THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Kaa: A Python Implementation of Reachable Set Computation Using Bernstein Polynomials

Edward Kim and Parasara Sridhar Duggirala
Department of Computer Science
UNC- Chapel Hill

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Introduction

- Reachable set computation is one of the many important tools available for the verification of dynamical and hybrid systems.

- One of the simpler and easier-to-understand reachable set computation algorithms for polynomial discrete dynamical systems utilizes Bernstein polynomials and parallelotope bundles.

- Tomasso Dreossi, Thao Dang and Carla Piazza implemented a tool called Sapo in C++ which leverages parallelotope bundles and the properties of Bernstein polynomials.

- Kaa is a reimplementation of Sapo using robust Python libraries. The result is a compact implementation with only around ~650 lines of code.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Preliminaries

- The state of a system, denoted as x, lies in a domain $D \subseteq \mathbb{R}^n$. A discrete-time polynomial nonlinear system is denoted as $x^+ = f(x)$
- The trajectory is denoted as $\xi(x_0)$, is the sequence $x_0, x_1, \ldots$ where $x_{i+1} = f(x_i)$.
- Given an initial set Θ, the reachable set at time k, denoted as $\Theta_k = \{\ \xi(x_0, k) \mid x_0 \in \Theta\}$ where $\xi(x_0, k) = x_k$.

Parallelotope Bundles

- A parallelotope P is a set of states in $\mathbb{R}^n$ denoted as $\langle \Lambda, c \rangle$ where $\Lambda \in \mathbb{R}^{2n \times n}$ and $c \in \mathbb{R}^{2n}$, $\Lambda_{i+n} = -\Lambda_i$ and $i \in \{1, \ldots, n\}$ such that:

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

- Λ is called the *direction matrix* where Λ_i denotes the i^{th}row of Λ. The vector c is called the *offset vector* where c_i is the i^{th} element of the vector.

- A parallelotope bundle Q is a set of parallelotopes $\{P_1, \ldots, P_m\}$ where $Q = \cap_{i=1}^{m} P_i$. Note that any polytope initial set can be expressed as a parallelotope bundle.

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Parallelotope Bundles

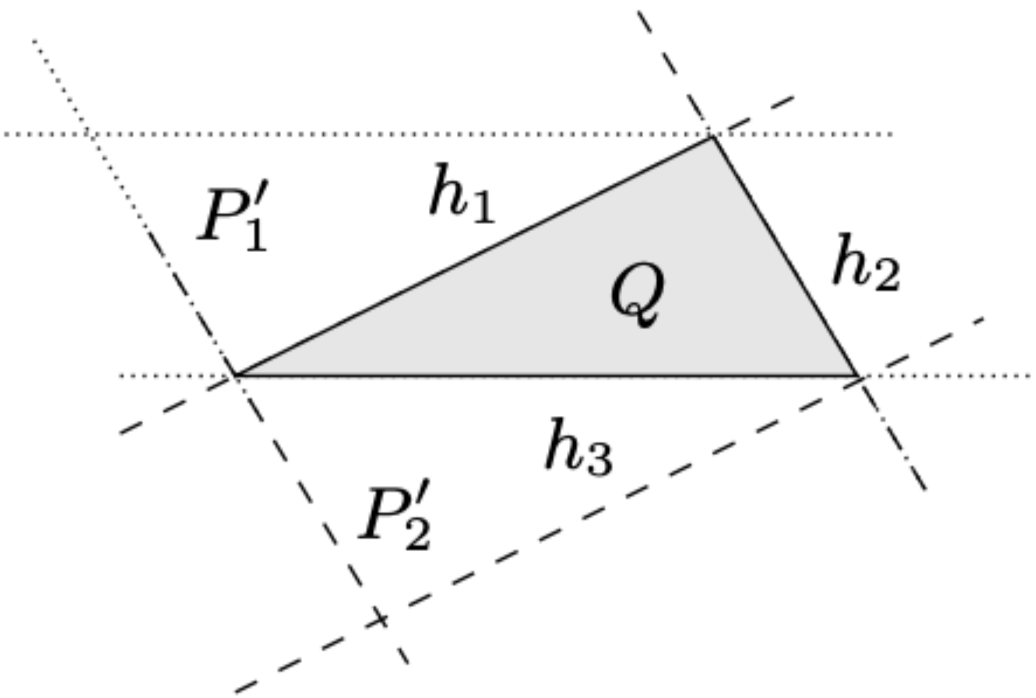


Figure 1 from Dreossi et. al: Parallelotope Bundles for Polynomial Reachability (2016)

Bernstein Polynomials

- Given two multi-indices i and d of size n, where $i \leq d$, the Bernstein polynomial of degree d and index i is

$$\mathscr{B}_{i,d} = \beta_{i_1,d_1}(x_1)\beta_{i_2,d_2}(x_2)\ldots\beta_{i_n,d_n}(x_n)$$

$$\beta_{i_m,d_m}(x_m) = \binom{d_m}{i_m} x_m^{i_m}(1 - x_m)^{d_m - i_m}$$

- Any polynomial function can be expressed in the Bernstein basis.

