STABILIZATION OF DISCRETE-TIME SYSTEMS WITH BOUNDED CONTROL INPUTS

by

Anes Jamak

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of

Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2000

To my Parents and Sisters.

Table of Contents

List of Figures

Abstract

In this paper we examine the stabilization of LTI discrete-time systems with control input constraints in the form of saturation nonlinearities. This kind of constraint is usually introduced to simulate the effect of actuator limitations.

Since global controllability can not be assumed in the presence of constrained control, the controllable regions and their characterizations are analyzed first. We present an efficient algorithm for finding controllable regions in terms of their boundary hyperplanes (inequality constraints). A previously open question about the exact number of irredundant boundary hyperplanes is also resolved here.

The main result of this research is a time-optimal nonlinear controller which stabilizes the system on its controllable region. We give an algorithm for on-line computation of control which is also implementable for high-order systems. Simulation results show superior response even in the presence of disturbances.

Acknowledgements

I would like to thank Prof. Daniel Miller, my supervisor, for his constant support and invaluable guidance during this research.

Chapter 1

Introduction

The study of controlling and stabilizing LTI discrete-time systems with magnitude constraints on the control inputs has been the subject of research for a long time. Since the magnitude of all practical control inputs are bounded due to actuator limitations, research in this area is highly practically motivated. Even for the systems where a large magnitude of control can be tolerated, achieving given control objectives with a minimum possible magnitude of control is desirable and can significantly reduce the cost of the actuator and the overall control system. This class of systems we are considering is usually modeled by introducing a saturation nonlinearity in the feedback configuration which makes the originally linear system a nonlinear one. As a consequence many results from linear system theory cannot be applied here.

Global controllability can not be assumed in the presence of the constrained control and the concept of constrained controllability and controllable regions is introduced instead. There has been a lot of effort to characterize the *null controllable region* C, which is the set of states that can be steered to the origin in a finite number of steps using constrained control as well as the *k-step controllable region* $C(K)$, which is the set of states that can be steered to the origin in K steps. The problem of finding controllable sets $C(K)$ for LTI discrete-time system was first formulated by Kalman [18] in 1957. The very first results regarding the problem were given by Desoer and Wing [6, 7, 8] where the K-step controllable set is described by its vertices. Later in 1970, Lin [22] introduced the so-called facial representation of $C(K)$ which characterizes it in terms of its boundary hyperplanes (faces). From that time on, it became convenient in related control literature [2, 9, 11, 13, 17, 19, 21, 26, 29, 32, 14, 15] to describe $C(K)$ by its vertices or by facial representation which is usually computed via a combination of the Fourier-Moltzkin projection algorithm and linear programming techniques used for the elimination of inactive inequalities. As K is increasing the complexity of the controllable set is rapidly increasing. The computation of $C(K)$ based on the previously described method becomes computationally very intensive and for high order systems it is practically infeasible. This is mainly the consequence of successive applying the Fourier-Moltzkin projection algorithm which results in a great number of inactive (redundant) inequalities what is known in literature as combinatorial explosion. An interesting results and simplifications for the special case of second order system are given in [12, 20].

In Chapter 2 we analyze controllable sets in an unusual way by observing its geometrical properties based on the theory of polytopes [34]. Using the observation that $C(K)$ is a zonotope (affine projection of cube in a hyperdimensional space) a surprisingly simple new result is presented which enables the direct computation of a facial representation of $C(K)$ without generating redundant inequalities which is a significant improvement to existing solutions. We give an algorithm in pseudocode for the computation of $C(K)$ in the general case which runs in polynomial time, i.e. $O(K^{(n-1)})$ where n is the order of system. A previously open question about the exact number of facets in $C(K)$ is also resolved.

Once the controllable sets for a given system are known, the natural next step is to design a stabilizing controller on a large subset of C (or hopefully on the whole null-controllable region C). Typically, C is approximated with $C(K)$ with K sufficiently large. One of the first stabilizing controllers on $C(K)$ is proposed by Desoer and Wing [6, 7, 8]. It is based on switching surfaces which separates positive and negative control. However, the calculation of switching surfaces is very hard to implement except for the case of second order system. After that, Cwikel and Gutman in [13] introduced controllers based on vertex representation of controllable sets and Lin [22] proposed an alternative approach based on the facial representation of $C(K)$. Computation in both cases is very difficult and the controller is hard to implement in the real world. From that time there have been a lot of new results and most of them can be classified as either vertex or facet based control laws. In summary, several efficient solutions have been proposed for semi-stable systems; however, for anti-stable systems which have poles outside the unit disk, the current results are very limited and in the general case there is the same computational problem as in dealing with controllable sets. For the special case of systems with only real poles a nice result is given in [14, 15].

In Chapter 3 we give a nonlinear state feedback controller for single input anti-stable systems that drives the state $x \in C(K)$ to the origin in minimum time subject to control input constraints. First we characterize all minimum time feedback control laws and between all those we propose that one which is also optimal with respect to disturbance rejection and robustness. This idea of minimum time control is not new and a similar approach can be found in Keerthi and Gilbert [19]. They also give a characterization of all minimum time controls, but that characterization is not further used to choose the control law which is also optimal regarding some other optimization criterion.

The key difference between our approach presented here and existing similar ideas, e.g. Keerthi and Gilbert, is a new way for the on-line computation of control. Based on geometrical observations we show that the time-optimal control input can be found by calculating the intersection of a line with a zonotope. In Chapter 4 we discuss how this computational problem can be efficiently solved. We present two solutions: one which make use of facial representation and another based on linear programming techniques and the dual simplex method. We show that the first approach using new results developed in Chapter 2 can be efficiently implemented on low cost computer architecture even for relatively high-order systems (e.g. the class of systems with 5 unstable poles). In the second approach we show that the computation of optimal control can be restated as a bounded variable linear program. We have shown that using the dual-simplex method this solution based on LP is much more efficient than the solution based on facial representation. More precisely, we have shown that any solution based on the facial representation of a controllable set (and its precomputation) is equivalent to solving a corresponding bounded variable linear program in the most naive approach by examining all its extreme points. Knowing that many existing results in the literature are based on a ”vertex” or ”facial” approach, this is very surprising result.

The main results of this paper are given in the previously described Chapters 2-4. In Chapter 5 we describe how ideas developed in Chapter 3 for single input system can be generalized to multi-input systems and systems with both stable and unstable poles. In Chapter 6 we give two examples where we describe design procedure and give close-loop response. Finally, Chapter 7 summarizes the results of the thesis and possibilities for future research are presented.

Chapter 2

Characterization of Controllable Regions

In this chapter we consider controllable sets of linear, discrete-time system when there is saturation on the control input. After introducing the problem, we design an algorithm which calculates the facial representation of the so-called K-step controllable set without generating inactive inequalities. The algorithm presented is an improvement to the existing solutions based on the Fourier-Moltzkin projection algorithm and linear programming. An open problem about exact number of faces of the controllable set is also resolved here.

2.1 Mathematical Preliminaries

The discrete-time system we are considering is of the form

$$x(k+1) = Ax(k) + Bsat[u(k)] \tag{2.1.1}$$

where $x(k) \in R^n$ is the state vector and $u(k) \in R^m$ is the control input. We also assume that matrix pair (A, B) is controllable (see C.T. Chen [4]) . The function $sat()$ models a saturation nonlinearity and is defined by

1. For $u_i \in R$

$$sat(u_i) = \begin{cases} u_i & \text{for } u_i \in [-1, 1] \\ +1 & \text{for } u_i > 1 \\ -1 & \text{for } u_i < -1 \end{cases}$$

2. For $u = [u_1 \dots u_m]^T \in R^m$

$$sat(u) = [sat(u_1) \dots \, sat(u_m)]^T$$

From the definition of the saturation function it follows that $\|sat(u(k))\|_\infty \leq 1$. In the presence of such a bounded control, global controllability of the state x cannot be assumed. Thus our first concern is to find the set of states that can be steered to the origin using a constrained control u. To that end we introduce the following definitions.

Definition 2.1. (K-step Controllable State)
A state x_0 is said to be *controllable at a given step K* if there exists a control u such that the time response x of the system (2.1.1) satisfies $x(0) = x_0$ and $x(K) = 0$.

Definition 2.2. (K-step Controllable Region)
The set of all states controllable at step K is called the *controllable region at step K* and is denoted by $C(K)$ or C_K.

Definition 2.3. (Controllable Region)
A state x_0 is said to be *controllable* if there exists a control u such that the time response x of the system (2.1.1) satisfies $x(0) = x_0$ and $x(k) \to 0$ as $k \to \infty$. The set of all such states is called the *controllable region* and is denoted by C.

From the above definitions it is straightforward to prove that the following proposition holds.

Proposition 2.4. *The controllable region C is given by*

$$C = \cup_{i=1}^{\infty} C(K).$$

The problem we are solving in this chapter is how to find a simple, exact description (characterization) of a K-step controllable region $C(K)$. Later in Chapter 3 and Chapter 5 we will show that for practical purposes the controllable region C can be well approximated by $C(K)$ for some sufficiently large K. We start with an algebraic representation of $C(K)$.

2.2 Algebraic Representation

Define the set $[-1, 1]^m$ by

$$[-1, 1]^m = \{u \in R^m \mid u_i \in [-1, 1],\ i = 1, \dots ,\ m\}. \tag{2.2.1}$$

The system (2.1.1) can be written equivalently as

$$x(k+1) = Ax(k) + Bu(k), \qquad u(k) \in [-1, 1]^m \tag{2.2.2}$$

In the above system the effect of saturation nonlinearity is replaced with the condition $u(k) \in [-1, 1]^m$. Applying the equation (2.2.2) recursively it can be easily shown that the following equalities hold:

$$\begin{aligned} x(1) &= Ax(0) + Bu(0) \\ x(2) &= A^2x(0) + ABu(0) + Bu(1) \\ &\vdots \\ x(K) &= A^Kx(0) + \sum_{i=0}^{K-1} A^iBu(K-1-i). \end{aligned} \tag{2.2.3}$$

It is quite simple to obtain an algebraic formula for a K-step controllable region $C(K)$ using equations (2.2.3). Obviously

$$C(0) = \{0\}.$$

Next, suppose that the state $x_0 \in C(1)$. By the definition of $C(1)$ there exists $u(0) \in [-1,1]^m$ such that $x(0) = x_0$ and $x(1) = 0$. Substituting this into (2.2.3) we have

$$x_0 = -A^{-1}Bu(0), \quad u(0) \in [-1,1]^m.$$

Thus, each point(state) in $C(1)$ has the above form and the whole set $C(1)$ can be obtained as $u(0)$ ranges over $[-1,1]^m$. We can write it simply

$$C(1) = \left\{-A^{-1}Bu(0) \mid u(0) \in [-1,1]^m\right\}.$$

Similarly, each point $x_0 \in C(2)$ can be represented in the form

$$x_0 = -A^{-2}Bu(1) - A^{-1}Bu(0), \;\; u(0), u(1) \in [-1,1]^m$$

and consequently

$$C(2) = \left\{-A^{-2}Bu(1) - A^{-1}Bu(0) \mid u(0), u(1) \in [-1,1]^m\right\}.$$

The algebraic formula for a representation of the K-step controllable region $C(K)$ is given in the next proposition.

Proposition 2.5. *(Algebraic Representation of $C(K)$)*

A K-step controllable region $C(K)$ is given by

$$C(K) = \left\{ -\sum_{i=0}^{K-1} A^{-(i+1)} B u(i) \;\middle|\; u(i) \in [-1,1]^m,\ i = 0, \ldots, (K-1) \right\}. \tag{2.2.4}$$

Proof. The proof follows from a straight-forward application of mathematical induction. $\square$

We use the following notation regarding algebraic representation of $C(K)$. Define the matrix W_K by

$$W_K = -[(A^{-1}B)\ (A^{-2}B)\ \ldots\ (A^{-K}B)]. \tag{2.2.5}$$

Throughout this paper we will use the matrix W_K very often. Observe from Proposition 2.5 that if x_0 is a point in $C(K)$ then there exists $U = [u(0)\, u(1) \ldots u(K-1)]^T \in [-1,1]^{mK}$ such that

$$x_0 = W_K U.$$

Henceforth, such a representation of the state x_0 in terms of control inputs is labeled a *U-decomposition* of x_0.

2.3 Geometric Representation

In the previous section we developed in a simple way an algebraic formula for a description of $C(K)$. However from the (2.2.4) it is very hard to tell anything about the properties of a controllable region, about its geometrical structure and, for example to check whether some state x_0 is inside $C(K)$. Obviously an alternative representation of $C(K)$ is desirable.

In this section we want to examine geometrical properties of K-step controllable sets. In the general case, $C(K)$ is a subset of R^n and we are dealing here with geometry in a hyperdimensional space. It is interesting that many results which are true in 2-D and 3-D space cannot be simply generalized to the hyperdimensional space and sometimes relying only on intuition can lead us to wrong conclusions. For that reason we will insist on mathematical completeness in the further text. Many ideas and the new results presented in this paper are based on some geometrical observations. The reader who wants to learn more about geometry in hyperdimensional space is referred to [34], [10], [25] and [33].

We start with some terms and notation from combinatorial and algebraic geometry and convexity theory.

2.3.1 Geometry Preliminaries and Notation

We introduce the following notation. For a fixed non-zero vector $u \in R^n$ and a scalar $a \in R$, the set $H = \{x \in R^n \mid \langle x, u \rangle = a\}$ is called a *hyperplane*. Here $\langle x, u \rangle = u^T x$ denotes the scalar product of the vectors x and u. The sets

$$H^+ = \{x \in R^n \mid \langle x, u \rangle \geq a\} \text{ and } H^- = \{x \in R^n \mid \langle x, u \rangle \leq a\}$$

are called *half-spaces* and are bounded by H. Observe that H is an $(n-1)$ dimensional set and $R^n = H^+ \cup H^-$. The vector u from the above definition is orthogonal to each vector laying in the hyperplane H and henceforth we call it *the outer normal (inner normal)* of H if u points to H^+ (H^-).

Definition 2.6. (Convex Hull)

For any set $K \subseteq R^n$, the smallest convex set containing K is called the *convex hull* of K, and denoted by *conv(K).*

Definition 2.7. (V-Polytope)

A *V-polytope* is a set obtained as the convex hull of finite set of points in R^n. If d is the dimension of the smallest subspace containing a *V-polytope* then we say that V is a d-dimensional polytope in R^n.

Definition 2.8. (H-Polytope)

An *H-polyhedron* is the intersection of a finite number of half-spaces in R^n. A bounded *H-polyhedron* is called an *H-polytope.*

Observe that each *H-polytope* is an intersection of finite number of hyperplanes and can be written in the form

$$P = \{x \in R^n \mid \langle x, u_1 \rangle \leq z_1,\ \langle x, u_2 \rangle \leq z_2,\ \ldots\ \langle x, u_k \rangle \leq z_k\}.$$

With $F = [u_1\ u_2\ ...\ u_k]^T \in R^{k \times n}$ and $z = [z_1 ...\ z_2]^T \in R^k$ we can write this more compactly as

$$P = \{x \in R^n \mid Fx \leq z\} = P(F, z)$$

Definition 2.9. (Polytope)

A set $P \subset R^n$ is a polytope if P is either a *V-polytope* or *H-polytope.*

To illustrate the above definitions consider the following examples. A triangle or any polygon in the plane are two dimensional polytopes in R^2. Points are 0-dimensional polytopes and line segments are 1-dimensional polytopes. We know from the Euclidean geometry that each polygon P in R^2 can be represented by its vertices or as the intersection of half-planes corresponding to its edges. This means that a polygon $P \subset R^2$ is both a *V-polytope* and an *H-polytope.* This result can be extended to the general n-dimensional case, which we state bellow.

Theorem 2.10. .

A polytope $P \subset R^n$ is a V-polytope *if and only if it is a* H-polytope.

