Paper Number: EAEC2017_A33

VIRTUAL SENSING ON AUTOMOTIVE EMBEDDED HETEROGENEOUS PLATFORMS

Antonio J. Rodriguez (University of A Coruña, Spain), **Roland Pastorino** (Siemens Industry Software NV, Belgium), **Miguel Á. Naya** (University of A Coruña, Spain), **Emilio Sanjurjo** (University of A Coruña, Spain),

Keywords: multibody dynamics, *real-time*, heterogeneous processor, FPGA, state estimation

Abstract

For the last decades, the automotive industry has been improving the ride & handling and the safety of vehicles. Many control strategies have been developed for this purpose. They make use of multiple sensors and actuators and are applied to different elements of the vehicle, such as (semi-)active suspension, electric power steering or brake assistance systems. These strategies require ever more complex algorithms, where more computational power and more sensor information are needed. New generation Electronic Control Units (ECUs) are in development to satisfy the demand of higher computational power in the automotive embedded platforms. However, to retrieve the required sensor information often implies the use of expensive sensors and, in some cases, the sensor information is not at all available. In the latter, model-based virtual sensors can be used as an alternative sensing technique. Most of the virtual sensors that are used nowadays in vehicles are based on analytical models because of its high computational efficiency. However, the amount of information can be increased by using multibody models. In conventional ECUs, the implementation of multibody models for *real-time* estimation is not possible due to the limited available computational power. Nevertheless, new generation ECUs offers the opportunity of executing state observers based on multibody models *on-board* and in *real-time*. The aim of this work is to implement multibody models on these ECUs in order to explore the performance that can be achieved.

Introduction

Multiple solutions have been developed by the automotive industry to deal with the goal of improving the ride & handling and the safety. Some are meant to assist the driver by means of control strategies, while others are focused in acquiring sensor data during vehicle testing for a high quality dynamics analysis.

In the field of control strategies, active systems together with Advanced Driver Assistance Systems (ADAS) are widely used. Active systems can be: powertrain control, to allow different driving modes depending on the demands of the driver; active suspensions, to modify the behavior of the suspension based on the road profile, giving higher ride comfort; electric power steering (EPS), to help the driver while steering the vehicle; brake assistance systems, to provide increased security during braking maneuvers. These active systems are all working under different control algorithms fed by sensor information. The evolution towards more complex control algorithms is linked to the information available in the vehicle. Model Predictive Control (MPC) algorithms are part of these new generation control strategies that can be used to improve the behavior of active systems. They however require full-state availability, which means that more sensor information than for conventional control algorithms is necessary. Virtual sensing appears therefore as a viable alternative to physical sensors to reduce solution cost and complexity. On the other hand, ADAS systems use information from 3D environment sensors (radar, lidar, cameras and ultrasonic sensors) to implement algorithms based on computer vision, machine learning or deep learning, for what high computational power is required.

For vehicle dynamics analysis, the objective to acquire relevant data during vehicle test drives. For acquiring part of this data, expensive and difficult-to-place sensors are required. The use of virtual sensors as an extension of a data acquisition systems is an alternative to collect more data without having to install additional sensors. The gathered data could then be used to analyze the vehicle dynamics and improve the ride & handling of the vehicle. As in the previous case, more computational power will be needed.

In order to implement the aforementioned control solutions, each vehicle has several ECUs. The number of ECUs is increasing, with luxury cars that can exceed the 100 processors. It is common to have one ECU for each control action that is taken in the vehicle, from simple task such as seat regulation, lights control or air conditioning to more complex tasks like the ones required for the engine, the suspension, the steering or the powertrain, for what advanced ECUs are needed. This situation leads to a vehicle with a very complex network and an important increment of weight.

As a solution to these problems, new generation ECUs are starting to be used in commercial vehicles. These ECUs offer an increase in computational power with respect to the conventional ECUs, and its use in commercial vehicles is an opportunity for reducing the number of *on-board* ECUs, and for implementing more complex control strategies, fed by virtual sensors based on multibody models.

