Modeling the Wind Turbine Benchmark with PWA Hybrid Automata (Experience Report)

Nikolaos Kekatos, Marcelo Forets, Goran Frehse

Modeling the Wind Turbine Benchmark with PWA Hybrid Automata (Experience Report)

Nikolaos Kekatos[1], Marcelo Forets[1], and Goran Frehse[1*]

University Grenoble Alpes, Verimag, France
{nikolaos.kekatos, marcelo.forets-irurtia, goran.frehse}@univ-grenoble-alpes.fr

Abstract

The wind turbine benchmark is part of the ARCH benchmark repository. It entails closed-loop requirements and encompasses nonlinear and hybrid dynamics. Owing its origin to industry based applications, the benchmark modeling is done with MATLAB/Simulink. Formal verification tools, however, do not operate on simulation models but on formal models, such as hybrid automata. Particularly efficient verification algorithms are known for systems with Piecewise Affine (PWA) dynamics. In this vein, we construct a PWA model of the wind turbine in the SX format, which formally describes a network of hybrid automata and can be used by several reachability tools. The model transformation follows a four-step approach with the aim of (i) adapting the Simulink model to obtain a verification model (ii) translating the Simulink blocks to equivalent blocks in SX format, (iii) conducting compositional, syntactic hybridization to obtain a PWA approximation of the dynamics of the nonlinear blocks, and (iv) performing model validation. We also report some preliminary experiments on the subsystems (network components) of the wind turbine that we conducted with SpaceEx.

1 Introduction

The benchmark was introduced by Simone Schuler, Fabiano Daher Adegas, and Adolfo Anta from General Electrics [18] in the ARCH workshop. The authors presented a simplified nonlinear model of a wind turbine equipped with switching controllers. The composition of the wind turbine and the controllers results in a hybrid system with nonlinear dynamics. Given that wind turbine systems form one of the fastest-growing industries in renewable energy worldwide, the authors consider them as a promising area for the application of verification tools. The accompanied control objectives including safety and performance constraints establish a pertinent benchmark for verification of requirements in hybrid systems [18].

*The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921 and by the Metro Grenoble through the project NANO2017.

The main issue that arises is that formal verification tools do not operate on simulation models but on formal models. As such, the MATLAB/Simulink model has to be transformed in order for the requirements of the wind turbine benchmark to be verified. This paper describes the process of constructing a Piecewise Affine (PWA) model expressed in SX format (SpaceEx modeling language) [8] from the Simulink model and presents preliminary verification results undertaken with the SpaceEx tool [14]. The model transformation includes four main steps. We first change the Simulink model to comply with the verification standards (adding nondeterminism, deleting unnecessary blocks). Then, we employ the SL2SX tool, [17] to assist with the translation and the process of building the SpaceEx model. Afterwards, we conduct compositional syntactic hybridization [15] to obtain a PWA approximation of the dynamics of the nonlinear blocks and finally we perform model validation to check that the individual approximations (base components) are correct and non-blocking.

Much work has been done towards the verification of Simulink models [3] and several tools have been designed to facilitate this task [19]. There are two main groups of approaches. The first one refers to *verification by simulation* methodologies that can be applied directly on the Simulink model and do not require any model transformation. Donzé [9] presented a MATLAB/Simulink based tool, *Breach*, which performs simulation-based verification (approximate reachability analysis) and conducts efficient signal monitoring of properties and requirements. Another relevant MATLAB/Simulink tool is S-Taliro [4]. It conducts fast and efficient simulations, with a formal focus on falsification. These two approaches have the drawback that the set of initial states must be sampled and are restricted to models with low-dimensional initial states. A tool that circumvents this pitfall is HySon [7]. It performs set-based simulation on Simulink models, and approximates the set of all possible executions. However, it is not publicly available and, to the best of our knowledge, it has been only tested in cases of relatively small size. The second group of approaches relies on model transformation and requires the construction of a verification model from its Simulink counterpart. Once the verification model is designed, different formal verification tools could be applied. Filipovikj et al. [12] transformed Simulink models into the input language of UPPAAL and used the UPPAAL Statistical Model checker to verify properties of automotive systems. The Simulink-based tool C2E2 [11] transforms Stateflow models to polynomial hybrid automata, saves them in the HyXML format, and then conducts simulation-based verification.

The rest of this paper is organized as follows. Section 2 reviews the existing tools that are utilized. In Section 3, we briefly present the benchmark model. In Section 4, we describe the model transformation process. We report some preliminary results in Section 5. We draw conclusions and describe our future work in Section 6. The model and configuration files for running the case study in SpaceEx are available as an attachment.

2 Background

In this section, we briefly present the three tools that are used for the model transformation, namely, Simulink, SpaceEx and SL2SX, and describe the hybridization technique that we apply.

2.1 Simulink

Simulink [1] is a graphical programming environment for modeling, simulating and analyzing dynamical systems. It includes a set of block libraries and is commonly used in automatic control and signal processing for multidomain simulation and Model-Based Design. A Simulink model consists of a set of input and outputs variables, blocks and connections.