Remark 2.1. While this theorem seems trivial and almost obvious in a two dimensional space, it is not simple to prove in the general case. A complete proof can be found in [34] and is based on the Fourier-Moltzkin projection algorithm and Farkas Lemma.

As a consequence of the previous theorem each polytope P can be written as $P = conv(V)$ and $P = P(F, z)$.

Definition 2.11. (Supporting hyperplane)
A hyperplane H is called a *supporting hyperplane* of a polytope P if $H \cap P \neq \emptyset$ and either $P \subset H^-$ or $P \subset H^+$. In a geometrical sense this means that the hyperplane H is tangent to P.

Definition 2.12. (Faces of Polytope)
If H is a supporting hyperplane of a polytope P then the intersection $H \cap P$ is said to be a *face* of polytope.

We can classify the faces of a polytope P by their dimesion. Thus if $\mathcal{F}$ is a face of the polytope P and $dim(\mathcal{F}) = k$ then $\mathcal{F}$ is denoted as a *k-face* of P. If $\mathcal{F}$ is a k-face of P then $\mathcal{F}$ is called a *vertex* for $k = 0$, an *edge* for $k = 1$ and a *facet* for $k = dim(P) - 1$.

Next, we give some basic properties of polytopes collected in several propositions. Many of these properties are quite simple to prove, so -the proofs are omitted or given in short. Detailed proofs can be found in [34, 33, 25]

Proposition 2.13. *Let $\mathcal{F}$ be a face of a polytope $P \subset R^n$ and let H be a supporting plane of $\mathcal{F}$ with its outer normal u. Then $\mathcal{F}$ is the set of all points in P which maximizes the function $\langle x, u \rangle$ on the set P.*

Proposition 2.14. *(Vertex Representation)*
Let $vert(P)$ denote the set of all vertices of a polytope $P \subset R^n$. Then $P = conv(vert(P))$ i.e. every polytope is the convex hull of its vertices.

Proposition 2.15. *(Facial Representation)*
Let $P = P(F, z) \subset R^n$ be an n-polytope *and suppose that there are no inactive inequalities in the system $Fx \leq z$. Then there is exactly one inequality for each facet of P.*

As a consequence of the last proposition it can be concluded that every polytope is the intersection of its facet-defining halfspaces (inequalities).

Definition 2.16. (Minkowski Sum)
For the two sets $P, Q \in R^n$, the set $P + Q = \{p + q \,|\, p \in P,\ q \in Q\}$ is called the *Minkowski sum* of P and Q.

Proposition 2.17. *Let P, Q be polytopes in R^n and $T : R^n \to R^m, S : R^{2n} \to R^n$ be linear operators. Then following holds*

1. *The Minkowski sum $P + Q$ of two polytopes is a polytope in R^n.*

2. *The projection $P' = TP$ is a polytope in R^m.*

3. *The projection of a direct product of two polytopes is a Minkowski sum of the projections: $S(P \times \emptyset_n) + S(\emptyset_n \times Q) = S(P \times Q)$.*

2.3.2 Zonotopes

In our further analysis we are particularly interested in a special kind of polytopes, the so-called zonotopes. As it will be shown, controllable regions $C(K)$ are actually centrally symmetric zonotopes. Henceforth, we call the set $[-1,1]^s$ a hyperdimensional cube.

Definition 2.18. (Zonotope)
A zonotope $Z \in R^n$ is an affine projection of the cube $[-1,1]^s$ from the space R^s to R^n. That is, with $W = [w_1 \dots w_s] \in R^{n \times s}$, $z \in R^n$, Z is of the form

$$Z = W[-1,1]^s + z = \left\{ x \in R^s \mid x = z + \sum_{i=1}^{s} t_i w_i, \ -1 \leq t_i \leq 1 \right\}.$$

Observe that an s-dimensional cube $[-1,1]^s$ can be represented as product of one-dimensional cubes (line segments) $[-1,1]^s = [-1,1] \times \dots \times [-1,1]$. Since W is a linear operator, using Proposition 2.17 we have that

$$\begin{aligned} W[-1,1]^s &= W([-1,1] \times \dots \times [-1,1]) \\ &= w_1([-1,1]) + ... + w_s([-1,1]) \\ &= [-w_1, w_1] + \dots + [-w_s, w_s] \end{aligned}$$

where w_i is the $i-th$ column of the matrix W. Each $[-w_i, w_i]$ defines line segment in R^n, thus $W[-1,1]^s$, and consequently the zonotope $Z = W[-1,1]^s + z$, can be represented as a Minkowski sum of line segments in R^n.

2.3.3 Controllable Region is Zonotope

Recall that the controllable region $C(K)$ can be written in the form

$$C(K) = -(A^{-1}Bu_0 + A^{-2}Bu_1 + \dots + A^{-k}Bu_{k-1})$$

where each $u_i = [-1, 1]^m$. Now define the $n \times mK$ matrix W to be

$$W = -[A^{-1}B\ A^{-2}B \ldots A^{-K}B] = [w_0\ w_1 \ldots w_{mK-1}]$$

where w_i is the $i-th$ column of matrix W. With this notation we have three equivalent expressions for $C(K)$:

$$C(K) = \left\{ \sum_{i=0}^{mK-1} w_i c_i \ : \quad c_i \in [-1, 1] \right\}$$
$$C(K) = W[-1, 1]^{mK}$$
$$C(K) = [-w_0, w_0] + \ldots + [-w_{mK-1}, w_{mK-1}]$$

This means that $C(K)$ can be represented as a set of bounded linear combinations of vectors w_i; a projection of a hyperdimensional cube; and a Minkowski sum of line segments. Obviously $C(K)$ is a zonotope. In what follows we will show how this fact together with some geometrical properties of zonotopes can be efficiently used to find facial representation of controllable region $C(K)$.

2.4 Finding Facial Representation

In this section we describe how to find directly a facial representation of the controllable region $C(K)$ without generating inactive inequalities. In related control literature it is usual to use the Fourier-Moltzkin projection algorithm and then to eliminate inactive inequalities using some linear programming techniques, see [19]. However this approach is extremely inefficient since the elimination of inactive inequalities in this case is computationally equivalent to solving a great number (usually an exponential number) of linear programs.

Our idea for finding facial representation presented in the theorem below and Section 2.4 is much simpler and provides a significant improvement to the previous approach.

Theorem 2.19. *(Main Theorem)*
A vector f is an outer normal of a facet $\mathcal{F}$ of the controllable region C_K if and only if there exist $(n-1)$ linearly independent vector-columns $w_{i_1}, w_{i_2}, \ldots, w_{i_{(n-1)}}$ of the matrix W which are orthogonal to the vector f.

Proof. Let $\mathcal{F}$ be a facet and f be a vector which is an outer normal of $\mathcal{F}$. Let us define the scalar M to be

$$M = \max\{f^T x : x \in C_K\}.$$

According to Proposition 2.13 the facet $\mathcal{F}$ is given as the set

$$\mathcal{F} = \{x \in C_K : f^T x = M\}.$$

Each point $x \in C_K$ can be written in the form $x = \sum_{i=1}^{mK} c_i w_i$ for the appropriate choice of $c_i \in [-1, 1]$ so consequently

$$\begin{aligned} f^T x &= f^T \sum_{i=1}^{mK} c_i w_i \\ &= \sum_{i=1}^{mK} c_i (f^T w_i). \end{aligned}$$

Suppose we want to maximize this summation. Since each summand in the above sum is independent of the others we can do that by maximizing each of the summands separately. Obviously if $(f^T w_i)$ is positive (negative) it is best to take c_i as large (small) as possible. Thus we take

$$c_i = -1 \quad \text{if } i \in I^- := \big\{ i \mid f^T w_i < 0 \big\}$$

$$c_i = +1 \quad \text{if } i \in I^+ := \big\{ i \mid f^T w_i > 0 \big\}$$

$$c_i \in [-1, 1] \quad \text{if } i \in I^0 := \big\{ i \mid f^T w_i = 0 \big\}$$

and it is clear that for such a choice of c_i's, $(f^T x)$ achieves a maximum on the set C_K.

This means that the facet $\mathcal{F}$ can be described by

$$\begin{aligned} \mathcal{F} &= \Big\{ -\sum_{i \in I^-} w_i + \sum_{i \in I^+} w_i + \sum_{i \in I^0} c_i w_i \ \mid \ c_i \in [-1, 1], \ \forall i \in I^0 \Big\} \\ &= \Big\{ z + \sum_{i \in I^0} c_i w_i \ \mid \ c_i \in [-1, 1], \ \forall i \in I^0 \Big\}. \end{aligned}$$

By definition the facet $\mathcal{F}$ is an $(n-1)$-dimensional face of a C_K and from the last equality we may conclude that so is the set

$$Z = \Big\{ \sum_{i \in I^0} c_i w_i \ \mid \ c_i \in [-1, 1], \ \forall i \in I^0 \ \Big\}.$$

Observe that Z is the set of all bounded linear combinations of the vectors w_i and in order for Z to be an $(n-1)$ dimensional set there must be exactly $(n-1)$ linearly independent vectors in the set $\{w_i : i \in I^0\}$. But we know from the definition of the set I^0 that $f^T w_i = 0$, $i \in I^0$, i.e. each such w_i is orthogonal to vector f. By this we have proved the first part of the theorem.

To prove the other way, let's suppose that $w_{i_1}, w_{i_2}, ..., w_{i_{(n-1)}}$ are $(n-1)$ linearly independent vector-columns arbitrarily chosen from matrix W. Choose a normalized vector f orthogonal to vectors w_{i_1}, $w_{i_2}, \ldots, w_{i_{(n-1)}}$. (Observe that there are exactly two such f's of opposite directions). Next, we will prove that f is the outer normal of some facet in C_K.

We can define sets I^-, I^+, I^0 in the same way as it is done in the first part of the proof. Now, define $\mathcal{F}'$ to be

$$\mathcal{F}' = \left\{ -\sum_{i\in I^-} w_i + \sum_{i\in I^+} w_i + \sum_{i\in I^0} c_i w_i \mid c_i \in [-1,1],\ \forall i \in I^0 \right\}$$

Then following is true about the set $\mathcal{F}'$

- $\mathcal{F}' \subseteq C_K$ since each $x \in F'$ is of the form $\sum_{i=1}^{mK} c_i w_i$ with $c_i \in [-1,1]$.
- $\mathcal{F}'$ is $(n-1)$ dimensional, since vectors $w_{i_1}, w_{i_2}, \ldots w_{i_{(n-1)}}$ are linearly independent
- $\mathcal{F}'$ is the set of all $x \in C_K$ which maximizes function $(f^T x)$ on the set C_K (by construction of sets I^-, I^+, I^0)

Consequently, by Proposition 2.13 $\mathcal{F}'$ is a facet of the set C_K. □

Although quite simple in nature, the last theorem give us a powerful method for the straightforward computation of outer normal vectors of all facets in C_K without generating redundant vectors (inequalities) as is the case with the Fourier-Moltzkin projection algorithm. Once we have all of the outer normal vectors of C_K collected in the matrix F it is very simple to find the right hand side in the system of inequalities $Fx \leq z$ defining C_K and thus obtain complete facial representation of the set C_K. Moreover we can make use of this theorem to find the exact number of facets in C_K.

If we define M as the set of all $n \times (n-1)$ dimensional minors M_i of the matrix W such that $rank(M_i) = (n-1)$ then the last theorem states that for each minor

$M_i \in M$ there exist two normalized vectors f and $-f$ such that $f^T M_i = [0 \ldots 0]$ and at the same time f and $-f$ are outer normal vectors to some facets in C_K. Further, since C_K is a convex set, for each outer normal vector exactly one facet of C_K can be associated. Hence, for each minor $M_i \in M$ there are two corresponding facets. We can summarize this in the following theorem

Theorem 2.20. *(Exact Number of Facets in C_K)*
Let M be the set of all $n \times (n-1)$ dimensional minors of the matrix W such that

- $rank(M_i) = n - 1, \ \forall M_i \in M$
- $rank([M_1 \ M_2]) = n, \ \forall M_1, M_2 \in M$

Then the number of facets in C_K is equal to $2\,|M|$. (Here $|M|$ denotes the number of elements in the set M)

Proof. The second condition, $\text{rank}([M_1 \ M_2]\ ) = n$ tell us that there is no vector f such that $f^{\,T}[M_1 \ M_2] = [0\ 0... \ 0]$ or equivalently there is no vector f such that $f^{\,T} M_1 = 0$ and $f^{\,T} M_2 = 0$. This further means that there are no two minors from M with the same normal vector. Now the statement of the theorem is the direct consequence of the previous theorem. □

Since the number of all $n \times (n-1)$ dimensional minors of the matrix W is $C_{mK}^{n-1} = \binom{mK}{n-1}$ the following theorem holds.

Theorem 2.21. *(Upper Bound Theorem)*
The number of facets in C_K is less than or equal to $2\binom{mK}{n-1}$.

Proof. This theorem follows directly from the Main Theorem. □

Remark 2.2. Although we call this theorem the 'upper bound theorem' it is likely that for arbitrarily chosen matrices A, B the number of facets of the controllable region C_K will be exactly $2\binom{mK}{n-1}$. Here is why. Loosely speaking we can think in the following way. Let W_i be the matrix containing the first i columns of W. Each combination of $(n-2)$ vectors from W_i spans a subspace in which those vectors lies and the dimension of such a subspace is no greater than $(n-2)$. We can construct exactly $\binom{i}{n-2}$ subspaces in such way. Denote the union of all such subspaces as S_u. Next suppose we want to append a new randomly chosen vector $w_{(i+1)}$ to matrix W_i. What is the chance that vector w_{i+1} together with some $(n-2)$ vector-columns from W_i forms a minor of rank less than $(n-1)$? This is equivalent to the question: what is the probability that w_{i+1} lies in S_u? Observe that S_u is the union of a finite number of subspaces of dimension less than or equal to (n-2). Consequently, S_u is not even dense in R^n and if w_{i+1} is randomly chosen then the answer to the previous question is that the 'probability of such an event is zero'. Of course, in our case the vector-column w_{i+1} is not randomly chosen but obtained as a column of the matrix $A^{-L}B$ for some L. However, the matrix A^{-L} has full rank and maps to all space R^n and we can expect that this will be also true in our case. Actually, examining several pairs of matrices A and B from the real world, the author observed that the number of facets is exactly $2\binom{mK}{n-1}$ as we supposed. Of course we can always construct special matrices A and B such that strict inequality holds (e.g., let the matrix A represent a rotation) but in the general case we should expect $2\binom{mK}{n-1}$ to be the number of facets.

2.4.1 Algorithm In Pseudocode

Now we provide an algorithm to put C_K into the form $\{\, x \mid Fx \leq z\}$. It should be clear from previous discussion how to find the matrix F. Each row in matrix F is the outer normal vector to some facet in the zonotope C_K and we described the way to find all of the outer normal vectors from the matrix W. Let's see how to find right hand side in the system of inequalities $Fx \leq z$. Consider the inequality $f_i x \leq z_i$ corresponding to $i - th$ facet of the zonotope C_K. This inequality is satisfied for all vectors $x \in C_K$ and the equality holds for all points on the $i - th$ facet. That is, z_i can be obtained as

$$z_i = \max \{f_i x \mid x \in C_K\}$$

It should be noted that we already solved this problem in the proof of the Main Theorem. A closed-form expression for z_i is given by

$$z_i = \sum_{j=1}^{mK} |f_i w_j| \,,$$

which is straightforward to compute.

As a part of our algorithm we need also to generate all combinations of $(n-1)$ elements of the set $\{1, 2, \ldots, mK\}$. It is a simple algorithm and we are not going to bother with it here. We will use the notation $C(mK, (n-1))$ to denote the set of all such combinations and we assume that $C(mK, (n-1))$ is ordered.

