Reachability Analysis of FMI Models Using Data-Driven Dynamic Sensitivity

Journal Title
XX(X):1–17

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Sergiy Bogomolov[1], Cláudio Gomes[2], Carlos Isasa[2], Sadegh Soudjani[1], Paulius Stankaitis[1], Thomas Wright[3]

Abstract

Digital Twin is a technology that facilitates a real-time coupling of a cyber-physical system and its virtual representation. The technology is applicable to a variety of domains and facilitates more intelligent and dependable system design and operation, but it relies heavily on the existence of digital models that can be depended upon. In realistic systems, there is no single monolithic digital model of the system. Instead, the system is broken into subsystems, with models exported from different tools corresponding to each subsystem.

In this paper, we focus on techniques that can be used for a black box model, such as the ones implementing the Functional Mock-up Interface (FMI) standard, formal analysis, and verification. We propose two techniques for simulation-based reachability analysis of models. The first one is based on system dynamics, while the second one utilises dynamic sensitivity analysis to improve the quality of the results.

Our techniques employ simulations to obtain the model's sensitivity with respect to the initial state (or model's Lipschitz constant) which is then used to compute reachable states of the system. The approaches also provide probabilistic guarantees on the accuracy of the computed reachable sets that are based on simulations. Each technique requires different levels of information about the black box system, allowing the readers to select the best technique according to the capabilities of the models.

The validation experiments have demonstrated that our proposed algorithms compute accurate reachable sets of stable and unstable linear systems. The approach based on dynamic sensitivity provides an accurate and, with respect to system dimensions, more scalable approach, while the sampling-based method allows a flexible trade-off between accuracy and runtime cost. The validation results also show that our approaches are promising even when applied to non-linear systems, especially, when applied to larger and more complex systems. The reproducibility package with code and data can be found at https://github.com/twright/SIM-Black-Box-Reachability.

Keywords

Reachability Analysis, Digital Twins, Functional Mock-up Interface, Dynamic Sensitivity Equations, Lipschitz constant

Introduction

Digital Twins (DT) are an emerging technology that makes it possible to monitor, optimise and control cyber-physical assets using their virtual representation (kept as a mirror of reality) in real-time[1]. They provide critical services such as state estimation, visualisation, what-if analysis, anomaly detection, and self-adaptation.

Because DT services rely heavily on the existence of models of the cyber-physical systems[2,3] (CPS), the dependability of the DT is a direct consequence of how much we can depend upon the models' simulation. For example, prior to adapting the controller of the CPS, the DT needs to find the optimal *and safe* configuration by, e.g., running simulations with alternative configurations on future predicted scenarios, while checking that safety properties are satisfied. If there is uncertainty in the model parameters, as there often is in continuous and hybrid system models whose parameters are identified from sensor data, then we may be interested in computing bounds that enclose all simulation results, based on the possible parameter values, in a technique called *reachability analysis*. An introduction and survey of the topic of reachability analysis are provided in[4] and an example application for DTs is presented in[5].

To compute reachable states of the system generally requires knowing a model of the system, which for CPSs can be hard to obtain or even unavailable because of the myriad of modelling and simulation tools used in engineering practice. Fortunately, the industry has formulated standards that make it possible to represent and integrate black box, IP-protected models. One such standard is the Functional Mock-up Interface (FMI)[6], which is currently supported by more than 150 tools. Because of these reasons, in this paper, we focus on a class of reachability analysis techniques that are data-driven (i.e., they rely on data generated from simulations), which can be applied to black box models. Even

[1]University of Newcastle, UK
[2]Aarhus University, Denmark
[3]University of York, UK

Corresponding author:
Paulius Stankaitis, School of Computing, Newcastle University, Newcastle upon Tyne, UK
Email: paulius.stankaitis@ncl.ac.uk

though several data-driven reachability analysis approaches have been proposed in the literature, they either do not provide probabilistic guarantees on the completeness of the exploration, or discuss handle coupled models.

Contribution. In this paper, we build upon our previous work [7] and propose a new method for computing reachable states of black-box coupled models. This reachability analysis method leverages advanced FMI standard functionality for retrieving partial derivatives of Functional Mock-up Unit (FMU) variables and numerical differential system solvers to solve dynamic sensitivity equations, which describe system sensitivity to changes in their initial conditions. The computed maximum sensitivity provides a scaling factor which together with a nominal initial state space trajectory is used to compute approximate reachable sets.

In summary, the novel contributions of this paper are: (1) a dynamic sensitivity-based reachability analysis method of black-box models and (2) a method for composing dynamic sensitivity equation systems from coupled models implementing the FMI standard. The paper evaluates the new approach against our previously introduced data-driven method [7] by comparing reachable sets computed for linear and non-linear dynamical systems. We also validate our approaches against a leading model-based reachability analysis tool — Flow* [8].

The paper is structured as follows. The following Related Work section, discusses related work and positions our reachability analysis approach. After that, our paper describes the preliminaries and the problem statement of the paper. The main contributions of the paper are presented in the Reachability Algorithms section in which we formally describe our proposed reachability analysis and dynamic sensitivity equation composition algorithms. The Validation Experiments section describes results obtained from comparing and validating algorithms, as well as discusses limitations and recommendations of the proposed methods. In the final section, we summarise our findings and propose directions for future work.

Related Work

This paper extends our previous work [7], where we proposed a data-driven method for computing the reachable states of black-box models with probabilistic accuracy guarantees, given a sufficient number of samples is used. This reachability method was based on estimating a maximum Lipschitz constant by simulating a model from independent and identically distributed initial conditions and their perturbations. However, for higher-dimension and more complex systems, the method requires a large number of samples to over-approximate accurately the reachable sets.

Over the years, the problem of computing the set of reachable sets of a given system has received considerable attention. In this section, we attempt to summarise this work and conclude with an argument for the novelty of the current manuscript. There are two main methods for reachability analysis: model-based and data-driven. Model-based reachability analysis uses a mathematical model of the system to compute reachable states from a given set of possible initial states. Over the years, several reachability tools have been developed, such as SpaceEx [9], JuliaReach [10], XSpeed [11], and Flow* [8], to name a few. The reachability methods have been widely used in applications that range from formal system verification to their synthesis [4].

We will focus on data-driven reachability analysis techniques, which have also been proposed for scenarios when a model of the system is unavailable or too complex, and we will use the following axes to compare and position related papers, as summarised in Table 1.

System-under-study (SUS) Denotes the kind of system supported by the technique. Systems can be: linear/affine (*L*, the two are equivalent since one can transform an affine system into a linear one through extension of the states); non-linear (*NL*, the type of Equation (1)); hybrid linear (*HL*, systems with different modes but within each mode the dynamics are linear); and hybrid (*H*, as in the most general hybrid automata). Within the hybrid category, there are kinds of systems, but we abstain from discerning those.

Modularity of SUS (MSUS) Represents the degree of support for decoupled SUS. The categories are: monolithic (*M*) and decoupled (*D*). For example, systems that are represented by communicating sub-models, like the one presented in Figure 1, are decoupled.

Information-required-from-SUS (IRS) Denotes the degree of information that the technique requires from the SUS. Possible categories are: full knowledge (*FK*) of the systems equations; partial knowledge (*PK*), where for example, the Jacobian of the system can be queried through an API, without the knowledge of the equations; and no knowledge (*NK*), where the model can be simulated through an API, without any knowledge of the equations.

Information-required-from-User (IRU) Denotes the kind of information the user needs to specify. At the very least we have information on numerical tolerances (*NT*), and on the opposite side, we have information on dynamic invariants (*DI*).

Guarantees (G) Denotes the level of guarantees offered by the technique. We can have: reachability up to numerical tolerance (*NTG*), probabilistic guarantees (*PG*), and guarantees including numerical approximations, i.e., full guarantees (*FG*).

Dynamic Sensitivity based Reachability Analysis

We begin with the works that are based on solving or estimating the solution to the dynamic sensitivity equations, and then using their solution to build the reachable set, as introduced in Background and Problem Statement. Among these, we highlight the methods described in [12], where a notion of expansion function is introduced, which can be seen as the application of the dynamic sensitivity to a given disturbance in the initial condition (cf. Theorems 3&4 in [12]). The benefit of this method is that more simulations can be run, and in fact, thanks to the dynamic sensitivity information, the initial conditions can be iteratively tried in a way that attempts to drive the system into an unsafe state (to quickly falsify a safety property). In the same way, more samples can be taken, if more accuracy is needed. In the same paper, the technique was extended to hybrid systems without reset actions (but reset actions could be included, provided they are differentiable with respect to their inputs).

The extension requires that the dynamic sensitivity of the jump time be computed as part of the system, and uses results developed earlier in, e.g., [13]. Later, [14] revisits the jump conditions required to apply *second-order* sensitivity analysis to hybrid systems (second-order sensitivity analysis permits an approximation of the flow around a nominal trajectory that will have an error in the order of ϵ^3). The guarantees given are subject to the numerical approximation errors made by the underlying solver library, and on how fined-grained the sampling is, which is controlled by a tolerance parameter provided by the user. This method has been implemented into the Breach tool [15], and we classify it in Table 1 as requiring full knowledge from the system because of its hybrid systems extension. For non-linear systems, only partial knowledge is required.

Another similar approach to sensitivity-based reachability analysis is proposed in C2E2 [16] which originally was designed for continuous and switched systems, and, in the later paper [17] extended to handle hybrid systems as well. Their work proposes a generic "discrepancy function" which provides a time-varying maximum distance bound on any two trajectories originating from the initial set. As far as we could assess, the notions of a discrepancy function and an expansion function are closely related, with both capable of being generated from the dynamic sensitivity equations of the system, or over-approximations of it. The reader can see various methods for computing discrepancy functions for different classes of models in [18], and the DryVR tool [19] expresses the problem of finding a discrepancy function as a problem of learning a linear separator. The tool also provides a probabilistic accuracy guarantee on the computed discrepancy function, given a sampling complexity formula is followed.

HS^{3}V [20,21] is a similar tool, which uses sampling and a Lipschitz-based discrepancy function to estimate reachable sets. Their approach also introduces a method called dynamic simulations-spawning (s-spawning) to bound error growth and adds new simulations to deal with discrete jumps. It is worth mentioning a few other simulation-based approaches [22,23] which provide methods to compute a time-varying function that provides a distance bound on trajectories between the system and a simpler counterpart. The simulations of the simpler model can be combined with the time-varying function to yield reachable sets.

Optimisation-based Reachability Analysis The paper by Xue et al. [24] uses samples obtained from simulating a black-box model to learn an underlying model by solving a robust optimisation problem, which provides probabilistic model accuracy guarantees. Different template models can be used for learning the black-box model (e.g., polynomial functions). A similar approach is presented in work [25] where the author's approach uses sampled noisy data to identify a set of models, which are then over-approximated with zonotopes.

The paper [26] presented a sampling-based reachability analysis approach which is based on random set theory and adversarial sampling. The main novelty of the work is utilising recent advances in deep learning to iteratively discover trajectories which help to converge the actual reachable set. In other learning-based reachability analysis work, the NeuReach tool [27], was introduced which efficiently computes reachable sets and provides a probabilistic accuracy guarantee.

Whilst learning-based methods can improve the performance of the reachability analysis, the main drawback is that the underlying deep learning model has to be retrained for different systems.

Decoupled Reachability Analysis Finally, we highlight the work in [28], which acknowledges the need for reachability analysis techniques that work in parallel for de-coupled models, such as those commonly found in co-simulation scenarios [29]. In the aforementioned paper, the authors introduce an interval-based reachability method which uses set-valued Runge-Kutta integration methods [30]. The reachability computation is done step-by-step, advancing time after the reachable set of each step has been computed. At each step, each sub-model is a black box simulation that computes the interval of outputs based on the interval of inputs. All sub-models' intervals are then exchanged and the step is repeated until a fixed point is reached.

Table 1. Positioning of the state of the art. Notes: [1] – Restricted to two continuous modes; [2] – Submodels have to implement set-based reachability methods.

