COMPUTER-BASED DEVELOPMENT OF CONTROL STRATEGIES FOR GROUND VEHICLES

M.A. NAYA, J. CUADRADO
Escuela Politecnica Superior, Universidad de La Coruña, Ferrol, Spain

SYNOPSIS

During the last years, our group has worked on real-time formulations for the dynamics of multi-body systems. Now, in order to find out if such methods are suitable to address real industrial problems, we intend to develop control algorithms for a car on its computer model (virtual prototyping), and evaluate the performance of such controllers when implemented on the corresponding physical prototype. This paper addresses the first part of the work. Two maneuvers are to be considered: straight line and obstacle avoidance.
The computer model of the car has been written in Fortran language. Fuzzy logic has been chosen to design the control algorithms, which have been implemented on Matlab environment. Several alternatives to connect Fortran- and Matlab-based functions have been studied, concluding that the most appropriate election depends on the purpose being pursued: controller tuning or onboard use of an already tuned controller. Simulator capabilities have been given to the program by means of a realistic graphical output and game-type driving peripherals (steering wheel and pedals), so that comparison may be established between human and designed automatic control.

1 INTRODUCTION

During the last years, our group has worked on real-time formulations for the dynamics of multi-body systems[1,2]. As a result, a robust and efficient method has been developed[2]: an index-3 augmented Lagrangian formulation with projections of velocities and accelerations, which features natural coordinates for the modelling, the trapezoidal rule as numerical integrator and sparse matrix technology for calculations. The method has shown to be robust and accurate, successfully facing singular configurations, changing topologies and stiff systems, as well as efficient, achieving real-time performance on a conventional PC when

simulating the full model of a car vehicle undergoing rather violent manoeuvres, like stairs descent.

Now, in order to find out if such formulation is suitable to address real industrial problems, we intend to develop control algorithms for a car on its computer model (virtual prototyping), and evaluate the performance of such controllers when implemented on the corresponding physical prototype. Figure 1 shows the general diagram describing the mentioned objective.

The actual prototype has been built, and its virtual counterpart has been implemented on a computer through the mentioned dynamic formulation. The instrumentation of the former and the programming of the latter have been carried out in such a way that the inputs (actuators) and outputs (sensors) of the model and the physical prototype are exactly the same. Then, control algorithms can be designed and tested on the computer model of the car, until satisfying behaviour of the controller is achieved. If the simulator is accurate enough, the resulting control algorithms should also work properly when implemented on the actual car. Two manoeuvres are to be considered: straight line and obstacle avoidance.