Now, let us summarize all previous results in the algorithm given in pseudocode.

Input: Matrix W

Output: Matrix pair (F,z)

```
1. Set $F = [\,]$; $z = [\,]$;
2. For i=1 to mK do
   (2.1) Generate the i-th combination $\{w_{i_1}, w_{i_2}, ...w_{i_{(n-1)}}\}$
         of the set $C(mK, (n-1))$.
   (2.2) Set $W_i = [w_{i_1}\ w_{i_2}\ \ldots w_{i_{(n-1)}}]$.
   (2.3) If $rank(W_i) = n-1$ then
         2.3.1. Find a vector $f$ as a solution of the linear system $f^T W_i = [0\,0 \ldots 0]$,
         2.3.2. Normalize the vector $f$.
         2.3.3. If $f$ is not already included as a row in the matrix $F$ then
            2.3.3.1. Set $F = \begin{bmatrix} F \\ f^T \end{bmatrix}$.
            2.3.3.2. Find $z_i = \sum_{j=1}^{mK} |f_i w_j|$.
            2.3.3.3. Set $z = \begin{bmatrix} z \\ z_i \end{bmatrix}$.
         2.3.4. end If
   (2.4) end If
3. end For
4. Set $F = \begin{bmatrix} F \\ -F \end{bmatrix}$; $z = \begin{bmatrix} z \\ z \end{bmatrix}$.
5. Return $(F, z)$.
```

Observe that the running time of the presented algorithm is $O(n^3\binom{mK}{n-1})$ and as per discussion after the 'Upper Bound Theorem' the expected memory requirements for storing (F, z) are $O(n\binom{mK}{n-1})$.

Chapter 3

Nonlinear State Feedback Controller

In this chapter we present a time-optimal nonlinear controller based on a characterization of the controllable regions given in Chapter 2.

3.1 Problem Formulation

Recall from Chapter 2 that the discrete-time model of the plant we are considering is given by

$$x(k+1) = Ax(k) + Bsat[u(k)]; \quad A : R^n \to R^n;\ B : R^m \to R^n;\ u(k) \in R^m$$

We also defined the set $C = \cup_{K=1}^{\infty} C_K$ as the controllable region of upper system. Assuming that the state vector $x(k)$ can be measured and the initial state $x_0 \in C$, our goal is to find the state feedback control law which drives x_0 to zero in a finite number of steps. Of course it is to our advantage that x_0 be steered to the origin as fast as possible. Let us define the so-called minimum time problem.

Minimum-Time Problem. Given the state $x_0 \in C$, define $t_{\min}$ as the minimum possible time for which state x_0 can be driven to zero. According to the definitions of the controllable regions C_K, $t_{\min}$ is given by

$$t_{\min} = \min \{K \mid x_0 \in C_K\}.$$

The minimum time problem can be stated as follows: For some state $x_0 \in C$ find its corresponding $t_{\min}$ and the control signal u such that $x(0) = x_0$ and $x(t) = 0$ for all $t \geq t_{\min}$.

In this chapter we will describe a nonlinear state feedback control law which solves the problem for each $x_0 \in C_K$. Knowing that the set C can be well approximated by C_K for large enough K we can consider that the presented controller is a solution to the minimum time problem. The idea behind this control low is quite simple and similar ideas can be found in literature. However, unlike some other solutions which are computationally very intensive and hard to implement we will show in Chapter 4 that the presented controller can be implemented in the real world even for high order systems.

For the time being we assume that our model is a single input system ($m = 1$) and all eigenvalues of the matrix A are unstable, i.e. all eigenvalues of A lie outside the unit disk in the complex plane. Later in Chapter 5 we will show how concepts developed for this special case can be extended to the more general case: multi-input systems and systems with both stable and unstable eigenvalues.

3.2 Approximation of the Controllable Region C

Since we are assuming that the magnitude of all eigenvalues of A are greater than one, it follows that all eigenvalues of A^{-1} are inside the unit disk. Now consider the matrix

$$W = [A^{-1}B \ \dots \ A^{-K}B] = [w_1 \dots w_K].$$

Since $m = 1$ by assumption, each vector w_i is given by $w_i = A^{-i}B$ and it follows that

$$\|w_i\|_1 \leq \left\|A^{-i}\right\| \|B\|_1 ;$$

here, $\|A^{-i}\|$ defines the corresponding induced norm on the matrix A^{-i}. Since all eigenvalues of A^{-1} have a magnitude less than 1, the norm $\|A^{-i}\|$ exponentially converges to zero as $i \to \infty$. Observe also that B is a constant vector, so from the last inequality we can conclude that $\|w_i\|_1$ exponentially converges to zero as $i \to \infty$. What does it mean? We said that controllable region C_K is the Minkowski sum of the line segments $C_K = \sum_{i=1}^{K}[-w_i, w_i]$. Alternatively, we can write C_K as

$$C_K = C_{K-1} + [-w_K, w_K].$$

If w_K is very small, the difference between the sets C_K and C_{K-1} is also very small. Loosely speaking, we can say that C_K exponentially converges to C as $K \to \infty$. To observe this trend consider the following example.

Example. For $K = 10$ and matrices

$$A = \begin{bmatrix} 1.6 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

the corresponding matrix W is

$$W_{10} = \begin{bmatrix} -0.62 & -0.39 & -0.24 & -0.15 & -0.09 & -0.06 & -0.04 & -0.02 & -0.01 & -0.01 \\ 0.67 & -0.44 & 0.29 & -0.19 & 0.13 & -0.09 & 0.06 & -0.04 & 0.03 & -0.02 \\ -0.50 & -0.25 & -0.12 & -0.06 & -0.03 & -0.02 & 0.01 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

In this case C_{10} will be a good approximation of the controllable region C.

Thus, for practical purposes it is enough to find a minimum time control law for initial states in C_K where K is 'large enough'. In real systems K will usually be between 10 and 30, assuming that sampling time is well-chosen.

3.3 Characterization of All Minimum Time Control Laws

Once we have introduced controllable regions it is quite simple to characterize all minimum time control laws. Assume that the initial state x_0 belongs to a controllable region C_K and assume also that K is the minimum time with respect to x_0. Applying the control input u_0, after one step the new state vector will be x_1. If u_0 is an optimal control input (generated by a minimum time control law) then obviously, starting from x_1 we can reach the zero-state in $(K-1)$ steps, i.e. $x_1 \in C_{K-1}$. Following this logic we may conclude that if $[u_0 \, u_1 \, \dots \, u_{K-1}]$ is a minimum-time control signal then for $i \in [1, \dots, K-1]$ the control input u_i is such that $Ax_i + Bu_i =: x_{i+1} \in C_{K-i-1}$. Thus, we can find a minimum-time state-feedback control law by solving the following problem.

Given the sets C_K and C_{K-1}, find a mapping $g : R^n \to [-1, 1]$ such that

$$Ax + Bg(x) \in C_{K-1}, \; x \in C_K$$

Next, we want to prove that this problem always has a solution.

Proposition 3.1. *For each $x \in C_K$, there exists a control input $u \in [-1, 1]$ such that*

$$Ax + Bu \in C_{K-1}$$

Proof. Suppose that $x \in C_K$. According to the definition of the controllable region C_K, there exist $c_i \in [-1, 1]$ such that

$$x = \sum_{i=1}^{K} c_i w_i = \sum_{i=1}^{K} c_i A^{-i} B, \quad c_i \in [-1, 1];\ i \in [1..K]$$

It follows that

$$\begin{aligned} Ax &= A \sum_{i=1}^{K} c_i A^{-i} B \\ &= c_1 B + A \sum_{i=2}^{K} c_i A^{-i} B \\ &= c_1 B + \sum_{i=1}^{K-1} c_{i+1} A^{-i} B \end{aligned}$$

Define the control input u to be $u = -c_1$ and the last equation yields

$$Ax + Bu = Ax - Bv_1 = \sum_{i=1}^{K-1} c_{i+1} A^{-i} B$$

, but $\sum_{i=1}^{K-1} c_{i+1} A^{-i} B$ is obviously point in C_{K-1}, so

$$Ax + Bu \in C_{K-1}$$

□

The previous proposition proves the existence of a control input which drives any state vector x from C_K to C_{K-1}, although the proof is non-constructive. Let's see how we can find such a control input. The idea we will present here is based on some geometrical observations.

3.3.1 Nonlinear Mapping $g(x)$

Assume that $x_r \in C_K$ is the state vector at the step r of discrete time. According to the previous proposition the set $\Omega(x_r)$ of all optimal control inputs u_r at step r is

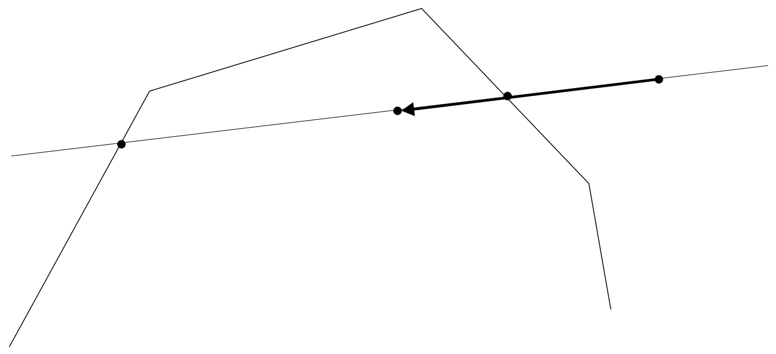

Figure 3.1: Intersection of the line L with region $C(K-1)$.

given by

$$\Omega(x_r) = \{u \in [-1,1] \mid Ax_r + Bu \in C_{K-1}\}.$$

Consider the set

$$L(x_r) = \{Ax_r + Bu \mid u \in [-1,1]\}.$$

Obviously the set $L(x_r)$ defines a line segment in R^n centered at the point Ax_r and parallel to the vector B. Alternatively, the set $\Omega(x_r)$ can be written as

$$\Omega(x_r) = \{u \in R \mid Ax_r + Bu \in (L(x_r) \cap C_{K-1})\}.$$

This means that optimal control inputs are indirectly determined by an intersection of the line segment $L(x_r)$ with the zonotope C_{K-1} (Figure 3.1).

From Chapter 2 we know that a zonotope is a convex and bounded set and geometrically it is clear that the intersection $L(x_r) \cap C_{K-1}$ will be a line segment and consequently all optimal control inputs will be within a segment of the form $[u_{\min}, u_{\max}] \subseteq [-1,1]$. The following proposition gives more details.

Proposition 3.2. *(Optimal Control Inputs)*

Given a state vector $x_r \in C_K$*, define the scalars* $\lambda_{\min}$ *and* $\lambda_{\max}$ *by*

$$\lambda_{\min} = \min \{\lambda \in R \mid Ax_r + B\lambda \in C_{K-1}\}$$

$$\lambda_{\max} = \max \{\lambda \in R \mid Ax_r + B\lambda \in C_{K-1}\}$$

Then the set $\Omega(x_r) = \{u \in [-1,1] \mid Ax_r + Bu \in C_{K-1}\}$ *of all optimal control inputs is given by*

$$\Omega(x_r) = [u_{\min},\, u_{\max}] = [\max\{-1, \lambda_{\min}\},\ \min\{1, \lambda_{\max}\}].$$

Proof. Define the straight line

$$L = \{\lambda \in R : Ax_r + B\lambda\}.$$

Since the set C_{K-1} is a convex and bounded set, then the set $T = L \cap C_K$ is also convex and bounded. Further it is obvious that T must be a line segment of the form $[x_{\min},\, x_{\max}]$ with boundary points determined by the intersection of the line L with the boundary of the set C_{K-1}. That is,

$$x_{\min} = Ax_r + B\lambda_{\min}$$

$$x_{\max} = Ax_r + B\lambda_{\max}$$

This means that T can also be given by

$$T = \{Ax_r + B\lambda \mid \lambda \in [\lambda_{\min}, \lambda_{\max}]\}\,. \tag{3.3.1}$$

By the definition of $\Omega(x_r)$ we know that the following holds:

$$\{Ax_r + B\lambda \mid \lambda \in \Omega(x_r)\} \subseteq C_{K-1} \text{ and}$$

$$\{Ax_r + B\lambda \mid \lambda \in \Omega(x_r)\} \subseteq L$$

and consequently

$$\{Ax_r + B\lambda \mid \lambda \in \Omega(x_r)\} \subseteq T.$$

The last equation, together with the definition of $\Omega(x_r)$ and 3.3.1 gives

$$\Omega(x_r) = [\max\{-1, \lambda_{\min}\},\ \min\{1, \lambda_{\max}\}]$$

$\square$

The last Proposition shows that all optimal control inputs at step r can be characterized by two boundary points $\lambda_{\min}$ and $\lambda_{\max}$. It is reasonable to suppose that it will not be computationally much harder to find these then it is to find only one optimal control input u_r. If we do so then we have the advantage of being able to choose that control input, among all minimum time control inputs, which is also optimal regarding some other optimizing criterion, e.g. optimal disturbance rejection or minimum power consumption. Such a control law is presented in the next section.

3.4 Proposed Nonlinear Control

From the previous section it should be clear that minimum-time control is typically not unique. For each step r and corresponding state vector x_r, it is enough to choose any $u_r \in \Omega(x_r)$ to satisfy the minimum time criterion. Between all of those optimal controls, we are interested in finding the one which is also optimal in the sense of robustness and disturbance rejection. The idea behind the control law presented in this section is quite simple and is motivated from geometrical observations.

3.4.1 Geometrical Observations

Suppose that the disturbance input w is superimposed on the control input u. The discrete-time system can now be written as

$$x_{k+1} = Ax_k + Bsat(u_k + w_k)$$

The last equation can be split into two parts, one with no noise and another which captures the effect of the noise:

$$\begin{aligned} x_{(k+1)d} &= Ax_k + Bsat(u_k) \\ \delta x_{k+1} &= Ax_k + Bsat(u_k + w_k) - Bsat(u_k) \qquad (3.4.1) \\ x_{k+1} &= x_{(k+1)d} + \delta x_{k+1}. \end{aligned}$$

Let's suppose that $x_k \in C_K$ and $u_k \in \Omega(x_k)$. The optimality criterion guarantees that $x_{(k+1)d} \in C_{K-1}$. However, due to presence of δx_{k+1} it is possible that the new state vector x_{k+1} may not be inside the controllable region C_{K-1}. Thus, the negative effect of the disturbance can be characterized as a tendency to move the state vector out of the set C_{K-1}. Of course, to avoid this it is best to choose the control input such that $x_{(k+1)d}$ is as 'deep' inside the set C_{K-1} as is possible. Now we'll explain what 'deep' means.

Observe that as the control input u_{k-1} ranges over $[\lambda_{\min}, \lambda_{\max}]$ (see the previous subsection for the definition of $\lambda_{\min}$ and $\lambda_{\max}$) the vector $x_{(k+1)d}$ moves along part of the line segment $L(x_{\min}, x_{\max})$ which is determined by two boundary points:

$$x_{\min} = Ax_k + B\lambda_{\min} \quad \text{and} \quad x_{\max} = Ax_k + B\lambda_{\max}$$

The same holds for the disturbance input and the perturbation δx_k can be considered to be a displacement of the state vector $x_{(k+1)d}$ along line L (Figure 4.2).

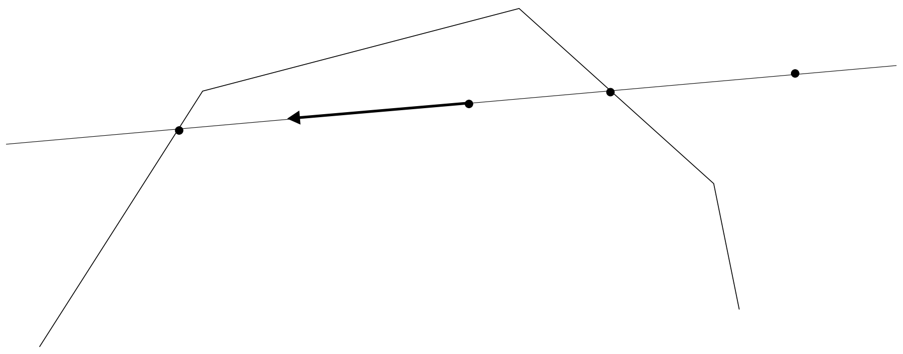

Figure 3.2: Displacement of state vector x_{k+1} due to perturbation δx_{k+1}.

