MASTER

l1 optimal control using controlled invariance

Philips, M.C.J.

Award date:
2018

ℓ^1 Optimal Control Using Controlled Invariance

M.C.J. Philips
Eindhoven University of Technology
m.c.j.philips@student.tue.nl

Supervisors: L.C.G.J.M. Habets
S. Weiland
D.P.M. van den Hurk

Program: Industrial and Applied Mathematics

Student number: 0809154

October 18, 2018

Abstract

This thesis considers the ℓ^1 optimal control problem. The objective of ℓ^1 optimal control is to minimize the effects of disturbances, which are measured by the ℓ^∞ norm. We consider a state-space representation of a linear dynamical system, which is discrete in time. We use an alternative approach to find an ℓ^1-optimal controller. In ℓ^1-optimal control, it is desired to find a feedback controller which aims to minimize the amplitude of the output, given a bounded persistence disturbance. In this thesis, we desire to find a feedback controller that aims to maximize the bound of the disturbance, such that for any kind of disturbance satisfying this bound, the output meets certain constraints.

This idea is based on a paper by Jeff Shamma [10]. Shamma discusses an algorithm to construct a subset of the state space that meets our constraint on the output, given the bound of the disturbance. The resulting set is called the controlled invariant set. By using a well-chosen notation, he is able to construct (various) input vectors which make sure that the next state vector remains in the controlled invariant set, regardless of the specific value of the disturbance. The algorithm provides the maximal bound of the disturbance for which the controlled invariant set exists.

In this thesis, we improve the original algorithm of Shamma in various ways. As Shamma's algorithm requires some restrictive assumptions on the system, we determined a method where we do not need these assumptions, while being able to obtain the same results. Besides that, we also improve the efficiency of the algorithm in this thesis. When the number of states and the number of controls become bigger, the algorithm needs to solve many finite linear programs. Therefore, the computational time increases rapidly. By keeping track of which calculations have already been done in the previous steps of the algorithm, we make sure that no unnecessary calculations are done. In this way the number of linear programs decreases significantly. We also implement the possibility to add extra constraints on the maximum values that the controlled input can attain. This is done for practical purposes. In theory, it may be possible to exert an enormous amount of power. However, in industrial applications there exists a limit to that. As a result, we give the possibility to add these restrictions for the controlled input. Finally, when the controlled invariant set does not exist, we are still able to make conclusions about the performance based on the completed steps of the algorithm.

Contents

Chapter 1

Introduction

When driving a car with the ability to use cruise control, people normally do not think about how this is achieved. Regardless of whether the car is turning, going up or down the hill, or is just simply going straight, when using cruise control a constant desired velocity is maintained. This is all realized with the use of control theory.

Control theory is concerned with the control of dynamical systems in engineering processes and machines. The objective of control theory is to impose a certain desired behavior on a system. We can categorize controllers in two main types:

1. Open-Loop controllers.

2. Feedback / Closed-loop controllers.

To explain the difference, we consider the example of cruise control in a car.

Suppose that, while cruise control is activated, the power of the engine is controlled. When using an open-loop controller, we would have to determine the amount of power that is needed from the engine before we start the cruise control. If the road is straight the entire drive, then the engine needs to maintain the same amount of power from beginning to end. If we happen to know that there is a hill on our route, then we can take this into account. We can adjust the amount of power at specific points in time, such that the speed will remain the same during the whole drive. Similarly, we can take into account any turns we have to make. But what happens when we are halfway to our destination and suddenly there is a very strong wind blowing in opposite direction. If it is hard enough, then it slows down our car. Meaning that it will take longer to get to the hill that is coming up. We did not calculate the effect of the wind before the start of the drive. As a result, there exists a point in time where we exceed our desired speed, because the controller says that we should increase the power, but we have not yet arrived at the start of the hill. This problem can be overcome by using a feedback controller.

When using a feedback controller, the controlled input is based on a certain measured quantity. Suppose that it measures the velocity of the car. When the wind starts blowing, resulting in the car slowing down a bit, this will be measured. The controller reacts on it, by giving a signal to the system that the amount of power needs to be increased. Because the amount of power increases, the speed of the car increases. If the amount of power becomes so high that the car starts driving faster than its desired speed, then this is measured again. As a reaction the controller corrects the amount of power again. This goes on in the same manner, until the desired speed is reached again.

The main difference between these two types of controllers is that an open-loop controller acts without knowing what the system actually does, while a feedback controller is acting based on some kind of measured output. In this thesis we consider feedback controllers only.

In optimal control, we define the control objective as a cost function that we want to maximize or minimize. Examples for objectives are tracking problem, performance robustness and disturbance rejection. In this paper we consider the objective to be the latter. Disturbance rejection means that we try to find a controller such that the output of the system maintains a certain performance for some kind of restricted disturbance. Obviously, if the disturbance did not have any restrictions and is unpredictable, it is very hard to create a controller that guarantees the system would remain stable and achieves a desired performance. We can phrase these conditions as an induced norm optimization problem. For this, we define an operator H, describing the relation from the disturbance input w to the output z. Next we define the induced norm on H as

$$||H||_{\cdot} = \sup_{w \neq 0} \frac{||z||_{\cdot}}{||w||_{\cdot}}.$$

Note that we do not specify the type of the norms yet. In the optimization problem, we want to find a feedback controller to minimize this induced norm. Hence, given the value of the norm of the disturbance, we would like to construct a feedback controller such that the norm of the output divided by the norm of the input is minimized. The final question is, what norm best captures the behavior we want to impose on our system? For this we discuss two concepts, $\mathcal{H}^{\infty}$-optimal control and ℓ^1-optimal control.

In $\mathcal{H}^{\infty}$-optimal control problems, the performance is measured in terms of the $\mathcal{L}_2$-induced norm. When applying this norm in our optimization problem, physically speaking, we minimize the maximum possible amplification of signal energy. The big advantage of these norms is that they are derived from an inner product. The maximal energy of the disturbance will be given, and we must determine a feedback controller such that the energy of the output is minimized. There are many ways to derive such a controller, e.g., state space solutions as shown in [5] and references therein. However, we do encounter a problem when using these norms. The performance is measured by energy in this case, while there are many cases where we are more interested in the performance measured in amplitude. Therefore we consider ℓ^1-optimal controllers.

ℓ^1-optimal control considers the performance measured in terms of the ℓ^∞-induced norm. Physically speaking, it measures the worst case amplification of the maximum amplitude of a disturbance signal to its output. Therefore, we want to construct a feedback controller to minimize

$$||H||_1 = \sup_{w \neq 0} \frac{||z||_\infty}{||w||_\infty}.$$

Unlike $\mathcal{L}_2$ norms, the ℓ^∞ norms are not based on an inner product. However, these norms are often more useful for problems in physical aspect. Consider the following examples.

When using a printer, to print this thesis for instance, it is desired that the letters do not overlap. If they do overlap, words become unclear, which makes it very hard to read the text. We describe the system of the printer in the following way. The ink roller is put on the paper just as the computer commands it. If the printer can follow the commands of the computer exactly, then we have our desired reference output z. However, if our printer has aged, some parts of it will wear out and it will be harder to print the letters exactly at the desired location. We can see the wear of the printer as a disturbance.

As long as the printed letters are not yet overlapping, the printer still works properly. However, if the maximum disturbance becomes to big, the letters will start overlapping and the print quality cannot be guaranteed anymore. Hence, we want a feedback controller that minimizes the amplitude of the output, given a maximal amplitude of the disturbance.

We can also consider our ℓ^1 control problem in another way. Consider the cruise control example again. As stated before, we want to find a feedback controller such that a constant desired velocity is maintained. We can define the reference output as the desired velocity of the car. We say that the cruise control is behaving adequate if the velocity does not deviate with more than 1 km/h from the desired velocity. In mathematical terms, we desire $||z||_\infty \leq 1$. We want to construct a feedback controller such that the desired performance is achieved for any kind of disturbance. Since this is not possible, we would like to maximize the amplitude of the disturbance for which there still exists a feedback controller such that $||z||_\infty \leq 1$ is achieved. This is exactly the problem we discuss in this thesis.

Over the years, there have been written many papers, books and articles about ℓ^1 optimal control. Blanchini and Miani wrote a book, [2], about a set-theoretic approach in control systems. The book has many intersections with other areas in control theory, making it a useful source. Some of the references in this book consider a state space approach, in particular [4], [10] and [11]. The paper that stands out is the one from Shamma [10]. Together with his earlier work, [9], we consider his work in this thesis.

The remainder of this thesis is organized in the following way. Chapter 2 presents the notation used in this thesis. Along with a detailed problem formulation, we state which approach we use to find an ℓ^1 controller. Chapter 3 provides some background materials, regarding set-valued maps. Chapter 4 is concerned with the notion of controlled invariance, which is the central concept in this thesis, and the notion of a controlled invariant kernel. Chapter 5 shows that the notions from Chapter 4 can be applied to the problem formulated in Chapter 2. The numerical implementation of the controlled invariant kernel algorithm is described in Chapter 6. Chapter 7 addresses the novelty of the developed approach. Numerical tests are presented in Chapter 8 and Chapter 9, starting with academic examples, we tested the method for an industrial application. Finally, we state some concluding remarks and perspectives in Chapter 10.

Chapter 2

State Space Systems and Feedback Control

2.1 Notation

We state some of the notation regarding norms and vector spaces, that will be used throughout this paper. The set $\mathbb{N}$ is all integers, including 0. The halfspace of all nonnegative real numbers is denoted as $\mathbb{R}_+$. Let $\ell_n^\infty(\mathbb{N})$ denote the vector space of all bounded one-sided sequences in $\mathbb{R}^n$.

For any matrix $A \in \mathbb{R}^{m\times n}$ we define the following:

$$\begin{aligned}
A_{(i,j)} &\text{ is the element of } A \text{ on row } i \text{ and column } j,\\
A_{(i,:)} &\text{ is the } i^{th} \text{ row of matrix } A,\\
|A_{(i,:)}| &= \sum_{j=1}^{n} |A_{(i,j)}|,\\
|A| &= \max_{i=1,\dots,m} |A_{(i,:)}|.
\end{aligned}$$

For any vector $x \in \mathbb{R}^n$ we define:

$$\begin{aligned}
x_i &\text{ denotes the } i^{th} \text{ component of vector } x,\\
|x| &= \max_{i=1,\dots,n} |x_i|, \text{ which we call the norm of a vector.}
\end{aligned}$$

For any sequence $f \in \ell_n^\infty(\mathbb{N})$ we define

$$||f|| = \sup_{k\in\mathbb{N}} |f(k)|, \text{ which we call the amplitude of a vector.}$$

For any operator $T : \ell_n^\infty(\mathbb{N}) \to \ell_m^\infty(\mathbb{N})$ we have the usual operator norm, defined as

$$||T|| = \sup_{\substack{f\in\ell_n^\infty(\mathbb{N})\\ ||f||>0}} \frac{||Tf||}{||f||}.$$

2.2 Problem Description

We consider a linear, time-invariant state-space system $\mathcal{S}$ that is discrete in time. The dynamics are described in the following way.

$$x(k+1) = Ax(k) + B_1w(k) + B_2u(k), \tag{2.1}$$
$$z(k) = Cx(k) + D_1w(k) + D_2u(k), \tag{2.2}$$
$$y(k) = x(k), \tag{2.3}$$
$$x(0) = 0, \tag{2.4}$$

where $k \in \mathbb{N}$. At any point in time, the system has a current state vector, denoted as $x(k) \in \mathbb{R}^n$. The system has two inputs, $u(k) \in \mathbb{R}^m$ and $w(k) \in \mathbb{R}^q$, representing the controlled input and disturbances acting on the system, respectively. The outputs of the system are denoted by the measured output $y(k) \in \mathbb{R}^n$ and the regulated outputs are $z(k) \in \mathbb{R}^p$. Based on this we can say that our system is a mapping
$\mathcal{S} : \ell_m^\infty(\mathbb{N}) \times \ell_q^\infty(\mathbb{N}) \to \ell_p^\infty(\mathbb{N}) \times \ell_n^\infty(\mathbb{N})$. The measured output is used as input for a feedback controller $\mathcal{K}$. It is a standing assumption that the whole state is measured, making the controller $\mathcal{K}$ a full state feedback controller. The controller $\mathcal{K} : \mathbb{R}^n \to \mathbb{R}^m$ is defined as a mapping from state vector $x(k)$ to the controlled input $u(k)$. The matrices A, B_1, B_2, C, D_1 and D_2 are assumed to be static and to have the appropriate dimensions. The dynamics of the closed-loop system are depicted in Figure 2.2.

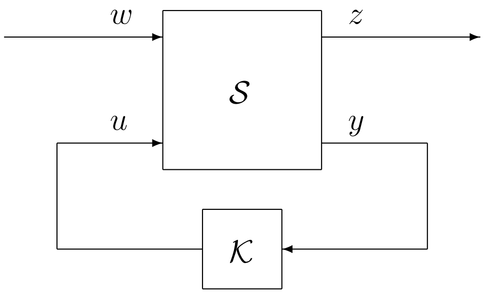

Figure 2.2: Graphical representation of a state space system $\mathcal{S}$ with feedback controller $\mathcal{K}$.

The closed-loop transfer from disturbance w to output z can be described by the mapping $\mathcal{S}_\mathcal{K} : \ell_q^\infty(\mathbb{N}) \to \ell_p^\infty(\mathbb{N})$ defined as

$$\mathcal{S}_\mathcal{K}(w) = z.$$

Our main goal is to find a controller $\mathcal{K}$, such that we minimize $||\mathcal{S}_\mathcal{K}||$, since it can be written as

$$||\mathcal{S}_\mathcal{K}|| = \sup_{\substack{w \in \ell_q^\infty(\mathbb{N}) \\ w \neq 0}} \frac{||z||}{||w||}.$$

In order to determine the controller, we start by stating what forms of $\mathcal{K}$ are allowed.

Definition 2.2.1. *We define two admissible classes for $\mathcal{K}$.*

1. *Linear dynamic state feedback. Then the mapping $\mathcal{K}$ is of the form*

$$\begin{aligned} x_{\mathcal{K}}(k+1) &= A_{\mathcal{K}} x_{\mathcal{K}}(k) + B_{\mathcal{K}} y(k), \\ u(k) &= C_{\mathcal{K}} x_{\mathcal{K}}(k) + D_{\mathcal{K}} y(k), \end{aligned}$$

   where $x_{\mathcal{K}}(0) = 0$. The matrices $A_{\mathcal{K}}$, $B_{\mathcal{K}}$, $C_{\mathcal{K}}$ and $D_{\mathcal{K}}$ have the appropriate dimensions.

2. *Nonlinear static state feedback. Then the mapping $\mathcal{K}$ is of the form*

$$u(k) = g(x(k)),$$

   where $g : \mathbb{R}^n \to \mathbb{R}^m$ is continuous and $g(0) = 0$.

Now we know which forms of $\mathcal{K}$ are considered. But any controller must satisfy certain conditions. Besides, we would like to measure the performance of the controller in some way. Therefore we introduce the following definition.

Definition 2.2.2. *Let $\mathcal{K}$ be an admissible controller. Then $\mathcal{K}$ is said to be internally stabilizing with performance γ if it satisfies the following conditions:*

1. *For $w = 0$, the system is globally exponentially stable. In other words, the state will decay exponentially to zero over time.*

2. *$||\mathcal{S}_{\mathcal{K}}|| \leq \gamma$.*

We define $\gamma_{optimal}$ as

$$\gamma_{optimal} = \inf_{\mathcal{K} \text{ admissible}} \left\{ ||\mathcal{S}_{\mathcal{K}}|| : \mathcal{K} \text{ internally stabilizing} \right\}.$$

Our main goal is to construct an algorithm such that we can determine an optimal admissible internally stabilizing controller $\mathcal{K}$ with performance γ for a linear system, where γ can be determined as close to $\gamma_{optimal}$ as is desired.

2.3 Invariance of Subsets in the State Space Approach

According to Shamma's earlier work in [9], we have the following theorem.

Theorem 2.3.1. *If there exists a linear dynamic feedback controller that is internally stabilizing with a performance of γ, then there exists a continuous nonlinear static feedback controller that is internally stabilizing with a performance of γ.*

In the proof of this theorem, it is assumed that there exists a linear dynamic feedback controller that derives a subset of the state space which can be made invariant under feedback. We define invariance under feedback later, in Definition 4.0.5. The invariance of this subset implies achieving the desired performance. Shamma continues by showing that there must exist a nonlinear static feedback which makes the same subset invariant and therefore also achieves the desired performance.

Hence, we can construct a nonlinear static feedback controller with a performance that is equal to or even lower than the performance of an optimal linear dynamic feedback controller. Therefore, we want to construct a nonlinear static feedback controller $\mathcal{K}$ that is internally stabilizing with a performance that is desirably close to $\gamma_{optimal}$.

The proof of Theorem 2.3.1 states that a subset of the state space can be made invariant under feedback. In order to obtain this subset, we are going to construct the so-called Controlled Invariant Kernel algorithm. It is based on the algorithm of Shamma, described in [10]. With a few adjustments, we want to improve the speed and reduce the number of inequalities that are needed in the original algorithm. In order for us to use the original algorithm in the first place, we need to introduce set-valued maps and some of their properties.

Chapter 3

Set-valued Maps

The definition of a set-valued map is very intuitive. Instead of mapping one point to another, we can map one point to a set. Although the term set-valued map is not so common, it is relatively easy to understand from the following example.

Example 3.0.1. Consider the inverse map of the sine function. We know that $\sin(x) = 0$ for $x = k\pi$, where $k \in \mathbb{Z}$. Therefore, we see that the pre-image of 0 can be defined as

$$\sin^{-1}(0) = \{\ldots, -2\pi, -\pi, 0, \pi, 2\pi, \ldots\}.$$

Similarly, the pre-image of $\sin(x) = y$ is a set for any $y \in [-1, 1]$. As a result, we can say that $\sin^{-1}$ is a set-valued map.

□

Most inverse functions of surjective maps can be called set-valued maps. Set-valued maps are used in optimal control theory, as in [10],[9], and viability theory [1]. Here it is used when there is a parameter uncertainty. In this thesis, the system is perturbed by an unknown disturbance, while we want to guarantee a certain performance of the system. Only the maximum amplitude of the disturbance is known. Based on that, we can define a set-valued map.

Definition 3.0.1. *A set-valued map is a map $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$, where some value $x \in \mathbb{R}^n$ has a multi-valued image. In other words, $F(x) = D$, with $D \subset \mathbb{R}^m$.*

$\mathcal{P}$ is defined as in Definition 3.0.2.

Definition 3.0.2. *We define $\mathcal{P}(\mathbb{R}^n)$ as the power set of $\mathbb{R}^n$, i.e., the set of all subsets of $\mathbb{R}^n$.*

With these definitions we can define a set-valued map for our regulated output, that we defined in (2.2). For example, if we have a single-input single-output (SISO) system and we know that the disturbance has the restriction $-10 \leq w \leq 10$, then $z(k) \in F(x(k), u(k))$, where the set-valued map F is defined as

$$F(x(k), u(k)) = \{Cx(k) + B_2u(k) + B_1w : w \in [-10, 10]\},$$

assuming $B_1 \neq 0$. Notice that we write $z(k) \in F(x(k), u(k))$ and not $z(k) = F(x(k), u(k))$. This is due to the fact that when the system is active, the disturbance will assume only one value. Hence, $z(k)$ is single-valued. However, we do not know its exact value, only that it is contained in $F(x(k), u(k))$.

We can do something similar for the state equation (2.1), even though this is a difference equation. In this difference equation, we calculate the next step of the state. We do not know its exact value, but we know the set in which it is contained, based on the restrictions of the disturbances as described above. Hence, we can define a set-valued map F such that $x(k+1) \in F(x(k), u(k))$. This is called a difference inclusion.

Definition 3.0.3. *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathcal{P}(\mathbb{R}^n)$ be a set-valued map. Then*

$$\begin{aligned} x(k+1) &\in F(x(k), u(k)), \text{ for every } k \in \mathbb{N}, \\ x(0) &= x_0, \end{aligned}$$

is called a difference inclusion.

In order to apply the algorithm of Shamma [10], we need to define multiple properties of set-valued maps. We start with the following.

Definition 3.0.4. *The domain of a map $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ is defined as*

$$dom(F) = \{x \in \mathbb{R}^n : F(x) \neq \emptyset\}.$$

Continuity is very different for set-valued maps than for single-valued maps. For set-valued maps there are multiple concepts of continuity, all with different properties. For single-valued maps, we have the following definition.

Definition 3.0.5. *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be continuous if*

$$\forall x_0 \in \mathbb{R}^n \; \forall \varepsilon > 0 \; \exists \delta > 0 \text{ such that } \forall x \in \mathbb{R}^n : 0 < |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \varepsilon.$$

This means that for any vector x that is close enough to x_0, the value of $f(x)$ is close to the value of $f(x_0)$. Unfortunately, we cannot simply say that the values of one set are close to the values of another set. Hence, for set-valued maps different concepts of continuity are required. There are two concepts for continuity that we consider in this thesis. They are called lower semicontinuity and upper semicontinuity.

Definition 3.0.6. [1, p.57] *A set-valued map $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$ is called lower semicontinuous if*

> *$\forall x \in dom(F)$, $\forall y \in F(x)$ and for any sequence $\{x_n\}_{n\in\mathbb{N}} \subseteq dom(F)$, where $x_n \to x$, there exists a sequence $\{y_n\}_{n\in\mathbb{N}}$ such that $y_n \in F(x_n)$ $\forall n \in \mathbb{N}$ and $y_n \to y$.*

Intuitively it means that for any fixed $x \in dom(F)$, where F is a lower semicontinuous set-valued map, we have the following. Suppose we construct a sequence with x as its limit. Every point in this sequence is projected to a set by F. Then, for any point $y \in F(x)$ we can construct a sequence that is contained in the projected sets and has y as its limit.

