.2.1, the surface of the polytope P is weakly invariant. The dynamics of (1.1) restricted to facet $i \in \{1, \dots, M\}$ are given by, dropping the subscript i,

$$\begin{aligned} \dot{\eta}_1 &= \eta_2 \\ \dot{\eta}_2 &= v^{\parallel} \\ \dot{\zeta} &= f_{\zeta}(0, \eta, \zeta) + g^{\parallel}(0, \eta, \zeta)v^{\parallel} + g_{\zeta}(0, \eta, \zeta)v^{\zeta}. \end{aligned} \tag{4.1}$$

As discussed in Section 3.1, the dynamics (4.1) model the dynamics of the Euler-Lagrange system while its output is restricted to evolve on facet i of the polytope P. Our objective in this chapter is to design M tangential controllers, one for each facet of P, such that the output of the system moves along the surface of the polytope in a desirable manner. We make the following simplifying assumption for the remainder of this Chapter.

Assumption 2. *The Euler-Lagrange system* (1.1) *satisfies* $N = m = p$.

Assumption 2 implies that the system has no redundant degrees-of-freedom. It means that the normal form (3.8) simplifies and that the tangential dynamics become linear and controllable

$$\begin{aligned}\dot{\eta}_1 &= \eta_2\\ \dot{\eta}_2 &= v^{\parallel}.\end{aligned} \tag{4.2}$$

Remark 4.1.1. Although we do not prove it here, it is straight forward to show that, under Assumption 2 and by selecting tangential controllers that make the origin $(\eta_1, \eta_2) = (0, 0)$ exponentially stable[1] on each facet, the surface $\mathcal{F}$ of the polytope P is practically asymptotically stable for the closed-loop system.

4.2 Path following

In this section we consider the problem of designing tangential controllers so that the output of the system follows a pre-defined path on the surface of the polytope P. We call this a path following problem on polytopes (PFPP). In a PFPP, the objective is not to be at a specific point on the path at a given time, but rather to traverse the path accurately while remaining on the path.

We begin by describing the type of path considered. Assume that we are given a continuous parameterized curve

$$\sigma : \mathbb{R} \to \mathcal{F}$$

which is periodic, i.e., there exists $L > 0$ such that, for all $\lambda \in \mathbb{R}$, $\sigma(\lambda + L) = \sigma(\lambda)$. Let $\mathcal{C} := \sigma([0, L))$ be the image of σ.

Assumption 3. *If $\mathcal{C} \cap \mathcal{F}_i \neq \varnothing$, $i \in \{1, \dots, M\}$, then $\mathcal{C} \cap \partial\mathcal{F}_i$ equals two distinct points.*

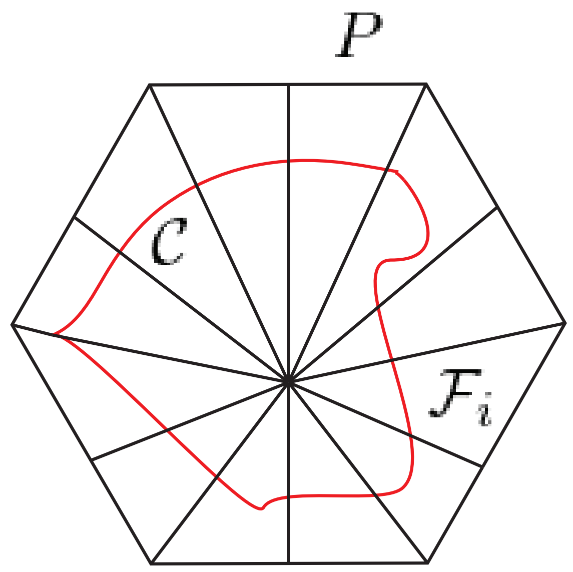
An illustration of a path that satisfies Assumption 3 is shown in Figure 4.1. The assumption means that the path can only enter and exit a particular facet once. It also disallows paths that are tangent to a ridge of the polytope P.

Partition the interval $[0, L) \subset \mathbb{R}$ into q half-open intervals

$$[0, L) = I_1 \cup I_1 \cup \cdots \cup I_q$$

where $I_j = [a_j, b_j) \subset \mathbb{R}$, $j \in \{1, \dots, q\}$, is a non-empty interval such that $\sigma(I_j) \subset \mathcal{F}_{i_j}$, $i_j \in \{1, \dots, M\}$. The number $q \leq M$ in the partition denotes the number of facets of P through which the curve $\mathcal{C}$ passes through. Given the interval $I_j = [a_j, b_j)$, let $\operatorname{int}(I_j) = (a_j, b_j)$ denote its interior. We assume that, for all $\lambda \in \operatorname{int}(I_j)$, σ is smooth.

[1]This is always possible since (4.2) is controllable.



.

Figure 4.1: Path defined on a polytope.

Problem 2. Given the set (3.13), M systems of the form (4.2) and a periodic curve $\sigma : \mathbb{R} \to \mathcal{F}$ that satisfies Assumption 3, find M tangential controllers such that, on each facet

(a) $y(t) = h(x_c(t)) \to \mathcal{C} \cap \mathcal{F}_i$ as $\to \infty$.

(b) The set $\mathcal{C} \cap \mathcal{F}_i$ is output invariant.

•

Remark 4.2.1. Problem 2 does not impose any constraints on path invariance when the controller switches across the boundaries of facets. Indeed, since the set $\mathcal{F}$ is only weakly invariant, and since $\mathcal{C} \subset \mathcal{F}$, it is not possible to impose path invariance across facets using the transversal controllers from Chapter 3.

To solve Problem 2 we use the approach of [28]. First, since for each $\lambda \in \mathbb{R}$, $\sigma(\lambda)$ is an element of $\mathbb{R}^p$, we apply the natural projection to the curve to get its local coordinates expression

$$(\forall i \in \{1, \dots, q\})\, (\forall \lambda \in I_i)\, \hat{\sigma}_i(\lambda) \coloneqq Q_i \sigma(\lambda).$$

The above definition of $\hat{\sigma}_i(\lambda)$ simply says that, if the path is passing through facet i, we apply the ith natural projection, the one corresponding to the ith facet. Recall that $\eta_1 = Q_i y = Q_i h(x_c))$. Next we introduce two functions based on $\hat{\sigma}_i$. First we assume that

$$(\forall i \in \{1, \ldots, q\})\, \hat{\sigma}(\mathrm{int}\,(I_i)) = \big\{\eta_1 \in U_i \subseteq \mathbb{R}^{p-q} : \rho_i(\eta_1) = 0\big\} \tag{4.3}$$

for some smooth function $\rho_i : U_i \subseteq \mathbb{R}^{p-q} \to \mathbb{R}^{p-q-1}$ defined on an open set U_i. In other words, we assume that the projection of that portion of the curve σ on $\mathcal{F}_i$ can be represented as the zero level set of a smooth function. Henceforth we drop the subscript i from the function $\rho_i(\eta_1)$.

Next we define M possibly nonlinear[2] projection functions

$$\begin{aligned} \varpi_i :& \mathbb{R}^{p-q} \to \mathrm{int}\,(I_i) \subset \mathbb{R} \\ &\eta_1 \mapsto \operatorname*{arg\,min}_{\lambda \in \mathrm{int}\,(I_i)} \|\eta_1 - \hat{\sigma}_i(\lambda)\|. \end{aligned} \tag{4.4}$$

We again drop the subscript on ϖ. By [28, Theorem 3.2], the tangential dynamics (4.2) with output $(\rho(\eta_1), \varpi(\eta_1))$ yield a relative degree of $\{2, \ldots, 2\}$ at each point on $\hat{\sigma}(\lambda)$ and is therefore feedback linearizable. To feedback linearize we use the coordinate transformation

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \upsilon_1 \\ \upsilon_2 \end{bmatrix} = \begin{bmatrix} \rho(\eta_1) \\ \frac{\partial \rho}{\partial \eta_1}\eta_2 \\ \varpi(\eta_1) \\ \frac{\partial \varpi}{\partial \eta_1}\eta_2 \end{bmatrix},$$

and the feedback transformation

$$v^{\parallel} = \begin{bmatrix} \frac{\partial \rho}{\partial \eta_1} \\ \frac{\partial \varpi}{\partial \eta_1} \end{bmatrix}^{-1} \left(\begin{bmatrix} \frac{\partial}{\partial \eta_1}\left(\frac{\partial \rho}{\partial \eta_1}\eta_2\right)\eta_2 \\ \frac{\partial}{\partial \eta_1}\left(\frac{\partial \varpi}{\partial \eta_1}\eta_2\right)\eta_2 \end{bmatrix} + w \right) \tag{4.5}$$

where $w = (w_1, \ldots, w_{p-q})$. In (ϵ, υ)-coordinates the tangential dynamics are

$$\begin{aligned} \dot{\epsilon}_1 &= \epsilon_2 \\ \dot{\epsilon}_2 &= \mathrm{col}\,(w_1, \ldots, w_{p-q-1}) \\ \dot{\upsilon}_1 &= \upsilon_2 \\ \dot{\upsilon}_2 &= w_{p-q}. \end{aligned} \tag{4.6}$$

[2]Since the path itself may be nonlinear.

Physically, the $(\epsilon_1, \epsilon_2) \in \mathbb{R}^{p-q-1} \times \mathbb{R}^{p-q-1}$ states represent motion that is tangent to facet $\mathcal{F}_i$ but transversal to $\mathcal{C} \cap \mathcal{F}_i$. The $(\upsilon_1, \upsilon_2) \in \mathbb{R} \times \mathbb{R}$ represent those dynamics that are tangent to both $\mathcal{F}_i$ and $\mathcal{C} \cap \mathcal{F}_i$.

To design our controllers, we apply the above coordinate transformation to the q sets of tangential dynamics associated with a facet through which the path passes through. An obvious question arises from this construction : what should we do for the $M - q$ facets through which the path does not pass through?

To properly function, this approach must have a solution for the cases where the selected plane does not have a path passing through it. In these cases, the point to set distance between the end effector output and the various path sections is found. This calculation is relatively simple since the path sections are defined in $\mathbb{R}^p$ and can be quickly computed. Once the closest path section is determined, a rudimentary path from the current location to the desired path is calculated using Dijkstra's algorithm, the polytope graph $\mathscr{G}$, and the verticies of the polytope.

Dijkstra's algorithm determines which facets the system output must pass through to reach the closest facet with a path defined. This target facet is reached by defining way points in the intermediate facets between the current facet and the target, and then stabilized using a point stabilization controller. These way points are placed on the ridges between the intermediate facets, and are defined as the midpoint between the vertices defining the ridge. By defining the way points on the ridges of the facets, the switching algorithm is invoked every time a way point is stabilized, and the rudimentary path updated if required. Once the target facet has been stabilized, the nested transverse feedback linearization controller can be implemented. Dijkstra's algorithm is explained in further detail in Appendix. A.

Example 4.2.1. Given a 4 DOF robotic manipulator of the form (1.3), we aim to stabilize a cube defined as

$$\begin{aligned} P &:= \operatorname{conv}\{\upsilon_0, \ldots, \upsilon_8\}, \\ &= \operatorname{conv}\left\{ \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}2\\1\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\end{bmatrix}, \begin{bmatrix}2\\2\\1\end{bmatrix}, \begin{bmatrix}1\\1\\2\end{bmatrix}, \begin{bmatrix}2\\1\\2\end{bmatrix}, \begin{bmatrix}1\\2\\2\end{bmatrix}, \begin{bmatrix}2\\2\\2\end{bmatrix} \right\}, \end{aligned}$$

and then move to the bottom facet and stabilize a circle defined on the bottom facet. The circle is centred at the middle of the bottom facet, with radius of 0.4.

Once the circle has been stabilized by the end effector of the robot, it will continue to follow trace the circle at a constant user defined speed of $0.4m/s$.

Given an initial position of

$$q = \begin{bmatrix} 5 & 15 & 15 & 3 \end{bmatrix}^\top$$

the surface of the cube is stabilized by determining the closest facet to the initial starting position of the manipulator, and then selecting the corresponding transverse controller as outlined in Chapter 3.

However, due to the starting position, the closest facet to the end effector is not the bottom facet with a path defined. Thus, the controller must apply Dijkstra's algoritm to determine the intermediate facets and construct a rudimentray path to reach the bottom facet.

Once the bottom facet has been reach, the nested transverse feedback linearization problem can be solved. This is accomplished by using the coordinate transformation above, to transform the tangential dynamics to the following

$$\begin{aligned} \epsilon_1 &= (\eta_1 - 0.15)^2 + (\eta_3 - 0.15)^2 - 0.4 \\ \upsilon_1 &= 0.4 \arctan\left(\frac{(\eta_1 - 0.15)}{(\eta_3 - 0.15)}\right). \end{aligned} \tag{4.7}$$

With these dynamics, we apply the nested feedback transformation outlined in (4.5), and select w to stabilize the circular path, and move along it with a constant velocity

$$w = \begin{bmatrix} K_1 \epsilon \\ K_2 \upsilon_2 \end{bmatrix}.$$

Simulation results are shown in Figure 4.2.

♦

△

4.3 Damped motion

Following [49], there are certain haptics applications in which it is desired that the motion along the surface of the polytope be damped. This can easily be incorporated by designing the tangential controller $v^{\parallel}$ for each polytope based on the tangential dynamics (4.2). To this end we choose

$$v^{\parallel} = -\operatorname{diag}(b_1, \ldots, b_{p-q})\eta_2$$

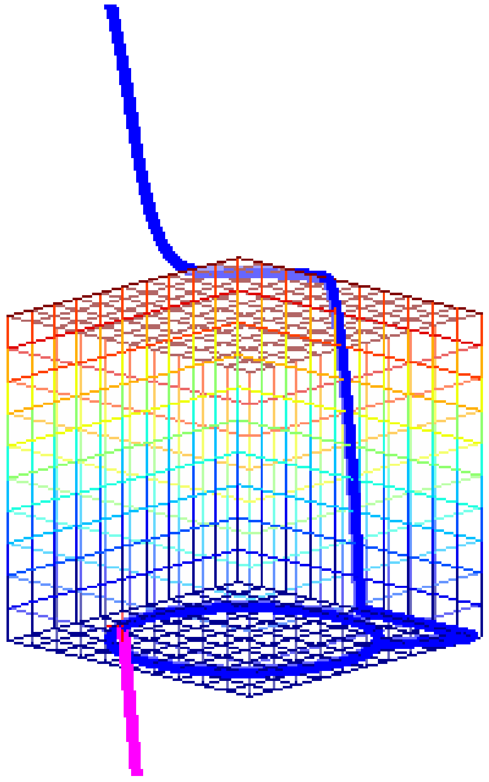


Figure 4.2: Desired Path Defined on a Single Face

where $b_i > 0$ represents the desired damping of tangential state i.