Paper	SUS	MSUS	IRS	IRU	G
12	H	M	PK	NT	NTG
17	H	M	PK	NT	NTG
16	H	M	FK	DI	NTG
19	H	M	PK	NT	PG
21	H	M	FK	NT	NTG
22	H[1]	M	FK	NT	NTG
24	NL	M	NK	NT	PG
25	NL	M	NK	NT	NTG
26	NL	M	NK	NT	NTG
27	NL	M	NK	NT	PG
28	NL	D	PK[2]	NT	FG
Our work	NL	*D*	*PK*	NT	PG

Novelty of Contribution

As summarised in Table 1, compared to the state of the art, the novelty of our contribution is in providing probabilistic guarantees for decoupled black-box models.

Background and Problem Statement

Continuous Time Systems

We consider continuous-time systems, characterised by a tuple $\Sigma = (X, x_0, f)$, where $X \subset \mathbb{R}^n$ is the state space and n the number of states in the system, $x_0 \in X$ represents the initial state, and $f : X \to X$ represents the vector field and is assumed to be locally Lipschitz continuous (any small changes in x result in bounded changes in $f(x)$). The evolution of the state of Σ satisfies the equation

$$\dot{x}(t) = f(t, x(t)), \quad x(0) = x_0, \tag{1}$$

which, thanks to the local Lipschitz assumption, always has a unique solution, regardless of the initial condition.

In order to represent the solution of Equation (1) as a function of time $t \in \mathbb{R}_{\geq 0}$, and the initial state $x_0 \in X$, we use the notation $\varphi(t, x_0) \in X$. For any finite simulation time $\tau \in \mathbb{R}_{\geq 0}$, and for all $t \in [0, \tau]$, the continuous function $\varphi(t, x_0)$ is a solution to Equation (1), and therefore satisfies

$$\dot{\varphi}(t, x_0) = f(t, \varphi(t, x_0)), \tag{2}$$

with $\varphi(0, x_0) = x_0$. Finally, note that $\varphi(t, x_0)$ is continuous both in t and in x_0.

Reachability Analysis

Reachability analysis is a technique for computing the set of all reachable states of the solution to Equation (1) for each possible initial condition from a set $X_0 \subseteq X$. The reachable set $\mathcal{R}_t$ at time t can be defined formally as:

$$\mathcal{R}_t(X_0) = \{\varphi(t, x_0) \mid x_0 \in X_0\} \tag{3}$$

To capture all reachable states, starting from the initial time, up to a given simulation time τ, then we construct a flowpipe, which is just the union of all reachable states up to τ:

$$\mathcal{R}_{[0,\tau]}(X_0) = \bigcup_{t \in [0,\tau]} \mathcal{R}_t(X_0) \tag{4}$$

Reachability methods provide a powerful approach to verifying safety requirements of dynamical systems under uncertainty[4], and are supported in a range of tools such as SpaceEx[9], Checkmate[31], and Flow*[8]. Furthermore, to efficiently and accurately over-approximate reachable sets, different convex and nonconvex set representations have been developed. We refer the reader to the aforementioned works for more details on how to over-approximate the reachable set in (4).

Co-simulation and the Functional Mock-up Interface Standard

Co-simulation is a technique where multiple black box simulators are coupled together (see[29,32] for introductions to the topic). The difference between a black box simulator and a black box model is that the simulator contains the sub-model and approximates its numerical solution, given an input signal. Since simulators are coupled in feedback loops, the coupled solution is computed iteratively, moving forward in time and approximating the solution at each new timepoint from the solution at previous timesteps. The Functional Mock-up Interface (FMI) standard[33] establishes the interface of the black box simulators, also called Functional Mock-up Units (FMU), in the nomenclature of the standard. An individual FMU is comprised of a description file (in XML), which declares visible state variables and other model information, and binaries that implement the application programming interface to interact with the FMU. Over the years a number of well-known modelling and simulation tools have been upgraded (e.g., Simulink, OpenModelica[34,35]) or developed (INTO-CPS tool[36]) to support FMI standard.

The mandatory interface functions, implemented by an FMU denoted as S, are: `doStep`(S, H) (asks S to advance time to $t + H$ and estimate internal state and outputs at the new time); `setIn`(S, u, v) (set the input of S identified by u to the value v for the current time t); and `getOut`(S, y) (get the value for the output of S identified by y for the current time t).

A co-simulation scenario is a set of FMUs and a description of how they are connected. It is often depicted in a diagrammatic form, as Example 1 shows.

Example 1. *Consider the canonical example of a double mass-spring-damper system, depicted in Figure 1. The system is decoupled into two different FMUs, with inputs and outputs as depicted in the same figure. Then, with an interface similar to the FMI standard, their co-simulation is computed as illustrated in Algorithm 1.*

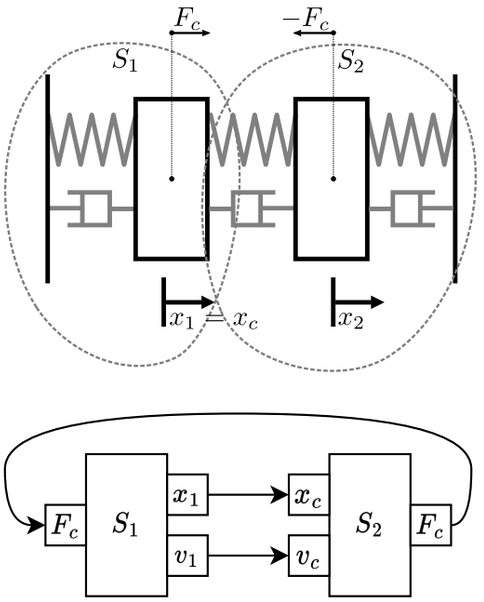

Figure 1. Example double mass-spring-damper system.

Algorithm 1: Example co-simulation orchestration for Example 1.

Inputs: A final simulation time $t_f > 0$, a communication step size $H > 0$, and FMUS S_1 and S_2

$t \leftarrow 0$

Initialise S_1 and S_2

while $t < t - f$ **do**

  `doStep`(S_1, H)

  `doStep`(S_2, H)

  `setIn`(S_1, F_c, `getOut`(S_2, F_c))

  `setIn`(S_2, x_c, `getOut`(S_1, x_1))

  `setIn`(S_2, v_c, `getOut`(S_1, v_1))

  $t \leftarrow t + H$

end

Output: A value for each input/output computed at each time $t \in [0, t_f]$.

In addition to the mandatory functions each FMU implements, the FMI also adds a number of optional functions, that can be optionally implemented by FMU exporting tools. From these, we highlight the functions

that allow one to compute partial derivatives. Neglecting efficiency issues, we denote this function as `getDer`(S, x, y), which returns $\frac{\partial x}{\partial y}$ for the current time and state of s. These will be used later in Section Building Sensitivity Equations Co-Simulation Scenarios to build the dynamic sensitivity equation system of a co-simulation scenario.

Problem Statement

In this paper, we address the problem of computing reachable states of Digital Twin virtual models as formally defined in Problem 1.

Problem 1. *Given a black-box Digital Twin model of a system Σ, initial set $\mathcal{X}_0$, and time-bound T, compute an approximation of the reachable set $\bar{\mathcal{R}}_{[0,T]}(X_0)$ using a finite number of randomly simulated trajectories of Σ. Provide the sample complexity of the computation, i.e., the required number of trajectories for achieving a certain level of approximation with probabilistic confidence.*

In the above problem statement, we assume that a black-box model of the system Σ is available, which can be used to generate sample trajectories from any initial state. These sample trajectories are sufficient for applying our first technique to solve the above problem. Our second technique requires also having access to trajectories of the dynamic sensitivity in the FMUs of the system.

Dynamic Sensitivity Equations

We define the *dynamic sensitivity* equations, also called the *variational equations* or just *sensitivity equations*, of the system in Equation (1) as the different derivatives of the n state variables with respect to the n initial conditions. For example, for a system with 1 dimension, the dynamic sensitivity equations represent how $\varphi(t, x_0)$ changes as a function of changes in the initial condition x_0. We represent this rate of change by the derivative $\frac{\mathrm{d}\varphi(t,x_0)}{\mathrm{d}x_0}$.

For a system with n dimensions, we will represent the state variable in each dimension i by x_i, such that each state $x \in X$ is represented by a vector $x = [x_1, \ldots, x_n]^T$. Furthermore, we will represent the restriction of the solution $\varphi(t, x_0)$ to the state variable x_i as $\varphi_i(t, x_0)$, so that $\varphi(t, x_0) = [\varphi_1(t, x_0), \ldots, \varphi_n(t, x_0)]^T$.

Given state variables x_i and x_j, we will use the shorthand notation $\delta_{i,j}(t, x_0)$ to denote the derivative of $\varphi_i(t, x_0)$ with respect to $x_{j,\text{in}}$ (the initial value for x_j): $\delta_{i,j}(t, x_0) = \frac{\partial \varphi_i(t,x_0)}{\partial x_{j,\text{in}}}$.

The *dynamic sensitivity* is a matrix represented as

$$S(t, x_0) = \begin{bmatrix} \delta_{1,1}(t, x_0) & \ldots & \delta_{1,n}(t, x_0) \\ \vdots & \ddots & \vdots \\ \delta_{n,1}(t, x_0) & \ldots & \delta_{n,n}(t, x_0) \end{bmatrix} \tag{5}$$

The dynamic sensitivity equations shown next represent an extension of Equation (1) with differential equations that relate $S(t, x_0)$ to its time derivative $\dot{S}(t, x_0)$ (derived below):

$$\begin{aligned} \dot{x}(t) &= f(x(t)), & x(0) &= x_0, \\ \dot{S}(t) &= J(x(t)) \cdot S(t), & S(0) &= I, \end{aligned} \tag{6}$$

where we have omitted the dependency to x_0 of each solution to improve readability, $\cdot$ is the matrix product, and

$$J(x(t)) = \begin{bmatrix} \frac{\partial f_1(x(t))}{\partial x_1} & \ldots & \frac{\partial f_1(x(t))}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x(t))}{\partial x_1} & \ldots & \frac{\partial f_n(x(t))}{\partial x_n} \end{bmatrix} \tag{7}$$

represents the *Jacobian matrix* of the continuous-time system and $\frac{\partial f_i(x(t))}{\partial x_j}$ denotes the partial derivative of the i-th state derivative with respect to the j-th state (recall that f is a vector function).

To derive Equation (6), we differentiate $S(t)$ with respect to time. Each entry $\dot{\delta}_{i,j}(t, x_0)$ of $\dot{S}(t)$ is therefore expanded as follows:

$$\begin{aligned} \dot{\delta}_{i,j}(t, x_0) &= \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial}{\partial x_{j,\text{in}}} \varphi_i(t, x_0) \quad \text{(expand notation)} \\ &= \frac{\partial}{\partial x_{j,\text{in}}} \frac{\mathrm{d}}{\mathrm{d}t} \varphi_i(t, x_0) \quad \text{(swap derivative order)} \\ &= \frac{\partial}{\partial x_{j,\text{in}}} f_i(t, \varphi(t, x_0)) \quad \text{(apply Equation (2))} \\ &= \frac{\mathrm{d} f_i(t, \varphi(t, x_0))}{\mathrm{d}x} \cdot \frac{\partial \varphi(t, x_0)}{\partial x_{j,\text{in}}} \quad \text{(apply chain rule)} \\ &= \underbrace{\begin{bmatrix} \frac{\partial f_i(t,\varphi(t,x_0))}{\partial x_1} & \ldots & \frac{\partial f_i(t,\varphi(t,x_0))}{\partial x_n} \end{bmatrix}}_{\text{i-th row of J(x(t))}} \underbrace{\begin{bmatrix} \frac{\partial \varphi_1(t,x_0)}{\partial x_{j,\text{in}}} \\ \vdots \\ \frac{\partial \varphi_n(t,x_0)}{\partial x_{j,\text{in}}} \end{bmatrix}}_{\text{j-th column of } S(\text{x(t)})} \end{aligned}$$

Taking all entries of $\dot{S}(t)$ together yields the equation $\dot{S} = J \cdot S$. Note that each entry depends on the full state solution of the original system $\varphi(t, x_0)$ and therefore the differential equation needs to be solved together with the original equations of the system. A system with n dimensions will therefore be extended to a system with $n + n^2$ dimensions.