Definition 3.0.7. [1, p.56] *A set-valued map $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$ is called upper semicontinuous if*

1. *$dom(F)$ is closed; and*
2. *$\forall x \in dom(F)$ and $\forall \varepsilon > 0 \exists \delta > 0$ such that $x' \in dom(F)$ and $|x' - x| < \delta$ together imply $\sup\limits_{y' \in F(x')} \inf\limits_{y \in F(x)} |y' - y| < \varepsilon$.*

Different from lower semicontinuity, upper semicontinuity has nothing to do with sequences. Let F be an upper semicontinuous set-valued map. Then the definition describes that for any $x \in dom(F)$ and for any $\varepsilon > 0$ we can determine some $\delta > 0$ with the following properties. For any $x' \in dom(F)$ that is close enough to x (based on δ), we know that $\sup\limits_{y' \in F(x')} \inf\limits_{y \in F(x)} |y' - y| < \varepsilon$. This statement seems complicated, but is actually quite simple. We break it down step by step. First we have $|y' - y|$, which is simply the distance between vectors y and y'. Then we consider $\inf\limits_{y \in F(x)} |y' - y|$. Note that y' is some value of $F(x')$. This statement describes the distance of the vector y' to the set $F(x)$. We can see that if $y' \in F(x)$, then $\inf\limits_{y \in F(x)} |y' - y| = 0$. For any $y' \notin F(x)$, the statement has a value greater than 0, which is also possible. This is where the supremum is applied. The full statement says that the maximal distance of any vector in $F(x')$ to the set $F(x)$ will be less than ε, given that the distance between x and x' is less than δ.

To fully grasp the concepts of both lower and upper semicontinuity, we state two examples in which it is proven whether a function satisfies the definitions or not.

Example 3.0.2. Define the set-valued map $F : \mathbb{R} \to \mathcal{P}(\mathbb{R})$ as

$$F(x) = \begin{cases} 0, & \text{for } x \neq 0, \\ [-1, 1], & \text{for } x = 0. \end{cases}$$

Since the function is obviously continuous for $x \neq 0$, we only look at the point $x = 0$.
To disprove lower semicontinuity, we choose $y \in F(0)$ as $y = 1$ and let $x_n = \frac{1}{n}$. Since $F(x_n) = 0 \; \forall n \in \mathbb{N}$, we have $y_n = 0 \; \forall n \in \mathbb{N}$. Hence, $y_n \not\to y$.
We can prove that this set-valued map is upper semicontinuous. Again, we only need to prove the conditions are met for $x = 0$. Take ε and δ fixed. For any $x' \in dom(F)$ such that $|x'| < \delta$ it should hold that

$$\sup_{y' \in F(x')} \inf_{y \in [-1,1]} |y' - y| < \varepsilon.$$

Regardless of the value of x', we have $y' \in [-1,1]$. Therefore,

$$\sup_{y' \in F(x')} \inf_{y \in [-1,1]} |y' - y| = \sup_{y' \in F(x')} 0 = 0 < \varepsilon.$$

Hence, this set-valued map is upper semicontinuous and it is not lower semicontinuous in 0.

$\square$

Example 3.0.3. Define the set-valued map $F : \mathbb{R} \to \mathcal{P}(\mathbb{R})$ as

$$F(x) = \begin{cases} [-1,1], & \text{for } x \neq 0, \\ 0, & \text{for } x = 0. \end{cases}$$

Just as in the previous example, we only look at $x = 0$.
We can easily prove lower semicontinuity. We have $y = 0$, since $F(0) = 0$. Regardless of our values of x_n, we have $y_n \in F(x_n) = [-1,1]$. A simple sequence to satisfy our condition would be $y_n = \frac{1}{n}$, given $n > 0$. Then our condition is met, since $\frac{1}{n} \to 0$.
Here we disprove upper semicontinuity. Let $\varepsilon = \frac{1}{2}$ and choose $\delta > 0$ arbitrarily. We choose $x' \neq 0$ such that it still satisfies $|x' - x| < \delta$. Then we find

$$\sup_{y' \in [-1,1]} \inf_{y \in \{0\}} |y' - y| = \sup_{y' \in [-1,1]} |y'| = 1 > \varepsilon.$$

Hence, this set-valued map is lower semicontinuous and it is not upper semicontinuous in 0.

$\square$

Based on these examples, we can conclude that upper and lower semicontinuity are not related to each other. But according to Aubin [1], we find that F is continuous at $x \in dom(F)$ if it is both lower and upper semicontinuous at $x \in dom(F)$. However, we do not discuss this in this thesis.

There are two more properties that are used in this thesis regarding set-valued maps. The first is locally boundedness. It is a definition that is the same for single-valued maps. Intuitively we can say that a map is locally bounded if the output is bounded for every bounded input. For example, even though the function $f(x) = x$ is unbounded over $\mathbb{R}$, the function is still bounded over a bounded set $K \subset \mathbb{R}$. Formally, we state the following.

Definition 3.0.8. *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$ be a set-valued map. Then F is called locally bounded if for every bounded set $K \subset \mathbb{R}^n$ there exists an $M \in \mathbb{R}$ such that*

$$|F(x)| \leq M, \quad \forall x \in K.$$

Finally, when using a single-valued map, we generally do not think about whether the image is open or closed. It is simply a point, which is closed. For a set-valued map, this does not have to be the case. We could define a map $F : \mathbb{R} \to \mathcal{P}(\mathbb{R})$ as

$$F(x) = (x, x+1).$$

Then the image of every $x \in \mathbb{R}$ is an open set. In this thesis, we require that any image of set-valued maps is closed. Therefore, we state the following definition.

Definition 3.0.9. *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$ be a set-valued map. We say that F is closed valued if $F(x) \subseteq \mathbb{R}^m$ is a closed set for every $x \in \mathbb{R}^n$.*

Chapter 4

Controlled Invariant Kernel Algorithm

In [10], Shamma introduced a Controlled Invariant Kernel (CIK) algorithm. The algorithm constructs a controlled invariant kernel, based on the maximum amplitude of the disturbance signal w, for which system states will always be contained inside the constructed kernel, regardless of the exact disturbance profile through time. This gives the major advantage that the exact disturbance profile does not need to be known, to guarantee performance of the system. We start with a notion of invariance, followed by an example.

Definition 4.0.1. *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$. A subset $K \subseteq dom(F)$ is invariant under F if $F(x) \subseteq K$ for every $x \in K$.*

Example 4.0.1. Let $F : [-2, 2] \to \mathcal{P}(\mathbb{R})$ be defined as

$$F(x) = [-x^2, x^2].$$

We have $dom(F) = [-2, 2]$. First we want to check if the domain of F is invariant under F. We find that $F(2) \nsubseteq [-2, 2]$, due to the fact that $F(2) = [-4, 4]$. Hence, $dom(F)$ is not invariant under F. We choose a subset $K = [-\frac{1}{2}, \frac{1}{2}] \subset dom(F)$. We have $F(\frac{1}{2}) = [-\frac{1}{4}, \frac{1}{4}] \supseteq F(x)$ for every $x \in K$. Since $[-\frac{1}{4}, \frac{1}{4}] \subset K$, we know that K is invariant under F.

$\square$

Invariance is a very useful property. When the set K is invariant under F for the next step, it consequently means that we can apply F multiple times to values of K, while never leaving the set K. From a control perspective it can therefore be useful to maximize the set K such that it is still invariant under F. Therefore, we define the following, based on [10].

Definition 4.0.2. *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ and $K \subseteq dom(F)$. The largest closed subset of K which is invariant under F is called the invariance kernel of K for F. It is denoted as $INV(K)$.*

By definition, we have $INV(K) \subseteq K$. $INV(K)$ always exists, but it may be empty. Meaning that there is no invariant kernel of K. When considering Example 4.0.1, we have $INV(K) = K$, since K is an invariant set. So this is not a very exciting example. The thing that we are interested in right now is $INV(dom(F))$. Recall that $dom(F)$ is not invariant under F, so what would be the largest closed subset of $dom(F)$ that is invariant under F? When looking at the definition of F, we can conclude that $INV(dom(F)) = [-1, 1]$. However, we want a more systematic way to construct this set, that is applicable to more complicated systems. That is why we define the following algorithm from [10].

Proposition 4.0.3 (Invariance Kernel Algorithm)**.** *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ be lower semi-continuous and let $K \subseteq dom(F)$ be closed. Define the subsets K_j recursively by*

$$\begin{aligned} K_0 &= K, \\ K_{j+1} &= \{x \in K_j : F(x) \subseteq K_j\}. \end{aligned}$$

Then

$$INV(K) = \bigcap_{j=0}^{\infty} K_j.$$

Proof. If $INV(K) = \emptyset$, then we know that there is no subset of K which is invariant under F. Therefore, we know that for every $\tilde{K} \subseteq K$ there exists some $x \in \tilde{K}$ such that $F(x) \not\subseteq \tilde{K}$. Since K_j are nested sets ($K_{j+1} \subseteq K_j$), we know that for j large enough we have $K_j = \emptyset$. Hence, $\bigcap_{j=0}^{\infty} K_j = \emptyset$. Then the inequality holds, because both are empty sets.

Now suppose $INV(K) \neq \emptyset$. By definition we find that $INV(K) \subseteq K_0$ and that for any $x \in INV(K)$ we have $F(x) \subseteq INV(K)$. Suppose for some j that we have $x \in K_j$, where $INV(K) \subseteq K_j$. If $x \in INV(K)$, then $F(x) \subseteq INV(K) \subseteq K_j$. Then we find that $INV(K) \subseteq K_{j+1}$. Since $INV(K) \subseteq K_0$, we find that $INV(K) \subseteq \bigcap_{j=0}^{\infty} K_j$.

Before we prove $INV(K) \supseteq \bigcap_{j=0}^{\infty} K_j$, we show that $\bigcap_{j=0}^{\infty} K_j$ is closed. Suppose K_j is closed. Let $\{x_n\}_{n\in\mathbb{N}}$ be a sequence in K_{j+1} which converges to $x \in K_j$. By lower semicontinuity of F, we know that for any $y \in F(x)$ there exists a sequence $\{y_n\}_{n\in\mathbb{N}}$ such that $y_n \in F(x_n)$ $\forall n \in \mathbb{N}$ and $y_n \to y$. Since $x_n \in K_{j+1}$, we know $F(x_n) \subseteq K_j$. Hence, $y_n \in K_j$. Because K_j is closed, we have $y \in K_j$. Since this holds for any $y \in F(x)$, we know $F(x) \subseteq K_j$. Hence, $x \in K_{j+1}$. Therefore K_{j+1} is closed if K_j is closed. Because K_0 is closed and the intersection of closed sets is closed, we know $\bigcap_{j=0}^{\infty} K_j$ is closed.

Finally, $x \in \bigcap_{j=0}^{\infty} K_j$ implies $F(x) \subseteq \bigcap_{j=0}^{\infty} K_j$. Therefore $\bigcap_{j=0}^{\infty} K_j$ is invariant under F. Hence, $INV(K) = \bigcap_{j=0}^{\infty} K_j$. ■

We can check whether we determined $INV(dom(F))$ from Example 4.0.1 correctly by applying the Invariant Kernel algorithm.

Example 4.0.2. Consider F from Example 4.0.1. Before using the Invariance Kernel algorithm, we first must check if F is lower semicontinuous. Let us choose an arbitrary $x \in dom(F) = [-2, 2]$. For any $y \in F(x) = [-x^2, x^2]$ and any sequence $\{x_n\}_{n\in\mathbb{N}} \in dom(F)$, where $x_n \to x$, we need to find a sequence $\{y_n\}_{n\in\mathbb{N}}$ with the properties $y_n \in F(x_n)$ $\forall n \in \mathbb{N}$ and $y_n \to y$. We separate two cases.
First, suppose that $y \in (-x^2, x^2)$. Then there must exist a large enough n such that $y \in F(x_n)$. So there exists an n_0 such that $y_n = y$ for every $n > n_0$.
Now let $y = \pm x^2$. Then we choose $y_n = \pm x_n^2 \in F(x_n)$. By continuity $x_n \to x$ implies $y_n \to y$.
Hence, F is lower semicontinuous. We choose $K_0 = dom(F) = [-2, 2]$. Then we have

$$K_1 = \big\{x \in [-2, 2] : [-x^2, x^2] \subseteq [-2, 2]\big\} = [-\sqrt{2}, \sqrt{2}].$$

For the next step we would have

$$K_2 = \Big\{x \in [-2, 2] : [-x^2, x^2] \subseteq [-\sqrt{2}, \sqrt{2}]\Big\} = [-\sqrt[4]{2}, \sqrt[4]{2}].$$

So we try to prove by induction that

$$K_j = [-\sqrt[2^j]{2},\ \sqrt[2^j]{2}].$$

Suppose that this is true for an arbitrary j. Then

$$\begin{aligned} K_{j+1} &= \Big\{x \in [-\sqrt[2^j]{2},\ \sqrt[2^j]{2}] : [-x^2, x^2] \subseteq [-\sqrt[2^j]{2},\ \sqrt[2^j]{2}]\Big\}, \\ &= [-\sqrt{\sqrt[2^j]{2}}, \sqrt{\sqrt[2^j]{2}}], \\ &= [-\sqrt[2^{j+1}]{2},\ \sqrt[2^{j+1}]{2}]. \end{aligned}$$

Since it is also true for $j = 1$, we know the exact set of every K_j. Then by taking the limit, we have

$$\begin{aligned} INV(dom(F)) &= \cap_{j=0}^{\infty} K_j, \\ &= \cap_{j=0}^{\infty} [-\sqrt[2^j]{2},\ \sqrt[2^j]{2}], \\ &= \lim_{j\to\infty} [-\sqrt[2^j]{2},\ \sqrt[2^j]{2}], \\ &= [-1, 1]. \end{aligned}$$

Hence, the algorithm constructs the same set that we determined earlier.

□

Evidently, this algorithm is working well. The big advantage of this is that we can determine the maximal amplitude of the state for the system, giving the initial value is contained in $INV(K)$. In Example 4.0.2, we have

$$x(k+1) \in F(x(k)) = [-x(k)^2, x(k)^2],$$

for any $x(0) \in [-1, 1]$, $k \in \mathbb{N}$, regardless of the specific values of $x(k)$, $k > 0$. We can determine that $||x|| \leq 1$. In this case, the result can be calculated easily, even without the algorithm. But we are missing one crucial part, which is the controlled input. In order for us to construct a similar algorithm with a controlled input, we need some more definitions. Instead of a standard difference inclusion, we are going to use an uncontrolled difference inclusion. The idea behind the uncontrolled difference inclusion is similar to the idea of a difference inclusion. When using a difference inclusion, the set-valued map determines all possible values that the next state can obtain, given the current state. For an uncontrolled difference inclusion these new possible states do not only depend on the current state, but also on all possible controlled inputs. We denote an uncontrolled difference inclusion in the following way, based on [10].

Definition 4.0.4. *Let $F : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ and $U : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^m)$ be set-valued maps and $f : \mathbb{R}^n \times \mathbb{R}^m \to dom(F)$ be a single-valued map. Define*

$$\tilde{F}(x(k)) = \left\{ \bigcup_{u(k) \in U(x(k))} F(f(x(k), u(k))) \right\}.$$

Then

$$x(k+1) \in \tilde{F}(x(k))$$

is the uncontrolled difference inclusion defined by (F, U, f).

Here, f is a function that relates the current state and the controlled input in a certain way, dependent on the system itself. $F(f(x(k), u(k)))$ states all possible values of the next state, given the current state and the controlled input. $U(x(k))$ denotes all possible values of the controlled input $u(k)$, given the state vector $x(k)$ and possibly desired constraints. In this thesis, we use the desired constraint $||z|| \leq 1$ to compute $U(x(k))$, as is described later in the next chapter. By taking the union for all $u(k) \in U(x(k))$ we have $\tilde{F}(x(k))$, which denotes all possible values of the next state, based on the current state and on every possible value of the controlled input.

Since the definition of an uncontrolled difference inclusion differs significantly from a standard difference inclusion, we propose a new notion of invariance, based on [10].

Definition 4.0.5. *Consider an uncontrolled difference inclusion defined by (F, U, f). A subset $K \subseteq dom(U)$ is controlled invariant under (F, U, f) if for every $x \in K$, there exists a $u \in U(x)$ such that $F(f(x, u)) \subseteq K$.*

In contrast with an invariant set, we do not choose a subset of $dom(F)$, but a subset of $dom(U)$. Hence, we choose a set K such that for every $x \in K$ there exists some input $u \in U(x)$. This set is called controlled invariant if we can find an input such that the state does not leave K. As a result, we have a set which satisfies desired constraints for every $k \in \mathbb{N}$.

Example 4.0.3. Consider the following functions.

$$\begin{aligned}
F(x) &= \left\{[-x^2, x^2] : x \in [-2,2]\right\}, \\
U(x) &= \{[-1,1] : x \in [-2,2]\}, \\
f(x,u) &= x + u.
\end{aligned}$$

The dynamics of F are the same as in Example 4.0.1. Recall that $K = [-2, 2]$ is not invariant under F. Now we will show that K is a controlled invariant set under (F, U, f). Note that

$$F(f(x,u)) = \left\{[-(x+u)^2, (x+u)^2] : x + u \in [-2,2]\right\}.$$

Obviously, for every $x \in [-2, 2]$ there exists a $u \in [-1, 1]$ such that $x + u \in [-1, 1]$, e.g., $u = -\frac{1}{2}x$. Hence, there exists a $u \in U(x)$ such that $F(f(x, u)) \in [-1, 1]$. Recall from Example 4.0.2 that $[-1, 1]$ is an invariant set under F. So for every $x \in [-2, 2]$ there exists a $u \in U(x)$ such that $F(f(x, u)) \subset [-2, 2]$. Meaning that $[-2, 2]$ is a controlled invariant set.

□

Similar to the invariant kernel, we can now define a controlled invariant kernel, as is done in [10].

Definition 4.0.6. *Consider an uncontrolled difference inclusion (F, U, f). Let $K \subseteq dom(U)$. The largest closed subset of K which is controlled invariant under (F, U, f) is called the controlled invariance kernel of K under (F, U, f). It is denoted as $CINV(K)$.*

Similar to the $INV(K)$, we have $CINV(K) \subseteq K$ and we know that $CINV(K)$ always exists, but it may be empty. Now we want to construct an algorithm, similar to the Invariant Kernel algorithm, to determine the controlled invariant kernel of some set K. But first we define local boundedness of a controlled difference invariance, based on [10].

Definition 4.0.7. *Consider an uncontrolled difference inclusion (F, U, f). It has the locally bounded property if either one of the following holds:*

1. *U is locally bounded.*
2. *(a) $\xi \in F(\xi)$ for any $\xi \in dom(F)$; and*

   (b) $|u| \to \infty$ implies $|f(x, u)| \to \infty$ for all $x \in \mathbb{R}^n$.

This means that when using an uncontrolled difference inclusion with the locally bounded property, we have one of the given cases. In the first one, we have a locally bounded U. Hence, for every value of $x \in dom(F)$ the set $U(x)$ is bounded. So there is a limited set of values that a controlled input u can assume, given the state vector x.

When the second statement holds, we have the same result, but this is due to different reasons. Note that the difference inclusion is used to describe the dynamics of a dynamical system in discrete time. We do not want a situation where for a bounded $x(k)$ the next step $x(k+1)$ could be infinitely large. We assume that $\xi \in F(\xi)$ for every $\xi \in dom(F)$. So for every x and u we have $f(x,u) \in F(f(x,u))$ if $f(x,u) \in dom(F)$. Then we know that there is only a bounded subset $I \subset U(x)$, such that $F(f(x,u))$ does not go to infinity, due to the fact that $|u| \to \infty$ implies $|f(x,u)| \to \infty$ for every $x \in \mathbb{R}^n$. Hence, there is a limited set of values that a controlled input u can assume, given the state vector x.

Now we have all the required components to construct the CIK algorithm from [10].

Proposition 4.0.8 (Controlled Invariance Kernel Algorithm)**.** *Let the uncontrolled difference inclusion (F,U,f) have the locally bounded property. Assume f is continuous, $dom(F)$ is closed, F is lower semicontinuous and U is upper semicontinuous and closed valued. Let $K \subseteq dom(U)$ be compact. Define the subsets K_j and $K_{j\frac{1}{2}}$ recursively by*

$$
\begin{aligned}
K_0 &= K, \\
K_{j\frac{1}{2}} &= \{\xi \in dom(F) : F(\xi) \subseteq K_j\}, \\
K_{j+1} &= \left\{x \in K_j : \exists u \in U(x) \text{ such that } f(x,u) \in K_{j\frac{1}{2}}\right\}.
\end{aligned}
$$

Then

$$CINV(K) = \bigcap_{j=0}^{\infty} K_j.$$

Proof. If $CINV(K) = \emptyset$, then there exists no subset of K that is a controlled invariant set. Hence, for every $\tilde{K} \subseteq K$ there exists some $x \in \tilde{K}$, such that for every $u \in U(x)$ we have $F(f(x,u)) \nsubseteq \tilde{K}$. Since K_j are nested sets, we know that there is a K_j, with j large enough, such that $K_j = \emptyset$. Hence, $\bigcap_{j=0}^{\infty} K_j = \emptyset$. Then the equality holds.

Now suppose $CINV(K)$ is non-empty. For every $x \in CINV(K)$ there exists a $u \in U(x)$ such that $F(f(x,u)) \subseteq CINV(K)$. By definition we have $CINV(K) \subseteq K_0$. Suppose that $CINV(K) \subseteq K_j$ for some j. Let $x \in K_j$. If $x \in CINV(K)$, then we already know we can find a control input $u \in U(x)$ such that $f(x,u) \subseteq CINV(K) \subseteq K_j$. Hence, we know $CINV(K) \subseteq K_{j+1}$. Since $CINV(K) \subseteq K_0$, we know $CINV(K) \subseteq \bigcap_{j=0}^{\infty} K_j$.

Before proving $CINV(K) \supseteq \bigcap_{j=0}^{\infty} K_j$, we show that $\bigcap_{j=0}^{\infty} K_j$ is closed. From the proof of Proposition 4.0.3, we already know that if K_j is closed, then $K_{j\frac{1}{2}}$ is closed. Now we show that if $K_{j\frac{1}{2}}$ is closed, then K_{j+1} is closed. Let $\{x_n\}_{n\in\mathbb{N}}$ be a sequence in K_{j+1} that converges to $x^* \in K_j$. Since we have $x_n \in K_{j+1}$, we know there exists some $u_n \in U(x_n)$ such that $f(x_n, u_n) \in K_{j\frac{1}{2}}$ for every $n \in \mathbb{N}$. Consider the set $R = \{u \in U(x) : x \in K_{j+1}\}$. By the locally bounded property, we can conclude that R is bounded.

Suppose 1 from Definition 4.0.7 holds. Then we know R is bounded, because $U(x)$ is bounded for any $x \in K_{j+1}$. Note that K_{j+1} is a bounded set, because K_j are nested sets and K_0 is compact.

Now suppose that 2 of Definition 4.0.7 holds. Because $x \in K_{j+1}$, we know that there must exist a $u \in U(x)$ such that $f(x,u) \in K_{j\frac{1}{2}}$. Since $K_{j\frac{1}{2}}$ is bounded, we know that there exists some $M \in \mathbb{R}$ such that $|f(x,u)| \leq M$ must be true. Therefore, we know that there must be an $N \in \mathbb{R}$ such that $|u| \leq N$. Hence, R is bounded because of the locally bounded property of (F, U, f).

