EPiC Series in Computer Science

Volume 34, 2015, Pages 89–97

ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems

Benchmark: A Nonlinear Reachability Analysis Test Set from Numerical Analysis

Hoang-Dung Tran[1], Luan Viet Nguyen[1], and Taylor T. Johnson[1]

University of Texas at Arlington, USA

Abstract

The field of numerical analysis has developed numerous benchmarks for evaluating differential and algebraic equation solvers. In this paper, we describe a set of benchmarks commonly used in numerical analysis that may also be effective for evaluating continuous and hybrid systems reachability and verification methods. Many of these examples are challenging and have highly nonlinear differential equations and upwards of tens of dimensions (state variables). Additionally, many examples in numerical analysis are originally encoded as differential algebraic equations (DAEs) with index greater than one or as implicit differential equations (IDEs), which are challenging to model as hybrid automata. We present executable models for ten benchmarks from a test set for initial value problems (IVPs) in the SpaceEx format (allowing for nonlinear equations instead of restricting to affine) and illustrate their conversion to several other formats (dReach, Flow*, and the MathWorks Simulink/Stateflow [SLSF]) using the HyST tool. For some instances, we present successful analysis results using dReach, Flow*, and SLSF.

Category: academic **Difficulty:** low through challenge

1 Context and Origins

Verification and validation are important tasks that are applied broadly in many fields in recent years such as embedded systems, power electronics, networked control systems, and aerospace systems [4, 16, 17]. Many different verification methods and tools have been developed for reachability analysis of hybrid systems [2, 3, 7, 14]. The challenges in verification of continuous and hybrid systems are many, and include for example complex nonlinear dynamics, high-dimensional state-spaces, and bounded vs. unbounded time. To evaluate novel verification methods and tools, we need to evaluate and test them using a variety of diverse benchmarks, that are ideally standardized. However, these benchmarks are not standardized, so it is difficult to evaluate whether particular state representations (e.g., zonotopes [1], Taylor models [7], support functions [13], polyhedra, hypercubes [5], symbolic/SMT formulas [14], etc.) and verification techniques are superior for different classes of hybrid automata.

In this paper, we present a set of ten different, executable benchmarks to aid in the development of a standardized set of benchmarks for the verification community to evaluate verification methods and tools. These benchmarks are derived from a test set for initial value problem (IVP)

solvers from numerical analysis [18,19], and include systems modeled by nonlinear ordinary differential equations (ODEs) and differential-algebraic equations (DAEs). While some of these benchmarks are standard and well-known in the hybrid systems verification community (e.g., the Van der Pol oscillator), the majority (to the best of the authors' knowledge) have not previously been considered for benchmarking hybrid systems reachability tools, although similar recent initiatives are ongoing [8]. This test set and others are used as standardized benchmarks to compare IVP solvers and helped move numerical analysis into the mainstream [10, 11, 15], and this was one of the original goals in developing standardized benchmarks: "The problems, methods and comparison criteria are specified very carefully. One objective in doing so is to provide a rigorous conceptual basis for comparing methods. Another is to provide a useful standard for such comparisons" [15]. We then hope that the development of similar standard benchmark sets for verification purposes will help move reachability analysis and verification into industrial adoption.

The benchmarks are shown in Table 2.2 and come from a variety of fields, including biology, environmental science, celestial mechanics, chemistry, and electronics. The benchmarks are independent from a specific approach to evaluate reachability algorithms, as they have different classes of nonlinear dynamics, dimensionality, etc. All the benchmarks are purely continuous, and do not include any hybrid or switched behavior. Most of them contain highly stiff nonlinear differential equations and a high number of state variables, which make them challenging to analyze with existing techniques and tools, but they may serve as benchmarks to evaluate the next generation of techniques and tools. All the benchmarks are IVPs and specify (1) the initial conditions, (2) the ODEs or DAEs, (3) a final state, and (4) the time to reach the final state; see Table A.1. For reachability analysis and safety verification, the final state—or actually a neighborhood about the final state to avoid minor numerical issues—specifies the bad set of states, and reachability from the initial state to the final state may be checked (either using the time or not). All of benchmarks are first described in the input format for SpaceEx[1], and are then translated to other formats including dReach, Flow*, and Matlab Simulink/Stateflow (SLSF) using the HyST model transformation tool [6]. To validate the conversion of the benchmarks from their original descriptions (from the paper describing them [18] and some Fortran code [19]) to hybrid automata, simulations in Matlab were conducted of equivalent continuous-time SLSF charts (generated using HyST), and compared to existing simulation results [18,19]. Additionally, several benchmarks are analyzed using Flow* and dReach. The problems are diverse, with the number of state variables varying from two to twenty eight, so these benchmarks may be useful to evaluate reachability algorithms, verification methods, and tools.

