Incremental Affine Abstraction of Nonlinear Systems

Syed M. Hassaan, Mohammad Khajenejad, Spencer Jensen, Qiang Shen and Sze Zheng Yong

***Abstract*—In this paper, we propose an incremental abstraction method for dynamically over-approximating nonlinear systems in a bounded domain by solving a sequence of linear programs, resulting in a sequence of affine upper and lower hyperplanes with expanding operating regions. Although the affine abstraction problem can be solved using a single linear program, existing approaches suffer from a computation space complexity that grows exponentially with the state dimension. Thus, the motivation for incremental abstraction is to reduce the space complexity of abstraction algorithms for high-dimensional systems or systems with limited on-board resources. Specifically, we start with an operating region that is a subregion of the state space and compute a pair of affine hyperplanes that bracket the nonlinear function locally. Then, by incrementally expanding the operating region, we dynamically update the two affine hyperplanes such that we eventually yield hyperplanes that are guaranteed to over-approximate the nonlinear system over the entire domain. Finally, the effectiveness of the proposed approach is demonstrated using several numerical examples.**

***Index Terms*—Computational methods, Large-scale systems, Optimization**

I. Introduction

ABSTRACTION-BASED methods for analyzing and controlling smart and complex (nonlinear or hybrid) systems have recently attracted a great deal of interest [1]. The abstraction procedure computes a simpler but over-approximated system that includes all possible behaviors of the original system while preserving properties of interest. For instance, to verify that a given complex system satisfies certain properties, we can test for the desired property on the abstracted simple system, and the test result is equivalent to or sufficient for testing for the property on the original complex system.

Literature Review. In general, abstraction is a systematic approximation method that partitions the state space/vector field of a complex system into finite subregions, and then approximates its dynamics in each subregion by a simpler one, resulting in a hybrid system [2], [3]. Multiple abstraction approaches have been developed for several classes of systems in the literature, including nonlinear systems [4]–[10], hybrid systems [11], and uncertain affine and nonlinear systems [12], [13]. In particular, *symbolic* approaches, e.g., [7]–[10], belong to an important class of abstraction methods, where the state and input spaces are discretized to obtain dynamical abstraction systems with *finitely* many number of states and inputs, which symbolizes sets of states and inputs of the original system. However, the number of symbolic states and inputs typically grows exponentially with state and input dimensions. To tackle these scalability issues, [14] presented multirate symbolic models for incrementally stable switched systems to reduce the size of the symbolic models. In addition, by exploiting priorities on the inputs and states, lazy abstraction-based methods can also be used to compute the discrete abstraction on the fly [15]–[17].

On the other hand, hybridization approaches [4], [18], [19] consider the over-approximation of nonlinear vector fields with piecewise affine systems, where the approximation error is accounted for with an additive disturbance. In [4], [18], a *static* single simpler function with a bounded error term is computed analytically for each subdomain, while the work in [4], [19] considers dynamic hybridization that dynamically determines the subdomains on the fly. By contrast, recent works in [5], [11]–[13] employ mesh-based optimization methods to obtain upper and lower affine functions that bracket the original system dynamics, in the sense of inclusion of all possible behaviors for each subregion, to obtain lower approximation errors than [4], [18], [19].

Control synthesis with these piecewise affine abstractions have been shown to yield comparable results to symbolic control approaches [11] and these abstractions were also shown to be effective for reachability analysis [19] and model discrimination [6], [13]. However, the mesh-based approaches that promise to improve the approximation errors in hybridization approaches do not scale well with the size of the mesh (with increasing resolution and state dimension). Hence, tools for computing tight mesh-based abstractions with only limited memory (statically or dynamically) remains an open and interesting challenge, especially for high-dimensional systems and for systems with extremely limited on-board resources, e.g., light-weight mobile robots.

Contributions. In this paper, we propose an incremental abstraction method to dynamically over-approximate nonlinear systems by pairs of bracketing hyperplanes to overcome the issue of space complexity in mesh-based algorithms [5], [11]–[13], which can then be integrated into existing static or on-the-fly hybridization and affine abstraction frameworks, e.g., [4], [6], [11], [13], [19], for solving reachability, control synthesis and model discrimination problems. Specifically, we propose a novel method to carry out the abstraction process sequentially, starting with a small operating region that is

S.M. Hassaan, M. Khajenejad, S. Jensen and S.Z. Yong are with School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ, USA; Q. Shen is with the School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai, P.R. China (email: {shassaan,mkhajene,sjensen8,szyong}@asu.edu, qiangshen@sjtu.edu.cn). This work was supported in part by DARPA grant D18AP00073. We acknowledge Research Computing at Arizona State University (http://www.researchcomputing.asu.edu) for providing High Performance Computing resources that have contributed to the results reported within this paper.

a subset of the entire domain and incrementally expanding to larger domains. At each increment, a local abstraction consisting of a pair of affine hyperplanes can be obtained by solving a linear program. This is in contrast to the conventional mesh-based abstraction methods, e.g., in [5], [11], that construct abstractions statically over the entire domain of interest and have the aforementioned space complexity issues. Similar to [5], [11], our approach can also be employed with iterative partitioning to obtain a piecewise affine abstraction incrementally for each subpartition.