2.2 SpaceEx

SpaceEx [14] is a verification platform for hybrid systems. Its main functionality is to verify that a given mathematical model of a hybrid system satisfies desired safety properties. SpaceEx offers three different algorithms for set-based reachability analysis, namely the PHAVer, the LGG and the STC, as well as a simulation option. The PHAVer (Polyhedral Hybrid Automaton Verifyer) algorithm is applicable to linear hybrid automata, i.e. hybrid systems with piecewise constant bounds on the derivatives, and produces exact reachibility results. The LGG Support Function algorithm implements a variant of the Le Guernic-Girard (LGG) algorithm [16]. The STC (Space-Time approximation with Clustering) algorithm relies on LGG but it computes fewer convex sets for a given accuracy and yields more precise results for discrete transitions. These algorithms over-approximate the reachable sets and can be applied to hybrid systems with piecewise affine dynamics and non-deterministic inputs.

The models for SpaceEx are specified in the SX modeling language [8]. The format is similar to the standard hybrid automata, syntactically extended with hierarchy and templates. An SX model consists of base and network components as well as binds. Base components correspond to single hybrid automata, whereas network components correspond to the parallel composition of several hybrid automata. A bind instantiates a component inside a network (similar to a template instantiation), possibly remapping its variables to variables of the network. The parameters of a component can be variables (changing over time), constants or synchronization labels (every transition is associated with a label). The variables can be real-valued or boolean (depending on their type), local or global (depending on their usage outside the base component) and controllable or uncontrollable (typically, distinguishing between variables that appear in differential and algebraic equations). More information can be found at [10].

SpaceEx requires two input files to run, a model file of the hybrid automata in SX format and a configuration file that specifies the analysis options and preferences. In the latter file, the user can define, among else, the initial states, output formats, number of iterations, e.g. searching for a fixed point, time horizon, forbidden states. An informal introduction to SpaceEx and its modeling paradigm is available at [13].

2.3 Simulink to SpaceEx Translator (SL2SX)

SL2SX [17] is a semi-automated tool that undertakes the translation of Simulink models into SpaceEx models. The translator accepts a Simulink model that is saved in XML format and generates as an output a network of hybrid automata in SX format. The translation preserves most of the structural aspects of the Simulink diagram, such as the names, hierarchy and graphical positions. Blocks that are not supported are translated into “empty” components, which must then be completed by the user.

2.4 Compositional Syntactic Hybridization

Hybridization is an abstraction method to obtain PWA over-approximations of nonlinear dynamics [5]. It consists of partitioning the state-space into a set of domains, and for each domain, approximating the nonlinear dynamics by simpler ones plus nondeterministic inputs to account for the abstraction error. Existing hybridization methods operate on the composed (flattened) system. Compositional syntactic hybridization, a recently proposed method [15], performs the PWA over-approximation in a compositional manner and takes advantage of the on-the-fly composition of hybrid systems that is supported by the SpaceEx platform. Three main steps are

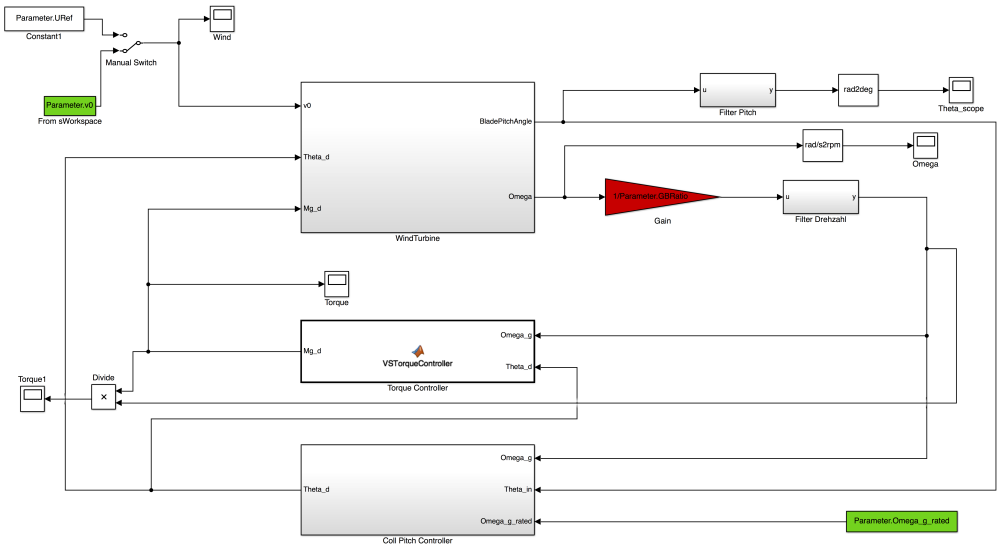


Figure 1: Wind turbine - Simulink model - Top Level [18].

