SimEx: The Art of Systematic Exploration

Amy Liffey[1], Davide Calvaresi[1], Lora Fanda[2] and René Schumann[1,*]

[1]HES-SO Valais-Wallis, University of Applied Sciences Western Switzerland
[2]University of Geneva, Switzerland

Abstract

Experimentation is essential in most research domains, serving as a good way to explore, test, and validate a system's behaviors. However, real-world experiments typically involve physical testing, observation, or interaction with the systems, which can be dangerous, unethical, very costly, or even impossible in some cases. Therefore, in-silicon experimentation (known as computer simulations) becomes beneficial and, in most cases, cost-effective, allowing the simulation of even impossible scenarios. However, the exploration of the whole environment and all possible scenarios often becomes computationally costly and trade-offs between accuracy and computational run time has to be made. This paper proposes a systematic exploration tool that approximates the system's behavior with the aim of significantly improving the efficiency of simulation runs in simulation studies. This tool holds the potential to revolutionize the way we approach simulation studies, making them more efficient and productive. The approximation is done using partial functions, which have been approximated based on obtained data points, i.e. results of simulation runs. We have presented an approach that allows for obtaining sufficiently good *enough* functional approximations while keeping the overall number of simulation runs as low as possible.

Keywords

Systematic exploration, Structured simulations, Function approximation

1. Introduction

Experimentation is a cornerstone in most research domains. It serves as a robust approach to exploring, testing, and validating environments, systems, and overall dynamics. However, real-world experiments typically involve physical testing, observation, and interaction, which can be impractical, dangerous, unethical, prohibitively expensive, or even impossible in some cases. Examples include meteorological [1], nuclear [2], pandemic modeling [3], drugs development [4], energy production and negotiation [5], and disaster scenarios [6]. Therefore, in-silicon experimentation (known as computer simulations) becomes beneficial and, in most cases, cost-effective. These simulations provide valuable insights into the simulated system within a controlled environment, allowing for the exploration of different parameters, combinations, search spaces, etc. Nonetheless, (i) simulation outcomes are highly dependent on the accuracy of the models [7], (ii) achieving *the best possible accuracy* and overall outcome quality can still entail humongous waiting time and costs. Thus, it is often necessary to make *trade-offs* between the computational time cost and what is considered *sufficiently good*. For example, in environmental studies, climate change scenarios can be simulated to understand the impacts of changing environmental conditions on ecosystems [8]. When building complex machines such as planes or cars, simulations are used to create initial ideas about aerodynamics and physical properties, leading to reduced costs and risks even before the manufacturing process starts. In intelligent transport systems, dynamic traffic management is studied using simulations to optimize traffic flow (e.g., on motorways [9]). These are just a few examples of use cases requiring numerous simulation runs. The identification of the needed *trade-off*, the meaning of *sufficient*, and what is *good* are, of course, highly context-specific. This also implies that an approach aiming at controlling the process of approximating the system's behavior has

PuK 24: Gemeinsamer Workshop der Fachgruppen PuK und Wissensmanagement, September 24, 2024, Würzburg, Germany
*Corresponding author.
amy.liffey@hevs.ch (A. Liffey); davide.calvaresi@hevs.ch (D. Calvaresi); rene.schumann@hevs.ch (R. Schumann)
https://www.hevs.ch/de/mini-sites/projekte-produkte/si-lab/ (R. Schumann)
0009-0003-4057-7816 (A. Liffey); 0000-0001-9816-7439 (D. Calvaresi); 0000-0003-3432-3353 (L. Fanda); 0000-0002-9039-5741 (R. Schumann)
© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

to work on the assumption of the simulation as a black box system. However, common to all types of simulation systems is the desire to perform them efficiently, which is addressed within the paper. To the best of our knowledge, the modelization of the needed procedures/components to reduce the computational burden is still unmet.

Therefore, we have developed a configurable framework named SimEx (*Sim*ulation *Ex*plorative), enabling researchers to steer the space of investigation through systematic explorations, simulations, and validations. This paper presents its underlying vision, rationale, and architecture and refers to its earliest adoption to test and evaluate the performance of a traffic controller [10].

The remainder of the paper is structured as follows. Section 2 describes the state of the art in search-based systematic exploration. Section 3 presents the methodology and general idea behind SimEx. Section 4 provides an overview of the architecture and implementation. Section 5 presents the use case on which we tested our tool. Finally, Section 6 discusses the findings, and Section 7 concludes the paper and outlines future work.Experimentation is a cornerstone in most research domains. It serves as a robust approach to exploring, testing, and validating environments, systems, and overall dynamics. However, real-world experiments typically involve physical testing, observation, and interaction, which can be impractical, dangerous, unethical, prohibitively expensive, or even impossible in some cases. Examples include meteorological [1], nuclear [2], pandemic modeling [3], drugs development [4], energy production and negotiation [5], and disaster scenarios [6]. Therefore, in-silicon experimentation (known as computer simulations) becomes beneficial and, in most cases, cost-effective. These simulations provide valuable insights into the simulated system within a controlled environment, allowing for the exploration of different parameters, combinations, search spaces, etc. Nonetheless, (i) simulation outcomes are highly dependent on the accuracy of the models [7], (ii) achieving *the best possible accuracy* and overall outcome quality can still entail humongous waiting time and costs. Thus, it is often necessary to make *trade-offs* between the computational time cost and what is considered *sufficiently good*. For example, in environmental studies, climate change scenarios can be simulated to understand the impacts of changing environmental conditions on ecosystems [8]. When building complex machines such as planes or cars, simulations are used to create initial ideas about aerodynamics and physical properties, leading to reduced costs and risks even before the manufacturing process starts. In intelligent transport systems, dynamic traffic management is studied using simulations to optimize traffic flow (e.g., on motorways [9]). These are just a few examples of use cases requiring numerous simulation runs. The identification of the needed *trade-off*, the meaning of *sufficient*, and what is *good* are, of course, highly context-specific. This also implies that an approach aiming at controlling the process of approximating the system's behavior has to work on the assumption of the simulation as a black box system. However, common to all types of simulation systems is the desire to perform them efficiently, which is addressed within the paper. To the best of our knowledge, the modelization of the needed procedures/components to reduce the computational burden is still unmet.