Obviously if we want to keep the state vector x_{k+1} inside the line segment $[x_{\min}, x_{\max}]$ even after a displacement due to the disturbance, it is best to place $x_{(k+1)d}$ in the middle of the line segment $[x_{\min},\, x_{\max}]$ at the point $x_M = (x_{\min} + x_{\max})/2$. This state vector can be obtained after applying the control input $u_k = (\lambda_{\min} + \lambda_{\max})/2$. It should be noticed that due to the constraint on the control input $u_k \in [-1,\, 1]$ it may not be possible that the new state vector reaches the point x_M. However, in that case our optimal strategy is to place the new state vector $x_{(k+1)d}$ as close to the point x_M as possible. Obviously the control input u_k which achieves this goal is given by

$$u_k = \begin{cases} (\lambda_{\min} + \lambda_{\max})/2 & \text{if } (\lambda_{\min} + \lambda_{\max})/2 \in [-1, 1] \\ -1 & \text{if } (\lambda_{\min} + \lambda_{\max})/2 < -1 \\ 1 & \text{if } (\lambda_{\min} + \lambda_{\max})/2 > 1 \end{cases}$$

or we can write it simply

$$u_k = sat[(\lambda_{\min} + \lambda_{\max})/2]$$

As we can see, the closed form for our 'double-optimal' control is quite simple and depends only on $\lambda_{\min}$ and $\lambda_{\max}$, which can be determined looking at the intersection of line and a zonotope. However, observe that if $k < n$, the zonotope C_k will not be n-dimensional and balanced, and the intersection of the line with the zonotope degenerates into only one point. This further means that if the current state vector is inside the region $C_k,\ k \leq n$ there is only one control input which drives the state vector to the region C_{k-1}. In reality with the presence of disturbances and numerical errors it would be impossible to reach C_{k-1} and the control algorithm may loop infinitely. Obviously, once when the state vector is inside C_n we need an alternative strategy to bring it 'close' to the zero-state and we should not follow exactly the path $C_n \rightarrow C_{n-1} \rightarrow \ldots C_0 = 0$. To overcome this problem the following control strategy can be used.

3.4.2 Control Strategy

Assume that we start from an initial state $x_0 \in C_K,\ K > n$. Our control strategy consists of two stages.

In the first stage our goal is to move state vector x to the controllable region C_n. To do this we use the 'double-optimal' control described previously. Thus, if $x_r \in C_k$ is currently the state vector then we use the control input $u_r = sat[(\lambda_{\min} + \lambda_{\max})/2]$. Since $u_r \in \Omega(x_r)$ satisfies the minimum-time criterion the new state vector x_{r+1} should be inside the region C_{k-1} (If this is not case, for example due to the disturbance, then the previous step is repeated until the state vector is inside C_{k-1}). Repeating this procedure with the new state vector x_{r+1} and so on, the controllable region C_n can be reached in a finite number of steps.

The second stage of the control law admits a state vector inside the controllable region C_n and the goal is to drive it to zero by applying a sequence of n control inputs. The controllable region C_n corresponds to $n \times n$ square matrix

$W_n = -[A^{-1}B\, A^{-2}\, \dots A^{-n}B]$. We assumed that matrix pair (A, B) is controllable, so by definition of controllability W_n has full rank. Consequently, for a given state vector $x_r \in C_n$ there is a unique solution $[u_1\, \dots u_n]^T$ of linear system

$$W_n[u_1\, \dots u_n]^T = x_r.$$

This means that x_r can be uniquely represented in the form $x_r = \sum_{i=1}^{n} u_i w_i$ and all entries of vector $[u_1 \dots u_n]^T$ satisfies the input constraint $u_i \in [-1, 1]$, $i = 1 \dots n$. The last result implies that the vector $[u_1\, \dots u_n]^T$ defines a u-decomposition(see section 2.1) of the state vector x_r and by applying a sequence of control inputs $u_1, u_2, \dots u_n$ we drive the state vector to zero. The second stage control strategy can be simply summarized as follows.

Precompute the matrix W_n^{-1}. For a given state $x_r \in C_n$ find $[u_1\, \dots u_n]^T = W_n^{-1}x_r$ and apply the control inputs $u_1, u_2, \dots u_n$. If the state vector is displaced from the zero-state due to disturbances, repeat the previous procedure for a new state-vector.

Next we give an algorithm in pseudocode which implements this control strategy.

```
Input: Measured state $x_r$
Output: Control Signal

1. Find min $\{k \mid x_r \in C_k\}$
2. If $k \leq n$ then
.        2.1 Set $[u_r,\, u_{r+1}, \dots u_{r+n-1}] = W_n^{-1}x_r$
.        2.2 Return control sequence $u_r,\, u_{r+1}, \dots u_{r+n-1}$
.  Else
.        2.3 Find $\lambda_{\min} = \min\{\lambda \in R \mid Ax_r + B\lambda \in C_{k-1}\}$
.        2.4 Find $\lambda_{\max} = \max\{\lambda \in R \mid Ax_r + B\lambda \in C_{k-1}\}$
.        2.5 Return $u_r = sat[(\lambda_{\min} + \lambda_{\max})/2]$
EndIf
```

In order to implement this optimal algorithm in real world we need to solve two serious computational problems.

- How to find the intersection of a line with a zonotope, i.e. $\lambda_{\min}$ and $\lambda_{\max}$.
- How to check whether a state vector x_r is inside some controllable region C_k.

Since the complexity of controllable regions rapidly increases with increasing K and the system order n, an efficient solution of these computational problems is key for practical implementation. For that reason we devoted a separate Chapter 4 to describe in detail two different approaches for solving the aforementioned problems.

3.5 Example of State Trajectory

For better insight into previously described control strategy we give an example of the state trajectory for a second order system (Figure 3.3).

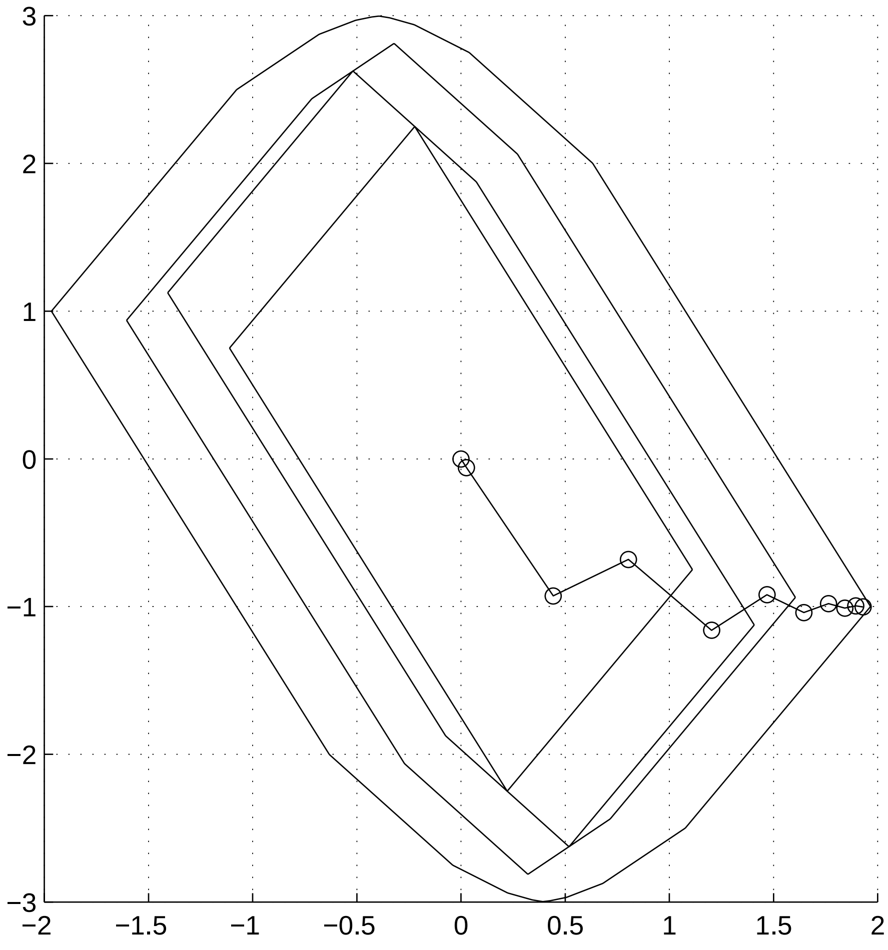

Figure 3.3: State trajectory of a second order sytem

Chapter 4

Computational Issues

In this chapter we describe two approaches for practical implementation of the non-linear controller introduced in the Chapter 3. The first approach is based on the facial representation of the controllable region C_k and the second one is based on some linear programming techniques and the dual simplex algorithm. We compare those two approaches and it is shown that the LP solution is superior, especially for high order systems.

4.1 Computation Using Facial Representation

Here we explain how to find optimal control input, i.e $\lambda_{\min}$ and $\lambda_{\max}$ and minimum time using advantages of facial representation of the k-step controllable region. We also discuss computational complexity of the proposed algorithm.

4.1.1 Computation Of Optimal Control Input

Let's suppose that $x_r \in C_k$ and C_{k-1} is given by its facial representation $C_{k-1} = P(F, z)$. Consider the set $\Lambda = \{\lambda \in R \mid Ax_r + B\lambda \in C_{k-1}\}$. All elements of Λ can be obtained as a solution of system of inequalities

$$F(Ax_r + B\lambda) \leq z \;\;\Leftrightarrow\;\; (FB)\lambda \leq z - FAx_r. \tag{4.1.1}$$

Let $z_i - f_iAx_r$ and f_iB denote i-th entries of the vectors $z - FAx_r$ and FB respectively. Define the sets

$$\begin{aligned} I^+ &= \{i \mid f_iB > 0\} \\ I^- &= \{i \mid f_iB < 0\} \end{aligned} \tag{4.1.2}$$

The system (4.1.1) can be split into two parts:

$$\begin{aligned} \lambda &\leq (z_i - f_iAx_r)(f_iB)^{-1}\;,\; i \in I^+ \\ \lambda &\geq (z_i - f_iAx_r)(f_iB)^{-1}\;,\; i \in I^- \end{aligned} \tag{4.1.3}$$

Obviously the solution to the above system of inequalities is given by

$$\begin{aligned} \lambda &\leq \min\left\{(z_i - f_iAx_r)(f_iB)^{-1} \mid i \in I^+\right\} = \lambda_{\max} \\ \lambda &\geq \max\left\{(z_i - f_iAx_r)(f_iB)^{-1} \mid i \in I^-\right\} = \lambda_{\min} \end{aligned} \tag{4.1.4}$$

Let h be the height of the matrix F. Observe the vector $[(f_1B)^{-1} \dots (f_hB)^{-1}]^T$ can be precomputed. Thus, to find $\lambda_{\min}$ and $\lambda_{\max}$ the only operations required are the basic arithmetic operations of adding, subtracting and multiplying as well as reading from memory. The following simple algorithm is suggested for computing $\lambda_{\min}$ and $\lambda_{\max}$.

```
Precomputed: $F, z, S = [(1/f_1B)\ (1/f_2B)\ \dots (1/f_hB)]^T$
Input: State vector $x_r$
Output: Optimal Control Input $u_r$
1. Set $T = z - F(Ax_r)$.
2. Set $\lambda_{\min} = -\infty$; $\lambda_{\max} = +\infty$.
```

```
3. For $i = 1$ to (height of $S$)
.    3.1. If $(s_i > 0)$ and $(t_i s_i < \lambda_{\max})$ then $\lambda_{\max} = t_i s_i$.
.    3.2. If $(s_i < 0)$ and $(t_i s_i > \lambda_{\min})$ then $\lambda_{\min} = t_i s_i$.
EndFor
4. Return $u_r = sat[(\lambda_{\min} + \lambda_{\max})/2]$.
```

4.1.2 Computation of the Minimum Time

Let us assume that C_K is already chosen as a good approximation of the controllable region C. For a given state vector $x_r \in C_K$ we want to compute

$$t_{\min}(x_r) = \min\{k \in \{1 \dots K\} \mid x_r \in C_k\}.$$

Let $P(F_k, Z_k)$ be the facial representation of the controllable region C_k, $k = 1 \dots K$. The computation of $t_{\min}$ using the facial representation is equivalent to finding the smallest k such that the system of inequalities $F_k x_r \leq Z_k$ is satisfied. At first it seems that we need to precompute and store all pairs (F_k, Z_k), $k = 1 \dots K$, in order to solve this problem. However, as it will be shown later, all the matrices F_k are embedded in the matrix F_K and precomputation and storage of F_K will be sufficient. Now we explain why this is so.

According to Theorem 2.19 (Main Theorem For Zonotopes) each row in the matrix F_k is obtained as a normalized vector orthogonal to $(n-1)$ vector-columns of the matrix $W_k = [w_1 \dots w_k]$. Each combination of $(n-1)$ vector-columns of W_k is also a combination of vector-columns of the matrix W_{k+1} and consequently each row in F_k is also row in F_{k+1}. That is, all matrices F_i, $i < k$, are contained in F_k. In the algorithm for finding facial representation (Section 2.4) we defined the ordered set

$C(n-1,k)$ as the set of all combinations of $(n-1)$ elements of the set $\{1,\dots,k\}$. We can assume that $C(n-1,k)$ is in the 'normal order', e.g.

$$C(2,4) = \{(1,2),(1,3),(2,3),(1,4),(2,4),(3,4)\}.$$

Observe that with such an ordering of $C(n-1,k)$, combinations of the elements in $\{1,\dots,(k-1)\}$ appear first and then combinations with element k follow. As a consequence, the rows of the matrix F_k will be ordered so that those rows which are also the rows of matrix F_{k-1} appear first. This means that all $F_k's$ have the following structure:

$$\begin{aligned} F_K &= [F^T_{(K-1)} \dots]^T, \\ F_{(K-1)} &= [F^T_{(K-2)} \dots]^T, \\ &\vdots \end{aligned}$$

With such a structure we need only to store the matrix F_K and keep information about where each matrix F_k ends in F_K. For that purpose a sequence of indices LastIndex() is assigned to F_K. Here LastIndex(k) is the index of the last row in the matrix F_k. For example, if equality in upper bound theorem holds then LastIndex(k)= $2\binom{k}{n-1}$. In our further discussion we assume that F_K and LastIndex() are precomputed.

It remains to see how to compute and organize vectors Z_k, $k = 1 \dots K$. Let z_{ki} denote i-th entry of the vector Z_k. Each z_{ki} can be computed as

$$z_{ki} = \sum_{j=1}^{j=k} |f_i w_j|$$

One approach is to precompute and store all $Z_k - s$ but it is too costly in the sense of memory requirements. Observe that our state vector mainly follows the trajectory

$C_K \to C_{K-1} \to \ldots \to C_n$ with possible variations in path due to disturbances. It should be noticed also that if $Z = Z_k$ is already computed then each entry of Z_{k-1} can be easily obtained as

$$z_{(k-1)i} = z_{ki} - |f_i w_k| \, , \; i = 1 \ldots LastIndex(k-1)$$

or we can show this equation as update of the vector Z

$$z_i := z_i - |f_i w_k| \, , \; i = 1 \ldots LastIndex(k-1)$$

and now the first LastIndex(k-1) entries of vector Z contains Z_{k-1}. Entries of vector Z from LastIndex(k-1)+1 to LastIndex(k) are not altered by this procedure and they correspond to Z_k. Thus if state vector is pushed back from $C_{k-1} \to C_k$ due to disturbance then Z_k can be obtained from Z by reversing previous update of Z. From the previous discussion it seems most convenient to use a temporary vector (variable) Z, assign it to Z_K initially, and then update Z as the control algorithm develops.