Example 4.3.1. Given a 4 DOF robotic manipulator, with system of the form (1.3), we aim to stabilize the surface of a sphere and provided damped motion along its surface.

This sphere is centered at $\begin{bmatrix} 2 & 0 & -2.5 \end{bmatrix}^\top$. The sphere is given a polytope approximation using the algorithm outlined in [43]. Approximating the sphere with 1200 facets, the transverse controller is able to stabilize and switch along the polytope surface, using the methods outlined in Chapter. 3. The transverse controller is a simple PD controller, with double poles at -3.

The tangential controller is able to simulated damped motion, with a damping coefficient of $b = 1N/cm$ by setting

$$v^{\parallel} = -\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \eta_2.$$

The resulting simulation can be seen in Figure 4.3.

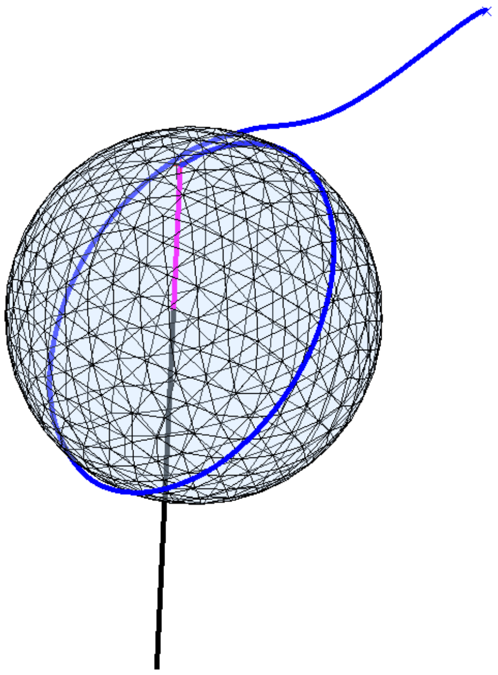


Figure 4.3: Course Approximation of Sphere

To demonstrate the flexibility of the algorithm, the approximation is increased to 12000 facets and re-simulated with no change to the controller. This result can be seen in Figure 4.4.

♦

△

4.3.1 Tracking control

Tracking control is a popular problem often presented in the literature, thus it is natural to attempt to solve the problems it presents. In this implementation, we assume that there exists an external representation of the desired trajectory known as the exomodel; explained in detail in [18]. This approach was implemented for a single plane, but was not fully explored for a general polytope. We begin by presenting the solution to the tracking problem on a single plane.

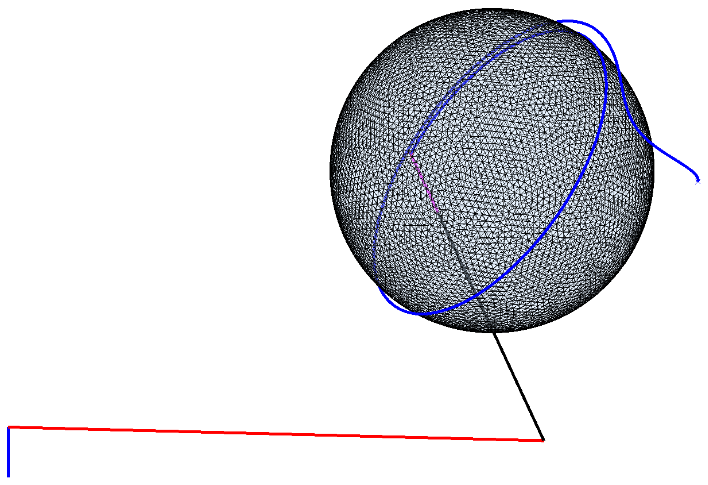


Figure 4.4: Precise Approximation of Sphere

In a local neighbourhood on the plane, we know that our tangential dynamics become the form outlined in (4.2). Using these dynamics, we define a state model with

$$\omega := \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix},$$

where the plant dynamics are defined by

$$\dot{\omega}_1 = A_1\omega_1 + A_3\omega_2 + B_1 v^{\|}$$

and the exomodel defined by

$$\dot{\omega}_2 = A_2\omega_2.$$

We then define a signal e as the tracking error between the system and the exomodel which we wish to regulate

$$e = D_1\omega_1 + D_2\omega_2.$$

This model allows us the flexibility for including external disturbances through the matrix A_3. However, as our system does not have any external disturbances we shall simply choose $A_3 = 0$.

Given the structure of our system, we are able to easily define the matrices A_1, A_3 and B_1 as follows

$$\begin{aligned} A_1 &:= \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \\ A_3 &:= 0 \\ B_1 &:= \begin{bmatrix} 0 \\ I \end{bmatrix}. \end{aligned}$$

A_2 can then be chosen to represent a desired external trajectory.

We can combine the state model as

$$\begin{aligned} \omega &= \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \\ \dot{\omega} &= A\omega + Bv^{\parallel} \\ e &= D\omega \\ A &= \begin{bmatrix} A_1 & A_3 \\ 0 & A_2 \end{bmatrix} \\ B &= \begin{bmatrix} B_1 \\ 0 \end{bmatrix} \\ D &= \begin{bmatrix} D_1 & D_2 \end{bmatrix}. \end{aligned}$$

We assume that A_2 only has unstable eigenvalues, as it represents an arbitrary trajectory. Applying a controller of the form $v^{\parallel} = F\omega$, the controlled system becomes

$$\begin{aligned} \dot{\omega} &= (A + BF)\omega \\ e &= D\omega. \end{aligned}$$

We then invoke the following theorem from [18] to solve for the coefficients for our controller to be both stabilizable and asymptotically track the exomodel.

Theorem 4.3.1. *Assume A_2 has only unstable eigenvalues. Then the regulator problem is solvable by some $v^{\parallel} = F\omega$ if and only if (A_1, B_1) is stabilizable and there exist a matrix K, U such that*

$$A_1K - KA_2 + A_3 + B_1U = 0, \quad (4.8)$$

$$D_1K + D_2 = 0 \quad (4.9)$$

Example 4.3.2. Given a 4 DOF Robotic Manipulator of the form (1.3), we seek to stabilize a hyperplane defined by $A = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$, $b = 100$. The state model can then be chosen as

$$\omega = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix}$$

where $\dot{\eta_1} = \eta_2, \dot{\eta_3} = \eta 4$.

The exosystem is a straight line defined with A_2 as

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and regulator error defined as

$$D_1 = -I_{4\times 4},\ D_2 = I_{4\times 4}.$$

The matricies A_1 and A_3 would then be defined as

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$A_3 = 0_{4\times 4},$$

and

$$B_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Letting $U = F_1 X + F_2$, we select F_1 to place four poles at -10, and

$$F_2 = U - F_1 X.$$

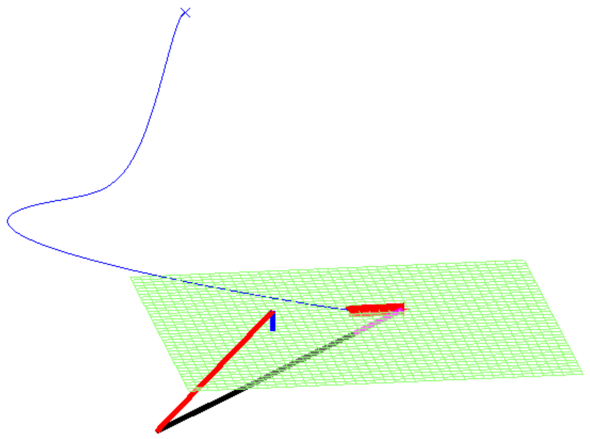


Figure 4.5: Tracking Controller Implementation

Invoking Theorem. 4.3.1, we solve for K_3.

$$K_3 = \begin{bmatrix} 105 & 20 & -4 & 0 & -105 & -20 & 4 & 0 \\ -4 & 0 & 107 & 21 & 4 & 0 & -107 & -21 \end{bmatrix}.$$

Simulating with these gains produces Figure 4.5.

♦

△

With the solution for a single plane presented, we now extend this solution to our polytope system. As with other tangential control methods, our switching algorithm will select the desired facet based on the distance. Once this has been established, the exosystem and associated system dynamics corresponding to the desired trajectory and facet will be activated. If there is no exosystem defined on the particular facet, we will once again rely on Dijkstra's algorithm to drive the system output to the closest facet where a trajectory has been defined.

Although this solution is simple and easy to understand, the definition of the exosystems on the desired facets is not a trivial task. If there is a somewhat complex trajectory desired, the definition of the exosystem becomes time consuming and difficult. Nevertheless, this solution is still an attractive method, and should be further considered and refined.

Chapter 5

Simulation and experimental results

In this Chapter several experiments are presented that demonstrate the effectiveness of the proposed controllers. The platform selected for the experiments was a four degree-of-freedom (DOF) robotic manipulator. Simulations were first conducted to test the viability of the experiments before they were applied on the experimental platform. These experiments included the stabilization of a liner affine hyperplane and following of a circular path defined on the plane, stabilization of a cube and following a path defined along the cube, as well as stabilization of a cube with damped motion along its surface.

5.1 Experimental configuration

The selected experimental platform was a four DOF robotic manipulator. This platform allowed for many complex variables, such as drag forces, to be ignored, while still providing an interesting system to demonstrate the viability of the proposed algorithms. This robot can be seen in Figure 5.1.

The model used in the simulations is described in further detail in [20] and is based on the Euler-Lagranage equations. The model introduced in [20] matches the desired form outlined in (1.3), and the algorithms presented are thus applicable. In these simulations and experiments, $N = 4$, $p = 3$ and $m = 4$.

Note that the 4-DOF manipulator is kinematically redundant. This leaves $N - p = 4 - 3 = 1$ degree-of-freedom with no obvious control specification with respect to the polytope stabilization problem. Hence the ζ-dynamics in the normal form (3.8) exist. In

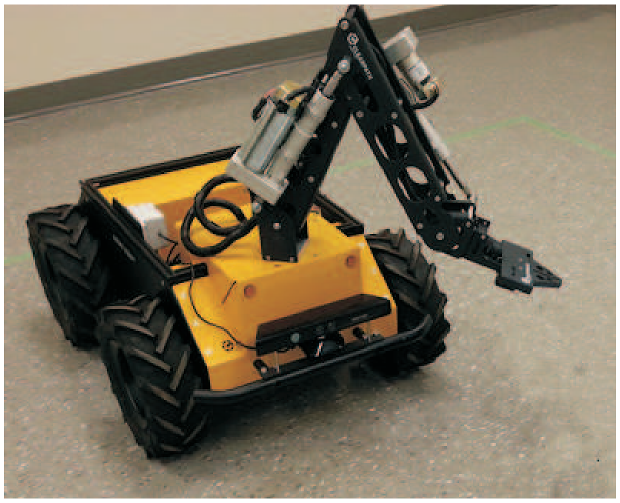


Figure 5.1: 4-Degree of Freedom Manipulator

K_p	16
K_d	8

Table 5.1: PD controller gains for wrist dynamics

this chapter we choose the $\phi(x)$ function in the coordinate transformation (3.6) to control the wrist dynamics

$$\begin{aligned}\phi_1(x) &= x_4\\ \phi_2(x) &= x_8.\end{aligned}$$

This choice of $\phi(x)$ allows for the system to be fully feedback linearized and the redundant dynamics stabilized with a PD controller. Thus, the poles of this simple system were placed at -4, using the gains outlined in Table 5.1.

The robot was controlled via a Labview interface, which drove a series of linear actuators. These linear actuators caused joint rotations, mimicking a rotary actuator. The joint positions and velocities were determined using incremental encoders, and the linear actuators controlled via PWM amplifiers with a 10 millisecond sampling time.

5.2 Simulations

5.2.1 Plane stabilization

Circle stabilization

Based on the work of [27], we attempt to apply the coordinate transformation outlined in (3.6) to the 4 DOF robotic manipulator. In this experiment, we aim to have the end effector of the manipulator follow a circular path defined in local coordinates on the plane. This is accomplished by applying the nested feedback linearization approach detailed in Section 4.2.

The plane is defined with A and b vectors as follows

$$A \coloneqq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\top, \ b = 0.1.$$

The target path can be defined as a circle, centered at $\begin{bmatrix} 0.075 & -0.15 \end{bmatrix}$ with radius 0.05 defined in local coordinates along the plane. To achieve the nested feedback linearization, another coordinate transformation is applied to transform the η dynamics to the following

$$\begin{aligned} \epsilon_1 &= (\eta_1 - 0.075)^2 + (\eta_3 + 0.15)^2 - 0.05 \\ \epsilon_2 &= 2\,(\eta_1 - 0.075)\,\eta_2 + 2\,(\eta_3 + 0.15)\,\eta_4 \\ \upsilon_1 &= 0.05 \arctan\left(\frac{(\eta_1 - 0.075)}{(\eta_3 + 0.15)}\right) \\ \upsilon_2 &= \frac{0.05\,(\eta_2\,(\eta_3 + 0.15) - \eta_4\,(\eta_1 - 0.075))}{\left((\eta_3 + 0.15)^2 + (\eta_1 - 0.075)^2\right)} - 0.01. \end{aligned} \tag{5.1}$$

This choice of transformation allows the nested transverse dynamics to represent the "distance" between the end effector and the target path, and the nested tangential dynamics to represent the motion along the path.

The plane was stabilized using a simple PD controller for the transverse dynamics. The controller was designed using the gains outlined in Table 5.2. With the appropriate nested coordinate transformation, the nested transverse dynamics were stabilized using a simple PD controller as well, while the nested tangential controller was simple designed to achieve a desired constant speed of $-0.01 m/s$. The nested transverse PD controller was designed using the gains outlined in Table 5.3.

K_p	9
K_d	6

Table 5.2: PD controller gains for transverse dynamics

	ϵ	υ
K_p	9	0
K_d	6	6

Table 5.3: PD controller gains for tangential dynamics

With the appropriate transformations applied, and controllers designed the system was simulated. In the simulation, the manipulator began with its end effector off of the plane and was forced to stabilize the surface. The result of this simulation can be seen in Figure 5.2.

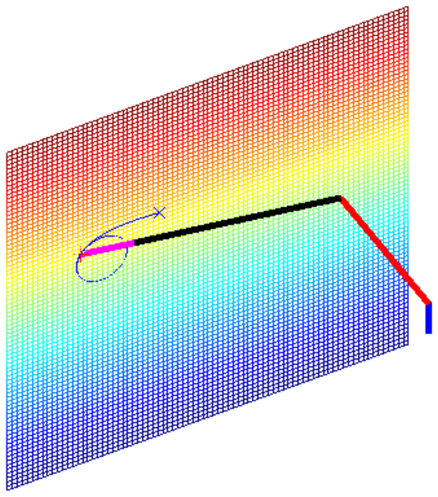


Figure 5.2: Simulated Circle Stabilization

The performance of the controller is emphasized when the ξ and η dynamics are analysed. As seen in Figure 5.3, the transverse controller is able to smoothly and rapidly drive the system towards the plane, where it remains invariant to the surface for all future time.

