

Estimating Vehicle Suspension Characteristics for Digital Twin Creation with Genetic Algorithm

Tamás Ormándi^{1*}, Balázs Varga¹, Tamás Tettamanti¹

¹ Department of Control for Transportation and Vehicle Systems, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, H-1111 Budapest, 2 Stoczek street, Hungary

* Corresponding author, e-mail: ormandi.tamas@edu.bme.hu

Received: 17 May 2021, Accepted: 15 June 2021, Published online: 30 July 2021

Abstract

Usage of simulation techniques like Vehicle-in-the-Loop, Scenario-in-the-Loop, and other mixed-reality systems are becoming inevitable in autonomous vehicle development, particularly in testing and validation. These methods rely on using digital twins, realistic representations of real vehicles, and traffic in a carefully rebuilt virtual world. Recreating them precisely in a virtual ecosystem requires many parameters of real vehicles to follow their properties in a simulation. This is especially true for vehicle dynamics, where these parameters have high impact on the simulation results. The paper's objective is to provide a method that can help reverse engineering a real car's suspension characteristics with the help of a genetic algorithm. A detailed description of the method is presented, guiding the reader through the whole process, including the meta-heuristic function's settings and how it interfaces with IPG Carmaker. The paper also presents multiple measurements, which can be effortlessly recreated without expensive devices or the need to disassemble any vehicle parts. Measurements are reproduced in two separate simulation tools with special scenarios providing an efficient way to analyze and verify the results. The provided method creates vehicle suspension characteristics with adequate quality, opening up the possibility to use them in the creation of digital twins or creating virtual traffic with realistic vehicle dynamics for high-quality visualization. Results show satisfying accuracy when tested with OpenCRG.

Keywords

digital twin, genetic algorithm, mixed-reality, reverse engineering, Scenario-in-the-Loop, simulation, suspension, Vehicle-in-the-Loop, traffic simulation

1 Introduction

Nowadays, engineers put much effort into creating simulation environments that can accurately represent reality. Simulation eases the financial and temporal burdens of complicated measurements, where several factors must be present before tests can be executed. This is especially true for autonomous vehicles and their functions, where safety, security, and even legal aspects must be taken into account. In addition, dangerous, rare, or even unrealistic scenarios cannot be reliably presented to self-driving cars in real world traffic conditions (Butenuth et al., 2017; Fehér et al., 2020a).

By creating new highly automated vehicle systems, new issues arise. In order to evaluate real vehicles in simulation a further step has to be taken starting from conventional Vehicle-in-the-Loop (VIL) simulations. To reproduce reality in a virtual environment, so-called digital twins must be created (Madni et al., 2019). These virtual copies of the physical objects should accurately represent

their appearance, physical properties, and every aspect, affecting a test scenario's goals for example path planning (Fehér et al., 2020b). Implementing vehicle dynamics with high accuracy for a digital twin is a tough nut to crack, the desired accuracy depends on the goal of the test. Although, suspension kinematics have well grounded literature e.g., (Pacejka, 2005), the challenge lies in estimating the parameters of the model (Herrerros et al., 2002).

Detailed suspension characteristics are often not available: either it is not provided by car manufacturers or wear and tear have taken its toll. The real suspension parameters differ from factory defaults.

Without the suspension characteristics, it is impossible to create a digital-twin for vehicle dynamics purposes, so there must be a way to find the necessary parameters. Such parameters can be measured physically, but in that case, the vehicle must be disassembled and costly

measuring devices are required to gather the demanded information. The idea behind this paper is to recreate these parameters by reverse engineering them with the help of easily executable measurements and the usage of meta-heuristic optimization tools, e.g., Genetic Algorithm (GA). The genetic algorithm is efficient in optimization or parameter identification in the automotive industry. Several papers are discussing this method to find the optimal values related to suspension designs (Hemati and Shoostari, 2019; Mitra et al., 2016) or to find optimal parameters for the Pacejka tire model (Vetturi et al., 1996) and it is effective even in the case of Advanced Driver Assistant Systems (ADAS) (Klück et al., 2019).

With the help of GA, it is possible to reverse engineer a unique car's suspension characteristics, which is accurate enough to create suitable vehicle dynamics in a virtual environment. This process can help to create virtual traffic, which has realistic movements in a high-quality visualization and can be used in digital twin creation. Additionally, it can support creating a mixed-reality system (Varga et al., 2020). Performing cumbersome disassembling and precise measuring of multiple vehicles is not needed. It is enough to reverse engineer some vehicles of different size classes to create a diverse, dynamic virtual traffic system. The paper presents multiple scenarios for measurements and virtual simulations, which can validate the genetic algorithm results and provide a sneak-peek into a system that can be used as a digital twin and can enhance traffic simulation with vehicle dynamics for complex automotive test scenarios. The whole method and validation are presented through real measurements and they are virtually recreated in two simulation software platforms/products:

1. high fidelity vehicle dynamics simulator IPG Carmaker (IPG Automotive GmbH, 2020) and
2. Unity 3D game engine (Unity Technologies, 2020).

The purpose of IPG Carmaker is to provide an accurate suspension model that can be calibrated with real-world measurements. Unity 3D is then used to create realistic visualization with the obtained vehicle parameters. For a higher level of accuracy, an Open Curved Regular Grid (CRG) based test scenario is also implemented, providing a more complex way of testing and validation of the results (Potó et al., 2018).