Because R is bounded, we know that there exists a subsequence of u_n, denoted as u_{n_l}, which converges to u^*. Since U is closed valued and upper semicontinuous, we can prove that $u^* \in U(x^*)$. Upper semicontinuity shows that if $|x_{n_l} - x^*| < \delta$, then

$$\sup_{u\in U(x^*)} \inf_{u_{n_l}\in U(x_{n_l})} |u_{n_l} - u| < \varepsilon.$$

Hence, the set of $U(x^*)$ is relatively close to the set $U(x_{n_l})$, for n large enough. Because U is closed valued and the fact that u_{n_l} goes to u^*, we can conclude that $u^* \in U(x^*)$.

By continuity of f, we have $f(x_{n_l}, u_{n_l}) \to f(x^*, u^*)$. Because $K_{j\frac{1}{2}}$ is closed, we see that $f(x^*, u^*) \in K_{j\frac{1}{2}}$. This implies that $x^* \in K_{j+1}$. Hence, K_{j+1} is closed. And since K_0 is closed, we have proven by induction that $\bigcap_{j=0}^{\infty} K_j$ is closed.

The final piece of the puzzle is to prove that $CINV(K) \supseteq \bigcap_{j=0}^{\infty} K_j$. To do this, we show that $\bigcap_{j=0}^{\infty} K_j$ is a controlled invariant set. We define the sets $R_0(x) = U(x)$ and

$$R_{j+1}(x) = \left\{ u \in U(x) : f(x,u) \in K_{j\frac{1}{2}} \right\}.$$

$R_{j+1}(x)$ represents the set of controlled input values u such that $x \in K_{j+1}$. Note that we have already shown that $R_{j+1}(x)$ is bounded by the locally bounded property. By continuity of f and the fact that $K_{j\frac{1}{2}}$ is closed, we can conclude that $R_{j+1}(x)$ is closed. By the Heine-Borel theorem, we know $R_{j+1}(x)$ is compact.

Since K_j is not empty, we have $R_j(x)$ is not empty for every $x \in K_j$. Therefore, $\bigcap\limits_{j=0}^{\infty} R_j(x)$ is not empty for every $x \in \bigcap\limits_{j=0}^{\infty} K_j$. By definition we have $x \in \bigcap\limits_{j=0}^{\infty} K_j$ and $u \in \bigcap\limits_{j=0}^{\infty} R_j(x)$ implies $F(f(x,u)) \subseteq \bigcap\limits_{j=0}^{\infty} K_j$. So we can conclude that $\bigcap\limits_{j=0}^{\infty} K_j$ is a controlled invariant set. Since $CINV(K)$ is the largest controlled invariant set, we have $CINV(K) \supseteq \bigcap\limits_{j=0}^{\infty} K_j$. Hence,

$$CINV(K) = \bigcap_{j=0}^{\infty} K_j. \qquad \blacksquare$$

The CIK algorithm produces the largest controlled invariant subset. If $CINV(K)$ exists in finitely many steps, then we obtain some K_j such that $K_{j+1} = K_j$. The beauty of this algorithm is that we can determine some useful invariance properties, even though the $CINV(K)$ is empty, or if the nonempty $CINV(K)$ is computed in infinitely many steps. For any $x(k) \in K_j$, we know that there exists a controlled input such that the state remains within K for j time steps, i.e., $x(k+j) \in K$. As a result, we can determine a number of steps before the CIK algorithm is terminated. For the remainder of this paper, we consider the case where $CINV(K)$ is computed in finitely many steps.

If $CINV(K)$ is nonempty, then we know that for every state $x(k) \in CINV(K)$ there exists an input $u(k)$, such that the next state vector remains in the kernel. Unfortunately, the algorithm does not synthesize a controller. However, it is clear that memoryless nonlinear static feedback suffices, because the input is based on the current state vector only! We now continue with a specific notation such that we can determine a controller in the case of system (2.1)−(2.4).

Chapter 5

Existence of an Admissible Controller

Henceforth, we consider the system defined as in (2.1)−(2.4) with the following properties.

1. The pair (A, B_2) is controllable, as defined below.
2. There are no redundant control inputs, i.e., the column rank of B_2 equals m.

Definition 5.0.1. *We say that the pair* (A, B_2) *of a system defined by* (2.1) − (2.4) *is controllable if the controlled input has the ability to move the state of the system from any initial state to any other final state in a finite time interval, given* $w = 0$*.*

We can check whether (A, B_2) is controllable or not, using the following definition, based on [7].

Definition 5.0.2. *The system defined by* (2.1) − (2.4) *with* $w = 0$ *is controllable if and only if the matrix* $\mathcal{C} \in \mathbb{R}^{n\times nm}$*, defined by*

$$\mathcal{C} = \begin{pmatrix} B_2 & AB_2 & A^2B_2 & \dots & A^{n-1}B_2 \end{pmatrix},$$

has full rank, i.e., rank$(\mathcal{C}) = n$*.*

Note that our assumptions are less restrictive than those of Shamma in [10]. For a system defined as in (2.1)−(2.4), he requires the following.

1. B_1 and C have column rank n.
2. $D_2^T C = 0$ and $D_2^T D_2 > 0$.
3. B_2 has column rank m.

Both the first and second condition of Shamma are restrictive. Shamma uses them, because they offer considerable simplifications. However, they are not realistic for industrial applications, e.g., Chapter 9, and the are not necessary, as is shown in this thesis.

We want to apply the CIK algorithm to our system. Therefore, we need to create an uncontrolled difference inclusion that describes the dynamics of our system. We start by rewriting

$$x(k+1) = Ax(k) + B_1 w(k) + B_2 u(k).$$

It is stated earlier that the disturbance w is assumed to be bounded. We assume that w has a maximal amplitude of $\frac{1}{\gamma}$, i.e., $||w|| \leq \frac{1}{\gamma}$, for some $\gamma > 0$. Note that the exact values of any $w(k)$ are not known beforehand. Therefore, we can write a difference inclusion of the form

$$x(k+1) \in \left\{ Ax(k) + B_1 w + B_2 u(k) \in \mathbb{R}^n : |w| \leq \frac{1}{\gamma} \right\}$$

for any $k \in \mathbb{N}$. Note that we do not write $w(k)$ anymore. Since the maximal amplitude of the disturbance is known, we know all possible values of $w(k)$ for any $k \in \mathbb{N}$. Because these possible values do not depend on k, we simply write w, where $w \in \mathbb{R}^q$.

To write this in the form of an uncontrolled difference inclusion, we define $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $F_\gamma : \mathbb{R}^n \to \mathcal{P}(\mathbb{R}^n)$ by

$$f(x,u) = Ax + B_2 u, \tag{5.1}$$

$$F_\gamma(\xi) = \left\{ \xi + \frac{1}{\gamma} B_1 w \in \mathbb{R}^n : |w| \leq 1 \right\}. \tag{5.2}$$

Then we can write

$$x(k+1) \in F_\gamma(f(x(k), u(k))).$$

This is a way to rewrite our state equation (2.1). Now consider output equation (2.2). Suppose we want our regulated output z to be bounded by $||z|| \leq 1$. Note that $||w|| \leq \frac{1}{\gamma}$ still holds. So we have to choose an controlled input $u(k)$ such that

$$|z(k)| = |Cx(k) + D_1 w + D_2 u(k)| \leq 1, \quad \forall |w| \leq \frac{1}{\gamma}, \quad \forall k \in \mathbb{N}.$$

Therefore, we define

$$U_\gamma(x(k)) = \left\{ u \in \mathbb{R}^m : \left| Cx(k) + \frac{1}{\gamma} D_1 w + D_2 u \right| \leq 1, \quad \forall |w| \leq 1 \right\}. \tag{5.3}$$

Now we have the difference inclusion (F_γ, U_γ, f), which describes the dynamics of (2.1) – (2.4) for $||w|| \leq \frac{1}{\gamma}$ and $||z|| \leq 1$. But in order to determine if we can use the CIK algorithm, we need the following proposition.

Proposition 5.0.3. *The difference inclusion (F_γ, U_γ, f), as defined by* (5.1), (5.2) *and* (5.3)*, satisfies the conditions of Proposition* 4.0.8 *for every $\gamma > 0$ such that dom(U_γ) is nonempty.*

Proof. First we prove that f is continuous, using Definition 3.0.5. Suppose $\delta = \max\left\{\frac{\varepsilon}{2|A|}, \frac{\varepsilon}{2|B_2|}\right\}$ and choose $\begin{pmatrix} x_0 \\ u_0 \end{pmatrix}$ arbitrarily. Then $\forall \begin{pmatrix} x \\ u \end{pmatrix} \in \mathbb{R}^n \times \mathbb{R}^m$ such that $\left|\begin{pmatrix} x \\ u \end{pmatrix} - \begin{pmatrix} x_0 \\ u_0 \end{pmatrix}\right| < \delta$ we have

$$\begin{aligned} |f(x,u) - f(x_0,u_0)| &= |Ax + B_2u - Ax_0 - B_2u_0|, \\ &= |A(x - x_0) + B_2(u - u_0)|, \\ &\leq |A||x - x_0| + |B_2||u - u_0|, \\ &< (|A| + |B_2|)\delta, \\ &\leq \varepsilon. \end{aligned}$$

Since x_0 and u_0 were chosen arbitrarily, we know f is a continuous function.

Then we need *dom*(F_γ) closed. This holds because *dom*$(F_\gamma) = \mathbb{R}^n$ from any $\gamma > 0$ and $\mathbb{R}^n$ is a closed set.

Next we prove that F_γ is lower semicontinuous by using Definition 3.0.6. Choose an arbitrary $\xi \in \mathbb{R}^n$ and any $\eta \in F_\gamma(\xi)$. We know that there exists a vector $v \in \mathbb{R}^q$ such that $\xi + v = \eta$. Since $\eta \in F_\gamma(\xi)$, we know that $v = \frac{1}{\gamma}B_1\tilde{v}$, where $|\tilde{v}| \leq 1$. Let $\{x_i\}_{i\in\mathbb{N}}$ be a sequence in $\mathbb{R}^n$ such that $x_i \to \xi$. We define $y_i = x_i + \frac{i-1}{i}v$ for any $i > 0$. Obviously we have $y_i \to \xi + v = \eta$, so we only need to prove that $y_i \in F_\gamma(x_i)$ for every $i > 0$. We can rewrite y_i as $y_i = x_i + \frac{1}{\gamma}B_1\left(\frac{i-1}{i}\tilde{v}\right)$. Since we have $|\frac{i-1}{i}\tilde{v}| \leq |\tilde{v}| \leq 1$ for every $i > 0$, we know $y_i \in F_\gamma(\xi)$ for every $i > 0$. Hence, F_γ is lower semicontinuous.

To prove that (F_γ, U_γ, f) has the locally bounded property, we show that the second statement of Definition 4.0.7 holds for every $\gamma > 0$. We know we have *dom*$(F_\gamma) = \mathbb{R}^n$. It is obvious that $|w| \leq 1$ is satisfied for $w = 0$. So we know that $\xi \in F(\xi)$ $\forall\xi \in \mathbb{R}^n$. And because B_2 has column rank m, we know $|u| \to \infty$ implies $|B_2u| \to \infty$. Therefore $|u| \to \infty$ implies $|f(x,u)| \to \infty$ for all $x \in \mathbb{R}^n$. Hence, (F_γ, U_γ, f) has the locally bounded property for any $\gamma > 0$.

Now we need to proof that U_γ is closed valued. We do this by using Definition 3.0.9.
Let $x \notin$ *dom*(U_γ). By Definition 3.0.4 we know $U_\gamma(x) = \emptyset$. The empty set is closed.
Now let $x \in$ *dom*(U_γ). Then we know that there exists a $u \in \mathbb{R}^m$ such that

$$|Cx + \frac{1}{\gamma}D_1w + D_2u| \leq 1, \quad \forall|w| \leq 1.$$

If $D_2 = 0$, then $U_\gamma(x) = \mathbb{R}^m$, which is a closed set.
Assume $D_2 \neq 0$. By using a row-by-row analysis, we can rewrite the equation above to

$$\begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x + \frac{1}{\gamma} \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)} w + \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u \leq 1, \quad \forall |w| \leq 1, \quad \forall i = 1, \dots, 2q.$$

Because $|w|$ is the infinity norm of a vector $w \in \mathbb{R}^q$, we can say that $u \in U_\gamma(x)$ must satisfy

$$\begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x + \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)} \right| + \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u \leq 1, \quad \forall i = 1, \dots, 2q.$$

Since $x \in dom(U_\gamma)$ is fixed, we have the explicit expression

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u \leq 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x - \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)} \right|, \quad \forall i = 1, \dots, 2q. \tag{5.4}$$

Now suppose we have a sequence $\{u_n\}_{n \in \mathbb{N}} \subset U_\gamma(x)$ such that $u_n \to u^*$, where $u^* \notin U_\gamma(x)$. Then we know that there exists some k such that

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(k,:)} u^* > 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(k,:)} x - \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(k,:)} \right|.$$

Let us choose $\varepsilon > 0$ such that

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(k,:)} u^* = 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(k,:)} x - \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(k,:)} \right| + \varepsilon. \tag{5.5}$$

By convergence we have

$$u_n - u^* \to 0.$$

Then it should also hold that

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} (u_n - u^*) \to 0, \quad \forall i = 1, \dots, 2q.$$

Consider $i = k$. Then by combining (5.4) and (5.5) we have

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(k,:)} (u_n - u^*) \leq -\varepsilon, \quad \forall n \in \mathbb{N},$$

which is a contradiction. Hence, $u^* \in U_\gamma(x)$. Therefore $U_\gamma(x)$ is closed for every $x \in \mathbb{R}^n$, i.e. U_γ is closed valued.

Finally, we need to prove that U_γ is upper semicontinuous. We use Definition 3.0.7. In case $D_2 = 0$, we know

$$dom(U_\gamma) = \left\{ x \in \mathbb{R}^n : |Cx + \frac{1}{\gamma} D_1 w| \leq 1, \quad \forall |w| \leq 1 \right\}.$$

We have assumed γ is large enough such that $dom(U_\gamma)$ is not empty. Just as above, we consider a row-by-row analysis to conclude the explicit expression

$$\begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x \leq 1 - \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)} \right|, \quad \forall i = 1, \ldots, 2q.$$

With a similar argument as above, we can conclude that $dom(U_\gamma)$ is a closed set. By using the fact that $U_\gamma(x) = \mathbb{R}^m$ for any $x \in dom(U_\gamma)$ in this case, we can conclude that

$$\sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u' - u| = 0,$$

for every $x, x' \in dom(U_\gamma)$. Hence, U_γ is upper semicontinuous when $D_2 = 0$.

Now suppose $D_2 \neq 0$. Now we have

$$dom(U_\gamma) = \left\{ x \in \mathbb{R}^n : \exists u \in U_\gamma(x) \text{ such that } |Cx + \frac{1}{\gamma} D_1 w + D_2 u| \leq 1, \quad \forall |w| \leq 1 \right\}.$$

We know $dom(U_\gamma)$ is not empty by assumption. We consider a row-by-row analysis to obtain

$$\begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x + \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u \leq 1 - \frac{1}{\gamma} \left| \begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)} \right|, \quad \forall i = 1, \ldots, 2q.$$

Reconsider our function $f(x, u)$, defined as in (5.1). We have proved this function is continuous. By replacing A with $\begin{pmatrix} C \\ -C \end{pmatrix}$ and B_2 with $\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}$, we can conclude that the left-hand side of the inequality above is continuous. Then we can conclude that $dom(U_\gamma)$ is closed by using similar arguments as when we proved $U_\gamma(x)$ is closed for every $x \in dom(U_\gamma)$.

Before proving the rest of the statement, we need the following statement. Consider some $u \in \mathbb{R}^m$. We use an orthogonal decomposition $u = u_{\mathcal{R}} + u_{\mathcal{N}}$, where $u_{\mathcal{N}}$ is in the null space of D_2, i.e., $D_2 u_{\mathcal{N}} = 0$. Then we know that there exists some $\alpha > 0$ such that $|D_2 u| \geq \alpha |u_{\mathcal{R}}|$. [6]

Now choose any $x \in dom(U_\gamma)$ and fix $\varepsilon > 0$. We want to find a $\delta > 0$ such that for any $x' \in dom(U_\gamma)$, where $|x' - x| < \delta$, we have

$$\sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u' - u| < \varepsilon.$$

Let $\delta = \frac{|C|}{\alpha}\varepsilon$. For any x' such that $U_\gamma(x') \subseteq U_\gamma(x)$, we know that

$$\sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u' - u| = 0.$$

So suppose $u' \in U_\gamma(x')$ and $u' \notin U_\gamma(x)$. Then u and u' must satisfy the following inequalities

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u \leq 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x - \frac{1}{\gamma}\left|\begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)}\right|, \quad \forall i = 1, \dots, 2q, \tag{5.6}$$

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u' \leq 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(i,:)} x' - \frac{1}{\gamma}\left|\begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(i,:)}\right|, \quad \forall i = 1, \dots, 2q, \tag{5.7}$$

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(k,:)} u' > 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(k,:)} x - \frac{1}{\gamma}\left|\begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(k,:)}\right| \text{ for some } k \in \{1, \dots, 2q\}. \tag{5.8}$$

To use these inequalities, we first use the orthogonal decomposition as described above. Since the null space of a matrix is closed, we have

$$\begin{aligned}
\sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u' - u| &= \sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u'_{\mathcal{R}} + u'_{\mathcal{N}} - u_{\mathcal{R}} - u_{\mathcal{N}}|, \\
&= \sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |u'_{\mathcal{R}} - u_{\mathcal{R}}|, \\
&\leq \sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} \frac{1}{\alpha}|D_2(u' - u)|, \\
&= \frac{1}{\alpha} \sup_{u' \in U_\gamma(x')} \inf_{u \in U_\gamma(x)} |D_2 u' - D_2 u|.
\end{aligned}$$

By definition we have

$$|D_2 u' - D_2 u| = \max_{i=1,\dots,2q} \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u' - \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u.$$

We can see that if we want to obtain the infimum of this, we need u to be as close as possible to u'. As said before, if $u' \in U_\gamma(x)$, then we simply choose $u = u'$ and we have $|D_2 u' - D_2 u| = 0$. For any $u' \notin U_\gamma(x)$, we obtain the infimum if $u \in \partial U_\gamma(x)$. Hence, there is some $j \in \{1, \dots, 2q\}$ such that

$$\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(j,:)} u = 1 - \begin{pmatrix} C \\ -C \end{pmatrix}_{(j,:)} x - \frac{1}{\gamma}\left|\begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(j,:)}\right|.$$

Since the maximum of $\begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u' - \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u$ is found in the row where (5.8) holds, take $j = k$ to obtain

$$\inf_{u \in U_\gamma(x)} \max_{i=1,\dots,2q} \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u' - \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(i,:)} u = \begin{pmatrix} D_2 \\ -D_2 \end{pmatrix}_{(k,:)} u' - 1 + \begin{pmatrix} C \\ -C \end{pmatrix}_{(k,:)} x + \frac{1}{\gamma}\left|\begin{pmatrix} D_1 \\ -D_1 \end{pmatrix}_{(k,:)}\right|.$$

By using (5.7) we get

$$\begin{aligned}\sup_{u'\in U_\gamma(x')}\inf_{u\in U_\gamma(x)}|D_2u'-D_2u| &\leq \begin{pmatrix} C \\ -C\end{pmatrix}_{(k,:)}(x-x'),\\ &\leq |C|\delta.\end{aligned}$$

Hence, for any $x'\in dom(U_\gamma)$ such that $|x'-x|<\delta$, we have

$$\sup_{u'\in U_\gamma(x')}\inf_{u\in U_\gamma(x)}|u'-u| < \frac{1}{\alpha}|C|\delta = \varepsilon.$$

So U_γ is upper semicontinuous. ■

Based on Proposition 5.0.3, we are allowed to construct a CIK of system (2.1)−(2.4) for $||w||\leq\frac{1}{\gamma}$ and $||z||\leq 1$. If $dom(U_\gamma)$ is closed and bounded, we can choose to construct $CINV(dom(U_\gamma))$. However, this is not necessarily the case, since we do not have the restriction that C is full rank. Therefore we define the following set.

Definition 5.0.4. *Let $K\subset\mathbb{R}^n$ be defined as*

$$K := \begin{cases} dom(U_\gamma)\cap\{x\in\mathbb{R}^n : |x|\leq L\}, & \textit{if } dom(U_\gamma) \textit{ is not bounded,}\\ dom(U_\gamma), & \textit{if } dom(U_\gamma) \textit{ is bounded.}\end{cases}$$

for $L\in\mathbb{R}$ sufficiently large.

For the remainder of this paper, we assume that K is defined according to Definition 5.0.4. How large L must be chosen is discussed later.

Now we can construct $CINV(K)$, regardless of the boundedness of $dom(U_\gamma)$. We want to show the following. If $CINV(K)$ exists and is nonempty, then there exists a nonlinear static feedback controller $\mathcal{K}$ such that $||\mathcal{S}_\mathcal{K}||\leq\gamma$. In order to prove this, we need the following lemma.

Lemma 5.0.5. *If $CINV(K)$ is nonempty, then it has the following properties.*

1. *It is symmetric with respect to the origin.*
2. *It is convex.*
3. *The origin is an interior point.*

Proof. We start by checking if $dom(U_\gamma)$ has all three properties. We have assumed that γ is large enough such that $\frac{1}{\gamma}|D_1w|<1$ for every $|w|\leq 1$, since $CINV(K)$ exists. Suppose $x\in dom(U_\gamma)$. Then there exists a $u\in U_\gamma(x)$ such that $|Cx+\frac{1}{\gamma}D_1w+D_2u|\leq 1$ for every $|w|\leq 1$. We have

$$U(-x)=\left\{u\in\mathbb{R}^m : |-Cx+\frac{1}{\gamma}D_1w+D_2u|\leq 1,\quad \forall|w|\leq 1\right\}.$$

By symmetry of $|w|$, we have the following equality.

$$\left\{|-Cx+\frac{1}{\gamma}D_1w+D_2u|\leq 1 : |w|\leq 1\right\} = \left\{|Cx+\frac{1}{\gamma}D_1w+D_2(-u)|\leq 1 : |w|\leq 1\right\}.$$

Hence, $-u \in U(-x)$. Then we can conclude that $-x \in dom(U_\gamma)$. Hence, $dom(U_\gamma)$ is symmetric.

