Automatic Dynamic Parallelotope Bundles for Reachability of Nonlinear Dynamical Systems

Edward Kim

A thesis submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science in the College of Arts and Science.

Chapel Hill
2022

Approved by:

Committee Member 1

Committee Member 2

Committee Member 3

Committee Member 4

Committee Member 5

Committee Member 6

ABSTRACT

Edward Kim: Automatic Dynamic Parallelotope Bundles for Reachability of Nonlinear Dynamical Systems
(Under the direction of Parasara Sridhar Duggirala)

Reachable set computation is an important technique for the verification of safety properties of dynamical systems. In this thesis, we investigate reachable set computation for discrete nonlinear systems based on parallelotope bundles. The crux of the reachability algorithm relies on computing an upper and lower bound on the supremum and infimum respectively of a nonlinear function over a rectangular domain. We cover two ways of computing these bounds: one method utilizing Bernstein polynomials and the other relying on a non-linear optimization tool developed by NASA, Kodiak. We aim to improve the traditional parallelotope-based reachability method by removing the manual step of parallelotope template selection in order to make the procedure fully automatic. Furthermore, we show that adding templates dynamically during computations can improve accuracy. To this end, we investigate two techniques for generating the template directions. The first technique approximates the dynamics as a linear transformation and generates templates using this linear transformation. The second technique uses Principal Component Analysis (PCA) of sample trajectories for generating templates. We have implemented our approach in a Python-based tool called Kaa. The tool is modular and use two types of global optimization solvers, the first using Bernstein polynomials and the second using the aforementioned Kodiak library. Additionally, we leverage the natural parallelism of the reachability algorithm and parallelize the Kaa implementation. Finally, we demonstrate the improved accuracy of our approach on several standard nonlinear benchmark systems, including a high-dimensional COVID19 model proposed by the Indian Supermodel Committee.

ACKNOWLEDGEMENTS

I would like to thank the UNC Computer Science community for supporting my work and providing me a warm community for pursuing this research. In particular, I would like to thank my advisor, Parasara Sridhar Duggirala, for the opporunity to perform research and participate in his group. Bineet Ghosh, Manish Goyal, Abel Karimi, and Meghan Stuart were all great colleagues and I enjoyed my time exchanging ideas, proofreading drafts, and socializing over group lunches. I would also like to thank Stanley Bak for helping me improve the implementation of Kaa and providing valuable feedback during this research. Juan Garcia made a wonderful collegue and an equally wonderful friend. The nights we raided Peabody and Phillips hall was certainly a memorable time. Finally, I would like to thank my parents and my family for their ardent, un-ending support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

One of the most widely-used techniques for performing safety analysis of non-linear dynamical systems is reachable set computation For example, reachability analysis has found many applications in formally verifying the safety properties of Cyber-physical Systems governed by Neural Network Controllers (Tran et al., 2019; Fan et al., 2020; Bak, 2021). The reachable set is defined to be the set of states visited by at least one of the trajectories of the system starting from an initial set and propagated forward in time by a finite fixed number of steps. Computing the exact reachable set for non-linear systems is challenging due to several reasons: First, unlike linear dynamical systems whose solutions can be expressed as closed form, non-linear dynamical systems generally do not admit such a nice form. Second, computationally speaking, current tools for performing non-linear reachability analysis are not very scalable. This is also in stark contrast to several scalable approaches developed for linear dynamical systems (Duggirala and Viswanathan, 2016; Bak and Duggirala, 2017). Finally, computing the reachable set using various set representations involves wrapping error which may be too conservative for practical use. That is, the overapproximation acquired at a given step would increase the conservativeness of the overapproximation for all future steps.

One of the several techniques for computing the overapproximation of reachable sets for discrete non-linear systems is to encode the reachable set through parallelotope bundles. Here, the reachable set is represented as a parallelotope bundle, an geometric data structure representing an intersection of several simpler objects called parallelotopes. One of the advantages of this technique is its exploitation of a special form of non-linear optimization problem to overapproximate the reachable set. The usage of a specific form of non-linear optimization mitigates many drawbacks involved with the scalability of non-linear analysis.

However, wrapping error still remains to be a problem for reachability using parallelotope bundles. An immediate reason stems from the responsbility of the practitioner to define the template directions specifiying the parallelotopes. Often, these template directions are selected to be either the cardinal axis directions or some directions from octahedral domains. However, it is not certain that the axis-aligned and octagonal directions are optimal for computing reachable sets over general non-linear dynamics. Additionally, even an expert user of reachable set computation tools may not be able to ascertain a suitable set of template directions for computing reasonably accurate over-approximations of the reachable set. Picking unsuitable template directions would only cause the wrapping error to grow, leading to the aforementioned issue of overly conservative reachable sets.

