R. Reussner, A. Koziolek, R. Heinrich (Hrsg.): INFORMATIK 2020,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 883

Development of a Vehicle Simulator for the Evaluation of a Novel Organic Control Unit Concept

Melanie Brinkschulte[1]

Abstract: New challenges in the field of automotive systems (e.g. autonomous driving) require innovative and highly robust vehicle architectures. These are intended to increase the reliability and fault tolerance of the system and therefore realize the transition from Fail-Save to Fail-Operational behavior. Organic computing is a possible approach to achieve these goals. Based on an artificial hormone system and an artificial DNA, a novel organic control unit concept exists.
In this paper we introduce an evaluation tool for this novel concept. Therefore, a simulator physically models the longitudinal and lateral dynamics of a vehicle. For an easy handling, visualization and reproducibility of experiments, an user interface and a scripting language are designed. In extensive evaluation runs the usability of the vehicle simulator is tested. Hereby, real vehicle data is used.

Keywords: Vehicle Simulator; Organic Computing; Fail-Operational

1 Introduction

In this paper we introduce an evaluation tool for a novel organic control unit concept based on an artificial hormone system (AHS) [vBP11] and an artifical DNA (ADNA) [Br15] . Therefore, a simulator physically models the longitudinal and lateral dynamics of a vehicle. The parameters of the vehicle (weight and measures, engine and gear parameters, brake parameters, air and roll resistance, . . .) can be individually chosen. For an easy handling, visualization and reproducibility of experiments, an extensive user interface and a scripting language is designed. Also, this simulator allows the evaluation of various automotive control components (ECUs) like ABS, ASR, power steering and cruise control. All input data like brake, throttle and steering positions can be given by the user interface or the scripting language. In addition, both input options can be used simultaneously. The output values like brake force, wheel speed, vehicle speed, steering angles, etc. are visualized in the user interface, timestamped and written to a logfile for detailed examination. Thereby, the user can choose which physical value is logged as well as the time resolution of the logging process. By fault injection, ECU failures at run-time can be induced at arbitrary times during a simulation run.
This paper is structured as follows: Section 2 gives a short overview of the designed physical models. Following, Section 3 describes the architecture and the interaction of the

[1] University of Mannheim, Chair of Information Systems II, Schloss, 68131 Mannheim, Germany & Goethe University Frankfurt am Main, Computer Science Department, Robert-Maier-Str. 11-15, 60325 Frankfurt am Main, Germany, brinkschulte@uni-mannheim.de

doi:10.18420/inf2020_79

components of the simulator. Section 4 presents an extract of the extensive evaluation and Section 5 discusses related work. Finally, Section 6 concludes this paper.

2 Models

In this work multiple physical models (vehicle dynamics, steering, brake and engine) are designed and used. For reason of space, we can only shortly enumerate these models here. For more details, please refer to [Br19].
The **vehicle dynamics model** is based on the linear single-track model. However, this is not sufficient for the desired purpose and is therefore extended (by adding of longitudinal dynamics, frictional conditions and accuracy by removing the small angle approximation, extension to an rudimentary two-track model) to an efficient nonlinear two-track model without small angle approximation. The **steering model** is a speed-dependent steering system with optional steering assistance. The **brake model** includes characteristic curve mappings and brake cylinder delay. The **engine model** includes an optional adjustable four-wheel drive, as well as drive delay and dead times. Furthermore, a simple **gearbox model** was realized.

3 Simulator

In this section, the architecture as well as the communication and interaction of the components (user interface, physical models, simulator-sensor/actuator interface) of the vehicle simulator is shown. The vehicle simulator is implemented in C++ while Qt 5.11.1 is used to implement the graphical user interfaces.