The paper is organized as follows. Section 1 provides information about the real-life measurements' setup hardware and software-wise. Section 2 gives information

about simulation software used for the realization of virtual reality. Section 3 describes the first scenario, gives information and introduces the real vehicle and its digital twin. It provides a detailed description of reverse engineering suspension characteristics with the genetic algorithm and presents the method's results. Section 4 grants a detailed description of a second, more complex scenario with a CRG road profile for validation purposes, showcasing the simulations' results and presenting a custom suspension model implemented in Unity 3D. It also provides a method to import CRG files in the game engine. Finally, Section 5 concludes the findings of the paper and projects further usages and developments of the method.

2 Reality – measurement setup

This section discusses the real-world data used for calibrating the simulated vehicle. This work focuses on suspension characteristics and the vertical motion of the car. To this end, the measured reference signal will be the vertical acceleration of the chassis. For these measurements, a device is needed to record this acceleration and some other required data with a predefined quality (Ma et al., 2013). Any industrial level sensor could be used to fulfill these requirements, but they can be expensive and further signal processing might be necessary. Thus, an alternative way is chosen. Fortunately, nowadays, everybody has powerful computers in their pockets with high-quality sensors and sufficient computational capacity. The device used for measurements contains a Bosch Sensortec BMI160 low power inertial measurement unit designed for applications like indoor navigation, 3D scanning, and even for 9-axis motion detection (Bosch Sensortec GmbH, 2020). This sensor provides a reliable base for acceleration measurements with supreme quality at 100 Hz frequency. Using a smartphone for this purpose is a perfect choice because the accelerometer signals can be obtained easily. In addition, it can be conveniently mounted on the vehicle. Accelerometer signals can be straightforwardly read in real-time with the help of Matlab Mobile application. As the digital twin is realized in IPG Carmaker for Simulink, Matlab is a comfortable choice.

3 Virtuality – simulations

The virtual reality is realized in two software. First, IPG Carmaker, an automotive simulation tool, is used for reverse engineering suspension characteristics. Second, Unity 3D game engine is used for real-time high definition visualization in conjunction with a simplistic vehicle model.

3.1 IPG Carmaker

For calibration purposes, IPG Carmaker was chosen, which has many options to rebuild the measurement's scenario virtually, and beyond that, it has a user-friendly Matlab Simulink interface, that allows the user to access any variables during simulations easily.

Carmaker is an industrial level virtual driving test software, with a complete model environment that starts from intelligent driver models to detailed vehicle models. Allowing users to modify or define their own models and proving many interfacing possibilities the software gives a really high flexibility level. With the help of virtual sensors, engineers can analyze every aspect of vehicles, including vehicle dynamics and autonomous functions. The software has its own tool environment for automated tests and a complete solution for data visualization. These features make this software reliable and powerful in case of validation and reverse engineering.

3.2 Unity 3D

Unity 3D is a real-time development platform for computer games by default; however, it is more and more often used for industrial purposes in the automotive sector. It provides wide range of opportunities for scenario building. As computer games have many scenarios, just like industrial simulations, it can be an adequate tool to build an environment for vehicle testing and validation. It is mainly used for visualization because it can provide stunning graphics, but the possibilities provided by this software go further. It has its own physics and it can be programmed in C# language, which makes this engine really flexible. If used wisely, its potential can be exploited in autonomous vehicle functions, vehicle dynamics, or artificial intelligence (Song et al., 2019). Unity 3D is an excellent digital twin creation solution providing visualization, virtual sensors, functions, and dynamics all packed in one environment. For maximum efficiency, it is easily interfaced with industrial level engineering software like Matlab. Thus, it can fill a multifunctional role in a mixed reality test system (Szalai et al., 2020).

4 Calibration

This section deals with the parameter identification of the simulated vehicle model based on real measurements.

4.1 Test scenario

Identification of the vehicle suspension parameters of IPG Carmaker's high fidelity vehicle model is carried out by a genetic algorithm. In order to construct the digital twin

of the vehicle, the following signals were recorded during measurements with Matlab Mobile:

- X, Y, Z accelerations
- Vehicle speed
- Angular velocities
- GPS coordinates.

First, a test scenario shall be defined where a decent excitation can be applied to the real vehicle in order to generate an adequate oscillation. The environment of the test (road profile) shall be accurately recreated in the virtual environment too. For measurement purposes, a straight road was chosen at the edge of a city at an unfrequented location with a speed bump (Fig. 1), serving as an engineering sandbox for the measurements.

The car was driven over the speed bump multiple times with a velocity of 15 km/h, which results in an excitation of the vehicle's suspensions. The dimensions of the speed bump were also measured because the generated excitation is a crucial factor and it should be carefully recreated in the virtual scenario to reproduce the same oscillation.

If the virtual version's dimensions were different, it would produce a significantly different result.

As the main parameter of interest was the vehicle's vertical acceleration, it was necessary to consider the mounting position of the sensor. The measurement was done by a smartphone fixed in position with a phone holder on the



Fig. 1 Speed bump (GPS coordinates: 47°24'09.0"N 19°17'02.5"E)

car's front windshield. The phone holder is rigid enough and because of the sensor's basic noise, the oscillation generated by the movements of the phone holder were filtered out. Measurements were taken with a sampling rate of 100 Hz. The measured vertical acceleration data was noisy and was biased by the gravitational acceleration. The gravitational acceleration was eliminated during the data processing and the noise was reduced with the help of a low pass filter. Processed data is shown in Fig. 2.

The results of multiple measurements were almost identical, so one of them was chosen to be the reference signal for the optimization.

4.2 Vehicle model

Aware of this information, the scenario can be rebuilt precisely in the virtual reality with the car's identical physical parameters and the environment having the same speed bump. Table 1 summarizes the most important parameters of the test car, which is a Ford Mondeo Mk-IV (2012) Estate (facelift). It has a MacPherson type front suspension and a multi-link rear suspension. The corresponding suspension characteristics are not provided by the manufacturer.