Therefore, we have developed a configurable framework named SimEx (*Sim*ulation *Ex*plorative), enabling researchers to steer the space of investigation through systematic explorations, simulations, and validations. This paper presents its underlying vision, rationale, and architecture and refers to its earliest adoption to test and evaluate the performance of a traffic controller [10].

The remainder of the paper is structured as follows. Section 2 describes the state of the art in search-based systematic exploration. Section 3 presents the methodology and general idea behind SimEx. Section 4 provides an overview of the architecture and implementation. Section 5 presents the use case on which we tested our tool. Finally, Section 6 discusses the findings, and Section 7 concludes the paper and outlines future work.

2. State of the Art

To the best of our knowledge, in the search-based systematic exploration, the Monte-Carlo-like simulation techniques have been commonly used for data point generation aiming to explore the behavior of the systems. However, due to the complexity and dynamic nature of the underlying processes, it

is hard and very costly to examine all the possible data point input-output pairs. These constraints lead to the investigation of alternative methods for these systems. As an example in the paper [11], the entropy-driven Monte Carlo simulation method has been proposed for approximating the survival signature of complex infrastructures. In this case, the entropy steers the sampling of Monte-Carlo simulation in the direction of the complex configurations in order to save computational costs. This is similar to our idea of systematic exploration, which proposes an alternative approach to fighting the computational costs of the simulation runs. The aim is to gather insights into the overall system behavior while minimizing the total number of data points, in this case, simulation runs. Thus, creating the meta-heuristic approach similar to [12], where the systematic search of space is performed to achieve the given objective.

In essence, this meta-heuristic approach is trying to find a sufficient surrogate model of the system, similar to [13], where the system behavior is explored with a Random forest meta-model and sequential sampling. However, these types of Machine Learning methods usually require extensive retraining of the model. Our approach aims to provide something effective and rather *simple*. Hence, the surrogate model of SimEx only consists of a function approximator attempting to fit the curve through the output points and estimating the overall behavior within the given domain.

3. Methodology: Searching for the missing

The general idea behind the SimEx is to provide a customizable systematic exploration tool as a part of the Structured simulation framework (SSF). The Structured simulation framework presented in [14] tries to tackle systematic testing of the complex systems by providing structured ways to explore the state space or output space. The SimEx concentrates on the exploration of the output space using a systematic search algorithm. This involves an iterative process of initial simulations, data collection, and focused exploration. The focused exploration technique identifies the problematic regions in the previous iteration run e.g. regions not meeting the criteria of sufficiency in terms of their approximation. Consequently, these regions require a more thorough investigation with a finer search granularity, which also leads to a larger number of simulation runs. This avoids unnecessary simulation and aims to approximate the system's behavior with minimal computational efforts, which are in large solely depending on the number of simulation runs.

The tool is designed for general-purpose studies and execution of distinct scenarios with fully customizable parameters and the possibility of third-party module integration. A general overview can be seen in Figure 1. The pipeline starts with the configuration and definition of the input data for the experiment, which is passed to the control loop. The control loop takes care of the operation of the whole application. It consists of the point generation, simulation, and validation components, which are steered by the main cycle loop based on the given configuration.

The `point generation component` is responsible for input data point generation for the simulation model based on the defined criteria. Thus, it will define for which points in the input space of the simulation a simulation run will be executed. In essence, it resamples and modifies the input domain, providing the input data points to the simulation component.

The `simulation component` takes the input data points and returns input and output pairs. Thus, each input has to initiate a corresponding simulation run and extract the required outputs of the run. Within the `validation component`, the obtained outputs are evaluated in terms of a resulting approximation based on all obtained outputs, which is of sufficient approximation quality. The default validation is based on the threshold method describing how far a point can be from the baseline approximation to be considered *good enough*. These threshold parameters are configured in the configuration stage at the start of the tool. These parameters have to be evaluated and chosen carefully. When defining the threshold for the given system, the experiment designers must consider the trade-off between accuracy and computational cost.