In this thesis, we investigate techniques for generating template directions automatically and dynamically, which is the culmination of several publications in different venues (Kim and Duggirala, 2020; Kim et al., 2021; Geretti et al., 2021). Specifically, we propose a method where instead of the user providing the template directions to define the parallelotope bundle, he or she specifies the number of templates whose template directions are to be generated by our algorithm automatically.

To this end, we study two techniques for generating the said template directions. First, we compute a local linear approximation of the non-linear dynamics and use the linear approximation to compute the template directions. Second, we generate a set of trajectories sampled from within the reachable set and use Principal Component Analysis (PCA) over these trajectories. We observe that the accuracy of the reachable set can be drastically improved by using templates generated using these two techniques. To address scalability, we demonstrate that even when the size of the initial set increases, our template generation algorithm returns more accurate reachable sets than both manually-specified and random template directions. Finally, we experiment with our dynamic template generation algorithm's effectiveness on approximating the reachable set of high-dimensional COVID19 dynamics proposed by the Indian Supermodel Committee (National Supermodel Committee , 2020). The results were published in an ACM blogpost detailing the utility of reachable set computation in modeling disease dynamics (Bak et al., 2021).

1.1 Related Work

Reachable set computation of non-linear systems using template polyhedra and Bernstein polynomials has been first proposed in (Dang and Salinas, 2009). In (Dang and Salinas, 2009), Bernstein polynomial representation is used to compute an upper bound of a special type of non-linear optimization problem. This enclosing property of Bernstein polynomials has been actively studied in the area of global optimization (Nataray and Kotecha, 2002; Garloff, 2003; Nataraj and Arounassalame, 2007). Furthermore, several heuristics have been proposed for improving the computational performance of optimization using Bernstein polynomials (Smith, 2009; Muñoz and Narkawicz, 2013).

Several improvements to this algorithm were suggested in (Dang and Testylier, 2012; Sassi et al., 2012) and (Dang et al., 2014) extends it for performing parameter synthesis. The representation of parallelotope bundles for reachability was proposed in (Dreossi et al., 2016) and the effectiveness of using bundles for reachability was demonstrated in (Dreossi, 2017; Dreossi et al., 2017). However, all of these papers used static template directions for computing the reachable set. In other words, the user must specify the template directions before the reachable set computation proceeds.

Using template directions for reachable set has been proposed in (Sankaranarayanan et al., 2008) and later improved in (Dang and Gawlitza, 2011). Leveraging the principal component analysis of sample trajectories for computing reachable set has been proposed in (Stursberg and Krogh, 2003; Chen and Ábrahám, 2011; Seladji, 2017). More recently, connections between optimal template directions for reachability of linear dynamical systems and bilinear programming have been highlighted in (Gronski et al., 2019). For static template directions, octahedral domain directions (Clarisó and Cortadella, 2004) remain a popular choice.

CHAPTER 2

Preliminaries

We begin with some basic definitions pertaining to reachability and parallelotopes. The definition of Bernstein polynomials and the reachable set computaion algorithm will be defined. Finally, an outline of the reachability algorithm given by (Dreossi et al., 2016) for polynomial dynamical systems will be presented.

2.1 Basic Definitions

As stated in the previous sections, this thesis pertains to the reachability analysis of dynamical systems. Roughly speaking, a dynamical system is governed by a set of differential equations such that the states of the system evolve according the solutions of the said differential equations. The state of a system, denoted as x, lies in a domain $D \subseteq \mathbb{R}^n$ where the solution to the differential equations is defined. We restrict our attention to a specific definition of these dynamical systems:

Definition 2.1. A discrete-time nonlinear system is denoted as

$$x^+ = f(x) \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function.

Intuitively, the function f takes input a state of the system and outputs the next step of the system evolved according to the non-linear dynamics. Here, the function f generally represents some discretized version of some specified continuous non-linear dynamical systems. Recall that a dynamical system is considered *linear* if its dynamics can be expressed as

$$x' = Ax, \quad A \in \mathbb{R}^{n \times n}$$

Otherwise, we deem the system to be *nonlinear*. Hence, in particular, the function f cannot be expressed as some matrix $A \in \mathbb{R}^{n \times n}$.

Examples of prominent non-linear dynamical systems include the Lotka-Volterra predator-prey model (Wangersky, 1978), FitzHugh Neuron model (FitzHugh, 1961), and recently introduced COVID19 disease model (National Supermodel Committee , 2020). Throughout this thesis, we discretize any continuous dynamics through the well-known Euler method. Thus, up to some error term of bounded degree, we can turn any non-linear system into the form given by Equation 2.1.