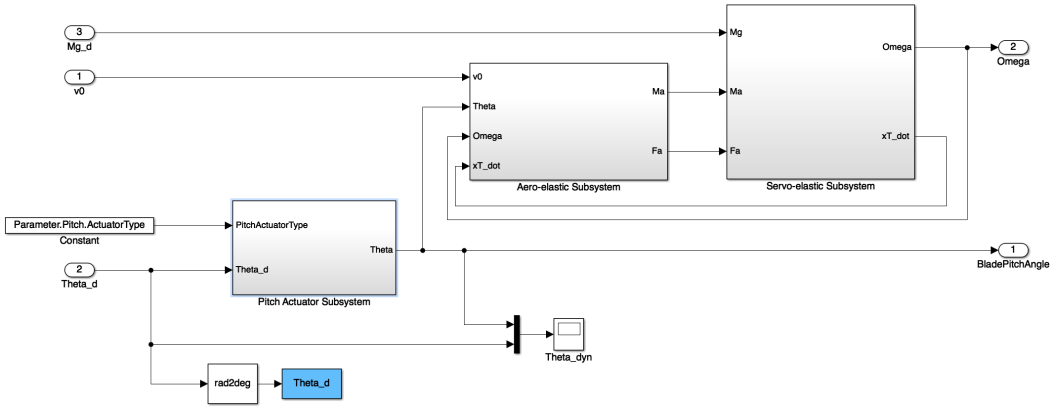


Figure 2: Wind turbine - Simulink model - Plant [18].

involved: *syntactic decomposition*, replacing the nonlinear ODE by a linear ODE with nonlinear algebraic equations; *hybridization*, constructing a PWA approximation for each algebraic equation and providing a sound over-approximation of the original system by adding an error term; and finally *composition* of the hybrid automata, and elimination of the algebraic equations.

3 Benchmark Model

The wind turbine modeling is done with MATLAB and Simulink [18]. Figure 1 shows the top-level model of the Simulink block diagram and Figure 2 depicts the physical part (plant) of the wind turbine. The wind turbine dynamics are highly nonlinear functions of the operating point defined by the rotor speed, wind speed and blade pitch angle.

The plant consists of three subsystems: a servo-elastic, an aeroelastic and a pitch-actuator. The servo-elastic subsystem describes the tower fore-aft dynamics and the rotor dynamics. It is interconnected with the aeroelastic subsystem through signals that correspond to the aerodynamic torque and thrust. While the servo-elastic subsystem consists of linear operators, the aeroelastic subsystem has several nonlinear blocks. There are two *Square* functions, two *Products*, two *Divisions* and two fourth-order *Polynomials*. The polynomials correspond to the aerodynamic torque coefficient (cP) and thrust coefficient (cT), respectively. They are computed through a regression model. The pitch actuator dynamics can be expressed by a second-order lag, a first-order lag or a time-delay. The pitch actuator subsystem has two input signals, the demanded pitch angle and the pitch actuator type and outputs the actual pitch angle to the aeroelastic subsystem. Apart from the linear operators, it contains a *Multiport Switch*, three *Compare to constant* blocks, three *Enable* blocks and three *Enabled Subsystems*.

As for the control part, there is a generator-torque controller and a collective blade-pitch controller. The blade pitch controller is a gain-scheduled PID whose objective is to minimize the speed error between the filtered generator speed and the rated generator speed. It incorporates an anti-windup scheme to prevent integration wind-up when the actuator is saturated. The nonlinear blocks correspond to a *Division* and a *Product*. The generator-torque controller aims to maximize the extracted maximum power from the wind by tracking the optimal tip-speed ratio λ_{opt}. It is a hybrid controller with 5 locations and 2 input signals, namely the filtered generator speed and the pitch angle. It is written as an *Embedded MATLAB Function* and it includes two nonlinear operators, a *Square* and a *Division*.

Note that the benchmark model is designed from an industrial viewpoint and its verification task is considered to be of high difficulty. That ensues from the existence of several nonlinearities (over 10 distinct ones) and blocks that cannot be expressed by linear or hybrid dynamics. As far as the model dimension is concerned, the controllers introduce two state variables and the plant introduces 3 - 5 states (depending on the pitch actuation). As such, the total number of state variables of the Simulink (closed-loop) model varies from 5 to 7. This number is further increased, if we consider the rate limiters and memory blocks. The SpaceEx model adds an extra variable to capture the time evolution. Another point is that the Simulink model is more detailed than the mathematical model described in the benchmark paper [18] (e.g. rate limiters). There are two minor points regarding the benchmark model that are worth mentioning; some MATLAB paths need to be modified in order to be platform/user independent and there is a call to a toolbox (`WAFO`) that is not included in the benchmark files.

4 Model Transformation

This section presents the model transformation process, covering the model adaptation, translation, hybridization and validation steps.

4.1 Model Adaptation

To apply formal verification techniques, it is necessary to obtain a verification model from the original simulation model. Such a transformation should consider the inherent differences between these two model classes. Typically, a simulation model includes details that should be obscured from the verification model. A verification model could also be enriched with nondeterminism so as to check the behavior of the system under uncertain or varying parameters, disturbances or user inputs.