2 Brief Descriptions

For brevity, we do not describe in detail all the benchmarks in Table 2.2, but refer to their origins and detailed mathematical definitions in [18, 19].[2] We provide executable models (as SpaceEx hybrid automata, SLSF, and the other formats supported by HyST) for all the benchmarks in the supplementary material. In this section, we focus on presenting the Chemical Akzo Nobel problem taken from the test set in [18,19] as it is a nonlinear DAE systems and is nontrivial to model as hybrid automata.

[1] This allows for nonlinear functions, which may be specified and parsed by SpaceEx, but not analyzed as only affine functions are supported.

[2] The executable models are included on the ARCH website and are also available online from the HyST website at: http://verivital.com/hyst/.

Chemical Akzo Nobel problem: This is a chemical process problem given by Akzo Nobel (Central Research, Arnhem, Netherlands) where two species, FLB and ZHU, are mixed while CO2 is continuously injected.

The mathematical description of the problem is a set of six non-linear DAEs of index one, where x_i, $i \leq 6$ represent the concentrations of $[FLB]$, $[CO_2]$, $[FLBT]$, $[ZHU]$, $[ZLA]$, $[FLB.ZHU]$, respectively:

$$M\frac{dx}{dt} = f(x), \quad x(0) = x_0, \quad x \in R^6, \quad 0 \leq t \leq 180,$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad f(x) = \begin{pmatrix} -2r_1 + r_2 - r_3 - r_4 \\ -\frac{1}{2}r_1 - r_4 - \frac{1}{2}r_5 + F_{in} \\ r_1 - r_2 + r_3 \\ -r_2 + r_3 - 2r_4 \\ r_2 - r_3 + r_5 \\ K_s x_1 x_4 - x_6 \end{pmatrix},$$

where the r_i and F_{in} are auxiliary variables defined as

$$r_1 = k_1 \cdot x_1^4 \cdot x_2^{\frac{1}{2}}, \quad r_2 = k_2 \cdot x_3 \cdot x_4, \quad r_3 = \frac{k_2}{K} \cdot x_1 \cdot x_5$$
$$r_4 = k_3 \cdot x_1 \cdot x_4^2, \quad r_5 = k_4 \cdot x_6^2 \cdot x_2^{\frac{1}{2}}, \quad F_{in} = klA \cdot \left(\frac{p(\mathrm{CO}_2)}{H} - x_2\right).$$

The values of the parameters $k_1, k_2, k_3, k_4, K, klA, p(CO_2)$ and H are presented in Table 2.1.

The problem is specified as a nonlinear DAE. However, most of verification tools support only ODEs[3]. Thus, to model the chemical Akzo Nobel problem as a hybrid automaton, we can convert the original DAEs to equivalent ODEs. There is a general solution to convert linear DAEs to linear ODEs that can be easily modeled as a hybrid automaton. However, it is generally more complicated for nonlinear DAEs. In this problem, it is simple to convert the original DAEs to equivalent ODEs by taking the derivative of the sixth sub-equation to obtain $\dot{x}_6 = K_s(x_1\dot{x}_4 + x_4\dot{x}_1)$. Table 2.2 gives a brief overview of all problems successfully converted from the test set.