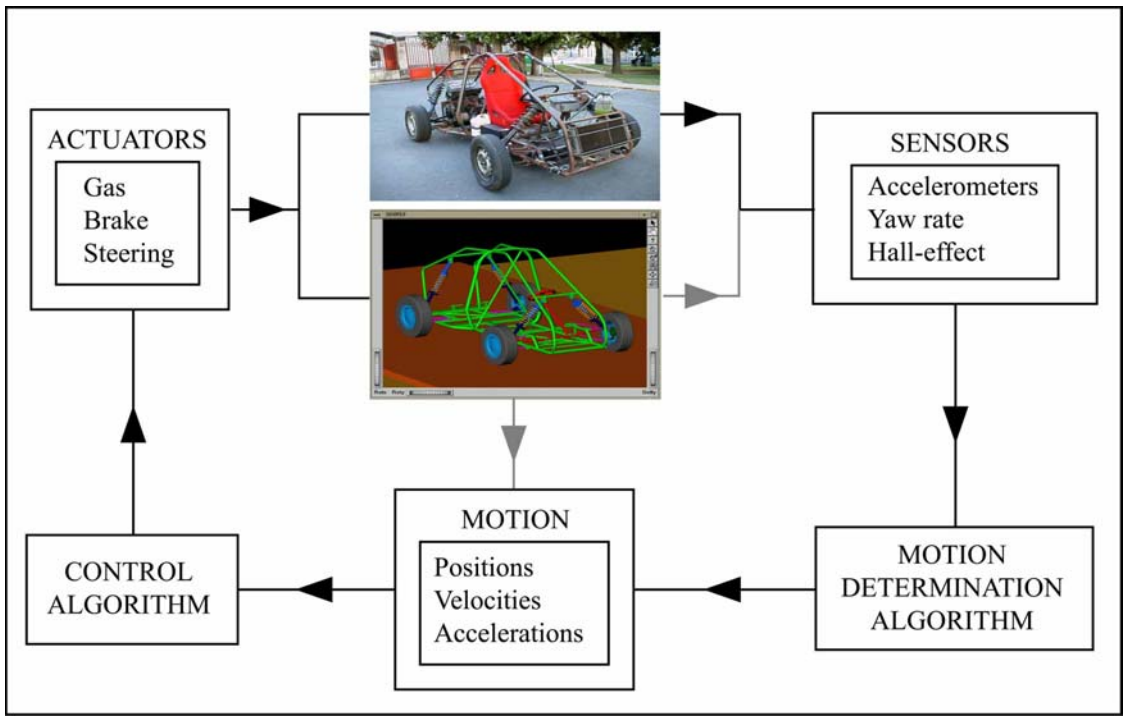

Fig. 1. General context of the work.

Many references can be found in the literature regarding the automatic control of car vehicles. Two good reviews of the state-of-the-art concerning the trajectory tracking problem were presented by Antos and Ambrosio[3], and Gordon et al[4]. Some other works have been focused on more specific aspects of the problem, like the so-called *intelligent cruise control* (ICC)[5,6], or the measurement of some kinematic magnitudes during the motion[7]. Conversely, the present work aims to address the problem from a more holistic and general point of view.

The paper is organized as follows: Section 2 focuses on the computational model of the car; Section 3 justifies the use of fuzzy logic in the present work; Section 4 explains the different alternatives available to connect the Fortran code containing the dynamics of the car with the Matlab functions implementing the fuzzy logic control algorithms, and points out their preferred contexts of application; Section 5 shows the comparison, for the two manoeuvres considered, between human and automatic control at simulation level; finally, Section 6 outlines the conclusions of the work.

2 COMPUTATIONAL MODEL OF THE CAR

The mathematical model of the car, illustrated in Figure 2 along with the physical prototype, has been carried out in natural coordinates[8]: 44 points, 7 unit vectors, 5 distances and 1 angle have been used as problem variables, leading to a total problem size of 159. Some few out of the mentioned variables have not been included for strict mechanical reasons, but rather for graphical representation requirements.