Similarly, when the nested transverse and nested tangential dynamics are analyzed, as seen in Figure 5.4, we see a similar trend in the nested transverse system.

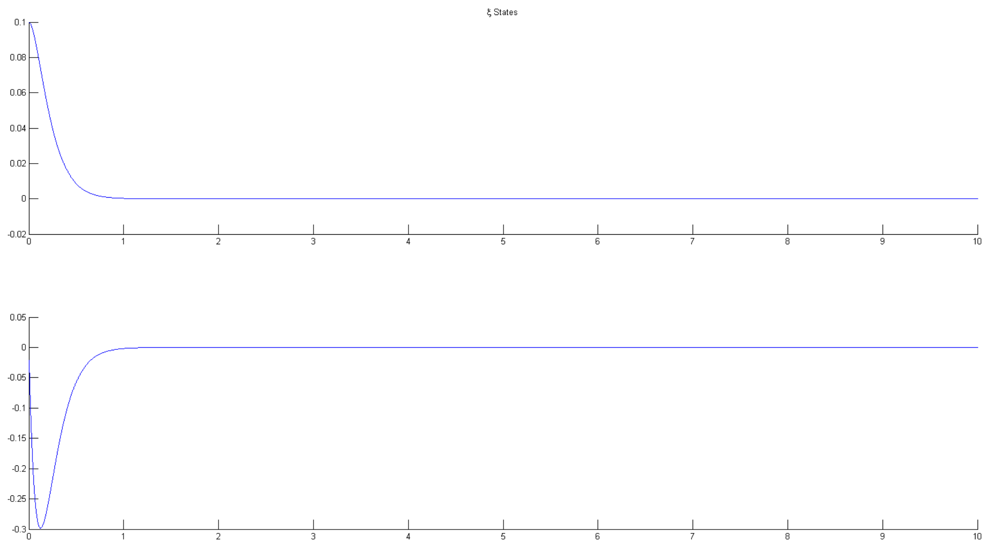


Figure 5.3: ξ dynamics of circle simulation

As the figure shows, the nested transverse dynamics are quickly and smoothly regulated, while the nested tangential dynamics achieve the desired constant velocity. These results are promising, especially when the control effort is analysed. As seen in Figure 5.5, the controller is able to achieve the above performance without excessively large values of control effort.

5.2.2 Cube stabilization

Path following

With the single plane simulation we are able to ensure that the transformation selected for the feedback linearization controller design is indeed diffeomorphic on our target set. Furthermore, we were able to test the transverse controllers, as well as the nested feedback approach.

The next level of complexity was to attempt to stabilize a polytope and follow a path defined on multiple facets. A cube was selected as the target polytope, as it demonstrated the ability of the algorithm to handle sharp corners as well as provide a natural extension from the single plane case.

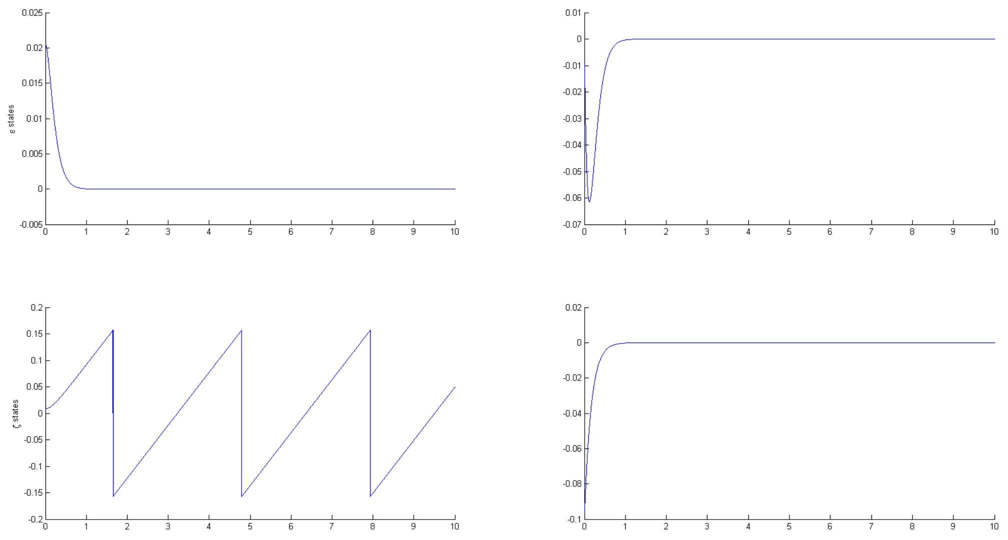


Figure 5.4: Nested dynamics of circle simulation

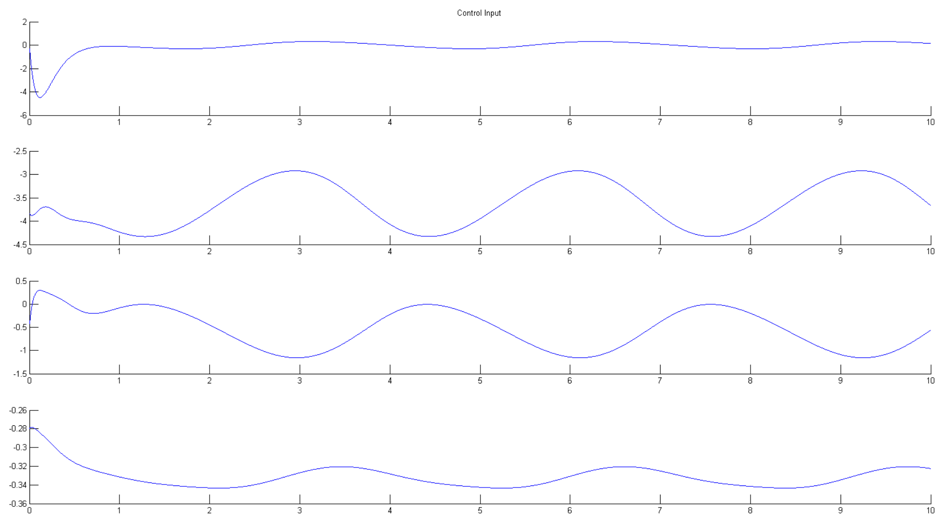


Figure 5.5: Control effort of circle simulation

The cube was selected as

$$\begin{aligned} P &\coloneqq \text{conv}\,\{v_0, \ldots, v_8\}, \\ &= \text{conv}\left\{ \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}2\\1\\1\end{bmatrix}, \begin{bmatrix}1\\2\\1\end{bmatrix}, \begin{bmatrix}2\\2\\1\end{bmatrix}, \begin{bmatrix}1\\1\\2\end{bmatrix}, \begin{bmatrix}2\\1\\2\end{bmatrix}, \begin{bmatrix}1\\2\\2\end{bmatrix} \right\}. \end{aligned}$$

centred at $\begin{bmatrix}0.5 & 0.5 & 0.5\end{bmatrix}^\top$, with $M = 6$, and the A_i, b_i and Q_i vectors and matrices generated according to the procedures outlined in Chapter 2.

In this simulation, we are interested in following a path that encircles the meridian of the cube. This path $\sigma(\lambda)$ is defined as follows

$$\sigma(\lambda) \coloneqq \begin{cases} \begin{bmatrix}0.10 + \lambda & 0.20 & 0.15\end{bmatrix}^\top \\ \begin{bmatrix}0.20 & 0.30 - \lambda & 0.15\end{bmatrix}^\top \\ \begin{bmatrix}0.40 - \lambda & 0.10 & 0.15\end{bmatrix}^\top \\ \begin{bmatrix}0.10 & \lambda - 0.20 & 0.15\end{bmatrix}^\top \end{cases}.$$

Since the path is defined on only four planes, we will have four distinct nested feedback linearization controllers, two controllers implementing the Djkstra's algorithm approach and six transverse controllers to stabilize each facet. These controllers will take the form of those outlined in Chapter 3.

To avoid infinite switching, a small dwell time was selected. The choice of the dwell time was driven by the size of the facets, and the speed at which the manipulator was to be moving. The value for dwell time was selected by iteratively increasing the time until the system became unstable. After this time, the dwell time was selected as half the value that caused instability at $t_{dwell} = 0.15\ s$.

Next, as outlined in Algorithm 3.2.1, a tolerance ϵ was required. To attempt to mimic the typical surface invariance conditions a tolerance of $\epsilon = 0.01\ m$ was selected. This choice was made to represent the end effector being functionally close to the surface.

A simple PD controller was selected for the transverse controller, with gains outlined in Table 5.2. The nested transverse and tangential controllers were also designed as PD controllers with gains outlined in Table 5.3. The projection algorithm detailed in Appendix B.1 was chosen for the plane selection due to its simplicity. The simulation was conducted with the end effector initialized off of the polytope and can be seen in Figure 5.6.

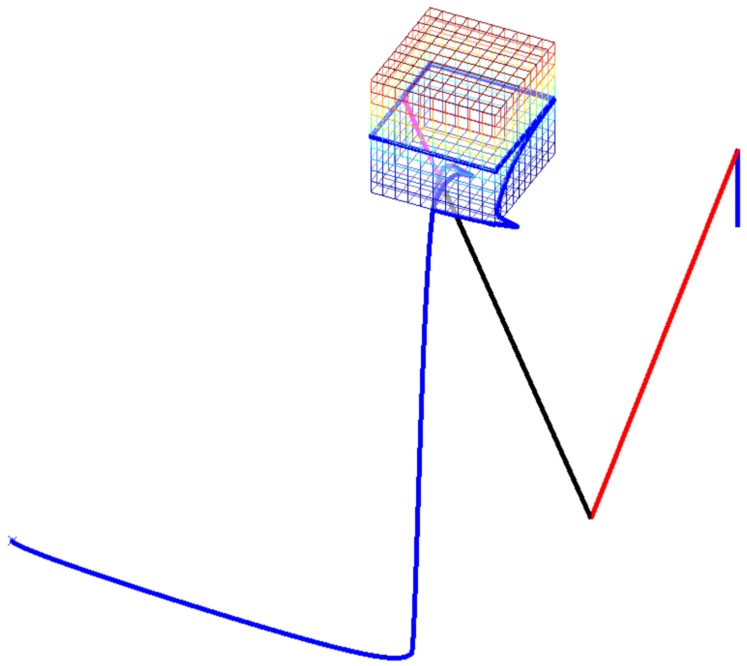


Figure 5.6: Desired path continued on different facets

As Figure 5.7 shows, the controller is able to appropriately select the corresponding facet as it switches along the facets of the polytope. Furthermore, although there is an initial overshoot when a switch occurs, the error is quickly eliminated and remains small for the entire time on the facet. This result is reiterated in Figure 5.8, when the ξ dynamics are compared. As the figure shows, there exists the same trend in the ξ dynamics, while the $\dot{\xi}$ dynamics remain relatively small.

When the control effort of the controller is analysed in Figure 5.9, we see that very little control effort is required, and is well within the 15V safety margins for the actuators.

In an attempt to improve the performance of the transverse dynamics, the gain of the system was increased to $K_p = 65,\ K_d = 16$. These more aggressive gains helped to reduce the time required to stabilize the dynamics, as seen in Figure 5.10, but did not reduce the magnitude of the overshoot.

This change in gain was also able to improve the performance of the system without a

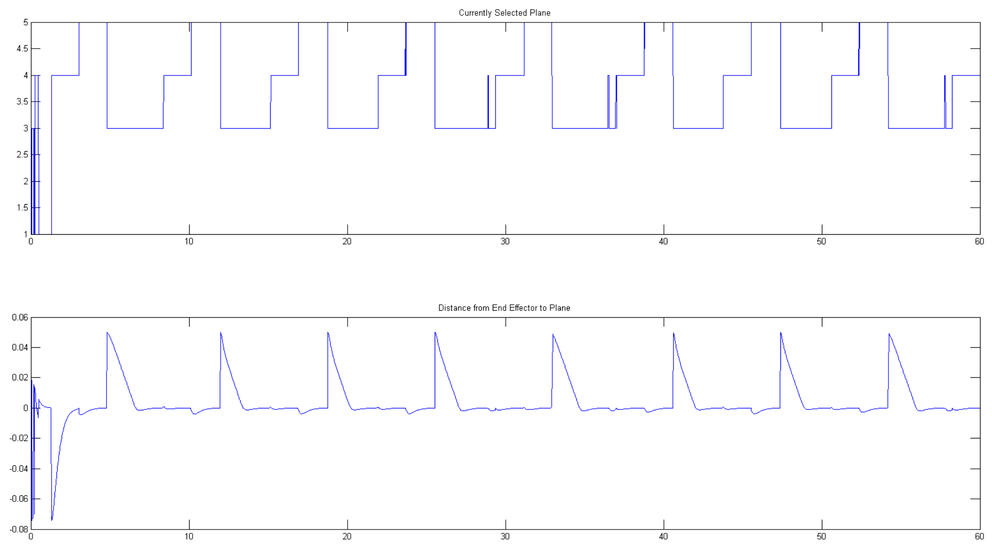


Figure 5.7: Distance to currently selected plane for Cube simulation

substantial increase in control effort. This is seen in Figure 5.9, where the control effort, although larger, is still within the safety margins.

As outlined in Chapter 3, due to the transformation applied to the system, the transverse dynamics are reduced to a simple linear system. Thus, we are able to apply any number of linear control techniques in an attempt to stabilize the transverse dynamics. Two choices are a simple PD controller, or an LQR based designed. In an attempt to compare the two control techniques the system was simulated with both approaches. The LQR controller was designed to penalize the state conditions heavily with cost functions as follows

$$Q_{LQR} = 10 I_{2\times2},\ R = 1.$$

The results of the two simulations can be seen in Figure 5.12 and Figure 5.13. As the figures show, both approaches produce stable systems that are able to follow the desired paths. However, the LQR approach produces a system with far less overshoot than the PD controller. Although one could argue that the PD controller could be tuned to reduce the overshoot, the LQR approach allows a systematic approach for reducing the overshoot. It should be noted that even with the LQR approach, some overshoot during switching is inevitable. This is largely caused by the physical inertia of the system, and may possibly

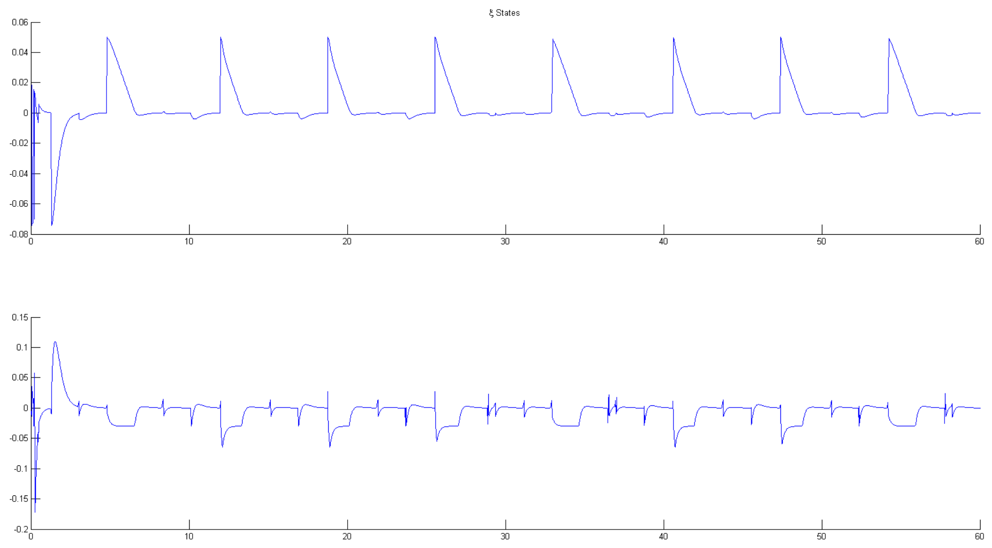


Figure 5.8: ξ states for Cube simulation

be eliminated with a speed profile slowing the system as it approaches the ridges.