Example 2. Taken from[37]. *Consider the system given by the differential equation $\dot{x} = -x + \sin(t) \quad x(0) = x_0$, and its solution given by $\varphi(t, x_0) = x_0 e^{-t} + 0.5(\sin(t) - \cos(t) + e^{-t})$. The solution is plotted for different initial conditions in Figure 2. Since the initial conditions stop making a difference in the system (because of the periodic forcing function), we expect the sensitivity to vanish after about 6 seconds.*

Applying Equation (6), the expanded system gives

$$\begin{aligned} \dot{x} &= -x + \sin(t) \quad x(0) = x_0 \\ \dot{S} &= -S \quad S(0) = 1, \end{aligned} \tag{8}$$

with solution

$$\begin{aligned} \varphi(t, x_0) &= x_0 e^{-t} + 0.5(\sin(t) - \cos(t) + e^{-t}) \\ S(t) &= e^{-t} \end{aligned} \tag{9}$$

plotted in Figure 3.

Example 3. *Consider a spring pendulum whose behaviour is given by the following dynamical system:*

$$\begin{bmatrix} \dot{r} \\ \dot{\theta} \\ \dot{v}_r \\ \dot{v}_\theta \end{bmatrix} = \begin{bmatrix} v_r \\ v_\theta \\ r v_\theta^2 + 9.8 \cos\theta - 2(r-1) \\ -\frac{2 v_r v_\theta + 9.8 \sin\theta}{r} \end{bmatrix} \tag{10}$$

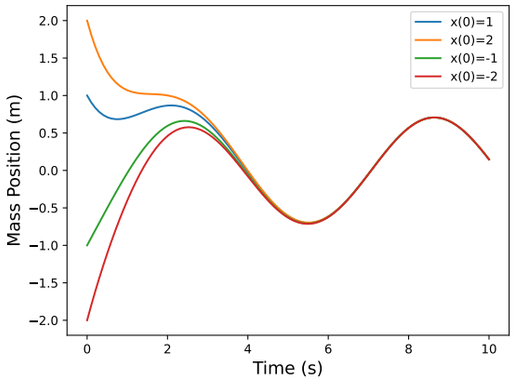

Figure 2. Example solutions for the system in Example 2.

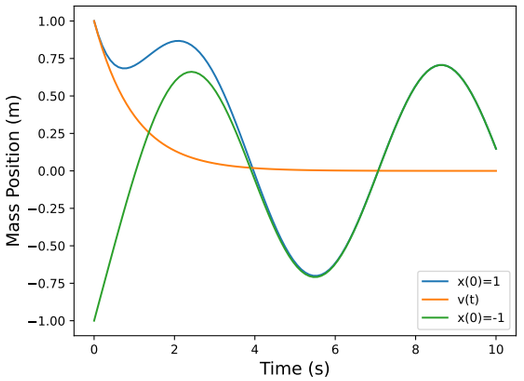

Figure 3. Example solutions for the system in Example 2 including the sensitivity.

The sensitivity matrix is therefore:

$$S(x) = \begin{bmatrix} \delta_{r,r} & \delta_{r,\theta} & \delta_{r,v_r} & \delta_{r,v_\theta} \\ \delta_{\theta,r} & \delta_{\theta,\theta} & \delta_{\theta,v_r} & \delta_{\theta,v_\theta} \\ \delta_{v_r,r} & \delta_{v_r,\theta} & \delta_{v_r,v_r} & \delta_{v_r,v_\theta} \\ \delta_{v_\theta,r} & \delta_{v_\theta,\theta} & \delta_{v_\theta,v_r} & \delta_{v_\theta,v_\theta} \end{bmatrix} \tag{11}$$

As we show next, the Jacobian, $J(x(t))$ in Equation (6), of this system is:

$$J(x) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ v_\theta^2 - 2 & -9.8\sin\theta & 0 & 2rv_\theta \\ \frac{2v_r v_\theta + 9.8\sin\theta}{r^2} & -\frac{9.8}{r}\cos\theta & -\frac{2}{r}v_\theta & -\frac{2}{r}v_r \end{bmatrix} \tag{12}$$

As we have seen before we can get the expression of $\dot{S}$, the time derivative of the dynamic sensitivity matrix, using $\dot{S} = J \cdot S$. We get the following 16 equations, that depend on the original system equations in Equation (10):

$$\begin{aligned}
\dot{\delta}_{r,r} &= \delta_{v_r,r},\ \dot{\delta}_{r,\theta} = \delta_{v_r,\theta},\ \dot{\delta}_{r,v_r} = \delta_{v_r,v_r},\ \dot{\delta}_{r,v_\theta} = \delta_{v_r,v_\theta} \\
\dot{\delta}_{\theta,r} &= \delta_{v_\theta,r},\ \dot{\delta}_{\theta,\theta} = \delta_{v_\theta,\theta},\ \dot{\delta}_{\theta,v_r} = \delta_{v_\theta,v_r},\ \dot{\delta}_{\theta,v_\theta} = \delta_{v_\theta,v_\theta} \\
\dot{\delta}_{v_r,r} &= (v_\theta^2 - 2)\delta_{r,r} - 9.8\sin\theta\delta_{\theta,r} + 2rv_\theta\delta_{v_\theta,r} \\
\dot{\delta}_{v_r,\theta} &= (v_\theta^2 - 2)\delta_{r,\theta} - 9.8\sin\theta\delta_{\theta,\theta} + 2rv_\theta\delta_{v_\theta,\theta} \\
\dot{\delta}_{v_r,v_r} &= (v_\theta^2 - 2)\delta_{r,v_r} - 9.8\sin\theta\delta_{\theta,v_r} + 2rv_\theta\delta_{v_\theta,v_r} \\
\dot{\delta}_{v_r,v_\theta} &= (v_\theta^2 - 2)\delta_{r,v_\theta} - 9.8\sin\theta\delta_{\theta,v_\theta} + 2rv_\theta\delta_{v_\theta,v_\theta} \\
\dot{\delta}_{v_\theta,r} &= C(r,\theta)\delta_{r,r} - K(r,\theta)\delta_{\theta,r} - \frac{2}{r}v_\theta\delta_{v_r,r} - \frac{2}{r}v_r\delta_{v_\theta,r} \\
\dot{\delta}_{v_\theta,\theta} &= C(r,\theta)\delta_{r,\theta} - K(r,\theta)\delta_{\theta,\theta} - \frac{2}{r}v_\theta\delta_{v_r,\theta} - \frac{2}{r}v_r\delta_{v_\theta,\theta} \\
\dot{\delta}_{v_\theta,v_r} &= C(r,\theta)\delta_{r,v_r} - K(r,\theta)\delta_{\theta,v_r} - \frac{2}{r}v_\theta\delta_{v_r,v_r} - \frac{2}{r}v_r\delta_{v_\theta,v_r} \\
\dot{\delta}_{v_\theta,v_\theta} &= C(r,\theta)\delta_{r,v_\theta} - K(r,\theta)\delta_{\theta,v_\theta} - \frac{2}{r}v_\theta\delta_{v_r,v_\theta} - \frac{2}{r}v_r\delta_{v_\theta,v_\theta}
\end{aligned}$$

where

$$C(r,\theta) = \frac{2v_r v_\theta + 9.8\sin\theta}{r^2}, \quad K(r,\theta) = \frac{9.8}{r}\cos\theta$$

Interpretation of Sensitivity Equations

We demonstrate here how dynamic sensitivity equations can be used to approximate the reachable set $\mathcal{R}_{[0,\tau]}(X_0)$ in Equation (4). First note how the distance between the system solutions in Figure 3 for Example 2 is correlated to the sensitivity solution. Since $\varphi(t, x_0)$ is a continuous function of x_0, we can perform a Taylor expansion around the value x_0:

$$\varphi(t, x_0 + \epsilon) \approx \varphi(t, x_0) + \underbrace{\frac{\mathrm{d}\varphi(t, x_0)}{\mathrm{d}x_0}}_{S(t,x_0)}\epsilon + \mathcal{O}\left(\epsilon^2\right) \tag{13}$$

where the $\mathcal{O}\left(\epsilon^2\right)$ denotes the order of the magnitude for the higher order terms in the rest of the Taylor series. Equation (13) gives us a direct method to estimate trajectories around a nominal system solution $\varphi(t, x_0)$. Note that the truncated terms are expected to be in the order of ϵ^2, which will be small in comparison with the first two terms of the Taylor expansion for small values of ϵ.

Example 4. *Following Example 2, we know $S(t) = e^{-t}$, so we can use it to estimate other trajectories around $\varphi(t, 1)$. The result is plotted in Figure 4 where the dotted trajectories represent estimates, and the solid represent the actual solutions. Note that there is no error in the estimates because the system is linear, and therefore the higher order terms in Equation (13) vanish.*

To summarise, for an expanded dynamic sensitivity system as in Equation (6), and a given initial set X_0 of potential initial conditions, the reachable set $\mathcal{R}_t(X_0)$ in Equation (3) can be approximated using the following procedure:

1. Discretise X_0 into smaller hyper-rectangles $\mathcal{X}_1, \ldots, \mathcal{X}_n$ such that the distance between any point contained in each hypercube and its centre is small enough (generally smaller than 1 because of the truncated term in Equation (13));

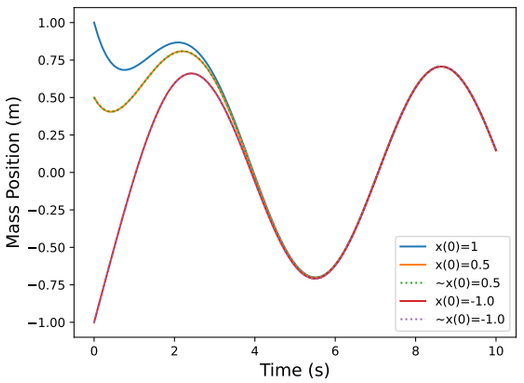

Figure 4. Example estimated solutions for the system in Example 2 around nominal trajectory $\varphi(t,1)$, as detailed in Example 4.

2. For each $\mathcal{X}_j$, compute the nominal solution at its centre, and apply Equation (13) to estimate all trajectories of interest in its vicinity (for linear and affine systems, it suffices to cover all the extremities of $\mathcal{X}_j$);
3. Because of continuity, any set of states between a trajectory and the estimated trajectories in its vicinity are reachable, so we can form flow pipes uniting the nominal trajectory and all trajectories of interest in its vicinity;
4. $\mathcal{R}_t(X_0)$ is then computed by the union of all flow pipes.

The above approach does not necessarily generate over-approximations of the reach set for nonlinear systems since the higher-order terms in the Taylor expansion are eliminated without appropriate quantification of the induced error. In the following sections, we provide two techniques that are based on random trajectories of the system and provide probabilistic correctness guarantees.

Robust Convex Programs

This section provides the mathematical details for robust convex programs (RCPs) and data-driven approximations of their solution. The content of this section is provided in its full generality. We will utilise Theorem 1 and Theorem 2 presented in the sequel to establish the correctness of our data-driven framework. The reader can refer to the papers[38,39] for the full exposition of the results presented in this section.

Let $T \subset \mathbb{R}^q$ be a compact convex set for some $q \in \mathbb{N}$ and $c \in \mathbb{R}^q$ be a constant vector. Let $\mathcal{D}$ be the space of *uncertainty* with $(\mathcal{D}, \mathscr{B}, \mathbb{P})$ denoting the uncertainty probability space ($\mathscr{B}$ is the Borel sigma-algebra on $\mathcal{D}$ and $\mathbb{P}$ a probability measure that assigns probabilities to sets in $\mathscr{B}$). Let $g\colon T \times \mathcal{D} \to \mathbb{R}$ be a measurable function, which is convex in the first argument for each $d \in \mathcal{D}$, and bounded in the second argument for each $\theta \in T$. The robust convex program (RCP) is defined as

$$\text{RCP:} \begin{cases} \min_\theta c^\top \theta \\ s.t.\ \theta \in T \text{ and } \ g(\theta, d) \leq 0 \quad \forall d \in \mathcal{D} \end{cases} \tag{14}$$

An example of the RCP used in our work is presented in Equation (23). Computationally tractable approximations of the optimal solution of the RCP (14) can be obtained using *scenario convex programs* (SCP) that only require gathering finitely many samples from the uncertainty space[39].

