

OPTIMIZING FAULT INJECTION IN FMI CO-SIMULATION THROUGH SENSITIVITY PARTITIONING

Mehrdad Moradi, Cláudio Gomes, Bentley James Oakes, Joachim Denil

University of Antwerp, Prinsstraat 13, 2000 Antwerpen, Belgium
Flanders Make, Oude Diestersebaan 133, 3920 Lommel, Belgium
{mehrdad.moradi,claudio.gomes,bentley.oakes,joachim.denil}@uantwerpen.be

ABSTRACT

As society and industry relies extensively on Cyber-Physical Systems (CPS), any malfunctions can have unforeseen catastrophic failures. Fault Injection (FI) techniques perturb a model of a CPS with the intention of causing a failure and measuring the robustness of the CPS. Naturally, the success of a FI simulation depends on three factors: (i) the realism of the faults injected; (ii) how quickly the faults cause catastrophic failure; and (iii) the fidelity of the model used.

This paper proposes to improve the success rate of FI studies by addressing each one of these factors. An algorithm is presented that leverages traditional sensitivity analysis in hybrid systems to reduce an uncountable fault search space to a optimal finite set (factors i,ii), and we use co-simulation as the model integration technique (factor iii). We evaluate our contribution on the power window system developed by MathWorks®.

Keywords: fault injection, co-simulation, Functional Mock-up Interface (FMI) standard, sensitivity analysis, hybrid systems.

1 INTRODUCTION

Cyber-Physical Systems (CPS) should be robust against failure, especially in common safety-critical systems such as the automotive and aeronautic domains. For testing robustness, Fault Injection (FI) has become an important tool. FI consists of perturbing the system (either its inputs, or internal components) and observing the impact in the system's behavior (Pintard et al. 2013). Naturally, this is an expensive process if applied to physical prototypes of the system. As such, most FI studies are currently performed on models of the CPS (Kooli and Di Natale 2014, Svenningsson et al. 2010).

According to (Benso and Prinetto 2003), faults have three main configuration dimensions: the *type*, *location*, and *value*. These configuration points induce a *fault space*, which, when combined with the complexity of a CPS, creates an uncountable space of fault candidates. As well, even for reasonably simple systems, as we shown in Section 3, it is hard to predict the system's reaction to failure. The success of a FI study depends on the ability to inject relevant and realistic faults, that cause the system to fail.

Contributions In this paper, we propose a technique to increase the success rate of a FI study. The faults we consider are realistic, the fault space non-countable, and the technique can be applied to high fidelity simulations.

The faults are realistic because we support uncertain faults. For example, the voltage drop across a brushed Direct-Current (DC) motor increases at an uncertain rate over time, due to the wear of the brushes. By allowing an uncertain voltage drop, we relieve the engineers from having to know the exact voltage drop for a realistic failure.

An uncertain fault represents an uncountable set of possible faults, which makes it harder to tackle the fault coverage. To address this, we make use of traditional Sensitivity Analysis (SA) of hybrid systems (e.g., see (Barton and Lee 2002, Hiskens and Pai 2000)) to find sets of faults that cause similar system (abnormal) behavior. Our key contribution is that these sets can then be abstracted as a single fault, since all faults in the set share equivalent impact.

As high-fidelity models are expensive to build, we use *co-simulation* to improve the fidelity factor. Co-simulation is a technique to compute the behavior of a coupled system through the coordination of simulators of the corresponding subsystems (Gomes et al. 2018, Kübler and Schiehlen 2000, Hafner and Popper 2017, Palensky et al. 2017). The simulators consist of Functional Mock-up Units (FMU). These FMUs are essentially black boxes, in that they are coupled using a master algorithm which communicates with each simulator via a standardized interface. An example of such an interface is the Functional Mock-up Interface (FMI) standard (FMI 2014, Blockwitz et al. 2012), which we assume the reader is familiar with. Co-simulation allows us to build high fidelity simulations because the black boxes may be provided by third parties, and accumulate years of experience in the encapsulated models, without risking that intellectual property.

In this paper, we apply our techniques to an example of a power window. This example has well-known discontinuity characteristics. We assume that the master algorithm works correctly, in that it accurately reproduces the normal, as well as the faulty, behavior of the system.

Even though our technique works on co-simulations, it still needs to know when a discontinuity happened in the system. This is a mild assumption as this information can be easily exposed even in black box simulators, through discontinuity indicator ports.

The paper is organized as follows. Background information is provided in Section 2, with our case study presented in Section 3. Section 4 describes the elements of our technique, while Section 5 demonstrates applying this technique to the case study. Related work to our approach is found in Section 6. Finally, Section 7 presents a brief conclusion of our approach, and future research directions.