3 Simulations and Reachability Analysis

To validate all models in the test set, we first run simulation of the SLSF models in Matlab to check if the systems behaviors and final solutions at the end of the running time are similar to those in the original test set [18, 19]. The simulation is run using Matlab 2014a on a personal computer with the following configuration: Intel (R) Core(TM) i7-2677M CPU at 1.80GHz, 4GB RAM, and 64-bit Window 7. A virtual machine running Ubuntu on the same computer

$k_1 = 18.7$	$k_4 = 0.42$	$K_s = 115.83$
$k_2 = 0.58$	$K = 34.4$	$p(CO_2) = 0.9$
$k_3 = 0.09$	$klA = 3.3$	$H = 737$

Table 2.1: The values of the parameters in Akzo Nobel problem.

[3]The naive approach is to specify the algebraic constraint as the invariant, but this does not work for many tools, e.g., Flow* and dReach, although this may be accommodated in SpaceEx using uncontrolled variables.

No.	Name	Class	n	Field	SpaceEx	SLSF	Flow*	dReach
1	Hires	NL ODE	8	Chemistry	×	+	+	+
2	Pollution	NL ODE	20	Environment	×	+	+	+
3	Ring modulator	NL ODE	15	Electric	×	+	+	+
4	OREGO	NL ODE	3	Chemistry	×	+	+	+
5	ROBER	NL ODE	3	Chemistry	×	+	+	+
6	E5	NL ODE	4	Chemistry	×	+	+	+
7	Akzo Nobel	NL DAE index 1	6	Chemistry	×	+	+	+
8	VDPOL	NL ODE	2	Electric	×	+	+	+
9	Small Circuit	NL DAE index 1	4	Electric	×	+	+	+
10	Pleiades	NL ODE	28	Mechanics	×	+	+	+

Table 2.2: Overview of the benchmark problems, where NL is short for nonlinear, n is the number of variables, × specifies that the model is incompatible with a particular tool at this time (e.g., due to nonlinear dynamics), and + specifies that the model file successfully parses and analysis starts (e.g., is executable). However, the large final time makes analysis of them challenging in many cases (see Table A.1).

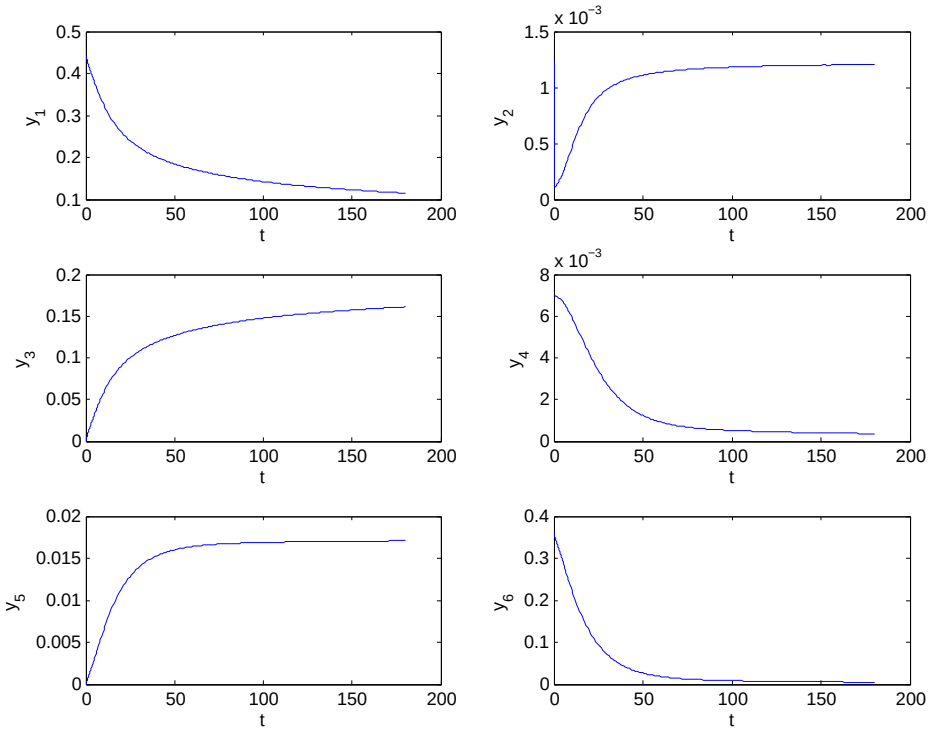

Figure 3.1: SLSF simulations of the chemical Akzo Nobel problem.