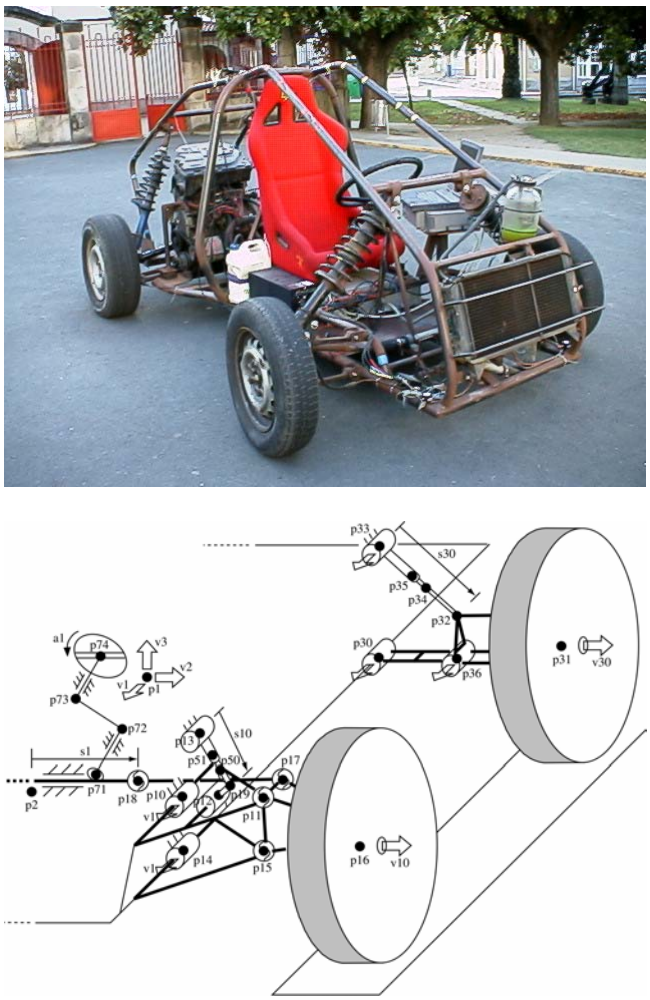

Fig. 2. a) The prototype; b) its model in natural coordinates.

The equations of motion have been derived by means of an index-3 augmented Lagrangian formulation with projections of velocities and accelerations[2]. The steering wheel is kinematically guided. Forces which deserve to be described are the following:

- Suspension forces: they have been considered through linear models of springs and dampers.
- Tire forces: lateral force and self-aligning torque have been introduced through the *magic formula*[9] with coefficients provided by the tire maker. The longitudinal effort has been neglected.
- Power transmission forces: from the engine relationships torque-speed and the gear ratios, both provided by the engine maker, the automatic gearing has been modelled. Then, for a certain value of the car velocity, the engine speed and, consequently, the engine torque, can be easily derived. The torque is applied on the rear wheels.
- Brake forces: the braking torque has been estimated from disk geometry[10], and applied on the four wheels.

A code that calculates the dynamics of the described model has been implemented in Fortran language, due to its high efficiency.

3 REASONS FOR THE USE OF FUZZY LOGIC

Designing a conventional controller, such as a proportional, integral and derivative (PID) controller, normally follows a standard procedure of modelling the plant, constructing a controller and evaluating the performance[11]. A complete ground vehicle is naturally a highly nonlinear system. Developing a model which preserves the nonlinear characteristics of the system, and is simple enough to represent the plant of the complete system, is not easy at all. In fact, it's common to resort to a simplified model, as the bicycle model, in order to design a controller for a whole vehicle.

Fuzzy control is knowledge-based control technology that can mimic human strategies to control complex systems[12]. Due to its capability of handling systems nonlinearity, this technique seems a good choice to control a ground vehicle, and has been used recently to control different parts of actual vehicles[13]. Reviewing the accomplishments reported on applications of fuzzy control to vehicle systems, a common feature was that natural language models were used to describe the control process[14], being similar for plants having similar implementation mechanisms. This characteristic of fuzzy control makes it possible to design a generic fuzzy controller for similar plants. Moreover, another valuable reason is that a reliable toolbox for fuzzy logic programming is provided by Matlab environment. This fact highly simplifies both the programming and tuning of the controller.

4 FORTRAN-MATLAB CONNECTION

In order to introduce the control on the prototype model, the Fortran code containing the dynamics of the car and the Matlab functions implementing the fuzzy logic control algorithms must be combined. Matlab always employs double-precision variables, so that the different types of Fortran variables must be converted to double-precision for Matlab compatibility. To connect Fortran and Matlab, two alternatives have been investigated: Matlab Engine and

MEX files. For both of them, compatibility between Matlab and the Fortran compiler is required.

Fortran-Matlab communication through Matlab Engine requires opening a communication channel from Fortran to Matlab. Matlab features several compilation functions which enable data transference between both languages, as well as executing functions in Matlab[15]. Then, when the Fortran program is run, a Matlab session is started, which implies some delay. Likewise, data transmission through the communication channel and the execution of Matlab instructions both slow down the program. In fact, to execute a Matlab function it must be written and executed on Matlab command window through the communication channel. Despite this fact, the simulation CPU-times obtained are kept moderate and, therefore, Matlab Engine can be considered as a suitable tool to design new control algorithms.