2 BACKGROUND

This section will briefly introduce the concepts required for this paper and provide the reader with further reading. Note that these concepts are exemplified in Section 3.

2.1 Hybrid Automata

Hybrid automata is a formalism used to model hybrid systems, which are systems that exhibit continuous behavior, interleaved with discontinuous mode changes. Informally, a hybrid automaton contains input, state, output variables, and multiple modes connected by transitions. Each mode contains a system of differential algebraic equations, involving input, state, and output variables. Each transition is a directed edge contains a trigger condition, and an action. The trigger condition represents an inequality relating state and I/O variables. The action encodes changes to be made to state and I/O variables, when the transition takes place.

In this paper, we adopt the *must-fire* semantics of hybrid automata. That is, a transition between a source mode and a target mode is triggered when the current mode of the automaton is the source mode, and as soon as the trigger condition of the automaton becomes true. The initial state of the automaton is always given. This means that the hybrid automata we consider are deterministic. For a formal definition of hybrid automata, see (Navarro-López and Carter 2011, Henzinger 2000), and (Frehse 2015).

2.2 Sensitivity Analysis

Sensitivity Analysis (SA) is a broad family of techniques that help identify the impact of changes to a model (Saltelli et al. 2004, Hutcheson and McAdams 2010). Here, we focus on local-trajectory sensitivity analysis of hybrid systems. In essence, when applied in a co-simulation setting, this technique consists of comparing a perturbed co-simulation with a non-perturbed one, for each possible parameter or initial state. The results provide a measure of the relative importance of each parameter, under the assumption that the change is small. For continuous systems, this assumption is reasonable, but not for hybrid automata (Hiskens and Pai 2000), which is why we adapt the technique to be usable in a co-simulation of hybrid systems.

Literature like (Saltelli et al. 2004, Hutcheson and McAdams 2010) provide a comprehensive overview of sensitivity analysis, and for efficient implementations of the technique in hybrid systems, we refer readers to (Barton and Lee 2002, Hiskens and Pai 2000), and (Galán et al. 1999).

3 MOTIVATING EXAMPLE

In this section, we introduce a simplified model of a power window control system. This example will be used throughout the paper to exemplify the techniques that we propose.

3.1 Description of Power Window

The model of the power window system, illustrated in Figure 1a, is a hybrid system. It is comprised of three FMUs implementing: a) a switch for controlling the window, b) a discontinuous software controller, and c) a continuous model of the physical window.

The controller is responsible for ensuring the safe operation of the window. In particular, it should detect obstructions to the window movement, such as an object being compressed when the window is closing, and act to ensure that no harmful forces are exerted on the obstruction.

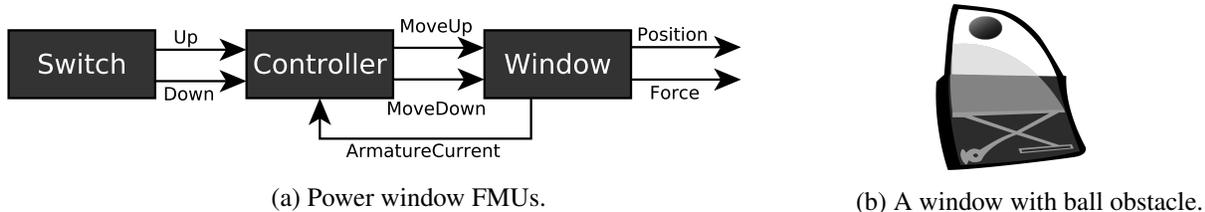


Figure 1: The power window motivating example.

In this paper, we will focus on a single scenario for the power window: the driver continuously pushes the `move-up` button of the window interface. Figures 2a and 2b show the behavior of the system. In both figures, the top plot shows armature current on the DC motor, the second plot shows the window's vertical position. Third plot shows the instructions of the control system to the DC motor, and last one is the force to the object.

As can be seen from the plots in Figure 2, whenever there is any obstruction to the movement of the window, be it Coulomb friction (cf. Figure 2a, interval $[0s, 0.5s]$), or the presence of an object (cf. Figure 2b, interval $[2s, 4s]$), the armature current in the DC motor spikes. The control system therefore uses the armature current to detect whether something has obstructed the movement of the window. To avoid false positives, there is a *cool-down* period of about one second after the controller instructs the motor to move the window up before an obstacle can be detected.