In the next sections, the way virtual sensors are implemented nowadays will be presented, followed by an explanation on why new generation ECUs offer more computational power than conventional ECUs. Then, the implementation of state observers based on multibody models will be discussed, focusing specially on the coordinates and observer selection regarding to the coordinates and observer selection. Preliminary research results on achieved computational performance and multibody selection will be discussed in the next sections. Finally, conclusions and future work will be presented.

State of the art

As explained in the previous section, control strategies are programmed in the vehicle ECUs for improving Ride and Safety. Most of these ECUs need information from multiple sensors placed of the vehicle to get all the information required to execute their control actions. However, in some cases, this information is not available or imposes the use of expensive sensors. As an alternative, a state observer based on a model of the vehicle (or part of it) and a reduced number of sensors can be used to create virtual sensors [1, 2, 3] for the missing information required by the ECU.

There exist multiple use-cases for estimation algorithms in *real-time* applications, such as the control algorithms for active suspension system [4]. For enabling the control in the suspension, multiple sensors are placed on the suspension itself, such as accelerometers, to gather information of the real situation of the suspension. The data from the sensors is sent to the ECU, where the state observer based on an analytical model of the suspension is executed and returns information from the virtual sensors. Once that the information of physical and virtual sensors is acquired, the control algorithm sends the control action to the actuator of the suspension to adapt the suspension system behavior to the road profile, thus increasing the ride comfort.

With respect to the vehicle dynamics analysis, the work presented in [5], shows the possibility of estimating the forces in the knuckle of the vehicle. With the data collected by strain gauges and accelerometers placed on the knuckle, a finite element model of the knuckle combined with a Kalman filter can be fed with enough information to estimate the forces endured by the knuckle. After the estimation, these forces can be used for the knuckle durability analysis. To acquire the same data with physical sensors, the cost of the setup would increase, as well as the sensor installation cost. Therefore, with virtual sensing, the number and cost of the sensors needed for system analysis can be reduced.

As the evolution of control strategies tends to require more sensor information, the use of virtual sensors appears to be an adequate solution to avoid the high costs and difficulties derived from installing more physical sensors. The virtual sensors implemented in the last years, are based on analytical models due to the limitations of computational power on ECUs. Therefore, it was not possible to implement virtual sensors based on multibody models in *real-time* instead of analytical models, despite the opportunity of acquiring data from more (virtual) sensors, as a direct consequence of having more detailed models. Thus, the challenge is to execute the state observer and the multibody model of the vehicle in *real-time* on the commercial vehicle ECUs. Several studies have developed models of some parts of the vehicle or simplified models of it with state observers for control applications [6-7]. In [8] the multibody model of a complete vehicle with a state observer is simulated with good accuracy, but the computational complexity of the simulation was too high to be able to simulate it in *real-time*. In [9], an indirect state observer is presented and the simulation of a multibody model and its state observer reached *real-time* in an *on-board* CPU. This strategy is promising but the computing hardware used in this research does not correspond to the computational power available in ECUs of commercial vehicles. As mentioned before, these ECUs have substantially less computational performance than CPU of commodity PCs. However, new generation ECUs have the computational power required for executing virtual sensors based on multibody models.

New generation ECUs are based on heterogeneous processors. A heterogeneous processor has one chip that embeds two different processors. It is also known as System-On-Chip (SoC). In the case of ECUs, it embeds an *ARM* processor and a Field Programmable Gate Array (FPGA) co-processor. Using both processors in the same chip allows reducing the communication overhead between the two processors. Therefore, higher computational power than separated can be achieved.

ARM processors are based on a RISC (Reduced Instruction Set Computer) architecture, with a set of attributes allowing them to have a better cost, power consumption and heat dissipation than processors based on CISC (Complex Instruction Set Computer) architecture. These advantages make *ARM* processors suitable for embedded applications as smartphones or automotive ECUs.