Next we give an algorithm in pseudo-code for finding $t_{\min}$ and updating the vector Z which keeps care of all details and possible variations in state trajectory due to disturbances. It is designed so that memory requirements and expected running time are as low as possible. The algorithm is easy to follow and self-explanatory so further comments are omitted.

Assumption: $x_{r-1} \in C_k$

Precomputed: $C_K = P(F, Z)$, *LastIndex*()

Input: Measured state $x = x_r$

Output: $t_{\min}(x_r)$, *updated vector* Z

```
1. $t_{\min} = k - 1$
2. For i=1 to LastIndex(k-1) do
.    2.1 $\Delta z_i = -|f_i w_k|$
.    2.2 If $(f_i x) > (z_i + \Delta z_i)$ then
.        $t_{\min} = k$, Last_i $= i$
.        Break for-loop
.    End If
End For
3. If $(t_{\min} = k)$ then
.    3.1 For i=Last_i to LastIndex(k) do
.        If $(f_i x > z_i)$ then
.            $t_{\min} = k + 1$, Last_i $= i$
.            Break for-loop
.        End If
.    End For
End If
```

```
4. If $(t_{\min} = k + 1)$ then
.    4.1 $i = \text{Last\_i} - 1$
.    4.2 Repeat
.        4.2.1 $i = i + 1,\ l = k,\ dz = 0$
.        4.2.2 While $f_i x > (z_i + dz)$ do
.            $l = l + 1,\ dz = dz + |f_i w_l|$
.            If $(l > K)$ then break all, state is not controllable
.        End While
.        4.2.3 If $l > t_{\min}$ then $t_{\min} = l$
.    Until $i = \text{LastIndex}(t_{\min})$
.    4.3 For i=1 to LastIndex($t_{\min}$) do
.        $\Delta z_i = \sum_{l=k+1}^{t_{\min}} |f_i w_l|$
.    End For
End If
5. If $(t_{\min} \neq k)$ then
.    For i=1 to LastIndex($t_{\min}$) do
.        $z_i = z_i + \Delta z_i$
.    End For
End If
6. Return($t_{\min}$)
```

4.1.3 Computational Complexity

In the algorithm for finding the optimal control input (see Section 4.1.1) memory requirements are determined by the variables(matrices) F_K, Z_K and S. The dimensions of those matrices are

$$n \times \binom{K}{n-1}, \;\; 1 \times \binom{K}{n-1}, \;\text{ and } 1 \times \binom{K}{n-1}, \tag{4.1.5}$$

respectively.

In the computation of minimum time(Section 4.1.2) the main variables are F_K, Z_K, the temporary vector Z and ΔZ. The dimension of Z and ΔZ is $1 \times \binom{K}{n-1}$. If we assume that an entry in a matrix requires one computer word then the total size of the memory required for storing F_K, Z_K, S, Z and ΔZ is

$$(n+4)\binom{K}{n-1} \text{ words.}$$

Next, we want to find the expected running time of the presented algorithm. In finding the optimal control input, the most computationally intensive part is in step 1 where we compute $z - F(Ax_r)$. Computation of this step requires

$$n \times 2\binom{K}{n-1} \text{ multiplications} \tag{4.1.6}$$

The execution time of the for-loop in the step 3 of the same algorithm can be neglected in comparison to step 1.

In the ideal case of computation of $t_{\min}$ only steps 1 and 2 of the algorithm in Section 4.1.2 will be executed. However in the presence of disturbances it is possible that sometimes the state x will be pushed back to its previous region. In that case step 3 will be executed also. We consider this the worse case scenario and assume that expected running time in finding $t_{\min}$ is determined by steps 1,2 and 3 together.

The computationally most intensive part is computation of $f_i w_k$ inside the for-loops in step 2 and step 3. The required number of multiplications for those operations is

$$2 \times n \times LastIndex(k-1)$$

what is less than $4n\binom{K}{n-1}$. Together with (4.1.6) this gives an expected total number of multiplications to be $6n\binom{K}{n-1}$. We can summarize our previous discussion below:

- Memory requirements : $(n+4)\binom{K}{n-1}$ words.
- Expected running time : $6n\binom{K}{n-1}$ multiplications.

What do these numbers means for practical implementation? For example, consider the system where n=3 (a system with three unstable poles) and K=15. The required memory is of size 1.5kbytes and the required number of multiplication for finding an optimal input is 1900. We can realize those requirements on a 5 dollar microcontroller. However, a great number of systems can still fit into this category of systems with 3 or less unstable poles.

For n=5 and K=20 the size of the required memory is about 100kbytes (very small) and the number of multiplications is 150000. With an microprocessor @ 66MHz the required time is t=2.3ms. This is still very fast.

Observe also that each inequality in the facial representation can be checked independently. This means that we can access simultaneously rows of the matrix F stored in the memory. If we organize the required memory, e.g. into ten parallel blocks, then we can reduce execution time by ten times. In short, the presented algorithm is also suitable for parallel processing.

We may conclude that the presented controller is easily implementable for many real world systems.

4.2 Linear Programming Preliminaries

For consistency of this thesis, some basic terminology and concepts from linear programming are introduced in this section so that reader who is not familiar with linear programming can follow the ideas given in section 4.3. We give definitions of primal-dual LP problems and their relationship, a description of the simplex algorithm and the dual simplex algorithm for solving bounded variable linear programs, as it is given in [24]. Our approach here is informal and rather descriptive with proofs and details omitted. The reader who is looking for more details is referred to [24], an excellent guide on linear programming.

4.2.1 Basic Definitions

In linear programming our goal is to optimize a linear objective function $z = cx$ subject to linear equality and inequality constraints on the input argument x. Since there are many ways that constraints on x can be defined it is convenient to use the so-called standard description of a linear program (LP).

Definition 4.1. (LP in standard form)
A linear program given by

$$\begin{aligned} &\text{Minimize} \quad && z(x) = cx \\ &\text{Subject to} \quad && Ax = b,\ x \geq 0 \end{aligned}$$

is said to be in standard form.

Every LP can be put in standard form by augmenting the argument x. For example, if there are inequality constraints of the kind

$$Ax \leq b,\ x \geq 0$$

then we can introduce a so-called slack variable

$$x_s = b - Ax$$

and put inequality constraints into an equivalent form

$$Ax + x_s = b,\ x \geq 0,\ x_s \geq 0.$$

Similar transformation can be used in other cases. Thus, without loss of generality we can assume in the further text that the LP being discussed is in standard form.

Definition 4.2. (Feasible solution)
A vector x that satisfies

$$Ax = b,\ x \geq 0$$

is called a *feasible solution* of the LP.

Let's define $\mathcal{K}$ to be the set of all feasible solutions. Observe that $\mathcal{K}$ is a convex polyhedron. For any convex set $\mathcal{G}$ we can define its extreme points.

Definition 4.3. (Extreme point)
Let $\mathcal{G}$ be a convex set. A point $x \in \mathcal{G}$ is called an *extreme or corner point* of the set $\mathcal{G}$ if and only if for every $x_1 \in \mathcal{G}$, $x_2 \in \mathcal{G}$ and $0 < \lambda < 1$ for which

$$x = \lambda x_1 + (1 - \lambda) x_2$$

we have that $x = x_1 = x_2$.

For example, all extreme points of a polytope $\mathcal{P}$ are its vertices. It can be easily shown that the minimum (maximum) of a linear function on the convex set $\mathcal{G}$ can be obtained only at an extreme point of $\mathcal{G}$. Thus, to find an optimal solution of an LP it is sufficient to consider the extreme points of the convex polyhedron $\mathcal{K}$.

Definition 4.4. (Basic feasible solution)
A feasible solution $x \in \mathcal{K}$ is called a *basic feasible solution (BFS)* if it is an extreme point of $\mathcal{K}$.

All BFSs for an LP given in standard form have simple algebraic characterizations. The following proposition describes it.

Proposition 4.5. *Let $x = [x_1 \dots x_n]^T \in \mathcal{K}$ be a feasible solution of an LP in standard form. Let A_i denote the i-th column of the matrix A and define the set of column vectors $\mathcal{C} = \{A_j \mid x_j > 0,\ j = 1 \dots n\}$. The feasible solution x is a BFS if and only if $\mathcal{C}$ is a set of linearly independent vectors.*

Definition 4.6. (Degenerate and Nondegenerate BFS)
Let x be a BFS and let $\mathcal{C}$ be defined as in Proposition 4.5. If there are exactly rank(A) columns in $\mathcal{C}$ then x is called a *nondegenerate* BFS. Otherwise x is called *degenerate* BFS.

We want to further investigate the relationship between the algebraic description of the LP and extreme points. Suppose that the number of columns in the matrix A is greater than the number of rows, that is $n > m$. Any nonsingular square submatrix of A of size m is called a *basis* of LP. Suppose that B is such a basis. All those columns of A which are also columns in the basis B are called *basic columns* and all the remaining columns of A are known as *nonbasic columns.* Rearranging the columns in A and the entries of x we can write the constraints of our LP in the form

$$Bx_B + Dx_D = b$$

$$x_B \geq 0;\ x_D \geq 0$$

Here, the vector x_B is known as the *basic vector* and its entries are called the *basic variables.* Similarly x_D is the *nonbasic vector* of *nonbasic variables.*

Assume that all nonbasic variables are set to zero and x_B is determined so that the equality $Ax = b$ is satisfied. Since B is nonsingular it yields a solution of

$$x_D = 0;\ x_B = B^{-1}b$$

It should be noted that a solution obtained in this way is not necessarily a feasible solution since the nonnegativity constraint $x_B \geq 0$ may not be satisfied.

Definition 4.7. (Basic Solution and Feasible Basis)
Let B be a basis and let x_B, x_D be the corresponding basic and nonbasic vectors respectively. The solution $x_D = 0$, $x_B = B^{-1}b$ is called a *basic solution* associated to the basis B. Furthermore, if the nonnegativity constraint $x_B \geq 0$ is satisfied then B is called a *primal feasible basis.*

From the above definition it should be clear that any basic solution associated with a primal feasible basis is a BFS of the LP. Similarly if we suppose that x is a nondegenerate BFS then there are exactly m variables in x which are greater than zero. We can group these variables into a basic vector x_B. Then the basis B associated with x_B is obviously a primal feasible basis. The result which follows gives a relationship between the primal feasible bases and BFSs.

Proposition 4.8. *For an LP given in standard form the following holds:*

1. *Every primal feasible basis has only one BFS associated with it.*

2. *Each nondegenerate BFS has a unique primal feasible basis associated with it.*

3. *A degenerate BFS may have many primal feasible bases associated with it.*

An obvious implication of the previous proposition is that the total number of distinct BFSs is less than or equal to total number of distinct bases of A. Knowing that the number of distinct bases in A in not greater than $\binom{n}{m}$ means that the total number of distinct BFSs is finite and less than or equal to $\binom{n}{m}$.

4.2.2 Simplex Method

From the previous subsection we can see that using an algebraic characterization of BFSs, all extreme points of LP can be found by traversing over the set of bases of the matrix A and checking whether some basis B is primal feasible. Thus, we have a way to find all BFSs of our LP. Now, one naive approach to find an optimal solution of our LP is to compute the value of the objective function for each BFS and then find the minimum among these. Of course, because of the possibility that there may be a great number of distinct BFSs this approach is computationally inefficient. The simplex method is an efficient alternative for finding the optimal solution of an LP through an iterative procedure. Instead of looking at all BFSs, the simplex method starts from one initial BFS and moves iteratively from one BFS to another always in a way that the objective function $z(x)$ is decreased, and it finally terminates when the minimum value is reached. Let's explain how these iterative steps are performed. First we start with geometrical description.

Suppose that a set of feasible solutions of an LP is a polytope $\mathcal{P}$. All BFSs of LP are vertices of the polytope $\mathcal{P}$. Henceforth we use the term *adjacent* vertices to refer to a pair of vertices which are connected by an edge of $\mathcal{P}$. Let V be the vertex associated to the current BFS in the simplex procedure. The candidate BFSs for the next step of the simplex procedure are all those vertices which are adjacent to V and in the direction of a decreasing objective function. Let e denote the edge connecting adjacent vertices V and V_1. The unit change of $z(x)$ along the edge e is the gradient $\partial z/\partial e$. Logically, between all those edges starting from vertex V, the simplex algorithm chooses that edge with a minimum gradient and moves along it to the next vertex (BFS). By this strategy we may 'hope' to reach an optimal solution in the smallest number of iteration steps. As we can see the idea behind the simplex method is quite simple. Let us see how to translate this approach into the algebraic domain.

Let $x = [x_B \, x_D]^T$ be the current BFS with a corresponding primal feasible basis B. Without loss of generality we can assume that the matrix A and the entries of x are rearranged so that the basic and nonbasic vectors are given, respectively, as $x_B = [x_1 \ldots x_m]$ and $x_D = [x_{m+1} \ldots x_n]$. Let $\hat{x}$ be a BFS adjacent to x. Than it can be easily proven that the basic vector $\hat{x}_B$ can be obtained from x_B by exchanging one basic variable from x_B, let's say x_l, $1 \leq l \leq m$, for one nonbasic variable $x_s, (m+1) \leq s \leq n$. That is, $\hat{x}_B$ is of the form $[x_1 \ldots x_{(l-1)} \, x_s \, x_{(l+1)} \ldots x_m]$. Thus, the main step in an iteration of the simplex procedure is to chose which nonbasic variable x_s is the best in the sense of decreasing objective function. Such nonbasic variables are called *entering variables*. Let's explain how to find the entering variable.

A basis B is a nonsingular matrix and the equality $Ax = b$ can be written as

$$x_B + B^{-1} D x_D = B^{-1} b.$$

We use the notation

$$B^{-1} D = \hat{A} = [\, \hat{a}_{i,j} \,]_{i=1 \ldots m,\, j=m+1 \ldots n}$$

$$B^{-1} b = \hat{b}$$

All basic variables can be expressed as a function of nonbasic variables

$$x_i = \hat{b}_i - \hat{a}_{i,(m+1)}x_{m+1} - \ldots - \hat{a}_{i,n}x_n, \;\; i = 1 \ldots m$$

Using these equations the basic variables can be eliminated from our LP problem, and the problem can be expressed purely in terms of the nonbasic variables. For the objective function $z(x)$ we have

$$z(x) = cx = c_B x_B + c_D x_D = c_B \hat{b} + (c_D - c_B \hat{A})x_D$$

In expanded form this leads to

$$z(x) = z_0 + \hat{c}_{m+1}x_{m+1} + \ldots + \hat{c}_n x_n$$

The coefficients $\hat{c}_j$ are known as relative cost coefficients. The previous transformations changes the LP problem into the following one:

$$\begin{aligned} &\text{Minimize} \quad z_0 + \hat{c}_{m+1}x_{m+1} + \ldots + \hat{c}_n x_n \\ &\text{Subject to} \quad \hat{b}_i - \hat{a}_{i,(m+1)}x_{m+1} - \ldots - \hat{a}_{i,n}x_n \geq 0, \;\; i = 1 \ldots m \\ &\qquad\qquad\quad x_j \geq 0, \;\; i = (m+1) \ldots n \end{aligned} \tag{4.2.1}$$

In the present BFS x all nonbasic variables x_D are zero. Let's suppose that the variable x_j is changed from zero to some value $\Delta x_j > 0$. It can be easily shown that the corresponding change in the objective function is $\Delta z(x) = \hat{c}_j \Delta x_j$. This means that the relative cost coefficient reflects a change of objective function per unit change of associated nonbasic variable. Since our goal is to decrease $z(x)$ it seems best to choose for the entering variable that x_s with relative cost coefficient

$$\hat{c}_s = \min\{\hat{c}_j \mid j = m + 1 \ldots n\}$$

Observe that if all relative cost coefficient are positive then there is no way that objective function can be further decreased and the present BFS is the optimal solution for which the minimum of the objective function is achieved. The condition that all relative cost coefficients are positive is labeled the *primal optimality criterion.*

Assume that $\hat{c}_s < 0$. Increasing x_s leads to the decreasing of $z(x)$. However, there is a limitation for x_s since all constraints in (4.2.1) must be satisfied. It is easy to prove that the maximum value which x_s can take is

$$\theta = \min \left\{ \hat{b}_i / \hat{a}_{i,s} \mid i \text{ such that } \hat{a}_{i,s} > 0 \right\}$$

Let i=r be an index which ties for the minimum in the previous equation. When x_s is given the value θ, the basic variable x_r becomes equal to zero. Now replace the present basic variable x_r by entering the variable x_s and the new BFS is obtained. By this, one iteration of simplex procedure is completed. Repeating the previous procedure we can reach the optimal solution.