For better accuracy, driver and passenger weights were also considered. Fig. 3 shows the virtual environment based on the real one. In the case of the tires, the default tire size (205/60-16) and the tire model were set for a medium-sized car.

4.3 Genetic algorithm

In this section, the steps are described towards reverse engineering the suspension characteristics.

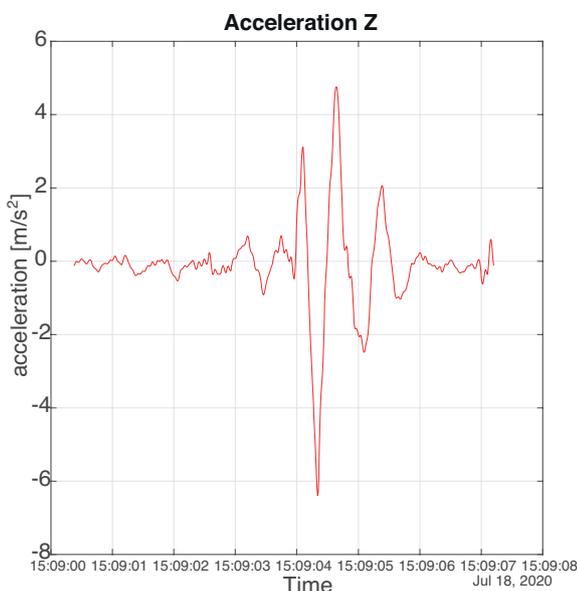


Fig. 2 Processed Z axis acceleration

Table 1 Physical parameters of the vehicle

Parameter	Value
Length	4837 mm
Width	1886 mm
Wheelbase	2850 mm
Weight	1575 kg
Front track	1588 mm
Rear track	1605 mm
Driver's weight	80 kg
Passenger's weight	70 kg



Fig. 3 Simulation environment with a speed bump

Thus, a detailed description of the genetic algorithm connected with the prebuilt IPG Carmaker simulation is given. During the evolutionary algorithm, one has to access and manipulate the parameters of the vehicle between each iteration.

Everything has to start somewhere and it is not different in the case of a simulation loop, which requires an initialization with a starting parameter set of the suspension variables and vehicle dimensions. For an optimal simulation start, the car's suspension parameters should be close to the measured car's values. The virtual vehicle's dimensions were set identical to the real ones. As mentioned before, the load of the driver's and passenger's weight was set in Carmaker, too, with the load's position in the car, recreating the real measurement's circumstances. Initially, the preset linear suspension characteristics of a medium-sized car in IPG Carmaker were set. It provided a common starting point for the simulation loop and stable vehicle dynamics for the first iteration.

It is expected to measure vertical acceleration in the same position virtually as in reality. Otherwise, simulation results would be biased and could mislead the genetic algorithm. In Carmaker, a virtual inertial sensor was positioned at the virtual vehicle's front windshield.

In suspension settings, buffers (shock-absorbing gas-kets installed between the turns of the suspension shock absorbers) were disabled because the car used for the

measurement does not have any spring buffers. Stabilizers were also disabled because the scenario contains only a straight movement and it would not have impact on the results. The genetic algorithm is looking for the optimal parameter setting for:

1. front and rear spring characteristics,
2. front and rear dampening characteristics in push direction,
3. front and rear dampening characteristics in pull direction.

The rest of the parameters were fixed.

Fig. 4 depicts the structure of the simulation setup, which is responsible for reverse engineering parameters related to the vehicle suspension characteristics.

A Matlab script as the core of the system was created to fulfill the required steps. This script does the following steps:

1. Loads the measurement data from the selected .mat file.
2. Processes the Z axis acceleration data to filter out noise and get rid of the sensor's gravitational acceleration.
3. Trims the data to extract only the important part of the signal, where the excitation was applied.
4. Sets up the genetic algorithm's options.
5. Runs the genetic algorithm function.
6. Plots the result and the measured data.

The execution of the system is quite straightforward. The operation begins with loading and processing the measured data as an initialization. Then it loads the options

and calls the genetic algorithm function for the first time. The objective function of the GA calls another function as a function handler and passes the processed measurement data and the investigated parameters. These parameters are indirect because they are not present in the cost function. The inner function will first start the IPG Carmaker Simulink model, which opens the Carmaker user interface connected with Matlab. Carmaker has a dedicated folder in its project containing the Simulink model and every needed script for the interface. When Carmaker is initialized, it starts the simulation for the first time with the pre-set parameter set. Carmaker uses a .car file (Medium.car), which contains every variable of the car model. Every line from the .car file will be stored as a unique cell array element. Carmaker handles every suspension characteristic as a lookup table. The parameters generated by the genetic algorithm are stored in an array, so the inner function will overwrite the lines corresponding to the identical variables line by line, and it will overwrite the .car file to make them have their impact on the next simulation run. With this method, passing the parameters generated by the genetic algorithm to the Carmaker simulation is realized. With the new parameters, Carmaker starts the next simulation. The results are read through the Simulink interface, which has a Simulink block developed for this purpose. After the inner function gets back the simulation result, it trims this signal to the same length as the measured and processed signals containing only the most important part and synchronizes the signals in time. After this, the function calculates the result's objective function value, which tells the genetic algorithm how good the result is with the currently used parameters compared to the measured signal.