Convexity of $dom(U_\gamma)$ is easily proven. Take $x, y \in dom(U_\gamma)$ and $\lambda \in (0,1)$. Let a sufficient controlled input for x be denoted as u_x and for y be denoted as u_y. Then you can show that $\lambda x + (1-\lambda)y \in dom(U_\gamma)$ by using controlled input $\lambda u_x + (1-\lambda)u_y$ and applying the triangle inequality.

To prove the origin is an interior point of $dom(U_\gamma)$, we state that γ is fixed such that $\frac{1}{\gamma}|D_1w| = 1-\varepsilon$, $\forall |w| \leq 1$, where $\epsilon > 0$. If this is not true, then $dom(U_\gamma)$ has no interior at all. Then we simply choose controlled input $u = 0$. By applying the triangle inequality, we have

$$\begin{aligned}\left\{x\in\mathbb{R}^n : |Cx+\frac{1}{\gamma}D_1w|\leq 1, \quad \forall|w|\leq 1\right\} &\supseteq \left\{x\in\mathbb{R}^n : |Cx|+\frac{1}{\gamma}|D_1w|\leq 1, \quad \forall|w|\leq 1\right\}, \\ &\supseteq \{x\in\mathbb{R}^n : |Cx|\leq\varepsilon\}.\end{aligned}$$

Since ε is fixed and greater than 0, we know 0 is an interior point of $dom(U_\gamma)$.

Now that we know $dom(U_\gamma)$ has all the properties of Lemma 5.0.5, we need to show that K has the same properties in the case that $dom(U_\gamma)$ is unbounded. Note that the set

$$\{x\in\mathbb{R}^n : |x|\leq L \text{ for } L\in\mathbb{R} \text{ sufficiently large}\}$$

satisfies all three properties. Therefore, K has symmetry with respect to the origin, because both sets are symmetric with respect to the origin. Because the intersection of two convex sets is convex, K has the second property as well. And since K is the intersection of two sets with the origin as an interior point, we can conclude that K has the origin as an interior point as well.

By an induction argument, we show that $CINV(K)$ is symmetric with respect to the origin. Let K_j and $K_{j\frac{1}{2}}$ be defined as in the Controlled Invariant Kernel algorithm. Suppose K_j is symmetric with respect to the origin. Because $F_\gamma(\xi)$ is a symmetric set with respect ξ for every $\xi \in \mathbb{R}^n$, we know that $K_{j\frac{1}{2}}$ is symmetric with respect to the origin. Suppose $x \in K_{j+1}$. Then we know that $f(x,u) \in K_{j\frac{1}{2}}$ for some $u \in U(x)$. We have shown already that $-u \in U(-x)$. And we know that $-f(x,u) \in K_{j\frac{1}{2}}$. The way we defined f shows us that $-f(x,u) = f(-x,-u)$. Then we can conclude that $f(-x,-u) \in K_{j\frac{1}{2}}$ for $-u \in U(-x)$, i.e., $-x \in K_{j+1}$. Hence, K_{j+1} is symmetric with respect to the origin. By induction we know that $CINV(K)$ is symmetric with respect to the origin.

Suppose K_j is a convex set. Since $F_\gamma(\xi)$ is a convex set for every $\xi \in \mathbb{R}^n$, we can deduce that $K_{j\frac{1}{2}}$ is a convex set as well. Now suppose $x, y \in K_{j+1}$. Let $u \in U(x)$ such that $f(x,u) \in K_{j\frac{1}{2}}$ and $v \in U(y)$ such that $f(y,v) \in K_{j\frac{1}{2}}$. Let $\lambda \in (0,1)$. By triangle inequality it is easy to show that $\lambda u + (1-\lambda)v \in U(\lambda x + (1-\lambda)y)$. By convexity of $K_{j\frac{1}{2}}$ we know that $\lambda f(x,u) + (1-\lambda)f(y,v) \in K_{j\frac{1}{2}}$. By our definition of f we have

$$\lambda f(x,u) + (1-\lambda)f(y,v) = f(\lambda x + (1-\lambda)y, \lambda u + (1-\lambda)v).$$

So we can conclude that $\lambda x + (1-\lambda)y \in K_{j+1}$. Therefore K_{j+1} is a convex set. Since the intersection of nested convex sets is a convex set, we know $CINV(K)$ is convex.

The only thing left is to prove that the origin is an interior point of $CINV(K)$. First we show that $0 \in CINV(K)$. We do this by using the symmetry with respect to the origin and the convexity of the set. For any $x \in CINV(K)$ we know $-x \in CINV(K)$. Then we also know $0 = \frac{1}{2}x + \frac{1}{2}(-x) \in CINV(K)$. Now consider a $u \in U_\gamma(0)$. Then $F_\gamma(B_2u) \subseteq CINV(K)$. Because of the symmetry with respect to the origin, we know $F_\gamma(-B_2u) = -F_\gamma(B_2u) \subseteq CINV(K)$. If B_1 has rank n, then both sets have a nonempty interior. By convexity we have $\frac{1}{2}F_\gamma(B_2u) + \frac{1}{2}F_\gamma(-B_2u) \subseteq CINV(K)$. Hence, the origin is an interior point. Now suppose B_1 has a rank lower than n. Then we have a direction which is not affected by the disturbance. Due to the fact that we have a controllable system, we know that this direction remains untouched through the whole process of K_j. Hence, we have the same bound as the original $dom(U_\gamma)$. For $dom(U_\gamma)$ we determined that the origin is an interior point already. Combining this with the argument of $\frac{1}{2}F_\gamma(B_2u) + \frac{1}{2}F_\gamma(-B_2u) \subseteq CINV(K)$, we can conclude that the origin is an interior point of $CINV(K)$ regardless of the rank of B_1. ■

Lemma 5.0.5 basically says that a nonempty controlled invariant kernel of a set based on system $\mathcal{S}$ is symmetric, convex and centered at the origin. These properties are very useful for multiple reasons, which we will discuss later.

Theorem 5.0.1. *Let $CINV(K)$ be nonempty for a given $\gamma > 0$. Then there exists an internally stabilizing controller $\mathcal{K}$ with a performance of γ of the form described in the second case of* Definition 2.2.1.

Proof. Define the set

$$K_{\infty\frac{1}{2}} = \left\{ \xi : \xi + \frac{1}{\gamma}B_1 w \in CINV(K), \quad \forall |w| \leq 1 \right\}.$$

And let the set-valued map $R : CINV(K) \to \mathcal{P}(\mathbb{R}^m)$ be defined as

$$R(x) = \left\{ u \in U_\gamma(x) : Ax + B_2 u \in K_{\infty\frac{1}{2}} \right\}.$$

$K_{\infty\frac{1}{2}}$ denotes the set where $f(x, u)$ should be in, because then we have the assurance that the state vector remains in the controlled invariant kernel. The set $R(x)$ is characterized by the set of admissible control values. Since $CINV(K)$ is a compact set, we can see that $K_{\infty\frac{1}{2}}$ is also compact. From the proof of Proposition 4.0.8, we see that a similarly defined set $R_{j+1}(x)$ is compact for every $x \in K_j$. By using the same arguments, we determine that $R(x)$ is compact for every $x \in CINV(K)$.

We know $CINV(K)$ is convex. Due to the definition of $|w|$, this implies $K_{\infty\frac{1}{2}}$ is also a convex set. For any fixed $x \in CINV(K)$, we can conclude that $R(x)$ must also be a convex set.

We want to prove that R is a lower semicontinuous map. We choose some $x \in CINV(K)$. Let $u \in R(x)$. Suppose we have a sequence $\{x_n\}_{n\in\mathbb{N}} \subseteq CINV(K)$ such that $x_n \to x$. Now we need to construct a sequence $\{u_n\}_{n\in\mathbb{N}}$ such that $u_n \in R(x_n)$ for every $n \in \mathbb{N}$ and $u_n \to u$. Note that if $u \in R(x_n)$ for every $n > n_0$, where n_0 is large enough, then we simply choose a sequence such that $u_n = u$ for every $n > n_0$. So suppose $u \notin R(x_n)$ for every $n > n_0$ for some $n_0 \in \mathbb{N}$. Let $\{u_n\}_{n\in\mathbb{N}}$ be a sequence such that $u_n \in R(x_n)$ for every $n \in \mathbb{N}$ with $Ax_n + B_2u_n \in \partial K_{\infty\frac{1}{2}}$. Here ∂K denotes the boundary of the set K. Because $K_{\infty\frac{1}{2}}$ is closed, we have $\partial K_{\infty\frac{1}{2}} \subset K_{\infty\frac{1}{2}}$. By continuity of f and the fact that $K_{\infty\frac{1}{2}}$ is closed, we know that $f(x_n, u_n) \to f(x, u)$ and $f(x, u) \in K_{\infty\frac{1}{2}}$. Therefore, we can conclude that R is lower semicontinuous.

By Theorem 10.2.1 in Appendix A, we now know that there exists a continuous selection function denoted as $g : \mathbb{R}^n \to \mathbb{R}^m$. How we determine this continuous selection is discussed in the next chapter. We know that g ensures that $CINV(K)$ is invariant under the difference inclusion

$$x(k+1) \in \left\{ Ax(k) + \frac{1}{\gamma}B_1w + B_2g(x(k)) : |w| \leq 1 \right\}.$$

Since $CINV(K)$ is a subset of the set for which $||z|| \leq 1$ for every disturbance that satisfies $||w|| \leq \frac{1}{\gamma}$, we know that this controller has a performance of γ. Note that $g : \mathbb{R}^n \to \mathbb{R}^m$ is a continuous function that can be used as an internally stabilizing controller with a performance of γ, but it does not necessarily have the property $g(0) = 0$. However, we see that $0 \in R(0)$. In a construction of a continuous selection function $g : \mathbb{R}^n \to \mathbb{R}^m$, we see that there exists a selection with $g(0) = 0$. How we construct this specific selection is discussed in the next chapter. ■

Recall that our goal is to find a controller with performance that is desirably close to $\gamma_{optimal}$. Based on these results, we can redefine $\gamma_{optimal}$ as

$$\gamma_{optimal} = \inf \{\gamma : CINV(K) \text{ is nonempty}\}.$$

Note that the existence of $CINV(K)$ depends on the system dynamics only and not on any particular controller.

By the use of the CIK algorithm we can approximate $\gamma_{optimal}$ in the following way.

1. Initialize $\gamma > 0$ and $K_0 = K$.
2. Execute the Controlled Invariant Kernel algorithm to compute K_j and $K_{j\frac{1}{2}}$.
   (a) If any K_j is empty, then $CINV(K)$ is empty. Increase γ and restart.
   (b) If $CINV(K)$ is nonempty, then execute one of the following:
      i. Decrease γ and restart.
      ii. Construct an internally stabilizing controller $\mathcal{K}$ which achieves a performance γ.

There are two problems with the presented procedure. The first is that we do not have an explicit way to construct a controller $\mathcal{K}$. The second is that we have no way to determine when to construct the controller or to decrease γ. We start with a way to construct the controller $\mathcal{K}$.

Chapter 6

Implementation of the Controlled Invariant Kernel Algorithm

In order to implement our algorithm in Matlab, we need to define some new concepts. Using these concepts, we can construct the CIK algorithm for 4 separate cases. This way it is easier to show how the implementation is executed. We start by providing an explicit implementation of the CIK algorithm in the case where $D_1 = 0$, $D_2 = 0$, $u \in \mathbb{R}$ and $dom(U_\gamma)$ is bounded. Followed by the case where $D_1 \neq 0$ and/or $D_2 \neq 0$. Then we consider $u \in \mathbb{R}^m$, with $m > 1$. And finally, we show the algorithm in the case that $dom(U_\gamma)$ is not bounded.

But we start with introducing a new concept. From Lemma 5.0.5 we know that K_j is a convex set for every $j \in \mathbb{N}$, if it is nonempty. The way we have defined our K_j enables us to view the set as the intersection of halfspaces [3]. In mathematical terms, we can create a corresponding matrix $M_j \in \mathbb{R}^{l \times n}$ and a corresponding vector $b_j \in \mathbb{R}^l$ such that

$$K_j = \{x \in \mathbb{R}^n : M_j x \leq b_j\}, \quad \forall j \in \mathbb{N}.$$

Here l is the number of halfspaces that is needed to construct K_j. Since the origin is an interior point of a nonempty set K_j, as proven in Lemma 5.0.5, we can use a scaling argument such that all inequalities can be written in the form $(M_j)_{(i,:)} x \leq 1$, for every $i = \{1, 2, \dots, l\}$. Now we can define the following.

Definition 6.0.1. *Let* $\mathbf{1} \in \mathbb{R}^n$ *denote the n-dimensional column vector, where every component equals* 1.

Definition 6.0.2. *Let* $M \in \mathbb{R}^{l \times n}$. *We define* $Set(M) \subseteq \mathbb{R}^n$ *such that*

$$Set(M) = \{x \in \mathbb{R}^n : Mx \leq \mathbf{1}\},$$

where $\mathbf{1} \in \mathbb{R}^l$.

Using these definitions, we construct the first algorithm.

6.1 Algorithm Case 1

In case 1 we have $u \in \mathbb{R}$ and assume $D_1 = D_2 = 0$, i.e., we have a single input and the output is not dependent on the controlled input or the disturbances. Then our initial set is

$$K_0 = dom(U_\gamma) = \{x \in \mathbb{R}^n : |Cx| \leq 1, \quad \forall |w| \leq 1\}.$$

It is fairly easy to see that we can rewrite K_0 to

$$K_0 = \left\{ x \in \mathbb{R}^n : \begin{pmatrix} C \\ -C \end{pmatrix} x \leq \mathbf{1} \right\}.$$

Hence, by choosing $M_0 = \begin{pmatrix} C \\ -C \end{pmatrix}$ we have $K_0 = Set(M_0)$.

The next step in the algorithm is to determine the set $K_{\frac{1}{2}}$, defined as

$$K_{\frac{1}{2}} = \{\xi \in dom(F_\gamma) : F_\gamma(\xi) \subseteq K_0\}.$$

Using (5.2) and our new notation for K_0, we can rewrite this to

$$K_{\frac{1}{2}} = \left\{ \xi \in \mathbb{R}^n : \xi + \frac{1}{\gamma} B_1 w \in Set(M_0), \quad \forall |w| \leq 1 \right\}.$$

We want to construct a matrix $M_{\frac{1}{2}} \in \mathbb{R}^{2q \times n}$ such that $K_{\frac{1}{2}} = Set(M_{\frac{1}{2}})$. Note that the size of M_0 is determined already and the size of $M_{\frac{1}{2}}$ is the same. In order to obtain the appropriate matrix, we look at the constraints of the definition of $K_{\frac{1}{2}}$.

$$\xi + \frac{1}{\gamma} B_1 w \in Set(M_0), \quad \forall |w| \leq 1.$$

We can rewrite this as

$$M_0 \left(\xi + \frac{1}{\gamma} B_1 w \right) \leq \mathbf{1}, \quad \forall |w| \leq 1.$$

We consider this inequality row by row. For any row we obtain

$$(M_0)_{(i,:)} \xi + \frac{1}{\gamma} (M_0 B_1)_{(i,:)} w \leq 1, \quad \forall |w| \leq 1.$$

Since these inequalities must hold for every w that satisfies $|w| \leq 1$, we only consider the worst case for this inequality, which also needs to be satisfied. Hence, we know that ξ must satisfy

$$(M_0)_{(i,:)} \xi + \frac{1}{\gamma} \left| (M_0 B_1)_{(i,:)} \right| \leq 1. \tag{6.1}$$

Since for every nonempty $K_{j\frac{1}{2}}$ we have symmetry with respect to the origin and know that the origin is an interior point, we can conclude that we need the following restriction.

$$\frac{1}{\gamma}\left|(M_0B_1)_{(i,:)}\right| < 1.$$

Since this needs to hold for every row, we get

$$\frac{1}{\gamma}\left|M_0B_1\right| < 1.$$

If this restriction does not hold, then γ needs to be increased. Now suppose that it holds. Then we can rewrite the inequality in (6.1) as

$$\frac{1}{1-\frac{1}{\gamma}\left|(M_0B_1)_{(i,:)}\right|}\,(M_0)_{(i,:)}\,\xi \leq 1.$$

By defining the matrix $M_{\frac{1}{2}}$ for every row as

$$\left(M_{\frac{1}{2}}\right)_{(i,:)} = \frac{1}{1-\frac{1}{\gamma}\left|(M_0B_1)_{(i,:)}\right|}\,(M_0)_{(i,:)}\,,$$

we can write our set as $K_{\frac{1}{2}} = Set(M_{\frac{1}{2}})$. The next step is to determine a matrix $M_1 \in \mathbb{R}^{l\times n}$ such that $K_1 = Set(M_1)$, where l denotes the number of constraints needed to represent K_1. Recall that we have

$$K_1 = \left\{x \in K_0 : \exists u \in U_\gamma(x) \text{ such that } f(x,u) \in K_{\frac{1}{2}}\right\}.$$

Note that $U_\gamma(x) = \mathbb{R}$ for every $x \in dom(U_\gamma)$, since $D_2 = 0$. With the use of (5.1) and our new notation of K_0 and $K_{\frac{1}{2}}$, we can rewrite the set as

$$K_1 = \left\{x \in Set(M_0) : \exists u \in \mathbb{R} \text{ such that } Ax + B_2u \in Set(M_{\frac{1}{2}})\right\}.$$

In other words, we want to find the values of $x \in \mathbb{R}^n$ that satisfy

$$M_0x \leq \mathbf{1}, \text{ and} \tag{6.2}$$
$$M_{\frac{1}{2}}\left(Ax + B_2u\right) \leq \mathbf{1}, \tag{6.3}$$

for some $u \in \mathbb{R}$. In order to determine the values of $x \in \mathbb{R}^n$ such that there exists a $u \in \mathbb{R}$ for which (6.3) holds, we need to rewrite the statement to an appropriate form, which is

$$\begin{pmatrix} M_{\frac{1}{2}}A & M_{\frac{1}{2}}B_2 \end{pmatrix}\begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}.$$

With our notation we can define the set of combinations of x and u that satisfies this inequality as

$$Set\left(M_{\frac{1}{2}}A \quad M_{\frac{1}{2}}B_2\right) = \left\{\begin{pmatrix} x \\ u \end{pmatrix} \in \mathbb{R}^{n+1} : \left(M_{\frac{1}{2}}A \quad M_{\frac{1}{2}}B_2\right)\begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}\right\}.$$

This set contains all the combinations of $x \in \mathbb{R}^n$ and $u \in \mathbb{R}$ such that the second inequality holds. But we are only interested in the values of $x \in \mathbb{R}^n$ for which there exists some $u \in \mathbb{R}$. In order to determine these values of x, we need the following definition, which is based on [10].

Definition 6.1.1. *Let $M \in \mathbb{R}^{l\times(n+1)}$. Define $Rack(M)$ as the set of matrices, $\tilde{M}$, such that*

$$x \in Set(\tilde{M}) \subseteq \mathbb{R}^n \Leftrightarrow \exists u \in \mathbb{R} \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set(M) \subseteq \mathbb{R}^{n+1}. \tag{6.4}$$

We apply $Rack(M)$ to find the projection of $Set(M) \subseteq \mathbb{R}^{n+1}$ to $Set(\tilde{M}) \subseteq \mathbb{R}^n$, where $\tilde{M} \in Rack(M)$. Note that we say $\tilde{M} \in Rack(M)$ because there exists more than one matrix to describe the projection. Although the constraints of the projection are fixed, the order of the rows of any matrix $\tilde{M} \in Rack(M)$ can alter, i.e., any $Set(\tilde{M})$ is the same set for any $\tilde{M} \in Rack(M)$. In order for us to construct a matrix $\tilde{M} \in Rack(M)$, we need the following proposition, from [10].

Proposition 6.1.2. *Let $M = \begin{pmatrix} M_1 & M_2 \end{pmatrix}$, where $M_1 \in \mathbb{R}^{l\times n}$ and $M_2 \in \mathbb{R}^l$. Let*

$$\begin{aligned} Z_+ &= \{i : (M_2)_i > 0\}, \\ Z_- &= \{i : (M_2)_i < 0\}, \\ Z_0 &= \{i : (M_2)_i = 0\}. \end{aligned}$$

Let the rows of the matrix $\tilde{M}$ be formed by

$$\begin{aligned} \rho_{i_+,i_-} &= \frac{1}{(M_2)_{i_+} - (M_2)_{i_-}}\left((M_2)_{i_+}(M_1)_{(i_-,:)} - (M_2)_{i_-}(M_1)_{(i_+,:)}\right), \quad \forall i_+ \in Z_+,\ \forall i_- \in Z_-, \\ \rho_{i_0} &= (M_1)_{(i_0,:)}, \quad \forall i_0 \in Z_0. \end{aligned}$$

Then $\tilde{M}$ satisfies (6.4)*, i.e., $\tilde{M} \in Rack(M)$.*

Proof. In order to prove that $\tilde{M} \in Rack(M)$, we show that $\tilde{M}$ satisfies (6.4).

First we show (6.4) '$\Rightarrow$': Since $\tilde{M} \in Rack(M)$, we know how the rows of $\tilde{M}$ are constructed. For any $x \in Set(\tilde{M})$ we have $\tilde{M}x \leq \mathbf{1}$. We consider that inequality row by row. The rows of $\tilde{M}$ are denoted as ρ_{i_+,i_-} and ρ_{i_0}, where $i_+ \in Z_+$, $i_- \in Z_-$ and $i_0 \in Z_0$. If $Z_0 = \emptyset$, then there does not exist a row of $\tilde{M}$ of the form ρ_{i_0}. If either Z_+ or Z_- is an empty set, then there does not exist a row of $\tilde{M}$ of the form ρ_{i_+,i_-}.

For every $x \in Set(\tilde{M})$ we want to determine a $u \in \mathbb{R}$ such that $M_{(i,:)}\begin{pmatrix} x \\ u \end{pmatrix} \leq 1$ for every $i = 1, \dots, l$. To prove this, we could consider all cases of the emptyness or nonemptyness of Z_+, Z_- and Z_0 separately. Fortunately, we do not need to check all the cases. For instance, it is not possible that all three sets are empty. Assume only Z_0 is nonempty, i.e., there is no constraint on u. By definition we have

$$\begin{aligned} M_{(i,:)}\begin{pmatrix} x \\ u \end{pmatrix} &= (M_1)_{(i,:)}x, \\ &= \rho_i x \leq 1. \end{aligned}$$

In this case every $u \in \mathbb{R}$ suffices, since there are no restrictions on it. Hence, a row of the form ρ_{i_0} does not apply any restriction on u for every $i_0 \in Z_0$. Therefore we assume Z_0 to be empty for the remainder of the proof.