Damped Motion

As outlined in Chapter 4, the control of the system along the polytope surface is structured in such a way that different control techniques can be implemented. As such, the controller outlined in Section 4.3 was constructed for the cube. The choice to implement this control technique was motivated by the successful application of a similar experiment in [49].

A damping coefficient of $b = 0.1N/m$ was selected for the tangential controller, and the transverse controller from the previous section was implemented without change. Since there is not particular path defined, the tangential controller for all facets was of the form

$$v_i^{\parallel} = -0.1\eta_2, i \in \{1, \ldots, 8\} .$$

The end effector was initialized off of the polytope surface, forcing the transverse controller to actively stabilize the polytope. Furthermore, the manipulator end effector was initialized at $0.5m/s$. This initial velocity caused the end effector to continue its motion once it had reach the surface of the polytope, causing the controller to switch to a new

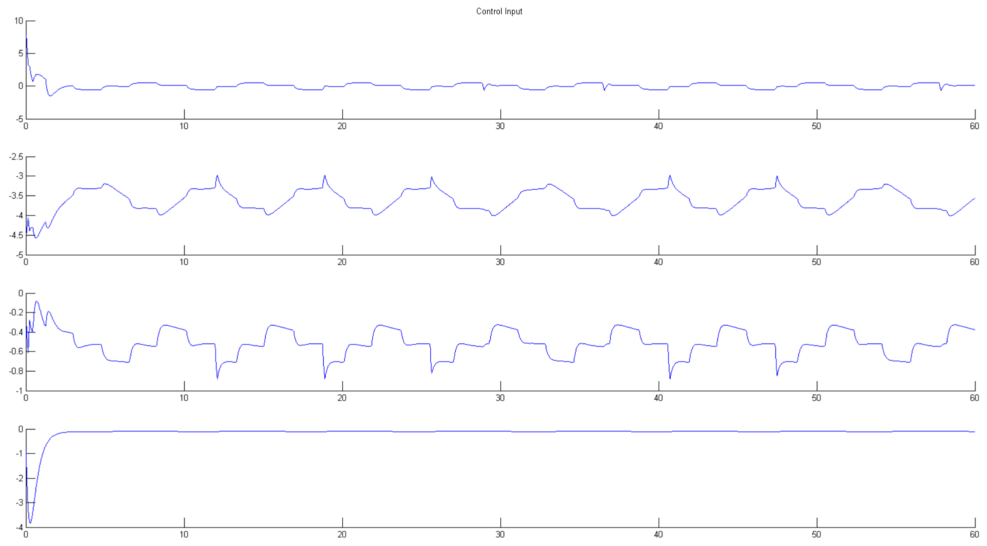


Figure 5.9: Control effort for Cube simulation

facet. Eventually, the damped motion along the polytope surface caused the end effector to cease its motion. This simulation result is seen in Figure 5.14.

5.3 Experiments

5.3.1 Plane stabilization

Point stabilization

To systemically implement the controllers, a simple point stabilization experiment was conducted. This experiment would allow for the transverse controller to be tested without major complications introduced from the tangential controller.

For simplicity, the transverse controller was selected as a simple PD controller, with gains as outlined in Table 5.2. The tangential controller was defined as follows

$$v^{\parallel} = \begin{bmatrix} K_{p_{\parallel}} & K_{d_{\parallel}} \end{bmatrix} \begin{bmatrix} \eta_1 - \eta_{desired_1} \\ \eta_2 - \eta_{desired_2}, \end{bmatrix}$$

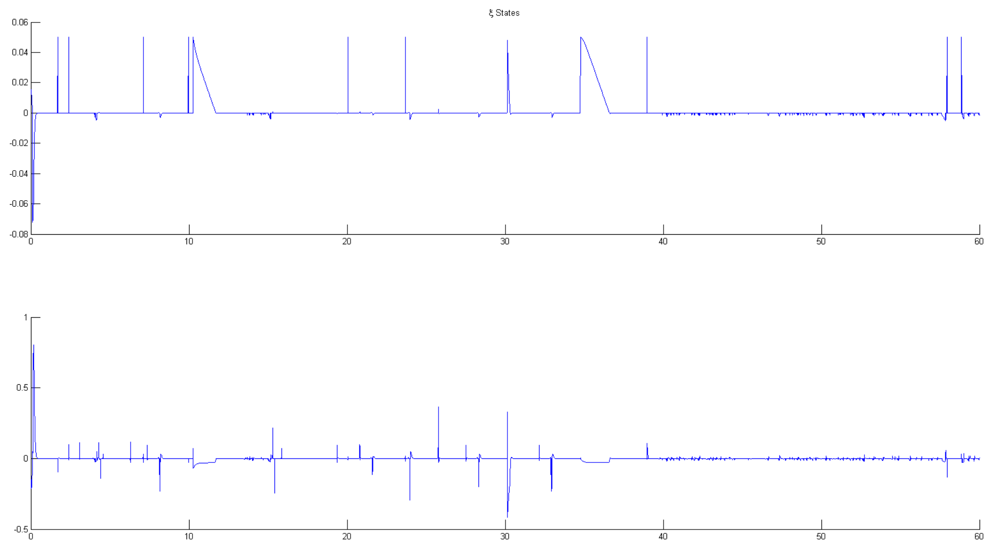


Figure 5.10: ξ states for Cube simulation with aggressive gains

with $K_{p_{\|}} = 1$ and $K_{d_{\|}} = 2$. As with the circle stabilization simulation, the plane was defined with the following A and b vectors

$$A \coloneqq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\top}, \; b = 0.1.$$

The manipulator was initialized off of the plane, and forced to stabilize the surface with the transverse controller, while the motion along the plane was controlled with the tangential controller. The results of this experiment can be seen in Figure 5.15.

The performance of the controller is evident when the ξ and η dynamics are analysed. As seen in Figure 5.16, the controller is able to drive the end effector towards the surface. However, due to frictional effects, there exists a constant steady state error between the end effector and the plane. The steady state error also manifests itself in the η dynamics, as η_3 can also be seen to have steady state error.

Nevertheless, the transverse and tangential controllers were able to drive the end effector output towards the desired location along the plane within reason. Thus, the complexity of the control along the surface was increased.

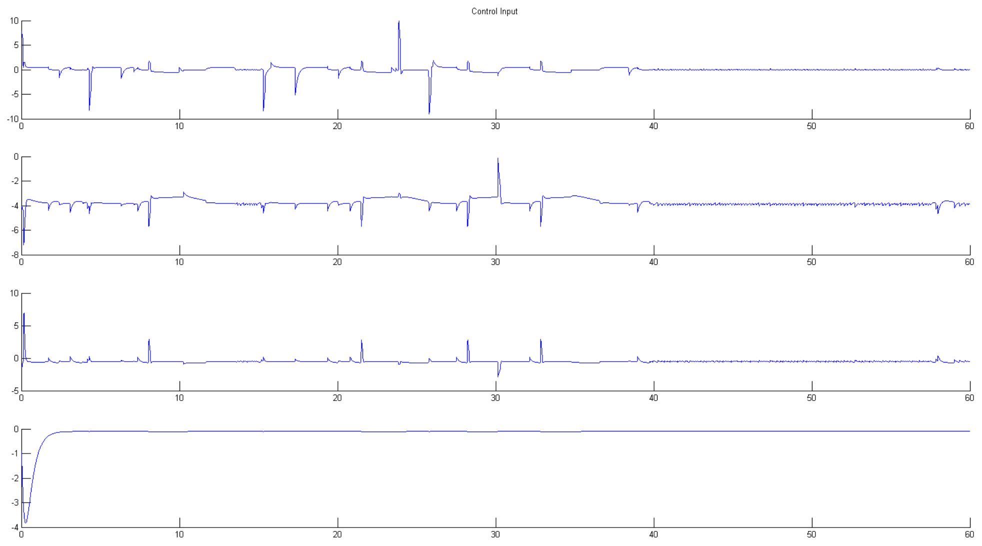


Figure 5.11: Control effort for Cube simulation with aggressive gains

Circle stabilization

With the completion of the experiments focused on point stabilization on a plane, the next logical step was to increase the complexity of the control along the surface. To this end, the nested feedback linearization for the following of a path was implemented.

The first attempt to implement the stabilization of the circular path on the plane was to apply the controller as it was outlined in the circle stabilization simulation. As with the point stabilization experiments, there was noticeable steady state error in the transverse dynamics, as seen in Figure 5.18. However, the tangential dynamics had larger errors, and produced a generally non-smooth path. The results of this experiment can be seen in Figure 5.19. Although the errors in the ξ dynamics were within the same order as those observed in the point stabilization experiments, the poor performance of the nested controllers warranted the investigation of robustification techniques.

The first attempt to robustify the transverse controller was the addition of an integrator to try and eliminate the steady state errors. This was accomplished by creating a virtual state to represent the steady state error, and attempting to regulate it. This approach resulted in the gains outlined in Table 5.4.

Although the addition of the integrator to the transverse controller helped improve the

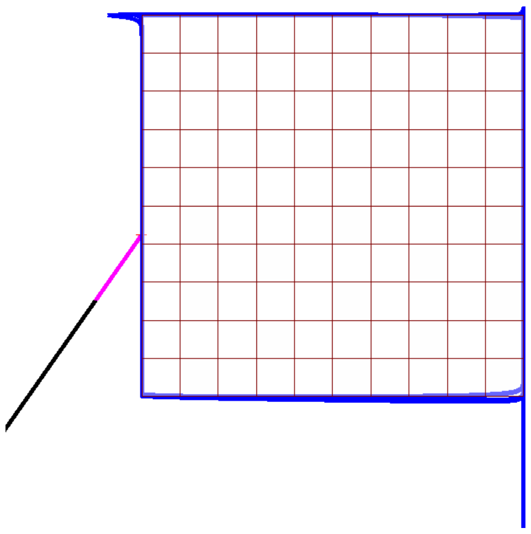


Figure 5.12: State Feedback Based Transverse Controller

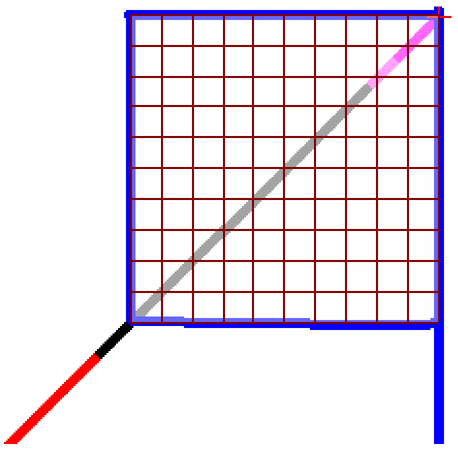


Figure 5.13: LQR Based Transverse Controller

ξ dynamics, it did not affect the tangential dynamics. Thus, to further attempt to deal with modelling uncertainties, such as friction, another robust modification was made. This

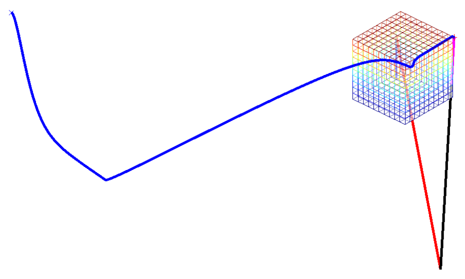


Figure 5.14: Simulation of Damped Motion on a Cube

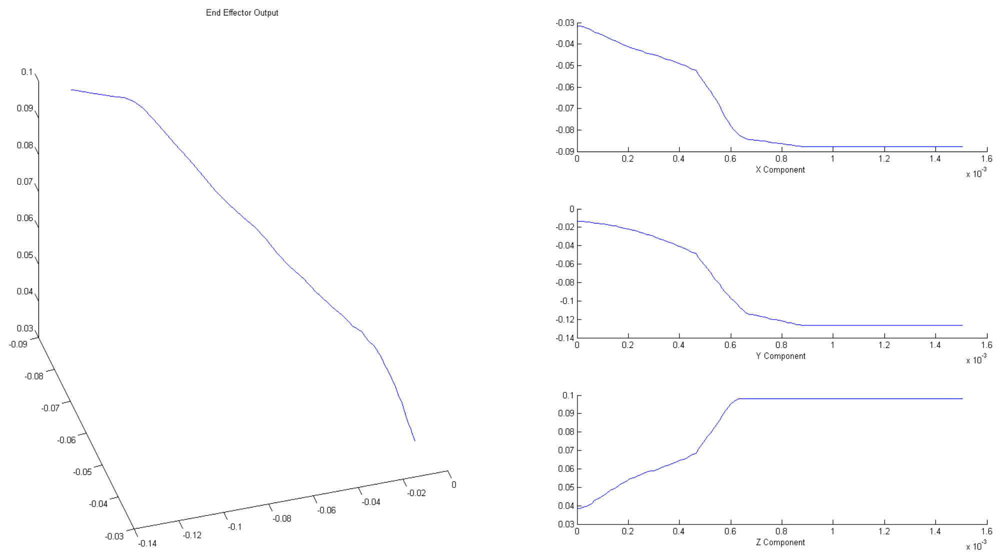


Figure 5.15: Experimental Point Stabilization on a Linear Affine Plane

robust modification followed the Lyapunov redesign algorithms outlined in Section 3.4.2. Following the design methods detailed in [20], the nested transverse controller was modified