On the other hand, an FPGA is a set of wires, logic gates and registers that can be combined with total freedom to build a dedicated computer unit for a specific application. FPGAs are also known as hardware accelerators: the code placed in the FPGA can outperform the same code in a CPU. However, each FPGA has limited resources, which means that a limited amount of code can be programmed on the FPGA. Also, FPGA code is programmed in hardware language, *VHDL*, which is significantly different from the conventional programming languages. Nevertheless, the available resources change depending on the model, and FPGA manufacturers embed ever more resources in FPGAs for heterogeneous processors. With respect to the programming language, FPGA manufacturers offer their own software for helping the designer when programming the FPGA, giving all the tools to translate the code from C/C++ to *VHDL* and to generate the files needed to program the hardware.

In this work, the Zynq7020 SoC from *Xilinx* was used. It includes an *ARM Cortex A-9* processor, more powerful than conventional ECUs, with a floating point unit for providing high precision for solving the multibody dynamics with no additional computational cost, and a FPGA *Artix-7* to complement the *ARM* processor, giving the opportunity of accelerate the task which consumes more time on the algorithm, so the efficiency can be improved.

Automotive multibody models

As explained in previous Sections, the use of a state observer based on multibody models provides more information via virtual sensors. As new generation ECUs offer higher computational power than conventional ECUs, the implementation of multibody models in these platforms needs to be studied. As an initial approach, a 3D multibody model of a quarter-car will be tested.

The multibody model developed for the 3D quarter-car model (figure 1.A) was defined in natural coordinates. This type of coordinates (see [10]) are easy to implement, at expenses of a higher number of variables which normally leads to higher computational times. As an alternative, in [11], an approach for the use of relative coordinates is presented. Using these coordinates, the number of variables is substantially reduced. Although using these coordinates increases the complexity of the model, the final computational time is lower than with natural coordinates for the case of models of high size as a full-car. Therefore, the multibody model of the full-car (figure 1.B) is developed using relative coordinates. Due to the higher performance of relative coordinates, their use seems to be more promising than natural coordinates for achieving *real-time*. Thus, the multibody formulation for relative coordinates used in this work will be explained in the next Section.

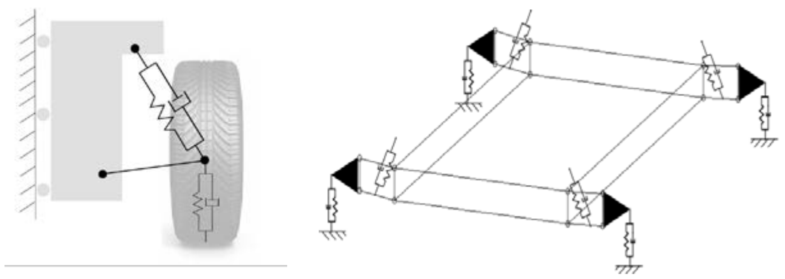

Figure 1. A) Multibody model of a quarter-car. B) Multibody model of a full-car.

Multibody formulations for relative coordinates

Modelling in relative coordinates means using a reduced set of variables for defining the dynamics of the model. The main disadvantage of using relative coordinates is the complexity for defining the dynamic equations of motion. As a solution, in [11], the semi-recursive formulation was presented.

The semi-recursive method defines a double set of coordinates in the modeling: six Cartesian coordinates (three translations plus three rotations) for each body, and the relative coordinates of the whole mechanism. The dynamic equations are defined in the Cartesian coordinates (or body coordinates) and, then, a velocity projection is used to obtain a set of equations in the relative coordinates. However, when the relative coordinates are dependent (a common situation in automotive applications), constraint equations should be imposed. In (1), the dynamic equations of motion using the index-3 augmented Lagrangian formulation [12] are presented.

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\alpha}\mathbf{\Phi} + \mathbf{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\lambda}^{*} = \mathbf{Q} \tag{1}$$

where **z** are the relative coordinates, **M** is the mass matrix of the mechanism expressed in terms of the relative coordinates, **Φ** is the constraint vector, $\mathbf{\Phi_z}$ is the Jacobian matrix of the constraints, **α** is the penalty factor, **Q** is the vector of applied and velocity-dependent forces, and $\boldsymbol{\lambda}^{*}$ is the vector of Lagrange multipliers.