An illustrative example is the Simulink block corresponding to the wind speed profile. The wind profile is an input to the model that is read from a data structure in the MATLAB workspace. In principle, such a profile could be translated into a hybrid automaton, but this would not be very efficient in the analysis. As a matter of fact, there are 33 possible profiles of user-defined wind speed signals in the MATLAB `aeromaps.mat` file. However, even if all these cases are tested, there is still no guarantee that the closed-loop system is going to operate correctly. Actually, there might be a different wind profile that could yield undesirable behaviors. On the contrary, under the SX format, this matter could be simply resolved by adding a non-deterministic input signal that covers the minimum and maximum wind speeds. It is possible to impose bounds on the rate of change of the wind speed.

It is also necessary to preprocess and further modify the Simulink model. In fact, all the *Scopes*, *Mux*, *Demux*, *Enabled Subsystems*, *Manual Switches*, *Save to workspace* have to be deleted/replaced as their corresponding actions are defined in a different way with SX format and SpaceEx[1]. Another modification concerns the three available pitch actuators that exist in the Simulink model. As the pitch actuator dynamics cannot be neglected for large wind turbines, we consider the most critical and representative case, i.e. the second-order lag.

4.2 Translation

The second step of the model transformation is to translate the Simulink model into an equivalent SpaceEx model; equivalent in the sense that the model remains the same while the syntax changes. For this task, we employ the SL2SX translator. Having the mechanical aspects of the model translation carried out by a tool significantly reduces errors. The translation takes a few seconds and produces an SX file with 18 Network and 89 Base Components. However, not all Simulink blocks can be exactly translated, as they cannot be expressed as hybrid automata. The Simulink blocks that are automatically translated into base components are the following: *Add, Subtract, Divide, Multiply, Constant, Gain, Saturation, Integrator, Subsystem, Inport, Outport*.

Finally, it is possible to make the SX model more compact by reusing base components and deleting the ones that are duplicate or serve the same role. With this configuration, the total number of base components can be reduced approximately up to 30 distinct ones[2]. Figure 3 and Figure 4, respectively, present the top level block (network component) and the plant subsystem of the wind turbine model in SX format, as shown in the Model Editor of SpaceEx.

Implementation Issues After modifying the Simulink blocks, we have to perform further preprocessing in order to be able to use the SL2SX translator. Special attention should be given to the names and the connections. As of version 1.0.1 of SL2SX, the block names should not extend to multiple lines, as the respective blocks will not be parsed by the translator. There may occur errors with models that are saved in old Simulink versions. Also, after constructing the SX Model with the SL2SX tool, there is a need for post-processing. In particular, the translator produces the expressions "; &*gt*" and "; &*lt*" instead of the correct inequality symbols "<" and ">", respectively[3].

[1] Currently, the preprocessing is done manually. However, in principle, this step could be automated. Deleting unnecessary blocks and checking naming issues should be a straightforward task.

[2] This process is currently manual but it could be automated; either as a functionality of SL2SX or as a post-processing step.

[3] This a known issue with the way that SAX parser (which is called by SL2SX) handles special characters.

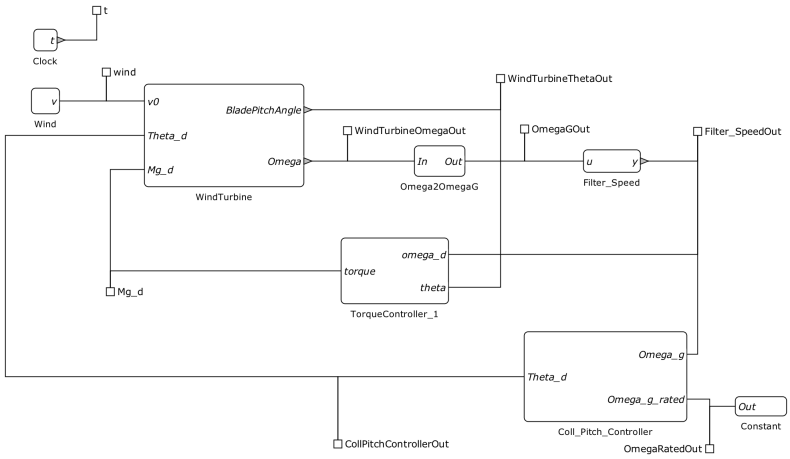


Figure 3: Wind turbine - SpaceEx model - Top Level.

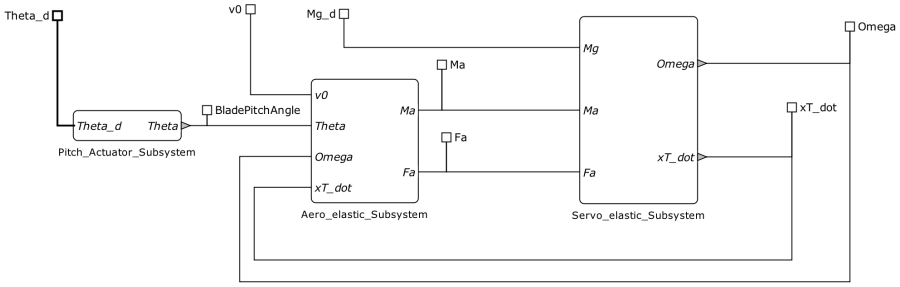


Figure 4: Wind turbine - SpaceEx model - Plant.