with 1GB RAM was used for Flow* and dReach. All the models are executable, and most of them are simulated with results similar to those in the original test set. However, some of them, such as Ring Modulator and E5, cannot be completely simulated with the full running time as the original test set due to the high stiffness of these systems. Since the solutions of stiff

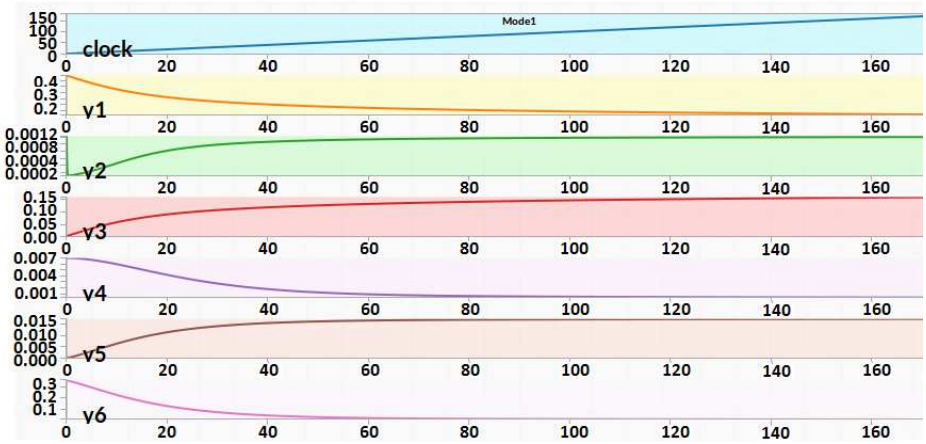

Figure 3.2: Analysis of the chemical Akzo Nobel problem in dReach showing the final state (x_β) is reachable (modulo δ-decidability) from the initial state (x_α) at time β.

differential equations can change on a very small time scale, it is hard to determine appropriate values for relative and absolute tolerances even when using an appropriate stiff solver (we use ode15s solver in Matlab Simulink). The second step for model validation is to analyze the problems using the Flow* and dReach verification tools. Notice again that, because of the error of numerical solvers, we cannot use the specific final solutions in the simulation step as a point to check reachability. Instead, we use a hyper-rectangle around the final solution computed numerically. An overview of the simulations and analysis results of the test set are shown in Table 2.2. The executable model files and additional reachability and simulation results from Matlab SLSF of all the problems are available in the supplementary materials. After selecting appropriate parameters, all the benchmarks can be executed in Flow* and dReach. However, some instances cannot fully be analyzed to show that the final state (Table A.1) is reachable. This may be caused by some of the following reasons: (a) due to computational processing and memory limits on the evaluation computer, the tools may run out of memory or may not terminate with the parameters provided in a given time threshold, (b) the high stiffness of some instances is a challenge, specifically to identify an appropriate time step size that yields termination in a given time threshold, (c) the final time is very long in many cases, or (d) we did not select ideal parameters. Next, we present some analysis for the Chemical Akzo problem using dReach, which was successfully analyzed in both dReach and Flow* to show that the final state is reachable at the final time.

Chemical Akzo Nobel problem: The chemical Akzo Nobel problem is first simulated using Matlab SLSF. The behavior of the system is shown in Figure 3.1. Its analysis, shown in Figure 3.2, is then done using dReach, where the final state is defined as the following: $\forall x_i \in x_\beta,\ i \leq 6$, then $x_i \in \epsilon_i,\ \epsilon_i \subset \Upsilon$, where $\Upsilon = \langle\ [0.1, 0.12],\ [0.001, 0.0015],\ [0.14, 0.18],\ [0.0003, 0.0004],\ [0.0015, 0.0019],\ [0.003, 0.006]\rangle$ is an interval vector of variable values at the final state.