3.1 Architecture

The simulator consists of three parts: the physical models of the vehicle, the user-interface and the simulator sensor/actuator interface (Figure 1). The physical models have internal state data (e.g. speeds, distances travelled, angles, etc.) and receive vehicle data (e.g. the vehicle mass, vehicle dimensions, etc.), environmental data (e.g. the static/sliding friction value between road and tires) and input data (e.g. a steering angle, an accelerator pedal position, etc.). They then use these to calculate the output data (e.g. forces and accelerations). Through the input and output data, the physical models are connected via the simulator sensor/actuator

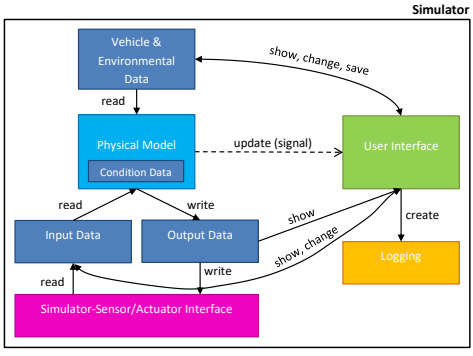


Fig. 1: Architecture of the simulator

interface to the AHS and the ADNA that realize the vehicle's control units. The user interface can display, change and save vehicle condition and environment data. Furthermore, the settings for the creation of a parameterizable log files can be defined in the interface. The visualization consists of a top view with optional fade-in of different force, track and speed vectors as well as a side view, which shows the wheel speeds and the adhesion conditions. Furthermore the simulator is real-time capable. The two-layer real-time architecture is shown in Figure 2. The outer layer uses a $10ms$ period to ensure smooth and jitter-free

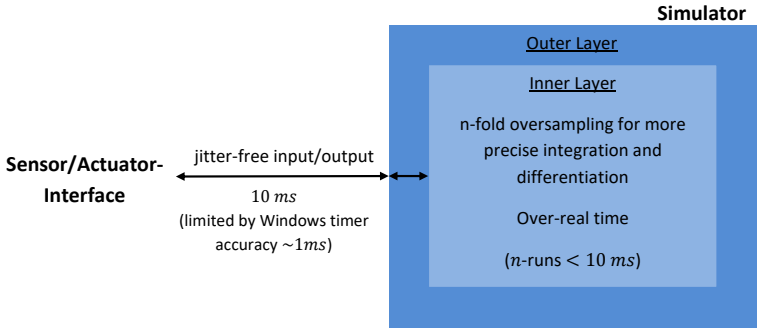


Fig. 2: Real-time architecture of the vehicle simulator

real-time input/output. This cycle time was chosen because the Windows timers used in the implementation have an accuracy of about $1ms$. In the inner layer, a n-fold oversampling (n-fold execution of a simulation run) is performed to achieve a more precise integration and differentiation. The number of passes of the inner layer can be individually adjusted by the user. The only condition is that the time required for this number of runs is less than $10ms$ (run time of outer layer). This means that over-real time is present there.

3.2 Interaction

In Figure 3 the interaction and data transfer between the user interface, the physical models and the simulator-sensor/actuator interface is shown. The physical models are divided into the submodels vehicle dynamics, brake, engine and steering. The sensors of the Simulator Sensor/Actuator Interface receive their inputs from the physical submodels and from the user (e.g. brake pedal sensor, accelerator pedal sensor, etc.). The user has two possibilities to create his inputs. On the one hand, he can make entries via the user interface to directly control the vehicle and influence its environment. On the other hand, in order to enable precisely repeatable experiments, it is possible to specify input in the form of a script file. The input data is then converted by the control units into corresponding actuator values (e.g. brake cylinder control, drive control, steering angle control, etc.) according to the artificial DNA running on them. These values then enter the physical models of the steering, the brake and the engine. The outputs of these models are then passed on to the vehicle dynamics model, which in turn generates new sensor signals in a closed control loop.