Bernstein Polynomials

- The corresponding *Bernstein Coefficients* can be explicitly calculated for multi-index i and polynomial degree d:

$$b_{i,d} = \sum_{j \leq i} \prod_{r} \frac{\binom{i_r}{j_r}}{\binom{d_r}{j_r}} a_j$$

- The upper and lower bounds of polynomial $h(x_1, \ldots, x_n)$ over unit box $[0,1]^n$ are bounded by the Bernstein coefficients:

$$min_{i \in I}\{b_i\} \quad \leq \quad inf_{x \in [0,1]^n} h(x) \quad \leq \quad sup_{x \in [0,1]^n} h(x) \quad \leq \quad max_{i \in I}\{b_i\} \, .$$

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Reachable Set Comp.

- A parallelotope P can also be represented as an affine transformation T_p from $[0,1]^n$ to P.

- Therefore, upper bounds on the supremum of a function h over P is equivalent to upper bound of $h \circ T_p$ over $[0,1]^n$.

- We denote the procedures for calculating such upper and lower bounds for a polynomial h over some parallelotope P as $\mathsf{BernsteinUpper}(h, P)$ and $\mathsf{BernsteinLower}(h, P)$ respectively.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Reachable Set Comp.

- Given parallelotope bundle $Q = \{P_1, P_2, \ldots, P_m\}$ and a discrete dynamical system $x^+ = f(x)$, we wish to compute an over-approximation of the image $f(Q)$ as a new bundle $Q' = \{P'_1, P'_2, \ldots, P'_m\}$.

- We ensure that direction matrix $\Lambda_{\_,i}$ of P'_i is same as P_i and the computation is required only to compute the offsets of the directions according to the following non-linear optimization problems:

$$c_{j,i} = \max_{x \in P_i} \Lambda_{j,i} \cdot f(x)$$

$$c_{j+n,i} = \max_{x \in P_i} -\Lambda_{j,i} \cdot f(x)$$

- Here, $c_{j,i}$ is the j^{th} offset of parallelotope P_i. Similarly, $\Lambda_{j,i}$ is the j^{th} row of the directions matrix for P_i.

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Reachable Set Comp.

- We can invoke BernsteinUpper(h, P) and BernsteinLower(h, P) to update the offsets according to the solutions found over all parallelotopes in the bundle $Q = \{P_1, P_2, \ldots, P_m\}$:

$$c_{j,i} = min_{l=1}^{m}\left\{ \text{BernsteinUpper}\left(\Lambda_{j,i} \cdot f(x), P_l\right) \right\} \text{ if } j \leq n\,.$$

$$c_{j+n,i} = max_{l=1}^{m}\left\{ \text{BernsteinLower}\left(\Lambda_{j,i} \cdot f(x), P_l\right) \right\} \text{ otherwise.}$$

- We iterate this over a certain number of time steps to produce the reachable set.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Sapo Drawbacks

- First published in: Dreossi, T. : Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems (2017).
- Current implementation is verbose. The main core of algorithm takes over a thousand lines of C++ code.
- It does not have native plotting functionality. Sapo generates MATLAB code which must be separately run through either MATLAB or Octave. Simultaneous visualization is clunky at best.
- Suffers from little to no documentation. The curious reader must delve into previously published papers to find an explanation of the inner workings.
- Consequently, it becomes difficult to accommodate experimentation.

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Motivations for Kaa

- Python is known for its powerful, well-tested symbolic and matrix-computation libraries.

- *Numpy* libraries are popular matrix-computation libraries which allow higher-level manipulation of matrices. This gives us an avenue of overcoming the verbosity and the possibility of memory leaks inherent in implementing identical features in C++.

- The library of *Sympy* has powerful symbolic manipulation tools which allow us to comfortably perform many sensitive symbolic subsititutions into polynomials.

- *Matplotlib* library has intuitive plotting facilities that we integrate into our tool for visualizing the reachable set. In particular, *Matplotlib* facilitates the ability to visualize several reachable sets simultaneously.