4 Key Observations

Converting the test set to hybrid automata input formats and validating the conversion highlighted several areas for improvement in existing tools and gave rise to several interesting technical problems. First, the long running time of the examples was challenging for the tools, as seen by the lack of many successful reachability analyses in Table 2.2. While parameter tuning for each of tool may make analysis feasible, this requires significant manual effort.

DAE to ODE Conversion and Tool Semantics Differences. Is there a general solution for modeling DAEs using hybrid automata? If this could be done, we could model many problems in different fields such as mechanical, biochemical, and electrical engineering as hybrid automata. The basic idea to do this is to convert the DAEs into ODEs and algebraic constraints that can be described as the flows and invariants of a hybrid automaton. The SpaceEx modeling language supports uncontrolled variables for modeling the algebraic variables and constraints, and this means algebraic variables can take any value satisfying their constraints over the evolution of the system (i.e., over the intervals of time defining trajectories).

Unfortunately, SpaceEx only supports hybrid systems with piecewise affine dynamic, meanwhile all the problems in the test set are nonlinear. In SpaceEx, one method is to model semi-explicit linear DAEs in terms of hybrid input/output automata (HIOA) [9]. Other tools like dReach and Flow* support reachability analysis for nonlinear hybrid systems. However, it is still a challenge to analyze nonlinear DAEs using these tools, since they do not have the above benefit in modeling uncontrolled variables like SpaceEx. These tools consider all variables as controlled variables that have to be defined by ODEs in the flows component of the hybrid automaton tuple. This means that modeling the algebraic constraint as an invariant is typically infeasible. Thus, these tools do not support the capability of modeling DAEs directly (as they do not support uncontrolled variables and some require a full assignment of the flows, e.g., every variable must be associated with an explicit flow), so a conversion from DAEs to ODEs may be necessary. Thus, in order to analyze nonlinear DAEs included in the test set using these tools, we need to convert to equivalent ODEs. Moreover, this approach can totally be applied for linear DAEs by taking the derivative of the linear constraints, but it is inapplicable in general when the constraints of the DAEs are nonlinear. As a solution for this problem, the level set method has been proposed to compute the reachable sets of hybrid systems with semi-explicit index one DAEs [20].

Standardized Test Sets for Reachability Analysis. As in the simulation and numerical analysis community [10, 11, 15], it is essential to build a standard test set for evaluating reachability analysis methods and their implementations in software tools. The executable models provided along with this paper may help researchers test and compare their methods to existing results effectively and quickly without incurring additional programming workload. This may also help industrial adoption, as engineers can more easily evaluate and select a suitable verification method for their class of problems. Though this benchmark starts with only some ODE and DAE problems, it is still useful for testing and comparing verification methods because of the problems' diversity, various number of variables, and long running time. While all benchmarks in this paper are one location hybrid automaton models, there are many other interesting hybrid systems benchmarks [8, 12], and standardizing a diverse set is critical.

5 Outlook

Overall, these verification benchmarks represent a diverse set from easy through challenging and may serve to develop a standard benchmark library to evaluate reachability and verification methods for nonlinear dynamics. The benchmarks range in dimensionality, dynamics types, and come from many different domains. The continuous and hybrid verification community may use these benchmarks for comparing methods and tools. In ongoing and future work, we intend to collect additional benchmarks [10, 11, 15], including ones originally encoded as DAEs with index greater than one and IDEs listed in [18, 19].

References

[1] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

[2] R. Alur. Formal verification of hybrid systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 273–278, 2011.

[3] Rajeev Alur, Thao Dang, and Franjo Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *ACM Trans. Embed. Comput. Syst.*, 5(1):152–199, February 2006.

[4] E. Haxthausen Anne and Jan Peleska. Formal development and verification of a distributed railway control system. *Software Engineering, IEEE Transactions on*, 26(9):687–701, 2000.