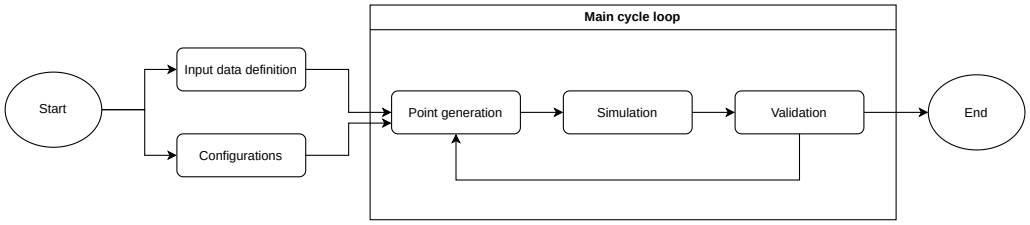


Figure 1: SimEx control loop

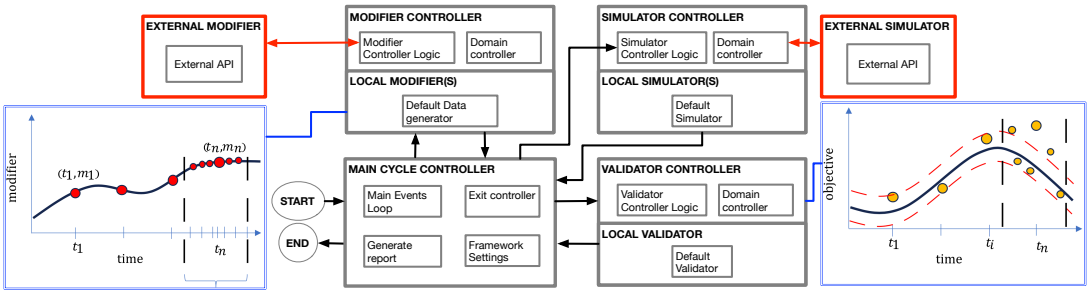


Figure 2: General SimEx scheme for modifying the granularity of the input space

4. Architecture and Implementation

The overall architecture of SimEx consists of four main blocks called `Modifier controller` in the following, also referred to as `Point generator controller`, `Simulator controller`, `Validator controller` and `Main cycle controller`,. The control flow and the interaction of the components are illustrated in Figure 2. Each of these components can be configured and adjusted for the specific use cases.

The purpose of the `Point generator controller` is to generate the input data points for the simulation using the predefined configurable functions or plugging an external one. The user defines the input domain with `domain_min_interval` and `domain_max_interval` values in the configurations, and these values are used for the point generation. In the first iteration, the whole input domain is sampled by the `Modifier function` with the configured initial step size `modifier_data_point`. This data point value is adjusted during the run till it reaches the minimal configured value by the user. The `Modifier function` in our case simple quadratic function as shown in Equation 1 is applied on each input value, and these modified values are then resampled within the given input domain defined by maximal and minimal interval value. These generated data points are then passed to the next stage.

$$modified\_value = input^2 \tag{1}$$

The `Simulator controller` main goal is to input the generated data points into the simulation model and run the resulting simulation to measure the reaction of the system for which we want to approximate the performance. Thus, it must not only modify the relevant files for the actual simulation model but also execute the simulator and extract the needed performance information from the simulator's outputs. As this part clearly depends on the underlying simulator, we will not give a particular insight into the implementation of the component. In essence, the `Simulator controller` can execute pre-implemented simulator functions or external components configured in the configurations to perform these actions. We provide a common wrapper implemented in the simulator controller. This wrapper takes the previously generated points as input and feeds them one by one to the defined custom function, which transforms them into the simulated y output as in 2. The obtained couples of input space value (x-*value* and the resulting y-*values* from the output space, obtained from executing the simulation, are

passed to the next validator stage.

$$y = func(modified\_value) \tag{2}$$

The `Validator controller` purpose is to approximate the performance of the system based on the obtained simulated data points from the previous stage. These points are injected into the validator controller, and the validator method attempts to fit polynomial curves along these points. This is done by a `polyfit` function from numpy library, while increasing the degree of the polynomial function until the curve is well-fitted, a maximum configured degree is reached, or no improvement is observed. This approximation is constrained by the performance parameters, which are use-case specific and should be calibrated by the domain experts for the given simulation study. The regions where the validator is not able to approximate the given interval within the constraints are marked as unfitted regions. These unfitted regions are further sent back to the `Modifier controller` to generate more data points within those intervals, and consequently, the generation-simulation-validation loop starts again. To create validator constraints estimating if the approximations are *good enough*, we use two main performance parameters for x and $y - axis$ called `y_threshold_fitting` and `x_threshold_interval`. The `y_threshold_fitting` parameter estimates how far from the approximated function a data point can be to be still considered good enough in terms of the objective function. The `x_threshold_interval` defines the percentage of how far we expand the unfitted interval on each side of the unfitted points. The improvement is observed by calculating weighted mean square error (`weighted_mse`) 3 during the polynomial fit to estimate improvement between polynomial coefficients. The penalty weight parameter is introduced in this error measurement to prevent overfitting for the higher polynomial dimensions. For each polynomial fit iteration, we then compare the `weighted mse` of the previous and new coefficients with the `improvement_threshold` parameter shown in Condition 4. The penalty weight, degree, max degree, and improvement threshold parameters can be set in the configuration file.

$$weighted\_mse = \frac{1}{N}\sum_{i=1}^{n}(y\_old_i - y\_current_i)^2 + penalty_{weight} \cdot \sum_{x=max\_degree}^{degree} x^2 \tag{3}$$

where:

- N = number of simulated output points
- y_old_i = previous simulated
- $y\_current_i$ = damage level
- $penalty\_weight$ = penalty weight for the higher coefficients
- $degree$ = minimal degree of the polynomial function
- max_degree = maximal degree of the polynomial function

$$(weighted\_mse\_old - weighted\_mse\_new) >= improvement\_threshold \tag{4}$$

Lastly, `Main cycle controller` is responsible for the overall loop from data generation to simulation and validation. These modules consist of all modules creating the final logic. As mentioned before, based on the configuration parameters, the validator identifies the unfitted regions and continues the loop until all the regions are estimated or the `modifier_data_point` reaches the `modifier_incremental_unit` value. The validation module is customizable within configuration files and can be omitted by plugging in a third-party component.