As Matlab's genetic algorithm function has many options, only the settings will be mentioned, which were changed from the default one. The first option is maximum generations, that sets the maximum iterations. It was set to 100. The second option is the population size, which is closely related to the maximum generations. This number represents how many iterations create a complete generation. By default, if the number of variables is less than or equal to 5, then it is set to 50. Otherwise, it should be set to 200. In our case it was set to 100 to save computational time. The next option is the number of stall generations, which is set to 50 by default. This value tells the function, how many generations it should run, when the average relative change in the best fitness function value over maximum stall generations is less than or equal to function tolerance. This number was set to 8 because after many

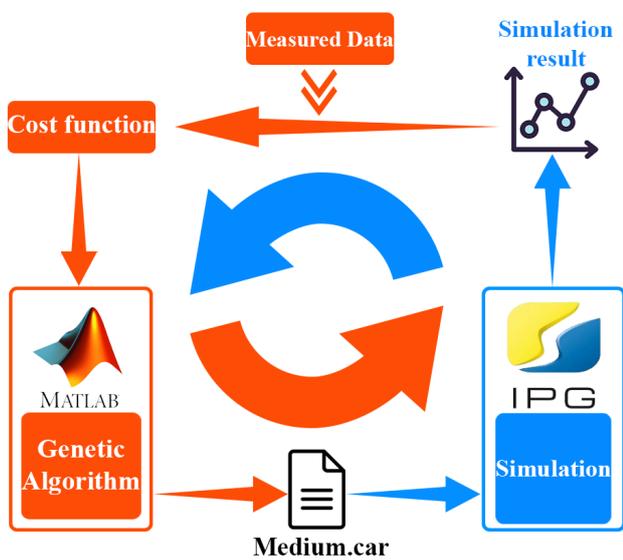


Fig. 4 Genetic algorithm system

experiments, the result did not show any further development when running in stall generations for many iterations and lowering this option's value can spare considerable time. Another important setting is the elite count. This option defines how many individuals are guaranteed to survive from the current generation. The default setting is 0.05 times the number of population size rounded in the direction of positive infinity. The multiplier was changed from 0.05 to 0.15 to save more good parameters for the next iteration. The last modified option was the hybrid function, which provides another optimization method to fine-tune the result after the genetic algorithm stops. By default, it is turned off. It was set to run a pattern search or also known as a direct search, to achieve the system's maximum effectiveness. Determining the optimal number of variables to be searched by the genetic algorithm was a real challenge. If the variable number is too low, the genetic algorithm will not achieve a close solution to the measurement. If it is set to too many variables, it will significantly increase the number of possibilities and the function run-time will skyrocket. The optimal solution was found with 24 variables, which was produced with a reasonable run-time. These 24 variables mean 4 points in every characteristic.

To achieve any results, it is vital to set up proper constraints for the variables searched by the algorithm. It is essential because the genetic algorithm has to be guided to avoid generating physically unacceptable parameters, like negative values in a characteristic or simply enormously high values. If Carmaker gets a parameter set that makes the virtual car leave the road or produces unrealistic dynamics, such as getting an extremely high value for angular rotations, the software immediately aborts the simulation and the genetic algorithm crashes. If the constraints are not chosen correctly for every parameter, it can cause the simulation to be aborted after thousands of iterations, meaning countless wasted time. Unfortunately, there is no way to turn off the simulation abort. Constraints were chosen to grow with the spring compression values or with the damper velocity and the magnitude of forces was chosen related to the starting characteristics. These values were redefined multiple times before it ended in a satisfactory result. The constraints can be defined by their lower and upper bounds for every variable. The chosen bounds and the corresponding characteristic points are summarized in Table 2.

By defining the range for every characteristic point, the simulation abortion was avoided, the genetic algorithm was successfully finishing its job, and even the pattern search was run.

Table 2 Variables and corresponding force constraints

#	Variable	Value	Lower Bound	Upper Bound
1	Front spring compression	0.1 m	500 N	5000 N
2	Front spring compression	0.3 m	5000 N	9000 N
3	Front spring compression	0.5 m	9000 N	15000 N
4	Front spring compression	1.0 m	15000 N	50000 N
5	Rear spring compression	0.1 m	500 N	5000 N
6	Rear spring compression	0.3 m	5000 N	9000 N
7	Rear spring compression	0.5 m	9000 N	15000 N
8	Rear spring compression	1.0 m	15000 N	50000 N
9	Front damper velocity push	0.125 m/s	40 N	1000 N
10	Front damper velocity push	0.25 m/s	1000 N	3000 N
11	Front damper velocity push	0.35 m/s	3000 N	5000 N
12	Front damper velocity push	0.4 m/s	5000 N	8000 N
13	Front damper velocity pull	0.125 m/s	500 N	2000 N
14	Front damper velocity pull	0.25 m/s	2000 N	3000 N
15	Front damper velocity pull	0.35 m/s	3000 N	5000 N
16	Front damper velocity pull	0.4 m/s	5000 N	6000 N
17	Rear damper velocity push	0.125 m/s	40 N	1000 N
18	Rear damper velocity push	0.25 m/s	1000 N	3000 N
19	Rear damper velocity push	0.35 m/s	3000 N	5000 N
20	Rear damper velocity push	0.4 m/s	5000 N	8000 N
21	Rear damper velocity pull	0.125 m/s	500 N	2000 N
22	Rear damper velocity pull	0.25 m/s	2000 N	3000 N
23	Rear damper velocity pull	0.35 m/s	3000 N	5000 N
24	Rear damper velocity pull	0.4 m/s	5000 N	6000 N

In the case of any meta-heuristic algorithm, the cost function is an essential element of the whole system. It guides the algorithm towards the optimal solution, so this function has to be defined very well. The experiments showed that in this case, the most successful cost function was the Euclidean distance of the measured and the simulation result signal, see Eq. (1):

$$d = \sqrt{\sum_{i=1}^n (a_{i_{\text{measured}}} - a_{i_{\text{simulated}}})^2}, \tag{1}$$

where $a_{i_{\text{measured}}}$ is the i^{th} measured vertical acceleration in m/s^2 and $a_{i_{\text{simulated}}}$ is the i^{th} simulated vertical acceleration in m/s^2 .