As mentioned above, expressing the terms of equation (1) in relative coordinates is not a trivial task. However, using body coordinates, the dynamic terms **M** and **Q** can be more easily expressed. A relationship between both coordinates should be imposed therefore to obtain **M** and **Q** in relative coordinates. A matrix **R** can be defined so that the following relationship stands,

$$\mathbf{Z} = \mathbf{R}\dot{\mathbf{z}} \tag{2}$$

where **Z** are the body coordinates. If this relation is applied to the dynamic equation of motion,

$$\mathbf{R}^{\mathbf{t}}\bar{\mathbf{M}}\mathbf{R}\ddot{\mathbf{z}} = \mathbf{R}^{\mathbf{t}}(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}) \tag{3}$$

which means that the mass matrix and the vector of applied and velocity-dependent forces of the model expressed in relative coordinates become,

$$\mathbf{M} = \mathbf{R}^{\mathbf{t}}\bar{\mathbf{M}}\mathbf{R} \tag{4}$$

$$\mathbf{Q} = \mathbf{R}^{\mathbf{t}}(\bar{\mathbf{Q}} - \bar{\mathbf{M}}\dot{\mathbf{R}}\dot{\mathbf{z}}) \tag{5}$$

Once that the terms of (3) are known, it is possible to integrate the equations and solve the dynamics of the model at a certain time step. As integration scheme, the implicit single-step trapezoidal rule has been adopted. The corresponding difference equations in velocities and accelerations are:

$$\dot{\mathbf{z}}_{\mathbf{n+1}} = \frac{2}{\Delta t}\mathbf{z}_{\mathbf{n+1}} + \hat{\dot{\mathbf{z}}}_{\mathbf{n}} \quad \text{with} \quad \hat{\dot{\mathbf{z}}}_{\mathbf{n}} = -(\frac{2}{\Delta t}\mathbf{z}_{\mathbf{n}} + \dot{\mathbf{z}}_{\mathbf{n}}) \tag{6}$$

$$\ddot{\mathbf{z}}_{\mathbf{n+1}} = \frac{2}{\Delta t^2}\mathbf{z}_{\mathbf{n+1}} + \hat{\ddot{\mathbf{z}}}_{\mathbf{n}} \quad \text{with} \quad \hat{\ddot{\mathbf{z}}}_{\mathbf{n}} = -(\frac{2}{\Delta t^2}\mathbf{z}_{\mathbf{n}} + \frac{2}{\Delta t}\dot{\mathbf{z}}_{\mathbf{n}} + \ddot{\mathbf{z}}_{\mathbf{n}}) \tag{7}$$

being Δt the time-step. Dynamic equilibrium can be established at time-step n+1 by introducing difference equations (6) and (7) into (1), leading to a nonlinear system of equations in which the positions at time-step n+1 are the unknowns,

$$\mathrm{f}(\mathbf{z}_{\mathbf{n+1}}) = 0 \tag{8}$$

Such a system can be solved through the Newton-Raphson iterative procedure, being the residual vector,

$$\mathrm{f}(\mathbf{z}) = \frac{\Delta t^2}{4}(\mathbf{M}\ddot{\mathbf{z}} + \mathbf{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\alpha}\mathbf{\Phi} + \mathbf{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\lambda}^{*} - \mathbf{Q}) \tag{9}$$

And the approximated tangent matrix,

$$\frac{\partial \mathrm{f}(\mathbf{z})}{\partial \mathbf{z}} = \mathbf{M} + \frac{\Delta t^2}{4}(\mathbf{\Phi}_{\mathbf{z}}^{\mathbf{t}}\boldsymbol{\alpha}\mathbf{\Phi}) \tag{10}$$

This procedure yields a set of positions that not only satisfies the equation of motion (1), but also the constraint conditions. Once that the positions are obtained, the velocities and accelerations should be calculated for the time-step n+1. Thus, the motion of the model is determined for each time-step.