All building blocks are implemented in an object-oriented manner, allowing the user to customize or omit/exchange certain components as mentioned in Section 3. All components are implemented in Python, and the main loop can be run as a script or Jupyter Notebook. These control loops contain the default implementation to be extended by the user. The simulator can be executed concurrently in separate Processes. Users can choose the maximum number of cores for a given machine and spawn the Processes accordingly. This allows to perform the simulation runs concurrently to speed up the execution of the overall simulation study.

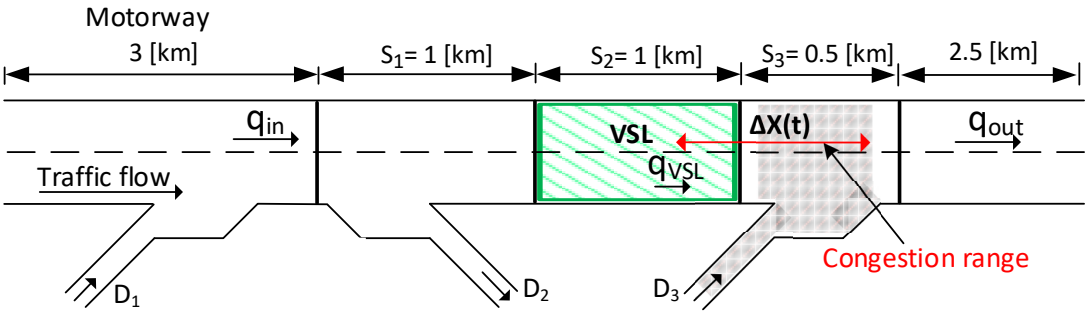


Figure 3: Application of VSL to increase throughput in congested area

5. Use Case: Evaluation of Traffic Controller Performance

As a use-case to test our tool, we choose the evaluation of a traffic controller's performance as described in [10]. Operations of traffic controllers are essential for the road infrastructure to be functional. Commonly, feedback-based controllers (e.g., variable speed limit (VSL) controllers, which require parameter tuning for specific traffic scenarios) are used to ensure better flow on the motorways. The variable speed limit zones are then created upstream of the motorways bottlenecks such as high congestion points to prevent traffic jam. Such controllers can be calibrated by running offline simulations with different types of parameters and traffic flows. These calibrations require simulation of larger traffic scenarios which are often computationally very expensive, and hence, reducing the computational burden with systematic exploration is highly desirable.

In our experiment, we use the mainstream traffic demand as an input variable for the variable speed limit controller. As a part of the simulation study two settings are compared; a no-control case, which serves as a baseline for the given scenario (referred to as the NOVSL case), and a variable speed limit controller case regulating the traffic flow (referred to as the VSL case).

For both cases, we apply the aforementioned SimEx approach by an initial sampling of the space and afterward systematically modify the input variable to provide higher granularity in more problematic, i.e., not well approximated, regions.

The simulation setup of the the simulator function was done by researchers from the University of Zagreb [10]. The simulator function consists of *Motorway model* 3 based on the previous work [15]. The speed limit is assigned for the chosen control time to the corresponding VSL zone, shown in green in Figure 3. Within the experiment the bottleneck is generated on the motorway section S_3, and effects of this performance of the control strategy is measuered for the travel time along the entire motorway segment. More details such as on-ramp traffic demands can be found in [15].

The configuration of the *Static Traffic Parameters* used in our experiment is synthetic, and we assume that the model is already calibrated. Within the SUMO simulator [16], we use the Krauss car-following model with the default parameters. The previously mentioned bottleneck in section S_3 (Figure 3) is induced due to interaction between mainstream and the traffic flow entering the motorway at the on-ramp D_3. The induced downstream congestion activates the VSL, serving as the main test for the SimEx tool of the VSL use case. In each simulation run, we cover all relevant aspects of the simulated rush hour scenario, resulting in a simulation lasting 90 minutes. More details can be found in the corresponding paper [10].

We use the *mainstream traffic demand* as input data starting from 2500 $[veh/h]$ up to 4000 $[veh/h]$ being the domain maximal and minimal values. For the first iteration, this domain gets sampled according to modifier incremental unit parameter 100 $[veh/h]$ and further continues the loop until all regions are estimated or the modifier data point reaches the minimum. These SimEx simulation setup parameters can be seen in Table 1. The objective by which we measure the system performance is called total time spent (TTS). The final approximated behavior by piece-wise functions for the motorway service with VSL and no control (NOVSL) is shown in Figure 4. The biggest benefit for the VSL is

Parameter	value	Parameter	value
domain min interval	2500	modifier incremental unit	25
domain max interval	4000	modifier data point	100
threshold y fitting	15	threshold x interval	0.80
degree	2	max degree	9
improvement threshold	0.1	penalty weight	1