This leaves us with three more cases. First assume that both Z_+ and Z_- are nonempty. Then every row of $\tilde{M}$ is of the form ρ_{i_+,i_-}, where $i_+ \in Z_+$ and $i_- \in Z_-$. We have

$$1 \geq \rho_{i_+,i_-}x = \frac{1}{(M_2)_{i_+} - (M_2)_{i_-}}\left((M_2)_{i_+}(M_1)_{(i_-,:)} - (M_2)_{i_-}(M_1)_{(i_+,:)}\right)x, \tag{6.5}$$

for any $i_+ \in Z_+$ and $i_- \in Z_-$. We choose

$$u = \min_{i_+ \in Z_+} \frac{1 - (M_1)_{(i_+,:)}x}{(M_2)_{i_+}}. \tag{6.6}$$

We need to show that this choice of u satisfies $M_{(i,:)}\begin{pmatrix} x \\ u \end{pmatrix} \leq 1$ for every $i = 1, \dots, l$. We know that i is an element of Z_+ or Z_-. For $i \in Z_+$ we have

$$\begin{aligned} M_{(i,:)}\begin{pmatrix} x \\ u \end{pmatrix} &= (M_1)_{(i,:)}x + (M_2)_i u, \\ &= (M_1)_{(i,:)}x + (M_2)_i\left(\min_{i_+ \in Z_+} \frac{1 - (M_1)_{(i_+,:)}x}{(M_2)_{i_+}}\right), \\ &\leq (M_1)_{(i,:)}x + (M_2)_i\left(\frac{1 - (M_1)_{(i,:)}x}{(M_2)_i}\right) = 1. \end{aligned}$$

Our condition is satisfied. Now suppose $i \in Z_-$. For convenience, we say that

$$\min_{i_+ \in Z_+} \frac{1 - (M_1)_{(i_+,:)}x}{(M_2)_{i_+}} = \frac{1 - (M_1)_{(j,:)}x}{(M_2)_j}.$$

Because $i \in Z_-$ and $j \in Z_+$, we can apply (6.5), resulting in

$$\begin{aligned}
M_{(i,:)} \begin{pmatrix} x \\ u \end{pmatrix} &= (M_1)_{(i,:)}x + (M_2)_i u, \\
&= (M_1)_{(i,:)}x + (M_2)_i \left(\min_{i_+ \in Z_+} \frac{1-(M_1)_{(i_+,:)}x}{(M_2)_{i_+}} \right), \\
&= (M_1)_{(i,:)}x + (M_2)_i \left(\frac{1-(M_1)_{(j,:)}x}{(M_2)_j} \right), \\
&= \frac{(M_2)_j}{(M_2)_j} \left((M_1)_{(i,:)}x + (M_2)_i \left(\frac{1-(M_1)_{(j,:)}x}{(M_2)_j} \right) \right), \\
&= \frac{1}{(M_2)_j} \left((M_2)_j (M_1)_{(i,:)}x + (M_2)_i - (M_2)_i (M_1)_{(j,:)}x \right), \\
&= \frac{1}{(M_2)_j} \left((M_2)_i + ((M_2)_j (M_1)_{(i,:)} - (M_2)_i (M_1)_{(j,:)})x \right), \\
&= \frac{1}{(M_2)_j} \left((M_2)_i + ((M_2)_j - (M_2)_i)\rho_{j,i}x \right), \\
&\leq \frac{1}{(M_2)_j} \left((M_2)_i + (M_2)_j - (M_2)_i \right) = 1.
\end{aligned}$$

Hence, we have found a $u \in \mathbb{R}$ such that $M \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}$ in this case.

However, if either Z_+ or Z_- is empty, then (6.5) does not hold, since ρ_{i_+,i_-} does not exist. Therefore, $Set(\tilde{M}) = \mathbb{R}^n$, i.e., $\tilde{M}$ is a zero matrix. This implies that we should be able to find a $u \in \mathbb{R}$ such that $M \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}$ for any $x \in \mathbb{R}^n$.

First we consider the case where only Z_+ is nonempty. Because we have not used (6.5) while proving the equality for $i \in Z_+$ in the previous case, we can conclude that (6.6) satisfies $M \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}$ for any $x \in \mathbb{R}^n$.

Finally, let only Z_- be nonempty. We choose

$$u = \max_{i_- \in Z_-} \frac{1-(M_1)_{(i_-,:)}x}{(M_2)_{i_-}}. \tag{6.7}$$

Then we have

$$\begin{aligned}
M_{(i,:)} \begin{pmatrix} x \\ u \end{pmatrix} &= (M_1)_{(i,:)}x + (M_2)_i u, \\
&= (M_1)_{(i,:)}x + (M_2)_i \left(\max_{i_- \in Z_-} \frac{1-(M_1)_{(i_-,:)}x}{(M_2)_{i_-}} \right), \\
&\leq (M_1)_{(i,:)}x + (M_2)_i \left(\frac{1-(M_1)_{(i,:)}x}{(M_2)_i} \right) = 1,
\end{aligned}$$

for every $i = 1, \dots, l$ and for every $x \in \mathbb{R}^n$. Note that $(M_2)_i$ is negative in this inequality.

As a result, we know $\tilde{M}$ satisfies (6.4) for '$\Rightarrow$'. Now we prove '$\Leftarrow$'.

We assume that there exists a $u \in \mathbb{R}$ such that $M \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}$. We need to prove that $\tilde{M}x \leq \mathbf{1}$. Again we consider a row-by-row analysis. We consider the relevant cases of emptyness and nonemptyness of Z_+, Z_- and Z_0. Just as mentioned above, $Set(\tilde{M}) = \mathbb{R}^n$ if only Z_+ is nonempty or only Z_- is nonempty. In that case we say $\tilde{M}$ is a zero matrix. Furthermore, we have already seen that if only Z_0 is nonempty, then

$$\begin{aligned} \tilde{M}x &= (M_1)x, \\ &= M \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1}. \end{aligned}$$

So the rows of $\tilde{M}$ based on ρ_{i_0} satisfy our inequality for every $i_0 \in Z_0$. In order for us to prove the inequality for any $\rho_{j,i}$, where $j \in Z_+$ and $i \in Z_-$, we first state that we already have the inequalities

$$\begin{aligned} (M_1)_{(j,:)}x + (M_2)_j u &\leq 1, \\ (M_1)_{(i,:)}x + (M_2)_i u &\leq 1. \end{aligned}$$

We can rewrite these inequalities to obtain

$$\begin{aligned} (M_1)_{(j,:)}x &\leq 1 - (M_2)_j u, \\ (M_1)_{(i,:)}x &\leq 1 - (M_2)_i u. \end{aligned}$$

These inequalities are used in the following

$$\begin{aligned} \rho_{j,i}x &= \frac{1}{(M_2)_j - (M_2)_i} \left((M_2)_j (M_1)_{(i,:)} - (M_2)_i (M_1)_{(j,:)} \right) x, \\ &= \frac{1}{(M_2)_j - (M_2)_i} \left((M_2)_j (M_1)_{(i,:)}x - (M_2)_i (M_1)_{(j,:)}x \right), \\ &\leq \frac{1}{(M_2)_j - (M_2)_i} \left((M_2)_j (1 - (M_2)_i u) - (M_2)_i (1 - (M_2)_j u) \right), \\ &= \frac{1}{(M_2)_j - (M_2)_i} \left((M_2)_j - (M_2)_j (M_2)_i u - (M_2)_i + (M_2)_i (M_2)_j u \right), \\ &= \frac{1}{(M_2)_j - (M_2)_i} \left((M_2)_j - (M_2)_i \right) = 1. \end{aligned}$$

Since j and i were chosen arbitrarily, we have proved $\tilde{M}x \leq \mathbf{1}$ holds. ■

We explain the *Rack* function in an intuitive way using the set K_1. We rewrite K_1 to

$$K_1 = \left\{ x \in Set(M_0) : \exists u \in \mathbb{R} \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set \begin{pmatrix} M_{\frac{1}{2}} A & M_{\frac{1}{2}} B_2 \end{pmatrix} \right\}.$$

We are interested in every $x \in \mathbb{R}^n$ such that there exists a $u \in \mathbb{R}$ that satisfies

$$\left(M_{\frac{1}{2}}A\right)_{(i,:)} x + \left(M_{\frac{1}{2}}B_2\right)_{(i,:)} u \leq 1,$$

for every $i = 1, \ldots, 2q$. Now we can construct the sets Z_+, Z_- and Z_0 as defined in Proposition 6.1.2. Consider $i \in Z_0$, then we have

$$\left(M_{\frac{1}{2}}A\right)_{(i,:)} x \leq 1.$$

According to Proposition 6.1.2 this row is used directly for the matrix $\tilde{M}$. And for good reason, since this inequality is a restriction to x only. Now suppose that we have $i \in Z_+$. Consider looking at the u-axis of this halfspace. Since $i \in Z_+$, we know that $\left(M_{\frac{1}{2}}B_2\right)_{(i,:)}$ must be a positive number. Then we know that our halfspace crosses the u-axis above the origin. Hence, for $x = 0$ we get $u \leq \frac{1}{\left(M_{\frac{1}{2}}B_2\right)_{(i,:)}}$, which is a positive number. If we would compare this restriction with any $j \in Z_+$, then we would get no restriction for x. Note that we can choose some $u \in \mathbb{R}$ with the restriction

$$u \leq \min \left\{ \frac{1 - \left(M_{\frac{1}{2}}A\right)_{(i,:)} x}{\left(M_{\frac{1}{2}}B_2\right)_{(i,:)}}, \frac{1 - \left(M_{\frac{1}{2}}A\right)_{(j,:)} x}{\left(M_{\frac{1}{2}}B_2\right)_{(j,:)}} \right\},$$

for any $x \in \mathbb{R}^n$. But if we compare any restriction of $i \in Z_+$ with some $j \in Z_-$, then we can have a situation where there is no u possible to maintain both inequalities. For any $x \in \mathbb{R}^n$ we can rewrite the corresponding inequalities such that $u \in \mathbb{R}$ must satisfy

$$\frac{1 - \left(M_{\frac{1}{2}}A\right)_{(j,:)} x}{\left(M_{\frac{1}{2}}B_2\right)_{(j,:)}} \leq u \leq \frac{1 - \left(M_{\frac{1}{2}}A\right)_{(i,:)} x}{\left(M_{\frac{1}{2}}B_2\right)_{(i,:)}}.$$

There are a lot of cases where we can find some $x \in \mathbb{R}^n$ for which there exists no $u \in \mathbb{R}$ that satisfies this inequality. But since we only want to know every $x \in \mathbb{R}^n$ for which there does exist a $u \in \mathbb{R}$ satisfying our inequalities, we must put a restriction on $x \in \mathbb{R}^n$. It is shown in the proof that the choice of ρ_{i_+,i_-} is sufficient.

By constructing a matrix that contains all rows ρ_{i_+,i_-} and ρ_{i_0}, we have defined an element of the *Rack* function. Since the order of the matrix N in our definition of *Set*(N) does not matter, we can make the following conclusion.

$$K_1 = Set(M_0) \cap Set(N),$$

where $N \in Rack\left(M_{\frac{1}{2}}A \quad M_{\frac{1}{2}}B_2\right)$. By choosing $M_1 = \begin{pmatrix} M_0 \\ N \end{pmatrix}$, we can denote

$$K_1 = Set(M_1).$$

In a similar fashion, we can repeat this process recursively. First try to generate $M_{j\frac{1}{2}}$ based on M_j. If it exists, then continue to construct M_{j+1} based on $M_{j\frac{1}{2}}$. If it does not exist, then we need to restart with an increased value of γ. The algorithm can be summarized in the following way.

1. Initialize $\gamma > 0$ and set $j = 0$ and
$$M_0 = \begin{pmatrix} C \\ -C \end{pmatrix}.$$
2. Execute the applicable step:
   (a) If $\frac{1}{\gamma} |(M_j B_1)| \geq 1$, then increase γ and restart the algorithm.
   (b) If $\frac{1}{\gamma} |(M_j B_1)| < 1$, then set for every row
$$\left(M_{j\frac{1}{2}}\right)_{(i,:)} = \frac{1}{1 - \frac{1}{\gamma}\left|(M_0 B_1)_{(i,:)}\right|} (M_0)_{(i,:)}.$$
3. Set
$$M_{j+1} = \begin{pmatrix} M_j \\ N \end{pmatrix}$$
for any $N \in \mathit{Rack}\left(M_{j\frac{1}{2}}A \quad M_{j\frac{1}{2}}B_2\right)$. Execute the applicable step:
   (a) If $\mathit{Set}(M_{j+1}) = \mathit{Set}(M_j)$, then output M_j.
   (b) If $\mathit{Set}(M_{j+1}) \neq \mathit{Set}(M_j)$, then increase j by 1 and return to step 2.

The algorithm will restart whenever $\gamma \leq \gamma_{optimal}$. Otherwise, we have
$$\mathit{CINV}(\mathit{dom}(U_\gamma)) = \bigcap_{i=0}^{\infty} \mathit{Set}(M_i) = \mathit{Set}(M_j).$$
Note that this algorithm also works when $B_1 = 0$, i.e., the state is not affected by disturbances. The only alteration is that $M_{j\frac{1}{2}} = M_j$, i.e., step 2 is skipped. However, physically speaking it is not interesting. Therefore, we do not consider it as a separate case.

6.2 Algorithm Case 2

We now construct a similar algorithm in the case where $D_1 \neq 0$ and $D_2 = 0$, i.e., the output is influenced by the disturbance and not by the controlled input. The only difference from the algorithm in case 1 is found in the initial set.
$$K_0 = \mathit{dom}(U_\gamma) = \left\{ x \in \mathbb{R}^n : |Cx + \frac{1}{\gamma} D_1 w| \leq 1, \quad \forall |w| \leq 1 \right\}.$$

Here we must construct M_0 in a different manner. Similarly to how we define $K_{j\frac{1}{2}}$, we now define the set $K_{-\frac{1}{2}}$ as

$$K_{-\frac{1}{2}} = \left\{ \xi \in \mathbb{R}^p : |\xi + \frac{1}{\gamma} D_1 w| \leq 1, \quad \forall |w| \leq 1 \right\}.$$

By using a row-by-row analysis, we can construct the matrix $M_{-\frac{1}{2}}$ as

$$(M_{-\frac{1}{2}})_{(i,:)} = \frac{1}{1 - \frac{1}{\gamma}|(D_1)_{(i,:)}|} \begin{pmatrix} I \\ -I \end{pmatrix}_{(i,:)},$$

where I is the identity matrix. This is done in a similar way as (6.1) is derived. Also similar as in the construction of $M_{j\frac{1}{2}}$, we get a constraint. But this time, our constraint is

$$\frac{1}{\gamma}|D_1| < 1.$$

If this constraint is not met, then γ needs to be increased. Now we have a matrix $M_{-\frac{1}{2}}$ such that $K_{-\frac{1}{2}} = Set\left(M_{-\frac{1}{2}}\right)$. Then we can write

$$\begin{aligned} K_0 &= \left\{ x \in \mathbb{R}^n : Cx \in K_{-\frac{1}{2}} \right\}, \\ &= \left\{ x \in \mathbb{R}^n : Cx \in Set\left(M_{-\frac{1}{2}}\right) \right\}, \\ &= \left\{ x \in \mathbb{R}^n : M_{-\frac{1}{2}} Cx \leq \mathbf{1} \right\}. \end{aligned}$$

Hence, we have

$$M_0 = M_{-\frac{1}{2}} C.$$

The rest of the algorithm is the same as in the previous case. So we alter the first step from the previous algorithm in the following way.

1. Initialize $\gamma > 0$ and set $j = 0$. Then execute the applicable action:
   (a) If $\frac{1}{\gamma}|D_1| \geq 1$, then increase γ and restart the algorithm.
   (b) If $\frac{1}{\gamma}|D_1| < 1$, then initialize

$$M_{-\frac{1}{2}} = \begin{pmatrix} \frac{1}{1-\frac{1}{\gamma}|(D_1)_{(1,:)}|} & & \\ & \ddots & \\ & & \frac{1}{1-\frac{1}{\gamma}|(D_1)_{(p,:)}|} \\ -\frac{1}{1-\frac{1}{\gamma}|(D_1)_{(1,:)}|} & & \\ & \ddots & \\ & & -\frac{1}{1-\frac{1}{\gamma}|(D_1)_{(p,:)}|} \end{pmatrix}$$

   and define $M_0 = M_{-\frac{1}{2}} C$.

Continue the algorithm with step 2 and 3 as defined in the previous case.

Now suppose $D_2 \neq 0$ and let $D_1 = 0$, i.e., the output is influenced by the controlled input and not by disturbances. Our initial set is now defined as

$$K_0 = dom(U_\gamma) = \{x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } |Cx + D_2 u| \leq 1\}.$$

We can rewrite this set in the following way.

$$\begin{aligned} K_0 &= \{x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } |Cx + D_2 u| \leq 1\}, \\ &= \left\{x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } \begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1} \right\}, \\ &= \left\{x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set\begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix} \right\}. \end{aligned}$$

Using the *Rack* function from Definition 6.1.1, we can say that $K_0 = Set(M_0)$ for any

$$M_0 \in Rack\begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix}.$$

With M_0 initialized, we can construct $M_{\frac{1}{2}}$ in the same manner as described in case 1. Hence, step 2 remains the same as in case 1. Step 3 is different again. We have

$$\begin{aligned} K_1 &= \left\{x \in K_0 : \exists u \in U_\gamma(x) \text{ such that } f(x,u) \in K_{\frac{1}{2}} \right\}, \\ &= \left\{x \in Set(M_0) : \exists u \in U_\gamma(x) \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set\begin{pmatrix} M_{\frac{1}{2}}A & M_{\frac{1}{2}}B_2 \end{pmatrix} \right\}. \end{aligned}$$

In case 1, we have $U_\gamma(x) = \mathbb{R}$. This does not hold in case 3. Now we have

$$U_\gamma(x) = \left\{u \in \mathbb{R} : \begin{pmatrix} x \\ u \end{pmatrix} \in Set\begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix} \right\}.$$

Hence, we have 3 restrictions.

1. $x \in Set(M_0)$.
2. $\exists u \in \mathbb{R}$ such that $\begin{pmatrix} x \\ u \end{pmatrix} \in Set\begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix}$.
3. $\exists u \in \mathbb{R}$ such that $\begin{pmatrix} x \\ u \end{pmatrix} \in Set\begin{pmatrix} M_{\frac{1}{2}}A & M_{\frac{1}{2}}B_2 \end{pmatrix}$.

In order to obtain all restrictions for x, we define $M_1 = \begin{pmatrix} M_0 \\ N \end{pmatrix}$, with

$$N \in Rack\begin{pmatrix} C & D_2 \\ -C & -D_2 \\ M_{\frac{1}{2}}A & M_{\frac{1}{2}}B_2 \end{pmatrix}.$$

Just as in case 1, we repeat step 2 and step 3. But step 3 is altered, in the sense that N is now defined as

$$N \in Rack \begin{pmatrix} C & D_2 \\ -C & -D_2 \\ M_{j\frac{1}{2}} A & M_{j\frac{1}{2}} B_2 \end{pmatrix}.$$

When we have $D_1 \neq 0$ and $D_2 \neq 0$, i.e., the output is influenced by both the controlled input and disturbances, we can almost combine the alterations of case 1 that are discussed above. We start with constructing $M_{-\frac{1}{2}}$ as described above. Then we have

$$\begin{aligned} K_0 &= \left\{ x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } |Cx + \frac{1}{\gamma} D_1 w + D_2 u| \leq 1, \quad \forall |w| \leq 1 \right\}, \\ &= \left\{ x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } M_{-\frac{1}{2}} \begin{pmatrix} C & D_2 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \leq \mathbf{1} \right\}, \\ &= \left\{ x \in \mathbb{R}^n : \exists u \in \mathbb{R} \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set \begin{pmatrix} M_{-\frac{1}{2}} C & M_{-\frac{1}{2}} D_2 \end{pmatrix} \right\}. \end{aligned}$$

Hence, $M_0 \in Rack \begin{pmatrix} M_{-\frac{1}{2}} C & M_{-\frac{1}{2}} D_2 \end{pmatrix}$. This difference from the case $D_1 = 0$ and $D_2 \neq 0$ comes back similarly in step 3. We summarize the algorithm in the case $D_1 \neq 0$ and $D_2 \neq 0$.

1. Initialize $\gamma > 0$ and set $j = 0$. Then execute the applicable action:

   (a) If $\frac{1}{\gamma}|D_1| \geq 1$, then increase γ and restart the algorithm.

   (b) If $\frac{1}{\gamma}|D_1| < 1$, then initialize

$$M_{-\frac{1}{2}} = \begin{pmatrix} \frac{1}{1-\frac{1}{\gamma}|(D_1)_{(1,:)}|} & & \\ & \ddots & \\ & & \frac{1}{1-\frac{1}{\gamma}|(D_1)_{(p,:)}|} \\ -\frac{1}{1-\frac{1}{\gamma}|(D_1)_{(1,:)}|} & & \\ & \ddots & \\ & & -\frac{1}{1-\frac{1}{\gamma}|(D_1)_{(p,:)}|} \end{pmatrix}.$$

   Now we define

$$M_0 \in Rack \begin{pmatrix} M_{-\frac{1}{2}} C & M_{-\frac{1}{2}} D_2 \end{pmatrix}.$$

2. Execute the applicable step:

   (a) If $\frac{1}{\gamma}\left|(M_j B_1)\right| \geq 1$, then increase γ and restart the algorithm.

(b) If $\frac{1}{\gamma}\left|(M_j B_1)\right| < 1$, then set for every row

$$\left(M_{j\frac{1}{2}}\right)_{(i,:)} = \frac{1}{1-\frac{1}{\gamma}\left|(M_0 B_1)_{(i,:)}\right|}\left(M_0\right)_{(i,:)}.$$

3. Set

$$M_{j+1} = \begin{pmatrix} M_j \\ N \end{pmatrix}$$

for any $N \in Rack\begin{pmatrix} M_{-\frac{1}{2}}C & M_{-\frac{1}{2}}D_2 \\ M_{j\frac{1}{2}}A & M_{j\frac{1}{2}}B_2 \end{pmatrix}$. Execute the applicable step:

(a) If $Set(M_{j+1}) = Set(M_j)$, then output M_j.

(b) If $Set(M_{j+1}) \neq Set(M_j)$, then increase j by 1 and return to step 2.

