Distributed control of a car suspension system

Mohamed El Mongi Ben Gaid, Arben Çela, Rémy Kocik

COSI - ESIEE - Cité Descartes - BP 99 - 2 Bd Blaise Pascal - F93162 Noisy-Le-Grand Cedex
{bengaidm,celaa,r.kocik}@esiee.fr

ABSTRACT

Active suspension control system are used in today cars because of their ability to manage the compromise between ride comfort and vehicle road-handling. They constitute a typical example of distributed control systems. In this paper, the ride controller part of an active suspension system is presented and evaluated, taking into account its distributed architecture. The simulations are realized with the Matlab/Simulink toolbox TrueTime, which allows the simulation of the controlled system, integrating simple models of its implementation (task execution, processor scheduling, network transmission...). We show, through a simulation example, how implementation related parameters can have a considerable impact on the robustness of the controlled system.

KEYWORDS

Active suspension control, simulation, network scheduling.

1 INTRODUCTION

Distributed embedded control architectures have become the first source of innovation and performance improvements in today cars. Actually, these architectures represent more than a quarter of the cost of manufacturing a car. As a consequence, a great effort has been made by car manufactures to improve their cost effectiveness [1]. Employing such architectures in control gives rise to new issues. In fact, using a distributed computer system in control is a source of delays and jitter which can degrade the control performance. Employing oversized computer architectures can solve these issues, but in the hard competition between the various manufactures, reducing costs is a crucial challenge.

Active suspension control systems are a typical example of distributed embedded control architectures. An active suspension consists of a spring, a shock absorber and a hydraulic actuator at each corner of the vehicle. Its role is to improve both driving comfort and road-holding by appropriately transmitting and filtering all forces between the body of the vehicle and the road. Controlling an active suspension system requires an amount of information which can be provided by a set of sensors situated in different locations in the vehicle. This information needs to be processed by one or more controllers in order to calculate the control forces that should be actuated by the four hydraulic actuators. Sensors, controllers and actuators usually communicate through a shared bus (the CAN is the most used one).

This paper describes a more realistic simulation of an active suspension system using the MATLAB/Simulink-based simulator TrueTime [2, 3], which allows the simulation of distributed real-time control systems, taking into account the effects of the execution of the control tasks and the data transmission on the controlled system dynamics. We show with a simple example, how the choice of parameters related to the implementation can have a considerable effect on the robustness of the whole control system.

2 THE SUSPENSION CONTROL SYSTEM

The simulated model (figure 1) was adopted from [4]. It consists of a seven degree-of-freedom system. In this model, the car body, or sprung mass, is free to heave, roll and pitch. In order to obtain a linear model, roll and pitch angles are assumed to be small. The suspension system connects the sprung mass to the four unsprung masses (front-left, front-right, rear-left and rear-right wheels), which are free to bounce vertically with respect to the sprung mass. The suspension system consists of a spring, a shock absorber and a hydraulic actuator at each corner. The shock absorbers are modelled as linear viscous dampers, and the tires are modelled as linear springs.

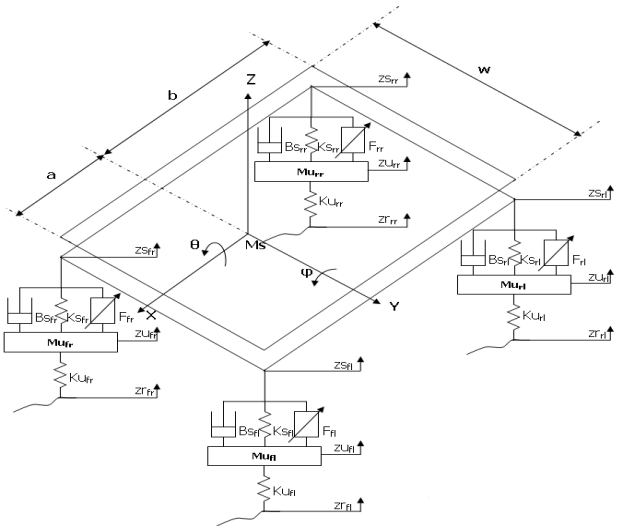

Figure 1. Model of the full vehicle

In order to describe this system, fourteen state variables need to be considered:

$x_1 = z$: heave position of the sprung mass
$x_2 = \dot{z}$: heave velocity of the sprung mass
$x_3 = \varphi$: pitch angle
$x_4 = \dot{\varphi}$: pitch angular velocity
$x_5 = \theta$: roll angle
$x_6 = \dot{\theta}$: roll angular velocity
$x_7 = zu_{fl}$: front-left unsprung mass height
$x_8 = \dot{z}u_{fl}$: front-left unsprung mass velocity
$x_9 = zu_{fr}$: front-right unsprung mass height
$x_{10} = \dot{z}u_{fr}$: front-right unsprung mass velocity
$x_{11} = zu_{rl}$: rear-left unsprung mass height
$x_{12} = \dot{z}u_{rl}$: rear-left unsprung mass velocity
$x_{13} = zu_{rr}$: rear-right unsprung mass height
$x_{14} = \dot{z}u_{rr}$: rear-right unsprung mass velocity

Applying a force-balance analysis to the model in figure 1, the state space equations can be derived from the equations of motion and are given by:

$$\dot{x}(t) = Ax(t) + Bu(t) + Ld(t) \tag{1}$$

$$y(t) = Cx(t) \tag{2}$$

where :
$x(t)$ is the state vector (14 variables)

$u(t)$ is the control vector : $u(t) = [F_{fl}, F_{fr}, F_{rl}, F_{rr}]^T$
$d(t)$ is the vector of road disturbance heights : $d(t) = [zr_{fl}, zr_{fr}, zr_{rl}, zr_{rr}]^T$
$y(t)$ is the output vector (7 variables) :
$y(t) = [z, \varphi, \theta, zs_{fl} - zu_{fl}, zs_{fr} - zu_{fr}, zs_{rl} - zu_{rl}, zs_{rr} - zu_{rr}]^T$

3 CONTROL DESIGN AND ARCHITECTURE

3.1 Active suspension control law

The control design for a vehicle's active suspension aims to maximize driving comfort (as measured by sprung mass accelerations) and safety (as measured by tire load variations) under packaging constraints (as measured by suspension deflections). However, comfort and safety are two conflicting criteras [6]. We adopt the control design methodology of [5], who divides the control design problem for a vehicle's active suspension into two sub-problems:

- The design of the ride controller, whose role is to improve ride comfort by isolating the sprung mass from road disturbances.
- The design of the attitude controller, responsible of maintaining load-levelling, performing convenient load distribution and controlling roll and pitch responses during vehicle maneuvers.

In this paper, we focus on the ride controller part of the suspension controller. The used ride controller is a linear quadratic regulator $u = -Gx$, the optimal gain matrix G is obtained such that the following cost function :

$$J = \int_0^\infty \left\{ \sum_{i=0}^{12} w_i y_i^2 + \sum_{j=0}^{4} c_j u_j^2 \right\} dt \tag{3}$$

is minimized, where w_i and c_j are weighting factors and :

y_1 : vertical acceleration of the sprung mass
y_2 : pitch angular acceleration of the sprung mass
y_3 : roll angular acceleration of the sprung mass
y_4 : sum of the suspension deflections at the four corners
y_5 : difference between the suspension deflections at the right- and left- hand sides
y_6 : difference between the suspension deflections at the front and rear of the vehicle
y_7 : difference between the suspension deflections at the diagonally opposite corners
y_8 : sum of the velocities of the unsprung masses
y_9 : difference between the velocities of the unsprung masses on the left- and right-hand sides
y_{10} : difference between the velocities of the unsprung masses at the front and rear of the vehicle
y_{11} : difference between the velocities of the unsprung masses at the diagonally opposite corners
y_{12} : wrap torque acting on the sprung mass