Table 1
Use case: SimEx parameters setup

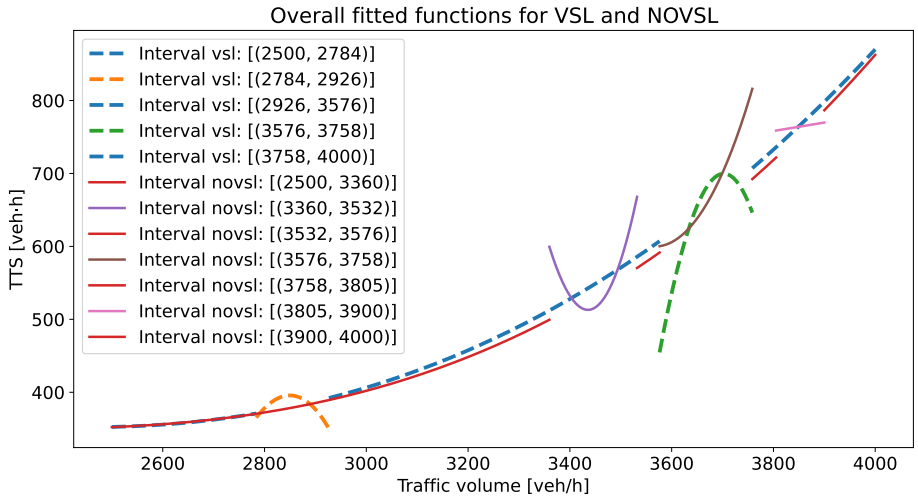


Figure 4: Final fitted functions for VSL and no control case

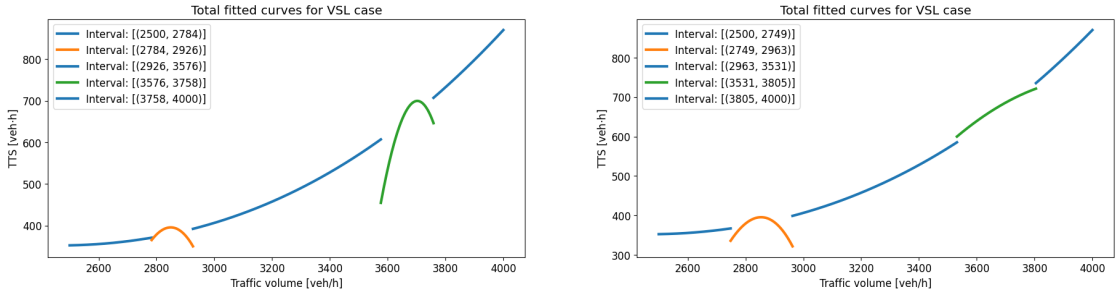
detected in the input interval $[3300, 3800]$, where it outperforms the NOVSL baseline. This behavior is expected since we set the $K_v = 4.5$ to the optimal value for the VSL controller operating near the critical point in the fundamental diagram for which traffic volume is close to its maximal rate. In other intervals, it is evident that VSL is not performing as well as NOVSL. The final fitted function Figure 4 is a crucial tool that provides detailed information about the behavior of the VSL controller. It underscores the importance of understanding the VSL's performance and the need for further evaluation in specific regions.

6. Discussion

In the previous sections, we have outlined the ideas and mechanisms as well as a use-case application of the SimEx approach. Although promising, the proposed approach is still in development and presents some limitations. This section discusses them as well as the contingency plan.

Our analysis and the case study highlight that, at the moment, the input and output spaces of the simulation (in terms of it being controlled by the SimEx approach) are one-dimensional. Thus, only one variable affects the simulated system to be approximated (input space), and the system's reaction is also considered only in one value (output space). This is not restricting in general. However, it simplifies the early stages of the SimEx development while allowing for fewer simulation runs. Indeed, more dimensions entiles a significant growth in the number of simulation runs. However, both (input and output space) can theoretically be scaled to a multi-dimensional one. Nevertheless, the approximation function may not be adequate in the long run and require more profound interventions.

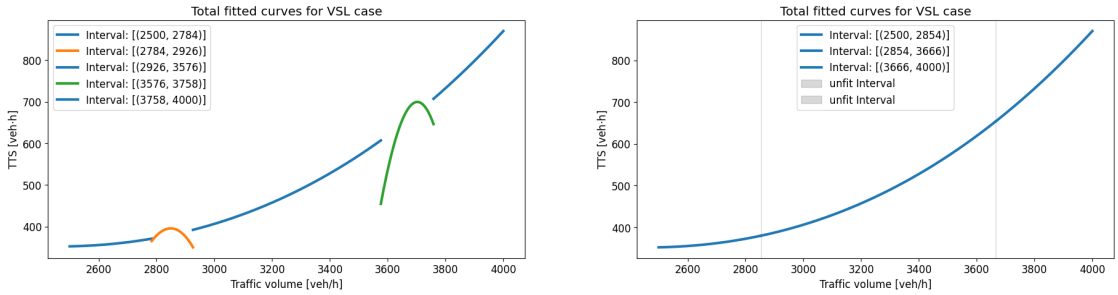
During the development, we experimented with the `threshold_x_interval` parameter tuning. The `x_threshold`, as mentioned before in Section 4, defines the percentage of how far the unfitted interval