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Accessibility

- We offer a Juypter Notebook to rapidly introduce the interested reader to the techniques and tools we offer through Kaa.
- Juypter notebooks are simple to create and well-known for their straightforward user interface.
- By leveraging the *Matplotlib* library for visualizing the reachable set, we were able to design an engaging interactive tutorial and experimentation platform for visualizing reachable sets of non-linear systems.
- We document the code extensively and offer resources for learning the internals of Kaa.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Results: SIR Model

- The SIR epidemic model is a 3-dimensional dynamical system governed by the following dynamics:

$$s_{k+1} = s_k - (\beta s_k i_k)\Delta$$
$$i_{k+1} = i_k + (\beta s_k i_k - \gamma i_k)\Delta$$
$$r_{k+1} = r_k + (\gamma i_k)\Delta$$

$\beta = 0.34$, $\gamma = 0.05$

$\Delta = 0.5$

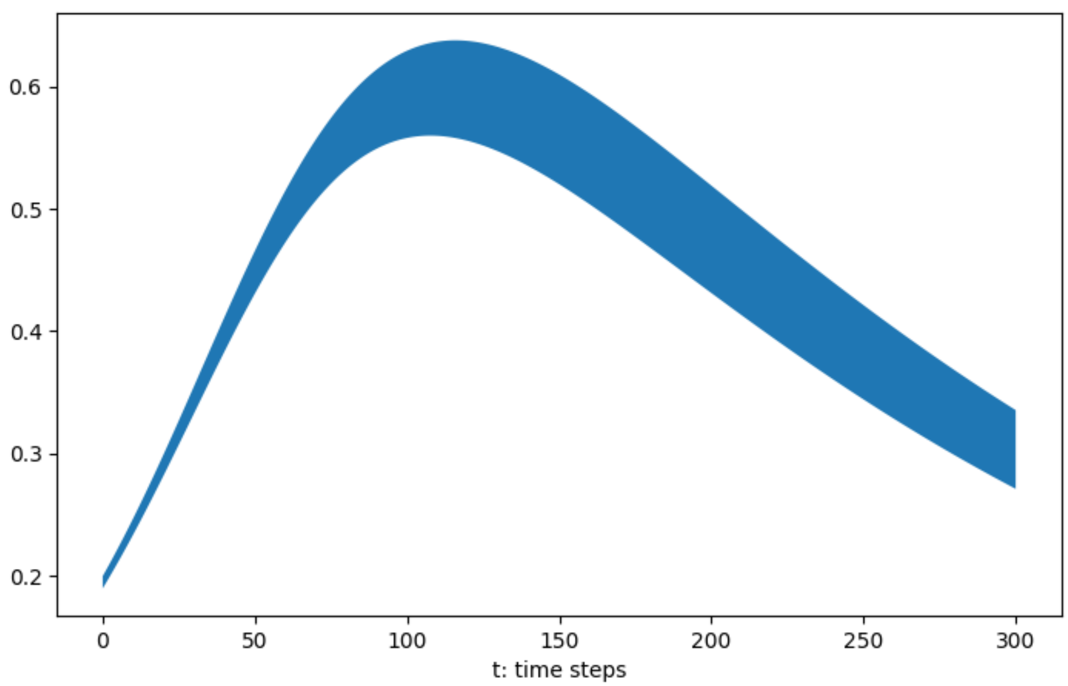
Upper and lower offsets for variable I

Time Steps	Kaa (offu)	Sapo (offu)	Kaa (offl)	Kaa (offl)
50	0.470716	0.470716	-0.435191	-0.43519
51	0.475839	0.475839	-0.439599	-0.439599
52	0.480906	0.480906	-0.443945	-0.443945
53	0.485915	0.485915	-0.448227	-0.448227
54	0.490862	0.490862	-0.452443	-0.452443
55	0.495747	0.495747	-0.456591	-0.456591
56	0.500566	0.500566	-0.460669	-0.460669
57	0.505317	0.505317	-0.464675	-0.464675
58	0.509999	0.509999	-0.4686075	-0.468608
59	0.514610	0.514610	-0.472465	-0.472465
60	0.519147	0.519147	-0.476246	-0.476246