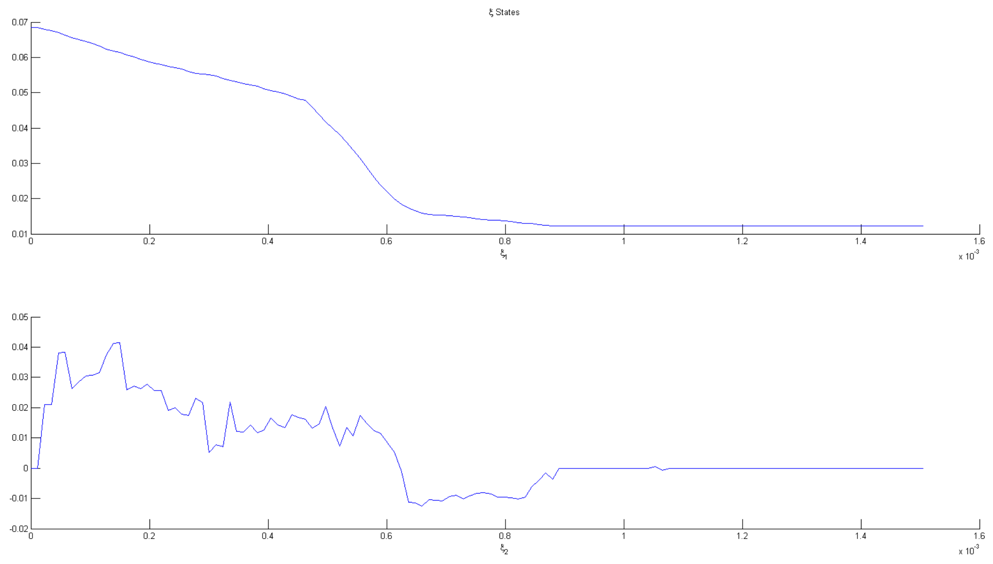


Figure 5.16: ξ dynamics for point stabilization on a plane

	ϵ	υ
K_p	9	0
K_i	12	0
K_d	6	6

Table 5.4: PD controller gains for tangential dynamics

to become

$$w_\epsilon = -\left(\left(\begin{bmatrix} K_0 & 0 \end{bmatrix} + \begin{bmatrix} K_p & K_d & K_i \end{bmatrix} \right) \begin{bmatrix} \epsilon \\ \|\epsilon\| \end{bmatrix} + \begin{cases} K_1 \frac{\epsilon}{\|\epsilon\|} & : \|\epsilon\| \geq \mu > 0 \\ K_2 \|\epsilon\| \epsilon & : \|\epsilon\| < \mu, \end{cases} \right)$$

$$w_\upsilon = -K_{d_\upsilon} \upsilon$$

where the gains K_p, K_i and K_d were defined in Table 5.4, and the Lyapunov redesign gains defined in Table 5.5.

These modifications produced a large improvement in both the transverse and nested transverse dynamics. As Figure 5.20 shows, the end effector approaches the plane as time passes and eventually minimizes the steady state error. This is emphasized when

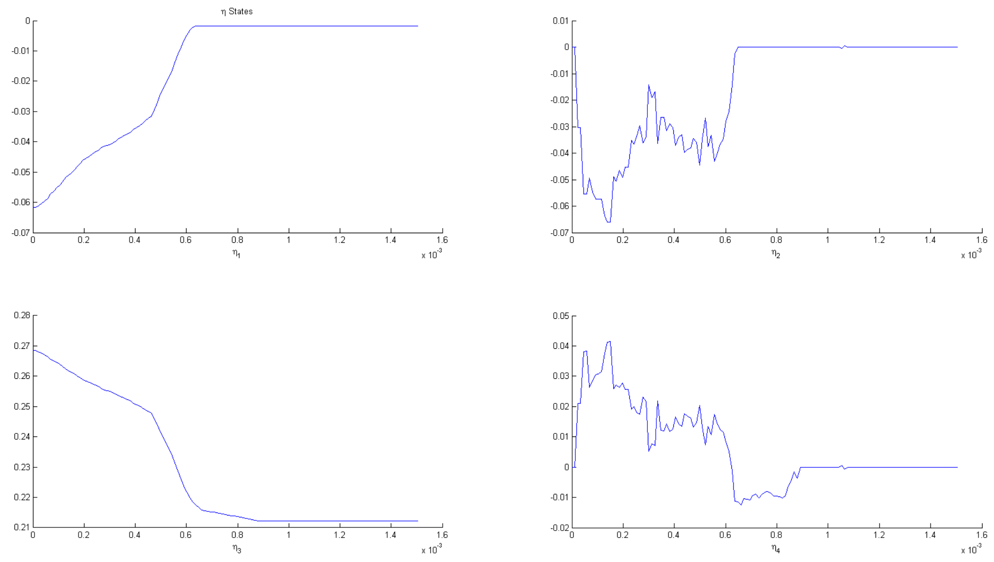


Figure 5.17: η dynamics for point stabilization on a plane

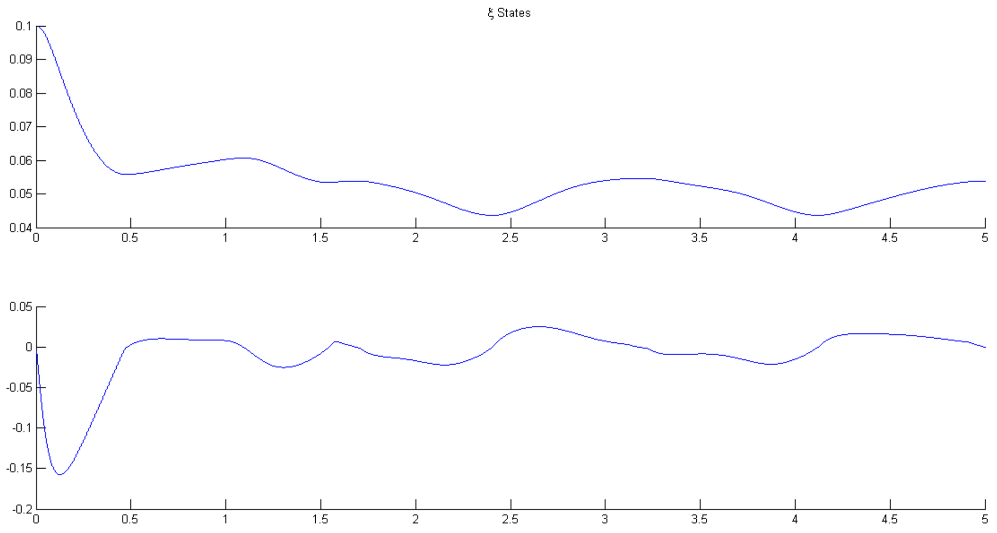


Figure 5.18: Experimental Circle Stabilization ξ dynamics

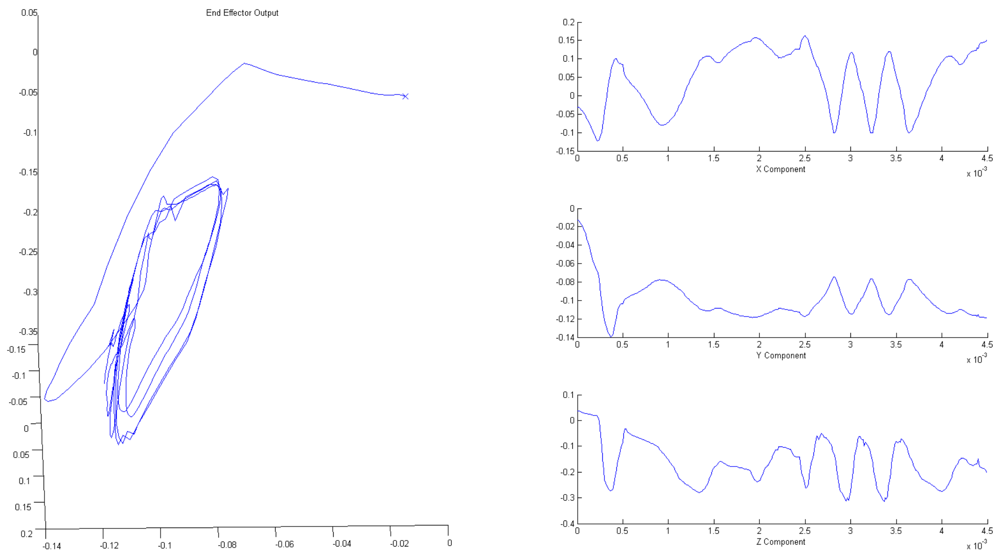


Figure 5.19: Experimental Circle Stabilization without compensation

μ	0.2	
K_0	20	20
K_1	70	70
K_2	70	70

Table 5.5: PD controller gains for tangential dynamics

Figure 5.21 is analysed, and the ξ dynamics approach and remain close to zero. A similar effect is seen with the ϵ dynamics, as outlined in Figure 5.22. As the figure shows, the ϵ dynamics approach zero and remain close for all future time.

The effect of the uncertainties to the system are large and emphasized when comparing the simulation and experimental results. This is particularly true when the control effort of the system is analysed in Figure 5.23. As the figure shows, the control effort for two of the joints nearly reaches the motor limit of 24V, indicating that considerable effort is being applied to attempt to overcome the uncertainties.

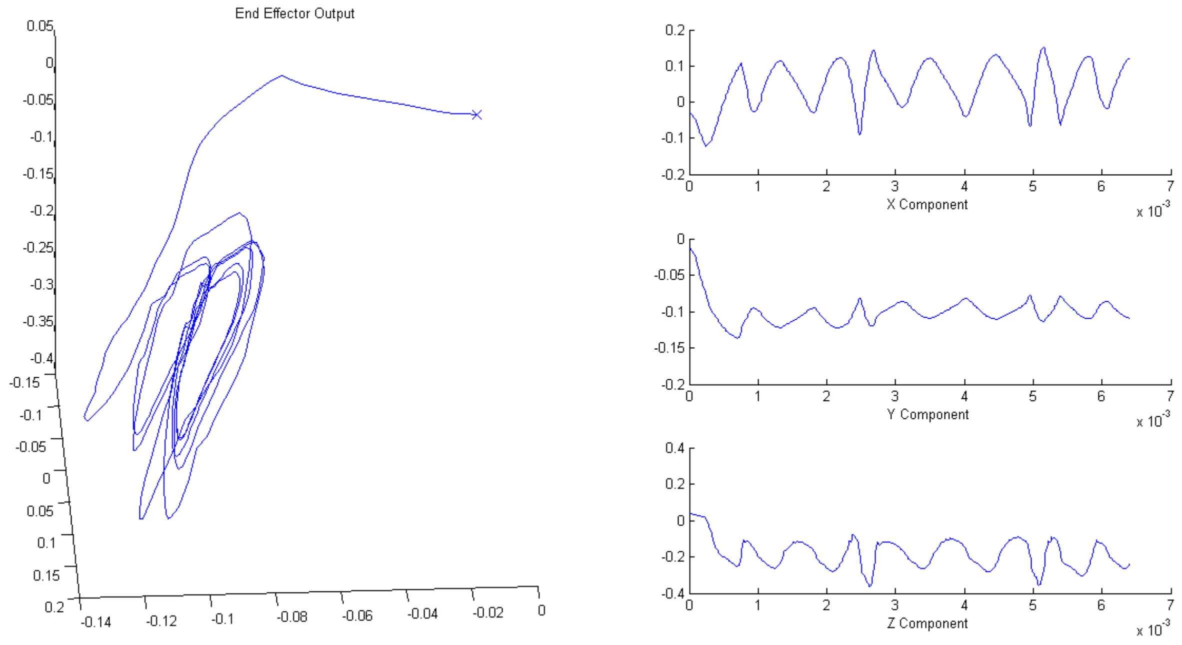


Figure 5.20: Experimental Circle Stabilization with Integral Action

5.3.2 Cube stabilization

Path following

With the lessons learned from the plane stabilization experiments, the stabilization of the cube was attempted. The transverse controller gains were selected to be the same as those used in the following of a circular path on a plane experiments rather than those used in simulation. Furthermore, the nested transverse and tangential dynamics also included the robust modifications in an attempt to deal with the effects of friction on the system. Thus,