4.3 Hybridization

The third step is to generate PWA approximations and describe them in SX format for all these Simulink blocks that cannot be handled automatically, either because no exact translation is available (e.g. nonlinearities) or because translation cannot be applied (e.g. Embedded MATLAB Function).

To obtain PWA approximations of nonlinear dynamics, we use an abstraction method we call syntactic hybridization. This method essentially partitions the state-space into a set of domains, and for each domain, it approximates the nonlinear dynamics by simpler ones with added nondeterministic inputs to account for the abstraction error. However, instead of operating on the composed (flattened) system, we decompose the original dynamics and carry out the state-space partitioning and PWA approximation on the components. In this way, we can avoid having intractably large models and the explosion in the number of partitions can be mitigated. In essence, we break down the nonlinear blocks into components that have a small number of input variables and interconnect them together.

A practical intermediate step is to get bounds on the behavior (min, max) of the input signals of the nonlinear Simulink blocks. The shorter the ranges of the signals, the smaller the number of locations that is required by the PWA abstraction, given a desired error bound. There are different ways to estimate them. In this paper, we run Simulink simulations for

different scenarios and initial conditions to obtain a (not necessarily conservative) estimation of the signal range. Note that the signal range serves only as an indication for the hybridization step. The approximation is equipped with out-of-range (error) states. So, if the range is found to be insufficient during reachability computations, it is revised (enlarged).

For the wind turbine benchmark, the constructed base components have only one or two input signals and one output signal, the state space is partitioned into a set of domains, described by hyper-rectangles, the approximations are linearized (first-order Taylor) around the center of each domain, and the abstraction (linearization) error is computed by the evaluation of the Lagrange remainder. The maximum error is computed for each location (box). We use a combination of interval arithmetic and global optimization to bound it [2]. All the computations (linearization domains, quantization parameters, operating points, PWA approximations, errors) are conducted with MATLAB. We also have a script to facilitate the construction of PWA approximations and transform them into equivalent base components models, described in SX format. Then, we integrate all these components into a single SX file.

Note that the approximations can be conservative or non-conservative. The error is computed for each location of the base components and is added as a non-deterministic input in the corresponding invariants. However, it is possible to disregard the errors either for debugging purposes or for the purpose of obtaining a deterministic model. Table 1 presents all the Simulink blocks that need to be approximated along with the approximation method, the maximum induced errors and the number of locations of each approximation. Let us underscore that the error corresponds to the maximum error that appears in one of the locations of the PWA approximation. The individual errors in the other locations can be significantly smaller.

The resulting model has only 72 locations in all components combined. To carry out reachability analysis, SpaceEx performs on-the-fly composition of the hybrid automata and instantiates only the reachable parts of the state-space. The upper bound of the number of locations of the composed model is 16 millions.

4.4 Model Validation

The fourth step of the model transformation concerns the model validation. We propose an empirical method to evaluate whether the generated base components are correct, insofar as they yield satisfactory behaviors and do not introduce any deadlocks. In this step, we are essentially testing the implementation. In this vein, we have created an SX signal library with predefined components that describe *trigonometric (sine, cosine)*, *step* and *ramp* functions. We have added a monitor to visualize the output[4]. Once we create a new approximation, we can integrate it along with the SX library and obtain a *tester module* (consisting of base and network components).

Our objective is twofold: on the one hand, to check that the approximation is non-blocking and on the other hand to check that it is indeed an over-approximation. For the first objective, the input signals are selected in a way that compels the PWA automaton (introduced by the hybridization process) to necessarily visit all of its locations. As soon as the input signals exceed the allowed operating range, the automaton goes out-of-bounds and the analysis terminates. That means that a fixed point has been reached. For the second objective, we perform reachability analysis on the tester module (containing the approximation and the input signals) and then compare the results with random simulations. In this way, we check whether the simulations are included in the reachable sets.

[4] Note that STC and LGG algorithms in SpaceEx do not output algebraic variables, so visualizing is possible by mapping the algebraic variable to the solution of an ordinary differential equation.

No	Block (Name)	Approx.	# Loc.	Error Bounds	Remarks
1	Product *(Anti windup)*	syntactic	4	2	$x \cdot y$
2	Division (C_p/λ)	syntactic	10	0.027613	(x/y)
3	Division (*GS factor*)	syntactic	2	0.020111	$\frac{1}{1+x}$
4	Division (λ)	syntactic	4	0.6983	x/y
5	Product ($C_T \cdot v_{ref}^2$)	syntactic	4	25.5781	$x^2 \cdot y$
6	Product ($C_p/\lambda \cdot v_{ref}^2$)	syntactic	4	3.23437	$x^2 \cdot y$
7	Polynomial (*aeromaps*)	syntactic	6	3.39472	4th-order
8	Polynomial (*aeromaps*)	syntactic	6	12.2809	4th-order
9	Embedded MATLAB	syntactic	28	10.48584	x^2 and $1/x$
10	Saturation	substituted	3	0	exact[a]
11	Read from workspace	substituted	1	–	range[b]
12	Mux, Scope	unnecessary	–	–	–
13	Save to workspace	unnecessary	–	–	–
14	Enabled Subsystem	unnecessary	–	–	–
15	Multiport Switch	unnecessary	–	–	–
16	Compare to Constant	unnecessary	–	–	–
17	Manual Switch	unnecessary	–	–	–
18	Rate Limiter	ignored[c]	–	–	DAE
19	Memory	ignored[c]	–	–	continuous delay

Table 1: SpaceEx base components - Approximations

[a] for visualization reasons an extra location should be added.
[b] replaced by a nondeterministic input.
[c] to mitigate state explosion, this block has not been considered.