Example 2.1. The SIR Epidemic model is a 3-dimensional dynamical system governed by the following continuous dynamics:

$$\begin{aligned} s' &= \beta \cdot s_k i_k \\ i' &= \beta \cdot s_k i_k - \gamma \cdot i_k \\ r' &= \gamma \cdot i_k \end{aligned} \tag{2.2}$$

where s, i, r represent the fractions of a population of individuals designated as *susceptible*, *infected*, and *recovered* respectively. There are two parameters, namely β and γ, which influence the evolution of the system. β is labeled as the contraction rate and $1/\gamma$ is the mean infective period. Discretizing Equation 2.2 according the Euler method yields the dynamics:

$$\begin{aligned} s_{k+1} &= s_k - (\beta \cdot s_k i_k) \cdot \Delta \\ i_{k+1} &= i_k + (\beta \cdot s_k i_k - \gamma \cdot i_k) \cdot \Delta \\ r_{k+1} &= r_k + (\gamma \cdot i_k) \cdot \Delta \end{aligned} \tag{2.3}$$

Here, Δ is the discretization step and the index $k \in \mathbb{N}$ simply represents the current step. Note the non-linear terms $s_k i_k$ which precludes the expression of the dynamics as a linear transformation. $\Diamond$

The trajectory of a system that evolves according to Equation 2.1, denoted as $\xi(x_0)$ is a sequence $x_0, x_1, \ldots$ where $x_{i+1} = f(x_i)$. The k^{th} element in this sequence x_k is denoted as $\xi(x_0, k)$.

Definition 2.2. Given an initial set $\Theta \subseteq \mathbb{R}^n$, the *reachable set at step k*, denoted as Θ_k is defined as

$$\Theta_k = \{\xi(x,k) \mid x \in \Theta\} \tag{2.4}$$

If we set the number of steps to be some $n \in \mathbb{N}$, we say the *reachable set* is

$$\Theta = \bigcup_{i=1}^{n} \Theta_i \tag{2.5}$$

We will see in a future section an example of the reachable set of the discretized SIR model presented in Equation 2.3.

2.2 Parallelotope-based Reachability

2.2.1 Parallelotopes

A parallelotope P is a set of states in $\mathbb{R}^n$ captured by the tuple $\langle \Lambda, c\rangle$ where $\Lambda \in \mathbb{R}^{2n\times n}$ is a matrix and ç is a column vector. We impose the condition that $\Lambda_{i+n} = -\Lambda_i$ for all $i \in \{1,\dots,n\}$ such that

$$x \in P \text{ if and only if } \Lambda x \leq c. \tag{2.6}$$

We deem Λ as the *template direction matrix* where Λ_i denotes the i^{th} row of Λ called the i^{th} *template direction*. The column vector c is called the *offset vector* with c_i denoting the i^{th} element of c. If we unpack Equation 2.6, we can re-express the inequalities as a conjection of half-space constraints. If we define $c_u = [c_1, c_2, \cdots, c_n]^T$ and $c_l = [c_{n+1}, c_{n+2}, \cdots, c_{2n}]^T$, then Equation 2.6 tells us that:

$$\Lambda_i x \leq (c_u)_i$$
$$-\Lambda_i x \leq (c_l)_i$$

Additionally, the definition of the paralleotope above requires that for each of n "postive" directions, there must exist a corresponding "negative" direction. This is encoded into the template matrix Λ by the condition $\Lambda_{i+n} = -\Lambda_i$. However, by the observation made above, we only need to keep the positive directions and divide our offset vector into equal components with the top half encoding the offsets for the positive directions and the bottom half encoding the offsets for the negative directions. Combining these remarks yields the *half-space representation* of parallelotope P.

Definition 2.3. The half-space representation of parallelotope P is tuple $\langle \Lambda, c_l, c_u \rangle$ where $\Lambda \in \mathbb{R}^{n \times n}$ and $c_l, c_u \in \mathbb{R}^n$ such that

$$P = \{x \mid c_l \leq \Lambda x \leq c_u\} \tag{2.7}$$

Example 2.2. Consider the 2D plane, namely $\mathbb{R}^2$. We can construct a couple of simple examples of parallelotopes. First, if we define our parallelotope's template direction matrix to be $\Diamond$

Alternatively, a parallelotope can also be represented in a *generator representation*

Definition 2.4. The generator representation of a parallelotope P is a tuple of vectors $\langle v, g_1, \dots, g_n \rangle$ such that $v, g_1, \cdots g_n \in \mathbb{R}^n$. The vector $v \in \mathbb{R}^n$ is called the *anchor* and the $g_i \in \mathbb{R}^n$, are called the *generators*. The parallelotope is defined as the set:

$$P := \{x \mid \exists \alpha_1, \dots, \alpha_n \in [0,1],\ x = v + \sum_{i=1}^{n} \alpha_i g_i\}$$

This representation is very similar to Zonotopes (Girard, 2005; Althoff et al., 2010) and Star sets (Duggirala and Viswanathan, 2016). In particular, a parallelotope is a special case of a zonotope where the number of generators is exactly the dimension of the system n.