Let $(d_i)_{i=1}^N$ be N independent and identically distributed (i.i.d.) samples drawn according to the probability measure $\mathbb{P}$. The SCP corresponding to the RCP (14) strengthened with $\gamma \geq 0$ is defined as

$$SCP_\gamma : \begin{cases} \min_\theta c^\top \theta \\ s.t.\ \theta \in T, \text{ and} \\ g(\theta, d_i) + \gamma \leq 0 \ \forall i \in \{1, 2, \ldots, N\} \end{cases} \tag{15}$$

An example of the SCP used in our work is presented in Equation (24). We denote the optimal solution of RCP (14) as θ^*_{RCP} and the optimal solution of SCP_γ (15) as θ^*_{SCP}. Note that θ^*_{RCP} is a single deterministic quantity but θ^*_{SCP} is a random quantity that depends on the i.i.d. samples $(d_i)_{i=1}^N$ drawn according to $\mathbb{P}$. The RCP (14) is a challenging optimisation problem since the cardinality of $\mathcal{D}$ is infinite and therefore the optimisation has an infinite number of constraints. In contrast, the SCP (15) is a convex optimisation with a finite number of constraints for which efficient optimisation techniques are available. The following two theorems provide sample complexity results for connecting the optimal solutions of the SCP_γ to that of the RCP.

Theorem 1.[38]. *Let $\beta \in (0,1)$ be a confidence value and $\epsilon \in (0,1)$ a given tolerance. Select the number of samples N according to*

$$N \geq \frac{1}{\epsilon}\left(\frac{e}{e-1}\right)\log\left(\frac{1}{\delta} + q\right) \tag{16}$$

where e is Euler number and q is the dimension of the decision vector $\theta \in T$. Then the solution of (15) with $\gamma = 0$ computed by taking N i.i.d. samples $(d_i)_{i=1}^N$ from $\mathbb{P}$ is a feasible solution for the constraint

$$\mathbb{P}(g(\theta, d) \leq 0) \geq 1 - \epsilon \tag{17}$$

with confidence $(1-\beta)$.

The above theorem states that if we take the number of samples appropriately, we can guarantee that the solution satisfies the robust constraint in (14) on all the domain $d \in \mathcal{D}$ except for a small subset that has measure at most ϵ.

Theorem 2.[39]. *Assume that the function $g\colon T \times \mathcal{D} \to \mathbb{R}$ $d \mapsto g(\theta, d)$ in (14) is Lipschitz continuous with respect to $d \in \mathcal{D}$ uniformly in $\theta \in T$ with Lipschitz constant L_d and let $h\colon [0,1] \to \mathbb{R}_{\geq 0}$ be a strictly increasing function such that*

$$\mathbb{P}(\Omega_\varepsilon(d)) \geq h(\varepsilon), \tag{18}$$

for every $d \in \mathcal{D}$ and $\varepsilon \in [0,1]$. Let θ^*_{RCP} be the optimal solution of the RCP (14) and θ^*_{SCP} the optimal solution of SCP_γ (15) with

$$\gamma = L_d h^{-1}(\varepsilon) \tag{19}$$

computed by taking N i.i.d. samples $(d_i)_{i=1}^N$ from $\mathbb{P}$. Then θ^*_{SCP} is a feasible solution for the RCP with confidence

$(1-\beta)$ *if the number of samples is at least* $N(\varepsilon,\beta)$*, where*

$$N(\varepsilon,\beta) := \min\left\{ N \in \mathbb{N} \,\Big|\, \sum_{i=0}^{q-1} \binom{N}{i} \varepsilon^i (1-\varepsilon)^{N-i} \leq \beta \right\}, \tag{20}$$

with q *being the dimension of the decision vector* $\theta \in T$*.*

The above theorem is stronger than Theorem 1 in guaranteeing that the solution will be feasible for the RCP (14) on the whole domain $d \in \mathcal{D}$. This is at the cost of requiring the knowledge of an upper bound on the Lipschitz constant of the function g and also being more conservative in the required number of samples. The confidence $(1-\beta)$ is a common feature of these two theorems and is due to the nature of the solution that depends on the sampled dataset $(d_i)_{i=1}^N$.

Reachability Algorithms

In this section, we describe two different algorithms for computing reachable states of black-box FMI models. The two algorithms compute a scaling factor S which is then used to compute edges of the reachable set, as follows:

$$\varsigma(t, x_c) \pm S(t) \left\| \boldsymbol{\eta}/2 \right\|_\infty$$

where $\varsigma(t, x_c)$ denotes a central trajectory and $\boldsymbol{\eta}$ denotes the size of the discretised initial state-space. This section also describes a curve-fitting approach for estimating an upper boundary of the scaling factor and a method for building up the sensitivity matrix from the FMI's dependency graph.

The first reachability algorithm uses simulated trajectories of a black-box model and Scenario Convex Programs (SCP) to compute a maximum Lipschitz constant of the black-box model. The computed Lipschitz constant together with a central trajectory is then used for computing an interval-based approximation of the reachable set. The alternative algorithm replaces the estimation of the model's Lipschitz constant in the previous algorithm with a solution of sensitivity equations, which describe the impact of perturbations of the system's initial conditions on the trajectories of the system.

These algorithms are presented in detail in the following sections.

Sampling-based Algorithm

For computing the reachable set from a set of initial states $\mathcal{X}_0$, a common approach is to partition the set $\mathcal{X}_0$ into a union of hyper-rectangles $\{\mathcal{X}_j,\ j = 1, 2, \ldots, m\}$ of size $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_n]$ by gridding the state space. Then for each $\mathcal{X}_j$, we find a vector $L_j(t) \in \mathbb{R}^n$ such that:

$$\begin{aligned} |\varsigma(t, x_0) - \varsigma(t, x_0')| \leq L_j(t) \left\| x_0 - x_0' \right\|_\infty \\ \forall x_0, x_0' \in \mathcal{X}_j,\ t \geq 0 \end{aligned} \tag{21}$$

where $\varsigma(t, x_0)$ and $\varsigma(t, x_0')$ are the state trajectories of the system at time t started from $x_0, x_0' \in \mathcal{X}_j$, and $|\cdot|$ denotes the element-wise absolute value. In the next step, the reachable set from each $\mathcal{X}_j$ is computed as the hyper-rectangle $\mathcal{Y}_j$ with edges

$$\varsigma(t, x_{cj}) \pm L_j(t) \left\| \boldsymbol{\eta}/2 \right\|_\infty \tag{22}$$

which gives a hyper-rectangle with centre $\varsigma(t, x_{cj})$ and size $L_j(t) \cdot \boldsymbol{\eta}$. The state x_{cj} is the centre of the initial hyper-rectangle $\mathcal{X}_j$. The union of all $\mathcal{Y}_j$, $j = 1, 2, \ldots, m$ gives an over-approximation of the reachable set from $\mathcal{X}_0$. The implementation of the above procedure requires computing $\varsigma(t, x_{cj})$, which is possible using a black-box model of the system.

Algorithm 2: Sampling-based reach set computation

Inputs: System as a black box, time instance t, initial set $\mathcal{X}_0 \subset \mathbb{R}^n$
Select discretisation $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_n]$ with $\eta_i > 0$
Partition $\mathcal{X}_0$ into hyper-rectangles $\mathcal{X}_j$, $j = 1, 2, \ldots, m$, of size $\boldsymbol{\eta}$ with centre x_{cj}
for $j = 1, 2, \ldots, m$ **do**
  Select N according to (16) or (20)
  Take N samples x_{0i} uniformly from $\mathcal{X}_j$
  Obtain trajectories $\varsigma(t, x_{0i})$ and $\varsigma(t, x_{cj})$ from the black box model
  Solve the SCP_γ in (24) to find $L_j(t)$
  Define $\tilde{\mathcal{Y}}_j$ as a hyper-rectangle with centre $\varsigma(t, x_{cj})$ and size $L_j(t) \left\| \boldsymbol{\eta}/2 \right\|_\infty$
end
Output: Sampling-based reach set $\tilde{\mathcal{Y}} := \cup_j \tilde{\mathcal{Y}}_j$

RCP Formulation and Sampling. The inequality (21) used in the reachability analysis can written as the Robust Convex Program:

$$\text{RCP:} \begin{cases} \min c^\top L_j(t) \\ s.t.\ c = [1; 1; \ldots; 1],\ \ L_j(t) \geq 0,\ \text{ and} \\ |\varsigma(t, x_0) - \varsigma(t, x_{cj})| - L_j(t) \left\| x_0 - x_{cj} \right\|_\infty \leq 0, \\ \forall x_0 \in \mathcal{X}_j. \end{cases} \tag{23}$$

We can define the associated SCP_γ

$$SCP_\gamma : \begin{cases} \min c^\top L_j(t) \\ s.t.\ c = [1; \ldots; 1],\ \ L_j(t) \geq 0,\ \forall i \in \{1, \ldots, N\}, \\ |\varsigma(t, x_{0i}) - \varsigma(t, x_{cj})| - L_j(t) \left\| x_{0i} - x_{cj} \right\|_\infty + \gamma \leq 0, \end{cases} \tag{24}$$

where $x_{0i} \in \mathcal{X}_j$ are taken randomly from a probability distribution $\mathbb{P}$.

Once the SCP_γ (24) is solved, the sampling-based reachable set from $\mathcal{X}_j$ is computed as the hyper-rectangle $\tilde{\mathcal{Y}}_j$ with edges $\varsigma(t, x_{cj}) \pm L_j(t) \left\| \boldsymbol{\eta}/2 \right\|_\infty$ where $L_j(t)$ is obtained by solving (24). The next theorem uses the results of the Section Robust Convex Programs for picking the number of samples N to connect $\tilde{\mathcal{Y}}_j$ with the true reachable set.

Theorem 3. *If* $\tilde{\mathcal{Y}}_j$ *is computed using the solution of* (24) *with* $\gamma = 0$ *and* N *selected according to* (16)*, then with confidence* $(1-\beta)$*, the set* $\tilde{\mathcal{Y}}_j$ *covers the whole true reachable set except for a small set with probability measure at most* ϵ*.*
If $\tilde{\mathcal{Y}}_j$ *is computed using the solution of* (24) *with* N *selected according to* (20)*, then with confidence* $(1-\beta)$*, the set* $\tilde{\mathcal{Y}}_j$ *covers the whole true reachable set.*

The full algorithm for our sampling-based reachability analysis is presented in Algorithm 2.

Lipschitz Constant via Extreme Value Theorem

For estimating L_d in Theorem 2 and making use of it in Theorem 3, we should estimate an upper bound for the fraction

$$\Delta(x,x') := \frac{\|\varsigma(t,x)-\varsigma(t,x')\|}{\|x-x'\|} \tag{25}$$

that holds for all $x, x' \in \mathcal{X}_j$. We follow the line of reasoning in [40,41] and use the Extreme Value Theorem for the estimation.

Let us fix a $\delta > 0$ and assign uniform distribution to the pair (x, x') over the domain $\{x, x' \in \mathcal{X}_j, \|x - x'\| \leq \delta\}$. Then $\Delta(x, x')$ is a random variable with an unknown cumulative distribution function (CDF). Based on the assumption of Lipschitz continuity of the system, the support of the distribution of $\Delta(x, x')$ is bounded from above, and we want to estimate an upper bound for its support. We take $\mathfrak{n}$ samples from (x, x') and compute $\mathfrak{n}$ samples $\Delta_1, \Delta_2, \ldots, \Delta_{\mathfrak{n}}$ for $\Delta(x, x')$. The CDF of $\max\{\Delta_1, \Delta_2, \ldots, \Delta_{\mathfrak{n}}\}$ is called the limit distribution of $\Delta(x, x')$. The Fisher-Tippett-Gnedenko theorem says that if the limit distribution exists, it can only belong to one of the three families of extreme value distributions - the Gumbel class, the Fréchet class and the Reverse Weibull class. These CDFs have the following forms:

$$\text{Gumbel (Type I):}\ G(s) = \exp\left(-\exp\left(\frac{s-a}{b}\right)\right)$$
$$\text{where } s \in \mathbb{R}$$

$$\text{Fréchet (Type II):}\ G(s) = \begin{cases} 0 & \text{if } s < a \\ \exp\left(-(\frac{s-a}{b})^{-c}\right) & \text{if } s \leq a \end{cases}$$

$$\text{Rvr. Weibull (Type III):}\ G(s) = \begin{cases} \exp\left(-(\frac{a-s}{b})^{c}\right) & \text{if } s < a \\ 1 & \text{if } s \leq a \end{cases}$$

where $a \in \mathbb{R}, b > 0, c > 0$ are respectively the location, scale and shape parameters.