Results

In order to evaluate the benefits of using the heterogeneous processors, different simulations were executed for the quarter-car and full-car model. Each simulation was based on the multibody model running over a flat ground passing over a 10 cm bump. The total simulation time was of 10 s, with an integration time step of 4 ms.

Concerning the implementation of the multibody models in the heterogeneous processor, there exist many possibilities for deploying the code on the *ARM* processor and the FPGA co-processor. Nevertheless, the limited resources of the FPGA imply that the selection of the code which will be implemented on this processor is not obvious. Therefore, different configurations for each simulation will be considered in order to determine which configuration gives better performance. A simulation made with the full multibody model in the *ARM* processor is taken as a reference for making a time comparison with the implementations that offload code to the FPGA.

<u>Quarter-car simulation</u>

The quarter-car model represents the suspension of a vehicle. Therefore, it is a small model compared to a full-car model, and it is suitable for getting preliminary results of using the FPGA for accelerating the simulation. The characteristics of this model are shown in table 1.

	SYSTEM	**FORMULATION**	**INTEGRATOR**	**ITERATOR**	**TIME STEP**	**TIME**
MULTIBODY MODEL	Quarter-car	Penalty	Trapezoidal rule	Newton-raphson	4 ms	10 s

Table 1. Quarter-car multibody model characteristics

The options taken for these simulations were mentioned in the previous section: executing the full model on the FPGA, offload the matrix multiplications to the FPGA or implement the matrix inversion on the FPGA. The results are show in table 2.

	ARM	**FPGA**	**TIME (S)**
REFERENCE	Full model	-	0.23
OPTION A	-	Full model	NOT ENOUGH RESOURCES
OPTION B	Model	All matrix multiplications	0.94
OPTION C	Model	All matrix inversions	0.02

Table 2. Results of the quarter-car model simulations

From table 2, it can be seen that the FPGA that the *Xilinx* ZYNQ-7020 embeds has not enough resources by far for implementing the code for the simulation of the quarter-car multibody model, and without any kind of code optimization. Nevertheless, this problem can be avoided using FPGA with more resources. The FPGA used in this work has 85k logic cells, while the top level FPGA of the Zynq7000 family has 444k logic cells. Obviously, this increment of resources is traduced into a cost increment.

Therefore, the code of the multibody model was split between the *ARM* processor and the FPGA. As a first option, the matrix multiplications were implemented on the FPGA, due to the simplicity of the algorithm. However, as shown in table 2, the execution time taken for the simulation is higher than the reference simulation, meaning that this option does not improve the performance of the simulation. This result might be surprising, as the FPGA is supposed to accelerate the code. This is due to the simplicity of the model, which results in small matrices and, therefore, the multiplication in the FPGA does not provide a significant reduction of time to compensate the required communication time, making the overall execution slower.

Thus, as a third option, the matrix inversion was implemented on the FPGA. This operation is involved in the calculation of the position increments of the model each iteration, which is the most time consuming operation of the simulation. As this operation is more complex than the matrix multiplication, the algorithm provided by *Xilinx* for the FPGA was implemented. With this implementation, the performance of the matrix operation in the FPGA is substantially higher than in the *ARM* processor. The communication times have therefore a lower impact.

Full-car simulation

In order to evaluate a problem of higher size than the quarter car model, a full-car multibody model was developed. This model was made for studying the vertical dynamics of the vehicle, thus it has 10 degrees of freedom (6 from the spatial movement of the chassis and 1 for each suspension). In table 3, a resume of the model is made.

	SYSTEM	**FORMULATION**	**INTEGRATOR**	**ITERATOR**	**TIME STEP**	**TIME**
MULTIBODY MODEL	Full-car	Semi-recursive ALI3P	Trapezoidal rule	Newton-raphson	4 ms	10 s

Table 3. Full-car multibody model characteristics

For the full-car model, the main option taken was to implement all the operations required for solving the system of equations that gives the position increments of the model each time step, instead of implementing just the matrix inversion, as in the quarter-car model. The implementation of the full model on the FPGA was also tested. In table 4, the results of these simulations are shown.