4.4 Genetic algorithm results

As mentioned before, the genetic algorithm was run several times with many settings before it produced a decent result with optimal running time.

The best result was achieved with 10842 iterations. The result was plotted along with the measured data to be compared in Fig. 5.

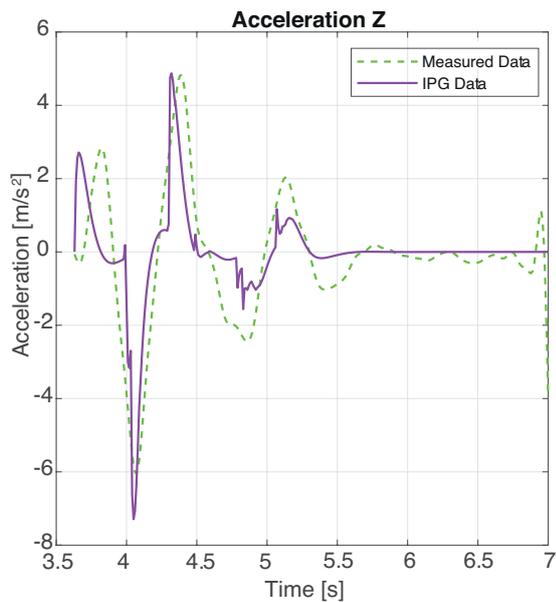


Fig. 5 Comparison of vertical accelerations between measured (green) and simulated (purple) data

The result is quite accurate in terms of the magnitude of accelerations and the frequency of the oscillation matches. There are some sharp spikes on the resulting signal, which can be a model-specific result of the default suspension model. The tire model was also not changed from the default, although it can impact the result. Considering the vehicle body as flexible instead of the rigid body could also affect the result. Obviously, the genetic algorithm settings could be fine-tuned further and further, but it would not significantly impact the results even if the used settings were not optimal. Nonetheless, the result is quite satisfying if we consider the goals. The resulting spring characteristics are shown in Fig. 6 and the damper characteristics in Fig. 7.

As there are no public suspension characteristics available, it is tough to compare the characteristics of the simulation and the real vehicle. The results show that the front spring produces forces with greater magnitude,

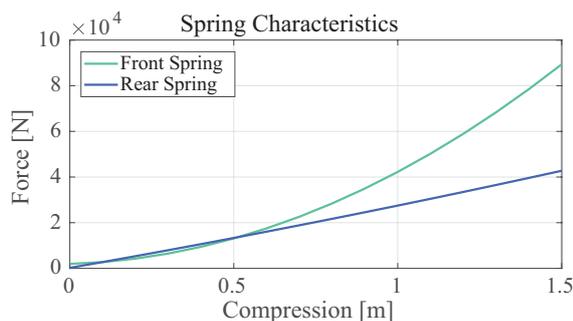


Fig. 6 Spring characteristics generated by the genetic algorithm

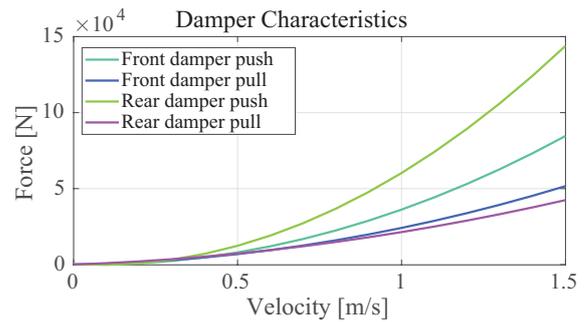


Fig. 7 Damper characteristics generated by the genetic algorithm

which is logical considering the car has its engine in the front. The situation is reversed in terms of damper forces. These results may not be perfect for a real car suspension, but they are perfectly fine for the digital twin to be used while producing similar results to the measurements.

5 Validation

For validation purposes, the calibrated suspension was tested again in another scenario incorporating a road surface available in OpenCRG format too. OpenCRG is an open-source project for the creation, modification, and evaluation of road surfaces. It has its own file format specification. It is an ideal solution for driving simulations related to vehicle dynamics where the road's smallest excitations can be simulated. The data in a CRG file can be easily processed with Matlab and many industrial software support this format. Such data can be recorded with the help of a terrestrial laser scanner (Barsi et al., 2018). Although CRG can be a really accurate solution for road modeling, it has its own drawbacks. It is not a widespread method right now and it is hard to find any laser-scanned roads with decent quality. The laser point clouds are noisy and processing them into a quality CRG is hard. Every disturbance of the laser point cloud must be eliminated; otherwise, it will create spikes on the road surface, resulting in an unreal simulation. This is especially true if only a single point models the tire's contact and the ground in the dynamics model. To create a clean, noise-free CRG surface, a Gaussian filter was used to smoothen the surface.

5.1 Test scenario

The second measurement was performed at the campus of Budapest University of Technology and Economics, which was already scanned and had a processed CRG file. The road section for the test has potholes and it is connected to a parking lot with a relatively large ramp.