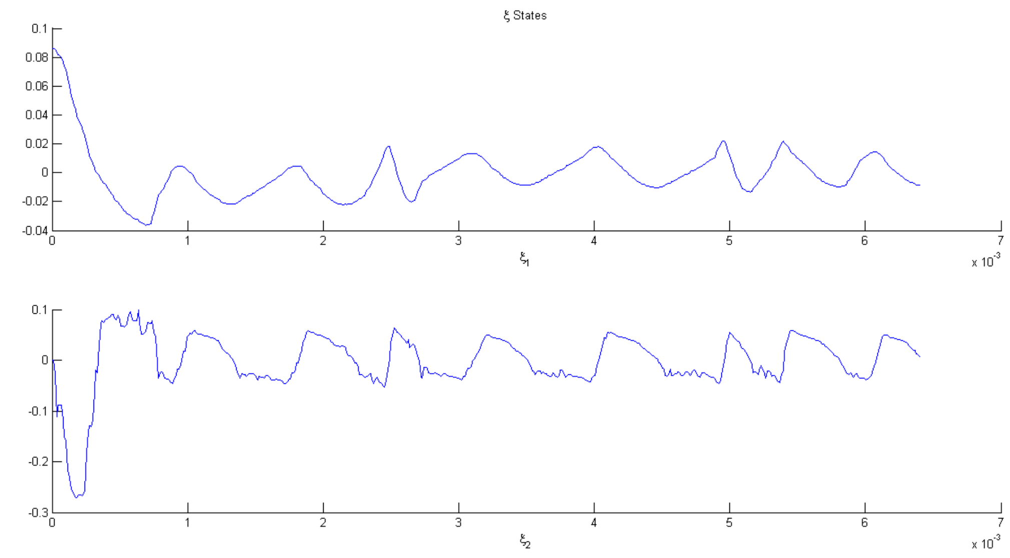


Figure 5.21: Experimental Circle Stabilization with Integral Action ξ dynamics

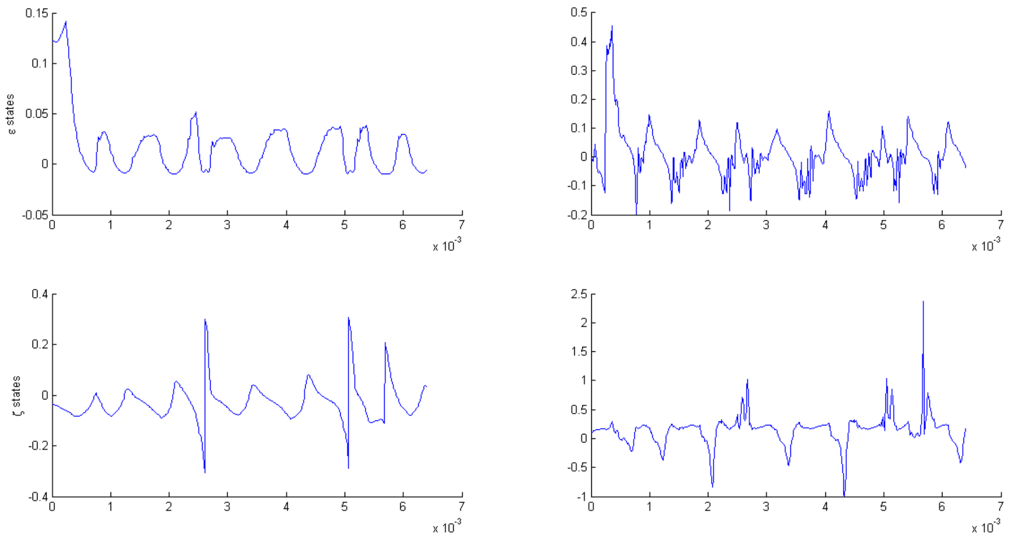


Figure 5.22: Experimental Circle Stabilization with Integral Action ϵ dynamics

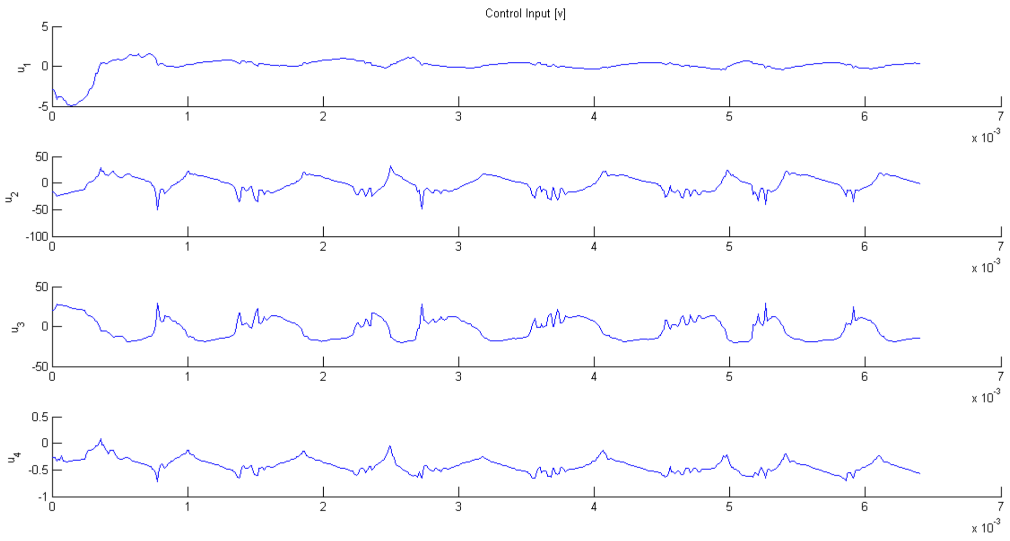


Figure 5.23: Experimental Circle Stabilization with Integral Action Control Effort

the controllers implemented were as follows

$$
\begin{aligned}
v_i^{\pitchfork} &= \begin{bmatrix} K_p & K_d & K_i \end{bmatrix} \begin{bmatrix} \xi_i \\ \dot{\xi}_i \\ \|\xi_i\| \end{bmatrix} \\
w_{\epsilon_i} &= -\left(\left(\begin{bmatrix} K_0 & 0 \end{bmatrix} + \begin{bmatrix} K_p & K_d & K_i \end{bmatrix} \right) \epsilon_i + \begin{cases} K_1 \frac{\epsilon_i}{\|\epsilon_i\|} & : \|\epsilon_i\| \geq \mu > 0 \\ K_2 \|\epsilon_i\| \epsilon_i & : \|\epsilon_i\| < \mu, \end{cases} \right) \\
w_{v_i} &= -K_{d_{v_i}} v_i,
\end{aligned}
$$

where the corresponding gains were tuned for each individual facet, but were within the neighbourhood of those outlined in Table 5.4 and Table 5.5.

Another modification between the simulations and the experiments were the tolerances ϵ and t_{dwell}. Both of these values were increase by fifty percent to accommodate the effect of the modelling uncertainties. Lastly, the speed at which the end effector was to traverse the meridian of the cube was reduced to decrease the overshoot observed when switching along facets.

To avoid any potential complexities, the experiment was separated into two parts. The first experiment had the end effector of the system initialized off of the polytope as it had been done in the previous experiments. The controller would then have the end effector stabilize the surface and followed the path along the surface and switch when required. The result of this experiment can be seen in Figure 5.24.

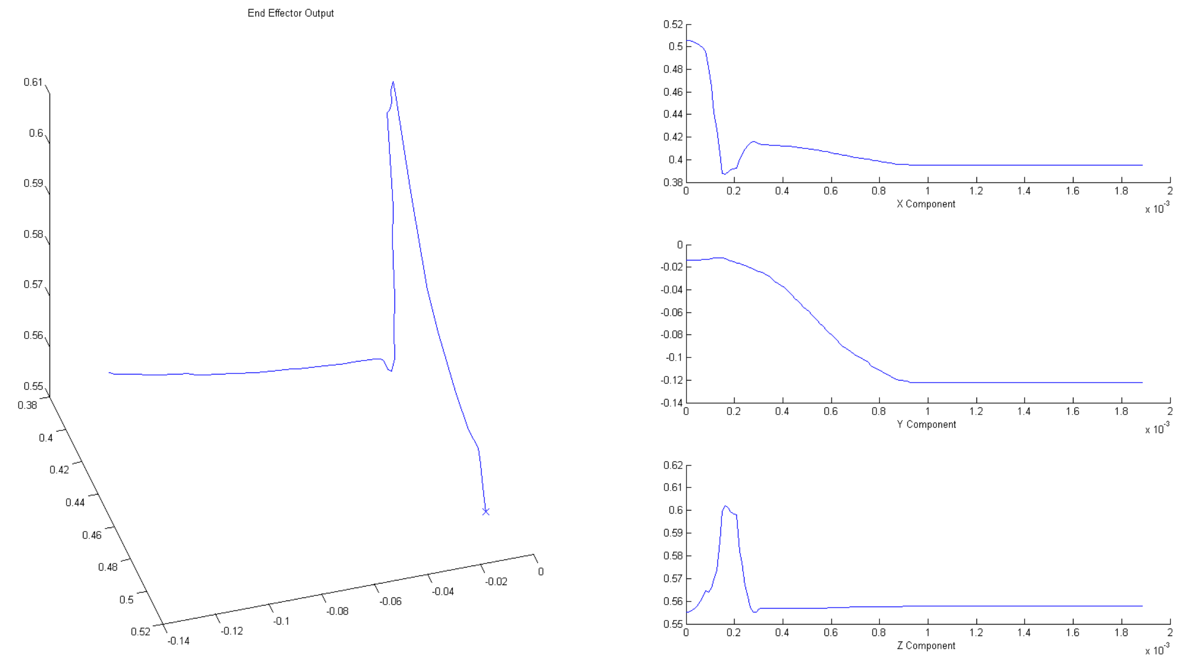


Figure 5.24: Cube Stabilization and Path Following

The second experiment had the end effector initialized on the polytope surface, and then traverse the path and switch on each facet. This experiment had the end effector make a complete traversal around the meridian of the cube, as opposed to the first experiment where only two facets were traversed. This result can be seen in Figure 5.25. The experiment was divided into two parts rather than one to avoid potential issues with the accumulation of error, that may cause instability in the integrators. Since the integrators were a rather late addition, and the robustifications had not fully been explored, this approach was done to ensure safety and prevent equipment damage.

Due to the robustifications and the decreased speed of the system, the controller is able to achieve better performance than the case of following the circular path. This is further emphasized when the ξ dynamics are analysed in Figure 5.26. Although there exists some deviations when switches occur in the system, the ξ state is approximately zero for the

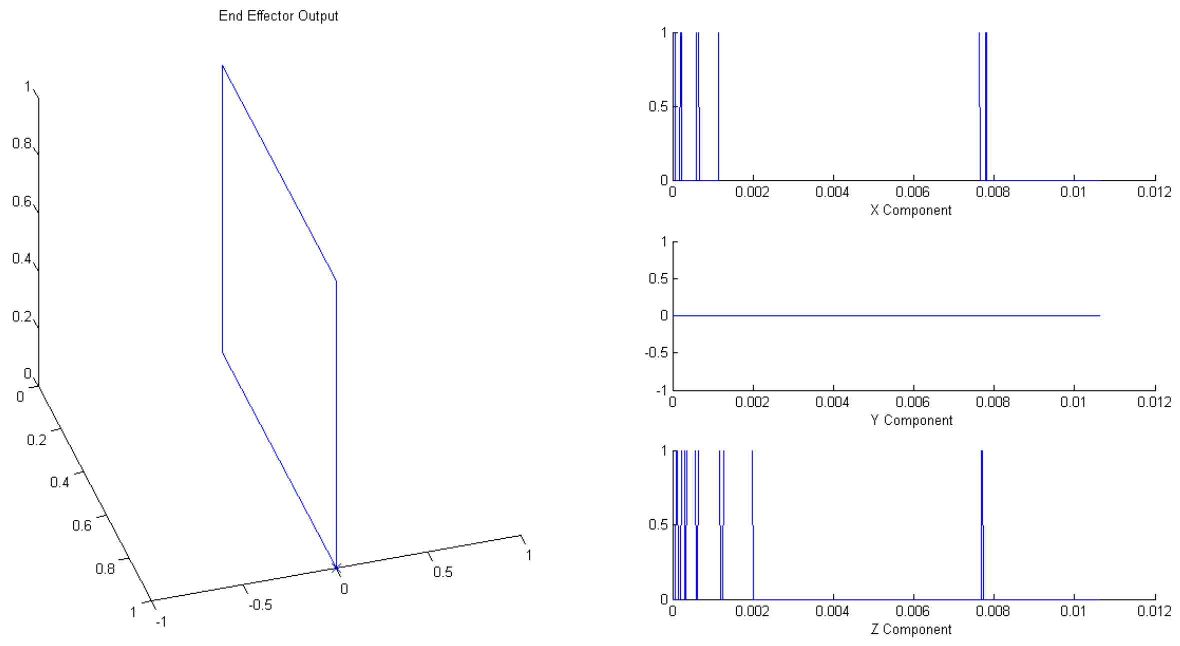


Figure 5.25: Path Following on Cube when Initialized on Surface

portion of the experiment traversing the meridian of the cube. This result is achieved with reasonable control effort, as seen in Figure 5.27.

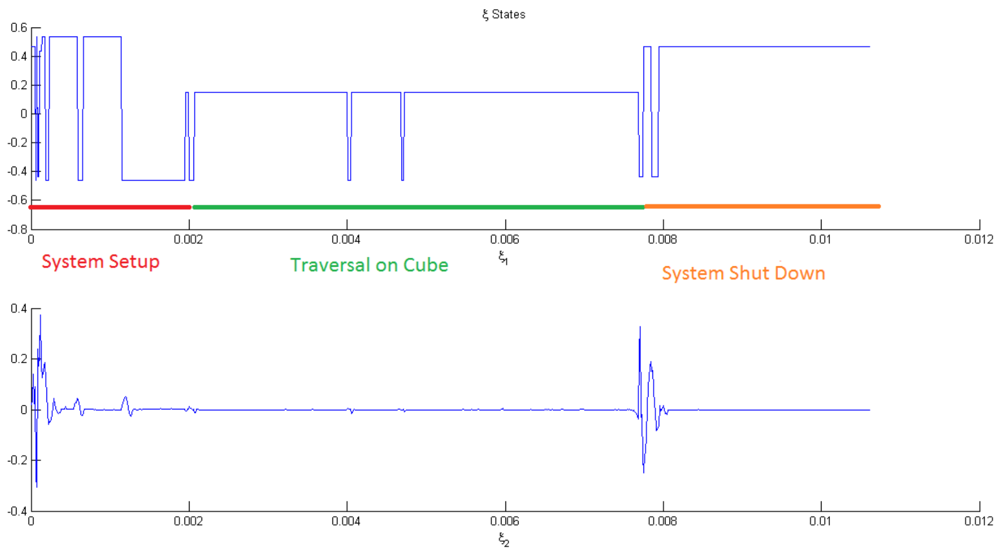


Figure 5.26: ξ dynamics for Cube stabilization with path following

Damped Motion

To conduct the tests for damped motion, the transverse and tangential controllers outlined in Section 5.2.2 were implemented on the experimental platform.

As with the simulation, the end effector of the manipulator was initialized off of the polytope. However, rather than have the manipulator initialized with an initial velocity, an external force was applied to test the damping abilities of the controller. Due to the construction of the manipulator, no external force in the horizontal direction could be applied, thus all external forces were applied in the vertical direction.

For this experiment, the end effector stabilized the surface of the polytope with the transverse controller. Once a facet of the polytope had been stabilized, the external forces were manually implemented to move the end effector in the vertical direction. The external forces were large enough to force the controller to switch facets. This switch occurred without causing any loss of stability in the system. The results of this experiment can be seen in Figure 5.28.

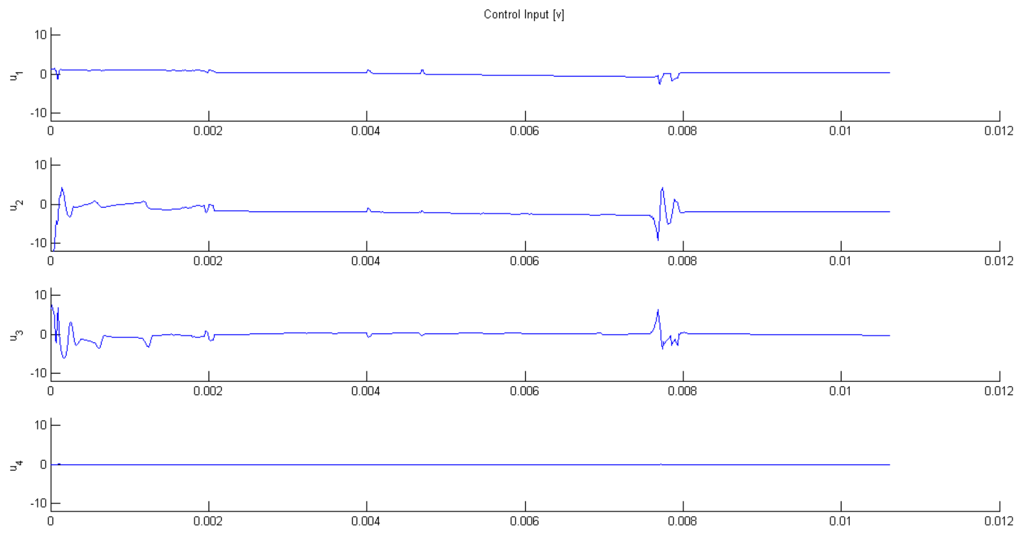


Figure 5.27: Control effort for Cube stabilization and path following

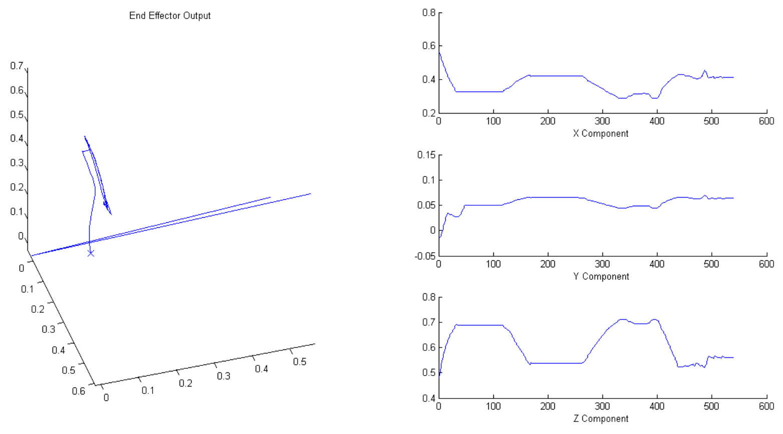


Figure 5.28: Experimental Results of Damped Motion on a Cube

Chapter 6

Conclusions and Future Work

Surface stabilization is a practical problem, with applications in robotics, haptics and telerobotics. Although there exists well defined algorithms and approaches for for stabilizing well defined smooth surfaces, the stabilization of complex polytope surfaces proves to be a challenging problem.