[5] Stanley Bak. Reducing the wrapping effect in flowpipe construction using pseudo-invariants. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, pages 40–43, New York, NY, USA, 2014. ACM.

[6] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HyST: A source transformation and translation tool for hybrid automaton models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[7] X. Chen, Erika Abraham, , and Sriam Sankaranarayanan. Talyor model flowpipe construction for non-linear hybrid systems. In *IEEE Real-Time Systems Symposium*, pages 183–192, 2012.

[8] Xin Chen, Stefan Schupp, Ibtissem Ben Makhlouf, Erika Abraham, Goran Frehse, and Stefan Kowalewski. A benchmark suite for hybrid systems reachability analysis. In *7th NASA Formal Methods Symposium*, 2015.

[9] Alexandre Donzé and Goran Frehse. Modular, hierarchical models of control systems in spaceex. In *Proc. European Control Conf. (ECC'13)*, Zurich, Switzerland, 2013.

[10] W. H. Enright and J. D. Pryce. Two fortran packages for assessing initial value methods. *ACM Trans. Math. Softw.*, 13(1):1–27, March 1987.

[11] W.H. Enright, T.E. Hull, and B. Lindberg. Comparing numerical methods for stiff systems of o.d.e:s. *BIT Numerical Mathematics*, 15(1):10–48, 1975.

[12] Ansgar Fehnker and Franjo Ivancic. Benchmarks for hybrid systems verification. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control (HSCC '04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 326–341. Springer Berlin Heidelberg, 2004.

[13] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[14] Sicun Gao, Soonho Kong, Wei Chen, and Edmund M. Clarke. Delta-complete analysis for bounded reachability of hybrid systems. *CoRR*, abs/1404.7171, 2014. Available at http://arxiv.org/abs/1404.7171.

[15] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick. Comparing numerical methods for ordinary differential equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637, 1972.

[16] Taylor T. Johnson, Jeremy Green, Sayan Mitra, Rachel Dudley, and Richard Scott Erwin. Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In Dimitra Giannakopoulou and Dominique Méry, editors, *Proceedings of the 18th International Conference on Formal Methods (FM 2012)*, volume 7436, pages 252–266. Springer Berlin Heidelberg, Paris, France, August 2012.

[17] Taylor T. Johnson, Zhihao Hong, and A. Kapoor. Design verification methods for switching power converters. In *Power and Energy Conference at Illinois (PECI), 2012 IEEE*, pages 1–6, February 2012.

[18] Francesca Mazzia and Cecilia Magherini. Test set for initial value problem solvers, release 2.4. Technical Report 4, Department of Mathematics, University of Bari, Italy, February 2008. Available at `http://pitagora.dm.uniba.it/~testset`.

[19] Francesca Mazzia and Cecilia Magherini. *Test Set for Initial Value Problem Solvers, release 2.4*. Department of Mathematics, University of Bari and INdAM, Research Unit of Bari, February 2008. Available at `http://www.dm.uniba.it/~testset`.

[20] Ian M.Mitchell and Yoshihiko Susuki. Level set methods for computing reachable sets of hybrid systems with differential algebraic equation dynamics. In *Hybrid Systems: Computation and Control. Lecture Notes in Computer Science*, volume 4981, pages 630–633, 2008.

A Appendix: Initial and Final States

Table A.1 shows, for each example, the initial conditions, final (or goal) states, and a specific time at which to reach the goal states (e.g., a final or goal time) from the initial conditions. In Table A.1, $[\alpha, \beta]$ denotes a running time interval between the initial state and the final state, where α is the initial time and β is the final time (i.e., the time at which the final state should be reachable from the initial state). Here, x_α, x_β are n-vector of the solutions of the state variables at initial time α and at final time β of a simulation, respectively. As these results come from numerical simulators, one may need to create a hyper-rectangle around the final state to create a neighborhood near the final state that is reachable at the final time.