The laser-scanned CRG file contains a section of this road with a length of about 90 meters. The measurement was executed on a shorter section of this road. This scenario provides insight into a more regular excitation of the vehicle suspension and it can provide a more complex possibility for data validation. The section of road in question can be seen in Fig. 8

5.2 IPG Carmaker

This software has a built-in function to support CRG files, so it is an effortless task to set up the second scenario. After importing the CRG file, it can be placed anywhere on the road, and it can be even scaled if needed. A high-quality visualization is also available. The software's only drawback is the lack of opportunity to move the virtual car parallel with the real vehicle in real-time because it is not supporting setting vehicle positions exactly.

The results of the second scenario in IPG Carmaker can be seen in Fig. 9.

The results show that the CRG based excitation causes very similar vertical accelerations in the simulation compared to the real measurement. Some particular parts of the signal's shape are almost identical. However, some additional oscillations are present with a higher frequency. The Mean Squared Error of the simulated vertical acceleration to the real measurement is $MSE_{IPG} = 2.4188 \text{ m/s}^2$.

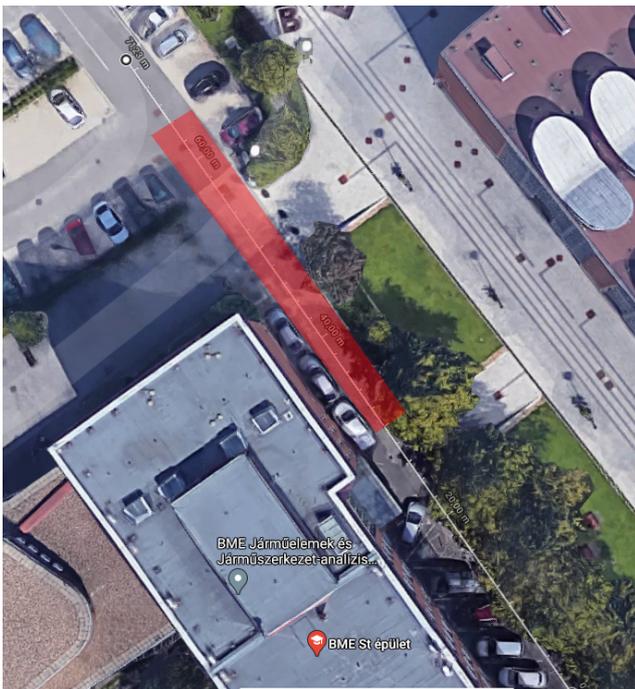


Fig. 8 The road of the second measurement highlighted with red (GPS coordinates: 47°28'43.2"N 19°03'24.4"E)

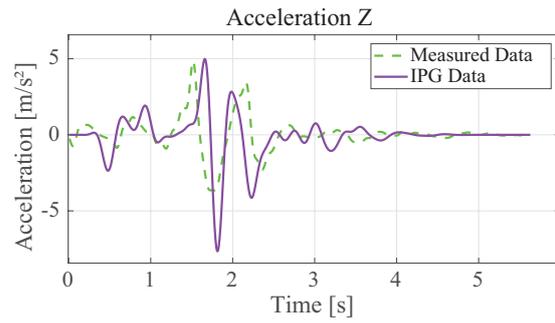


Fig. 9 Carmaker with CRG excitation

5.3 Unity 3D

In Unity 3D, a custom vehicle dynamics model was implemented. In this case, a simple Mass-Spring-Damper system was used to represent the vehicle's suspension with the supplement of the vehicle's weight transfer dynamics. Eq. (2) shows the equation of the Mass-Spring-Damper system (MSD), and Eq. (3) shows the equations used for the weight transfer considering front and rear axles (Jiang et al., 2014).

$$F_i = \frac{m}{4} \ddot{x}_i = -d_i (\dot{x}_i) \dot{x}_i + c_i (x_i)(y_i + x_{0,i} - x_i), \quad (2)$$

where $i = 1, \dots, 4$ corresponds to the front right, front left, rear right, and rear left suspensions. m is the weight of the vehicle in kg, \ddot{x}_i is the acceleration of the mass in vertical direction in m/s^2 , \dot{x}_i is the velocity of the mass in m/s , x_i is the actual position of the mass, $x_{0,i}$ is the length of the unloaded spring in m , c_i is the spring coefficient of the corresponding axle, d_i is the damper coefficient of the corresponding axle and y_i is the excitation of the road surface.

The equation of MSD is divided into damper and spring forces. The suspension's compression and velocity are calculated in the game engine and forces are evaluated by the characteristics.

$$F_{z_{\text{rear}}} = \frac{m}{2} \left(\frac{L_2}{L} - \frac{(a_x + g)h}{(\ddot{x} + g)L} \right) (\ddot{x} + g) - m \left(\frac{L_2}{L} - \frac{(a_x + g)h}{(\ddot{x} + g)L} \right) \left(\frac{(a_y + g)h}{E_1} \right), \quad (3)$$

where m is the vehicle mass in kg, L is the wheelbase in m , L_2 is the distance of the Center of Gravity (CoG) and the rear axle in m , a_x , a_y are the X and Y direction accelerations in m/s^2 , g is the gravitational acceleration in m/s^2 , E_1 is the front track width in m , and h is the height of the CoG in m .

$$F_{z_{\text{front}}} = \frac{m}{2} \left(\frac{L_1}{L} - \frac{(a_x + g)h}{(\ddot{x} + g)L} \right) (\ddot{x} + g) - m \left(\frac{L_1}{L} - \frac{(a_x + g)h}{(\ddot{x} + g)L} \right) \left(\frac{(a_y + g)h}{E_2} \right), \quad (4)$$

where m is the vehicle mass in kg, L is the wheelbase in m, L_1 is the distance of the CoG and the front axle in m, a_x, a_y are the X and Y direction accelerations in m/s^2 , g is the gravitational acceleration in m/s^2 , E_2 is the rear track width in m, and h is the height of the CoG in m.