6.3 Algorithm Case 3

Now we assume that we have a multivariable controlled input, i.e., $u \in \mathbb{R}^m$ with $m > 1$. In all the cases above, we use the Rack function. This function is based on a single control input. Hence, we cannot apply it in the same way when u is not a single input. However, we can use it in a different manner. Suppose we follow the algorithm in case 1 or 2, whichever is applicable, but now we have $u \in \mathbb{R}^m$ with $m > 1$. Then we would encounter a situation where we would need to use the *Rack* function at some point. The situation can be sketched as the following problem:
We want to determine the set

$$S = \left\{ x \in \mathbb{R}^n : \exists u \in \mathbb{R}^m \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set(M) \right\},$$

for some matrix $M \in \mathbb{R}^{l\times(n+m)}$. To construct this set, we define $v = \begin{pmatrix} u_1 \\ \vdots \\ u_{m-1} \end{pmatrix}$. Then we can denote $u = \begin{pmatrix} v \\ u_m \end{pmatrix}$. The first step is to construct the set

$$\left\{ \begin{pmatrix} x \\ v \end{pmatrix} \in \mathbb{R}^{n+(m-1)} : \exists u_m \in \mathbb{R} \text{ such that } \begin{pmatrix} x \\ v \\ u_m \end{pmatrix} \in Set\left(M\right) \right\}.$$

This set can simply be denoted as $Set(M_{temp})$, where $M_{temp} \in Rack(M)$. Using this, we can rewrite S as

$$S = \left\{ x \in \mathbb{R}^n : \exists v \in \mathbb{R}^{m-1} \text{ such that } \begin{pmatrix} x \\ v \end{pmatrix} \in Set(M_{temp}) \right\}.$$

We can repeat these steps recursively m times to obtain a matrix $N \in \mathbb{R}^{l_m \times n}$ such that $S = Set(N)$, where l_m denotes the appropriate number of rows of N. In other words, we apply the *Rack* function m times. Therefore we have the following definition.

Definition 6.3.1. *Let $M \in \mathbb{R}^{l \times (n+m)}$ for $m > 1$. We define $Rack^m(M)$ as the set of matrices $\tilde{M}$, such that $\tilde{M} \in Rack(M_{temp})$ for some $M_{temp} \in Rack^{m-1}(M)$.*

Note that $Rack^1(M)$ is simply $Rack(M)$ as defined in Definition 6.1.1. As a result we have the following.

Proposition 6.3.2. *Let $M \in \mathbb{R}^{l \times (n+m)}$. Let $\tilde{M} \in Rack^m(M)$. Then*

$$x \in Set(\tilde{M}) \subseteq \mathbb{R}^n \Leftrightarrow \exists u \in \mathbb{R}^m \text{ such that } \begin{pmatrix} x \\ u \end{pmatrix} \in Set(M) \subseteq \mathbb{R}^{n+m}.$$

When we want to apply the Controlled Invariant Kernel algorithm with multi-variable input, we simply follow the steps from either case 1 or case 2 (whichever is applicable), but instead of *Rack* we use *Rack*^{m}.

6.4 Algorithm Case 4

We assumed in all the cases above that $dom(U_\gamma)$ is a bounded set, so we can construct $CINV(dom(U_\gamma))$. Now assume that $dom(U_\gamma)$ is unbounded. Then we want to construct $CINV(K)$, where K is defined as in Definition 5.0.4. We alter the algorithms from the previous cases in step 1 only. Because the only difference is that K_0 is defined as K, instead of $dom(U_\gamma)$. We start with the case $D_1 = D_2 = 0$. Then we have

$$K_0 = \{x \in \mathbb{R}^n : |Cx| \leq 1, \quad |x| \leq L\}.$$

If we choose L very small, then we try to construct a controlled invariant subset of a very small square/cube/hypercube. Since this is not useful most of the time, we want to choose a large enough L such that the restrictions of $dom(U_\gamma)$ are not redundant. Hence, we want to cut off $dom(U_\gamma)$ in the direction that is infinitely large. In order to do this, we would need to construct $dom(U_\gamma)$ in the usual way. Then we want to choose L such that no restrictions of $dom(U_\gamma)$ become redundant for K. Even better, we want to choose L such that most constraints of $|x| \leq L$ become redundant. This can be calculated, and therefore we have a lower bound for L. However, to determine this lower bound, or the exact value for L for that matter, we need to know the dynamics of the system itself. It is even possible that you can construct $CINV(dom(U_\gamma))$ even though $dom(U_\gamma)$ is unbounded, because there is some K_j that must be bounded. This is the case in the following example.

Example 6.4.1. Consider the system

$$\begin{aligned} x(k+1) &= \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} x(k) + w(k) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(k), \\ z(k) &= \begin{pmatrix} 1 & 0 \end{pmatrix} x(k). \end{aligned}$$

Then we have

$$dom(U_\gamma) = \{x \in \mathbb{R}^n : |x_1| \leq 1\},$$

which is clearly an unbounded set. Now if we choose $L \leq 1$, then we have

$$K_0 = K = \{x \in \mathbb{R}^n : |x_1| \leq 1, \quad |x| \leq L\} = \{x \in \mathbb{R}^n : |x| \leq L\}.$$

Hence, the constraints from $dom(U_\gamma)$ are redundant. Therefore it is possible that we consider a set that is too small to have a controlled invariant subset. Because computing the algorithm takes time, we want to do this as efficiently as possible. In this case, we can try to compute the next step of the Controlled Invariant Kernel algorithm with $K_0 = dom(U_\gamma)$, with $\gamma > 0$ sufficiently large. We have

$$K_{\frac{1}{2}} = \left\{\xi \in \mathbb{R}^n : \xi + \frac{1}{\gamma}w \in K_0 \quad \forall |w| \leq 1\right\} = \left\{\xi \in \mathbb{R}^n : \frac{1}{1-\frac{1}{\gamma}}|\xi_1| \leq 1\right\}$$

and

$$K_1 = \left\{x \in K_0 : \exists u \in \mathbb{R} \text{ such that } Ax + B_2u \in K_{\frac{1}{2}}\right\} = \left\{x \in K_0 : \frac{1}{1-\frac{1}{\gamma}}|x_1 - x_2| \leq 1\right\}.$$

Due to the dynamics of the system, we get restrictions in K_1 that assures that the set is bounded. Hence, $CINV(dom(U_\gamma))$ is bounded, while $dom(U_\gamma)$ is unbounded.

$\square$

In conclusion, an educated guess has to be made in order to determine L, but it is possible to construct a bounded controlled invariant subset, regardless of the boundedness of K_0.

In the case where $D_1 \neq 0$ and $D_2 \neq 0$, we construct K_0 as shown in the algorithm. For example, we have constructed $M_0 \in \mathbb{R}^{l \times n}$ such that

$$dom(U_\gamma) = Set(M_0).$$

Then we define

$$K_0 = Set \begin{pmatrix} M_0 \\ \frac{1}{L}I \\ -\frac{1}{L}I \end{pmatrix},$$

where I denotes the identity matrix.

Now that we have shown how the Controlled Invariant Kernel algorithm can be implemented in Matlab, we can construct the value of a controller.

6.5 Construction of a Controller; Single-Input Case

When the applicable algorithm is finished, we have a nonempty $CINV(K)$, that can be described as the intersection of half-spaces. Hence, we have a matrix $\tilde{M} \in \mathbb{R}^{\tilde{l} \times n}$ such that $CINV(K) = Set(\tilde{M})$. If we would apply step 2 and step 3 of the applicable algorithm one more time, then we would construct a matrix $M \in \mathbb{R}^{l \times (n+m)}$, such that $\tilde{M} \in Rack^m(M)$. Based on the matrix M, we construct explicit continuous selection functions.
We consider $u \in \mathbb{R}$. During the proof of the *Rack* function, we have already shown two controlled inputs of the form $g : \mathbb{R}^n \to \mathbb{R}$.

Proposition 6.5.1. *In the framework of Proposition 6.1.2, let Z_+ and Z_- be nonempty. Based on* (6.6)*, we define $\phi_+^M : \mathbb{R}^n \to \mathbb{R}$ by*

$$\phi_+^M(x) = \min_{i_+ \in Z_+} \frac{1 - (M_1)_{(i_+,:)}\, x}{(M_2)_{i_+}}.$$

Based on (6.7)*, we define $\phi_-^M : \mathbb{R}^n \to \mathbb{R}$ by*

$$\phi_-^M(x) = \max_{i_- \in Z_-} \frac{1 - (M_1)_{(i_-,:)}\, x}{(M_2)_{i_-}}.$$

Then ϕ_+^M and ϕ_-^M are continuous and satisfy

$$\begin{pmatrix} x \\ \phi_+^M(x) \end{pmatrix} \in Set(M), \quad \forall x \in Set(\tilde{M}),$$
$$\begin{pmatrix} x \\ \phi_-^M(x) \end{pmatrix} \in Set(M), \quad \forall x \in Set(\tilde{M}),$$

where $\tilde{M} \in Rack(M)$.

This proposition is derived from the proof of Proposition 6.1.2. The functions ϕ_+^M and ϕ_-^M are explicitly constructed continuous selections from the set-valued map $R(x)$, defined in the proof of Theorem 5.0.1. In the proof of Theorem 5.0.1, it is also shown that $R(x)$ is compact-valued for every $x \in CINV(K)$. Therefore, we can conclude that any convex combination

$$\lambda \phi_+^M + (1 - \lambda)\phi_-^M,$$

where $\lambda \in [0, 1]$, is also a continuous selection function. Recall that we desire a controller with the property $g(0) = 0$. Since we have

$$\phi_+^M(0) = \min_{i_+ \in Z_+} \frac{1}{(M_2)_{i_+}},$$
$$\phi_-^M(0) = \max_{i_- \in Z_-} \frac{1}{(M_2)_{i_-}},$$

we know that $\phi_+^M(0) = -\phi_-^M(0)$ by the symmetry with respect to the origin, from Lemma 5.0.5. Hence, if we define $g : \mathbb{R}^n \to \mathbb{R}^m$ as

$$g(x) = \frac{1}{2}\phi_+^M(x) + \frac{1}{2}\phi_-^M(x),$$

then we have a continuous selection with the property $g(0) = 0$.

6.6 Construction of a Controller; Multiple-Input Case

For $u \in \mathbb{R}^m$ we cannot apply the *Rack* function. We worked around this problem by defining $Rack^m$. For the construction of a controller, we have shown that we are able to construct it using the input of the *Rack* function. Recall that $Rack^m$ is actually the *Rack* function applied recursively m times. We use this to construct a multi-variable controller, as shown in [10].

We consider the situation where $CINV(K) = Set(\tilde{M})$, with $\tilde{M} \in Rack^m(M)$, as described in the previous section. We want to construct a continuous selection function $g : \mathbb{R}^n \to \mathbb{R}^m$. We construct this function step by step, starting with $g_1 : \mathbb{R}^n \to \mathbb{R}$. To do that, we construct a matrix $M^* \in \mathbb{R}^{l^* \times (n+1)}$ such that $M^* \in Rack^{m-1}(M)$. By the definitions of *Rack* and $Rack^m$, we know $\tilde{M} \in Rack(M^*)$. Then we apply the way we constructed a controller in case we have single-input controller for g_1, resulting in

$$g_1(x) = \frac{1}{2}\phi_+^{M^*}(x) + \frac{1}{2}\phi_-^{M^*}(x).$$

Next, we apply a similar step using the now known value of $g_1(x)$. We construct a matrix $\hat{M} \in \mathbb{R}^{\hat{l} \times (n+2)}$ such that $\hat{M} \in Rack^{m-2}(M)$. Similar as above, we have $M^* \in Rack(\hat{M})$. Then we define $g_2 : \mathbb{R}^n \to \mathbb{R}$ as

$$g_2(x) = \frac{1}{2}\phi_+^{\hat{M}}\begin{pmatrix} x \\ g_1(x) \end{pmatrix} + \frac{1}{2}\phi_-^{\hat{M}}\begin{pmatrix} x \\ g_1(x) \end{pmatrix}.$$

In order to construct the other component of g, we apply the same steps, expressing $g_i(x)$ in terms of x and every $g_j(x)$ where $j < i$. The last component is described as

$$g_m(x) = \frac{1}{2}\phi_+^{\hat{M}}\begin{pmatrix} x \\ g_1(x) \\ \vdots \\ g_{m-1}(x) \end{pmatrix} + \frac{1}{2}\phi_-^{\hat{M}}\begin{pmatrix} x \\ g_1(x) \\ \vdots \\ g_{m-1}(x) \end{pmatrix}.$$

As a result, we have $g : \mathbb{R}^n \to \mathbb{R}^n$ with $g(0) = 0$.

Chapter 7

Improvements on the Controlled Invariant Kernel Algorithm

We have introduced a number of improvements in the CIK algorithm with respect to the original algorithm. In this chapter we discuss these improvements.

7.1 Stopping criterion

One of the issues that arises in the algorithm is to determine whether we should decrease γ or if we should construct a controller. We do that based on a pre-determined precision. The CIK algorithm terminates whenever $\gamma \leq \gamma_{optimal}$ and produces a matrix $\tilde{M}$ such that $CINV(K) = Set(\tilde{M})$ whenever $\gamma > \gamma_{optimal}$. Therefore we create the following algorithm.

1. Give the system and a desired precision as input.
2. One of the following is executed:
   (a) If the system is not controllable, the algorithm is terminated.
   (b) If the system is controllable, the needed parameters are initialized. We set $\gamma = 0$ and $\delta = 10$.
3. The applicable CIK algorithm is executed.
   (a) If the CIK algorithm is terminated, set $\gamma = \gamma + \delta$ and start the CIK algorithm again.
   (b) If the CIK algorithm produces a controlled invariant set, then execute one of the following:
      i. If δ is greater than the precision, then set $\delta = \frac{\delta}{2}$ and $\gamma = \gamma - \delta$. Then start the CIK algorithm again.
      ii. If δ is less than or equal to the precision, then output the matrix $\tilde{M}$ and the matrix M.

We choose $\delta = 10$ based on trial and error. $\tilde{M}$ is the matrix for which $CINV(K) = Set(\tilde{M})$. The matrix M is needed to construct the feedback controller. In order to apply the knowledge gained from this algorithm, it is sufficient to use the matrix M only. One can start the system with zero initial conditions, and then determine the feedback controller based on M. The matrix $\tilde{M}$ is given so the user can check if the state actually remains in $CINV(K)$.

This algorithm requires a lot of computational power. It is beneficial if we can decrease the computational time of this algorithm. We approach this problem in two ways.

7.2 Deletion of Redundancies

Shamma proposes in [10] that linear programs are required for computationally efficient implementation to remove redundant constraints. Due to the *Rack* function, we may gain several redundant constraints when making the step from M_j to M_{j+1}. We implemented our algorithm in Matlab, enabling us to use the command convhulln. We give our matrix M_{j+1} as input. The matrix represents the set denoted by $Set(M_{j+1})$. The command convhulln determines the convex hull of the given set. A convex hull of X is the smallest convex set containing X. If $Set(M_{j+1})$ is not a convex hull, then convhulln(M_{j+1}) is described as a subset of the rows of M_{j+1}. We remove all rows of M_{j+1} that are not needed to construct the convex hull, i.e., are redundant. With the removal of these rows, we have to make less calculations when executing the next steps.

Unlike Shamma, we do not only remove redundant rows when computing M_{j+1}. We also implemented several steps in our *Rack* function to remove redundant constraints. Since $Rack^m$ is defined as applying the *Rack* function m times recursively, the number of constraints increases rapidly.

Consider the framework of Proposition 6.1.2. Suppose that we want to determine an element of $Rack^m(M)$, where $M \in \mathbb{R}^{l\times(n+m)}$. In the worst case, we have Z_0 as the empty set and both Z_+ and Z_- with length $\frac{1}{2}l$. Since we combine every element of Z_+ with every element of Z_-, we get a matrix $\hat{M} \in Rack(M)$, where $M \in \mathbb{R}^{\frac{1}{4}l^2\times(n+m-1)}$. Hence, the number of constraints have already gone up by applying only one *Rack* function. When we determine $\tilde{M} \in Rack^m(M)$ without removing any redundant constraints, the matrix $\tilde{M}$ implies a number of constraint of the order $\mathcal{O}(l^{2m})$.

To reduce the number of constraints that $Rack^m$ produces, we implement a series of steps. By Lemma 5.0.5, we know that any set K_j is symmetric with respect to the origin. Therefore, when applying $Rack(M)$, we know that

$$\forall i \in Z_+ \; \exists j \in Z_- \text{ such that } M_{(i,:)} = -M_{(j,:)}.$$

As a result, $\rho_{i,j}$ is a row of zeros. Since a row of zeros implies no constraint at all, we can remove these. This is easily done with the command any.

Next, we remove any copies of rows. Since the ρ_{i_+,i_-} are scalings of $M_{(i_+,:)}$ and $M_{(i_-,:)}$, it can happen that two different combinations result in the same rows. To remove the same row, we use the command `unique`.

Finally, we apply our `convhulln`, which is already explained. The *Rack* function is implemented in the following way.

1. Give the matrix M and the number of controlled inputs m as input.
2. Define M_1, M_2, Z_+, Z_- and Z_0 as described in Proposition 6.1.2.
3. Construct all ρ_{i_+,i_-} and ρ_{i_0} and add them together in a matrix N. (The order does not matter).
4. Remove any rows of zeros of N.
5. Remove any copies of rows of N.
6. Remove any redundant rows of N.
7. Execute one of the following:
   (a) If $m = 1$, then output N.
   (b) If $m > 1$, then execute the *Rack* function with the input N and $m - 1$.

By adding steps 4 through 6, the number of calculations is generally decreased a lot. Since steps 4 and 5 hardly take any time to compute, this is very beneficial to reduce the computational time.

7.3 Computational Storage

Suppose we start the CIK algorithm for a fixed γ. The matrices M_0 and $M_{\frac{1}{2}}$ are determined before applying the *Rack* function to the matrix $N = M_{\frac{1}{2}} \begin{pmatrix} A & B_2 \end{pmatrix}$. As mentioned in the previous section, any row ρ_{i_+,i_-} in the *Rack* function is a scaling from the rows $N_{(i_+,:)}$ and $N_{(i_-,:)}$. As a result, we construct the matrix $M_1 = \begin{pmatrix} M_0 \\ \tilde{M} \end{pmatrix}$, where $\tilde{M} \in \mathit{Rack}(N)$. After the redundant rows have been removed from M_1, it is possible that some of the rows of M_1 are also stated in M_0. As a result, after determining $M_{1\frac{1}{2}}$ and applying the *Rack* function to $N_2 = M_{1\frac{1}{2}} \begin{pmatrix} A & B_2 \end{pmatrix}$, there are some rows $\hat{\rho}_{i_+,i_-}$ based on $(N_2)_{(i_+,:)}$ and $(N_2)_{(i_-,:)}$ such that $(N_2)_{(i_+,:)} = N_{(i_+,:)}$ and $(N_2)_{(i_-,:)} = N_{(i_-,:)}$. Hence, the calculation is unnecessarily repeated, since the resulting $\hat{\rho}_{i_+,i_-}$ is already represented by ρ_{i_+,i_-}. This is filtered by step 5 in the previous algorithm, but it would be more efficient to remove the calculation in the first place. Therefore, we have implemented the following solution.

When M_{j+1} is computed and $Set(M_{j+1}) \neq Set(M_j)$, we check if there are rows which also occur in M_j. We do this with the command ismember. The indices of the rows of M_{j+1} are stored in a new vector, called *compRows*. The next step in the CIK algorithm is to construct $M(j+1)\frac{1}{2}$. Note that neither the size, nor the order of the rows is changed with respect to M_{j+1}. Then the *Rack* function the applicable matrix. To ease the notation, we assume the next step is to construct *Rack* of the matrix $M_{(j+1)\frac{1}{2}}\begin{pmatrix} A & B_2 \end{pmatrix}$. This matrix has not changed the order and the number of rows with respect to M_{j+1}. Therefore, we know that the combination of rows, with the indices stored in *compRows*, have already been determined when $Rack\left(M_{j\frac{1}{2}}\begin{pmatrix} A & B_2 \end{pmatrix}\right)$ was computed. By not determining the rows ρ_{i_+,i_-}, where $i_+,\ i_- \in compRows$, we do not loose any constraints, since they were already produced in M_{j+1}. To show how effective this can be, consider the following example.

Example 7.3.1. Assume we apply the CIK algorithm on a system with scalar control and arrive at the matrix $M_j \in \mathbb{R}^{8\times n}$. Based on M_j, we construct a matrix $N \in \mathbb{R}^{8\times(n+1)}$, such that we need to calculate $Rack(N)$. Suppose $Z_+ = \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \end{pmatrix}$ and $Z_- = \begin{pmatrix} 2 \\ 4 \\ 6 \\ 8 \end{pmatrix}$. The number of combinations from Z_+ and Z_- is then 16. Then we have to check whether constraints are redundant for a matrix $M_{j+1} \in \mathbb{R}^{24\times n}$.

It is realistic to assume that 6 of the rows M_j were already in M_{j-1}. For example $compRows = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$. This would imply that we do not have to check any combinations involving two indices from *compRows*. Hence, we only have 7 combinations left, i.e., we have to check for redundancies for a matrix $M_{j+1} \in \mathbb{R}^{15\times n}$.

$\square$

In this example, the dimensions are relatively low, so the result should not be noticeable when timing the algorithm. However, when the number of constraints is increased, the impact of the *compRows* becomes more significant.

There is one problem with this algorithm. In case we have a system where $D_2 \neq 0$ and $u \in \mathbb{R}^m$, with $m > 1$, then we need to determine an element from $Rack^m\begin{pmatrix} C & D_2 \\ -C & -D_2 \\ M_{j\frac{1}{2}}A & M_{j\frac{1}{2}}B_2 \end{pmatrix}$. The indices of the first rows are elements of *compRows* after the first step for obvi-

ous reasons. Now suppose that $N \in Rack\begin{pmatrix} C & D_2 \\ -C & -D_2 \\ M_{j\frac{1}{2}}A & M_{j\frac{1}{2}}B_2 \end{pmatrix}$. If we use our knowledge from *compRows*, a lot of constraints are not computed in N. The matrix $\tilde{M} \in Rack^m\begin{pmatrix} C & D_2 \\ -C & -D_2 \\ M_{j\frac{1}{2}}A & M_{j\frac{1}{2}}B_2 \end{pmatrix}$ is based on $\tilde{M} \in Rack^{m-1}(N)$. Therefore, we left out a lot of constraints, before applying a *Rack* function.
The idea of *compRows* is that known rows are not compared with each other. When we have $u \in \mathbb{R}^m$ with $m > 1$, we need to apply $Rack^m$. In the first step of the $Rack^m$, we apply *Rack*. There we would filter out combinations of known rows, which results in all new rows. Then the next steps of $Rack^{m-1}$ consist of all new rows, which seems to be what we want. However, we are missing a lot of constraints in this case. Consider the first step again, but instead of filtering out the combinations of known rows, we store the combinations of known rows as $NewcompRows$. As a result, the next step of $Rack^{m-1}$ consists of combinations of known rows and new rows, resulting in constraints that we would miss if we filtered combinations of *compRows* out in the first place. We keep track of all known combinations during the first $m-1$ times the *Rack* function is applied. Then, in the final step, we actually filter out all known combinations. To realize this construction, we define the following *Rack* function.