Our approach gives us control over the amount of memory that is allocated to solve each linear program, thus enabling us to abstract high-dimensional nonlinear systems with limited space resources. Moreover, we derive a rigorous proof that guarantees that the incremental abstraction is an over-approximation/abstraction of the original system, which is an important feature when used for reachability analysis and control synthesis. The simulation results demonstrate that the proposed incremental approach is able to obtain abstractions with limited resources, but at the cost of obtaining a worse over-approximation and a longer total computation time due to the sequence of linear programs that need to be solved.

II. Preliminaries

For a vector $v \in \mathbb{R}^n$ and a matrix $M \in \mathbb{R}^{p\times q}$, $\|v\|_i$ and $\|M\|_i$ denote their (induced) i-norm with $i = \{1, 2, \infty\}$.

A. Modeling Framework and Definitions

Consider the nonlinear system:

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

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state with a bounded and closed interval domain $\mathcal{X}$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the known control input with a bounded and closed interval domain $\mathcal{U}$ and vector field $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is a continuous function. For discrete-time systems, x^+ denotes the state at the next time instant while for continuous-time systems, $x^+ = \dot{x}$ is the time derivative of the state. We denote a *sample point* with $(x, u) \in \mathbb{R}^{n+m}$ throughout the paper.

To incrementally abstract the nonlinear system (1), we introduce the following definitions for each increment $k \in \mathbb{N}$ and each partition of the domain, i.e., $I \subseteq \mathcal{X} \times \mathcal{U}$.

Definition 1 (Uniform Mesh, Mesh Elements and Grid Points)**.** *A uniform mesh of each domain I is a collection of polytopic partitions, called mesh elements, with a total of $s_{\max}$ number of points, called grid points, uniformly distributed along all directions and dimensions. The set of grid points is denoted as $\mathcal{M}$ and by construction, the convex hull of $\mathcal{M}$ is the domain I, i.e., $I = Conv(\mathcal{M})$.*

Definition 2 (Diameter)**.** *The diameter δ of a uniform mesh is the greatest distance between vertices of each mesh element.*

Definition 3 (Sample Set and Operating Region)**.** *At any increment k, a set $\mathcal{S}_k$ is called a sample set if it is a subset of all the existing grid points. Moreover, all grid points in the convex hull of the sample set is called the operating region and is denoted by $\mathcal{R}_k$, i.e., $\mathcal{R}_k \triangleq Conv(\mathcal{S}_k) \cap \mathcal{M}$.*

Definition 4 (Expanding Operation Region)**.** *At each increment k, the operation region $\mathcal{R}_k$ is expanding if $\mathcal{R}_{k-1} \subset \mathcal{R}_k$, i.e., the new operating region at the current increment is a strict superset of the previous operating region.*

Definition 5 (Vertex Set)**.** *Given an operating region $\mathcal{R}_k$ at increment k, the set of all vertices of the convex hull of $\mathcal{R}_k$ is called the vertex set, and denoted as $\mathcal{V}_k \triangleq Ver(Conv(\mathcal{R}_k))$. Note that the convex hull of the operating region is a polytope and has a well-defined vertex set.*

The process of over-approximating a nonlinear function as given in (1) can be defined as follows, similar to [5]:

Definition 6 ((Incremental) Affine Abstraction Model)**.** *Given a function $f(x, u)$ and a bounded domain I, the affine functions $\overline{f}(x, u) = \overline{A}x+\overline{B}u+\overline{h}$ and $\underline{f}(x, u) = \underline{A}x+\underline{B}u+\underline{h}$, are called upper and lower affine functions of $f(x, u)$, respectively, if $\forall (x, u) \in I$, $\underline{f}(x, u) \leq f(x, u) \leq \overline{f}(x, u)$. The pair of functions $\mathcal{F} \triangleq \{\overline{f}(x, u), \underline{f}(x, u)\}$ forms an affine abstraction model that over-approximates the given function $f(x, u)$ over domain I. Moreover, an affine abstraction model over the operating region $\mathcal{R}_k$ is called an incremental affine abstraction and is denoted by $\mathcal{F}_k = \{\overline{f}_k(x, u), \underline{f}_k(x, u)\}$.*

One major goal when finding affine abstractions is to get them as tight as possible with a low abstraction error, i.e., with a small distance between the affine hyperplanes:

Definition 7 ((Incremental) Abstraction Error [5])**.** *The abstraction error of an affine abstraction model $\mathcal{F}$ of a nonlinear function $f(x, u)$ over its domain I, is defined as $\theta = \max_{(x,u)\in I} \|\overline{f}(x, u) - \underline{f}(x, u)\|_p$, for any choice of $p \in \{1, 2, \infty\}$. Further, for incremental abstraction $\mathcal{F}_k$, the incremental abstraction error is $\theta_k = \max_{(x,u)\in\mathcal{V}_k} \|\overline{f}_k(x, u) - \underline{f}_k(x, u)\|_p$.*

Next, we reproduce a lemma from [20] that we will rely on to find linear interpolation error bounds over mesh elements:

Lemma 1 ([20, Theorem 4.1 & Lemma 4.3])**.** *Let S be an $(n+m)$-dimensional mesh element such that $S \subseteq \mathbb{R}^{n+m}$ with diameter δ (see Definition 2). Let $f : S \to \mathbb{R}$ be a nonlinear function and let f_l be the linear interpolation of $f(.)$ evaluated at the vertices of the mesh element S. Then, the approximation error σ defined as the maximum error between f and f_l on S, i.e., $\sigma = \max_{s\in S}(|f(s) - f_l(s)|)$, is upper-bounded by*