(a) Default: X threshold = 0.8

(b) X threshold = 1.20

Figure 5: Compare default unfitted intervals to x threshold 1.20



(a) Default: X threshold = 0.8

(b) X threshold = 0

Figure 6: Compare total figure default with x threshold 0

gets expanded on each side of the unfitted points. In this use-case, a value of 0.8 was used, meaning that 80% from the interval between the points is added on each side of the unfitted point, creating the unfitted regions. In the following, we compare outcomes on the sensitivity of this parameter. Figure 6 shows that the value has been set to 0, resulting in the unfitted point becoming the unfitted interval, which is not enough to fit any curve illustrated by the final gray unfitted regions. Similarly, the second extreme scenario setting the parameter to 1.20 taking 120% on each side of the unfitted point is shown in Figure 5. The effect of the parameter can be clearly seen in the interval $(3531, 3805)$. Compared to the default setting, the approximate function is considerably smoother. Thus, as the approximations of the system's behavior shown to be sensitive towards the configuration of the aforementioned parameter, careful judgment is required in identifying correct parameterization for the approximation process in the simulation study design.

As it can be seen within the previous Figure 4 of the approximated functions, the fitted function often has *tails* in the transitions due to the differences between them; in the following, this phenomenon will be referred to as tails. This tailing is an actual limitation of the existing approach. So far, different approaches have been investigated how to improve this issue with the default configuration mentioned in Section 5.

The first approach to handle tailing is to add the bordering points, i.e., endpoints of the corresponding interval in the list of input data, for which appropriate *y-values* will be created. The idea behind this is to achieve a smoother transition between the approximated functions, as they have to include the endpoints of the intervals. As can be seen in Figure 7, so far, the transitions are smoothed in some regions. However, due to additional points in the intervals, the `validator` identifies more unfitted regions. Thus, the overall approximation consists of more piece-wise functions and introduces more but smaller tails.

Secondly, the transition is smoothened by fitting the sigmoid function in the border points. At first, we initialize final smoothened function as $f_{smoothened}(x) = f_1(x)$. In turn, the $\sigma_{transition}$ is calculated as shown in Equation 5, where the *width* parameter controls the smoothness of the transition around

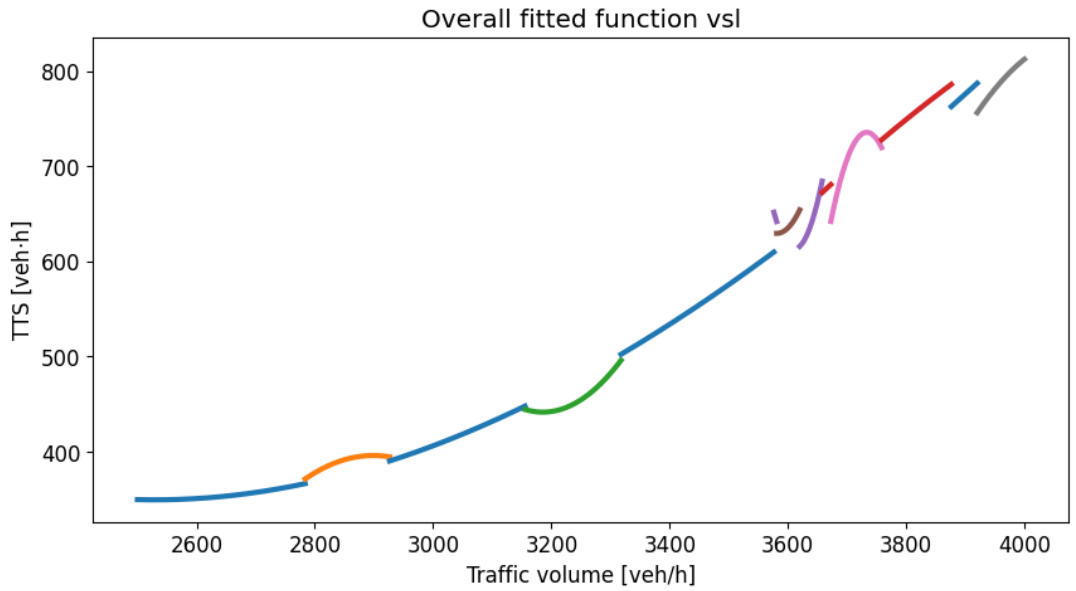


Figure 7: Tailing problem adding border points

border points, x_{conn}, of the functions. After calculating the transition values for N number of functions and $N-1$ border points, the final approximated function is computed by Equation 6. Hence, the final result presents cleaner/smoother transitions (see Figure 8).

$$\sigma_{\text{transition}_i} = \frac{1.0}{1.0 + \exp\left(\frac{-2}{\text{width}\cdot(x - x_{\text{conn}_i})}\right)} \tag{5}$$

$$f_{\text{smoothened}}(x) = (1 - \sigma_{\text{transition}_i}) \cdot f_{\text{smoothened}}(x) + \sigma_{\text{transition}_i} \cdot f_{i+1}(x) \quad \text{for} \quad i = 1 \text{ to } N-1 \tag{6}$$

Finally, we combined the both ideas adding the bordering points and using sigmoid for the smoothen transitions as can be seen in the Figure 9

7. Conclusions

This paper outlined the need for efficient usage of simulation runs in simulation studies, as simulations also incur costs (typically in terms of computation time). For the approximation of the system's behavior, we propose an approximation in partial functions, which have been approximated based on obtained data points (i.e., results of simulation runs). We have presented an approach that allows obtaining sufficiently good enough functional approximations while keeping the overall number of simulation runs as low as possible. The approach is configurable so that the required domain knowledge (e.g., about the underlying simulation model) and the purpose of the overall simulation study can be considered, for instance, in how good an approximation needs to be. Furthermore, we have implemented the SimEx approach so that it can be operated independently from a specific simulator, and all functionalities can be configured. In cooperation with the case study originated in [10], we have demonstrated its practical applicability and potential value for research projects that rely on investigating the behavior of designed artifacts by simulating them in their target environment.