Among the above three distributions, only the Reverse Weibull class has support bounded from above. Therefore, the limit distribution of $\Delta(x, x')$ will be from this class and the location parameter a is such an upper bound. As a result, we can estimate the location parameter of the limit distribution of $\Delta(x, x')$ to get an estimation of the Lipschitz constant.

A procedure for estimating the Lipschitz constant is presented in Algorithm 3. This uses obtained Lipschitz constants to compute approximate reachable sets. For each state of the system, a single Lipschitz constant value is obtained from a previously sampled set. In this work, we considered two operations for obtaining a final $L_s(x, t)$: a maximum value and a value produced via curve-fitting and the Extreme Value Theorem [42]. The algorithm then computes a central trajectory of the model by simulating it from the set of initial values which are midway between the lower and upper limits of the initial set.

Remark *The estimated Lipschitz constant from Algorithm 3 can also be used directly for estimating the reachable sets. Unfortunately, this quantity is just an estimation and will converge to the true Lipschitz constant in the limit. When it is computed with a finite number of samples, it is not associated with a quantitative closeness guarantee. In contrast, using the vector $L_j(t)$ for reachability computations is more likely to give less conservative reach sets with formal probabilistic closeness guarantees.*

Algorithm 3: Lipschitz constant estimation using Reverse Weibull distribution

Inputs: System as a black box, time instance t, initial set $\mathcal{X}_j \subset \mathbb{R}^n$
Parameters: $\delta > 0$, number of samples $\mathfrak{n}, \mathfrak{m}$
for $k = 1, 2, \ldots, \mathfrak{m}$ **do**
  Take $\mathfrak{n}$ samples (x_i, x_i') uniformly from the set $\{x, x' \in \mathcal{X}_j, \|x - x'\| \leq \delta\}$
  Compute $\{\Delta(x_i, x_i'), i = 1, 2, \ldots, \mathfrak{n}\}$ using (25) and trajectories from the black box model
  Define $\mathcal{L}_k = \max_i \Delta(x_i, x_i')$
end
Fit a Reverse Weibull distribution to the dataset $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{\mathfrak{m}}\}$
Get the location, scale and shape parameters of the fitted distribution
Output: Estimated Lipschitz constant as the location parameter of the fitted distribution

Sensitivity-based Algorithm

In this section, we describe an alternative algorithm, which uses solutions of dynamic sensitivity equations to replace scaling of the initial region with a Lipschitz constant factor $L_j(t)$, with rescaling based on the sensitivity matrix $S(x_{\text{in}}, t)$.

The algorithm similarly partitions initial region $\mathcal{X}_0$ into a union of hyper-rectangles $\{\mathcal{X}_j,\ j = 1, 2, \ldots, m\}$ of size $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_n]$. The algorithm then requires obtaining a system of sensitivity equations $\dot{S(t)}$ and solving them numerically together with black-box system $\dot{x}(t)$ from an N number of randomly sampled initial conditions x_{0i} within each hyper-rectangle $\mathcal{X}_j$.

The reachability algorithm then over-approximates the image $\mathcal{Y}_j$ of the hyper-rectangle $\mathcal{X}_j$, by first computing *expansion vectors*

$$\boldsymbol{\xi}^i = [\xi_1^i, \ldots, \xi_n^i] \quad \text{where } \xi_k^i = |S(t, x_{0i})| \cdot (\boldsymbol{\eta}/2)^{\mathrm{T}}$$

which use the sensitivity matrix $S(t, x_{0i})$ (or rather, its element-wise absolute value $|S(t, x_{0i})|$) to compute the maximum expansion in each direction of the sample point x_{0i}. The method then takes the element-wise maximum $\boldsymbol{\xi}^{\max} = \left[\max_{i=1}^N \xi_1^i, \ldots, \max_{i=1}^N \xi_n^i\right]$ which is used to compute the edges of $\mathcal{Y}_j$ by expanding around the central trajectory $\varsigma(t, x_{cj})$.

The full algorithm is described in Algorithm 4.

Building Sensitivity Equations Co-Simulation Scenarios

In this section, we describe how to extend Algorithm 4 to handle networks of FMUs implementing the FMI interface.

Given the network structure of FMUs, in order to get the Jacobian matrix required to compute the sensitivity

Algorithm 4: Sensitivity-based reach set computation

Inputs: Time instance t, initial set $\mathcal{X}_0 \subset \mathbb{R}^n$
Select discretisation $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots, \eta_n]$ with $\eta_i > 0$
Partition $\mathcal{X}_0$ into hyper-rectangles $\mathcal{X}_j$, $j = 1, 2, \ldots, m$, of size $\boldsymbol{\eta}$ with centre x_{cj}
Acquire system of dynamic sensitivity equations $\dot{S}(t)$
for $j = 1, 2, \ldots, m$ **do**
  Select N according to (16) or (20)
  Take N samples x_{0i} uniformly from $\mathcal{X}_j$
  Obtain central trajectory $\varsigma(t, x_{cj})$ and sensitivity matrix $S(t, x_{0i})$ from the black box model
  Compute *expansion vectors* $\boldsymbol{\xi}^i = [\xi_1^i, \ldots, \xi_n^i]$ where $\xi_k^i = |S(t, x_{0i})| \cdot (\boldsymbol{\eta}/2)^{\mathrm{T}}$
  Compute *maximum expansion vector* $\boldsymbol{\xi}^{\max} = \left[\max_{i=1}^N \xi_1^i, \ldots, \max_{i=1}^N \xi_n^i\right]$
  Define $\tilde{\mathcal{Y}}_j$ as a hyper-rectangle with centre $\varsigma(t, x_{cj})$ and size $\boldsymbol{\xi}^{\max}$
end
Output: Sampling-based reach set $\tilde{\mathcal{Y}} := \cup_j \tilde{\mathcal{Y}}_j$

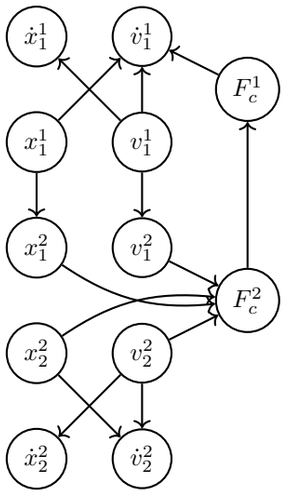

Figure 5. The dependency graph example of the mass spring damper example.

matrix, we need a way to differentiate a variable in one FMU with respect to a variable in another FMU (recall (6)). For that reason, we build a dependency graph before the sampling starts. The vertex of this graph are the state variables of each FMU, their time derivatives, and the input and output variables. The edges represent the dependency of the target on the source. For example, given the system and implementation in Figure 1, its dependency graph is depicted in Figure 5.

Remark *If a dependency graph has a cycle, a variable depends on itself. This is not a typical behaviour of systems in the form of* (1) *and is therefore outside the scope of this paper.*

We can use the dependency graph to know what computations we need to do in order to calculate a derivative, as follows. Given variables α and β, let $\mathcal{D}$ denote all cycle-free paths from α to β. The derivative of α with respect to β is:

$$\frac{\mathrm{d}\alpha}{\mathrm{d}\beta} = \sum_{p \in \mathcal{D}} \prod_{i=0}^{|p|-2} \frac{\partial p[i+1]}{\partial p[i]} \tag{26}$$

where, given path p, $|p|$ denotes its length and $p[n]$ denotes the n-th element of p.

For example, in Figure 5, $\frac{d\dot{v}_1^1}{dx_1^1}$ is given as follows. There are two paths: $x_1^1 \to \dot{v}_1^1$ and $x_1^1 \to x_1^2 \to F_c^2 \to F_c^1 \to \dot{v}_1^1$. Hence

$$\begin{aligned} \frac{\mathrm{d}\dot{v}_1^1}{\mathrm{d}x_1^1} = \frac{\partial \dot{v}_1^1}{\partial x_1^1} & \quad (1^{\text{st}} \text{ path}) \\ + \frac{\partial x_1^2}{\partial x_1^1}\frac{\partial F_c^2}{\partial x_1^2}\frac{\partial F_c^1}{\partial F_c^2}\frac{\partial \dot{v}_1^1}{\partial F_c^1} & \quad (2^{\text{nd}} \text{ path}) \end{aligned} \tag{27}$$

In order to compute the sensitivity matrix, we initialise it to an identity matrix of the correct dimension. After that, each sample step is a co-simulation run, where we compute the Jacobian at every co-simulation step, calling a function that computes every partial derivative that makes an element of the Jacobian matrix $J(x(t))$ using (26). Once we have the Jacobian for time t we estimate the dynamic sensitivity matrix using a numerical solver. For simplicity, we use the Forward Euler method: $S(t+H) = S(t) + \dot{S}(t) * H = S(t) + J(x(t)) \cdot S(t) * H$, where $\dot{S}(t)$ is computed as in (6) and H is the co-simulation step-size parameter. We provide a formalised summary of the algorithms in Algorithm 5.

Algorithm 5: Compute the sensitivity matrix of a system in the FMI standard

Input: A set of FMUs FS and their inter-connections, the communication step size H, the final simulation time $t_f > 0$.
Initialise S and J to the identity matrix
$t \leftarrow 0$
while $t < t_f$ **do**
  Exchange data among all FMUs
  Compute J using Equation (26)
  forall $F \in FS$ **do**
    `doStep`(F, H)
  end
  $S \leftarrow S + J \cdot S * H$
  $t \leftarrow t + H$
end
Output: The dynamic sensitivity matrix S after an arbitrary number of steps.

Validation Experiments

This section presents validation exercises which evaluate our reachability algorithms as presented in the previous section. The validation exercises cover both affine dynamical systems and non-linear systems and aim to evaluate the conservativeness of the computed reachable sets and the associated computation time. We also obtain reachable sets (and computation time) produced by the model-based reachability tool Flow* and compare them against ones produced by our methods. To select non-linear system

benchmarks and Flow* parameters, we followed a well-known verification competition ARCH[43].

Experiment setup. All timing results in this section were measured on an HP EliteBook 840 G7 with an Intel Core i5-10310U processor under Ubuntu 22.04 (Linux 5.14.0). For the methods described in this paper, the results are based on a prototype implementation in Python. In particular, we relied on the SciPy[44] `solve_ivp` function and the LSODA solver[45] for solving dynamical systems (with an absolute tolerance parameter of $\texttt{atol} = 10^{-6}$ and a relative tolerance parameter $\texttt{rtol} = 10^{-3}$), whilst SCP optimisation problems were solved via the CVXPY library[46,47] with the parameter $\gamma = 0$. Comparison results and timings for Flow* were produced by Flow* toolbox*.

Affine Systems

We can start to evaluate the performance of our method on Linear/Affine Initial Value Problems of form

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = A\mathbf{x}(t) + \mathbf{b}; \quad \mathbf{x}(0) \in \mathbf{x}_0 \tag{28}$$

with state matrix $A \in \mathbb{R}^{n\times n}$ and offset vector $\mathbf{b} \in \mathbb{R}^n$, and interval vector initial region $\mathbf{x}_0 \in \mathbb{IR}^n$. Whilst linear systems pose a significantly easier reachability challenge than general non-linear systems — in this case, sensitivity analysis is exact, whilst Flow* and SpaceEx both provide very efficient special-purpose reachability algorithms — they allow us to effectively evaluate how well the methods of this paper approximate a given linear system's dynamics, since these are well understood and admit explicit solutions.

Sample reachability results for different classes of linear systems are shown in Figure 6. We can see that Flow* and sensitivity-based reachability analysis both produce indistinguishable flowpipes, whilst applying reachability analysis based on the Lipschitz constant computed from sampled trajectories alone gives a coarser reachable set estimation shown.

Lipschitz constant estimation accuracy To assess the overall accuracy of our methods, we will consider uniformly randomly selected N-dimensional Affine Systems of the form Equation (28), restricted such that $A \in [-1,1]^N$, $\mathbf{b} \in [-1,1]$, and $\mathbf{x}_0 \subseteq [-1,1]^N$. We will consider separately the classes of stable systems (those for which every eigenvalue of A has a negative real part) and unstable systems (those for at least one eigenvalue of A has a positive real part), and take 100 systems of each class.