(i) $\sigma \leq 2\max_{s\in S} \|f(s)\|_\infty$, *if* $f \in C^0$ *on* S,
(ii) $\sigma \leq \lambda\delta_s$, *if* f *is Lipschitz continuous on* S,
(iii) $\sigma \leq \delta_s \max_{s\in S} \|f'(s)\|_2$, *if* $f \in C^1$ *on* S,
(iv) $\sigma \leq \frac{1}{2}\delta_s^2 \max_{s\in S} \|f''(s)\|_2$, *if* $f \in C^2$ *on* S,

where C^0, C^1 and C^2 are sets of continuous, continuously differentiable and twice continuously differentiable functions respectively, λ is the Lipschitz constant, $f'(s)$ is the Jacobian of $f(s)$, $f''(s)$ is the Hessian of $f(s)$ and δ_s satisfies

$$\delta_s \leq \sqrt{\frac{n+m}{2(n+m+1)}}\delta.$$

III. Problem Formulation

When memory resources are scarce (e.g., when the state dimension is high or when on-board memory resources are lim-

ited in small robots), one way to obtain sufficiently tight affine abstractions is to *incrementally* compute over-approximations on smaller subregions of the domain $I \subseteq \mathcal{X} \times \mathcal{U}$ of $f(x,u)$ over κ total increments[1]. Then, combining the incremental abstractions, we obtain the final abstraction over the entire domain of $f(x,u)$. With this in mind, we now first formally define our notion of *limited memory resources*.

Definition 8 (Maximum Number Of Points)**.** *Limited memory resources can be expressed in terms of the limit on maximum number of points, denoted as $\overline{s}$, that can be processed at any increment. Thus, for a user-specified $\overline{s}$, the total number of increments, denoted as κ, required to process all the grid points $s_{\max}$, can then be computed as:*

$$\kappa = \frac{s_{max} - \overline{s}}{\overline{s} - \gamma} + 1, \tag{2}$$

where γ is the number of points carried over to $\mathcal{R}_k$ from $\mathcal{R}_{k-1}$. In Section IV, we will discuss the choice of γ.

Using the concept of incremental abstraction, the problem of affine abstraction of the system in (1) can be recast as:

Problem 1 (Piecewise Affine Abstraction of a High-Dimensional System)**.** *Given a high-dimensional nonlinear function in* (1)*, along with the requirement that at most $\overline{s}$ sample points can be taken into consideration at each increment, find the affine abstraction of f over domain $I \subseteq \mathcal{X} \times \mathcal{U}$ (cf. Definition 6) using a sequence of incremental abstractions over expanding operating regions.*

Mathematically speaking, our goal is to compute a piecewise affine abstraction model $\mathcal{F}$ of f over $I \subseteq \mathcal{X} \times \mathcal{U}$, using $\{\mathcal{F}_k\}_{k=1}^{\kappa}$, obtained from incremental abstractions over κ increments, each with at most $\overline{s}$ samples, such that $\mathcal{F} \equiv \mathcal{F}_\kappa$, by solving the following problem at each increment k:

$$\begin{aligned} &\min\ \theta_k \\ &\text{s.t.}\ \begin{array}{l} \overline{f}_k(x,u) \geq f(x,u) \geq \underline{f}_k(x,u), \\ \overline{f}_k(x,u) - \underline{f}_k(x,u) \leq \theta_k \mathbb{1}_n, \end{array} \ \forall (x,u) \in \mathcal{R}_k, \end{aligned} \tag{3}$$

$\forall k \in \{1, \ldots, \kappa\}$, and $\mathcal{R}_k$ is expanding from $\mathcal{R}_0 = \emptyset$ to $\mathcal{R}_\kappa = \mathcal{M}$, i.e., $\emptyset = \mathcal{R}_0 \subset \mathcal{R}_1 \subset \ldots \subset \mathcal{R}_\kappa$.

IV. Main Results

To overcome the limitations on space complexity, we propose an incremental abstraction approach, in which at each increment, at most $\overline{s}$ number of sample points are processed to obtain an affine abstraction. To achieve this, we first propose an incremental abstraction approach, as follows.

Lemma 2. *Given the affine abstraction model $\mathcal{F}_{k-1} = \{\underline{f}_{k-1}(x,u), \overline{f}_{k-1}(x,u)\}$ for the nonlinear function $f(x,u)$ over an operating region $\mathcal{R}_{k-1}$, at increment k, solving the following minimization problem over the sample set $\mathcal{S}'_k = (\mathcal{R}_k \setminus \mathcal{R}_{k-1}) \cup \mathcal{V}_{k-1}$, where $\mathcal{V}_{k-1} \triangleq Ver(Conv(\mathcal{R}_{k-1}))$, obtains an affine abstraction[2] of $f(x,u)$ over $\mathcal{R}_k$:*

$$\min_{\theta_k, \overline{A}_k, \underline{A}_k, \overline{B}_k, \underline{B}_k, \overline{h}_k, \underline{h}_k} \theta_k \tag{4}$$

subject to:

$\forall (x,u) \in \mathcal{R}_k \setminus \mathcal{R}_{k-1}$:

$$\begin{aligned} \overline{A}_k\, x + \overline{B}_k\, u + \overline{h}_k &\geq f(x,u), \\ \underline{A}_k\, x + \underline{B}_k\, u + \underline{h}_k &\leq f(x,u), \end{aligned} \tag{4a}$$