1. Give the matrix M, the vector *compRows* and the number of controlled inputs m as input.
2. Define M_1, M_2, Z_+, Z_- and Z_0 as described in Proposition 6.1.2.
3. Compute all combinations of Z_+ and Z_- and denote it as matrix Z.
4. Execute one of the following:
   (a) If $m = 1$:
      i. Remove all elements of Z_0 that are also in *compRows*.
      ii. Remove all rows of Z of which both elements also occur in *compRows*.
   (b) If $m > 1$:
      i. Define all elements of Z_0 that are also in *compRows* as $NewcompRows$.
      ii. Add all indices of rows of Z of which both elements occur in *compRows* to $NewcompRows$.
5. Construct all ρ_{i_+,i_-} and ρ_{i_0} based on Z_+, Z_- and Z_0, and add them together in a matrix N. (The order does not matter)
6. Remove any rows of zeros of N.
7. Remove any copies of rows of N.

8. Remove any redundant rows of N.
9. Execute one of the following:
   (a) If $m = 1$, then output N.
   (b) If $m > 1$, then execute the *Rack* function with the input N, $NewcompRows$ and $m - 1$.

7.4 Implementation of Controlled Input Constraints

Up until this point, we have constructed constraints on our controlled input based on the state and the output vectors. In the case where $D_1 = D_2 = 0$, we have $U_\gamma(x) = \mathbb{R}^m$ for every $x \in dom(U_\gamma)$. In other words, we are able to apply an infinitely large controlled input, in theory. This is not possible in practical applications. Therefore, we implemented an extra input to apply upper and lower bounds on the controlled input.

Suppose that the set of possible controlled input values can be described as $Set(E)$, where $E \in \mathbb{R}^{l\times m}$. Here l is the number of constraints and m is the dimension of the controlled input. This notation already implies that the origin must be an interior point, and the set of possible values must be convex, in order to implement the restrictions on the controlled input.

The matrix E is given as another input in our algorithm. After determining that the pair (A, B_2) is controllable, the first step is to construct M_0. Based on the system itself, a different approach is applied. One way or another, we want to construct $dom(U_\gamma)$. Recall that this set is given by

$$dom(U_\gamma) = \left\{ x \in \mathbb{R}^n : \exists u \in U_\gamma(x) \text{ such that } |Cx + \frac{1}{\gamma}D_1 w + D_2 u| \le 1, \quad \forall |w| \le 1 \right\},$$

where $U_\gamma(x)$ is defined as

$$U_\gamma(x) = \left\{ u \in \mathbb{R}^m : |Cx + \frac{1}{\gamma}D_1 w + D_2 u| \le 1, \quad \forall |w| \le 1 \right\}.$$

When the constraints on the controlled input u are applied, we obtain

$$U_\gamma(x) = \left\{ u \in Set(E) : |Cx + \frac{1}{\gamma}D_1 w + D_2 u| \le 1, \quad \forall |w| \le 1 \right\}.$$

In the case where $D_2 \neq 0$, the matrix E is instantly needed, as the *Rack* function needs to be applied. Recall that

$$M_0 \in Rack^m \begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix},$$

or

$$M_0 \in Rack^m \begin{pmatrix} M_{-\frac{1}{2}}A & M_{-\frac{1}{2}}B_2 \end{pmatrix},$$

dependent on whether $D_1 = 0$ or not. For argument sake, we assume that $D_1 \neq 0$. In order to implement the constraints on the controlled input, we add the matrix $\begin{pmatrix} 0 & E \end{pmatrix}$, where 0 denotes a matrix of n by n, as input in our *Rack* function. Resulting in

$$M_0 \in Rack^m \begin{pmatrix} 0 & E \\ M_{-\frac{1}{2}}A & M_{-\frac{1}{2}}B_2 \end{pmatrix}.$$

As a result, the matrix M_0 represents the set of state vectors for which it is possible to attain $||z|| \leq 1$ given the desired constraints on our controlled input.

Similar to the matrices $\begin{pmatrix} C & D_2 \\ -C & -D_2 \end{pmatrix}$ and $\begin{pmatrix} M_{-\frac{1}{2}}A & M_{-\frac{1}{2}}B_2 \end{pmatrix}$, the matrix $\begin{pmatrix} 0 & E \end{pmatrix}$ reoccurs every time the *Rack* function is applied. Note that when we do not assume that $Set(E)$ is symmetric with respect to the origin, then the resulting $CINV(domU_\gamma)$ is also not symmetric with respect to the origin. However, the advantage of using a matrix E such that $Set(E)$ is a bounded set is that K_1 has to be a bounded set, i.e., $CINV(dom(U_\gamma)$ may be constructed.

Chapter 8

Numerical Results

In this chapter we consider several examples to describe different parts of the algorithm proposed in Chapter 7. The first example is used to show how a controlled invariant kernel is constructed step by step. Afterwards, we discuss the precision of the approximation of $\gamma_{optimal}$ affects the computational time of the algorithm.

8.1 Step-by-Step Analysis

We start by showing how the presented algorithm constructs a controlled invariant set. Consider the following system

$$\begin{aligned} x(k+1) &= \begin{pmatrix} -1 & 1 \\ 0 & 2 \end{pmatrix} x(k) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(k), \\ z(k) &= \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} x(k), \end{aligned}$$

with initial condition $x(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

The first step is to construct K_0, depicted in Figure 8.1.

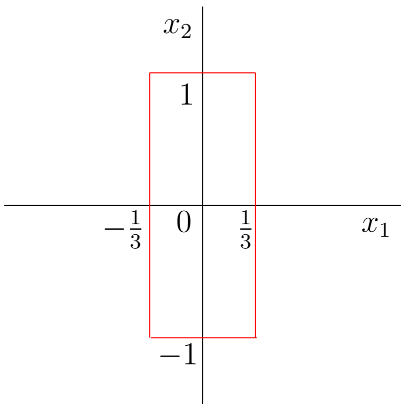

Figure 8.1: The boundary of K_0.

The set K_0 is represented by the matrix $M_0 = \begin{pmatrix} 3 & 0 \\ 0 & 1 \\ -3 & 0 \\ 0 & -1 \end{pmatrix}$. We assume for now that $\gamma = 4$. Then we construct $M_{\frac{1}{2}} = \begin{pmatrix} 12 & 0 \\ 0 & \frac{4}{3} \\ -12 & 0 \\ 0 & -\frac{4}{3} \end{pmatrix}$, which represents $K_{\frac{1}{2}}$. In Figure 8.2, we depict a sketch of both K_0 and $K_{\frac{1}{2}}$. We see that $K_{\frac{1}{2}}$ has only shrunk in size.

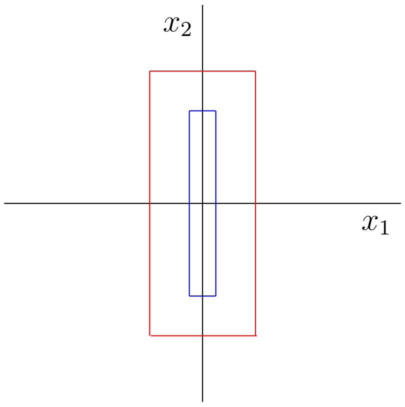

Figure 8.2: Sketch of K_0 and $K_{\frac{1}{2}}$, where they are represented by the red and blue lines respectively.

The next step in the algorithm is constructing K_1 with the use of the *Rack* function. This time, we construct K_1 step by step. Recall

$$K_1 = \left\{ x \in K_0 : \exists u \in \mathbb{R} \text{ such that } Ax + B_2 u \in K_{\frac{1}{2}} \right\}.$$

In words, for every value of $x \in K_0$, we compute Ax. Then we need to determine for what values of x there exists a $u \in \mathbb{R}$ such that $Ax + B_2 u \in K_{\frac{1}{2}}$. Therefore, our first step is to depict Ax for every $x \in K_0$, as done in Figure 8.3.

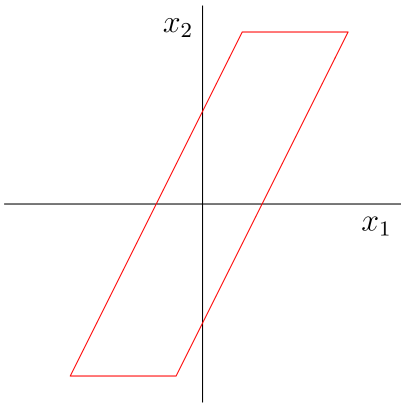

Figure 8.3: Sketch of Ax for every value of $x \in K_0$.

Now we check for what values of $u \in \mathbb{R}$ we have $Ax + B_2u \in K_{\frac{1}{2}}$. We depict it in Figure 8.4.

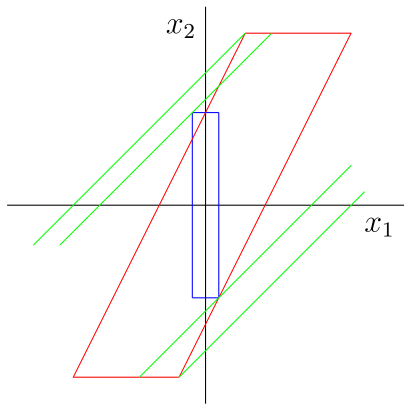

Figure 8.4: Sketch of Ax for every value of $x \in K_0$, $K_{\frac{1}{2}}$ and lines constructed by B_2u, represented by the red, blue and green lines respectively.

The green lines represent the possible ways that the controlled input can push the state vector to. Hence, every state x such that Ax lies between the two lines in the middle, can be controlled into $K_{\frac{1}{2}}$. The points that lie outside these lines cannot be controlled into $K_{\frac{1}{2}}$, as can be seen by following the top or bottom green line. As a result, we obtain the extra constraints for K_1, as depicted in Figure 8.5. In the algorithm, K_1 is described by the matrix $M_1 = \begin{pmatrix} 3 & 0 \\ 0 & 1 \\ -3 & 0 \\ 0 & -1 \\ \frac{6}{5} & \frac{6}{5} \\ -\frac{6}{5} & -\frac{6}{5} \end{pmatrix}$.

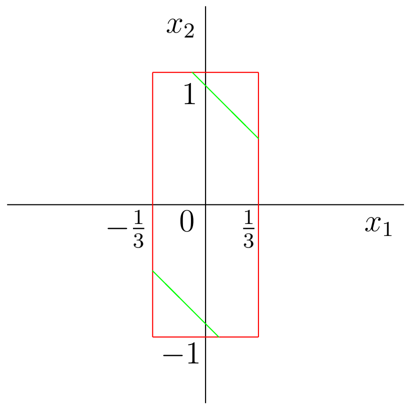

Figure 8.5: The boundary of K_1, where the red lines are based on K_0 and the green lines are the new constraints.

Following these steps in the same manner eventually results in one of two ways. Either the controlled invariant set, or there is some $K_{j\frac{1}{2}}$ which is empty.

8.2 Computational Times

In this section, we time the program for multiple cases to show where the weak points of our algorithm are. Using the same system as described in the previous section, we apply our algorithm with the stopping criterion described in chapter 6. By trial and error, we found that $\gamma_{optimal} = 6$ for this system before applying our algorithm. However, we will not anticipate on this, and simply apply our stated stopping criterion.

In case we choose a precision of 0.1, we end up with $\gamma = 6.0156$ after 0.48 seconds. When we choose a precision of 0.01, the algorithm ends after 6.67 seconds with $\gamma = 6.0059$. Based on this data, we can conclude that having a better precision is not necessarily beneficial. Note that γ is used to determine the maximum amplitude of the disturbance as $||w|| \leq \frac{1}{\gamma}$. Since the difference between these maximum amplitudes is approximately 0.003, we can conclude that it was not worth the extra 6 seconds to compute a controller that performs just a little bit better.

The reason that the algorithm is a lot slower when the precision is 0.01 is easily explained. When we try to compute a controlled invariant set with a value of γ that is just below $\gamma_{optimal}$, then the CIK algorithm makes a lot of steps before terminating. When the precision is chosen as 0.1, the last step before completing the algorithm takes up 94 loops. In the case of 0.01, the number of loops is significantly greater with 1535. Therefore, the precision must be chosen with caution. In Appendix B we have added simulation results using the constructed controllers for comparison in Figure 10.1 through 10.4.

Next we consider the system

$$\begin{aligned} x(k+1) &= \begin{pmatrix} -1 & 1 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} u(k), \\ z(k) &= \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} x(k) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ -1 \end{pmatrix} u(k), \end{aligned}$$

with initial condition $x(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$. With a precision of 0.1 we find that $\gamma = 9.921$ after 0.64 seconds. The output can be found in Figure 10.5 in Appendix B. With a precision of 0.01, we find $\gamma = 9.8926$ after only 0.72 seconds. As a result, we see that a smaller precision does not necessarily mean that the computational time is increased. In other words, in this case it may be beneficial to find a sharper bound in γ, since the computational time of the programm has not increased that much.

Note that the dimension of the state is increased in this case, but the computational time does not change by that much. In fact, the computational time in case the precision is 0.01 is decreased in the second example. However, when the dimension of both the state and the controlled input is increased much more than in the systems above, we are not able to compute a controlled invariant set anymore. Due to the fact that the *Rack* function increases the number of constraints every time it is applied, the program becomes slower. However, Matlab runs out of memory due to rounding errors. At some point, several constraints appear that actually should be the same. The only way to compute any result with larger systems is to do it manually, which would take a very long time. Another option is described in the next chapter.

Chapter 9

Industrial Application

The problem description of this application is given to us by David van den Hurk, MSc. This study is a lithographic application in which a wafer is exposed by a moving heat load. Due to this exposure, the wafer will absorb significant amounts of heat, which results in undesirable thermal stresses and deformations. To counteract these deformations, actuators are placed at certain locations below the wafer such that forces can be applied directly onto it. The aim of this study is to design an L_1-optimal controller which will generate optimal actuator forces to minimize the deformations induced by the scanning heat load. To slightly simplify the problem, the wafer is considered to be a one-dimensional beam, as shown in Figure 2.

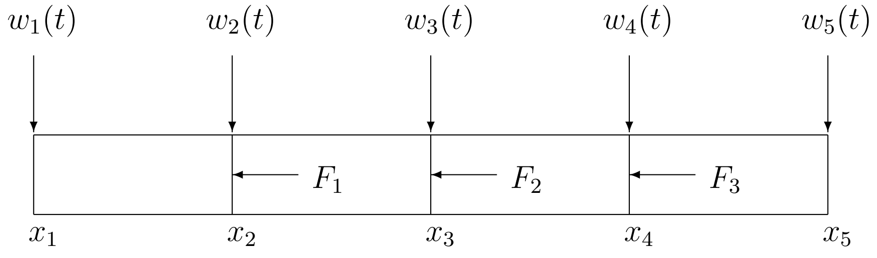

Figure 9.0: Problem setting.

The beam is discretized into a finite number of elements. At each boundary/node of these elements (x_1 through x_5), we can define state variables of the system, e.g. temperature or deformations. The scanning heat load is represented by individual heat loads $w_1(t)$ through $w_5(t)$ at every node of the discretized beam.

9.1 Lithographic Model

The beam is assumed to be a flexible structure, of a finite number of segments, with stiffness

$$k = \frac{EA}{L_i},$$

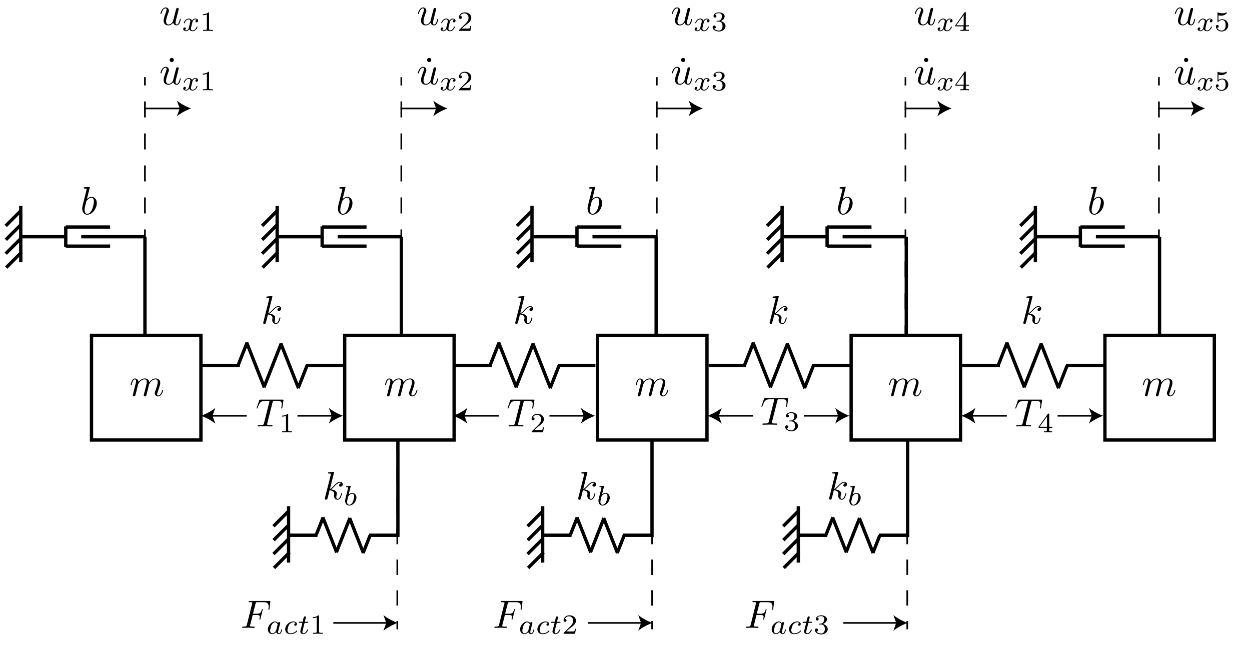

Figure 9.1: Mass spring damper representation of the beam.

where E is the Young's modulus of the material, A is the area of the cross-section of the beam, and L_i is the length of one segment. At the actuator nodes, a spring k_b is connected to the fixed world, and to prevent unwanted oscillations of the system, every node of the beam is connected to the fixed world using a damper with coefficient b. Additional inputs to the beam are actuator forces F_{act1} through F_{act3}, and the forces due to thermal expansion. The magnitude of these forces is equal to $EA\alpha T_i$, where α is the coefficient of thermal expansion (CTE) of the material and T_i is the average temperature increase of a beam segment between two nodes with respect to T_0 (typically $T_0 = 293.15$ [K]), i.e.,

$$T_i = \frac{T_{i_{\text{left}}} + T_{i_{\text{right}}}}{2},$$

with $T_{i_{\text{left}}}$ and $T_{i_{\text{right}}}$ the temperature at the left and right node of the beam segment. The direction of the force due to thermal expansion is then negative for the left node, and positive for the right node in case the material is expanding. A graphical representation of the described model is given in Figure 9.1, and the corresponding dynamics are derived as follows:

The dynamics of the system are based on Newton's second law,

$$\frac{d^2 u_x}{dt^2} = \frac{F_{internal}}{m} + EA\alpha\Delta T + \frac{F_{act}}{m},$$

where $F_{internal}$ are the internal forces on the nodes due to the damping, and elasticity of the beam, $F_{thermal}$ are the forces due to thermal expansion, and F_{act} are the actuator forces.

We can then write this for every node as,

$$\begin{aligned}
m\frac{d^2u_{x1}}{dt^2} &= k(u_{x2}-u_{x1}) - b\dot{u}_{x1} - EA\alpha T_1,\\
m\frac{d^2u_{x2}}{dt^2} &= -k(u_{x2}-u_{x1}) - k_b u_{x2} + k(u_{x3}-u_{x2}) - b\dot{u}_{x2} + EA\alpha T_1 - EA\alpha T_2 + F_{act1},\\
m\frac{d^2u_{x3}}{dt^2} &= -k(u_{x3}-u_{x2}) - k_b u_{x3} + k(u_{x4}-u_{x3}) - b\dot{u}_{x3} + EA\alpha T_2 - EA\alpha T_3 + F_{act2},\\
m\frac{d^2u_{x4}}{dt^2} &= -k(u_{x4}-u_{x3}) - k_b u_{x4} + k(u_{x5}-u_{x4}) - b\dot{u}_{x4} + EA\alpha T_3 - EA\alpha T_4 + F_{act3},\\
m\frac{d^2u_{x5}}{dt^2} &= -k(u_{x5}-u_{x4}) - b\dot{u}_{x5} + EA\alpha T_4.
\end{aligned}$$

To reduce the number of states in the state space description of this system, the temperature profiles, and thus forces due to thermal expansion, will be provided, and are determined by solving a separate model. Then the state space model is given with states $x = \begin{pmatrix} u_{x1} \\ \dot{u}_{x1} \\ \vdots \\ u_{xn} \\ \dot{u}_{xn} \end{pmatrix}$, outputs $z = \begin{pmatrix} u_{x1} \\ \vdots \\ u_{xn} \end{pmatrix}$, and disturbance $w = \begin{pmatrix} T_1 \\ \vdots \\ T_{n-1} \end{pmatrix}$, where $n = 5$, and three actuators $u = \begin{pmatrix} F_{act1} \\ \vdots \\ F_{actj} \end{pmatrix}$, where $j = 3$, of the form,

$$\begin{aligned}
\dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t),\\
z(t) &= Cx(t).
\end{aligned}$$

The matrices are defined as

$$A = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-\frac{k}{m} & -\frac{b}{m} & \frac{k}{m} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
\frac{k}{m} & 0 & -\frac{2k+k_b}{m} & -\frac{b}{m} & \frac{k}{m} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & \frac{k}{m} & 0 & -\frac{2k+k_b}{m} & -\frac{b}{m} & \frac{k}{m} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & \frac{k}{m} & 0 & -\frac{2k+k_b}{m} & -\frac{b}{m} & \frac{k}{m}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & \frac{k}{m} & 0 & -\frac{k}{m} & -\frac{b}{m}
\end{pmatrix},$$

$$B_1 = \frac{EA\alpha}{m}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad B_2 = \frac{1}{m}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

As stated in the introduction: The aim of this study is to design an L_1-optimal controller for the presented model which will generate optimal actuator forces to minimize the deformations induced by the scanning heat load/temperature profiles. Please refer to table below for the parameter values.