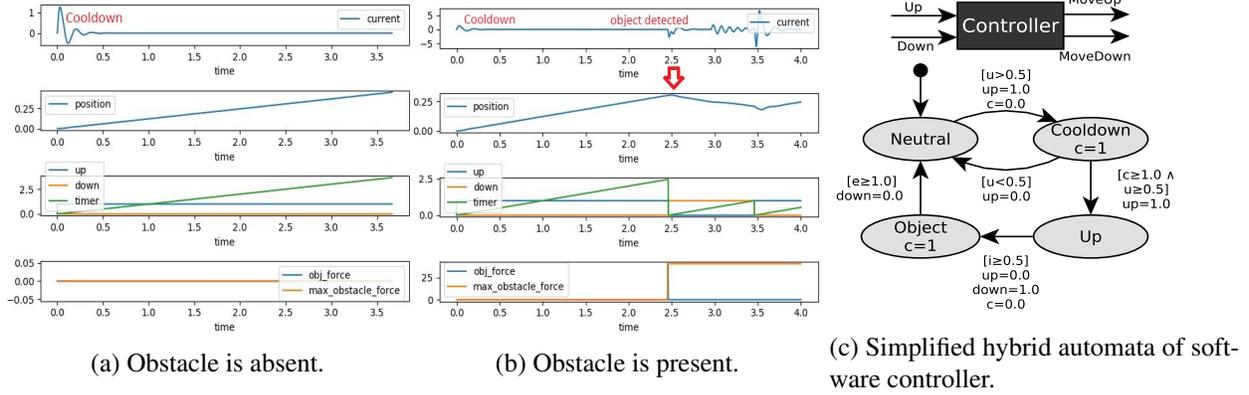


Figure 2: Power window traces and hybrid automata.

Figure 2c shows the simplified hybrid automata of the software controller. In the results section, we apply techniques to the full Simulink® example. The controller has four main modes of operation: *Neutral*, *Cooldown*, *Up*, and *Object*. The u variable represents the driver/passenger commands, and the $up/down$ variables represent the controller commands (to the motor). The c variable represents an internal clock, and the i represent the armature current reading. The conditions in brackets are monitored constantly and trigger the mode transition as soon as the condition becomes true.

3.1.1 Specifications for Correct Behavior

The power window system has to satisfy the requirements described in (Prabhu and Mosterman 2004). In this paper, we select two main requirements. Careful examination of Figures 2a and 2b shows that the power window system satisfies both these requirements.

Specification 1. *Window fully opened/closed within 4 s*

Specification 2. *The force to detect when an object is present should be less than 100 [N].*

3.1.2 Faults to Inject

As an example fault, consider the real-world operation of the power window. As the brushes in the DC motor wear down over time, the current passing through them meets more resistance. This causes a voltage drop on the motor, which makes it operate more slowly.

A realistic fault is therefore to adjust the operating voltage of the DC motor. However, uncertainty lies in how much the voltage is decreased by. For illustrative purposes, we suppose that the voltage lies in the interval $[7, 12]$, where 12 is the normal operation voltage. Such fault can be represented as a hybrid automaton with a single state setting the output to a value in the interval $[7, 12]$, which is then connected to the window system’s voltage parameter.

3.1.3 Sensitivity

We exemplify the sensitivity of specification 1 with respect to the fault introduced above. Figure 3 shows two co-simulations: one where the DC motor operates with 12V (normal behavior), and another where it operates with 7V.

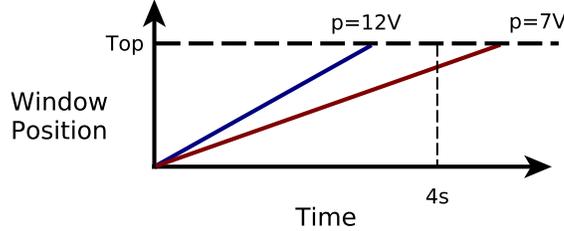


Figure 3: Sensitivity of the motor Simulink® model.

Suppose there is a function o that measures how well specification 1 is satisfied. That is, if a simulation result produces $o > 0$, then the specification is satisfied. If, $o = 0$, then the specification is not satisfied. As Section 4 shows, such function can be implemented as a hybrid automata that outputs the value $o = 4s - TimeToReachTop$ when the window has reached the top.

Assuming that the system is continuous with respect to the voltage parameter in the interval $[7, 12]$, we can estimate the sensitivity of o w.r.t. the voltage parameter V as $\frac{\partial o}{\partial V} \approx (-1 - 1)/(7 - 12)$. This value can then be used to estimate the voltage drop $V_0 \in [7, 12]$ that causes specification 1 to fail, that is, $o(V_0) \approx 0$: $V_0 = 12 - o(7)/\frac{\partial o}{\partial V}$.