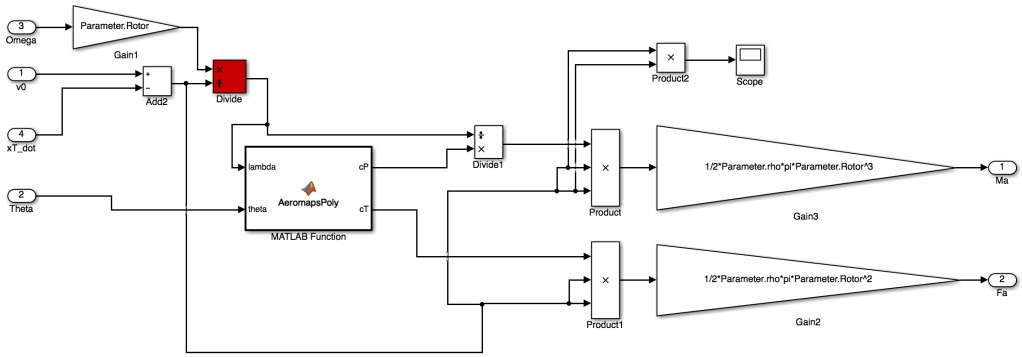


Figure 5: Aero-elastic subsystem - Simulink - Nonlinear *Divide* block shown in red.

As an illustration, let us consider a Simulink block of the aero-elastic subsystem of the wind turbine (plant). Figure 5 shows the entire aero-elastic subsystem. The block that we focus on is the *Divide* block and it is highlighted in red.

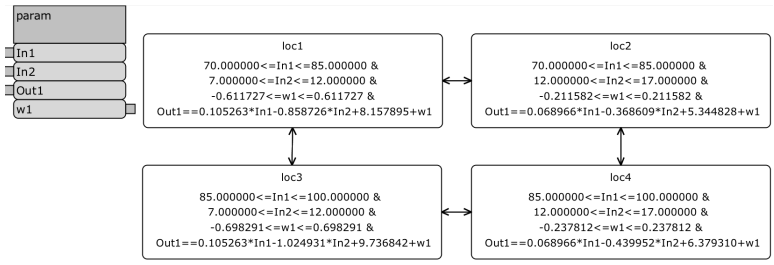
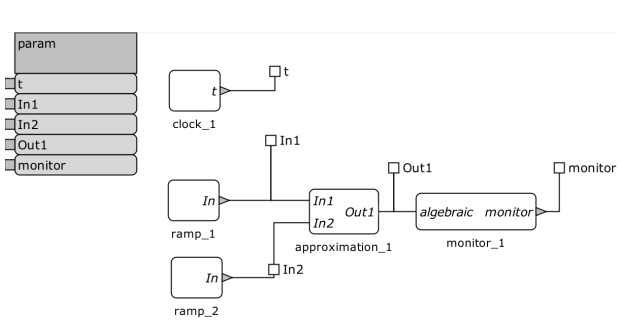
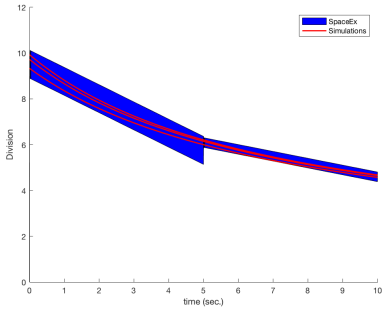


Figure 6: *Divide* block - PWA approximation in SpaceEx.



(a) Tester - Division λ.

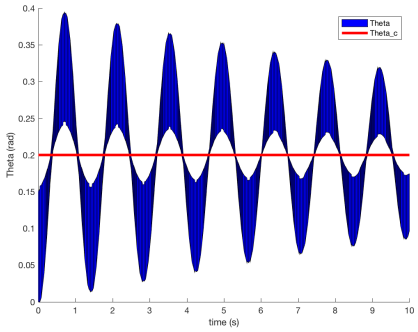


(b) Reachable sets and Simulations.

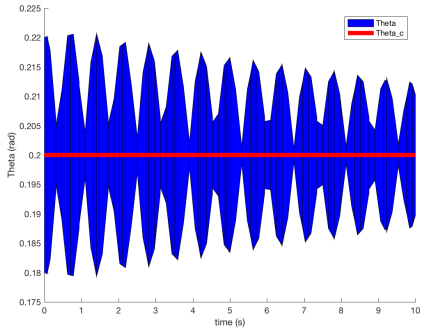
Figure 7: SpaceEx Base Component - Division λ - Model Validation. The reachable sets of the approximate SpaceEx block are indicated with blue, the simulation runs of the nonlinear function are denoted in red. The simulation runs are contained in the reachable sets.