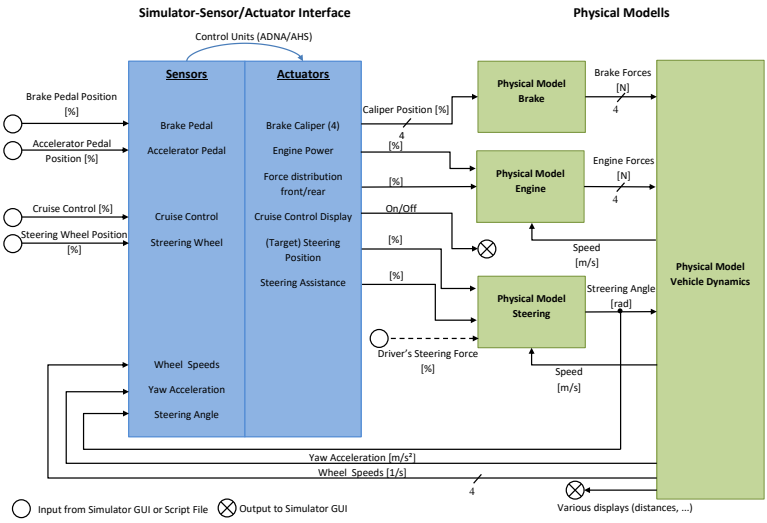
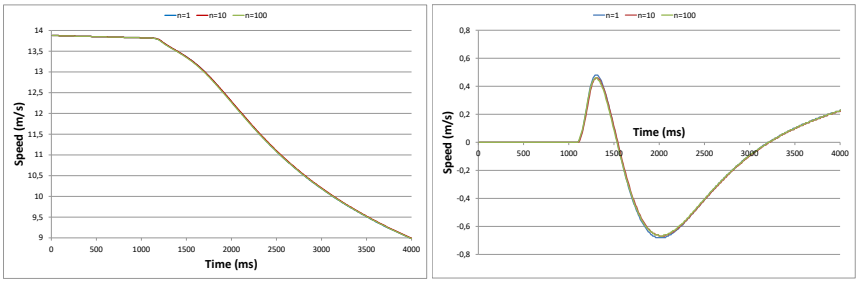


Fig. 3: Interaction and data transfer between the user interface, the physical models and the simulator-sensor/actuator interface

4 Evaluation

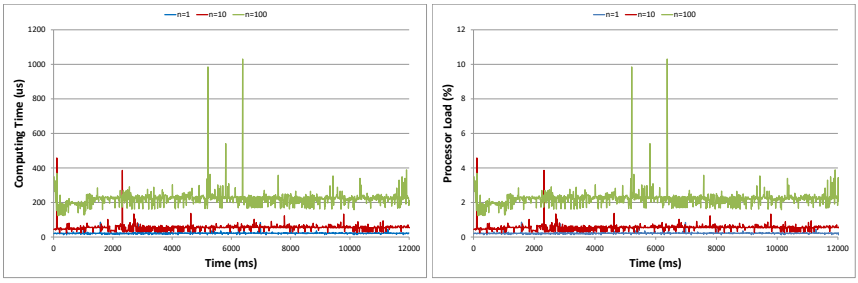
Extensive evaluations of the presented work are made. Unfortunately in the scope of this paper only the most important evaluation results can be presented in detail.
At first, the trade-off between simulator processor load and accuracy is evaluated. Therefore, a simple reproducible experiment (vehicle initiates a curve at a given starting speed of $50\frac{km}{h}$, the static friction is set to a value so that the vehicle doesn't slide) with different numbers of steps ($n = [1, 10, 100]$) per simulation period in the inner layer of the simulator is used. To determine the gain in accuracy, the speed in x- and y-direction are compared exemplary. In the diagrams, minimal deviations of the curves from each other can be seen.



(a) Comparison of the speed in x-direction (b) Comparison of the speed in y-direction

Fig. 4: Comparison of speed in x- and y-direction at $n = 1$, $n = 10$ and $n = 100$ steps per simulation period

The speed in x-direction (Figure 4a) shows the smallest deviations. These never exceed 0.1%, hence we have nearly one line in the figure. For the speed in y-direction (Figure 4b), slightly larger deviations of a maximum of 4% are visible. Nevertheless, the curves are almost identical at $n = 10$ and $n = 100$ steps per simulation period. Only the curves resulting from only one step per simulation period differ slightly more from the others at two points ($1280ms - 1340ms$ and $1810ms - 2140ms$). The resulting computing time for a simulation period and the resulting processor load are shown in Figure 5. As expected, the required computation time per simulation period and thus the processor load increases with growing number of simulation steps n. This results in a maximum calculation time of $1.031ms$ with $n = 100$ which corresponds to a maximum processor load of 10%. It turns out that the