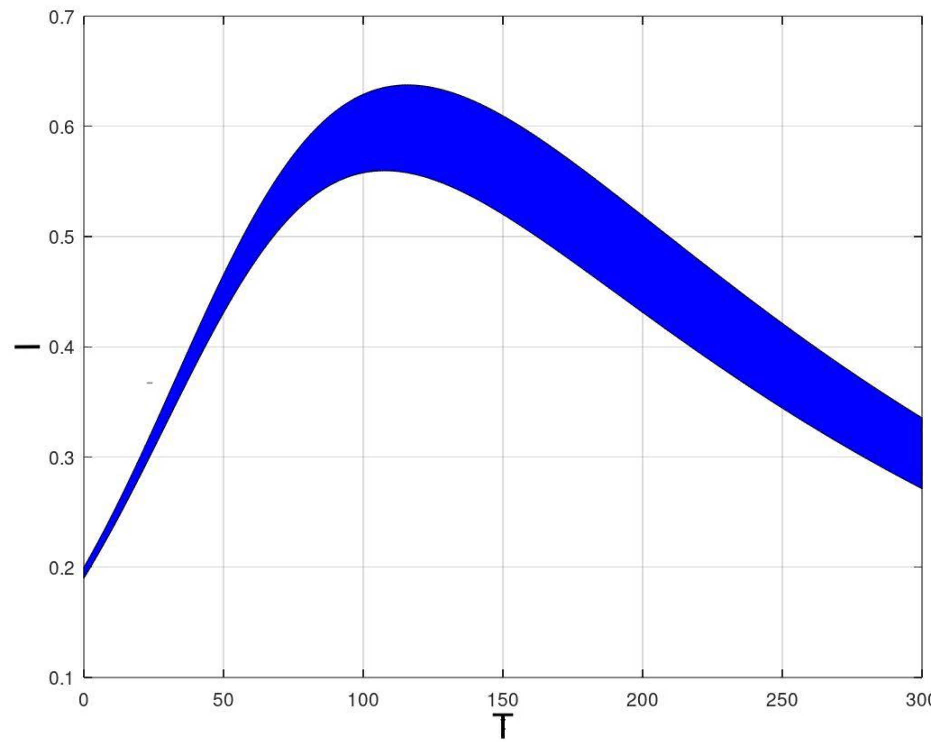
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Results: SIR Model