We will assess how accurately each of the different methods captures the dynamics of the underlying system based on the vector of Lipschitz constants which they use to compute reachable sets. Whilst the SCP optimisation directly computes a vector $L_{\mathrm{SCP}}(t)$ of Lipschitz constants for the system, we are also able to compute a similar vector of Lipschitz constants from the sensitivity matrix as $L_{\mathrm{sens}}(t) \triangleq \mathbf{c}\,|S(t)|$ where $\mathbf{c} = [1,\ldots,1]$ and $|M|$ is the element-wise absolute value of the matrix M. and considering the accuracy of each method to estimate the Lipschitz constant of the system over-approximate the system dynamics. We will compare each of these approximations to the true vector of Lipschitz constants (with respect to $\|\cdot\|_\infty$) for the system which we can compute using the general solution of a linear ODE as, $L(t) = \mathbf{c}\,|\exp(At)|$. Then we may measure the relative absolute error of an approximated Lipschitz constant vector $L'(t)$ at a given time-point t as

$$\mathrm{RAE} \triangleq \frac{\|L'(t) - L(t)\|_2}{\|L(t)\|_2}.$$

Then, we may estimate the overall performance by taking the Geometric Mean Relative Absolute Error (GMRAE)† of multiple sampled relative absolute errors RAE_i via the formula

$$\mathrm{GMRAE} \triangleq \left(\prod_{i=1}^{n} \mathrm{RAE}_i\right)^{\frac{1}{n}}.$$

In the special case of 2D systems, Figure 7 shows the evolution of the GMRAE of the Lipschitz constant vector estimate produced using dynamical sensitivity analysis, and SCP optimisation for varying numbers of samples. We see that the relative error from SCP optimisation decreases with an increasing number of samples, and is roughly consistent over the whole simulation time. Additionally, the *relative* error of the method is similar between stable and unstable systems; this result is somewhat surprising given that typical Lipschitz constants for random unstable systems can be orders of magnitude larger than those of stable systems (and, indeed, the *absolute* error of the method will be correspondingly larger for the same number of samples). Figure 8 shows the trade-off between the total runtime of each method and the relative error achieved. We observed a relationship between the number of samples and the relative error improvement in the relative error trailing off after 80 samples. Finally, we observed that, as expected, dynamical sensitivity analysis (with a single sampled sensitivity matrix) approximates the true Lipschitz constant vector almost perfectly for linear systems, and provides by far the best accuracy/runtime trade-off for 2D systems.

Additionally, Figure 9 shows how the runtime and relative error of each method varies with the dimension of the system, based on 100 randomly sampled stable and unstable system for dimensions 1 through 6. We can see that the runtime of each method increases exponentially with the system dimension and that the rate of increase of sampling runtime increases with the number of samples, whilst the runtime of dynamic sensitivity analysis increases significantly more rapidly than the SCP optimisation-based approximation with any of these numbers of samples. However, dynamic sensitivity consistently produced the best approximation of the system Lipschitz constant vector, and indeed, its relative error decreased with the dimension of the system. This suggests that the dynamic sensitivity equations are a reliable method of estimating the Lipschitz constant of linear systems, with consistent accuracy regardless of system dimension, whilst sampling offers a flexible cost/accuracy trade-off for higher-dimensional systems.

*https://github.com/chenxin415/flowstar/tree/master/flowstar-toolbox

†The geometric mean is preferred over the mean when aggregating error rates due to the latter's sensitivity to outliers[48], such as those arising from numerical errors when computing the RAE of small quantities.

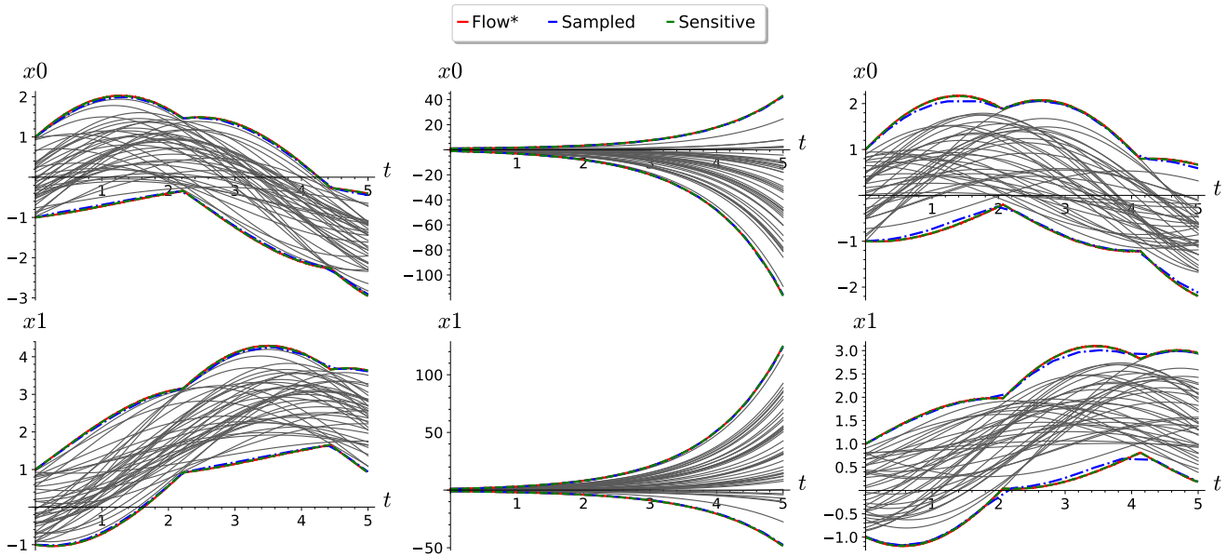

Figure 6. Comparison of reachability from sampled Lipschitz constants with Flow* and Sensitivity Analysis results for a randomly generated 2D stable system (left), an unstable system (middle), and an oscillator (right) from the unit initial region $[-1, 1]^2$. Numerical simulations (grey) for 100 randomly sampled initial conditions are shown for comparison.

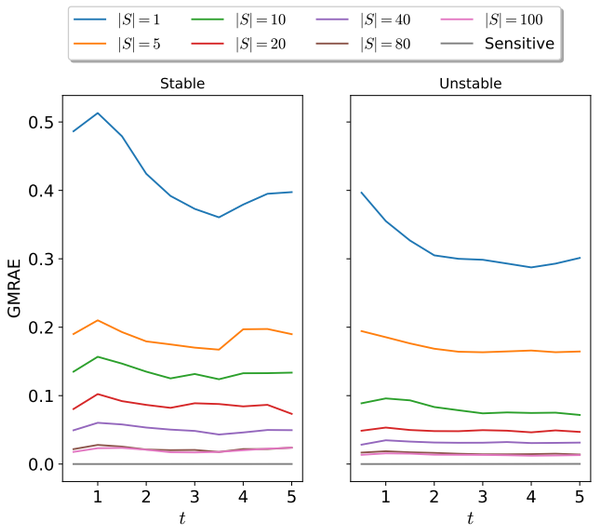

Figure 7. Errors of different methods of Lipschitz estimation at different time points between 1.0 and 5.0 for stable and unstable random 2D linear systems

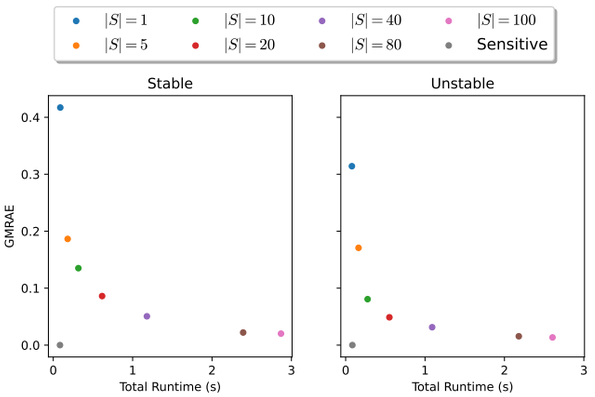

Figure 8. Comparison of total runtime against GMRAE for stable and unstable random 2D linear systems

Nonlinear Systems

This section compares our proposed algorithms for computing reachable sets and validates them against a model-based reachability analysis tool - Flow*. Let us start by considering 2D nonlinear Van Der Pol system:

$$\begin{cases} \dot{x}(t) = y(t) \\ \dot{y}(t) = (1 - x(t)^2) \cdot y - x \end{cases} \tag{29}$$

Figure 10 compares the reachable set for the initial set $[1.1, 2.4] \times [2.35, 3.45]$ computed using by Algorithms (2) - (4) and Flow*. The top figures show reachable sets produced by sensitivity-based Algorithm (4) (blue curve), Flow* (red curve) and some randomly sampled trajectories (grey curves) for x, y states of the Van Der Pol system respectively. Flow* was not able to produce reachable sets over the whole time horizon $[0, 5]$ with the given initial region.

The rest of the section considers four additional non-linear models with varying number of dimensions: coupled Van Der Pol (4D), Rossler System (3D), Spring Pendulum (4D, model from the Dynamic Sensitivity Equations section) and Biological Model (7D). We evaluate the runtime and flowpipe volume accuracy produced by Algorithms (2) and (4). The latter is measured by using Equation (30):

$$\mathcal{A} = \sum_{t=0}^{T} \left(100 - \left(\frac{\mathcal{V}ol(\mathcal{R}_S(t)) - \mathcal{V}ol(\mathcal{R}_F(t))}{\mathcal{V}ol(\mathcal{R}_F(t))} \times 100 \right) \right) \tag{30}$$

where $\mathcal{V}ol(\mathcal{R}_S(t))$ and $\mathcal{V}ol(\mathcal{R}_F(t))$ are volumes of reachable sets produced respectively by one of our algorithms and Flow* at time t with δ size step. The metric measures an accumulated proportional volumetric difference between two

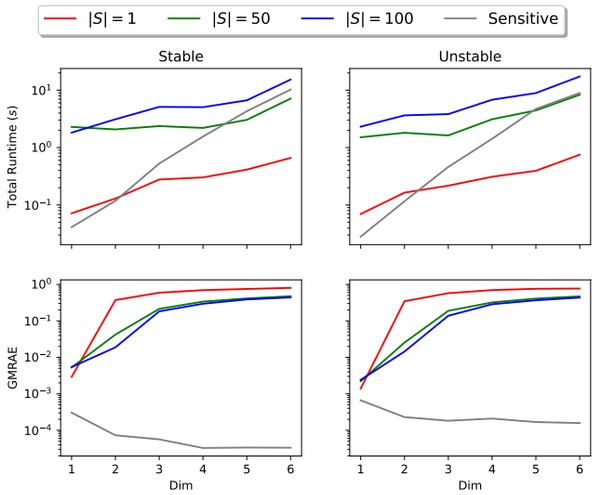

Figure 9. Errors and runtimes of different methods of Lipschitz estimation at time point $t = 5.0$ for randomly sampled linear systems of up to 6 dimensions.

flowpipes (e.g., negative $\mathcal{A}$ would indicate that in comparison to Flow* one of our algorithms produces a less conservative flowpipe). From Figure 10 we can observe that the sampling-based algorithm computes a more conservative flowpipes, however, this comes at a cost of requiring more samples, hence computation time, to guarantee an over-approximation, especially for larger initial regions.

Similar findings can be observed from Figure 12 in which we summarise our accuracy results from three models for different number of samples: Van Der Pol initial state:$x_1 = [1.1,\ 1.4],\ y_1 = [2.35,\ 2.45]$, coupled Van Der Pol parameters $x_{1,2} = [1.25,\ 1.55], y_{1,2} = [2.35,\ 2.45], T = [0,\ 5]$, while Rossler system $x = [0.7,\ 1]$ and $y, z = 1$; all systems analysed for $[0,\ 5]$ seconds. We decided to exclude results from Spring Pendulum and Biological models as Flow* was only able to produce reachable sets from small initial sets and for short time horizons, resulting in minuscule flowpipe volumes.

The runtime validation experiments are summarised in Figure 12. In these experiments, we again increased the number of samples for Algorithms 2 and 4 and observed reachable set computation time. We also include the runtime performance of the Flow* tool. Important to note that at this stage, we did not attempt to improve the computational performance of the proposed methods.

Figure 12 clearly shows that Algorithm 2 is considerably slower in comparison to Algorithm 4 and does not scale well with an increased number of samples. The main reason for this is the computation overhead of solving $SCPs$. We can see this in Figure 13 in which we demonstrate the proportion of runtime it takes to sample and solve the SCP in Algorithm 2 and solve sensitivity equations in Algorithm 4 for different models and numbers of samples. Except for the case of the Biological model, solving sensitivity equations in Algorithm 4 makes up a significantly smaller proportion of computation time, while the opposite is true in the case of obtaining maximum Lipschitz constant with $SCPs$ in Algorithm 2.