Our technique will assist the user in configuring the faults by determining the ranges of parameters under which the continuity assumption holds, and therefore specification breaking points can be identified.

4 APPROACH FOR CONFIGURING FAULT INJECTION

In this section, we describe our technique using the power window as a running example. Before detailing the main contribution, we explain how the faults and specifications are defined. To make our exposition clearer, and without loss of generality, we will assume that there is a single fault to be injected, with a single uncertain parameter, and a single specification being measured.

The faults and specifications are FMUs as detailed in Figure 5 and Figure 6, and orchestrated in a co-simulation using a Jacobi algorithm. By assuming that the co-simulations are accurate, any disturbance on the behavior computed by the co-simulation is due to the faults, and not the co-simulation algorithm. For the power window system, Figure 4 illustrates the co-simulation scenario. In this scenario, Figure 5b shows the fault and Figure 6b displays the specification, as both an FMU and a hybrid automata.

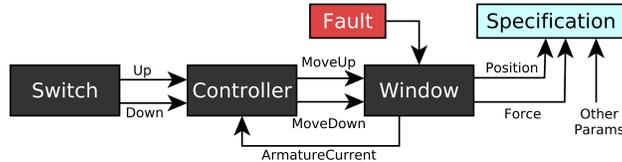


Figure 4: Power window FMU in co-simulation.

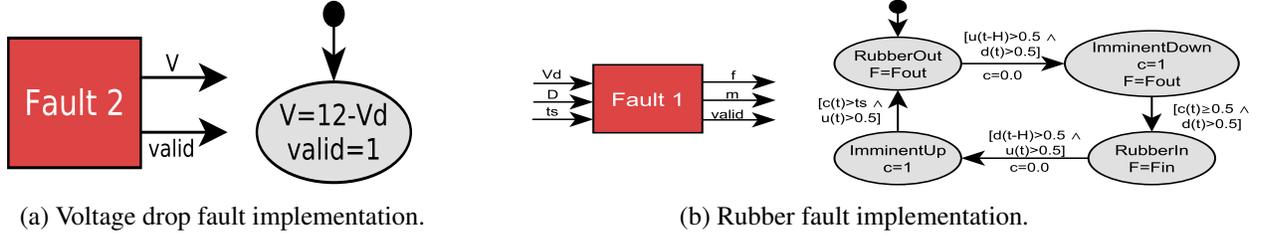


Figure 5: Fault implementations and their interface.

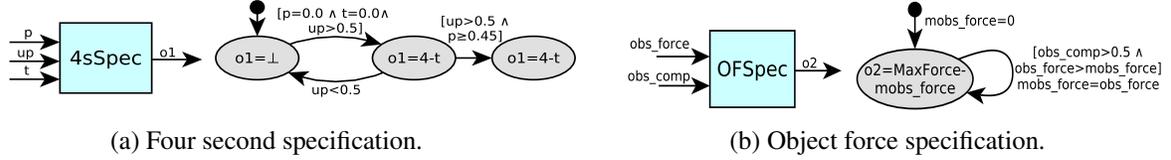


Figure 6: Implementation example of specifications.

4.1 Defining the Faults

Each fault is assumed to have at least one input that allows us to control the fault intensity (or deactivate the fault). Note that the fault intensity does not mean that more intensity leads to a greater degree of specification failure. As for outputs, we assume that each fault discloses its discrete mode signal, and whether the fault is valid or not. The validity is important because faults can have complex pre-conditions. For example, the DC motor voltage drop fault, introduced in Section 2.2, is represented as a hybrid automaton in Figure 5a.

As a more complex example, consider the example in Figure 5b, where the rubber seal slips into the door as the window moves down. The effect is an adjustment of the friction on the window movement. The fault is valid unless the co-simulation causes it to deadlock.

In Figure 5b, f_{out} is a constant denoting the normal friction of the rubber, f_{in} is the friction caused by the rubber slipping into the door, and H corresponds to the co-simulation step size, so that $d(t-H)$ refers to the value of signal d at the previous co-simulation step. To keep the explanation simple, the uncertainty in this fault is captured only by the input t_s , which controls the timing of the rubber slipping out.

4.2 Defining the Specifications

In order to automatically measure the degree to which a co-simulation satisfies the specifications, we will assume the existence of an oracle for each specification, that receives values computed in the co-simulation, and outputs a specification measure, or \perp , in case such a measure is not applicable (i.e., the pre-conditions are not satisfied), and a signal encapsulating its internal discrete mode. Figures 6a and 6b show the implementation of specifications 1 and 2.