There is a simple method to convert from the half-space representation of P to its generator representation. Obtain vertex v_1 by solving the linear equation $\Lambda x = c_l$. The $j+1$ vertex is obtained by solving the linear equation $\Lambda x = \mu_j$ where $\mu_j[i] = c_l[i]$ when $i \neq j$ and $\mu_j[j] = c_u[j]$. The anchor v of the parallelotope is the vertex v_1 and the generators will be $g_i = v_{i+1} - v_1$. Refer to (Dang et al., 2014) for a more detailed exposition of this procedure.

As a final remark, notice that for a parallelotope P, the generator representation also defines an affine transformation that maps $[0,1]^n$ to P. Let us denote this affine transformation associated to P as T_P.

Example 2.3. Once again, consider the space $\mathbb{R}^2$ and the parallelotope P given in half-plane representation as $0 \leq x - y \leq 1, 0 \leq y \leq 1$. This is a parallelotope with vertices at $(0,0)$, $(1,0)$, $(2,1)$, and $(1,1)$. In the half-space representation, the template directions of the parallelotope P are given by the directions $[1, -1]$ and $[0, 1]$. The half-space representation in matrix form is given as follows:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{2.8}$$

To compute the generator representation of P, we need to compute the *anchor* and the *generators*. The anchor is obtained by solving the linear equations $x - y = 0, y = 0$. Therefore, the anchor a is the vertex at origin $(0, 0)$ To compute the two generators of the parallelotope, we compute two vertices of the parallelotope. Vertex v_1 is obtained by solving the linear equations $x - y = 1, y = 0$. Therefore, vertex v_1 is the vertex $(1, 0)$. Similarly, vertex v_2 is obtained by solving the linear equations $x - y = 0, y = 1$. Therefore, v_2 is the vertex $(1, 1)$. The generator g_1 is the vector $v_1 - a$, that is $(1, 0) - (0, 0) = (1, 0)$ The generator g_2 is the vector $v_2 - a$, that is $(1, 1) - (0, 0) = (1, 1)$. Therefore, all the points in the paralellotope can be written as $(x, y) = (0, 0) + \alpha_1(1, 0) + \alpha_2(1, 1)$, $\alpha_1, \alpha_2 \in [0, 1]$. $\Diamond$

Definition 2.5. A parallelotope bundle Q is a set of parallelotopes $\{P_0, \ldots, P_m\}$ such that

$$Q = \bigcap_{i=1}^{m} P_i$$

.

Remark 2.1. There is a slight abuse of notation above where we refer to the parallelotope bundle Q as both the set of parallelotopes and the region in $\mathbb{R}^n$ of the intersection of all the parallelotopes P_i. To specify the *set of parallelotopes* which consists the bundle, we will write

$$\mathcal{P}(Q) = \{P_0, \ldots, P_m\}$$

This parallelotope bundle will be the geometric data structure which will enclose the region we compute to be the approximation of the exact reachable set.

2.2.2 Bernstein Polynomials

In this section, we define Bernstein polynomials and state some of their enclosure properties. A *multi-index* $\mathbf{i}$ of length n is defined as tuple of n elements $\mathbf{i} = (i_1, \cdots, i_n)$ such that each $i_k \in \mathbb{N}$. Furthermore, we order the multi-indices as follows: if $\mathbf{i}$ and $\mathbf{j}$ are two multi-indices of length n, then

$$\mathbf{i} \leq \mathbf{j} \iff i_k \leq j_k, \quad 1 \leq k \leq n$$

Finally, we generalize the product of binomial coefficients as:

$$\binom{\mathbf{i}}{\mathbf{j}} := \prod_{k=1}^{n} \binom{i_k}{j_k}$$

Given two multi-indices $\mathbf{i}$ and $\mathbf{d}$ of size n, where $\mathbf{i} \leq \mathbf{d}$, the Bernstein basis polynomial of degree $\mathbf{d}$ and index $\mathbf{i}$ is defined as:

$$\mathcal{B}_{(\mathbf{i},\mathbf{d})}(\mathbf{x}) = \beta_{i_1,d_1}(x_1)\beta_{i_2,d_2}(x_2)\ldots\beta_{i_n,d_n}(x_n). \tag{2.9}$$

where for $i, d, x \in \mathbb{R}$:

$$\beta_{i,d}(x) = \binom{d}{i} x^i (1-x)^{d-i} \tag{2.10}$$

Let $p : \mathbb{R}^n \to \mathbb{R}$ of be a real polynomial of degree at most $\mathbf{d}$. We can express p as a linear combination of monomials of degree at most $\mathbf{d}$:

$$p(\mathbf{x}) = \sum_{\mathbf{i} \leq \mathbf{d}} a_{\mathbf{i}} \cdot \mathbf{x}^{\mathbf{i}}$$

where $\mathbf{x}^{\mathbf{i}}$ represents the monomial $x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$. Every such real polynomial p can be represented as linear combination of Bernstein basis polynomials of of degree $\mathbf{d}$:

$$p(\mathbf{x}) = \sum_{\mathbf{i} \leq \mathbf{d}} b_{\mathbf{i}} \cdot \mathcal{B}_{(\mathbf{i},\mathbf{d})}(\mathbf{x}) \tag{2.11}$$

where $b_{\mathbf{i}}$ denotes the $\mathbf{i}^{th}$ *Bernstein Coefficient*:

$$b_{\mathbf{i}} = \sum_{\mathbf{j} \leq \mathbf{i}} \frac{\binom{\mathbf{i}}{\mathbf{j}}}{\binom{\mathbf{d}}{\mathbf{j}}} \cdot a_{\mathbf{j}} \tag{2.12}$$

In other words, given a polynomial $p(x_1, \ldots, x_n) = \sum_{j \in J} a_j \mathbf{x}_j$ where J is a set of multi-indices iterating through the degrees found in p with $a_j \in \mathbb{R}$, then $p(x_1, \ldots, x_n)$ can be converted into its counterpart under the Bernstein basis, $p(x_1, \ldots, x_n) = \sum_{j \in J} b_j \mathcal{B}_j$ where b_j are the corresponding Bernstein coefficients.

The primary advantage of the Bernstein representation of a polynomial $p(x_1, ..., x_n)$ is that an upper bound on the supremum and lower bound on the infimum of $p(x_1, ..., x_n)$ in $[0, 1]^n$ can be computed purely by observing the coefficients of the polynomial in the Bernstein basis. Specifically, the upper and lower bounds of $p(x_1, \ldots, x_n)$ over $[0, 1]^n$ are bounded by the Bernstein coefficients. We state this as a property without proof.

Property 2.1. (Enclosure Property) Let $p : \mathbb{R}^n \to \mathbb{R}$ be a real mutlivarite polynomial of degree $\mathbf{d}$, and let $p(\mathbf{x}) = \sum_{\mathbf{i} \leq \mathbf{d}} b_{\mathbf{i}} \cdot \mathcal{B}_{(\mathbf{i},\mathbf{d})}(\mathbf{x})$ be the Bernstein expansion of p, then

$$\min_{\mathbf{i} \leq \mathbf{d}} \{b_{\mathbf{i}}\} \leq \inf_{x \in [0,1]^n} p(x) \leq \sup_{x \in [0,1]^n} p(x) \leq \max_{\mathbf{i} \leq \mathbf{d}} \{b_{\mathbf{i}}\}$$

As mentioned earlier, a parallelotope P can also be represented as an affine transformation T_p from $[0, 1]^n$ to P. Therefore, upper bounds on the suprenum of a polynomial function p over P is equivalent to upper bound of $p \circ T_p$ over $[0, 1]^n$. A similar argument follows for the lower bound on the infimum. The crux of the reachability algorithm involves exploiting this property of Bernstein polynomials to approximate the solution of certain non-linear optimization problem invovilng polynomial predicates over the unitbox, $[0, 1]^n$. We will cover this algorithm in the upcoming section. For a more rigorous exposition on Bernstein polynomials and Property 2.1, refer to (Garloff, 2003).

2.2.3 The Static Algorithm

We will end with an outline of the static algorithm first investigated in works (Dang and Testylier, 2012; Dreossi et al., 2016). As mentioned in the previous section, the building block of the reachability algorithm relies on approximate solutions to a non-linear optimization problem over the unitbox domain. Consider a nonlinear function $h : \mathbb{R}^n \to \mathbb{R}$. The most general form of this optimization problem can be expressed as:

$$\begin{aligned} &\max\ h(x) \\ &s.t.\ x \in [0,1]^n. \end{aligned} \tag{2.13}$$

In the static algorithm, the user manually specifies the number of parallelotopes and a set of static directions for each parallelotope. In other words, the user must specify the template matrix Λ and its corresponding offset vector c for each parallelotope $P = \langle \Lambda, c \rangle$ contained in the bundle *before* the computation begins.