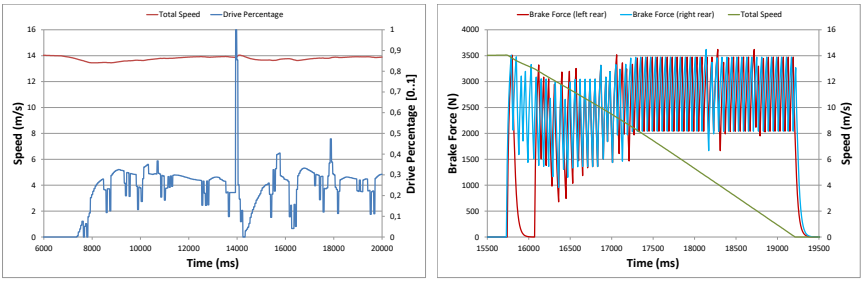
(a) Computing time (b) Processor load

Fig. 5: Required computing time and resulting processor load at $n = 1$, $n = 10$ and $n = 100$ steps per simulation period

model works with high accuracy at low computational effort. Especially $n = 10$ turns out to be a good choice with low overhead and high accuracy.
Next, the longitudinal and lateral dynamics have been evaluated with different experiments[2]. The vehicle is also operated in the non-linear range (skidding). It turns out the simulator

[2] Longitudinal: reality comparison of acceleration time from 0 to 100 $\frac{km}{h}$, maximum final speed and braking distance from 100 $\frac{km}{h}$ to standstill. Lateral: pure steering and steering in combination with braking experiments

behaves as expected and the results are very close to those of a real car. In this paper, we focus on the evaluation of the closed control loop between simulator and control units by means of a cruise control and ABS. The aim of the simulator is to create an evaluation tool for the AHS together with the ADNA as an organic control unit concept. This ECU concept should keep the system operational even in case of component failure and thus show the desired *Fail-Operational* behaviour.
Therefore, in further evaluation the behaviour of the closed control loop in case of failure of individual DNA processors is examined. Failures of processors (ECUs) were added during ABS braking in a corner or speed regulation by cruise control. In Figure 6a the failure of the processor is clearly shown by a peak in the drive power percentage. The processor failure causes the wheel speed sensors of the front wheels to be temporarily lost. The cruise control implemented uses this data to determine the speed of the vehicle. If it now receives no more rotation speed, it assumes that the vehicle has a speed of $0\frac{m}{s}$ and thus accelerates at maximum to reach the set target speed. As soon as the AHS/ADNA has distributed the wheel speed detection to the remaining processors in one of the next hormone cycles by self-healing, the failure is eliminated and the cruise control works correctly again. Here the failure lasts about $100ms$. In Figure 6b the failure of a processor is also clearly visible.



(a) Simulation results of the speed experiment in case of processor (ECU) failure during cruise control

(b) Simulation results of the ABS corner brake experiment in case of processor (ECU) failure

Fig. 6: Simulation results of processor (or control unit) failure experiments

The braking force at the rear left wheel drops to zero shortly after the start of braking at an approximate simulation time of about $15860ms$. After a simulation time of about $16060ms$, the braking force is again controlled by the ABS. The failure was therefore corrected after about $200ms$. In Figure 7 the comparison between the speeds and the distances covered with and without failure is shown. The speed curve during failure shows a slight bend, which is caused by the short-term reduced braking effect during failure. As a result, the vehicle comes to a halt about $250ms$ later than without failure. At the time the vehicle comes to a halt without failure, the vehicle still has a speed of $1.2\frac{m}{s}$ during the failure. This results in an extension of the braking distance of approximately $0.15m$. These evaluations show the suitability of the developed vehicle simulator for the intended application. The simulator is able to simulate failures of processors or control units, which allows to investigate and analyze such failure scenarios.

5 Related Work

Vehicle modeling has always been of big importance for the automotive industry [WSK11]. A general overview of vehicle models can be found in [SHB13] and [MW14]. Here a suitable compromise between model complexity and the number of model parameters or computing time should be found for the respective application, as is also described in [Un13]. This enables an optimal use of the respective model within the scope of the intended application. Highly complex multi-mass models are used, for example, to evaluate the driver's driving experience [Un13] or in comfort simulation [Am13]. If, on the other hand, the lane control of vehicles [Ar15], [He09] or the evaluation of vehicle measurement data during road tests [Se05] are concerned, simple models such as the linear single lane model are usually sufficient.