Matlab allows the user to write new functions by means of the so-called MEX files. Through this method, the user can write the whole program in Matlab language with the exception of bottleneck functions, which could be written in more efficient languages, like Fortran or C, and executed directly from Matlab as its own functions. A command library is available in order to communicate Matlab with the other programming language. The compilation is carried out in Matlab. Consequently, this option is just opposite to the previous one. In the present work, the whole Fortran program containing the dynamics of the car has been converted to a MEX file, which can be executed in Matlab, thus enabling the access to the functions of the fuzzy logic toolbox. The efficiency has shown to be a little lower than that obtained with the first option.

5 HUMAN VS AUTOMATIC CONTROL IN SIMULATION

The computational model of the car has been used to create a driving simulator, shown in Figure 3a, by combining the already mentioned vehicle dynamics code, along with a realistic graphical output and game-type driving peripherals (steering wheel and pedals), so that comparison may be established between human and designed automatic control.

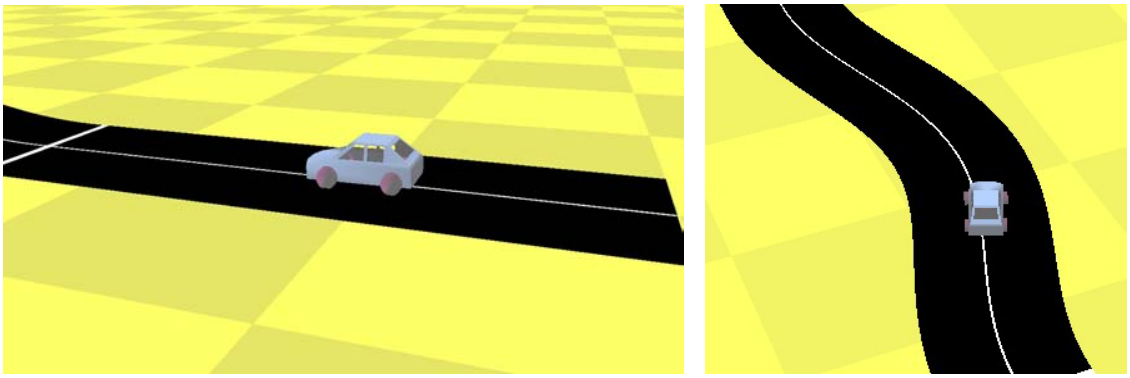

Fig. 3. Driving simulator: a) general view; b) first manoeuvre; c) second manoeuvre.

As said in the Introduction, two manoeuvres have been performed. In the first one, the car starts from rest, covers a distance of 20 m following a straight line, and stops. The maximum speed allowed is 5 m/s, but the time spent in the manoeuvre has not been limited. First, a human driver has carried out the manoeuvre. Figure 3b shows the corresponding simulator environment.

The control scheme for automatic driving has been represented in Figure 4. It receives position, velocity and acceleration of the car, and acts upon gas and brake. The values sent to both actuators range from 0 to 1, corresponding to null and maximum displacement of the respective devices. Simultaneous operation of gas and brake has been avoided. No error function has been used, but the position of the car. The total travelling distance has been divided into four intervals –start, taxiing, brake and stop–, and control rules have been specified for each interval. A fifth interval has been defined for the rest period after the arrival, in order to avoid an undesired behaviour in that phase.

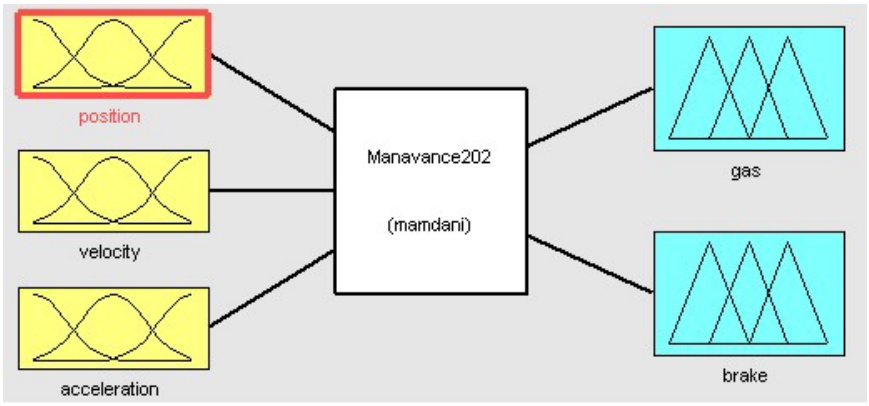

Fig. 4. Control diagram.