We now proceed to formally describe the static algorithm. First, a small remark on the template matrix of the parallelotopes P_i contained in some bundle Q. It is possible that some of the parallelotopes share the same template matrix directions. In other words, for $P_i = \langle \Lambda^{P_i}, c^{P_i} \rangle$, $P_j = \langle \Lambda^{P_j}, c^{P_j} \rangle$ such that $P_i, P_j \in \mathcal{P}(Q)$, there could exist some k such that $\Lambda^{P_i}_k = \Lambda^{P_j}_k$ as row vectors. Thus, a more compact method of encoding the bundle is by taking the *distinct* template directions as rows of a new template matrix Λ^Q along with its corresponding offset vector c^Q. To distinguish between the distinct parallelotopes contained in the bundle, we add a new matrix called $\mathcal{T}^Q \in \mathbb{N}^{p \times n}$ such that $\mathcal{T}^Q_i$ is a vector of row indices of Λ^Q which specify the template directions defining parallelotope $P_i \in \mathcal{P}(Q)$.

Remark 2.2. If none of the paralleotopes $P_i \in \mathcal{P}(Q)$ share common template directions, then Λ^Q will simply be the template direction matrices $\{\Lambda^P\}_{P \in \mathcal{P}(Q)}$ concatented along their rows. This will generally be the matrix generated by the dynamic algorithm we will outline in a future section.

Example 2.4. $\diamondsuit$

Another input to the algorithm is the initial set, given as a parallelotope P_0. When the initial set is a box, P_0 will be defined by the axis-aligned template directions.

The output of the algorithm is, for each step k, the set $\overline{\Theta}_k$, which is an overapproximation of the reachable set at step k, $\Theta_k \subseteq \overline{\Theta}_k$. The total overapproximation of the reachable set for a finite number of steps n will be $\Theta = \cup_{k=1}^{n} \Theta_k$. The high-level pseudo-code is written in Algorithm **??**.

The algorithm simply calls `TransformBundle` for each step, producing a new parallelotope bundle computed from the previous step's bundle. To compute the image of Q, the algorithm computes the upper and lower bounds of $f(x)$ with respect to each template direction Λ^Q_i. Since computing the maximum value of $f(x)$ along each template direction on the Q in one-shot is computationally difficult, the algorithm instead computes the maximum value over each of the constituent parallelotopes and uses the minimum of all these maximum values.

The `TransformBundle` operation works as follows. Consider a parallelotope P in the bundle Q. Given a template direction Λ_i^Q, the maximum value of $\Lambda_i^Q f(x)$ for all $x \in Q$ is less than or equal to the maximum value of $\Lambda_i^P \cdot f(x)$ for all $x \in P$ such that Λ_i^Q is a row in Λ^P. Similar argument holds for the minimum value of $\Lambda_i^Q \cdot f(x)$ for all $x \in Q$. Observe that these inequalities hold by virtue of the fact that $Q \subseteq P$ by definition. To describe this more formally: if $\lambda_i^Q = \{P \in \mathcal{P}(Q) \mid \Lambda_i^Q = \Lambda_k^P \text{ for some } k\}$, then

$$\max_{x \in Q} \Lambda_i^Q \cdot f(x) \leq \min_{P \in \lambda_i^Q} \max_{x \in P} \Lambda_i^P \cdot f(x) \tag{2.14}$$

$$\min_{P \in \lambda_i^Q} \min_{x \in P} \Lambda_i^P \cdot f(x) \leq \min_{x \in Q} \Lambda_i^Q \cdot f(x) \tag{2.15}$$

To compute the upper and lower bounds of each template direction $\Lambda_i f(x)$, for all $x \in P$, we perform the following optimization.

$$\begin{aligned} \max\ & \Lambda_i^P \cdot f(x) \\ s.t.\ & x \in P. \end{aligned} \tag{2.16}$$

Note that $\Lambda_i^P \cdot f(x)$ is a dot product between the row vector Λ_i^P and the component-wise dynamics of $f(x)$. This is similar to the method of computing support functions over convex sets (Boyd et al., 2004).

Given that P is a parallelotope, all the states in P can be expressed as a vector summation of anchor and scaled generators. Let $\langle v, g_1, \cdots, g_n \rangle$ be the generator representation of P. The optimization problem given in Equation 2.16 would then transform as follows.

$$\begin{aligned} \max\ & \Lambda_i^P \cdot f(a + \Sigma_{i=1}^n \alpha_i g_i) \\ s.t.\ & \overline{\alpha} \in [0,1]^n. \end{aligned} \tag{2.17}$$

Equation 2.17 is a form of $\mathsf{optBox}(\Lambda_\mathsf{i} \cdot \mathsf{f})$ over $[0,1]^n$. One can compute an upper-bound to this nonlinear optimization by computing the Bernstein coefficients of $\Lambda_i \cdot f(a + \Sigma_{i=1}^n \alpha_i g_i)$ and taking the maximum and minimum coefficients as shown in Property 2.1. Similarly, we compute the lower-bound of $\Lambda_i \cdot f(x)$ for all $x \in P$ by computing the upperbound of $-1 \times \Lambda_i \cdot f(x)$.