As highlighted in the previous discussion, the current SimEX implementation is a work in progress, with ongoing improvements planned for future work. Specifically, we are addressing the issue of tailing, where trade-offs, such as the degree of disc-continuity between the partial approximated functions,

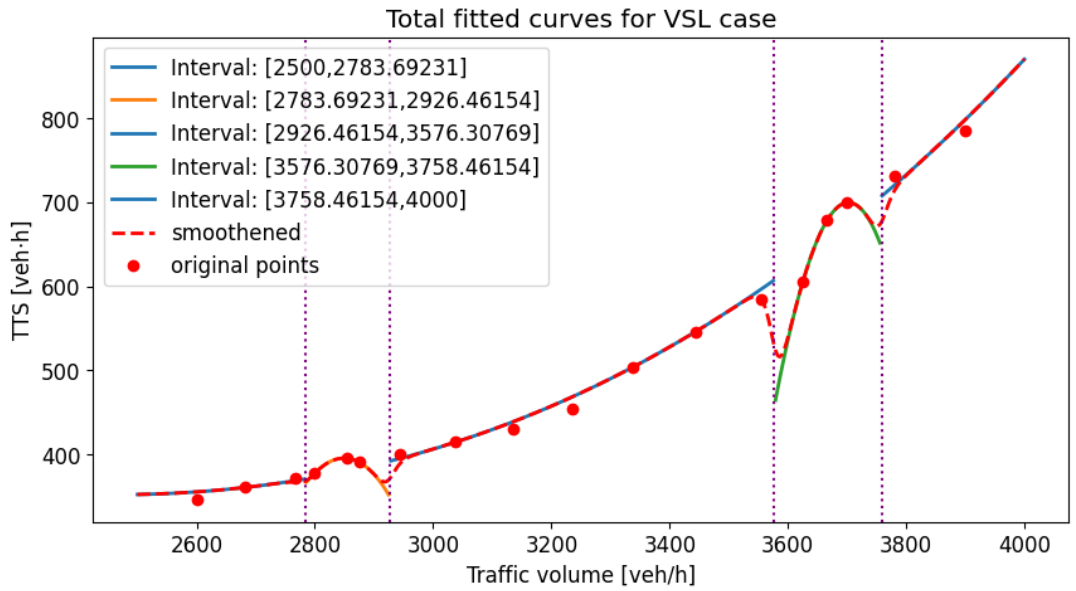


Figure 8: Tailing problem sigmoid fitting

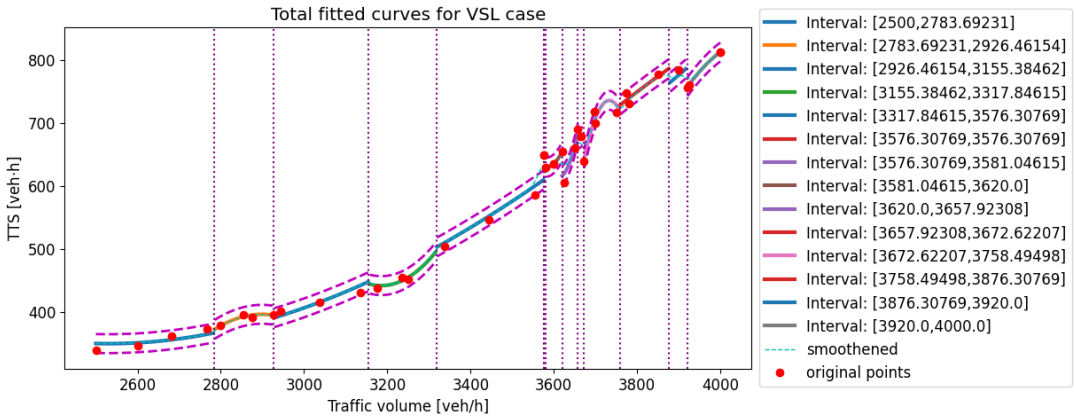


Figure 9: Tailing problem combined solution

need to be considered. This adaptability and continuous improvement underscore the potential for future use and development of the SimEx approach. As demonstrated above, the configuration of the software significantly influences outcomes and related computational effort. This software is designed to be used by experts in their respective domains, who may require recommendations for the configuration, such as best practices for determining suitable parameters for the specific case at hand. To further enhance its usability, more case studies are needed to provide appropriate recommendations for software configuration. Further conceptual extension is to increase the dimensionality of both the input and output space (i.e., handling various inputs for a simulation system) and potentially several non-combinable outputs of the simulations.

Our partners will also continue using the SimEx approach by taking advantage of its functionality to compare the performance of their traffic controllers automatically and identify traffic conditions in which the system's performance is insufficient. This will allow the design of tools to support designers during the design process of traffic controllers, giving them feedback about the performance of their ideas in a reasonable time and highlighting conditions for which their approach still needs to be improved. In the case of controllers based on Machine Learning techniques, this approach can be

expanded by identifying weak performing traffic conditions and automatically creating corresponding elements to expand the training set of the controller and trigger a re-training of the controller.