In short, the results presented in this section have shown that our algorithms produce reasonably conservative reachable sets for non-linear systems. Although, with the current algorithm implementation their runtimes do not scale well with the increased number of samples, we have shown accurate results can be produced even with a fairly small number of samples. The main limitation of the Algorithm (2) is the need for a larger number of samples to provide probabilistic accuracy guarantees, while solving SCP is a major contributor to a large runtime. The sensitivity-based algorithm provides much less conservative results but offers a more scalable approach.

Sensitivity Matrix Cosimulation

In order to validate the results of the algorithms given in Building Sensitivity Equations Co-Simulation Scenarios we are going to use the mass spring damper system visualised in Figure 1. The equations that describe this system's behaviour are provided in Equation (31).

$$\begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \\ \dot{x}_2 \\ \dot{v}_2 \\ \dot{F}_c \end{bmatrix} = \begin{bmatrix} v_1 \\ \frac{-c_1 \cdot x_1 - d_1 \cdot v_1 + F_c}{m_1} \\ v_2 \\ \frac{-c_2 \cdot x_2 - F_2}{m_2} \\ c_c \cdot (x_2 - x_1) + d_c \cdot (v_2 - v_1) \end{bmatrix} \tag{31}$$

We are going to solve this system together with the coupled sensitivity equations using the SciPy `solve_ivp` solver. In Figure 14 we validate the value of δ_{x_1,x_1} (an element of the sensitivity matrix computed with Algorithm 5) against the analytical solution, with a time step of 0.01. We will then compute the error between the sensitivity matrix computed by Algorithm 5 and the `solve_ivp` solver function as:

$$e(t) = \|S(t) - S'(t)\|_2 \tag{32}$$

where $S(t)$ denotes the sensitivity matrix computed by the Algorithm 5, $S'(t)$ denotes the sensitivity matrix computed by the `solve_ivp` function and $\|\cdot\|_2$ denotes the 2-norm for matrices. In Figure 15 we show different error functions for different step sizes, which shows that the smaller the step size, the smaller the error.

We can see in the results that our approximation is close enough to the `solve_ivp` function. Figure 14 shows that both functions are almost indistinguishable. Furthermore, Figure 15 shows that by decreasing the step size of the co-simulation scenario we can reduce the error, which allows us to get as close as we want to standard numerical algorithms.

Validation Discussion

In this section, we explored the ability of each of our methods to accurately and efficiently approximate the dynamics of black-box models and to conservatively compute reachable sets.

Firstly, we saw that in the case of linear systems, the sampling-based approach is able to approximate the sensitivity of the system to its initial conditions (as captured in the vector of Lipschitz constants), and the accuracy of this approximation can be increased by increasing the number of samples. This is consistent with, Theorem 3

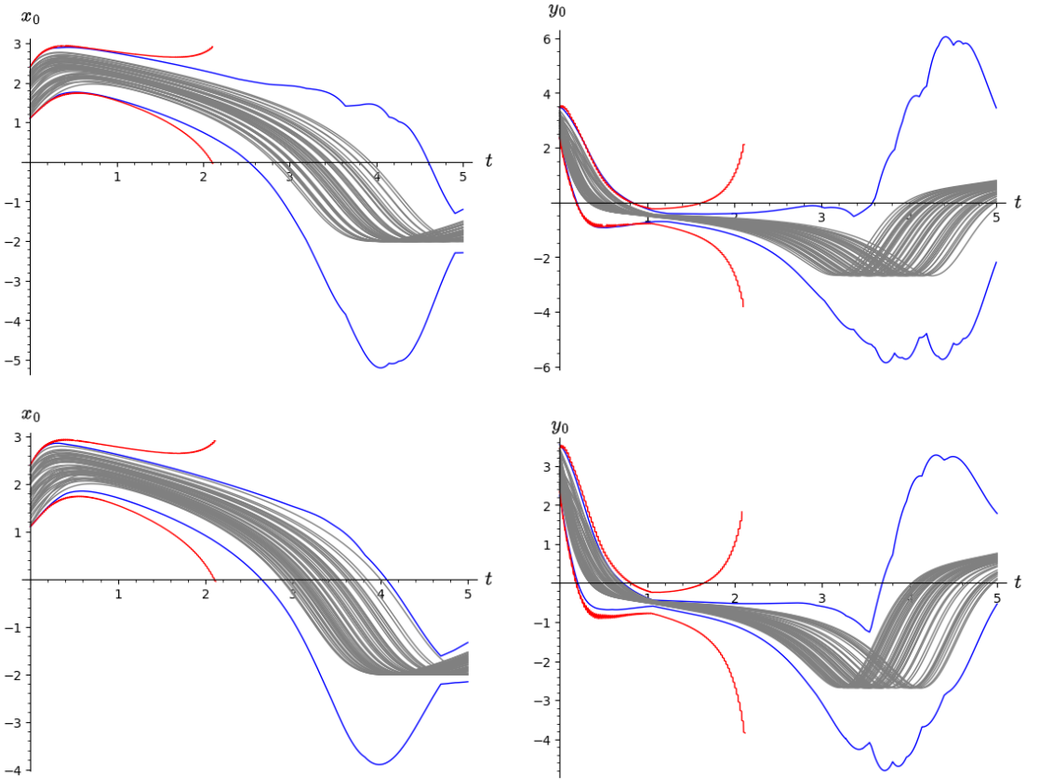

Figure 10. Reachable set comparison of the nonlinear Van Der Pol system for the initial set $[1.1, 2.4] \times [2.35, 3.45]$ for $T = [0, 5]$. **Top:** reachable set produced by a sensitivity-based algorithm for x state (left) and y state (right), **Bottom:** reachable sets produced by a sampling-based algorithm for x state (left) and y state (right). Both algorithms were used with 100 samples.

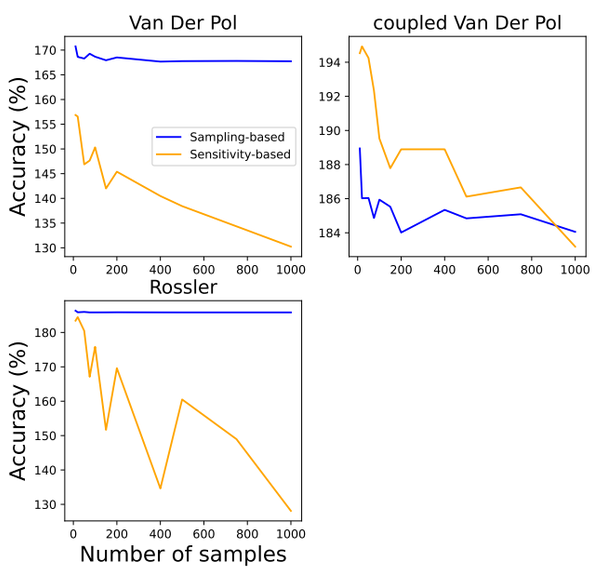

Figure 11. Volume error exercise that demonstrates the number of samples effects on volume accuracy. We consider the following number of samples $[10, 20, 50, 75, 100, 150, 200, 400, 500, 750, 1000]$.

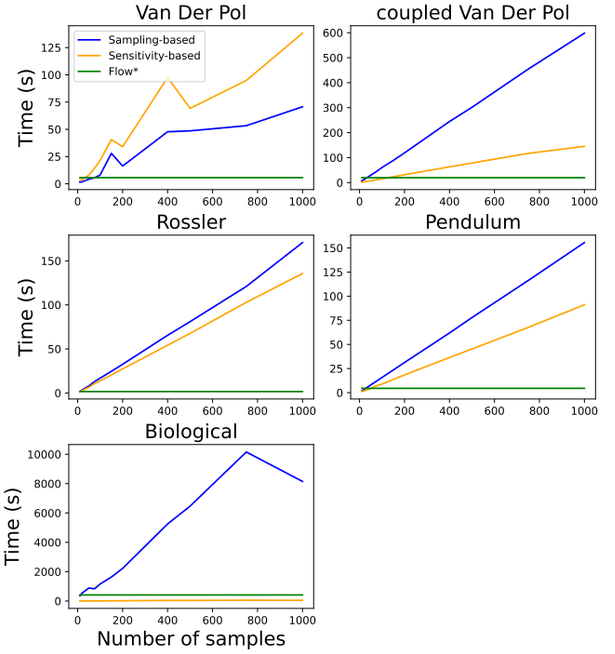

Figure 12. Runtime validation exercise that demonstrates the number of samples effects to computation time of reachable sets for different non-linear models.

which specifies the number of samples required to achieve a given probability of over-approximating the true Lipschitz constants and, consequently, the true reachable set. We also saw that for linear systems, dynamic sensitivity analysis gives an almost exact approximation of the true Lipschitz

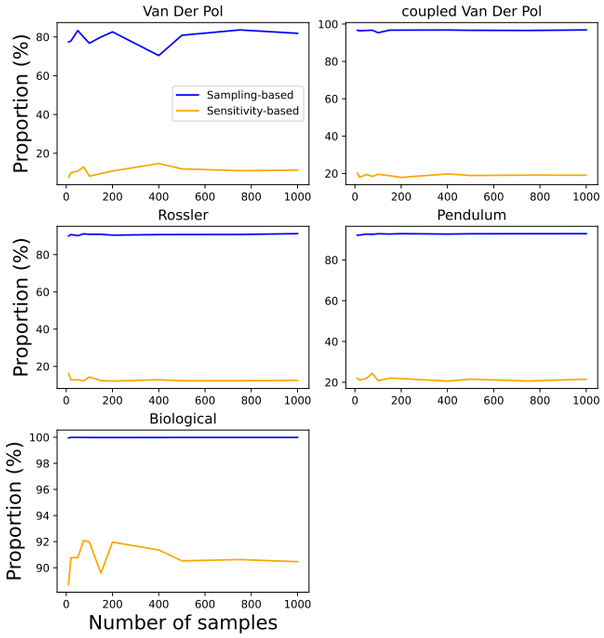

Figure 13. The proportion of runtime in Fig.12 to perform sampling and solving SCP in Algorithm (2), and sample and solve sensitivity equations in Algorithm (4). Note: the rest of the runtime is used for computing flowpipe.

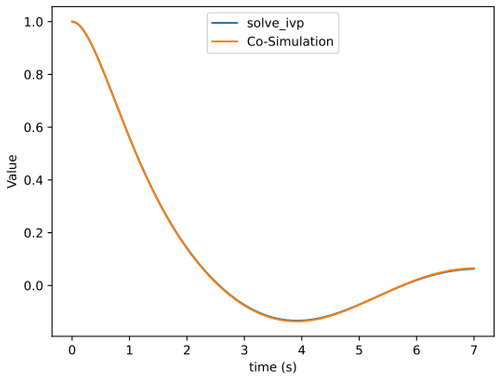

Figure 14. Comparison of δ_{x_1,x_1} computed by the Algorithm (5) and the solve_ivp ODE solver.

constants regardless of system dimension, although its runtime increases rapidly with the dimension of the system.

For non-linear systems, both sampling and dynamic sensitivity analysis give approximate results, whilst their conservativeness can both be increased by increasing the number of samples used. For most of our systems, we saw that dynamic sensitivity analysis gives reasonable results for a reasonably low runtime. However, the sampling-based approach is able to give more conservative results for higher numbers of samples and is also able to give probabilistic guarantees on containment.

We also saw how we can sensitivity analysis to decoupled FMUs, by dynamically tracking the sensitivity matrix of the system. This is limited by the fact that our current co-simulation technique relies on the Forward Euler

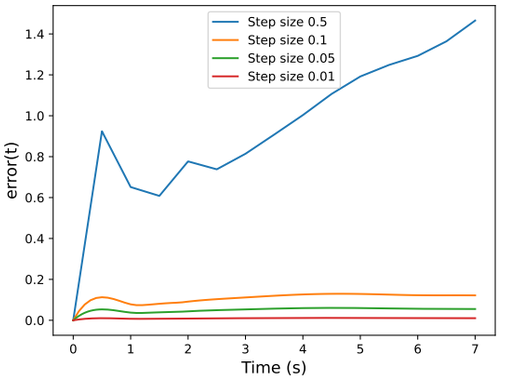

Figure 15. Errors between the sensitivity matrix computed by Algorithm (5) and solve_ivp method with varying time steps sizes.

method, which produces larger errors than more competitive numerical integration methods. The use of better numerical methods is important to reduce these errors, but this would impose additional requirements on the FMUs being simulated. In practice, we observed relatively small errors between the sensitivity matrices computed via this method and the conventional open-box method using the LSODA solver.