4.2.3 The Dual Simplex Method

Before we proceed with a description of the dual simplex method we need some terms from duality theory for linear programing.

Definition 4.9. (Dual LP)

For a LP given in standard form its *dual* LP is defined as

$$\begin{aligned} &\text{Maximize} \quad v(y) = yb \\ &\text{Subject to} \quad yA \le c\,;\; y \text{ unrestricted} \end{aligned} \tag{4.2.2}$$

Observe some facts about primal-dual relationship. There is one dual variable associated with each equality constraint in the primal problem. There is one dual constraint corresponding to each primal variable. If the primal is a minimization problem, the dual is a maximization problem and vice-versa. In the same way we did for the primal LP we can define the *dual feasible solution* as a feasible solution of the dual problem. Using this analogy we can define other terms like a *dual BFS, dual feasible basis, dual optimality criterion, etc.*

Proposition 4.10. *In the primal dual pair of LPs the following holds.*

1. *The primal objective value of any primal feasible solution is an upper bound to the maximum value of the dual objective in the dual problem.*

2. *The dual objective value of any dual feasible solution is a lower bound to the minimum value of the primal objective in the primal problem.*

3. *If either the primal or the dual problem has an optimal feasible solution, then the other does also, and the two optimal objective values are equal.*

4. *For a given basis B the optimality criterion in the primal simplex method is the dual feasibility criterion for the same basis and vice versa.*

5. *If a basis B is both a primal and dual feasible basis then B is the basis associated with the optimal solution.*

From the above results we can make some conclusions about the primal simplex method. The primal simplex method always deals with primal feasible bases and terminates only when the primal optimality criterion is satisfied. Thus, all but the final optimal basis used in the simplex algorithm are dual infeasible. That is, the simplex algorithm starts with a primal feasible but dual infeasible basis and it tries to attain dual feasibility, keeping primal feasibility throughout the algorithm. However, it is possible to develop an algorithm for solving the primal LP that starts with a dual feasible but primal infeasible basis and tries to attain primal feasibility, keeping dual feasibility throughout the algorithm. An algorithm of this kind is known as a dual simplex algorithm. We present it here in short.

When we say bounded-variable linear programs we consider the LP of the following form:

$$\begin{aligned} &\text{Minimize} \quad z(x) = cx \\ &\text{Subject to} \quad Ax = b;\ 0 \le x_j \le U_j,\ \forall j = 1 \ldots n \end{aligned} \tag{4.2.3}$$

where A is a matrix of dimension $m \times (n + m)$ and rank m. For solving such a bounded variable problem the dual simplex is particularly efficient. Here is why. By introducing slack variables we can transform the upper bound constraints into equalities and put the bounded variable LP into standard form. However, in that case there will be $n + m$ equalities which means that if we are going to apply the primal simplex method we need to deal with bases of order $n + m$. This can be very inconvenient if n is much larger than m and that is usually the case in practical applications. The dual simplex algorithm works directly with bases of order m and much smaller set of BFSs than the primal simplex method.

Let B be a basis of the matrix A and let x_B and x_D be the corresponding vectors of basic and nonbasic variables. Further denote by c_B the cost coefficients corresponding to the basic vector x_B. For a given basis $y = c_B B^{-1}$ defines the corresponding dual variables and $\hat{c} = c - yA$ are updated cost coefficients. Let N be the set of subscripts of current nonbasic variables. Define a solution for $Ax = b$ by

$$\begin{aligned} x_j &= U_j \quad \text{if } \hat{c}_j < 0 \\ x_j &= 0 \quad \text{if } j \in N \text{ and } \hat{c}_j \geq 0 \\ x_B &= B^{-1}(b - Dx_D) \end{aligned} \tag{4.2.4}$$

Observe that by this solution all nonbasic variables are uniquely determined for a given basis and it is done in such manner that the current solution satisfies the optimality criterion (dual feasibility criterion). However, it may be possible that upper or lower bounds for variables in x_B may not be satisfied. That is, the current solution is not primal feasible. Once primal feasibility is satisfied that solution is an optimum feasible solution.

Let's explain in short an iterative procedure of the dual simplex method. In an iteration step of the dual simplex method, a basic variable x_{b1} which does not satisfy the upper or lower bound is the candidate for exchange with some nonbasic variable. The new entering variable x_{d1} is chosen such that the following conditions are satisfied:

1. After entering of x_{d1} into the vector of basic variables and obtaining the new solution, both variables x_{b1} and x_{d1} satisfy the boundary constraints.
2. The new solution obtained after an iterative step results in the increasing of the dual objective function.

By this iterative procedure it is guaranteed that the dual simplex algorithm always terminates and that there is no cycling over the set of solutions (extreme points) given by (4.2.4).

Remark 4.1. Observe some properties of the dual simplex algorithm. For each basis B there is unique solution given by (4.2.4). Since the number of different basis is not greater than $\binom{n+m}{m}$ this means that number of extreme points in the dual simplex algorithm is not greater than $\binom{n+m}{m}$. In a practical application of the dual simplex algorithm as with the primal simplex method, we can expect that the number of iterations required to solve a bounded linear program will be significantly smaller than the total number of extreme points.

4.3 Computation Using Linear Programming

In this section we show how the dual simplex algorithm can be used to find $\lambda_{\max}$ and $\lambda_{\min}$. It is also shown that this approach based on linear programming is significantly faster than any approach based on facial or vertex representation of K-step controllable regions. More precisely, we prove that solutions based on facial representations of some controllable regions is equivalent to solving dual simplex algorithm by looking at all its extreme points, which is a naive approach.

4.3.1 Computation Of Optimal Control Input

Here, we explain how to find

$$\lambda_{\min} = \min\{\lambda \in R \mid Ax_k + B\lambda \in C_K,\ x_k \in C_{K+1}\} \tag{4.3.1}$$

using the dual simplex method. The same idea can be applied to find $\lambda_{\max}$.

Denote $W = -[A^{-1}B\ A^{-2}B \ldots A^{-K}B]$. By definition of the K-step controllable region there exists a vector $U = [u_1 \ldots u_K]^T$ such that

$$Ax_k + B\lambda = WU,\ \ u_i \in [-1, 1],\ i = 1, \ldots, K \tag{4.3.2}$$

or equivalently

$$[W \; - B][U \; \lambda]^T = Ax_k, \;\; u_i \in [-1, 1], \; i = 1, \dots, K \tag{4.3.3}$$

Our minimization problem given by 4.3.1 can be restated as an LP problem:

> Minimize the objective function $z(U, \lambda) = \lambda$ subject to constraints (4.3.3) and λ unconstrained.

Observe that all variables in the given LP problem are bounded except λ. Next we want to express λ in term of U and eliminate it from the system (4.3.3) and objective function z. Of course we can do it since there is no inequality constraints on λ. Denote

$$W = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}, \; A = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}, \; B = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Here w_i and a_i are vector rows and b_i is a scalar. Assume that $b_i \neq 0$. From the i-th equation in the system (4.3.3) it follows

$$\begin{gathered} a_i x_k = w_i U - b_i \lambda \\ \lambda = (w_i U - a_i x_k)/b_i \end{gathered} \tag{4.3.4}$$

Denote by W_i, A_i and B_i the matrices W, A and B without i-th row. Substituting the last expression for λ into (4.3.3) and the objective function z we have

$$\begin{gathered} (W_i - B_i w_i/b_i)U = A_i x_k + (a_i x_k)/b_i \\ z(U, \lambda) = z(U) = (w_i U)/b_i - (a_i x_k)/b_i \end{gathered} \tag{4.3.5}$$

Define $T = W_i - B_i w_i/b_i$ and $C = A_i x_k + (a_i x_k)/b_i$. Obviously C is a constant vector. Observe also that the term $(a_i x_k)/b_i$ in the objective function is constant and not important in finding the minimum of $z(U)$. Taking the previous results into account our LP problem can be restated as a bounded variable LP problem:

Minimize the objective function $z(U) = (w_i U)/b_i$ subject to the constraints

$$\begin{aligned} TU &= C \\ u_i \in [-1,1],\ i &= 1,\dots,K \end{aligned} \tag{4.3.6}$$

To solve the above problem we can use the dual simplex method. Once an optimal value U_o is found $\lambda_{\min}$ can be obtained from (4.3.4).

4.3.2 Computational Complexity

Let's explain why the dual simplex approach is faster than the solution based on a facial representation of controllable region.

From the expression for matrix T we know that T is an $(n-1) \times K$ matrix. Consequently, the number of possible working bases B in the bounded LP (4.3.6) is $\binom{K}{n-1}$ and by remark 4.1 it is also the total number of extreme points in the dual simplex method used to solve (4.3.6). Recall from the Main Theorem in Chapter 2 that the expected number of faces required for facial representation of the K-step controllable region is $2\binom{K}{n-1}$ which is twice the number of extreme points in the corresponding dual simplex algorithm.

Furthermore, it is easy to check that each solution (extreme point) in our bounded LP obtained by equation (4.2.4) is an intersection of a line with a facet of the controllable region. When we found $\lambda_{\min}$ in Section 4.1.1 using a facial representation of C_K we looked at all possible intersections of the line with the facets of C_K what is computationally equivalent to finding the value of the objective function $z(U)$ at all extreme point in the bounded LP. But when we introduced the simplex method we said that the most naive approach for solving LP is to look at all extreme points. Now, it should be clear why the LP approach should be superior in comparison to facial representation approach, especially for higher order systems.

Chapter 5

Generalization

In this chapter we show how the nonlinear control law given in Chapter 3 can be generalized to multi-input systems and systems with both stable and unstable poles.

5.1 Multinput Systems

Consider the system with m control inputs. Assume that a state vector $x_k \in C_{k+1}$. Similar to the single input case we want to solve the following problem:

> Given the $x_k \in C_K$ find a control input vector $u_k = [u_{k1} \dots u_{km}]^T$,
> $u_k \in [-1, 1]^m$ such that $x_{k+1} = Ax_k + Bsat(u_k) \in C_{K-1}$.

By the definition of controllable regions (see Chapter 2) we know that this problem always has a solution. It remains to see how to find such control input vector u_k. In the further text we will show that u_k can be simply obtained by repeating m times the previously described procedure for single input systems.

Denote by b_i the i-th column of matrix B. With this notation $B = [b_1 \dots b_m]$. Define the sets

$$\begin{aligned} C_{K-1,1} &= C_{K-1} + [-b_1, b_1] \\ C_{K-1,2} &= C_{K-1,1} + [-b_2, b_2] \\ &\vdots \\ C_{K-1,m} &= C_{K-1,m-1} + [-b_m, b_m]. \end{aligned} \tag{5.1.1}$$

Henceforth, we call the sets $C_{K-1,1}, \dots C_{K-1,m}$ *connecting regions.* Observe that the connecting regions are zonotopes and all results from Chapter 2 can be applied to connecting regions as well.

If u_k is an optimal control input then the following holds:

$$\begin{aligned} Ax_k = x_{k+1} - b_1 u_{k1} - b_2 u_{k2} - \dots - b_m u_{km} \\ x_{k+1} \in C_{K-1},\ u_{k1}, \dots u_{km} \in [-1, 1] \end{aligned} \tag{5.1.2}$$

The last equation gives

$$Ax_k \in C_{K-1} + [-b_1, b_1] + \dots + [-b_m, b_m]$$

or equivalently

$$Ax_k \in C_{K-1,m}.$$

According to the definition of connecting regions we also know that if a point(vector) $x_{k,i} \in C_{K-1,i}$ then there exists scalar $u_{ki} \in [-1, 1]$ such that $x_{k,i} + u_{ki} b_i = x_{k,i-1} \in C_{K-1,i-1}$. Moreover, we can use the same nonlinear mapping $g(x) = sat[(\lambda_{\min} + \lambda_{\max})/2]$ from Chapter 3 to find such scalar u_{ki}. Here $\lambda_{\min}$ and $\lambda_{\max}$ are determined with respect to the intersection of the set $C_{K-1,i-1}$ and the line $x_{k,i} + \lambda b_i$, $\lambda \in R$.

Now it should be obvious how we can use those facts to find an optimal control for multi-input systems. The following control strategy is proposed.

We start from the point $Ax_k \in C_{K-1,m}$. Then the $m - th$ entry u_{km} of the control input u_k is determined using the nonlinear mapping from Chapter 3, such that $Ax_k - b_m u_{km} \in C_{K-1,m-1}$. Denote $x_{k,m-1} = Ax_k + b_m u_{km}$. In the next step we determine $u_{k(m-1)}$ such that $x_{k,m-1} + b_{m-1} u_{k(m-1)} \in C_{K-1,m-2}$. Denote $x_{k,m-2} = x_{k,m-1} + b_{m-1} u_{k(m-1)}$ and repeat the previous procedure to find $u_{k(m-2)}$. After m-steps we end up with the point $x_{k,0} = x_{k+1} \in C_{K-1}$ and obviously the control input $u_k = [u_{k1} \dots u_{km}]^T$ obtained by previously described procedure is an optimal control input.

By this we have shown that our concept developed for a single input system can be easily extended to multi input case.

5.2 Systems with both Stable and Unstable Poles

Let's see how we can generalize our control law for systems with both stable and unstable poles. In Chapter 3 we gave a control law which stabilizes all state vectors inside any controllable region $C_k, k \in Z$. We assumed also that the system we are considering has unstable poles only and we needed that assumption to show that there exists relatively small $k \in Z$ such that C_k is a good approximation of the controllable region C. In the general case for a system with both stable and unstable poles the same control law can be used to stabilize any state vector inside some k-step controllable region C_k. The only difference is that now we cannot assume that C_k will be a good approximation of the controllable region C. Yet, for practical applications the stabilization of all state vectors inside some k-step controllable region C_k is usually satisfactory and we need not consider the whole controllable region C. Let's explain why it is so.

In the general case we can apply a similarity transformation to the system we are considering and write it into alternative form

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} sat[u(k)] \tag{5.2.1}$$

with A_1 having all unstable eigenvalues and A_2 having all stable and marginally stable eigenvalues. Let n_1, n_2 be the order of matrices A_1 and A_2 respectively. Define the sets

$$C_j(K) = \left\{ \sum_{i=1}^{K} A_j^{-i} B_j u(i) \mid u(i) \in [-1,1]^m \right\},\ j = 1,2$$

and

$$C_j = \lim_{K\to\infty} C_j(K),\ j = 1,2$$

It is straightforward to check that the K-step controllable region C_K of the system (5.2.1) can be written as $C_K = C_1(K) \oplus C_2(K)$ and $C = C_1 \oplus C_2$. Furthermore, by properties of the matrices A_1 and A_2 it is obvious that C_1 is a bounded region and $C_1(K)$ exponentially converges to C_1, and at the same time $C_2 = R^{n_2}$ and $C_2(K)$ is exponentially increasing as $K \to \infty$.