Equation (5) summarizes forces acting on the vehicle chassis:

$$m\ddot{x} = \sum_{i=1}^4 F_i + F_{z_{rear}} + F_{z_{front}} \quad (5)$$

Unity 3D does not support CRG files, so they must be processed before importing. OpenCRG has a C-Application Programming Interface (C-API) for Matlab, which can load the CRG file 3D data and create the surface contained in it. With the help of an addon called surf2stl, we can create a stereolithography (STL) file from the chosen surface. The last step to import the road surface to Unity is to convert the STL file into another 3D format called OBJ. It can be done with any CAD software, which supports OBJ files or directly using an online converter.

Unity 3D can provide a gorgeous visualization and a platform to implement any custom model for vehicle dynamics.

It is possible to inject the ego vehicle into the simulation in real-time with the help of its localization sensors (e.g., differential GPS), creating a digital-twin in real-time (Varga et al., 2020). Offline trajectory logs can be used to recreate the test scenario too.

With the recorded parameters, it is easy to calculate every necessary vehicle state like accelerations to support the implemented vehicle dynamics functionality of Unity 3D. Moving the digital twin by GPS positions means locking some degrees of freedom like X and Y coordinates and the Z -axis angle, which provides the vehicle's heading. This obviously affects the virtual vehicle dynamics and limits it to vertical dynamics, roll, and pitch. The biggest impact is on the pitch and roll dynamics because their magnitude will be corrupted. Nonetheless, with Unity's help and the genetic algorithm's characteristics, virtual cars' realistic motion can be rendered and can provide valid vertical dynamics.

The second scenario was run in Unity 3D with the custom dynamics model and its results are provided in Fig. 10.

This model produced a higher frequency oscillation than the measurement and the results from Carmaker. The magnitude of the accelerations is quite similar to both other measurements. The mean squared error of the result in contrast to the real measurement is $MSE_{U3D} = 2.1165 \text{ m/s}^2$.

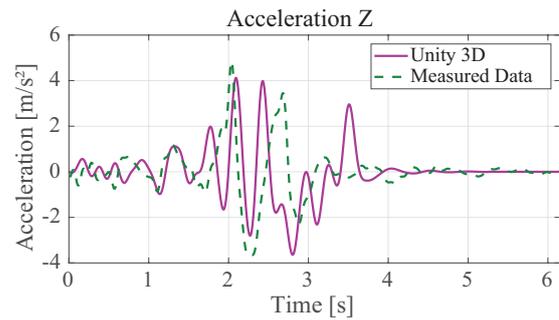


Fig. 10 Unity 3D with CRG excitation

5.4 Spectral density

Another way to analyze the results is to compare real measurements and simulations in the frequency domain with spectral density. Fig. 11 depicts the real measurement's spectral density and the simulations, which were executed on the CRG road surface. The range of frequency is between -10 and 10 Hz. The most conspicuous difference in the results is between the maximum amplitudes. The real measurement shows the highest amplitudes, while the lowest one belongs to the simulation in Unity 3D. The results are realistic because the road excitation belongs to this frequency range and simulations show quite accurate results.

6 Conclusions

The acquired results show that the method provided by this paper has its *raison d'être*. It is an optimal way to gather the required information in any use case and the results demonstrated the accuracy of the method. It provides parameters about a real vehicle without the need for expensive and time-consuming measurements, which can be used to recreate a real vehicle's suspension virtually, and it opens up the opportunity to use it in cases like VIL, SCIL, and systems using mixed-reality or even in the creation of a digital twin. With the help of this method, adequate vehicle dynamics visualization can be created,

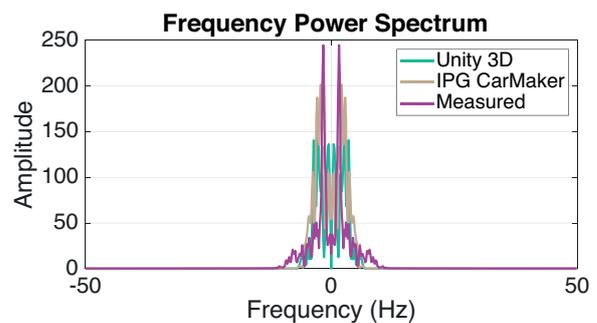


Fig. 11 Power Spectrum

and the parameters can be used for vehicle dynamics simulations. Besides, it can contribute to perception sensor tests where roll and pitch motions are significant. It can be concluded that a simplistic suspension model can sufficiently reproduce vertical vehicle dynamics, that are adequate for simple test scenarios and visualization purposes.

It can also be a useful tool even at vehicle test tracks (e.g., ZalaZone in Hungary, (Németh et al., 2019)) where mixed-reality testing is expected and engineers need a lot of accurate data to recreate their scenarios in the virtual versions of the test track and test scenarios. In the future, complex virtual suspension models like MacPherson can be implemented in both software to simulate oscillations

more precisely and provide a more accurate validation method. If more precision is needed, the genetic algorithm's settings can be fine-tuned more as a future job.

Acknowledgement

The research reported in this paper and carried out at BME has been supported by the NRDIFund (TKP2020 IES, Grant No. BME-IE-MIFM) based on the charter of bolster issued by the NRDIFund Office under the auspices of the Ministry for Innovation and Technology. Supported by the ÚNKP-20-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation fund.