In Figure 6b, p denotes the position of the window, up is the controller command signal, t is the time, and $o1$ is the measure of the specification. The output $o1 \geq 0$ means the co-simulation satisfies the specification, and violates it otherwise. The inputs and outputs of the specifications are functions of time. For Figure 6b, when force to object is greater than a specified value, $o2$ became negative and specification is violated. Note that the oracle will produce a measure of the specification (different than \perp) only when the window moves from fully open to fully closed.

4.3 Parameter Interval Partitioning

The first part of our technique lies in identifying the fault intensity intervals for which the behavior of the co-simulation does not change substantially. We consider two co-simulation traces to be substantially different if the sequence of discrete modes of at least one FMU is different. Given a co-simulation trace, we define the aggregated untimed sequence of modes as the set product of all discrete mode signals converted to untimed sequences, preserving only causality. For example, the aggregated untimed version of the discrete mode signals *up* and *down* plotted in Figure 2b is $(1,0), (0,1), (1,0)$. This operation allows us to define an equivalence relation: two co-simulation results are equivalent iff their aggregated untimed sequence of modes is the same. With these definitions, our technique is described next.

Procedure 1. *Given a co-simulation scenario that includes a fault and a specification FMU:*

1. *Let $N \in \mathbb{N}$ be a parameter given by the user;*
2. *Let the fault intensity input p be in the range $[0, 1]$ with 0 being the lowest intensity and 1 being the highest;*
3. *Let $P = \{p_0, p_1, \dots, p_N\}$ be a set where $p_i = \frac{i}{N}$;*
4. *For each $p_i \in P$:*
 - (a) *Run cosim with fault intensity p_i ;*
 - (b) *Let o_i be the resulting specification measure;*
 - (c) *Let m_i be the resulting aggregated untimed sequence of discrete modes of all FMUs (including specification and fault);*
5. *Create a partition of P according to the equivalence relation defined above. In other words, p_i and p_{i+1} are equivalent if m_i is equivalent to m_{i+1} , for $i = 0, \dots, N$.*
6. *Return the partition.*

An example application of Procedure 1 is given in Section 5. We assume that every FMU in the co-simulation scenario discloses its interval mode as a signal.

4.4 Determining Sensitivity

Given a partition created with Procedure 1, and two values p_i and p_{i+1} in the same set, the sensitivity equation holds:

$$\frac{\partial o}{\partial p} \approx \frac{o_{i+1} - o_i}{p_{i+1} - p_i}. \quad (1)$$

Within the same set of equivalent fault intensity values, traditional optimization techniques can be applied to find the maximum, minimum, and zero of the specification.

4.5 Ranking and Configuring Faults

From each equivalence class of fault intensity parameter, we pick the value that minimizes the specification measure, which is then ranked against the best of the other classes. This ranking is input for the fault injection part of our technique, which extracts values based on how close they are in violating the specification. Hence, our technique can rank values in a large fault space instead of exploring it totally.

Until now we have explained technique for a single fault intensity parameter and a single specification. Extending this to multiple specifications is straightforward: Procedure 1 stores the results of multiple specification measures, and Equation (1) becomes a vector. Note however that the discrete modes of each specification have to be taken into account when determining the equivalence classes.

In our framework, we model the injection of multiple faults (such as rubber slip-in and DC motor wear-out) as instead a single fault with multiple intensity parameters. When multiple faults are given, Procedure 1 is applied to each one individually. The ranking is then performed as described above. The next section gives examples of this procedure.

5 RESULTS

In this section, we apply Procedure 1 to the power window system introduced in Section 3. We consider only the fault where rubber slips into the window, as detailed in Figure 5b, and the object compression specification, introduced in Section 3.1.1 and detailed in Figure 6b. In Figure 6b, $\text{MaxForce} = 100$ is a constant, and the mobs_force keeps track of the maximum force exerted on the object, whenever it is being compressed (signaled by the input obs_comp).

Figure 7 shows the co-simulation results for increasing values of fault intensity parameter t_s . As can be seen, when $t_s \in [0.0, 0.7]$, there is no substantial difference in the behavior. That is, the discrete modes (software controller, and object being compressed) do not change. However, when $t_s = 0.9$, in Figure 7e, there is a specification violation (object force exceeds $100N$). This violation deserves clarification: as the motor is drawing more current to move the window up under extra friction, the rubber slips out. This causes an impulse in the window movement and subsequent oscillations.