Kaa



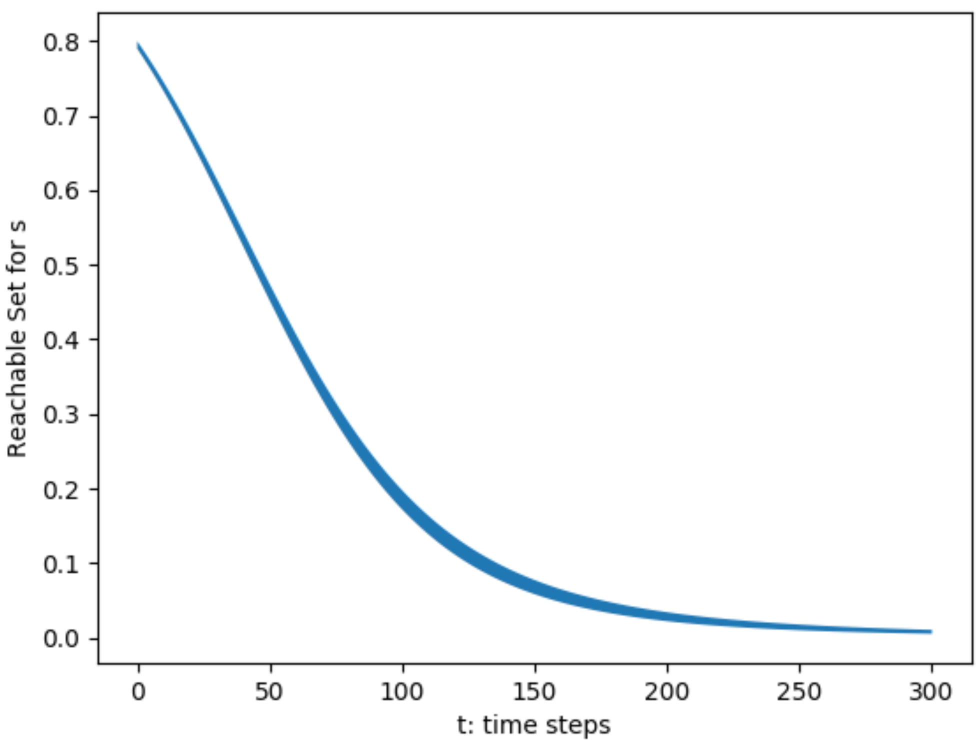
Sapo



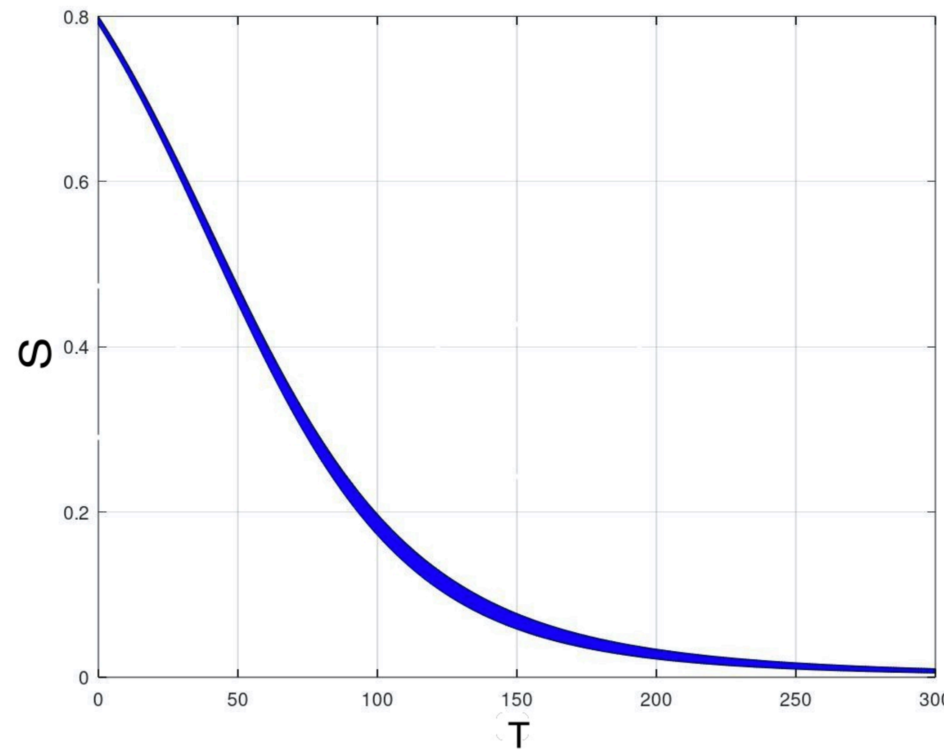
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Results: SIR Model

Kaa



Sapo



Results: Rossler Model

- The Rossler model is another 3-dimensional system governed under the dynamics:

.

$$x_{k+1} = x_k + -(y - z)\Delta$$
$$y_{k+1} = y_k + (x_k + a y_k)\Delta$$
$$z_{k+1} = z_k + (b + z_k(x_k - c))\Delta$$

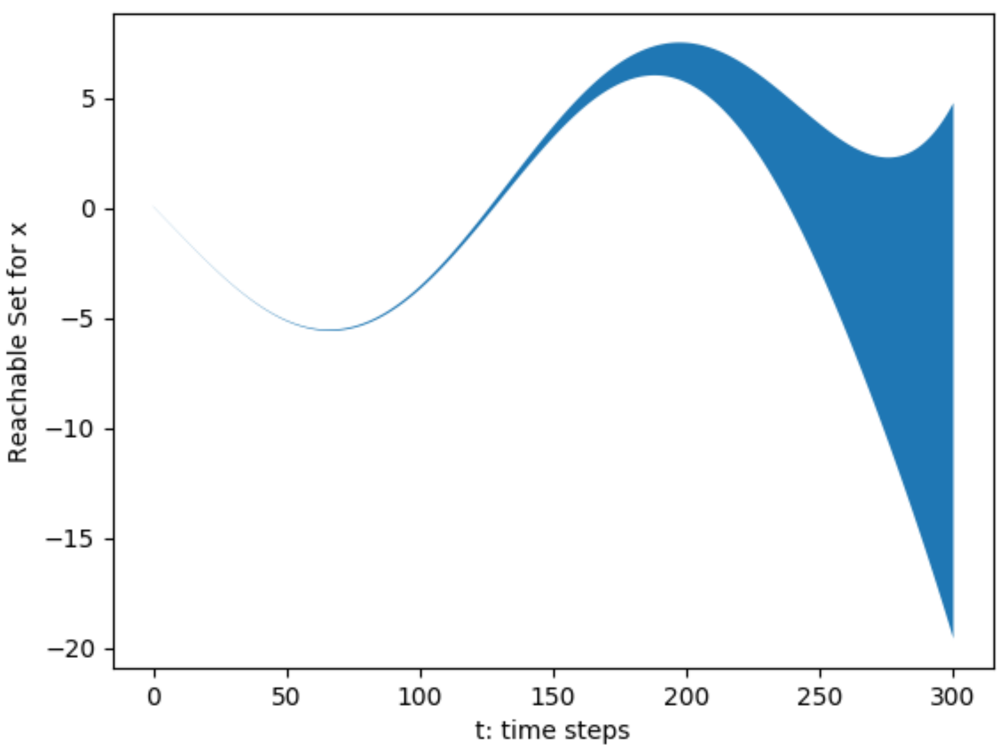
$a = 0.1$, $b = 0.1$, $c = 14$
$\Delta = 0.025$