References

- Barsi, A., Potó, V., Tihanyi, V. (2018) "Creating OpenCRG Road Surface Model from Terrestrial Laser Scanning Data for Autonomous Vehicles", In: Jármai, K., Bolló, B. (eds.) *Vehicle and Automotive Engineering 2*, Springer, Cham, Switzerland, pp. 361–369, https://doi.org/10.1007/978-3-319-75677-6_30
- Bosch Sensortec GmbH "BMI160 Small, low power inertial measurement unit", [online] Available at: <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmi160-ds000.pdf> [Accessed: 24 October 2020]
- Butenuth, M., Kallweit, R., Prescher, P. (2017) "Vehicle-in-the-Loop Real-world Vehicle Tests Combined with Virtual Scenarios", *ATZ Worldwide*, 119(9), pp. 52–55. <https://doi.org/10.1007/s38311-017-0082-4>
- Fehér, Á., Aradi, S., Bécsi, T. (2020a) "Fast Prototype Framework for Deep Reinforcement Learning-based Trajectory Planner", *Periodica Polytechnica Transportation Engineering*, 48(4), pp. 307–312. <https://doi.org/10.3311/PPtr.15837>
- Fehér, Á., Aradi, S., Bécsi, T. (2020b) "Hierarchical Evasive Path Planning Using Reinforcement Learning and Model Predictive Control", *IEEE Access*, 8, pp. 187470–187482. <https://doi.org/10.1109/ACCESS.2020.3031037>
- Hemati, A., Shooshtari, A. (2019) "Suspension damping optimization using genetic algorithms", *International Journal of Automotive Engineering and Technologies*, 8(4), pp. 178–185. <https://doi.org/10.18245/ijaet.531810>
- Herreros, A., Baeyens, E., Perán, J. R., Melgar, A. (2002) "Parameter Identification of a Car Suspension System Using Non-intrusive Signals", *IFAC Proceedings Volumes*, 35(1), pp. 427–432. <https://doi.org/10.3182/20020721-6-es-1901.01539>
- IPG Automotive GmbH "CarMaker: Virtual testing of automobiles and light-duty vehicles", [online] Available at: <https://ipg-automotive.com/products-services/simulation-software/carmaker/> [Accessed: 13 October 2020]
- Jiang, K., Pavelescu, A., Victorino, A., Charara, A. (2014) "Estimation of vehicle's vertical and lateral tire forces considering road angle and road irregularity", In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, pp. 342–347. <https://doi.org/10.1109/ITSC.2014.6957714>
- Klück, F., Zimmermann, F., Wotawa, F., Nica, M. (2019) "Genetic Algorithm-Based Test Parameter Optimization for ADAS System Testing", In: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), Sofia, Bulgaria, pp. 418–425. <https://doi.org/10.1109/QRS.2019.00058>
- Ma, Z., Qiao, Y., Lee, B., Fallon, E. (2013) "Experimental evaluation of mobile phone sensors", In: 24th IET Irish Signals and Systems Conference (ISSC 2013), Letterkenny, Ireland, Article number: 49. <https://doi.org/10.1049/ic.2013.0047>
- Madni, A. M., Madni, C. C., Lucero, S. D. (2019) "Leveraging Digital Twin Technology in Model-Based Systems Engineering", *Systems*, 7(1), Article number: 7. <https://doi.org/10.3390/systems7010007>
- Mitra, A. C., Desai, G. J., Patwardhan, S. R., Shirke, P. H., Kurne, W. M. H., Banerjee, N. (2016) "Optimization of passive vehicle suspension system by genetic algorithm", *Procedia Engineering*, 144, pp. 1158–1166. <https://doi.org/10.1016/j.proeng.2016.05.087>
- Németh, H., Hány, A., Szalay, Z., Tihanyi, V., Tóth, B. (2019) "Proving Ground Test Scenarios in Mixed Virtual and Real Environment for Highly Automated Driving", In: Proff, H. (ed.) *Mobilität in Zeiten der Veränderung*, Springer Gabler, Wiesbaden, Germany, pp. 199–210. https://doi.org/10.1007/978-3-658-26107-8_15
- Pacejka, H. (2005) "Tire and vehicle dynamics", Butterworth-Heinemann, Warrendale, PA, USA.
- Potó, V., Csepinszky, A., Barsi, Á. (2018) "Representing Road Related Laserscanned Data in Curved Regular Grid: A Support to Autonomous Vehicles", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42(2), pp. 917–921. <https://doi.org/10.5194/isprs-archives-xlii-2-917-2018>
- Song, R., Horridge, P., Pemberton, S., Wetherall, J., Maskell, S., Ralph, J. (2019) "A Multi-Sensor Simulation Environment for Autonomous Cars", In: 2019 22th International Conference on Information Fusion (FUSION), Ottawa, ON, Canada, pp. 1–7.

- Szalai, M., Varga, B., Tettamanti, T., Tihanyi, V. (2020) "Mixed reality test environment for autonomous cars using unity 3D and SUMO", In: 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, pp. 73–78.
<https://doi.org/10.1109/sami48414.2020.9108745>
- Unity Technologies "Unity Real-Time 3D Development Platform", [online] Available at: <https://unity.com/> [Accessed: 01 November 2020]
- Varga, B., Szalai, M., Fehér, Á., Aradi, S., Tettamanti, T. (2020) "Mixed-reality Automotive Testing with SENSORIS", *Periodica Polytechnica Transportation Engineering*, 48(4), pp. 357–362.
<https://doi.org/10.3311/pptr.15851>
- Vetturi, D., Gadola, M., Manzo, L., Faglia, R. (1996) "Genetic Algorithm for Tyre Model Identification in Automotive Dynamics Studies", In: *The 29th ISATA2 International Symposium on Automotive Technology and Automation*, Florence, Italy, 1996, pp. 24–31.