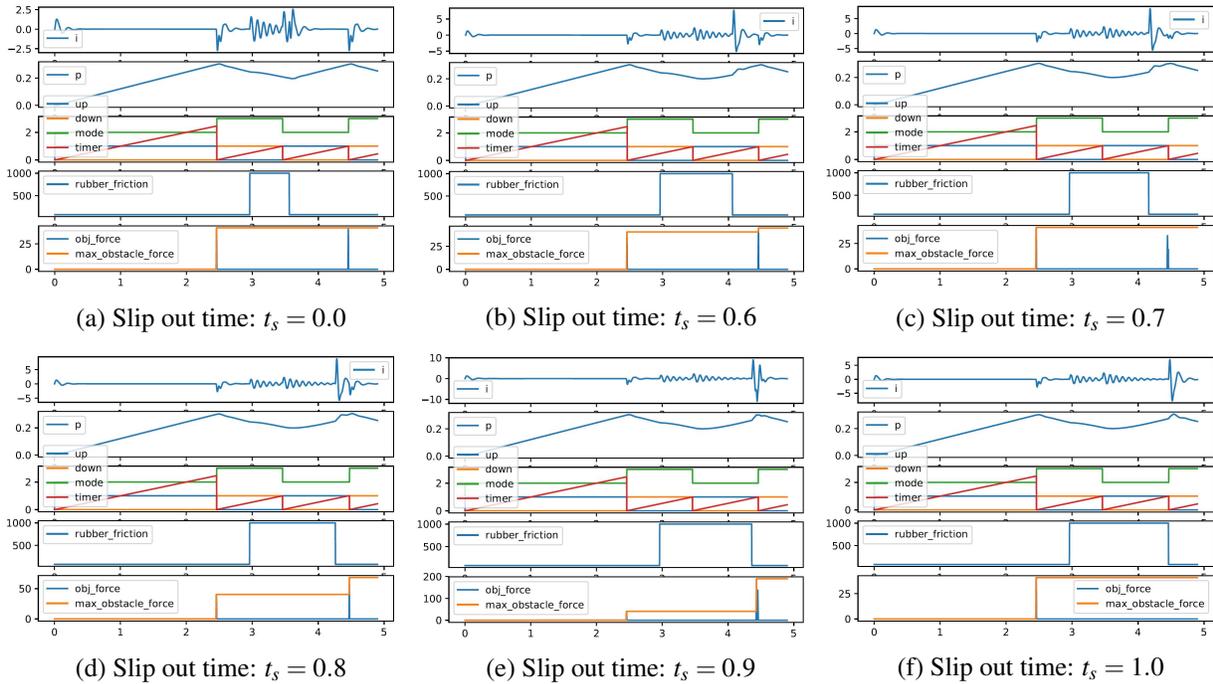


Figure 7: Rubber Slip fault analysis results.

6 RELATED WORK

To the best of our knowledge, there is no work that uses an interval partitioning technique like the one we propose in Procedure 1 to guide fault injection in FMI co-simulation. However, related work can be found in methods, techniques and tools that leverage SA.

Reference (Hiskens and Pai 2000) presents a technique to compute the trajectory SA for a hybrid system, more specifically described using differential-algebraic-discrete models. The work in (Galán and Barton 1998) uses earlier results to elicit a classification of optimization problems for hybrid systems. It focuses on using the sensitivity for solving an optimization problem. In contrast, in our work, we leverage SA when the sequence of modes of the nominal trajectory, and its perturbed trajectory, are the same. Hence, if an event ceases to exist due to the perturbation, we conclude that the sensitivity to that perturbation is infinite.

Furthermore, different techniques are available to compute the sensitivities in different formalisms and tools. E.g. (Han and Mosterman 2013) proposed a method that enables direct SA on system models via an implementation in the Simulink software. The approach relies on the existing ordinary differential equation solver of Simulink and the block-by-block analytic Jacobian computation to provide the analytic Jacobian for solving the sensitivity equations.

Reference (Kane et al. 2014) describes a monitor to detect faulty system behavior with run-time verification using hardware-in-the-loop. They study the system under faults such as random bit flips, random value injections, and exceptional value injections. Verification of the system is performed with a domain-specific language for monitoring, with properties specified in Metric Temporal Logic. An important argument of that work is the fact that any run-time monitors require access to the source code, and this is not feasible with commercial systems.

7 CONCLUSION

In this paper we proposed a method based on Sensitivity Analysis (SA) in the FMI co-simulation to improve the success rate of Fault Injection (FI) studies. This method differs from previous studies in two key ways. First, this framework uses SA to understand the system behavior. Then, the framework reduces a uncountable fault space to a countable space of equivalence classes, which can then be ranked according to the impact they have in the specifications. Second, our technique works for black box co-simulations, under the assumption that the discontinuities of each simulator are published. This means that high fidelity models can be used in the FI study.