Upper and lower offsets for variable y

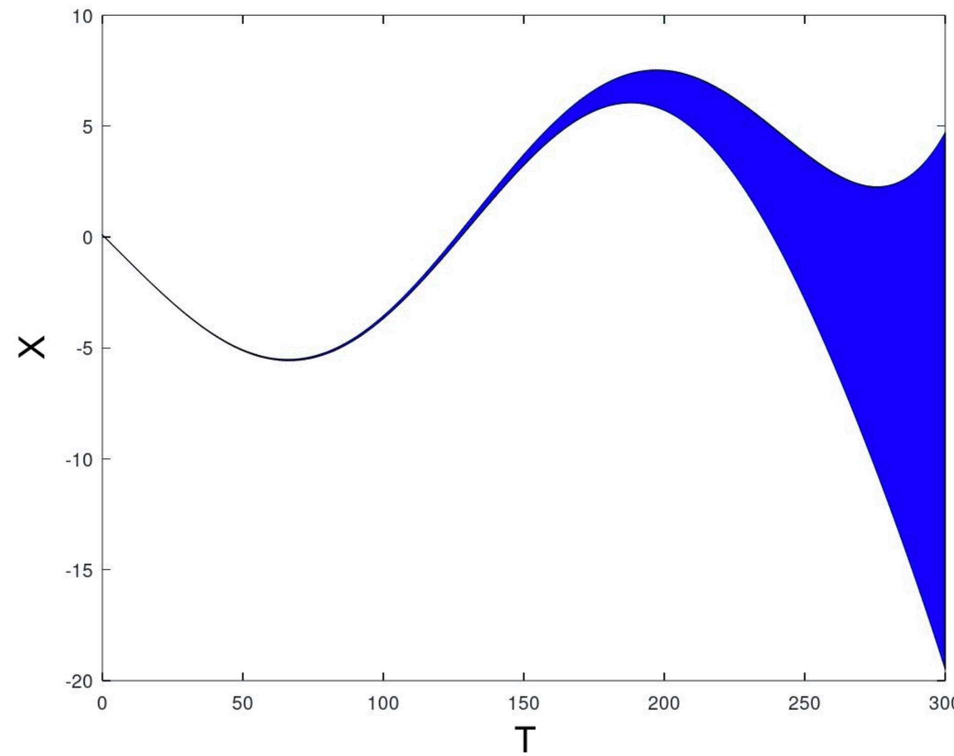
Time Steps	Kaa (offu)	Sapo (offu)	Kaa (offl)	Kaa (offl)
50	1.95908	1.96043	-1.9209	-1.9193
51	1.83552	1.83688	-1.7963	-1.7947
52	1.71044	1.71181	-1.67016	-1.6685
53	1.58392	1.58531	-1.54255	-1.5409
54	1.45604	1.45744	-1.41355	-1.4119
55	1.32687	1.32829	-1.28324	-1.2815
56	1.19649	1.19792	-1.15168	-1.1500
57	1.06499	1.06642	-1.01896	-1.0172
58	0.932432	0.933877	-0.885157	-0.88339
59	0.798905	0.800358	-0.75035	-0.74857
60	0.664489	0.665949	-0.614619	-0.61282