No.	**Name**	$[\alpha, \beta]$	**IC**, **FS**	**Value**
1	Hires	$[0, 321.81]$	x_α	$(1, 0, 0, 0, 0, 0, 0, 0.0057)^T$
			x_β	$(0.737 \cdot 10^{-3}, 0.144 \cdot 10^{-3}, 0.589 \cdot 10^{-4}, 0.118 \cdot 10^{-2}, 0.239 \cdot 10^{-2}, 0.624 \cdot 10^{-2}, 0.285 \cdot 10^{-2}, 0.285 \cdot 10^{-2})^T$
2	Pollution	$[0, 12]$	x_α	$(0, 0.2, 0, 0.04, 0, 0, 0.1, 0.3, 0.01, 0, 0, 0, 0, 0, 0, 0, 0, 0.007, 0, 0)^T$
			x_β	$(0.056, 0.134, 0.414 \cdot 10^{-8}, 0.006, 0.202 \cdot 10^{-6}, 0.147 \cdot 10^{-6}, 0.078, 0.324, 0.007, 0.162 \cdot 10^{-7}, 0.114 \cdot 10^{-7}, 0.002, 0.002, 0.139 \cdot 10^{-4}, 0.008, 0.435 \cdot 10^{-17}, 0.007, 0.1008 \cdot 10^{-3}, 0.177 \cdot 10^{-5}, , 0.568 \cdot 10^{-4})^T$
3	Ring Modulator	$[0, 0.001]$	x_α	$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T$
			x_β	$(-0.023, -0.007, 0.258, -0.406, -0.404, -0.261, 0.1107, 0.294 \cdot 10^{-6}, -0.284 \cdot 10^{-7}, 0.727 \cdot 10^{-3}, 0.793 \cdot 10^{-3}, -0.726 \cdot 10^{-3}, -0.794 \cdot 10^{-3}, -0.709 \cdot 10^{-4}, 0.239 \cdot 10^{-4})^T$
4	OREGO	$[0, 360]$	x_α	$(1, 2, 3)^T$
			x_β	$(1.0008, 1228.178, 132.055)^T$
5	ROBER	$\left[0, 10^{11}\right]$	x_α	$(1, 0, 0)^T$
			x_β	$(0.208 \cdot 10^{-7}, 0.833 \cdot 10^{-13}, 0.999)^T$
6	E5	$\left[0, 10^{13}\right]$	x_α	$(1.76 \cdot 10^{-3}, 0, 0, 0)^T$
			x_β	$(0.115 \cdot 10^{-290}, 0.887 \cdot 10^{-22}, 0.885 \cdot 10^{-22}, 0)^T$
7	Akzo Nobel	$[0, 180]$	x_α	$(0.444, 0.001, 0, 0.007, 0, 0.359)^T$
			x_β	$(0.115, 0.001, 0.161, 0.0003, 0.017, 0.004)^T$
8	VDPOL	$[0, 2000]$	x_α	$(2, 0)^T$
			x_β	$(1.171, -0.893 \cdot 10^{-3})^T$
9	Small Circuit	$[0, 20]$	x_α	$(0, 0, 0, 0)^T$
			x_β	$(-0.99, 0.0001, 0.99, -0.99)^T$
10	Pleiades	$[0, 2]$	x_α	$(3, 3, -1, -3, 2, -2, 2, 3, -3, 2, 0, 0, -4, -4, 0, 0, 0, 0, 0, 1.75, -1.5, 0, 0, 0, -1.25, 1, 0, 0)^T$
			x_β	$(0.371, 3.247, -3.222, 0.66, 0.3426, 1.562, -0.7, -3.944, -3.271, 5.225, -2.591, 1.198, -0.243, 1.091, 3.417, 1.355, -2.59, 2.025, -1.156, -0.807, 0.595, -3.741, 0.377, 0.939, 0.367, -0.347, 2.344, -1.947)^T$

Table A.1: Configurations for reachability analysis using the test set, specifying the initial conditions (x_α), final states (x_β), and the time at which the final state is reachable (β) when starting from the initial time (α).