In this thesis, we consider a specific set of surfaces, namely convex polytopes. We stabilize the polytope by first stabilizing the individual facets of the polytope, rather than the entire surface. This allows the surface to be stabilized, and with a wise choice of switching algorithm, allows for movement along the surface.

We stabilize the facets by using transverse feedback linearization and a linear controller. Once the transverse dynamics have been stabilized, the tangential dynamics can be independently controlled, so long as they do not violate the conditions imposed by the switching algorithm. In this thesis we presented approaches to solve the path following problem, provide damped motion along the polytope surface and tracking of a trajectory on the surface. These algorithms were accompanied by simulations and experimental results demonstrating their feasibility.

The results of the analysis show that the polytope surface stabilization method allows for complex polytopes to be stabilized using only a simple representation of the facets and the switching algorithm. This is useful, as CAD programs generate surfaces using polytopes.

To provide the required information for path following applications, a considerable amount of data is required in the path definition. This additional complexity is not always possible if the desired surface is particularly complex, and does not allow for great flexibility.

Lastly, the effect of modelling uncertainty is rather profound, as seen in the experimental results. Without robust control methods, complex tangential tasks cannot be accomplished without great inaccuracy. Although the main criteria, such as surface invariance and practical asymptotic stability, are maintained, there still lies great room for improvement.

Future Work

The framework provided by the algorithm allows many different applications to be explored. However, there are also many refinements that can be made to the algorithm to increase the accuracy and efficiency of algorithm.

One method for increasing the efficiency of the algorithm is by using higher order equations for the facets of the surface. In this thesis we have represented the system with a series of hyperplanes. Hyperplanes allow for simplicity in understanding, as well as a simple coordinate transformation to obtain the dynamics along the surface. However, if the surface stabilization is extended to a higher order surface, for example a hyperboloid, then more complex curvature may be more accurately stabilized. In this application, rather than a simple natural projection, we would require a more complex mapping, as those outlined in [10].

Another major area for further development is the addition of robust control techniques. Although robust control techniques were briefly outlined in this thesis, and demonstrated in the case of transversal dynamics, they were not rigorously integrated into the other aspects of the controller. One of the major components of this algorithm is the reliance on accurate information. However, as observed in the experimental results, accurate and precise information is not always available. Thus the incorporation of robust control design into the surface stabilization algorithm would most likely provide great improvements to the experimental performance of the system.

The last aspect for further consideration is the investigation of other tangential control techniques. With proper feedback linearization, the tangential dynamics becomes a simple linear system. This opens up many different control techniques to be applied, as much of the nonlinear complexities have been removed. One particularly difficult area in the path following problem is the definition of the path information on the various facets of the approximated surface. Thus, to circumvent this problem, re-framing the path following problem as a form of reach control problem would allow for much of the complexity to be removed. One possible approach to this would be to reduce the facet size, so that each section of a desired path could be specified by entry and exit points of the facet alone. A reach controller could then be implemented to force the system output along the path, without the complication of defining the path on the facets. Given the work in [37], we see that the linearized tangential dynamics meet the necessary and sufficient conditions to apply the reach controllers.

The numerous possibilities outlined in the future work shows the potential for the algorithms presented in this thesis. Investigation of this future work is likely to provide even more insight and more opportunities to be explored.

Appendix A

Dijkstra's Algorithm

Dijkstra's algorithm can be broken down into the following steps [50].

Let the current node be denoted as the initial node. Let the distance from the current node to the target node be stored in a variable distance. The algorithm will assign initial some arbitrary distance value and try to improve it during each iteration.

1 Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.

2 Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes.

3 For the current node, consider all of its unvisited neighbours and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbour B has length 2, then the distance to B (through A) will be $6 + 2 = 8$. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.

4 When we are done considering all of the neighbours of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.

5 If the target node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited

set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

6 Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

Appendix B

Distance calculation methods

B.1 Closest distance to polytope calculation via projection in $\mathbb{R}^3$

Problem 3. Given an m-dimensional polytope $P \subset \mathbb{R}^p$ and a point $y \in \mathbb{R}^p$, determine the distance to the surface of the polytope

$$\|y\|_P = \inf_{p \in P} \|y - p\|.$$

This algorithm makes use of both polytope definitions outlined in Section 2.3. The objective of the algorithm is to determine the closest point between y and P. Using this information we calculate the distance to the surface using

$$\|y\|_p = \min\{\|y\|_{\mathcal{F}_1}, \dots, \|y\|_{\mathcal{F}_M}\}, \tag{B.1}$$

where $\|y\|_{\mathcal{F}_i}$ is the point-to-set distance to the ith facet of the polytope P.

For clarity, we present the algorithm for determining the point-to-set distance to facet i, which must be repeated M times in order to calculate (B.1).

Let $v \in \{v_0, \dots, v_m\} \cap \mathcal{F}_i$. The distance from y to the hyperplane $\{y \in \mathbb{R}^p : \langle a_i,\ y\rangle + b_i = 0\}$ is given by

$$d = \frac{1}{\|a_i\|} \left\langle y - v,\ \frac{a_i}{\|a_i\|} \right\rangle.$$

The point $y' \in \{y \in \mathbb{R}^p : \langle a_i,\ y\rangle = b_i\}$ closest to y is given by

$$y' = y - d\frac{a_i}{\|a_i\|}.$$

Next determine whether or not y' lies within the facet $\mathcal{F}_i$. Let, without loss of generality,

$$\{v_1, \dots, v_q\} = \{v_0, \dots, v_m\} \cap \mathcal{F}_i.$$

Define

$$\begin{aligned} e_{i-1,i} &:= v_i - v_{i-1}, \\ e_{i+1,i} &:= v_i - v_{i+1}, \end{aligned}$$

for $i \in \{1, \dots, q\}$. We assume that the vertices $v_1, \dots, v_q$ are arranged such that v_i is adjacent to v_{i-1} and v_{i+1} in the graph representation of the polytope.

Now construct vectors $f_1, \dots, f_m$ and $s_i, \dots, s_m$, to help determine the orientation of y' relative to the sides of the polytope.

$$\begin{aligned} s_i &:= \frac{e_{i-1,i}}{\|e_{i-1,i}\|} + \frac{e_{i+1,i}}{\|e_{i+1,i}\|} \\ f_i &:= \langle (s_i) \times (y' - v_i),\ a_i \rangle . \end{aligned}$$

These vectors are used to evaluate the position of y' relative to s_i, and determine the position of y' relative to a particular polytope vertex. For example, if $f_i > 0$ then y' is counter-clockwise to s_i, and if $f_i < 0$ then y' is clockwise to s_i. This is useful, as s_i originates from a vertex of the polytope, and therefore can be used to determine the orientation of y' relative to the polytope.

Next, we compare the vectors f_1 to f_m and attempt to find the region where $f_{i-1} < 0$ and $f_i > 0$ or $f_i < 0$ and $f_{i+1} > 0$ simultaneously. For simplicity, assume that the projected point falls in the region of $f_i < 0$ and $f_{i+1} > 0$, with other cases being handled similarly. This case is illustrated in Figure. B.1

Next we determine if y' lies within the polytope. The point y' lies outside of the polytope if the following inequality holds

$$\langle ((v_i - y') \times (v_{i+1} - y'))\,,\ a_i \rangle < 0.$$

If the point lies within the polytope, then the point to set distance is

$$\|y\|_{\mathcal{F}} = \|y - y'\|. \tag{B.2}$$

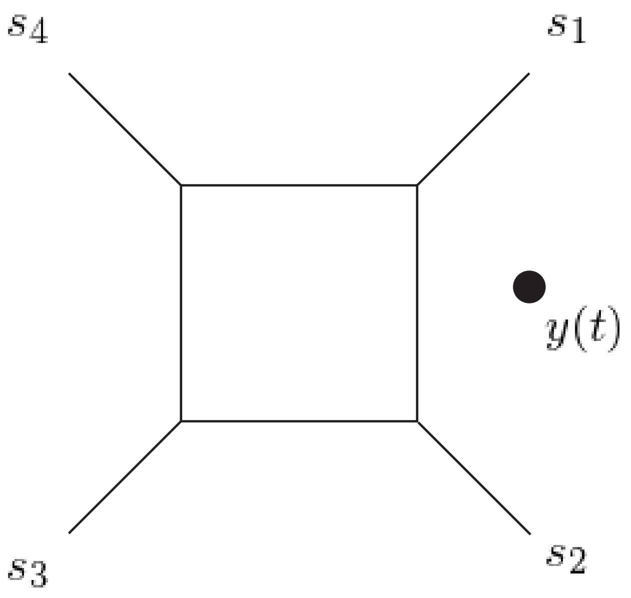


Figure B.1: Orientation of Vectors for Closest Distance Calculation

If the point is outside of the polytope, the vector orthogonal to the edge of the polytope must be found. This vector r can be calculated as

$$r = ((v_{i+1} - y') \times (v_i - y')) \times (v_{i+1} - v_i).$$

The point $y'' \in \{y \in \mathbb{R}^p : e_i + v_i = 0\}$ closest to y' is given by

$$y'' = y' - \frac{1}{\|r\|} \left\langle (v_i - y'),\ \frac{r}{\|r\|} \right\rangle \cdot \frac{r}{\|r\|}.$$

We determine if y'' lies within the interior of the two vertices of the edge by using the parametrized equation of the edge. We determine the parameter t as follows

$$t = \frac{y'' - v_i}{v_{i+1} - v_i}.$$

If $\|t\| < 0$, then y'' lies closest to v_i and the point to set distance can be calculated as

$$\|y\|_{\mathcal{F}_i} = \|y - v_i\|.$$

Similarly, if $\|t\| > 1$ then y'' lies closest to v_{i+1} and the point to set distance can be calculated as

$$\|y\|_{\mathcal{F}_i} = \|y - v_{i+1}\|.$$

However, if $0 < \|t\| < 1$ then the point to set distance can be calculated as

$$\|y\|_{\mathcal{F}_i} = \sqrt{\|y'' - y'\|^2 + \|y - y'\|^2}.$$

Algorithm B.1.1. *Given a point $y \in \mathbb{R}^p$, a facet $\mathcal{F}$ and its corresponding vertices $v_0, \ldots, v_m$, find the distance to the facet.*

```
    $\mathcal{F} = \{y \in \mathbb{R}^p : \langle a_i,\ y\rangle + b_i = 0\} \cap P$
 1: $d = \frac{1}{\|a_i\|} \left\langle y - v,\ \frac{a_i}{\|a_i\|} \right\rangle$
 2: $y' = y - d\frac{a_i}{\|a_i\|}$
 3: Let $\{v_1, \ldots, v_q\} = \{v_0, \ldots, v_m\} \cap \mathcal{F}_i$
 4: for $i = 1$to $q$ do
 5:     $e_{i-1,i} := v_i - v_{i-1},$
        $e_{i+1,i} := v_i - v_{i+1},$
 6: end for
 7: for $i = 1$to $m$ do
 8:     $s_i := \frac{e_{i-1,i}}{\|e_{i-1,i}\|} + \frac{e_{i+1,i}}{\|e_{i+1,i}\|}$
        $f_i := \langle (V_i) \times (y' - v_i),\ a_i \rangle .$
 9: end for
10: if $f_1 < 0$ and $f_2 > 0$ then
11:     if $\langle ((v_1 - y') \times (v_2 - y')),\ a_i \rangle < 0$ then
12:         $\|y\|_{\mathcal{F}} = \|y - y'\|$
13:     else
14:         $r = ((v_2 - y') \times (v_1 - y')) \times (v_2 - v_1)$
15:         $y'' = y' - \frac{1}{\|R\|} \left\langle (v_1 - y'),\ \frac{r}{\|r\|} \right\rangle \cdot \frac{r}{\|r\|}$
16:         $t = \frac{y'' - v_1}{v_2 - v_1}$
17:         if $t < 0$ then
18:             $\|y\|_{\mathcal{F}} = \|y - v_1\|$
19:         else if $t > 1$ then
20:             $\|y\|_{\mathcal{F}} = \|y - v_2\|$
21:         else
22:             $\|y\|_{\mathcal{F}} = \sqrt{\|y'' - y'\|^2 + \|y - y'\|^2}$
23:         end if
```

```
24:         end if
25:     end if
26:     ⋮
27:     if
28:         if   thenthen$\langle((v_i - y') \times (v_{i+1} - y'))\,,\ a_i\rangle < 0$
29:             $\|y\|_{\mathcal{F}} = \|y - y'\|$
30:         else
31:             $r = ((v_{i+1} - y') \times (v_i - y')) \times (v_{i+1} - v_i)$
32:             $y'' = y' - \frac{1}{\|r\|} \left\langle (v_i - y'),\ \frac{r}{\|r\|} \right\rangle \cdot \frac{r}{\|r\|}$
33:             $t = \frac{y'' - v_i}{v_{i+1} - v_i}$
34:             if $t < 0$ then
35:                 $\|y\|_{\mathcal{F}} = \|y - v_i\|$
36:             else if $t > 1$ then
37:                 $\|y\|_{\mathcal{F}} = \|y - v_{i+1}\|$
38:             else
39:                 $\|y\|_{\mathcal{F}} = \sqrt{\|y'' - y'\|^2 + \|y - y'\|^2}$
40:             end if
41:         end if
42:     end if
```

Once the minimum distance to each face is found, (B.1) can be determined and the hyperplane that corresponds to the intersecting facet can be selected as the closest facet.