In real-world systems all state variables are usually some physical parameters like position, velocity etc. and by their physical nature they are bounded. Even after applying similarity transform it is reasonable to suppose that state vector $x_2(k)$ is always inside some bounded region. Since $C_2(K)$ is exponentially increasing as $K \to \infty$ we can assume that there exists $K_2 \in Z$ such that $x_2(k)$, $k \in N$ is always inside the set $C_2(K_2)$.

Let $K_1 \in Z$ be such that $C_1(K_1)$ is a good approximation of C_1. Since A_1 has only unstable eigenvalues such K_1 always exists. If we define $K = \max\{K_1, K_2\}$ then all state vectors of practical importance are inside the K-step controllable region C_K. Although C_K can not be considered an approximation of the controllable region C, for practical purposes it is sufficient to find stabilizing controller for all states inside C_K. To achieve this goal we can use the same time-optimal control law given in Chapter 3.

The previous strategy is obviously a time-optimal control law for the general case. However it may be hard to implement it if the order of the system is too high. It is interesting to see whether we can simplify implementation in the general case. In what follows we give an alternative approach which is easier for practical implementation but which is not time-optimal.

Assume that A_2 has strictly stable eigenvalues. Let $C_1(K_1)$ be an approximation of C_1. Consider the subsystem corresponding to the state vector $x_1(k)$

$$x_1(k+1) = A_1 x_1(k) + B_1 sat[u(k)] \tag{5.2.2}$$

Applying the nonlinear state feedback controller from Chapter 3 we can find a control signal $u(k)$ which is a time optimal control signal for the subsystem (5.2.2) on the controllable region $C_1(K_1)$. By properties of the time-optimal control we know that $x_1(k)$ and $u(k)$ converges to zero in the finite number of steps. Since A_2 is a stability matrix, the finite duration of signals $x_1(k)$ and $u(k)$ implies that $x_2(k)$ tends exponentially to zero. That is, the time-optimal control law determined with respect to the subsystem (5.2.2) is a stabilizing control law for the system (5.2.1).

By this strategy the complexity of our nonlinear controller is determined by the number of unstable eigenvalues only but it is not a time-optimal control anymore.

Chapter 6

Examples and Simulation Results

6.1 Example 1: The Second Order System

In this section we consider the second order system

$$x(k+1) = \begin{bmatrix} 1.5 & 0 \\ 0 & -2 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 3 \end{bmatrix} sat[u(k)]. \tag{6.1.1}$$

In this case the eigenvalues of the matrix A are outside of the unit disk and the system is unstable.

The first step in the design of nonlinear controller is to find such K so that C_K is a good approximation of C. Examining the matrix W_K of the system 6.1.1 we see that 12-th column in the matrix W_K is

$$w_{12} = [-0.0077 \; -0.0007]^T,$$

what is quite small in comparison to the first few columns of W_K. This means that $C(12)$ will be a good approximation of C. Figures 6.1 and 6.2 depict the controllable regions of our system.

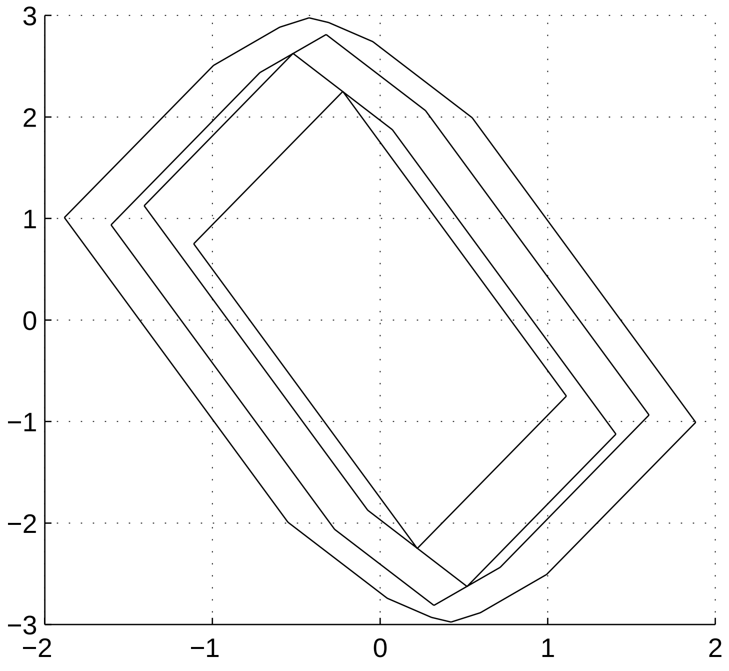

Figure 6.1: Controllable regions $C(2)$, $C(3)$, $C(4)$ and $C(7)$.

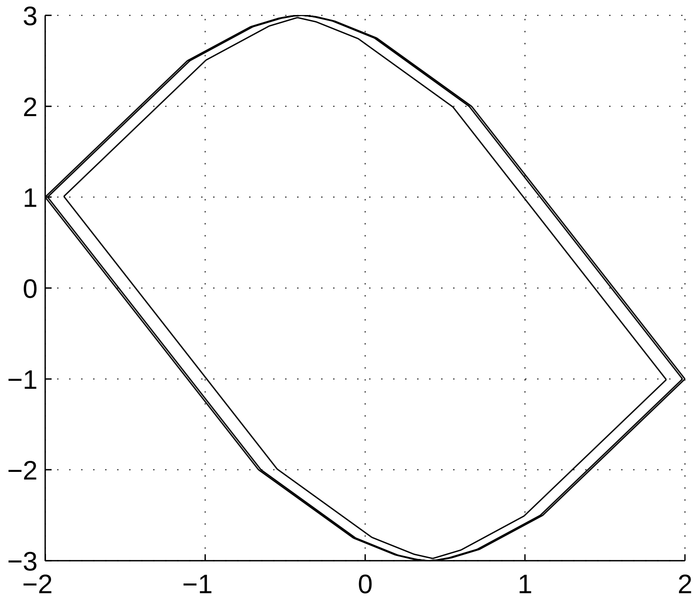

Figure 6.2: Controllable regions $C(7)$, $C(12)$ and $C(20)$.

Observe from Figure 6.2 that the distinction between regions $C(12)$ and C(20) is hardly visible. This confirms our choice of $K = 12$.

The next step is to compute the stabilizing nonlinear controller. We use the facial representation approach in this example. According to the Main Theorem the expected number of inequalities for the description of $C(12)$ is 24. We used the algorithm described in Section 2.4 to find matrices the F and z corresponding to $C(12)$. With F precomputed, the algorithm given in Section 4.1 is used for computation of an optimal control input at each step k of time.

Applying our time-optimal control for an initial condition of

$$x_0 = \begin{bmatrix} 1.9300 \\ -1.0025 \end{bmatrix}$$

which is close to a vertex of the region $C(10)$ we expect that the system will be stabilized in 10 steps. Figures 6.3 and 6.4 show a closed loop response.

According to expectations both states are steered to the zero in only 10 steps of discrete time. Observe from Figure 6.4 that control signal u is quite smooth and goes to zero as well. In Figure 6.5 we can see the state trajectory as it develops from a vertex of $C(10)$ to the origin. It should be noticed that with each new step k, the state $x(k)$ gets deeper inside its corresponding region and in that way becomes more resistant to input disturbances.

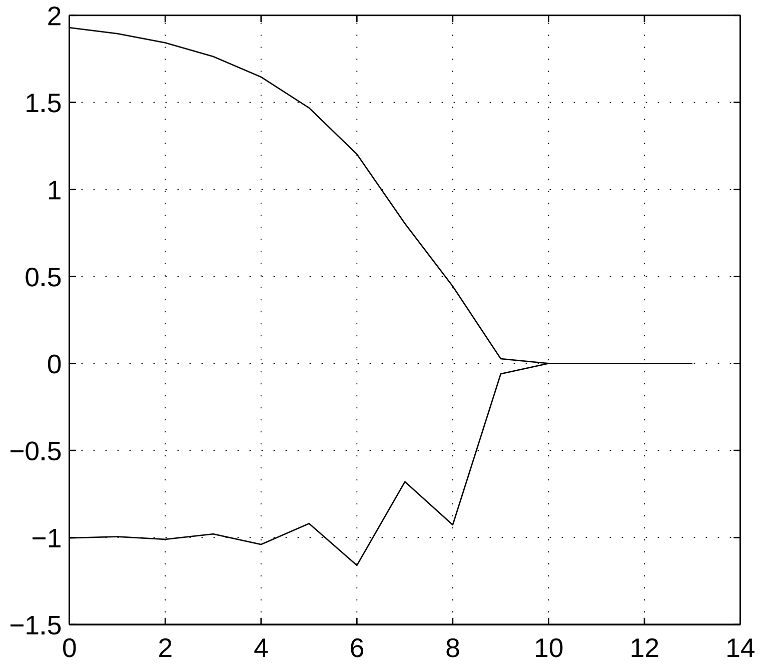

Figure 6.3: The regulated state x.

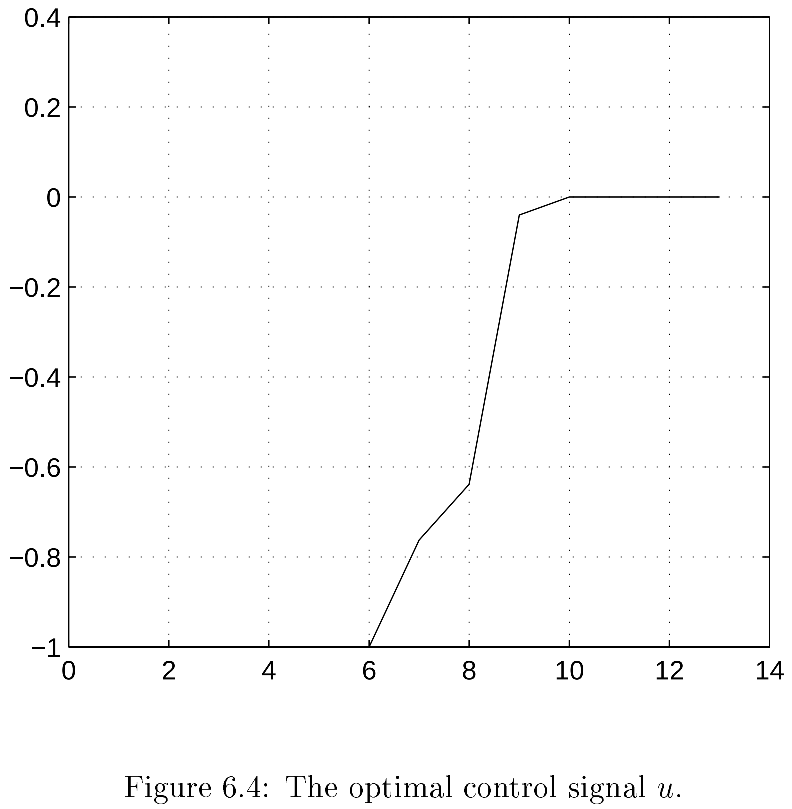

Figure 6.4: The optimal control signal u.

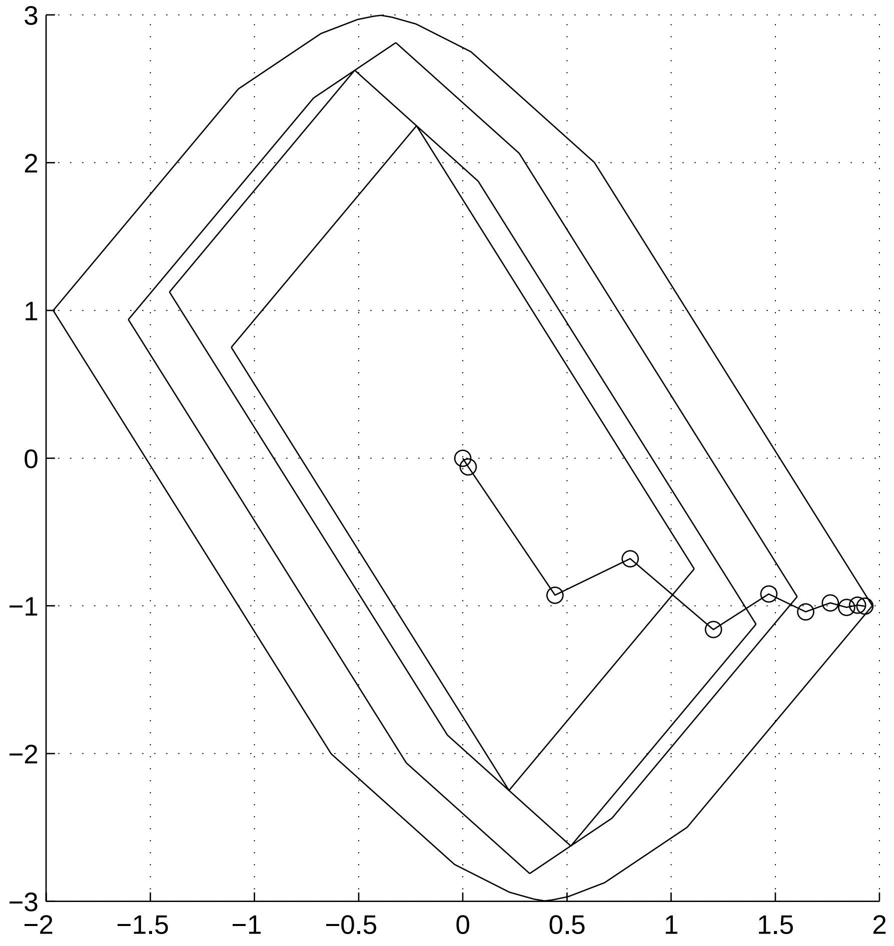

Figure 6.5: The state trajectory and controllable regions $C(2)$, $C(3)$, $C(4)$ and $C(10)$.

6.2 Example 2: The 4-th Order System

Consider the 4-th order system

$$x(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -2.5 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} sat[u(k)]. \qquad (6.2.1)$$

with one marginally stable eigenvalue and three unstable eigenvalues. To design a stabilizing nonlinear controller for this system we use the generalization described in Section 5.2. The system (6.2.1) can be split into two subsystems: one semi-stable associated with the state variable x_1 and the remaining unstable part associated with the state variables x_2, x_3 and x_4. If we define $C_1(K)$ and $C_2(K)$ to be K-step controllable regions corresponding of those subsystems then

$$C(K) = C_1(K) \oplus C_2(K) \text{ and } C = R \oplus C_2.$$

Theoretically, an initial state vector $x(0)$ can be controllable even if the state variable x_1 is arbitrarily large. In the real world we always have some physical limitations. Assume that in our case the state variable x_1 is limited to the interval $[-20, 20]$. With this assumption, the next step in the design procedure is to find K so that all states of interest are inside $C(K)$.

Obviously $C_1(k) = [-k, k]$ and consequently

$$C(k) = [-k, k] \oplus C_2(k).$$

$C_2(k)$ corresponds to the unstable subsystem and we can always find some K_2 so that $C_2(K_2)$ is a good approximation of C_2. In this case $K_2 = 8$ satisfies. As k is increasing over 8, $C(k)$ is extending mainly along the coordinate of the state variable x_1. Consequently we can approximate $C(k)$ by

$$C(8) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} [-(k-8), (k-8)], \;\; k \geq 8.$$

Here "+" denotes Minkowski sum. Knowing that $[1\,0\,0\,0]^T[-8, 8] \subset C(8)$ we can see that if $k \geq 20$ then $[1\,0\,0\,0]^T[-20, 20] \subset C(k)$. Thus, we can conclude that all initial states of interest are inside the controllable region $C(20)$.