$\forall (x,u) \in \mathcal{V}_{k-1}$:

$$\begin{aligned} \overline{A}_k\, x + \overline{B}_k\, u + \overline{h}_k &\geq \overline{A}_{k-1}\, x + \overline{B}_{k-1}\, u + \overline{h}_{k-1}, \\ \underline{A}_k\, x + \underline{B}_k\, u + \underline{h}_k &\leq \underline{A}_{k-1}\, x + \underline{B}_{k-1}\, u + \underline{h}_{k-1}, \end{aligned} \tag{4b}$$

$\forall (x,u) \in \mathcal{V}_k = Ver(Conv(\mathcal{S}'_k))$:

$$(\overline{A}_k - \underline{A}_k)\, x + (\overline{B}_k - \underline{B}_k)\, u + \overline{h}_k - \underline{h}_k \leq \theta_k \mathbb{1}_n. \tag{4c}$$

Proof. In the optimization problem given in (4), the constraints (4a) and (4b) make sure that the two hyperplanes at increment k bracket the nonlinear function for all newly added grid points and the vertices of operating region $\mathcal{R}_{k-1}$, respectively. Moreover, in light of [5, Lemma 1], it is obtained from (4b) that $\forall (x,u) \in \mathcal{R}_{k-1}$,

$$\overline{f}_k(x,u) \geq \overline{f}_{k-1}(x,u),\ \underline{f}_k(x,u) \leq \underline{f}_{k-1}(x,u). \tag{5}$$

Since the given two affine hyperplanes $\mathcal{F}_{k-1} = \{\underline{f}_{k-1}(x,u), \overline{f}_{k-1}(x,u)\}$ over-approximate the nonlinear function over operating region $\mathcal{R}_{k-1}$, i.e, $\underline{f}_{k-1}(x,u) \leq f(x,u) \leq \overline{f}_{k-1}(x,u)$, $\forall (x,u) \in \mathcal{R}_{k-1}$, we further have

$$\underline{f}_k(x,u) \leq f(x,u) \leq \overline{f}_k(x,u), \quad \forall (x,u) \in \mathcal{R}_{k-1}. \tag{6}$$

As a result, it follows from (4a) and (6) that

$$\underline{f}_k(x,u) \leq f(x,u) \leq \overline{f}_k(x,u), \quad \forall (x,u) \in \mathcal{R}_k, \tag{7}$$

which implies that the affine hyperplanes obtained at increment k over-approximate the nonlinear function $f(x,u)$ overall the current operating region $\mathcal{R}_k$.

Finally, the constraint in (4c) ensures that the two affine hyperplanes obtained at the increment k are as close to each other as possible, i.e., the abstraction error is minimized. □

Using the above lemma, Algorithm 1 incrementally solves the abstraction problem formulated in Problem 1, where a sequence of linear problems are solved with expanding operating regions until we achieve the entire domain. The following theorem guarantees that incremental affine abstraction also yields an affine abstraction model of system (1).

Theorem 1. *Consider the nonlinear system* (1) *with $(x,u) \in \mathcal{X} \times \mathcal{U}$. Let $\overline{s}$ indicate the maximum number of sample points allowed to be taken at each iteration k. Algorithm 1 incrementally solves the abstraction problem formulated in Problem 1, i.e., $\forall (x,u) \in I \subseteq \mathcal{X} \times \mathcal{U}$, it returns upper and lower affine functions $\overline{f}(x,u) = \overline{f}_k(x,u) + \sigma \mathbb{1}$ and $\underline{f}(x,u) = \underline{f}_k(x,u) - \sigma \mathbb{1}$ that over-approximate the nonlinear system* (1)*, with the corresponding interpolation error σ in Lemma 1 and $\mathbb{1}$ is a vector of ones.*

[1]For ease and clarity of exposition, throughout this paper, we consider affine abstraction models with only a single region $I \subseteq \mathcal{X} \times \mathcal{U}$. The results in this paper can be extended in a straightforward manner to allow the entire domain $I \subseteq \mathcal{X} \times \mathcal{U}$ to be iteratively partitioned into multiple subdomains, as was done in the literature, e.g., [5], [6], [11], to further decrease abstraction errors, resulting in piecewise affine abstractions.

[2]For brevity, we only provide the linear program for when the abstraction error is defined with $p = \infty$. Corresponding formulations for $p = \{1, 2\}$ can be obtained with slight modifications, where we have a linear program when $p = 1$, and a quadratically constrained linear program when $p = 2$.