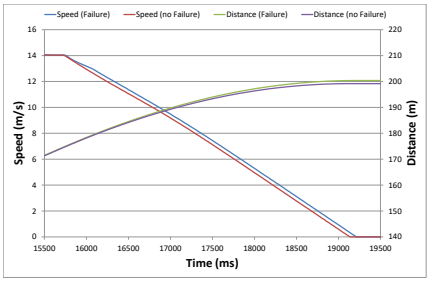


Fig. 7: Comparison between speeds and distances in the ABS-experiments with and without failure

In the presented work, the focus is on vehicle dynamics in connection with a novel, robust control unit concept. In case of an ECU failure, the evaluation of the lateral and longitudinal dynamics of the vehicle is of great importance, whereby the ECUs receive information from all four wheels. In the event of skidding (oversteer and understeer), the vehicle also leaves the linear range. The linear single-track model was therefore too simplified and not suitable for the desired purpose. However, driving comfort was also not in the focus of the work, i.e. there was no need to use a complex multi-mass model.

For this reason, an adapted model was developed, which extends the linear single-track model to a nonlinear model with two-track components. With this model, meaningful simulations for the evaluation of the novel ECU concept can be performed on the basis of an ADNA (especially regarding failure and robustness of ECUs), while the model complexity remains at a reasonably low level.

6 Conclusion

In this paper a vehicle simulator is presented as an evaluation tool of an organic control unit concept (represented by the AHS in combination with the ADNA) in the automotive field. For this purpose, physical models for steering, brake, engine and vehicle dynamics are developed, validated and implemented. With these models a comprehensive evaluation of driving situations for the evaluation of the organic ECU concept or robust ECU is possible. To enable reproducible experiments, a script language was developed and implemented. This allows flexible experiments under identical conditions and thus enables the comparison of different ECU concepts against each other. In future work, the failure behavior of more sophisticated control units for autonomous driving will be evaluated with this tool.

Bibliography

[Am13] Amelunxen, Hendrik: Fahrdynamikmodelle für Echtzeitsimulationen im komfortrelevanten Frequenzbereich. Dissertation, Universität Paderborn, 2013.

[Ar15] Arndt, Albrecht: Querregelung eines spurgeführten Modellfahrzeugs. chapter 5 Modellbildung, 2015.

[Br15] Brinkschulte, Uwe: An artificial DNA for self-descripting and self-building embedded real-time systems. In: Concurrency and Computation: Practice and Experience, volume 28. Wiley Online Library, 2015.

[Br19] Brinkschulte, Melanie: , Entwicklung eines Fahrzeugsimulators zur Evaluation eines neuartigen organischen Steuergerẽkonzepts im Automotiven Bereich, 2019. Masterthesis at Goethe Universität Frankfurt am Main.

[He09] Hensel, Enrico: Führungskonzept eines autonomen Fahrzeugs, Vorbetrachtung und Bewegungsmodell. Seminarbericht, Hochschule für Angewandte Wissenschaften Hamburg, 2009.

[MW14] Mitschke, Manfred; Wallentowitz, Henning: Dynamik der Kraftfahrzeuge. Springer, 2014.

[Se05] Sentürk, Fikret: Durchführen von Fahrversuchen hinsichtlich einer Optimierung von FHTW-Fahrdynamikfahrzeug. Diplomarbeit, Fachhochschule für Technik und Wirtschaft Berlin, 2005.

[SHB13] Schramm, Dieter; Hiller, Manfred; Bardini, Roberto: Modellbildung und Simulation der Dynamik von Kraftfahrzeugen. Springer, 2013.

[Un13] Unterreiner, Michael: Modellbildung und Simulation von Fahrzeugmodellen unterschiedlicher Komplexität. Dissertation, Universität Duisburg-Essen, 2013.

[vBP11] von Renteln, Alexander; Brinkschulte, Uwe; Pacher, Mathias: The Artificial Hormone System - An Organic Middleware for Self-organising Real-Time Task Allokation. In (Müller-Schloer, Christian; Schmeck, Hartmut; Ungerer, Theo, eds): Organic Computing - A Paradigm Shift for Complex Systems, chapter 4.4. Springer, 2011.

[WSK11] Wiedemann, Jochen; Schröck, David; Krantz, Werner: Fahrzeugdynamik, Themenheft Forschung, volume 7. Universität Stuttgart, 2010-2011.