This block corresponds to the division operator and computes the division between two input signals over time. From a physical point of view, it computes the tip-speed ratio, which is a dimensionless variable denoted by *lambda* (λ). This block, being a nonlinear one, is replaced by a PWA over-approximation with 4 locations. Figure 6 shows the approximation as a SpaceEx base component. The variables *In1* and *In2* relate to the inputs, *Out1* is the PWA approximation of the division operator *In1/In2* and corresponds to λ. The approximation error $w1$ is added in the form of a non-deterministic disturbance. Note that all the variables should be uncontrollable, and the error term should be included in the invariant, declared as a local variable; see Section 2.2 for more information about SpaceEx modeling language.

After incorporating the base component of the *Divide* block with our input library, we want to check the previously mentioned objectives (no deadlocks and over-approximation). For this reason, we consider that the inputs are ramp signals with varying initial conditions. The first input is described by $\frac{\mathrm{d}In1}{\mathrm{d}t} = 1$ with initial conditions in $69 < In1 < 71$ and the second input by $\frac{\mathrm{d}In2}{\mathrm{d}t} = 1$ with initial conditions in $6.9 < In2 < 7.1$. It is necessary to add a monitor in order to visualize the output, as we intend to use the STC algorithm. Then, we conduct reachability analysis, compare the results with random simulations of the original nonlinear function and observe that for the considered scenario SpaceEx yields an over-approximation. The tester module is shown in Figure 7a, the reachable sets and the simulation runs are shown in 7b.

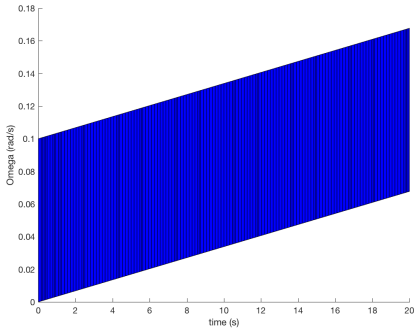


(a) Initial Conditions - $0 \leq \theta \leq 0.15$.

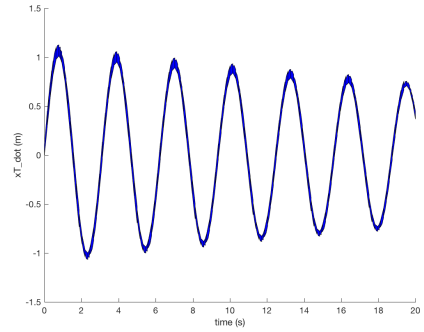


(b) Initial Conditions - $0.18 \leq \theta \leq 0.22$.

Figure 8: Pitch actuator - SpaceEx Network Component - Reachable Sets (blue: Output, red: Input)



(a) Output - Omega



(b) Output - Displacement (xT_dot).

Figure 9: Servo-elastic - SpaceEx Network Component - Reachable Sets

The same approach (in a hierarchical way) is utilized in order to check that compositions of multiple base components, network components or larger subsystems operate correctly.

5 Preliminary Results

In this section, we present validation runs for some of the subsystems (network components). We start with the **pitch-actuator** subsystem of the wind turbine plant. The input of this block is the commanded pitch angle and the output is the actual pitch angle. Considering the STC scenario, a global time horizon of 10s, and a constant input signal, we get a fixed point. Figure 8 depicts the reachable sets for different initial conditions. The input signals are shown in red and the output signals in blue.

Next, we consider the **servo-elastic** subsystem of the wind turbine plant. Given constant inputs, a global time horizon of 20s and the STC scenario, SpaceEx finds a fixed point. Figure 9 depicts the reachable sets for initial conditions: $0 \leq$ Omega ≤ 0.1 and $0 \leq$ xT_dot ≤ 0.1.

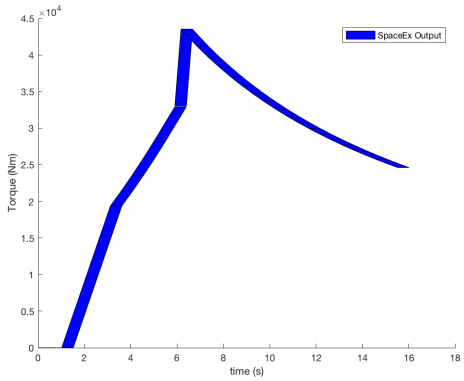
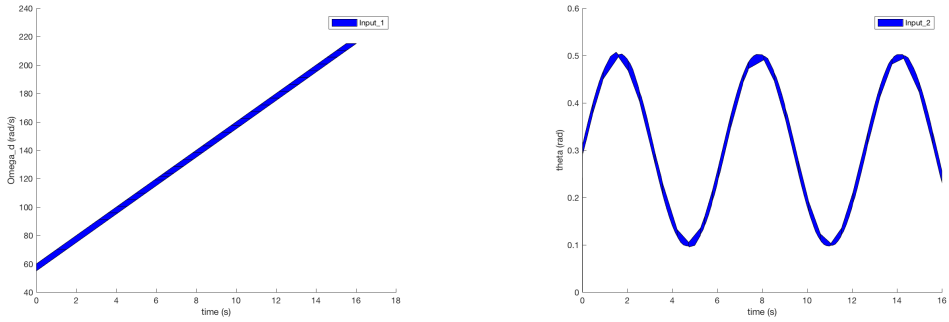


Figure 10: Torque Controller - SpaceEx Network Component - Output.