The state-space vector is obtained from the state observer :

$$\dot{\hat{x}} = A\hat{x} + Bu + K_o(y - C\hat{x}) \tag{4}$$

3.2 Simulated control architecture

Sensors and actuators are connected to computer nodes, according to their positioning in the vehicle. If some sensors and/or actuators are close, they are connected to the same computer node. The implementation architecture is described in figure 2 (left). It consists of 7 computer nodes, that communicate through a CAN network. They are distributed in the following way:

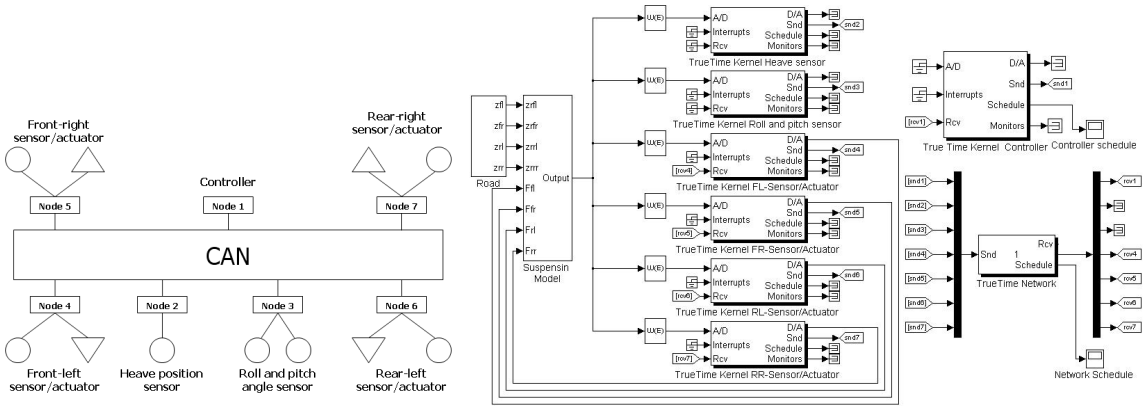

Figure 2. Simulated architecture. Left : schematic view - Right : implementation in TrueTime

- 1 computer node is responsible of the calculations of the control law **(Node 1)**. Messages sent from sensors to the controller node are stored in a shared data structure : the measurements vector. Each time the controller node receives a message from a sensor, it throws an interruption which updates the current measurements vector. The control task executed by the controller node is periodic.
- 2 computer nodes are connected to the sensors located in the center of the vehicle (**Nodes 2 and 3)**. They periodically read the values of heave, roll and pitch position and send them to the controller node.
- 4 computer nodes are placed at each corner **(Nodes 4 to 7**). They are connected to local sensors and actuators. Each node periodically reads the sensed values (local suspension deflection position), sends them to the controller using its network interface. Each node listens to the network, each time it receives a message from the controller, an interruption is generated and the associated actuator is updated.

The figure 2 (right) displays the implementation of this control architecture in the simulator TrueTime. This implementation uses the Simulink-based blocks : TrueTime Kernel and TrueTime network. The TrueTime Kernel is a Simulink block that simulates a computer integrating a real-time operating system and executing user defined tasks. TrueTime Network is a Simulink block than can be used in combination with a TrueTime Kernel block. The TrueTime Network block includes simple models of the most used networks. These models take into account both the medium access protocol (CSMA/CD, CSMA/AMP, TDMA...) and the packet transmission (only packet level simulation is supported).

4 SIMULATION RESULTS

In this section we first evaluate the performance of the designed active suspension and compare it to the passive suspension. Second we compare the traditional simulations with Matlab/Simulink (which assume a constant sampling period, no delays or jitter) to the simulations with TrueTime. The suspension system is evaluated by subjecting the right side of the vehicle to a "chunck hole" discrete road disturbance [5]. The "chunck hole" road disturbance, the heave, roll and pitch acceleration responses of the sprung mass and the network schedule are illustrated in figure 3, for a vehicle speed of 40 km/h.