Conclusions and Future Work

Ensuring the dependability of Digital Twins relies on proving that the formal system models underpinning them are safe. In some cases, accurate models of complex systems are too difficult to obtain or unavailable due to IP protection (as facilitated by the FMI standard). In this work, we develop methods to provide formal analysis for models featuring uncertainty or unavailability of their dynamics, by introducing algorithms for performing reachability analysis of black-box models. We were particularly focused on the FMI standard-based black box dynamical system models. The developed data-driven and dynamic-sensitivity–based reachable set computation methods have been thoroughly evaluated for linear and non-linear dynamical systems, and results have shown that conservative reachable sets can be computed. Although, as discussed, for large numbers of samples and high-dimensional systems, the runtime performance of the algorithms offers scope for improvement (particularly the sampling-based algorithm), we saw that algorithms do not require a large number of samples to produce accurate reachable sets.

There are several interesting directions for future work:

1. We could investigate extending each of the methods proposed in this paper from reachability analysis, to monitoring Signal Temporal Logic properties of the system's behaviour following the methodology of[49]. This would allow us to verify whether black box models satisfy high-level temporal logic specifications, whilst accounting for the impact of uncertainty on the result of verification via three-valued logic and probabilistic guarantees.

2. We could investigate the application of each of the methods to parametric black-box models, as a way to soundly account for the impact of uncertain model parameters on the behaviour of the system.

3. Our sampling-based approach can in general be applied to hybrid models as long as trajectories are continuous functions of the initial state. To apply the dynamic sensitivity-based approach to hybrid automata, we would like to investigate how dynamic sensitivity equations could be obtained for a black-box hybrid system.

Acknowledgements

Thomas Wright gratefully acknowledges the support of the UK EPSRC for grant EP/V026801/2, UKRI Trustworthy Autonomous Systems Node in Verifiability. The work of Sadegh Soudjani is supported by the UK EPSRC New Investigator Award CodeCPS (EP/V043676/1) and the European Research Council project SymAware (101070802).

References

1. Tao F, Zhang H, Liu A et al. Digital Twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics* 2019; 15(4): 2405–2415. DOI:10.1109/TII.2018.2873186.
2. Feng H, Gomes C, Thule C et al. Introduction to Digital Twin Engineering. In *2021 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE. DOI:10.23919/annsim52504.2021.9552135.
3. Feng H, Gomes C, Gil S et al. Integration of the Mape-K loop in digital twins. In *2022 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE. DOI:10.23919/annsim55834.2022.9859489.
4. Althoff M, Frehse G and Girard A. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems* 2021; 4(1): 369–395. DOI:10.1146/annurev-control-071420-081941.
5. Wright T, Gomes C and Woodcock J. Formally verified self-adaptation of an incubator Digital Twin. In *Leveraging Applications of Formal Methods, Verification and Validation. Practice: 11th International Symposium (ISoLA 2022), Part IV*. p. 89–109. DOI:10.1007/978-3-031-19762-8_7.
6. Junghanns A, Gomes C, Schulze C et al. The functional mock-up interface 3.0 - new features enabling new applications. In *Linköping Electronic Conference Proceedings*. Linköping University Electronic Press, 2021. DOI:10.3384/ecp2118117.
7. Bogomolov S, Fitzgerald J, Soudjani S et al. Data-driven reachability analysis of Digital Twin FMI models. In *International Symposium on Leveraging Applications of Formal Methods (ISOLA)*. Springer Nature Switzerland, 2022. pp. 139–158. DOI:10.1007/978-3-031-19762-8_10.
8. Chen X, Ábrahám E and Sankaranarayanan S. Flow*: An analyzer for non-linear hybrid systems. In Sharygina N and Veith H (eds.) *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 258–263. DOI:10.1007/978-3-642-39799-8_18.
9. Frehse G, Guernic CL, Donzé A et al. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification*. Springer Berlin Heidelberg, 2011. pp. 379–395. DOI:10.1007/978-3-642-22110-1_30.
10. Bogomolov S, Forets M, Frehse G et al. JuliaReach: A toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. HSCC '19, New York, NY, USA: Association for Computing Machinery, p. 39–44. DOI:10.1145/3302504.3311804.
11. Ray R, Gurung A, Das B et al. XSpeed: Accelerating reachability analysis on multi-core processors. In *11th International Haifa Verification Conference (HVC 2015)*, *LNCS*, volume 9434. Springer, pp. 3–18. DOI:10.1007/978-3-319-26287-1_1.
12. Donzé A and Maler O. Systematic simulation using sensitivity analysis. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control*. HSCC'07, Berlin, Heidelberg: Springer-Verlag, p. 174–189. DOI:10.1007/978-3-540-71493-4_16.
13. Hiskens I and Pai M. Trajectory sensitivity analysis of hybrid systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 2000; 47(2): 204–220. DOI:10.1109/81.828574.
14. Geng S and Hiskens I. Jump Conditions for Second-Order Trajectory Sensitivities at Events. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. pp. 1–5. DOI:10.1109/ISCAS.2018.8351697.
15. Donzé A. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In Touili T, Cook B and Jackson P (eds.) *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 167–170. DOI:10.1007/978-3-642-14295-6_17.
16. Duggirala PS, Mitra S and Viswanathan M. Verification of annotated models from executions. In *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)*. pp. 1–10. DOI:10.1109/EMSOFT.2013.6658604.
17. Duggirala PS, Mitra S, Viswanathan M et al. C2E2: A verification tool for stateflow models. In Baier C and Tinelli C (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-662-46681-0, pp. 68–82. DOI:10.1007/978-3-662-46681-0_5.
18. Fan C and Mitra S. *Data-Driven Safety Verification of Complex Cyber-Physical Systems*. Cham: Springer International Publishing, 2019. pp. 107–142. DOI:10.1007/978-3-030-13050-3_5.
19. Fan C, Qi B, Mitra S et al. DryVR: Data-driven verification and compositional reasoning for automotive systems. In Majumdar R and Kunčak V (eds.) *Computer Aided Verification*. Cham: Springer International Publishing, pp. 441–461. DOI:10.1007/978-3-319-63387-9_22.
20. Ren H and Kumar R. Step Simulation/Overapproximation-Based Verification of Nonlinear Deterministic Hybrid System with Inputs. *IFAC-PapersOnLine* 2015; 48(27): 21–26. DOI:10.1016/j.ifacol.2015.11.147.
21. Ren H and Kumar R. Simulation-based verification of bounded-horizon safety for hybrid systems using dynamic number of simulations. *IET Cyber-Physical Systems: Theory & Applications* 2019; 4(3): 250–258. DOI:10.1049/iet-cps.2018.5017.
22. Girard A and Pappas G. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*. pp. 684–689. DOI:10.1109/CDC.2005.1582235.

23. Kapinski J, Krogh BH, Maler O et al. On systematic simulation of open continuous systems. In Maler O and Pnueli A (eds.) *Hybrid Systems: Computation and Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 283–297. DOI: 10.1007/3-540-36580-X_22.
24. Xue B, Zhang M, Easwaran A et al. PAC Model Checking of Black-Box Continuous-Time Dynamical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 2020; 39(11): 3944–3955. DOI:10.1109/TCAD.2020.3012251. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
25. Alanwar A, Koch A, Allgöwer F et al. Data-driven reachability analysis from noisy data. *IEEE Transactions on Automatic Control* 2023; 68(5): 3054–3069. DOI:10.1109/tac.2023.3257167.
26. Lew T and Pavone M. Sampling-based reachability analysis: A random set theory approach with adversarial sampling. In Kober J, Ramos F and Tomlin C (eds.) *Proceedings of the 2020 Conference on Robot Learning*, *Proceedings of Machine Learning Research*, volume 155. PMLR, pp. 2055–2070.
27. Sun D and Mitra S. NeuReach: Learning Reachability Functions from Simulations. In Fisman D and Rosu G (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. Cham: Springer International Publishing, pp. 322–337.
28. Coënt AL, dit Sandretto JA and Chapoutot A. Guaranteed master for interval-based cosimulation. *Software and Systems Modeling* 2021; 20(3): 711–724. DOI:10.1007/s10270-020-00858-7.
29. Gomes C, Thule C, Broman D et al. Co-Simulation: A Survey. *ACM Comput Surv* 2018; 51(3). DOI:10.1145/3179993.
30. Gajda K, Jankowska M, Marciniak A et al. A survey of interval runge–kutta and multistep methods for solving the initial value problem. In *Parallel Processing and Applied Mathematics*. Springer Berlin Heidelberg, 2008. pp. 1361–1371. DOI: 10.1007/978-3-540-68111-3_144.
31. Chutinan A and Krogh BH. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid Systems: Computation and Control*. Springer Berlin Heidelberg, 1999. pp. 76–90. DOI:10.1007/3-540-48983-5_10.
32. Fitzgerald J, Larsen PG and Verhoef M. Collaborative design for embedded systems. *Academic Press* 2014; 10: 978–3.
33. Blochwitz T, Otter M, Arnold M et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference*. pp. 105 – 114. DOI:10.3384/ecp11063105.
34. The MathWorks. Simulink User's Guide, 2021.
35. Fritzson P, Aronsson P, Pop A et al. OpenModelica - a free open-source environment for system modeling, simulation, and teaching. In *2006 IEEE Conference on Computer Aided Control System Design*. pp. 1588–1595. DOI:10.1109/CACSD-CCA-ISIC.2006.4776878.
36. Larsen PG, Fitzgerald J, Woodcock J et al. Integrated tool chain for model-based design of Cyber-Physical Systems: The INTO-CPS project. In *2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data)*. pp. 1 – 6. DOI:10.1109/CPSData.2016.7496424.
37. Robinson RC. Scalar Ordinary Differential Equations. Technical report, 2013. URL https://sites.math.northwestern.edu/~clark/dyn-sys/scalar.pdf.
38. Tempo R, Calafiore G and Dabbene F. *Randomized algorithms for analysis and control of uncertain systems: with applications*. Springer Science & Business Media, 2012.
39. Mohajerin Esfahani P, Sutter T and Lygeros J. Performance bounds for the scenario approach and an extension to a class of non-convex programs. *IEEE Transactions on Automatic Control* 2015; 60(1): 46–58. DOI:10.1109/TAC.2014.2330702.
40. Weng TW, Zhang H, Chen PY et al. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*. DOI: 10.48550/arXiv.1801.10578.
41. Wood G and Zhang B. Estimation of the Lipschitz constant of a function. *Journal of Global Optimization* 1996; 8(1): 91–103. DOI:10.1007/BF00229304.
42. De Haan L, Ferreira A and Ferreira A. *Extreme Value Theory: an introduction*, volume 21. Springer, 2006.
43. Frehse G, Althoff M, Schoitsch E et al. (eds.). *Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)*, *EPiC Series in Computing*, volume 90. EasyChair, 2022.
44. Virtanen P, Gommers R, Oliphant TE et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 2020; 17: 261–272. DOI:10.1038/s41592-019-0686-2.
45. Hindmarsh AC. ODEPACK, a systemized collection of ODE solvers. *Scientific computing* 1983; .
46. Diamond S and Boyd S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 2016; 17(83): 1–5.
47. Agrawal A, Verschueren R, Diamond S et al. A rewriting system for convex optimization problems. *Journal of Control and Decision* 2018; 5(1): 42–60.
48. Armstrong JS and Collopy F. Error measures for generalizing about forecasting methods: Empirical comparisons. *International journal of forecasting* 1992; 8(1): 69–80. DOI:10.1016/0169-2070(92)90008-W.
49. Wright T and Stark I. Property-Directed Verified monitoring of Signal Temporal Logic. In *Runtime Verification: 20th International Conference, RV 2020, Los Angeles, CA, USA, October 6–9, 2020, Proceedings*. Berlin, Heidelberg: Springer-Verlag, p. 339–358. DOI:10.1007/978-3-030-60508-7_19.