Results: Rossler Model

Kaa



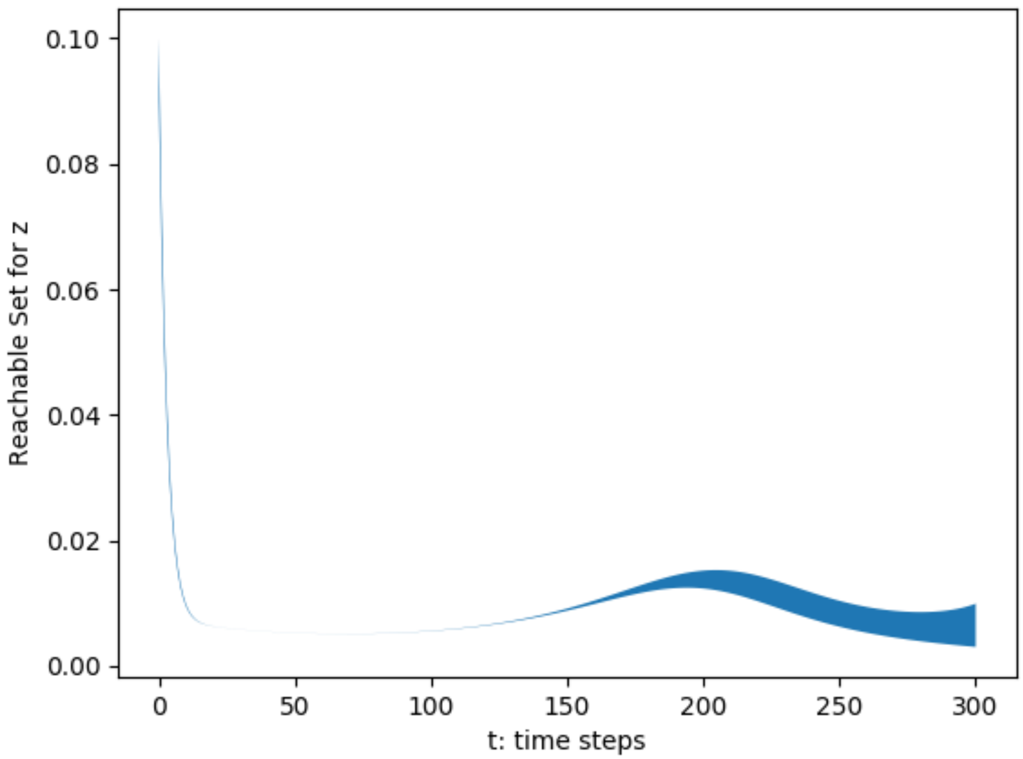
Sapo



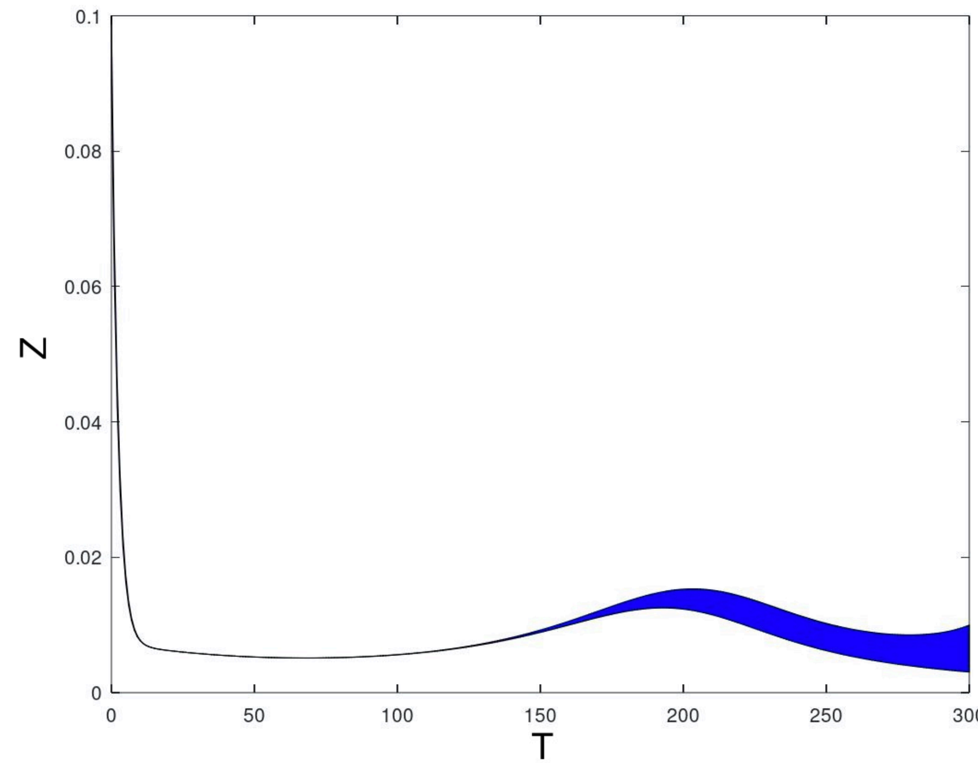
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Results: Rossler Model