7.1 Risks

Limitation of our exploratory study need to be acknowledged, most notably the limits of Procedure 1. First, we make the assumption that the co-simulation results are accurate, which is in itself an open research question (see, e.g., (Gomes, Oakes, Moradi, Gamiz, Mendo, Dutre, Denil, and Vangheluwe 2019)) Second, Procedure 1 has to run N co-simulations, and N determines the accuracy of the partitioning scheme. If N is too small, there might be values p_i, p_{i+1} , for which the resulting discrete modes are the same, but there exists a value $p_i < p_j < p_{i+1}$ for which the behavior is different. This value has not been considered because of the choice of N . Third, there's no information telling the user how to map domain faults to parameter faults in the co-simulation. This type of mapping relates to the knowledge about system implementation, the underlying computations, and the experience of the FI user.

7.2 Future Work

Future research will tackle the aforementioned risks in the following ways. First, we will explore how to reuse prior simulations to reduce the need for future simulations. A starting point is to use branching simulations, as is used in (Lawrence, Gomes, Denil, Vangheluwe, and Buchs 2016). Second, we will investigate multiple points of FI, including injecting sequences of faults. Sensitivity regarding delay should be taken into account (tools like Jitterbug and Jittertime (Lincoln and Cervin 2002) can perform this analysis in

Simulink). Additionally, in (Christopher Frey and Patil 2002, Gomes et al. 2017), the authors describe some methods for SA and automatic differentiation, that we could use. In order to help the user map domain faults to parameter faults, we propose to use of a domain-specific language which could guide FI to a specific part of model with the fault parameters that the user intends to simulate.

Other domains like model-based testing are also related to FI. We can use their methodology for finding the best order of scheduling co-simulation to optimize the run time.

ACKNOWLEDGEMENTS

This work is partly supported by INES (Innovation in the Development of Electrical Systems For Aeronautics) under project no. 11172. We thank Hans Vangheluwe, Romain Franceschini, our colleagues at Siemens Industry Software in Leuven, Belgium, and our colleagues at Boeing Research & Technology Europe. Additionally, this research was partially supported by a PhD fellowship grant from the Research Foundation - Flanders (File Number 1S06316N).

REFERENCES

- Barton, P. I., and C. K. Lee. 2002. "Modeling, Simulation, Sensitivity Analysis, and Optimization of Hybrid Systems". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* vol. 12 (4), pp. 256–289.
- Benso, A., and P. Prinetto. 2003. *Fault injection techniques and tools for embedded systems reliability evaluation*, Volume 23. Springer Science & Business Media.
- Blockwitz, T., M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel. 2012, November. "Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models". In *9th International Modelica Conference*, pp. 173–184. Munich, Germany, Linköping University Electronic Press.
- Christopher Frey, H., and S. R. Patil. 2002. "Identification and Review of Sensitivity Analysis Methods doi:10.1111/0272-4332.00039". *Risk Analysis* vol. 22 (3), pp. 553–578.
- FMI 2014. "Functional Mock-up Interface for Model Exchange and Co-Simulation". Technical report, FMI development group.
- Frehse, G. 2015. "An Introduction to Hybrid Automata, Numerical Simulation and Reachability Analysis". In *Formal Modeling and Verification of Cyber-Physical Systems*, pp. 50–81. Wiesbaden, Springer Fachmedien Wiesbaden.
- Galán, S., and P. I. Barton. 1998, March. "Dynamic Optimization of Hybrid Systems". *Computers & Chemical Engineering* vol. 22, Supple, pp. S183–S190.
- Galán, S., W. F. Feehery, and P. I. Barton. 1999. "Parametric sensitivity functions for hybrid discrete/continuous systems". *Applied Numerical Mathematics* vol. 31 (1), pp. 17–47.
- Gomes, C., B. J. Oakes, M. Moradi, A. T. Gamiz, J. C. Mendo, S. Dutre, J. Denil, and H. Vangheluwe. 2019. "HintCO - Hint-Based Configuration of Co-Simulations". In *International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pp. accepted. Prague, Czech Republic.
- Gomes, C., C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. 2018, April. "Co-Simulation: A Survey". *ACM Computing Surveys* vol. 51 (3), pp. Article 49.
- Gomes, C., Y. Van Tendeloo, J. Denil, P. De Meulenaere, and H. Vangheluwe. 2017. "Hybrid System Modelling and Simulation with Dirac Deltas". In *Proceedings of the Symposium on Theory of Modeling*