The dash-dot blue line correspond to the passive suspension, the solid green line to the active suspension assuming an ideal implementation (like the classical simulations in Matlab/Simulink)

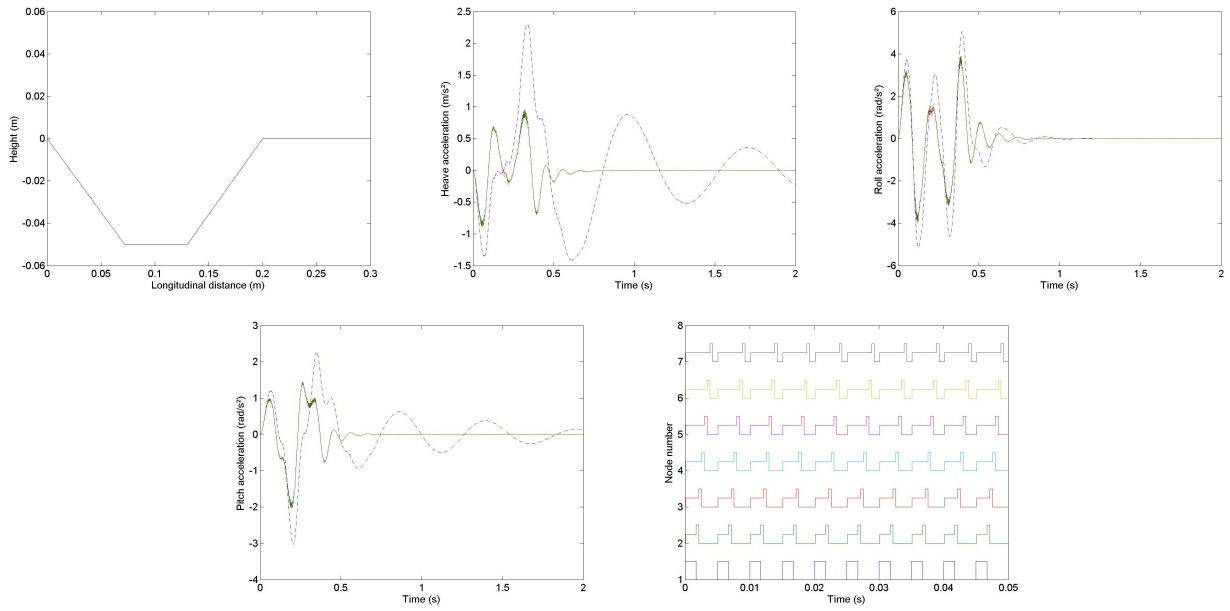

Figure 3. "Chunck hole" road disturbance, heave, roll and pitch accelerations of the sprung mass and network schedule. In the network schedule plot, a low signal means idle, a medium signal means waiting and a high signal means sending

and the dotted red line to the TrueTime simulation of the active suspension system. In the TrueTime simulations we assume that the control task takes 2 ms to calculate the control output, that the network uses the CSMA/CA arbitration, and have a bandwidth of 150000 bits/s and finally that the priorities of the messages are the same as the sending node (the **node 1** has the greatest priority and the **node 9** the smallest one).

The simulation results show that the active suspension induces an important improvement of the ride performance compared to the passive suspension (smaller and better damped accelerations). The simulation using an ideal implementation and those using the TrueTime model give approximately the same results. The delays due to the implementation (task execution, processor scheduling, network transmission and network scheduling) have a negligible impact on the control performance.