Algorithm 1 Procedures of Incremental Abstraction

1) Initialize $k = 1$, $\mathcal{R}_0 = \emptyset \implies \mathcal{V}_0 = \emptyset$.
2) At increment k, consider a new sample set $\mathcal{S'}_k = (\mathcal{R}_k \setminus \mathcal{R}_{k-1}) \cup \mathcal{V}_{k-1}$ of size $\overline{s}$, where the set $(\mathcal{R}_k \setminus \mathcal{R}_{k-1}) \neq \emptyset$ denotes the newly added grid points such that $\mathcal{R}_k$ is expanding with k.
3) For the sample set $\mathcal{S'}_k$, use Lemma 2 to obtain hyperplanes $\mathcal{F}_k = \{\overline{f}_k, \underline{f}_k\}$ that over-approximate the nonlinear function (1) over $\mathcal{S'}_k$.
4) Go to step 2 with $k = k + 1$ if $k < \kappa$.
5) After obtaining the final hyperplanes $\mathcal{F}_\kappa = \{\underline{f}_\kappa(x,u), \overline{f}_\kappa(x,u)\}$, the affine abstraction over the domain $\mathcal{X} \times \mathcal{U}$ for the system (1) is:

$$\overline{f}(x,u) = \overline{A}_\kappa x + \overline{B}_\kappa u + \overline{h}_\kappa + \sigma \mathbb{1}_n,$$
$$\underline{f}(x,u) = \underline{A}_\kappa x + \underline{B}_\kappa u + \underline{h}_\kappa - \sigma \mathbb{1}_n,$$

where σ is the approximation error in Lemma 1.

Proof. Using mathematical induction, we will prove that Theorem 1 solves the Problem 1 incrementally.

In the first increment $k = 1$, we have the operating region $\mathcal{R}_1$. Since $\mathcal{R}_0 = \emptyset$, we have $\mathcal{V}_0 = \emptyset$. Therefore, we further have $\mathcal{S'}_1 = (\mathcal{R}_1 \setminus \mathcal{R}_0) \cup \mathcal{V}_0 = \mathcal{R}_1$. Based on Algorithm 1, solving the optimization problem defined in Lemma 2 over $\mathcal{S'}_1$ will yield the affine hyperplanes $\mathcal{F}_1 = \{\underline{f}_1(x,u), \overline{f}_1(x,u)\}$ with:

$$\underline{f}_1(x,u) = \underline{A}_1 x + \underline{B}_1 u + \underline{h}_1,\ \overline{f}_1(x,u) = \overline{A}_1 x + \overline{B}_1 u + \overline{h}_1.$$

Since $\mathcal{S'}_1 = \mathcal{R}_1$, these two hyperplanes also bracket the function $f(x)$ at all sample points in $\mathcal{R}_1$, i.e.

$$\underline{f}_1(x,u) \leq f(x,u) \leq \overline{f}_1(x,u), \quad \forall (x,u) \in \mathcal{R}_1.$$

At increment $k > 1$, suppose that the obtained affine hyperplanes $\mathcal{F}_k = \{\underline{f}_k(x,u), \overline{f}_k(x,u)\}$ over $(x,u) \in \mathcal{S'}_k = (\mathcal{R}_k \setminus \mathcal{R}_{k-1}) \cup \mathcal{V}_{k-1}$ satisfy:

$$\underline{f}_k(x,u) \leq f(x,u) \leq \overline{f}_k(x,u), \quad \forall (x,u) \in \mathcal{R}_k.$$

Then, following the same lines in the proof of Lemma 2 for increment $k+1$, we have

$$\underline{f}_{k+1}(x,u) \leq f(x,u) \leq \overline{f}_{k+1}(x,u),\ \forall (x,u) \in \mathcal{R}_{k+1}.$$

Therefore, the affine hyperplane obtained at any future increment will also over-approximate the nonlinear function over all the past operating regions, hence at the last increment $k = \kappa$, the final two affine hyperplanes $\mathcal{F}_\kappa = \{\underline{f}_\kappa(x,u), \overline{f}_\kappa(x,u)\}$ will over-approximate the nonlinear function over the entire mesh since the operating region $\mathcal{R}_\kappa = Conv(\mathcal{S}_\kappa) \cap \mathcal{M} = \mathcal{M}$ contains all s_{max} samples. Finally, using a combination of the result in [5, Lemma 2] and Lemma 1, the desired affine abstraction can be obtained by accounting for the interpolation errors when extending from grid points of the mesh to the entire continuous domain (cf. step 5 of Algorithm 1). This completes the proof. □

To reduce space complexity, the proposed incremental abstraction algorithm only computes affine hyperplanes for $\overline{s}$ sample points at each increment k. As shown in step 2 of the Algorithm 1, at each increment k, we consider a new sample set $\mathcal{S'}_k = (\mathcal{R}_k \setminus \mathcal{R}_{k-1}) \cup \mathcal{V}_{k-1}$ of size $\overline{s}$ and discard the previous points from the set $\mathcal{R}_{k-1} \setminus \mathcal{V}_{k-1}$ to accommodate new points. Then, in Lemma 2, we show that retaining these $\overline{s}$ grid points at each increment k is enough to provide conservative over-approximation over all other discarded points at $k-1$.

Bounds on the total number of increments κ of the incremental abstraction can be calculated if $\overline{s}$ is given. For a state-input domain $I \subseteq \mathcal{X} \times \mathcal{U} \subset \mathbb{R}^{n+m}$, in general at least $n+m+1$ grid points are required to define a hyperplane. Moreover, since we require the operating region to expand with each increment, the maximum number of points γ that can be carried over future increments cannot exceed $\overline{s} - 1$. Therefore, γ is bounded as follows: $n + m + 1 \leq \gamma \leq \overline{s} - 1$. Hence, using (2), the following bounds on κ apply:

$$\kappa \in \left[\frac{s_{max} - \overline{s}}{\overline{s} - (n+m+1)} + 1, s_{max} - \overline{s} + 1 \right].$$

V. Examples and Discussion

In this section, we demonstrate the capability of the proposed incremental abstraction approach[3] in the limited resource setting using 2 high-dimensional nonlinear systems.