	ARM	**FPGA**	**TIME (S)**
REFERENCE	Full model	-	5.99
OPTION A	-	Full model	NOT ENOUGH RESOURCES
OPTION B	Model	Solve equation system	4.41

Table 4. Results of the full-car model simulations

The first result of these simulations is that implementing a multibody model entirely on this FPGA is not possible. The resources of the ZYNQ-7020 are not enough, and using a bigger FPGA increases the cost making it not worth. Nevertheless, as these platforms evolve, it could be possible in the future to have more resources at the same cost than nowadays.

Thus, the most interesting option is to implement the code that consumes more time during the simulation and, in the case of multibody simulations, it is the resolution of the position increments. For this operation, *Xilinx* does not provide an algorithm. Therefore, a Gauss-Jordan algorithm was implemented. The results in table 4 show that there is a reduction of time when executing this operation on the FPGA. However, the improvement is not as good as it could be expected from the results obtained in the quarter-car simulation.

The blueprint of the FPGA implementation of the Gauss-Jordan algorithm is shown in figure 2, where the clear blue represents the resources consumed for this implementation. For the full-car model in relative coordinates used in this work, the matrix size of the system is 26x26. As it can be seen, almost all the elements of the FPGA are employed. In case of using natural coordinates, the size of the system would be substantially higher and there would not be enough resources for implementing the Gauss-Jordan algorithm on the FPGA. In this situation, the system should be solved using matrix partitioning, leading to an increase of complexity and higher computational cost.

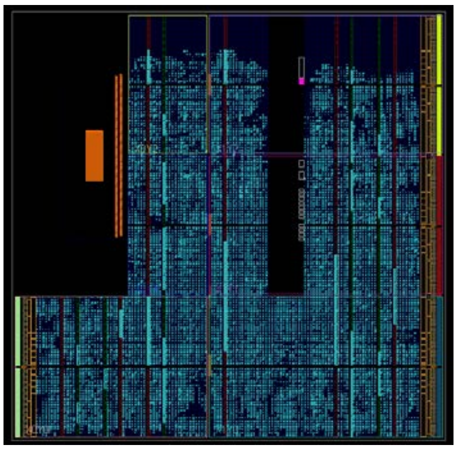

Figure 2. Blueprint of the Gauss-Jordan implementation on the FPGA used in this work

In [13-14], Gauss-Jordan implementations for FPGA are presented, showing an important reduction of time when compared with the same algorithm in a personal computer. Therefore, the actual implementation of the algorithm can be improved, increasing more the performance of the simulation when using the FPGA.

Conclusions and future work

The evolution of control algorithms for automotive applications is limited by the sensors available in commercial vehicles. In some cases, these sensors do not provide the most suitable data or cannot provide it at all. The use of virtual sensors in the automotive industry can alleviate the problem of gathering information of elements where the sensors cannot be placed or their installation implies a prohibitive cost. One of the main difficulties in implementing virtual sensors in a vehicle is related to the computational power of ECUs. Although virtual sensors based on analytical models can be implemented on conventional powerful ECUs, the information that can be gathered from these models is limited. In contrast, the use of multibody models can increase the number of virtual sensors and therefore the amount of information available.

The appearance of heterogeneous processors as processors for new generation ECUs means a high increase in computational power in ECUs. New generation ECUs embed an *ARM* processor and an FPGA co-processor. FPGAs are platforms that can be fully programmed for each specific application thus reaching higher performance than with conventional processors. One of the main disadvantages of FPGAs is the limitation on the amount of code that can be programmed on them. Therefore, the FPGA should ideally be used to accelerate the slower parts of the program that is being executed on the ECU.

In this paper, the implementation of multibody models in this new generation ECUs were made in order to evaluate the potential of these platforms. In an initial phase, a 3D multibody model of a quarter-car was executed, showing promising results. Then, a 3D full-car multibody model for analyzing the vertical dynamics of a vehicle was implemented. In this case, although there was an increase in the performance of the simulation, the results were not as good as in the initial approach.