In the second simulation, we compare two TrueTime implementations of the active suspension control system under network overload conditions (a network bandwidth of 110000 bits/s). The two implementations (**Configuration 1** et **Configuration 2**) are identical, except that in **Configuration 2**, the messages sent by the controller node have the smallest priority whereas in **Configuration 1**, theses messages have the greatest priority. The simulation results are illustrated in figure 4. It can be seen that the suspension system of **Configuration 1** remains stable, with a degradation of the control performance whereas the suspension system of **Configuration 2** becomes instable. In **configuration 1**, the network overload causes the loss of the informations provided by the rear-right sensor and induces important delays in the informations provided by the rear-left sensor. In **configuration 2**, all sensor data are transmitted to the controller node within 5 ms (which is the sampling period of the system), but the control values sent to the actuatos suffer from important and increasing delays which cause the instability of the controlled system. The delays are increasing in this case because each time the controller tries to send a message, the previous message is not completely sent (as shown in figure 4). These results demonstrate the importance of the choice of the implementation parameters (scheduling policy, priorities...) when network resources are scarce. In fact, when the network is overloaded, it is preferable to loose a sensor data than the control values that should be actuated.

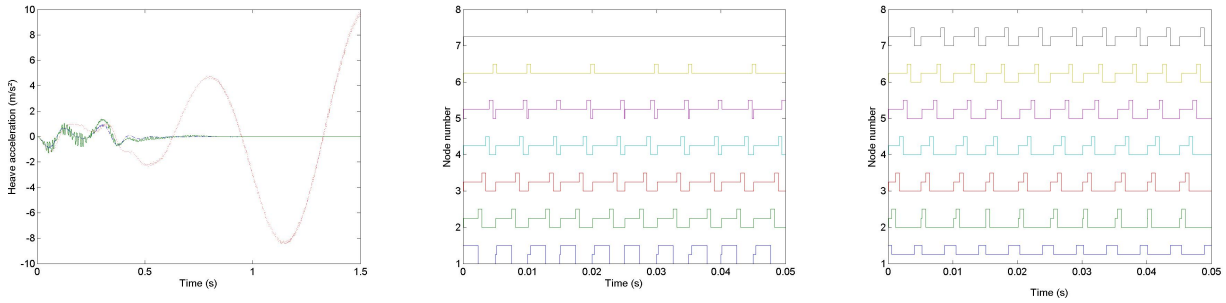

Figure 4. Heave acceleration response and network schedule for the configurations 1 and 2. In the left plot, the dash-dot blue line corresponds to the ideal implementation, the solid green line to the heave response of **Configuration 1** and the dotted red line to the heave response of **Configuration 2**

5 CONCLUSION

A distributed simulation of an active suspension system, incorporating models of the implementing architecture was presented. The simulations showed that the choice implementation parameters, especially those related to scheduling have an important impact on the robustness of the application in overload conditions. Those observations are the starting point of our future work, which aims at developing adaptive network scheduling algorithms, which take into account the states of both the network and the controlled system, to achieve optimal use of the available resources, especially in overloaded conditions.

REFERENCES

[1] ARTIST (2003). Hard Real-Time Development Environments. Report IST-2001-34820, Information Society Technologies.

[2] D. Henriksson, A. Cervin and K. E. Årzén (2003). TrueTime: Real-time Control System Simulation with MATLAB/Simulink. In *Proceedings of the Nordic MATLAB Conference*. Copenhagen, Denmark.

[3] D. Henriksson and A. Cervin (2003). TrueTime 1.13—Reference Manual. Technical report, Department of Automatic Control, Lund Institute of Technology.

[4] S. Ikenaga, F. L. Lewis, J. Campos and L. Davis (2000). Active Suspension Control of Ground Vehicle based on a Full-Vehicle Model. In *Proceedings of the American Control Conference (ACC)*. Chicago, USA.

[5] R. M Chalasani (1986). Ride performance potential of active suspension systems part II: Comprehensive analysis based on a full-car model. In *Proceedings of the Symposium on Simulation and Control of Ground Vehicles and Transportation Systems, ASME AMD*. Anaheim, CA.

[6] U. Rettig and O. von Stryk (2001). Numerical Optimal Control Strategies for Semi-Active Suspension with Electrorheological Fluid Dampers. In *Proceedings of the workshop: Fast solution of discretized optimization problems*. International Series on Numerical Mathematics (Birkhuser). 221–241.