A. Nonlinear Rastrigin's function [21]

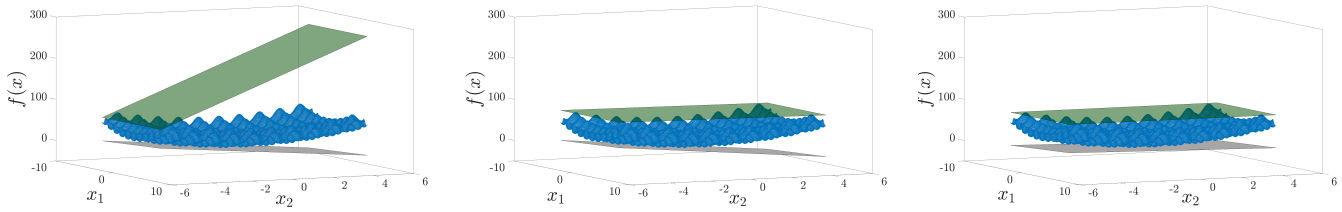
First, we consider a nonlinear system with dynamics described by Rastrigin's function [21]:

$$\dot{x}_i = f(x) = 10d + \textstyle\sum_{j=1}^{d} [x_j^2 - 10\cos(2\pi x_j)] \qquad (8)$$

where $x = [x_1, \ldots, x_d]^T \in \mathbb{R}^d$ with d being the dimension of state x. In addition, we also assume that $x_i \in [-5.1, 5.1]$ for all $i \in \{1, \ldots, d\}$. All simulations are performed on Arizona State University's Agave Cluster on a single thread of one of the cores of Intel Xeon E5-2680 v4 CPU processor running at 2.40GHz. The script is written and run on MATLAB® version 2017a, and uses Gurobi [22] as the linear program solver. The amount of RAM available for the simulations is also adjusted to cater to the required environment for the sake of a fair comparison.

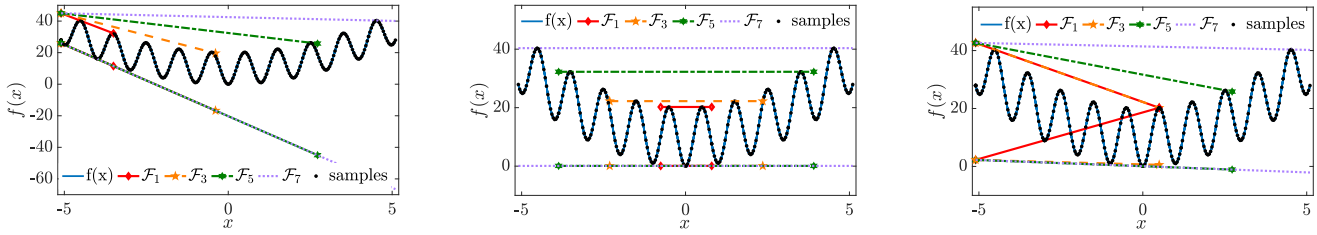
a) Effects of sample size on abstraction error performance with unlimited memory: For our first study, we emulate a virtually unlimited resource environment by setting the maximum available system RAM to 64GB, and use the function (8) with a 2-dimensional domain. In each dimension, we consider 51 points, resulting in a total of $51^2 = 2601$ grid points. The computational times are compared for different cases of maximum number of grid points that can be considered for each linear program. In the first case, $\overline{s} = 50$ is chosen, which takes 57 increments to find the over-approximation of the 2-dimensional nonlinear system. For the second case, $\overline{s} = 500$ solves the problem in 6 increments. Finally, the last case considers all the points at once, as in [5], to solve the problem. Figure 1 depicts the resulting lower and upper affine hyperplanes as well as the original nonlinear function under these three cases. In all cases, the nonlinear system is over-approximated by the affine hyperplanes obtained from the proposed abstraction method. Table I shows the computational times for each case and the corresponding maximum distances

[3]We chose $p = 1$ for the abstraction errors, since the results appear "tighter" than when $p = \infty$, while $p = 2$ is computationally intensive.



(a) Abstraction with $\bar{s} = 50$ (b) Abstraction with $\bar{s} = 500$ (c) Abstraction with all points (same as [5])

Fig. 1: Comparison of abstractions for varying maximum numbers of grid points $\bar{s}$ (memory allocation) of (8) with $d = 2$.



(a) With $\mathcal{R}_1$ to the left (b) With $\mathcal{R}_1$ at the center (c) With grid point $x = 0.5$ as warm-start

Fig. 2: Comparison of affine abstractions of (8) with $d = 1$ for different heuristics. The hyperplanes in $\mathcal{F}_k$ for $k = 1, 3, 5, 7$ show the evolution of the abstraction after respective increments. The lengths of each $\mathcal{F}_k$ vary as the domain varies.