We iterate this process (i.e., computing the upper and lower bound of $\Lambda_i^Q \cdot f(x)$) for each parallelotope in the bundle Q according to Equation 2.14 and Equation 2.15). Therefore, the tightest upper bound on $\Lambda_i^Q \cdot f(x)$ over Q is the least of the upper bounds computed from each of the parallelotopes. A similar argument holds for lower bounds of $\Lambda_i^Q \cdot f(x)$ over Q. Therefore, the image of the bundle Q will be the bundle Q' where the upper and lower bounds for templates directions are obtained by solving a series of nonlinear optimization problems of the form presented in Equation 2.16.

Finally, once the loop on step two of Algorithm **??** halts at step S, the outputted reachable set will be the computed over-approximations $\overline{\Theta}_1, \cdots, \overline{\Theta}_S$. As step four within the loop implies, this is simply the image bundles Q' returned by our `TransformBundle` procedure.

Example 2.5. We return to the SIR model briefly treated in Section 2.1. Figure **??** shows the reachable set computed with the static algorithm and plotted using the following parameters:

- The parameters of the model are set to $\beta = 0.34$ and $\gamma = 0.05$. The discretization step is set to $\Delta = 0.1$.

- The parallelotope only has one static parallelotope, namely the initial box. This shows that our template matrix for P is

$$\Lambda^Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathcal{T}^Q = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$$

$$c_l^P = \begin{bmatrix} -0.79 & -0.19 & 0 \end{bmatrix}^T \quad c_u^P = \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix}^T$$

- We set the number of time steps $S = 300$.

There a few points worth noting here. First, by the discussion leading to Definition 2.3, the initial set would be the box $[0.79, 0.8] \times [0.19, 0.2] \times 0$. This can be interpreted as initializing the model such that $79 - 80\%$ of the population is susceptible (not yet infected) with $19 - 20\%$ of the population is infected. As the simulation is beginning, no percentage of the population has recovered from the disease. Hence, the third parameter r is set to zero.

Second, since we only have the axis-aligned parallelotope in our initial bundle, the matrix $\mathcal{T}^Q$ will consist of only one row indicing the axis-aligned directions expressed as distinct rows in Λ^Q.

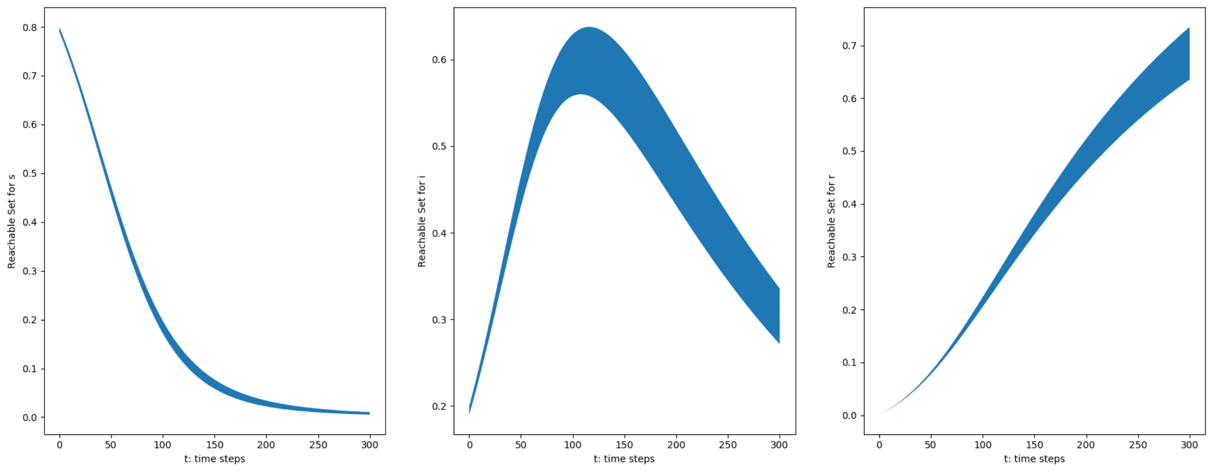

Figure 2.1: Projection of Reachable Set of SIR propagated 300 steps in time.

$\Diamond$

2.2.4 Optimization Modes

CHAPTER 3

Dynamic Paralleotope Bundles

3.0.1 Local Linear Approximations

3.0.2 Principal Component Analysis

3.0.3 The Dynamic Algorithm

CHAPTER 4

Experimental Results

4.0.1 Kaa

4.0.2 Benchmarks

4.0.3 COVID19 Supermodel

4.0.4 Comparison of Template Generation Techniques

BIBLIOGRAPHY

Althoff, M., Stursberg, O., and Buss, M. (2010). Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear analysis: hybrid systems*, 4(2):233–249.

Bak, S. (2021). nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pages 19–36. Springer.