Once when we determined K to be 20, the next step is to compute a nonlinear controller which stabilizes all initial states inside $C(20)$. The order of the system is 4 and K=20 is relatively large. In this case it seems more convenient to use the LP approach and a dual simplex method for the computation of the nonlinear control. To examine performance of the optimal controller we consider an initial condition of

$$x_0 = \begin{bmatrix} 10.0500 \\ -0.9414 \\ -0.2890 \\ -0.4873 \end{bmatrix}.$$

This initial condition is close to the boundary of $C(16)$.

Figure 6.6 and 6.7 depict the closed loop response of the system. All four state variables are steered to zero in 16 steps of discrete time which confirms the time-optimality of the proposed nonlinear controller. On Figure 6.8 we can see activity of the optimal control signal.

Simulation results for this example are obtained in the real-time with the sample time of the system, T=20ms. This confirms that computation based on the LP approach is quite fast and suitable for practical applications.

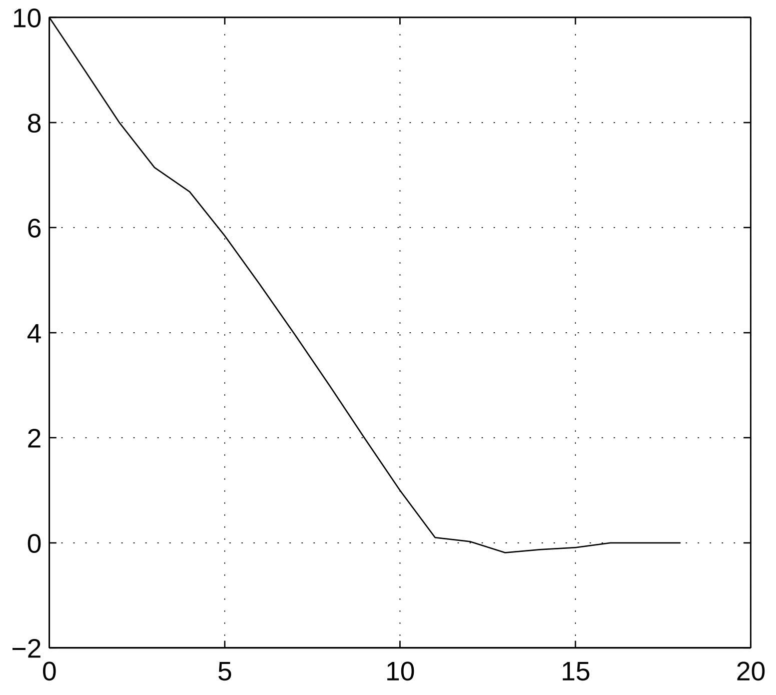

Figure 6.6: The regulated state variable x_1.

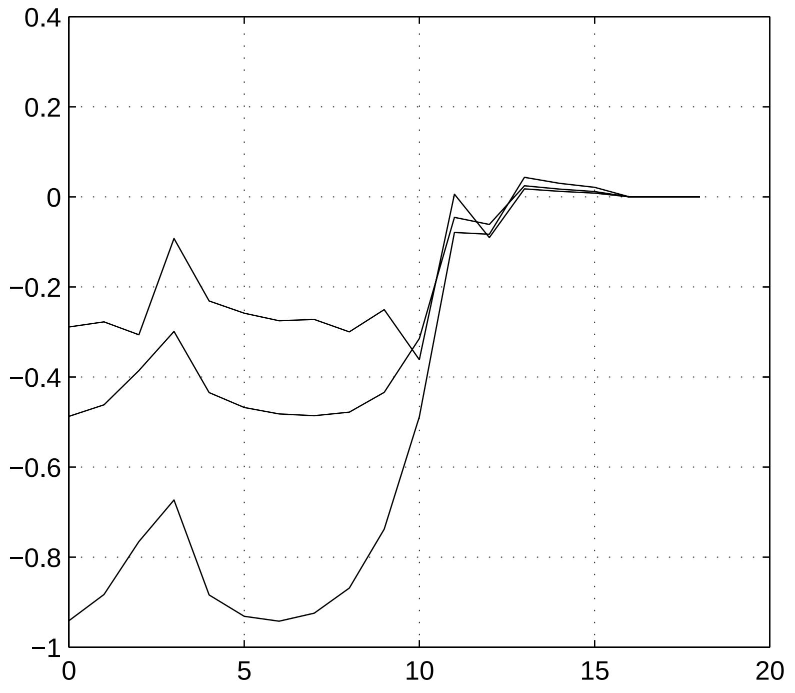

Figure 6.7: The regulated state variables x_2, x_3 and x_4.

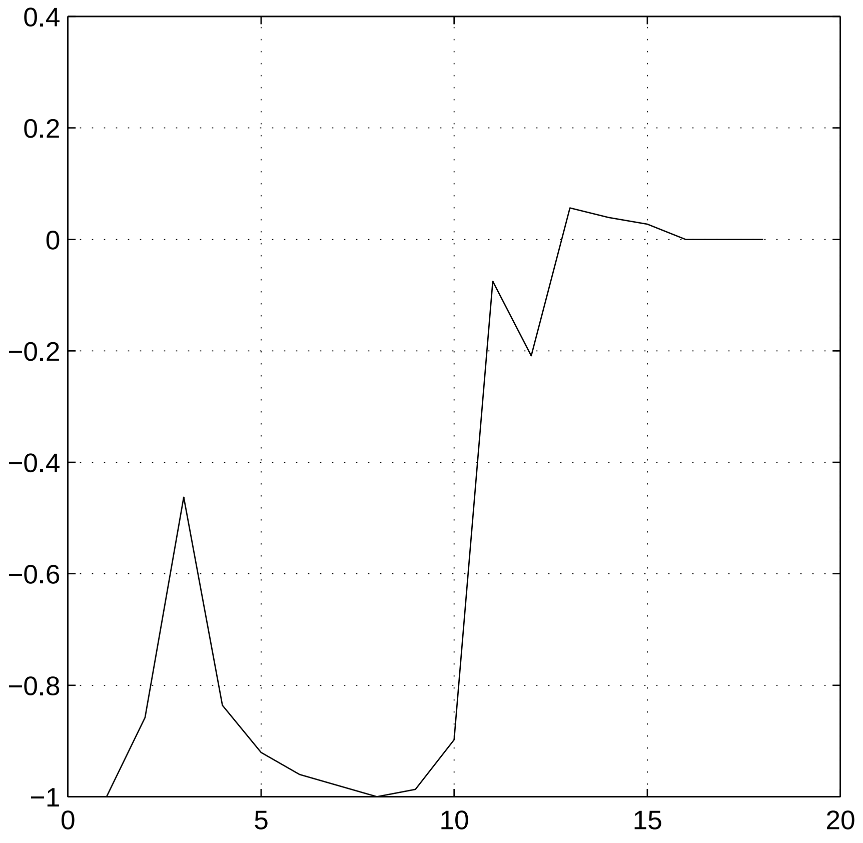

Figure 6.8: The optimal control signal u.

Chapter 7

Summary and Conclusions

7.1 Results and Findings

In this thesis, we have considered issues in controlling a linear discrete-time system when there is saturation on the control input and the state is measurable.

In Chapter 2, representation of controllable sets of this class of systems is examined first. A new result regarding the representation of controllable sets by their faces is given in the Main theorem. We used that new result to design an algorithm which calculates the facial representation of a K-step controllable set without generating inactive inequalities. The algorithm presented is an improvement to existing solutions based on the Fourier-Moltzkin projection algorithm and linear programming. It was shown also that the expected number of faces in a K-step controllable set is $2\binom{mK}{n-1}$, where n is the order of the system and m is the number of control inputs.

In Chapter 3, our attention is turned to the design of a nonlinear state feedback controller for single-input anti-stable system. The main result in this chapter is a minimum-time (time-optimal) control law which stabilizes the given system on its

K-step controllable set. First, it was shown how to characterize all minimum-time controls using controllable sets and then between all of those optimal controls, we proposed the one which is also optimal in the sense of robustness and disturbance rejection. We showed that the implementation of the proposed controller requires finding the intersection of a line with the K-step controllable set; this can be computationally intensive as the complexity of the controllable set increases with the order of the system.

The aforementioned computational problem is addressed in Chapter 4 where we presented two different approaches for solving the given problem. In the first approach we used a facial representation of the controllable set to design an efficient algorithm for on-line computation of the optimal control. Complexity analysis of the algorithm has shown that it can be practically implemented on a low-cost computer architecture even for the systems with 4-5 unstable poles. In the second approach it is demonstrated that our computational problem can be restated as a bounded variable LP problem. Then, it is shown that using the dual simplex method this problem can be solved much faster in comparison to the first approach. A surprising result is that the solution based on facial representation is computationally equivalent to solving a corresponding bounded variable LP problem in a naive fashion by looking at all its extreme points. All results we developed for single input anti-stable systems can be generalized to multi-input systems and systems with both stable and unstable poles.

Results of simulation and obtained closed-loop response in two examples are given in Chapter 6, and show superior performance of the presented time-optimal controller.

7.2 Future Research

Implementation of the proposed time-optimal controller based on the dual simplex method is very fast and practically implementable even for high order systems, e.g. for $n = 15$ the corresponding LP problem is still considered small in LP theory. According to the author's knowledge the presented solution is currently the fastest time-optimal control for the problem we considered. However, it seems that there is still a lot of room for improvements.

The dual simplex method is a general method for solving bounded variable linear programs. In our case the corresponding bounded variable LP has a special structure, e.g. columns in the matrix W are exponentially decreasing as well as the cost coefficients in the objective function. It may be interesting to see whether we can modify the dual simplex method to make use of this special structure and reduce the number of expected iterations. There is also a possibility that an interior-point method for solving linear programs can be more efficient in our case.

Finally the results of closed-loop response presented in Chapter 6 are obtained in simulation only. Experimental examination and implementation in real-time would be desirable to confirm whether this approach is practically feasible.

Bibliography

[1] D. S. Berstein and A. N. Michel, *A Chronological Bibliography on Saturating Actuators*, Int. Journal of Robust and Nonlinear Control, **vol. 5**, pp. 375-380, 1995.

[2] P. Bruck, *A New Result in Time Optimal Control of Discrete Systems*, IEEE Transaction on Automatic Control, **AC-19**, pp. 597-598, 1974.

[3] R. J. Caron, J. F. McDonald and C. M. Ponic , *Classification of Linear Constraints as Redundant or Necessary*, Dept. of Mathematics, Univ. of Windsor, Windsor, Canada, Windsor Math. Rep. WMR 85-09, Nov. 1985

[4] C. T. Chen, *Linear System Theory and Design*, Oxford University Press, New York, 1999.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, 1989.

[6] C. A. Desoer and J. Wing, *An Optimal Strategy for a Saturating Sampled-Data System*, IRE Transaction on Automatic Control, **AC-6**, pp. 5-15, Feb. 1961.

[7] C. A. Desoer and J. Wing, *A Minimal Time Discrete System*, IRE Transaction on Automatic Control, **AC-6**, pp. 111-125, May 1961.

[8] C. A. Desoer and J. Wing, *Minimal Time Regulator Problem for Linear Sampled-data Systems (General Theory)*, J. Franklin Inst., **vol. 271**, pp. 208-228, Sep. 1961.

[9] M. E. Evans and D. N. P. Murthy, *Controllability of Discrete Time Systems with Positive Controls*, IEEE Transaction on Automatic Control, **AC-22**, pp. 943-945, 1977.

[10] G. Ewald, *Combinatorial Convexity and Algebraic Geometry*, Springer-Verlag, New York, 1996.

[11] A. Feuer and M. Haymann, *Admissible Sets in Linear Feedback Sysytems with Bounded Controls*, International Journal of Control, **23**, pp. 381-392, 1976.

[12] M. E. Fisher. and J. E. Gayek, *Estimating Reachable Sets for Two Dimensional Linear Discrete Systems*, J. Opt. Th. Appl., **56**, pp. 67-88, 1987.

[13] P. O. Gutman and M. Cwikel, *Admissible Sets and Feedback Control for Discrete-Time Linear Dynamical Systems with Bounded Controls and States*, IEEE Transaction on Automatic Control, **AC-31**, pp. 373-376, pp. 457-459, 1986.

[14] T. Hu, D. E. Miller, L. Qiu, *Controllable Regions of LTI Discrete-Time Systems with Input Saturation*, Proc. of the 37th IEEE CDC, WA13-1, 1998.

[15] T. Hu, D. E. Miller, L. Qiu, *Null Controllability and Stabilization of LTI Discrete-time Systems with InputSaturation*, to apper.

[16] T. Hu, L. Qiu and Z. Lin, *Controllabilty and Stabilization of Unstable LTI Systems with Input Saturation*, Proc. of the 36th IEEE CDC, pp. 4498-4503, 1997.

[17] T. Hu and L. Qiu, *Controllable Regions of Linear Systems with Bounded Inputs*, System & Control Letters, **33**, pp. 55-61, 1998.

[18] R. E. Kalman, *Optimal Nonlinear Control of Saturating Systems by Intermittent Action*, IRE Wescon Convention Record, **4**, pp. 130-135, 1957.

[19] K. S. Keerthi and E.G. Gilbert , *Computation of Minimum-Time Feedback Control Laws for Discrete-Time Systems with State-Control Constraints*, IEEE Transactions on Automatic Control, **AC-32**, pp. 432-435, 1987.

[20] J. B. Lasserre, *On Reachable and Controllable Sets for Two-Dimensional Linear Discrete-time Systems*, J. Opt. Th. Appl., **70**, pp. 583-595, 1991.

[21] J. B. Lasserre, *Reachable ,Controllable Sets and Stabilizing Control of Constrained Linear Sysytems*, Automatica, **29**, pp. 531-536, 1993.

[22] J. N. Lin, *Determination of Reachable Set for a Linear Discrete System* , IEEE Transaction on Automatic Control, **AC-15**, pp. 339-342, 1970.

[23] Z. Lin and A. Saberi, *A Semi-Global Low-and-High Design for Linear Systems with Input Saturation: Stabilization and Disturbance Rejection*, Int. Jounal of Robust and Nonlinear Control, **5**, pp. 381-398, 1995.

[24] K. G. Murty, *Linear Programming*, Wiley, New York, 1983.

[25] S. A. Robertson, *Polytopes and Symmetry*, Cambridge University Press, New York, 1983.

[26] W. E. Schmitendorf and B. R. Barmish, *Null Controllability of Linear Systems with Constrained Control*, SIAM Journal of Control and Optimization, **18**, pp. 327-345, 1980.

[27] J. S. Shamma, *Optimization of the l^∞-Induced Norm Under Full State Feedback*, IEEE Transactions on Automatic Control, **AC-41**, pp. 533-544, 1996.

[28] S. S. Skiena, *Algorithm Design Manual*, TELOS-the Electronic Library of Science, Santa Clara, 1998.

[29] E. D. Sontag, *An Algebraic Approach to Bounded Controllabillity of Linear Systems*, Int. Journal of Control, **39**, pp. 181-188, 1984.

[30] H. J. Sussmann, E. D. Sontag and Y. Yang, *A General Result on the Stabilization of Linear Systems Using Bounded Controls*, IEEE Transaction on Automatic Control, **AC-39**, pp. 2411-2425, 1994.

[31] A. R. Teel, *Global Stabilization and Restricted Tracking for Multiple Integrators with Bounded Controls*, Systems & Control Letters, **vol. 18**, pp. 165-171, 1992.

[32] R. P. V. Til and W. E. Schmitendorf, *Constrained Controllability of Discrete-Time Systems*, International Journal of Control, **43**, pp. 941-956, 1986.

[33] R. Webster, *Convexity*, Oxford University Press, New York, 1994.

[34] G. M. Ziegler, *Lectures on Polytopes*, Springer-Verlag, New York, 1995.