- & Simulation: *DEVS Integrative M&S Symposium*, pp. Article No. 7. Virginia Beach, Virginia, USA, Society for Computer Simulation International. Series Title: DEVS '17.
- Hafner, I., and N. Popper. 2017. "On the terminology and structuring of co-simulation methods". *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools* vol. 3, pp. 67–76.
- Han, Z., and P. J. Mosterman. 2013. "Towards sensitivity analysis of hybrid systems using Simulink". In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pp. 95–100. ACM.
- Henzinger, T. A. 2000. "The Theory of Hybrid Automata". In *Verification of Digital and Hybrid Systems*, pp. 265–292. Springer.
- Hiskens, I. A., and M. Pai. 2000. "Trajectory sensitivity analysis of hybrid systems". *IEEE Transactions on Circuits and Systems I* vol. 47 (2), pp. 204–220.
- Hutcheson, R. S., and D. A. McAdams. 2010. "A Hybrid Sensitivity Analysis for Use in Early Design". *Journal of Mechanical Design* vol. 132 (11), pp. 111007.
- Kane, A., T. Fuhrman, and P. Koopman. 2014, June. "Monitor Based Oracles for Cyber-Physical System Testing: Practical Experience Report". In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 148–155. Atlanta, GA, USA, IEEE.
- Kooli, M., and G. Di Natale. 2014. "A survey on simulation-based fault injection tools for complex systems". In *2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–6. IEEE.
- Kübler, R., and W. Schiehlen. 2000, June. "Two Methods of Simulator Coupling". *Mathematical and Computer Modelling of Dynamical Systems* vol. 6 (2), pp. 93–113.
- Lawrence, D. P. Y., C. Gomes, J. Denil, H. Vangheluwe, and D. Buchs. 2016. "Coupling Petri Nets with Deterministic Formalisms Using Co-Simulation". In *Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pp. 6:1–6:8. Pasadena, CA, USA.
- Lincoln, B., and A. Cervin. 2002, Dec. "JITTERBUG: a tool for analysis of real-time control performance". In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, Volume 2, pp. 1319–1324 vol.2.
- Navarro-López, E. M., and R. Carter. 2011, November. "Hybrid Automata: An Insight into the Discrete Abstraction of Discontinuous Systems". *International Journal of Systems Science* vol. 42 (11), pp. 1883–1898.
- Palensky, P., A. A. V. D. Meer, C. D. Lopez, A. Joseph, and K. Pan. 2017. "Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling". *IEEE Industrial Electronics Magazine* vol. 11 (1), pp. 34–50.
- Pintard, L., J.-C. Fabre, K. Kanoun, M. Leeman, and M. Roy. 2013. "Fault Injection in the Automotive Standard ISO 26262: An Initial Approach". In *Dependable Computing*, pp. 126–133. Berlin, Heidelberg, Springer Berlin Heidelberg.
- Prabhu, S. M., and P. J. Mosterman. 2004. "Model-Based Design of a Power Window System: Modeling, Simulation and Validation". In *Proceedings of IMAC-XXII: A Conference on Structural Dynamics, Society for Experimental Mechanics, Inc., Dearborn, MI*.
- Saltelli, A., S. Tarantola, F. Campolongo, and M. Ratto. 2004. "Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models". *Chichester, England*.

Svenningsson, R., J. Vinter, H. Eriksson, and M. Törngren. 2010. “MODIFI: a MODEL-implemented fault injection tool”. In *International Conference on Computer Safety, Reliability, and Security*, pp. 210–222. Springer.

AUTHOR BIOGRAPHIES

MEHRDAD MORADI is a Ph.D. student at the University of Antwerp (Belgium). The topic of his Ph.D. is the fault injection and co-simulation. His email address is mehrdad.moradi@uantwerp.be and his web address is <https://www.uantwerpen.be/nl/personeel/mehrdad-moradi>.

CLÁUDIO GOMES is a Ph.D. student at the University of Antwerp (Belgium). The topic of his Ph.D. is the foundations of co-simulation. His email address is claudio.gomes@uantwerp.be and his web address is <http://msdl.cs.mcgill.ca/people/claudio>.

BENTLEY JAMES OAKES is a post-doctoral researcher at the University of Antwerp (Belgium). His research focuses on the engineering of domain-specific languages and formal verification. His email address is bentley.oakes@uantwerpen.be and his web address is <http://msdl.cs.mcgill.ca/people/bentley>.

JOACHIM DENIL is an Assistant Professor in the University of Antwerp (Belgium). His main research interest is multi-paradigm modeling of software-intensive systems. His email address is joachim.denil@uantwerpen.be and his web address is <http://msdl.cs.mcgill.ca/people/joachim>.