Bak, S. and Duggirala, P. S. (2017). Simulation-equivalent reachability of large linear systems with inputs. In *International Conference on Computer Aided Verification*, pages 401–420. Springer.

Bak, S., Kim, E., and Duggirala, P. S. (2021). Covid infection prediction using cps formal verification methods.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Chen, X. and Ábrahám, E. (2011). Choice of directions for the approximation of reachable sets for hybrid systems. In *International Conference on Computer Aided Systems Theory*, pages 535–542. Springer.

Clarisó, R. and Cortadella, J. (2004). The octahedron abstract domain. In *International Static Analysis Symposium*, pages 312–327. Springer.

Dang, T., Dreossi, T., and Piazza, C. (2014). Parameter synthesis using parallelotopic enclosure and applications to epidemic models. In *International Workshop on Hybrid Systems Biology*, pages 67–82. Springer.

Dang, T. and Gawlitza, T. M. (2011). Template-based unbounded time verification of affine hybrid automata. In *Asian Symposium on Programming Languages and Systems*, pages 34–49. Springer.

Dang, T. and Salinas, D. (2009). Image computation for polynomial dynamical systems using the Bernstein expansion. In *International Conference on Computer Aided Verification*, pages 219–232. Springer.

Dang, T. and Testylier, R. (2012). Reachability analysis for polynomial dynamical systems using the Bernstein expansion. *Reliable Computing*, 17(2):128–152.

Dreossi, T. (2017). Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 29–34.

Dreossi, T., Dang, T., and Piazza, C. (2016). Parallelotope bundles for polynomial reachability. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 297–306.

Dreossi, T., Dang, T., and Piazza, C. (2017). Reachability computation for polynomial dynamical systems. *Formal Methods in System Design*, 50(1):1–38.

Duggirala, P. S. and Viswanathan, M. (2016). Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*, pages 477–494. Springer.

Fan, J., Huang, C., Chen, X., Li, W., and Zhu, Q. (2020). Reachnn*: A tool for reachability analysis of neural-network controlled systems. In *International Symposium on Automated Technology for Verification and Analysis*, pages 537–542. Springer.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466.

Garloff, J. (2003). The Bernstein expansion and its applications. *Journal of the American Romanian Academy*, 25:27.

Geretti, L., Althoff, M., Benet, L., Chapoutot, A., Collins, P., Duggirala, P. S., Forets, M., Kim, E., Linares, U., et al. (2021). Arch-comp21 category report: Continuous and hybrid systems with nonlinear dynamics.

Girard, A. (2005). Reachability of uncertain linear systems using zonotopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 291–305. Springer.

Gronski, J., Sassi, M.-A. B., Becker, S., and Sankaranarayanan, S. (2019). Template polyhedra and bilinear optimization. *Formal Methods in System Design*, 54(1):27–63.

Kim, E., Bak, S., and Duggirala, P. S. (2021). Automatic dynamic parallelotope bundles for reachability analysis of nonlinear systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 50–66. Springer.

Kim, E. and Duggirala, P. S. (2020). Kaa: A Python implementation of reachable set computation using Bernstein polynomials. *EPiC Series in Computing*, 74:184–196.

Muñoz, C. and Narkawicz, A. (2013). Formalization of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196.

Nataraj, P. S. and Arounassalame, M. (2007). A new subdivision algorithm for the Bernstein polynomial approach to global optimization. *International journal of automation and computing*, 4(4):342–352.

Nataray, P. and Kotecha, K. (2002). An algorithm for global optimization using the Taylor–Bernstein form as inclusion function. *Journal of Global Optimization*, 24(4):417–436.

National Supermodel Committee (2020). Indian Supermodel for Covid-19 Pandemic.

Sankaranarayanan, S., Dang, T., and Ivančić, F. (2008). Symbolic model checking of hybrid systems using template polyhedra. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 188–202. Springer.

Sassi, M. A. B., Testylier, R., Dang, T., and Girard, A. (2012). Reachability analysis of polynomial systems using linear programming relaxations. In *International Symposium on Automated Technology for Verification and Analysis*, pages 137–151. Springer.

Seladji, Y. (2017). Finding relevant templates via the principal component analysis. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 483–499. Springer.

Smith, A. P. (2009). Fast construction of constant bound functions for sparse polynomials. *Journal of Global Optimization*, 43(2-3):445–458.

Stursberg, O. and Krogh, B. H. (2003). Efficient representation and computation of reachable sets for hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 482–497. Springer.

Tran, H.-D., Manzanas Lopez, D., Musau, P., Yang, X., Nguyen, L. V., Xiang, W., and Johnson, T. T. (2019). Star-based reachability analysis of deep neural networks. In *International symposium on formal methods*, pages 670–686. Springer.

Wangersky, P. J. (1978). Lotka-volterra population models. *Annual Review of Ecology and Systematics*, 9:189–218.