TABLE I: Effects of Sample Size on Performance

Performance Parameter	Incremental Abstraction		1-Step Abstraction [5]	1-Step Abstraction [11]
$\bar{s}$	50	500	All Points	All Points
Time Taken (sec)	15	6.21	0.334	0.348
Abstraction Error, θ_κ	300.4	112.4	80.23	84.19

between the hyperplanes, which demonstrates that the proposed incremental abstraction is suboptimal when compared to 1-step abstraction approaches in [5], [11] and its performance in terms of abstraction error at the final iteration, θ_κ, and total time is dependent on the amount of allocated memory in terms of $\bar{s}$. Therefore, taking $\bar{s}$ as a controllable parameter, the proposed abstraction method allows the users to determine the trade-off between computational time and resources required to solve higher-dimensional nonlinear function abstractions and the tightness of the resulting abstraction.

b) Effects of sample size on abstraction error performance with limited memory: Next, we consider the limited memory case by setting the maximum available system RAM to 4GB. Here, in each dimension, 5 grid points are chosen, so, depending on the dimension d of the domain, the total number of points will be 5^d. For incremental abstraction with 4GB RAM, the maximum number of grid points we can consider in each increment is set to $\bar{s} = 10^6$ points. Under this resource limitation, the comparison between incremental abstraction and the 1-step abstraction in [5] is summarized in Table II. We observed that with incremental abstraction, abstractions of higher dimensional nonlinear systems using only limited resources can be achieved with more time, whereas the 1-step abstraction methods in [5], [11] return an error and cannot compute any abstraction for $d \geq 9$. Further, the results suggest that if time is not a concern (e.g., in offline implementations), the incremental abstraction problem can, in principle, be solved for systems with an arbitrarily high state dimension by computers with limited memory.

c) Effects of dynamic expansion of operating regions: Inspired by lazy constraints in Gurobi [22] (that still need

TABLE II: Performance Under Limited Resources

Dimension (d)	Time Taken (sec.)			Abstraction Error, θ_κ		
	Incremental		1-Step	Incremental		1-Step
	Regular	Dynamic		Regular	Dynamic	
1	2.091	2.201	2.179	55.8	55.8	55.8
5	2.26	2.229	2.189	279.2	279.2	279.2
8	14.23	14.88	9.5	446.72	446.72	446.72
9	69.96	106.3	N/A	731	731	N/A
11	2261.2	2763.4	N/A	1226.3	1103.4	N/A
13	69027.8	69326.6	N/A	1442.3	1157.3	N/A

to be formulated within a single optimization problem), a lazy/dynamic variation of our approach can be employed to exploit intermediate abstractions to discard samples that already satisfy the intermediate abstraction, resulting in less increments. However, from Table II, we observed that the discard step also takes non-negligible time; hence, we only expect reduced computational time (when compared to regular expansion) if the total number of increments is high.

d) Effects of heuristics on abstraction error performance with limited memory: Additionally, we observed that heuristics can improve the performance of our incremental abstraction in terms of decreased abstraction error. To better visualize the effects of the heuristics, we consider the example in (8) with $d = 1$. The example has $s_{max} = 250$ grid points and the maximum number of points $\bar{s}$ is set to 40.

From our analysis, two major reasons are associated with increased suboptimality of the incremental procedure: (i) conservative approximations due to constraints in (4b) for guaranteeing future abstractions, and (ii) when using expanding operating regions, we may start from a closely located cluster of samples, whose abstraction for very small $\bar{s}$ may have higher slope than the Lipschitz constant of the system in (1). Thus, we conjecture that one of the ways to tackle the first issue is by choosing the starting region $\mathcal{R}_1$ smartly. In Figures 2a–2b, we show the effects of selecting different starting points on the final abstraction for (8) in 1D. By choosing the starting region at the center of the domain $\mathcal{X}$, the overall abstraction

TABLE III: Performance of Abstraction of Swarm Dynamics

N	Func -tion	Time Taken (sec.)			Abstraction Error, θ_κ		
		Incremental		1-Step	Incremental		1-Step
		Regular	Dynamic		Regular	Dynamic	
3	$f_i^{\mathbf{x}}(x)$	5.05	5.23	5.5	0.1118	0.1118	0.1118
	$f_i^{\mathbf{y}}(x)$	4.73	4.77	4.66	0.8798	0.8798	0.8798
	$f_i^{\theta}(x)$	6.81	6.25	5.24	2.9157	2.9157	2.9157
5	$f_i^{\mathbf{x}}(x)$	2976.6	4100.4	N/A	0.1377	0.1388	N/A
	$f_i^{\mathbf{y}}(x)$	2959	4191.9	N/A	1.1575	1.1637	N/A
	$f_i^{\theta}(x)$	3014.6	6315.3	N/A	28.0253	27.9949	N/A

is less conservative than the one obtained when the starting region is on one end of the domain as in Figure 2a. Further, we conjecture that the second issue can be resolved by picking sample points that are more spread-out in the domain as a warm-start for the incremental abstraction. This will prevent the closely clustered region to be formed in $\mathcal{R}_1$. In Figure 2c, providing a random grid point at $x = 0.5$ as a warm-start also results in a better abstraction than the one obtained without any warm-starts. Instead of random samples, certain properties of the nonlinear function $f(x, u)$ can also be used for warm-starting, e.g., global minima or global maxima of $f(x, u)$.

B. Rendezvous of a Robot Swarm

We consider the dynamics of a swarm of robots described in [23], in the form of (1), with the following parameters: $n = 2N + 1$, where N is the number of agents/robots, $m = 0$ and x is the augmented state of the whole swarm.