Kaa



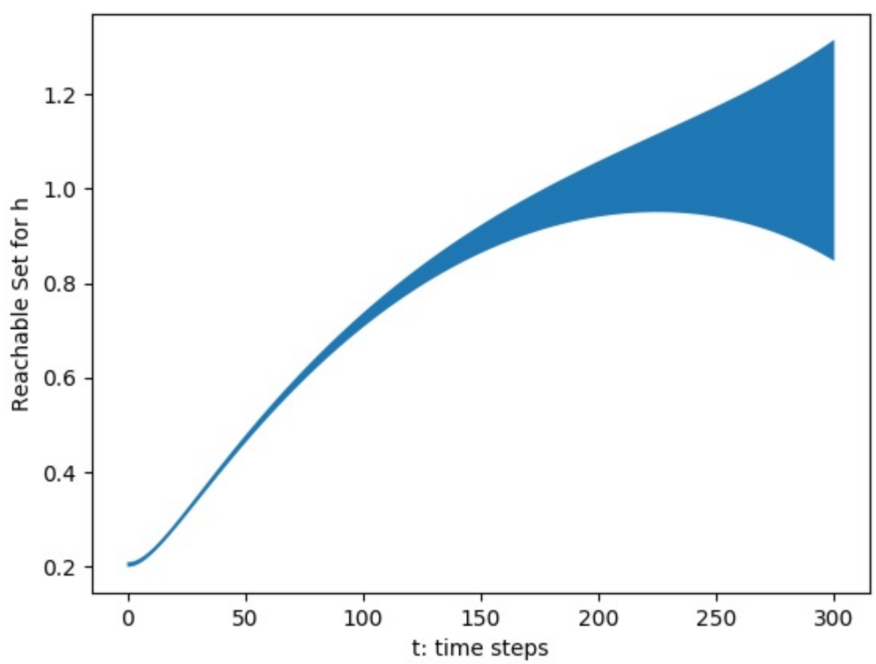
Sapo



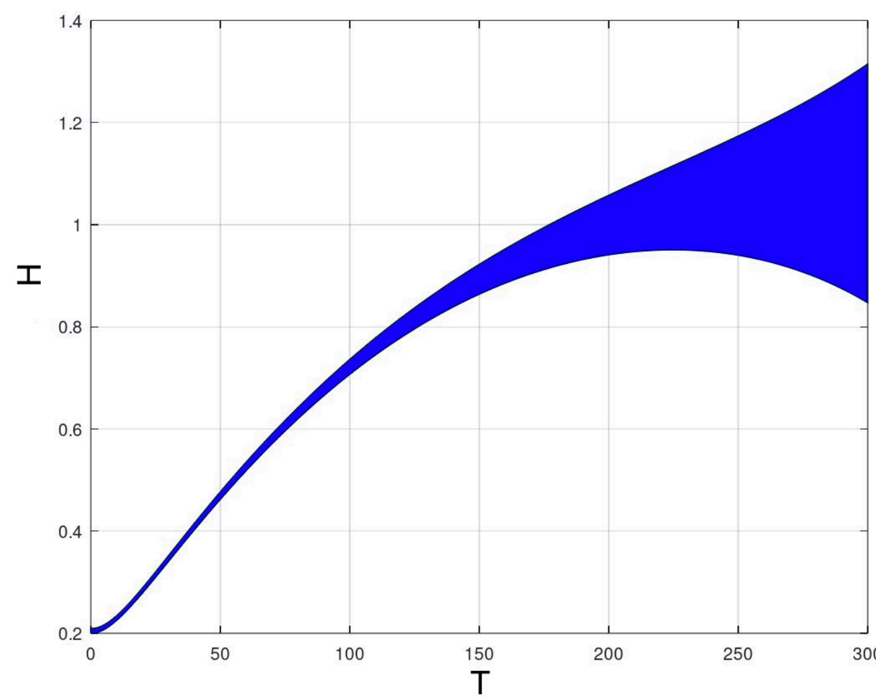
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Results: Quadcopter Model

Kaa



Sapo



THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Performance Drawbacks

- While the current implementation in Python is very intuitive and concise, it incurs severe performance penalties. We believe this is due to some extraneous library calls in the core loop of the reachable set computation.

- An immediate next step is to deploy extensive profiling to find performance bottlenecks and subsequently improve on them.

Model	**Kaa**	**SAPO** (C++)
SIR	11.41 sec	0.16 sec
Rossler	41.92 sec	1.17 sec
Quadcopter	78.21 sec	11.98 sec
Lotka-Volterra	18 min 95.05 sec	57.48 sec
Phosphoraley	103.81 sec	24.86 sec

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Conclusions

- We present Kaa, a Python implementation of reachable set computation of nonlinear systems which is focused towards accessibility and pedagogical use.

- We include Juypter Notebooks and documentation through: https://github.com/Tarheel-Formal-Methods/kaa

- While we do incur performance drawbacks from selecting Python for implementing this algorithm, we believe that it aids in fast prototyping and enables students to easily build on top of the library.

- Immediate future work includes improving on the running time and creating a more streamlined format for defining models and visualizing reachable sets.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

References

- Dreossi, T.: Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. In: Proceedings of the 20th International Conference on Hybrid Systems: Computationand Control. pp. 29–34 (2017)
- Dreossi, T., Dang, T., Piazza, C.: Parallelotope bundles for polynomial reachability. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. pp.297–306 (2016)