References

[1] Q. Wang, Time series simulation method of meteorological elements based on arima model, in: 2021 2nd International Conference on Big Data Artificial Intelligence Software Engineering (ICBASE), 2021, pp. 358–362. doi:10.1109/ICBASE53849.2021.00073.

[2] S. Liu, J. Liang, J. Yu, Q. He, Y. Liu, Editorial: Advanced modeling and simulation of nuclear reactors, Frontiers in Energy Research 11 (2023). URL: https://www.frontiersin.org/journals/energy-research/articles/10.3389/fenrg.2023.1189328. doi:10.3389/fenrg.2023.1189328.

[3] P. Wang, X. Zheng, H. Liu, Simulation and forecasting models of covid-19 taking into account spatio-temporal dynamic characteristics: A review, Frontiers in Public Health 10 (2022). URL: https://www.frontiersin.org/journals/public-health/articles/10.3389/fpubh.2022.1033432. doi:10.3389/fpubh.2022.1033432.

[4] S. A. Visser, M. Aurell, R. D. Jones, V. J. Schuck, A.-C. Egnell, S. A. Peters, L. Brynne, J. W. Yates, R. Jansson-Löfmark, B. Tan, M. Cooke, S. T. Barry, A. Hughes, U. Bredberg, Model-based drug discovery: implementation and impact, Drug Discovery Today 18 (2013) 764–775. URL: https://www.sciencedirect.com/science/article/pii/S135964461300158X. doi:https://doi.org/10.1016/j.drudis.2013.05.012.

[5] I. Mauleón, A statistical model to forecast and simulate energy demand in the long-run, Smart Energy 7 (2022) 100084. URL: https://www.sciencedirect.com/science/article/pii/S2666955222000223. doi:https://doi.org/10.1016/j.segy.2022.100084.

[6] M. Windirsch, Empirical modelling of man-made disaster scenarios, in: T. Walker, D. Gramlich, M. Bitar, P. Fardnia (Eds.), Ecological, Societal, and Technological Risks and the Financial Sector, Springer International Publishing, Cham, 2020, pp. 329–363. URL: https://doi.org/10.1007/978-3-030-38858-4_15. doi:10.1007/978-3-030-38858-4_15.

[7] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and analysis of computer experiments, Statistical Science 4 (1989) 409–423. URL: http://www.jstor.org/stable/2245858.

[8] A. George, S. Joseph, Approaches for modelling the climate change impacts on ecosystems, in: S. Dhyani, D. Adhikari, R. Dasgupta, R. Kadaverugu (Eds.), Ecosystem and Species Habitat Modeling for Conservation and Restoration, Springer Nature Singapore, Singapore, 2023, pp. 87–99. URL: https://doi.org/10.1007/978-981-99-0131-9_5. doi:10.1007/978-981-99-0131-9_5.

[9] E. R. Müller, R. C. Carlson, W. Kraus, M. Papageorgiou, Microsimulation Analysis of Practical Aspects of Traffic Control With Variable Speed Limits, IEEE Transactions on Intelligent Transportation Systems 16 (2015) 512–523. doi:10.1109/TITS.2014.2374167.

[10] K. Kušić, D. Calvaresi, A. Liffey, L. Fanda, M. Gregurić, R. Schumann, Evaluation of Traffic Controller Performance via Systematic Exploration, in: Post-proceedings of the 66th International Symposium ELMAR-2024, 2024.

[11] F. Di Maio, C. Pettorossi, E. Zio, Entropy-driven monte carlo simulation method for approximating the survival signature of complex infrastructures, Reliability Engineering System Safety 231 (2023) 108982. URL: https://www.sciencedirect.com/science/article/pii/S095183202200597X. doi:https://doi.org/10.1016/j.ress.2022.108982.

[12] E. Cuevas, J. Gálvez, K. Avila, M. Toski, V. Rafe, A new metaheuristic approach based on agent systems principles, Journal of Computational Science 47 (2020) 101244. URL: https://www.sciencedirect.com/science/article/pii/S1877750320305421. doi:https://doi.org/10.1016/j.jocs.2020.101244.

[13] M. Edali, G. Yücel, Exploring the behavior space of agent-based simulation models using random forest metamodels and sequential sampling, Simulation Modelling Practice and Theory 92 (2019) 62–81. URL: https://www.sciencedirect.com/science/article/pii/S1569190X18301941. doi:https://doi.org/10.1016/j.simpat.2018.12.006.

[14] R. Schumann, C. Taramarcaz, Towards systematic testing of complex interacting systems, Proceedings of the First Workshop on Systemic Risks in Global Networks co-located with 14. Internationale Tagung Wirtschaftsinformatik (WI 2019) (2019) 55–63.

[15] K. Kušić, E. Ivanjko, F. Vrbanić, M. Gregurić, I. Dusparic, Dynamic variable speed limit zones allocation using distributed multi-agent reinforcement learning, in: In Proceedings of the 2021 IEEE 24th Int. Conference on Intelligent Transportation Systems (ITSC), 2021, pp. 1–8.

[16] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, E. Wiessner, Microscopic Traffic Simulation using SUMO, in: 2018 21st International Conference on Intelligent Transportation Systems, 2018, pp. 2575–2582. doi:10.1109/ITSC.2018.8569938.