In this simulation, we consider swarms with $N = 3$ and $N = 5$ robots[1], which correspond to 7- and 11-dimensional nonlinear systems, respectively. The available system RAM is set to 500MB to emulate limited on-board resources. In each dimension, 5 points are taken and $\bar{s} = 10^5$. As shown in Table III, both the proposed incremental abstraction and the 1-step abstraction in [5] obtain comparable results in terms of computational time and abstraction error when $N = 3$. However, with $N = 5$, the 1-step abstraction in [5] is unable to generate an affine abstraction due to the limited memory, while the proposed incremental approach still can.

VI. Conclusions

We proposed an incremental affine abstraction approach to over-approximate a class of nonlinear systems as piecewise affine systems, by dynamically computing pairs of affine hyperplanes to envelop the nonlinear systems with expanding operating regions. Initially, we consider a small operating region and solve a linear program to obtain two affine hyperplanes that locally over-approximate the nonlinear system. Then, expanding the operating region with new grid points incrementally, we can find the corresponding affine hyperplanes for a larger domain until the entire domain is covered. The proposed incremental abstraction approach has the capability of reducing computational space complexity, especially when the nonlinear system has high dimensions.

Future work will include a comparison of the incremental abstraction approach with symbolic approaches in the context of reachability analysis and control synthesis. Moreover, we will exploit system structure, e.g., incremental stability [14], priorities [15]–[17], decomposability [24] and submodularity [25], to further reduce the size of the abstraction problem and to more formally analyze the abstraction errors.

[1]The states are bounded as $x_1 \in [-5, 5]$, $x_2 \in [-5, 5]$, $x_3 \in [-7, 7]$, $x_4 \in [-7, 7]$, $x_5 \in [-7, 7]$, $y_1 \in [0, 0.4]$, $y_2 \in [0.5, 0.9]$, $y_3 \in [1, 5]$, $y_4 \in [0, 0.876]$, $y_5 \in [0, 1.67]$ and $\theta_i \in [-0.02, 0.02], \forall i \in \{1, \ldots, 5\}$.

References

[1] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.

[2] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.

[3] ——, "Reachability analysis of nonlinear systems using conservative approximation," in *Int. Workshop on Hybrid Systems: Computation and Control*. Springer, 2003, pp. 20–35.

[4] A. Girard and S. Martin, "Synthesis for constrained nonlinear systems using hybridization and robust controller on symplices," *IEEE Trans. on Automatic Control*, vol. 57, no. 4, pp. 1046–1051, 2012.

[5] K. R. Singh, Q. Shen, and S. Z. Yong, "Mesh-based affine abstraction of nonlinear systems with tighter bounds," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3056–3061.

[6] K. R. Singh, Y. Ding, N. Ozay, and S. Z. Yong, "Input design for nonlinear model discrimination via affine abstraction," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 175–180, 2018.

[7] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, 2015, pp. 58–67.

[8] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[9] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1781–1796, 2016.

[10] M. Zamani, P. Esfahani, R. Majumdar, A. Abate, and J. Lygeros, "Symbolic control of stochastic systems via approximately bisimilar finite abstractions," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3135–3150, 2014.

[11] V. Alimguzhin, F. Mari, I. Melatti, I. Salvo, and E. Tronci, "Linearizing discrete-time hybrid systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5357–5364, 2017.

[12] Q. Shen and S. Z. Yong, "Robust optimization-based affine abstractions for uncertain affine dynamics," in *American Control Conference (ACC)*. IEEE, 2019, pp. 2452–2457.

[13] Z. Jin, Q. Shen, and S. Z. Yong, "Optimization-based approaches for affine abstraction and model discrimination of uncertain nonlinear systems," in *IEEE Conference on Decision and Control*, 2019.

[14] A. Saoud and A. Girard, "Optimal multirate sampling in symbolic models for incrementally stable switched systems," *Automatica*, vol. 98, pp. 58–65, 2018.

[15] G. Pola, A. Borri, and M. D. Di Benedetto, "Integrated design of symbolic controllers for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 534–539, 2011.

[16] O. Hussien and P. Tabuada, "Lazy controller synthesis using three-valued abstractions for safety and reachability specifications," in *IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3567–3572.

[17] A. Saoud, E. Ivanova, and A. Girard, "Efficient synthesis for monotone transition systems and directed safety specifications," in *IEEE Conference on Decision and Control (CDC)*, 2019.

[18] S.-I. Azuma, J.-I. Imura, and T. Sugie, "Lebesgue piecewise affine approximation of nonlinear systems," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 1, pp. 92–102, 2010.

[19] T. Dang, O. Maler, and R. Testylier, "Accurate hybridization of nonlinear systems," in *ACM International Conference on Hybrid Systems: Computation and Control*, 2010, pp. 11–20.

[20] M. Stämpfle, "Optimal estimates for the linear interpolation error for simplices," *Jour. of Approximation Theory*, vol. 103, pp. 78–90, 2000.

[21] H. Pohlheim, "Geatbx examples examples of objective functions," 2005, available: http://www.geatbx.com/.

[22] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

[23] S. Nagavalli, N. Chakraborty, and K. Sycara, "Automated sequencing of swarm behaviors for supervisory control of robotic swarms," in *IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 2674–2681.

[24] L. P. Nilsson, "Correct-by-construction control synthesis for high-dimensional systems," Ph.D. dissertation, 2017.

[25] A. Bernstein, Y. Disser, and M. Groß, "General bounds for incremental maximization," *arXiv preprint arXiv:1705.10253*, 2018.