As future work, new FPGA implementations will be tested in order to further increase the performance of the multibody simulations. Improving the multibody simulations will allow combining the multibody model with a state observer, in order to measure virtual sensors in *real-time* and to test the application in a real situation.

Acknowledgments

The support of the Spanish Ministry of Economy and Competitiveness (MINECO) under project TRA 2014-59435-P, cofinanced by the European Union through the EFRD program, is greatly acknowledged.

References

[1] J. Cuadrado, D. Dopico, J. A. Perez, R. Pastorino. Automotive observers based on multibody models and the extended Kalman filter. *Multibody System Dynamics*, Vol. 27, No. 1, pp. 3–19, 2012.

[2] Sanjurjo, E., Naya, M.Á., Blanco-Claraco, J.L., Torres-Moreno, J.L. Giménez-Fernández, A. . Nonlinear Dyn (2017) 88: 1935. doi:10.1007/s11071-017-3354-z

[3] Pastorino, R., Richiedei, D., Cuadrado, J., Trevisani, A. (2013). State Estimation Using Multibody Models and Nonlinear Kalman Filters. *International Journal of non-Linear Mechanics*, 53, 83-90.

[4] A. Chokor, R. Talj, A. Charara, H. Shraim, C. Francis. Active Suspension Control to Improve Passengers Comfort and Vehicle's Stability. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC).* Rio de Janeiro, Brazil, November 1-4, 2016.

[5] B. Cornelis, E. Risaliti, T. Tamarozzi, W. Desmet. Virtual Load Sensing Methods for Efficient Durability Testing and Customer Load Acquisition. *4th International Commercial Vehicle Technology Symposium (CVT),* At Kaiserslautern, Germany. March, 2016.

[6] T. Wenzel, K. Burnham, M. Blundell and R. Williams. Dual extended Kalman filter for vehicle state and parameter estimation, *Vehicle System Dynamics*, 44, 153-171, 2006.

[7] V. Sankaranarayanan et al. Observer Based Semi-Active Suspension Control Applied to a Light Commercial Vehicle.*2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.* September 4-7, 2007, Zurich, Switzerland.

[8] E. Sanjurjo, R. Pastorino, P. Gallo, M.A. Naya. Implementation Issues of an on Board Real-Time Multibody Model. *Proceedings of the 3rd Joint Int. Conference on Multibody System Dynamics (IMSD) and 7th Asian Conference on Multibody Dynamics* (ACMD

2014), 249-250. Busan, Korea, 2014.

[9] E. Sanjurjo, E. Sinigaglia, M.A. Naya. Multibody-based State Observer for Navigation Applications. *Proceedings of the 4th Joint Int. Conference on Multibody System Dynamics* (IMSD 2016), a179. Montreal, Canada, 2016-05.

[10] J. García de Jalón, E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: The Real Time Challenge*, Springer-Verlag, 1994.

[11] J. Cuadrado, D. Dopico, M. Gonzalez, M. A. Naya. A Combined Penalty and Recursive Real-Time Formulation for Multibody Dynamics. *Journal of Mechanical Design*. July 2004, vol. 126.

[12] J. Cuadrado, R. Gutiérrez, M. A. Naya, P. Morer. A Comparison in Terms of Accuracy and Efficiency Between a MBS Dynamic Formulation With Stress Analysis and a Non-linear FEA Code. *Int. J. Numer. Methods Eng.*, 51(9), pp. 1033-1052. 2001.

[13] J.P. David. Low latency and division free Gauss-Jordan solver in floating point arithmetic. *Journal of Parallel and Distributed Computing*. Volume 106, August 2017, Pages 185-193.

[14] M. Ziad, Y. Alkabani, M. El-Kharashi. On Hardware Solution of Dense Linear Systems via Gauss-Jordan Elimination. *Communications, Computers and Signal Processing (PACRIM), 2015 IEEE Pacific Rim Conference*. 24-26 Aug. 2015