However, if the closest point exists on multiple facets, e.g an intersection point, then an additional constraint must be applied to select the facet. In this case, the velocity $\dot{y}$ is found along each facet. The facet with the lowest corresponding velocity is then selected as the desired hyperplane.

Example B.1.1. Given a point

$$y^\star = \begin{bmatrix} 560 \\ 49 \\ -424 \end{bmatrix},$$

and square polytope defined by

$$P = \text{conv}\left\{ \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix}, \begin{bmatrix} 200 \\ 100 \\ 100 \end{bmatrix}, \begin{bmatrix} 100 \\ 200 \\ 100 \end{bmatrix}, \begin{bmatrix} 200 \\ 200 \\ 100 \end{bmatrix} \right\},$$

and spanned by the hyperplane

$$S(y) \left\{ y \in \mathbb{R}^3 : \left\langle \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top, \; y \right\rangle + 1 \right\},$$

we aim to determine the closest point between $y^\star$ and P.

We begin by calculating the distance d as

$$d = -524.06.$$

Using this, we calculate the projected point as

$$y' = \begin{bmatrix} 560.1 \\ 49 \\ 100 \end{bmatrix}.$$

Calculating vectors s_1 to s_4, we have

$$s_1 = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, s_2 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, s_3 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, s_4 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Using these vectors, we are able to construct vectors f_1 to f_4.

$$f_1 = \left\langle \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 460.1 \\ -51.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle = -511.1$$

$$f_2 = \left\langle \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 360.1 \\ -51.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle = -309.1$$

$$f_3 = \left\langle \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 460.1 \\ -151.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle = 309.1$$

$$f_4 = \left\langle \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \times \begin{bmatrix} 360.1 \\ -151.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle = 511.1.$$

Thus, the projected point lies in the section where $f_2 < 0$ and $f_4 > 0$.

We now determine if the projected point lies within the polytope, by evaluating the following inequality

$$\left\langle \begin{bmatrix} 560.1 \\ 49.0 \\ 100.0 \end{bmatrix} \times \begin{bmatrix} -360.1 \\ 150.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle = 3.6 * 10^4,$$

which indicates that the projected point lies outside of the polytope.

We now construct the vector r as follows

$$r = \left(\begin{bmatrix} -360.1 \\ 51.0 \\ 0.0 \end{bmatrix} \times \begin{bmatrix} -360.1 \\ 150.0 \\ 0.0 \end{bmatrix} \right) \times \begin{bmatrix} 0 \\ 100 \\ 0.0 \end{bmatrix} = 10^6 \begin{bmatrix} 3.6 \\ 0 \\ 0.0 \end{bmatrix}.$$

Using the vector r we project the point y' onto the ridge of the polytope.

$$y'' = \begin{bmatrix} 200.0 \\ 49.0 \\ 100.0 \end{bmatrix}.$$

We now calculate the parameter t to determine if the project point lies within the interior of the vertices

$$t = \frac{\begin{bmatrix} 200.0 \\ 49.0 \\ 100.0 \end{bmatrix} - \begin{bmatrix} 200.0 \\ 200.0 \\ 100.0 \end{bmatrix}}{\begin{bmatrix} 200.0 \\ 100.0 \\ 100.0 \end{bmatrix} - \begin{bmatrix} 200.0 \\ 200.0 \\ 100.0 \end{bmatrix}} = 1.51.$$

This indicates that the projected point lies outside of the polytope, and is closest to the point

$$\begin{bmatrix} 200.0 \\ 200.0 \\ 100.0 \end{bmatrix}.$$

Thus the distance between the point y and the polytope P is simply

$$\|y\|_P = \left\| \begin{bmatrix} 560 \\ 49 \\ -424 \end{bmatrix} - \begin{bmatrix} 200.0 \\ 200.0 \\ 100.0 \end{bmatrix} \right\|.$$

♦

$\triangle$

B.2 Closest distance to a polytope via convex optimization

Given a point $y \in \mathbb{R}^p$ we wish to determine the minimum distance to a polytope P. First determine if the point y lies within the interior of the polytope or its exterior. This can be accomplished by substituting y into the half plane equations of the polytope

$$P = \{y \in \mathbb{R}^p : \forall i \in \{1, \ldots, k\}, \langle a_i,\ y\rangle \leq b_i\},$$

where there exists an integer $k \geq p + 1$, non-zero vectors $a_1, \ldots, a_k \in \mathbb{R}^p$ and scalars $b_1, \ldots, b_k \in \mathbb{R}$.

If

$$\exists i \in \{1, \ldots, k\}, \langle a_i,\ y\rangle - b_i < 0$$

then we can determine that the point y lies within the interior of the polytope, and the point to set distance to the polytope is simply

$$\min(| \langle a_i,\ y\rangle - b_i | \forall i \in \{1, \ldots, k\}).$$

However, if the point y lies outside of the polytope, we can apply a convex optimization approach to determine the point to set distance. We apply the following parameters to a convex optimization solver to determine the closest point on the polytope.

$$\begin{gathered}\min(\|y - \lambda \sum_{i=0}^{m} v_i\|) \\ \text{subject to} \sum(\lambda) = 1, \\ \lambda \geq 0\end{gathered}$$

Once the closest point has been found, the distance between the point y and the point on the polytope is found, and the facet corresponding to the point is selected as the closest hyperplane. As with the projection case, if the point exists on multiple facets, then the velocity of the system at point y is found in reference to each of the facets and the facet with the lowest corresponding velocity is selected as the desired hyperplane.

This result is summarized as follows

Algorithm B.2.1. *Given a point $y \in \mathbb{R}^p$ and a polytope $P \in \mathbb{R}^p$, find the closest facet.*

```
1:  for $i = 1 to\ k+1$ do
2:      if $i = k+1$ then
3:          break;
4:      end if
5:      if $\langle a_i,\ y\rangle - b_i < 0$ then
6:          Closest Facet = $\mathcal{F}_i$
7:          return;
8:      end if
9:  end for
10: if $i = k+1$ then
11:     $\min(\|y - \lambda\,\{v_0, \ldots, v_m\}\,\|)$
12:     subject to $\sum(\lambda) = 1, \lambda \geq 0$
13:     $MinDist = \|y - (\lambda) * v_1, \ldots, v_m\|$
14:     $J = find(MinDist = \|A((\lambda) * v_1, \ldots, v_m) + b\|)$
15:     if $size(J) > 1$ then
16:         $ClosetFacet = min(L_f(h(x)), \forall x \in J)$
17:     else
18:         $ClosetFacet = J$
19:     end if
20: end if
```

References

[1] F. Albertini and E.D. Sontag. Continuous control-lyapunov functions for asymptotically controllable time-varying systems. *Int. J. Control*, 72:1630–1641, 1999.

[2] A. Bacciotti and L. Mazzi. An invariance principle for nonlinear switched systems. *Systems and Control Letters*, 2005.

[3] A. Banaszuk and J. Hauser. Feedback linearization of transverse dynamics for periodic orbits. *Systems and Control Letters*, 26(2):95–105, Sept. 1995.

[4] A. Banaszuk and J. Hauser. Feedback linearization of transverse dynamics for periodic orbits in $\mathbb{R}^3$ with points of transverse controllability loss. *Systems and Control Letters*, 26(3):185–193, Oct. 1995.

[5] E.A. Barbashin. The method of sections in the theory of dynamical systems. *Mat. Sbornik*, 29:501–518, 1951. (Russian).

[6] C. Belta, L.C.G.J.M. Habets, and V. Kumar. Control of multi-affine systems on rectangles with applications to hybrid biomolecular networks. In *Proceedings of the 41th IEEE Conference on Decision and Control*, pages 534–539, Las Vegas, Nevada, Dec. 2002.

[7] C. Belta, V. Isler, and G. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21:864–874, 2004.

[8] N.P. Bhatia and G.P. Szegö. *Dynamical Systems : Stability theory and applications.* Springer-Verlag, Berlin, 1967.

[9] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.

[10] W. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry.* Academic Press, 1975.

[11] V.A. Bushenkov and G.V. Smirnov. Stabilization of sets. *Journal of Computer and Systems Sciences International*, 32(3):17 – 34, 1994.

[12] A. Das, M. Diu, N. Matthew, C. Schafenberger, J. Servos, A. Wong, J. Zelek, D. Clausi, and S. Waslander. Mapping, planning and sample detection strategies for autonomous exploration. *Journal of Field Robotics*, 31:75–106, 2013.

[13] A. Das, J. Servos, and S. Waslander. 3d scan registration using the normal distributions transform with ground segmentation and point cloud clustering. In *2013 IEEE International Conference on Robotics and Automation*, pages 2207–2212, Karlsruhe, May. 2013.

[14] C. Canudas de Wit, H. Olsson, and K.J Astrom. A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, 40(3):419–425, 1995.

[15] C. Canudas de Wit and O.J. Sordalen. Exponential stabilization of mobile robots with nonholonomic constraints. In *IEEE Transactions on Automatical Control*, 1992.

[16] M. El-Hawwary and M. Maggiore. Global path following for the unicycle and other results. In *2007 American Control Conference*, pages 3500–3505, Seattle, WA, USA, June 2008.

[17] E. Farzad and H. Khalil. Output feedback stabilization of fully linearizable systems. *International Journal of Control*, 56(5):1007–1037, 1992.

[18] B.A. Francis and W. M. Wonham. The internal model principle for linear multivariable regulators. *Applied Mathematics and Optimization*, 2(2):170–194, 1975.

[19] D. Gates. Boeing using robots to boost 777 output. The Seattle Times, May 2013.

[20] R. Gill, D. Kulic, and C. Nielsen. Robust path following for robot manipulators. In *2013 IEEE/RSJ International Conference on Intelligent Robots*, pages 3412 – 3418, Tokyo, November 2013.

[21] A. Girard and S. Martin. Motion planning for nonlinear systems using hybridizations and robust controllers on simplices. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 239–244, Cancun, Dec. 2008.

[22] O. Golubitsky, V. Mazalov, and S. Watt. An algorithm to computer the distance from a point to a simplex. Technical report, Western Universty, 2012.

[23] J. Guldner and V. Utkin. Sliding mode control for gradient tracking and robot navigation using artifical potential fields. *IEEE Transactions on Robotics and Automation*, 1995.

[24] L.C.G.J.M. Habets, P.J. Collins, and J.H. van Schuppen. Control to facet by piecewise-affine output feedback. *IEEE Transacations on Automatic Control*, 2012.

[25] L.C.G.J.M. Habets and Jan H. van Schuppen. Control of piecewise-linear hybrid systems on simplices and rectangles, 2001.

[26] L.C.G.J.M. Habets and J.H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 2003.

[27] A. Hladio. Path following for mechanical systems applied to robotic manipulators. Master's thesis, University of Waterloo, 2010.

[28] A. Hladio, C. Nielsen, and D. Wang. Path following for a class of mechanical systems. *IEEE Transactions on Control Systems Technology*, 21:2380–2390, 2013.

[29] A. Isidori. *Nonlinear Control Systems.* Springer, New York, third edition, 1995.

[30] M. Jones. 3d distance from a point to a triangle. Technical report, University of Wales Swansea, 1995.

[31] M. Kermani, R. Patel, and M. Moallem. Friction identification and compensation in robotic manipulators. *IEEE Transactions on Instrumentation and Measurement*, 2007.

[32] H. Khalil. *Nonlinear Systems.* Prentice Hall, 2002.

[33] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[34] M. Krstic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and Adaptive Control Design.* Wiley Interscience, 1995.

[35] V. Lakshmikantham, S. Leela, and A. A. Martynyuk. *Practical Stability of Nonlinear Systems.* World Scientific Publishing Co. Pte. Ltd, 1990.

[36] Y. Lin, E.D. Sontag, and Y. Wang. Input to state stabilizability for parameterized families of systems. *Intern. J. Robust & Nonlinear Control*, 5:187–205, 1995.

[37] Z. Lin and M. Broucke. Reachability and control of affine hypersufrace systems on polytopes. In *Proceedings of the 46th IEEE Conference on Decision and Control New Orleans*, pages 733 – 738, New Orleans, LA, USA, Dec 2007.

[38] M.Wu, G. Yan, Z. Lin, and Y. Lan. Synthesis of output feedback control for motion planning based on ltl specications. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5071 – 5075, St. Louis, MO, USA, Oct 2009.

[39] C. Nielsen. Local transverse feedback linearization with partial information for single-input systems. *SIAM J. Control and Optimization*, 2014. Accepted.

[40] C. Nielsen and M. Maggiore. On local transverse feedback linearization. *SIAM J. Control and Optimization*, 47(5):2227–2250, 2008.

[41] H. Nijmeijer and G. Campion. Feedback linearization of parameterized nonlinear time-varying systems. In *Proceedings of the* 2^{nd} *European Control Conference*, pages 31 –36, Groningen, NLD, June 1993.

[42] H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer - Verlag, New York, 1990.

[43] P. Persson. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, June 2004.

[44] R. Tyrrell Rockafellar. *Convex Analysis.* Princeton Landmarks, 1970.

[45] B. Roszak and M. Broucke. Necessary and sufficient conditions for reachability on a simplex. In *44th IEEE Conference on Decision and Control*, pages 4706 – 4711, Seville, Dec 2005.

[46] J. Schutter, D. Torfs, H. Bruyninckx, and S. Dutre. Invariant hybrid force/position control of a velocity controlled robot with compliant end effector using modal decoupling. *The International Journal of Robotics Research*, 1997.

[47] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Motion and Control.* John Wiley and Sons, 2006.

[48] T. Tran, T. Chung, H. Kim, S. Kim, and M. Oh. Trajectory tracking of mobile manipulator for welding task using sliding mode control. In *30th Annual Conference of the IEEE Industrial Electronics Society*, pages 407–412, Busan, Nov 2004.

[49] K. Walker. *Surface Geometry and the Haptic Rendering of Rigid Point Contacts*. PhD thesis, University of Waterloo, 2013.

[50] S. Waslander. Me 597: Lecture notes. Technical report, University of Waterloo, 2011.

[51] W. Wonham. *Linear Multivariable Control: A Geometric Approach*. Springer-Verlag, 1985.

[52] C. Yang, Q. Zhang, and L. Zhou. Practical stabilization and controllability of descriptor systems. *International Jounral of Information and Systems Sciences*, 5(2):138–144, 2008.

[53] W. Yim and S.N. Singh. Sliding mode force and motion control and stabilization of elastic manipulator in the presence of uncertainties. In *IEEE International Conference on Robotics and Automation*, pages 2113 – 2118, San Diego, CA, USA, May 1994.

[54] G. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1994.

[55] V.I. Zubov. *Methods of A.M. Lyapunov and their Application*. Groningen, The Netherlands, 1957. English Translation: P. Noordhoff 1964.