(a) Input 1 - Omega_d. (b) Input 2 - Theta.

Figure 11: Torque Controller - SpaceEx Network Component - Inputs.

Finally, we analyze the **torque controller**. The approximation consists of 28 locations, has two input signals (omega_d, theta) and one output (torque). We consider the input signals of Figure 11 (a ramp and a sine function), as they cover, when combined, the entire 2D operating range (from minimum to maximum allowed values). Indeed, with this configuration, SpaceEx visits all the controller locations and finds a fixed point after 16 seconds. The reachable sets are computed with the PHAVer scenario. The reachable sets of the torque controller are displayed in Figure 10.

6 Conclusions

Model transformation plays an important role in bridging the gap between industrially relevant models and verification tools. This work aims to assist the application of hybrid system reachability tools to industrial-sized models described by MATLAB/Simulink. The existence of a translator from Simulink to SpaceEx already facilitates the use of Simulink models within SpaceEx environment. However, the large variety of Simulink blocks and their diverse features

pose difficulties in mapping Simulink subsystems to SpaceEx components. There are several Simulink blocks that cannot be described by hybrid automata with PWA dynamics and therefore cannot be processed by SpaceEx. In practice, most of these blocks correspond to nonlinear functions and operations.

The wind turbine benchmark constitutes a relevant example, insofar as it represents an industrial case study, while including many nonlinear blocks. Through syntactic hybridization, it is possible to over-approximate the dynamical behavior of these blocks and seamlessly integrate them within SpaceEx components. Note that the resulting model of hybrid automata is expressed in the general SX format. As such, it can be fed directly into SpaceEx platform, or translated into formats that comply with other verification tools using the HyST [6] tool.

Our future work is targeted towards verifying the requirements of the wind turbine benchmark. As the proposed approximation contains around 16 million locations, there is a shortcoming concerning the computational costs and execution time. Currently, SpaceEx identifies the initial locations of the approximation through enumeration, which does not scale. A potential way to tackle this issue is to use compositional reasoning to identify the initial locations and instantiate as few locations as possible during the analysis.

Acknowledgments

The authors would like to thank Simone Schuler for valuable discussions on wind turbine models.

References

[1] MATLAB 9.0 and Simulink 8.7. The MathWorks, Inc., Natick, Massachusetts, United States.

[2] Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4042–4048. IEEE, 2008.

[3] Rajeev Alur, Aditya Kanade, S Ramesh, and KC Shashidhar. Symbolic analysis for improving simulation coverage of Simulink/Stateflow models. In *Proceedings of the 8th ACM international conference on Embedded software*, pages 89–98. ACM, 2008.

[4] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[5] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[6] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, pages 128-133. ACM, 2015.

[7] Olivier Bouissou, Samuel Mimram, and Alexandre Chapoutot. HySon: Set-based simulation of hybrid systems. In *2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP)*, pages 79–85. IEEE, 2012.

[8] Scott Cotton, Goran Frehse, and Olivier Lebeltel. The SpaceEx modeling language, 2010.

[9] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[10] Alexandre Donzé and Goran Frehse. Modular, hierarchical models of control systems in SpaceEx. In *Control Conference (ECC), 2013 European*, pages 4244–4251. IEEE, 2013.

[11] Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *CAV'16*, pages 531–538, 2016.

[12] Predrag Filipovikj, Nesredin Mahmud, Raluca Marinescu, Cristina Seceleanu, Oscar Ljungkrantz, and Henrik Lönn. Simulink to UPPAAL statistical model checker: Analyzing automotive industrial systems. In *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*, pages 748–756. Springer, 2016.

[13] Goran Frehse. Introduction to SpaceEx v0.8. http://spaceex.imag.fr/sites/default/files/introduction_to_spaceex_0.pdf.

[14] Goran Frehse, Colas Le Guernic, Scott Cotton Alexandre Donzé, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In CAV'11, pages 379-395, 2011.

[15] Nikolaos Kekatos, Marcelo Forets, and Goran Frehse. Constructing verification models of nonlinear Simulink systems via syntactic hybridization. Submitted, https://hal.archives-ouvertes.fr/hal-01487658, 2017.

[16] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[17] Stefano Minopoli and Goran Frehse. SL2SX translator: From Simulink to SpaceEx models. In Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, pages 93-98. ACM, 2016.

[18] Simone Schuler, Fabiano Daher Adegas, and Adolfo Anta. Hybrid modelling of a wind turbine (benchmark proposal). Applied Verification for Continuous and Hybrid Systems (ARCH), 2016.

[19] N. Zhan, S. Wang, and H. Zhao. *Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach*. Springer International Publishing, 2016.