E	130e9 [Pa]
ρ	2329 [kg/m^3]
L_i	0.025 [m]
A	$0.775 \cdot 10^{-3}$[m^2]
k_b	0.5e7 [N/m]
b	1000 [Ns/m]
α	2.6e-6 [1/K]

Table 9.1: Model Parameters.

9.2 Numerical Results

The algorithms discussed in this paper are based on dynamical systems that are discrete in time. Therefore, in order to apply our algorithm, we need to discretize this system in time. There are multiple ways to discretize a system. The standard ways to discretize are the Forward and Backward Euler and the Central Difference methods. We choose to apply the Forward Euler method, as an example.

The method results in the discretized system

$$\begin{aligned} x(k+1) &= A_d x(k) + B_{1_d} w(k) + B_{2_d} u(k), \\ z(k) &= Cx(k), \end{aligned}$$

where the matrices are defined as

$$\begin{aligned} A_d &= I + hA, \\ B_{1_d} &= hB_1, \\ B_{2_d} &= hB_2, \end{aligned}$$

where h denotes the step size. We take $h = 0.001$, since it is reasonable for the given situation that the actuators can be adjusted after every thousandth of a second.

Next, we need to specify desired criteria. In this study, any deformation of the nodes is unwanted. However, we need to determine a criterion in order to apply one of the discussed algorithms. Therefore, we say that the system performs well if the deformation is not bigger than one nanometer, i.e., we want $||z|| \leq 10^{-9}$.

Since our algorithms construct a controlled invariant set such that $||w|| \leq \frac{1}{\gamma}$ and $||z|| \leq 1$, we need to rewrite our current system. This is done by a simple scaling argument.

We want to express z in nanometers. Therefore, we choose to substitute every 1 meter to 10^9 nanometers for every parameter. As a result, the matrices A_d, B_{1_d} and B_{2_d} alter and the state x is expressed in nanometers. Then we can choose to not change the matrix C_d, since the output z is then expressed in nanometers, i.e., the desired criterion is $||z|| \leq 1$.

Finding a controlled invariant set, given a fixed γ, for a system with this many dimensions takes a long time. In the algorithm designed to approach $\gamma_{optimal}$, a controlled invariant set is constructed for multiple values of γ. Hence, the time to compute the best possible controller for this system takes a very long time. However, it might not even be necessary to apply this algorithm, as we already know the maximal amplitude of the disturbance. We have $||w|| \leq 0.35349$, hence, $\gamma = 2.829$. These numbers are provided by David van den Hurk, MSc.

Notice that $dom(U_\gamma) = \{x \in \mathbb{R}^{10} : |Cx| \leq 1\}$ is an unbounded set for every $\gamma > 0$. As stated in Chapter 6, we choose a bounded set $K_0 = K \subset dom(U_\gamma)$ as defined in Definition 5.0.4, so we may construct $CINV(K)$. However, if we choose $K_0 = dom(U_\gamma)$, then the algorithm produces a bounded set K_1. Therefore we simply choose to construct $CINV(dom(U_\gamma))$.

For $\gamma = 0.2829$, the CIK algorithm terminates after K_1 is constructed. This is not surprising, considering the value of γ. We cannot base any conclusions on this result, besides that we are not able to maintain our performance for this amplitude of the disturbance. However, this is an exceptional situation where we can control the disturbance input. The heat load is applied by a laser, which we can switch off at any point in time. Since we know the temperature profile of the model, we have a lot more knowledge about this problem than we are able to use. It also means that we can search for a controlled invariant set with

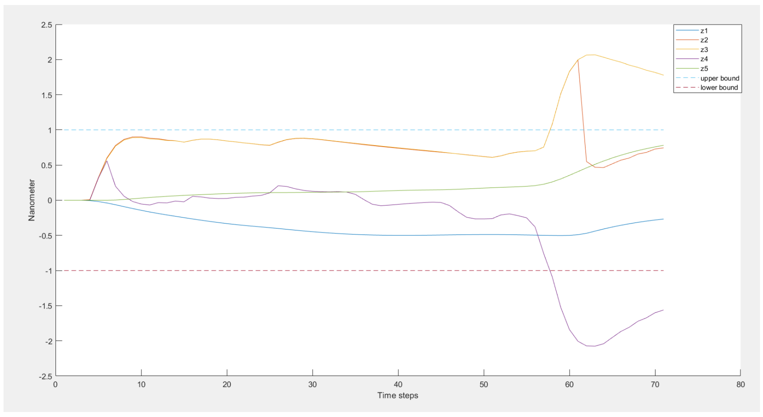

Figure 9.2: A graphical representation of the regulated output.

a lower disturbance amplitude. At the specific point in time where the heat load exceeds our desired amplitude, we simply switch the laser off, until the heat load has decreased.

Due to a lack of time, we are not able to determine any $\gamma > 0$ for which there exists a controlled invariant set. However, that does not imply that we have no result. For $\gamma = 30$ we are able to construct the set K_{11} (the final step before the algorithm terminates). Since our system has the initial value $x(0) = 0$, we know $x(0) \in K_{11}$. The physical interpretation of K_{11} is that for any value $x(k) \in K_{11}$ we can construct an input u such that $x(k + 11) \in dom(U_{30})$ when the worst possible values of the disturbance is applied. The set K_{11} is represented by the matrix $M_{11} \in \mathbb{R}^{20\times 10}$. Hence, the set K_{11} is an intersection of 20 halfspaces.

The algorithm produced another matrix $N \in \mathbb{R}^{20\times 13}$, which is needed in order to construct the controlled input. The matrix is defined as $N = M_{10\frac{1}{2}} \begin{pmatrix} A & B_2 \end{pmatrix}$. As a result, we have $M_{11} \in Rack^3(N)$. Based on the matrix N, we construct a controller as defined in chapter 6. The constructed controller ensures us that our performance is guaranteed for 0.011 seconds in the worst case scenario. Since the disturbance is known, and it is not the worst case, we expect that this controller succeeds in maintaining our desired performance for longer than 0.011 seconds.

Our expectations are met when applying a scaled heat load, as can be seen in Figure 9.2. 5 lines are depicted, representing the 5 regulated output components. To clearly show that the system satisfies our desired condition $||z|| \leq 1$ for over 50 steps, i.e., more than 0.05 seconds, we added two dashed lines, representing the upper and lower bound.
Therefore, we can conclude that we may give a recommendation to use this information in the following way. Since our constructed controller is discrete in time, we recommend that the L_1 controller is designed to adjust accordingly after every thousandth of a second, based on the constructed ℓ^1 controller. Since we measure the state constantly, we know the specific point in time when the system does not meet our desired condition anymore.

When this happens, we are able to temporarily shut off the laser, while the actuators remain active. According to Definition 2.2.1, the system is globally exponentially stable when the disturbance disappears. In other words, the amplitude of the output decreases. The laser can be switched back on again once the state has reached K_{11} again.

This specific case is not realistic in the sense that it is not beneficial to turn the laser on and off again approximately every 0.05 seconds. Therefore, more research needs to be done to find a more suitable controller based on the presented methods.

Chapter 10

Conclusion

In this thesis, a new algorithm to solve state-space systems with feedback control has been proposed. The constructed algorithm is applicable to all systems defined as (2.1)−(2.4), with a controllable pair (A, B_2). In order to prove the effectiveness of the algorithm, several properties of set-valued maps and invariance have been defined. Based on a paper of Jeff Shamma [10], the proposed algorithm involves several new ingredients leading to improved performances. One major advantage is that the required assumptions meet more realistic problems, while assuring the same accurate result. Moreover, the efficiency of the algorithm has been improved, thanks to the introduction of new features. Numerical tests have been successfully performed, including the application to an industrial problem.

The main ingredients of the proposed algorithm are summarized in the following.

- The matrices of a linear time-invariant system that is discrete in time,
- possible constraints on the controlled input,
- a tolerance.

When the algorithm starts running, a number of controlled invariant sets is constructed. After each successfully constructed controlled invariant set, we check if the stopping criterion is met. If it is, then the algorithm produces two matrices and a number γ as output. The first matrix is needed to compute the CIK, using Definition 6.0.2. The second matrix is required to construct a nonlinear continuous controller that is internally stabilizing with performance γ. Here γ is an approximation of $\gamma_{optimal}$, with a precision that is defined as input. When applying this feedback controller, the regulated output has a maximum amplitude of 1 or less. This holds for every kind of disturbance with a amplitude of $\frac{1}{\gamma}$ or less. In conclusion, the algorithm allows us to construct a near-optimal ℓ^1 controller.

10.1 Future Research

With the improvements made in the original algorithm of Shamma [10], there are still a lot of ways to make the algorithm run more efficient. In this section, we discuss what types of research can be done in order to improve the algorithm presented in this thesis.

10.1.1 New Stopping Criterion

We have defined a stopping criterion in chapter 7. In the algorithm we set $\delta = 10$. We keep dividing δ by 2 if a certain criterion is met, until δ is less than or equal to our desired precision. The value 10 is based purely on executed simulation results. It has no theoretical support of any kind. Besides that, this approach has a slow convergence. With the use of different optimization techniques, we should be able to develop a stopping criterion that is way more efficient and theoretically supported.

10.1.2 Redundancy Determination

One reason that the presented algorithm has a large computational time when the dimensions increase, is due to the way we determine which constraints are redundant. We currently use the command `convhulln`, because it is the fastest approach that we are aware of. However, it still takes a long time to compute the convex hull when the number of constraints is large. It might be possible to determine a faster approach. As a suggestion, we recommend that the properties of Lemma 5.0.5 are used as an advantage. For example, when we know that some row $M_{(i,:)}$ represents a redundant constraint, then we may also conclude that $-M_{(i,:)}$ is a redundant constraint, based on the symmetry with respect to the origin.

10.1.3 Constructing Controller

As Shamma has showed us in [10], it is possible to construct multiple controllers that have the same performance. In Chapter 6 we have stated that both controllers in Proposition 6.5.1 and any convex combination of them achieves a performance of γ. Even though our choice of $g(x) = \frac{1}{2}(\phi_+^M(x) + \phi_-^M(x))$ is the only one such that $g(0) = 0$, this does not mean that it outperforms any other convex combination of ϕ_+^M and ϕ_-^M. Besides, this is not the only way to construct controllers, as is shown in [9], where they use a scaling argument to obtain $g(0) = 0$. It is possible that these controllers outperform the previously stated controllers.

10.1.4 Implementing Knowledge about Disturbances

During the computation of any of the K_j and $K_{j\frac{1}{2}}$, the maximum amplitude is the only aspect taken into account, regarding the disturbances. However, we should be able to construct bigger sets K_j and $K_{j\frac{1}{2}}$ if we have some information about the disturbance at specific points in time.

Consider the following. Assume we have a compact set K_0. By definition, we have

$$K_{\frac{1}{2}} = \left\{ \xi \in \mathbb{R}^n : \xi + B_1 w \in K_0, \quad \forall |w| \leq \frac{1}{\gamma} \right\}.$$

The set $K_{\frac{1}{2}}$ is literally based on every possible value that the disturbance w could assume. If we know that w is going to be bounded within a certain set (so we use information besides the amplitude), then we would obtain a bigger set than $K_{\frac{1}{2}}$. For instance, suppose that we have a disturbance $w \in \mathbb{R}_+$. Then we obtain the set

$$\hat{K}_{\frac{1}{2}} = \left\{ \xi \in \mathbb{R}^n : \xi + B_1 w \in K_0, \quad \forall w \in [0, \frac{1}{\gamma}] \right\} \supset K_{\frac{1}{2}}.$$

However, the set $\hat{K}_{\frac{1}{2}}$ is not symmetric with respect to the origin.

We now propose another approach. Suppose that we have a disturbance $w \in \mathbb{R}^2$ and we know that $||w_1|| \leq \frac{1}{\gamma}$ and $||w_2|| \leq \frac{1}{5\gamma}$. Normally, we would construct $K_{\frac{1}{2}}$ as defined above. But we can apply this knowledge about w_2 to prevent a situation where $K_{\frac{1}{2}}$ is based on values of $w_2 > \frac{1}{5\gamma}$. Consider the following set:

$$\tilde{K}_{\frac{1}{2}} = \left\{ \xi \in \mathbb{R}^n : \xi + B_1 \begin{pmatrix} \frac{1}{\gamma} & 0 \\ 0 & \frac{1}{5\gamma} \end{pmatrix} w \in K_0, \quad \forall |w| \leq 1 \right\}.$$

This is an effective way to make use of the provided information, without losing any of the properties.

10.1.5 Different Performance Criteria

Throughout this thesis, we stated that we desired $||z|| \leq 1$, and wanted to find the maximum amplitude of the disturbance for which there still exists a controlled invariant set. However, in the practical application we encountered a different situation. Here we know the maximum amplitude of the disturbance and want to determine the smallest possible amplitude for which there exists a controlled invariant set.

This can be achieved by using the presented algorithms. If we apply one of the algorithms to our system, then we derive a controlled invariant set, which has the properties $||z|| \leq 1$ and $||w|| \leq \frac{1}{\gamma}$. Suppose that we know $||w|| = \beta$. Then we can simply multiply the whole system with a factor $\beta\gamma$, to obtain a controlled invariant set with the properties $||z|| \leq \beta\gamma$ and $||w|| \leq \beta$.

10.2 Recommendation

Because of its numerous applications, we started a research on ℓ^1 optimal control. It turns out that there is little information to be found about this subject using a state space approach. The most substantial research is done by Jeff Shamma in [10] and [9]. In this thesis, we have shown a few improvements on the algorithms that Shamma has discovered. However, given the possibilities to expand this research, we conclude that the full potential of ℓ^1 optimal control has not been discovered yet.

The most prominent subject, that the presented algorithms are based on, is viability theory. Therefore, we recommend that in order to continue this research, one has to have some basic knowledge about this subject. Note that we do not recommend that the continuation of this research must be based on viability theory alone. It is likely that further improvements on ℓ^1 optimal control are based on different fields within mathematics.

Bibliography

[1] J.P. Aubin, *Viability Theory.* Boston, MA: Birkhäuser, 1991.

[2] F. Blanchini, S. Miani, *Set-Theoretic Methods in Control.* Second ed., Systems & Control : Foundations & Applications, Birkhäuser: Cham, 2015.

[3] S.P. Boyd, L. Vandenberghe, *Convex Optimization.* Cambridge University Press: UK, 2004.

[4] M.A. Dahleh, J. Boyd Pearson, Minimization of a regulated response to a fixed input. *IEEE Transaction on automatic control*, Volume 33, Issue 10, October 1988, pages 924-930.

[5] J.C. Doyle, K. Glover, P.P. Khargonekar, B. Francis, State-space solutions to standard $\mathcal{H}^2$ and $\mathcal{H}^\infty$ control problems. *IEEE Transactions on Automatic Control*, Volume 34, Issue 8, August 1989, pages 831-847.

[6] H.A. Eiselt, C.L. Sandblom, *Linear Programming and Its Applications.* Springer: Berlin, 2007.

[7] T. Kailath, *Linear Systems.* Prentice-Hall Information and System Science Series: Englewood Cliffs, N.J., 1980.

[8] D. Repovš, P.V. Semenov, Ernest Michael and theory of continuous selections. *Topology and its Applications*, Volume 155, Issue 8, April 2008, pages 755-763.

[9] J.S. Shamma, Nonlinear state feedback for ℓ^1-optimal control. *Systems & Control Letters*, Volume 21, Issue 4, October 1993, pages 265-270.

[10] J.S. Shamma, Optimization of the l^∞-Induced Norm Under Full State Feedback. *IEEE Transactions on Automatic Control*, Volume 41, Issue 4, April 1996, pages 533-544.

[11] L.A. Wood, C.N. Scott, State-space approach to ℓ^1-optimal robust tracking. *Journal of Guidance, Control and Dynamics*, Volume 23, Issue 5, September 2000, pages 844-849.

Appendix A

Consider the setting of Theorem 5.0.1. We need a so-called selection theorem. Ernest Michael specialized in selection theorems. We use the following theorem of his.

Theorem 10.2.1. [8, Theorem 1]. *The following properties of T_1-space X are equivalent:*

- *X is paracompact; and*
- *If Y is a Banach space, then every lower semicontinuous set-valued map $\phi : X \to \mathcal{F}_c(Y)$ admits a single-valued continuous selection.*

$\mathcal{F}_c(Y)$ is defined as the family of all closed convex subsets of Y.

Here we show that we are allowed to apply this theorem to our set-valued map $R : CINV(K) \to \mathcal{P}(\mathbb{R}^m)$. We start at the beginning.

A T_1-space is a topological space where for any two distinct pairs, they both have a neighbourhood that does not contain the other. Note that $\mathbb{R}^n$ is obviously a T_1-space. And so is every subset of $\mathbb{R}^n$, in particular $CINV(K)$.

We need to show that $CINV(K)$ is a paracompact set. By definition, $CINV(K)$ is a closed and bounded set. By the Heine-Borel theorem, we know that $CINV(K)$ is compact. Since every compact space is paracompact, we satisfy the desired condition.

We have $Y = \mathbb{R}^m$. Since $\mathbb{R}^m$ is a Banach space, this part is not so exciting.

We have shown that R is lower semicontinuous in the proof of Theorem 5.0.1. But note that R is a standard set-valued map, while ϕ is defined differently. So the last question that needs to be answered is whether R is of the same form as ϕ?

We need $R(x) \subset \mathbb{R}^m$ to be a closed convex set for every $x \in CINV(K)$. In the proof of Theorem 5.0.1 it is shown that R satisfies these conditions. Hence, we are allowed to use this selection theorem.

Appendix B

We consider system A as

$$\begin{aligned} x(k+1) &= \begin{pmatrix} -1 & 1 \\ 0 & 2 \end{pmatrix} x(k) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(k), \\ z(k) &= \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} x(k), \end{aligned}$$

with initial condition $x(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. We consider system B as

$$\begin{aligned} x(k+1) &= \begin{pmatrix} -1 & 1 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} u(k), \\ z(k) &= \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} x(k) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} w(k) + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u(k), \end{aligned}$$

with initial condition $x(0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

In the following figures, we depicted the CIK or the output of system A or B, with a given precision 0.1 or 0.01. Every picture contains a pink set, which represents the constraints, and a blue line, which represents a simulation based on a random disturbance, with the property $||w|| \leq \frac{1}{\gamma}$. The values of γ, which is our approximation of $\gamma_{optimal}$, and the time it takes to compute these results are stated in Chapter 8. Note that CIK of system B is not depicted, because this is an unclear 3-dimensional figure.

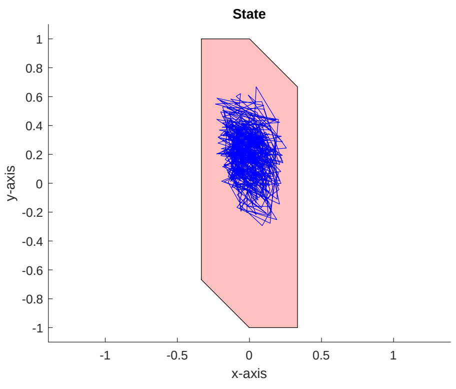

Figure 10.1: CIK of system A with precision 0.1.

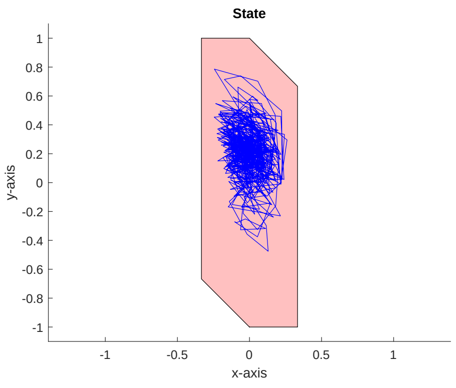

Figure 10.2: CIK of system A with precision 0.01.

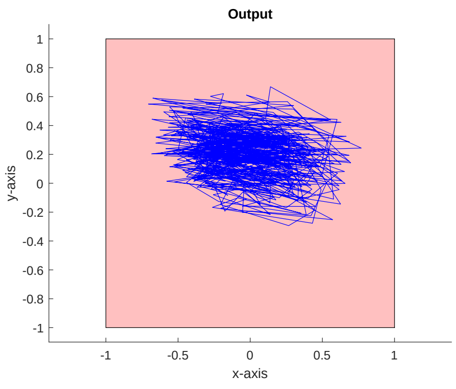

Figure 10.3: Output of system A with precision 0.1.

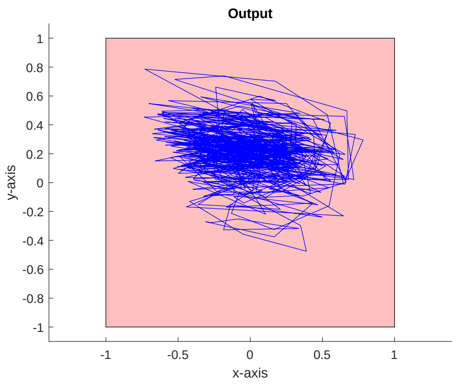

Figure 10.4: Output of system A with precision 0.01.

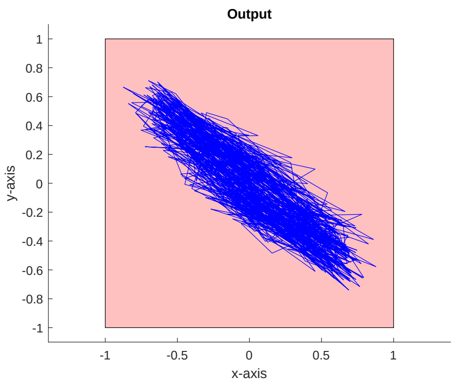

Figure 10.5: Output of system B with precision 0.1.

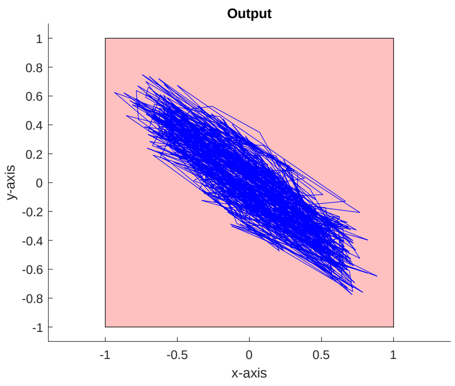

Figure 10.6: Output of system B with precision 0.01.