Figure 5 shows the comparison of the position history, gas actuation and brake actuation carried out by both the human driver and the controller. As it can be observed, the manoeuvre performed by the controller is cleaner. The controller does not need to make several approximations, as the human driver does. Actuation on both the gas and brake is more efficient when the car is automatically controlled, and less time is needed to complete the manoeuvre.

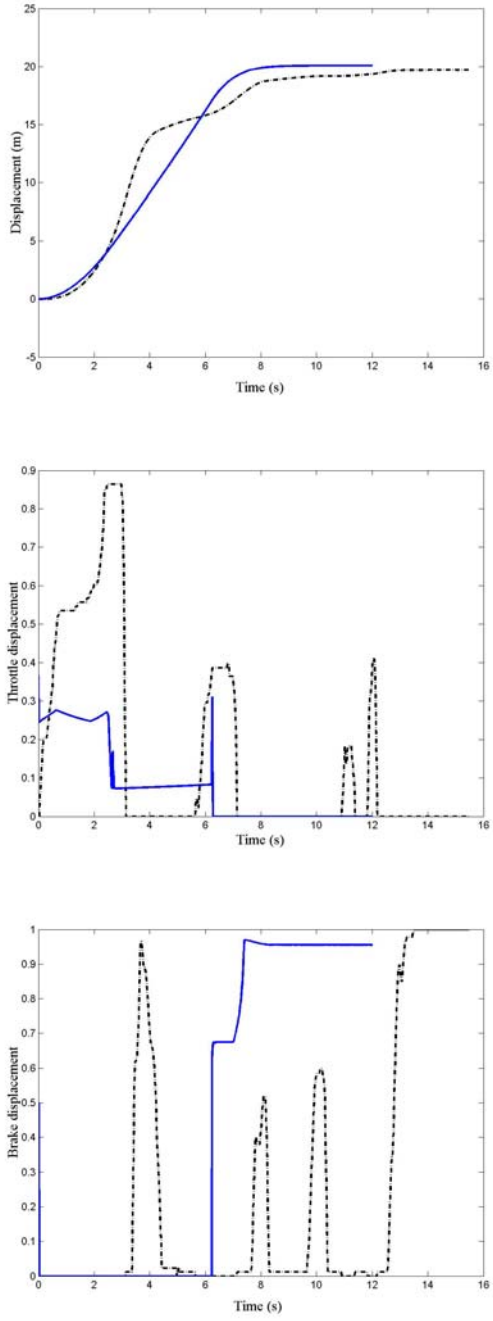

Fig. 5. Comparison between human driving (dashed) and automatic control for the first manoeuvre: a) position history; b) gas actuation; c) brake actuation.

The second manoeuvre consists in obstacle avoidance: starting from rest, the car covers an initial straight path of 20 m, then follows a full period (from peak to peak) of a sinoidal path of amplitude 1.75 m and, finally, must return to the straight line. The speed is kept under 8 m/s. First, a human driver has carried out the manoeuvre. Figure 3c shows the corresponding simulation environment.

For this second manoeuvre, the controls of both the steering and the gas-brake couple have been addressed separately, as illustrated in Figure 6. The pursued objective is that the controller slows down the car when the error in path tracking increases.

For steering control, two error functions have been defined. The first one is the position error of the vehicle at the current time. The second one aims to anticipate the behaviour of the car in the next instants. For this purpose, the tangent to the trajectory at the current point is compared with the tangent to the desired trajectory at a more advanced point. The anticipated distance depends on the car speed, being greater for larger speeds.

For gas-brake control both the velocity and acceleration of the car are taken into account, as well as the path tracking error. In this case, the error is calculated as the mean of the two errors used for the steering control.

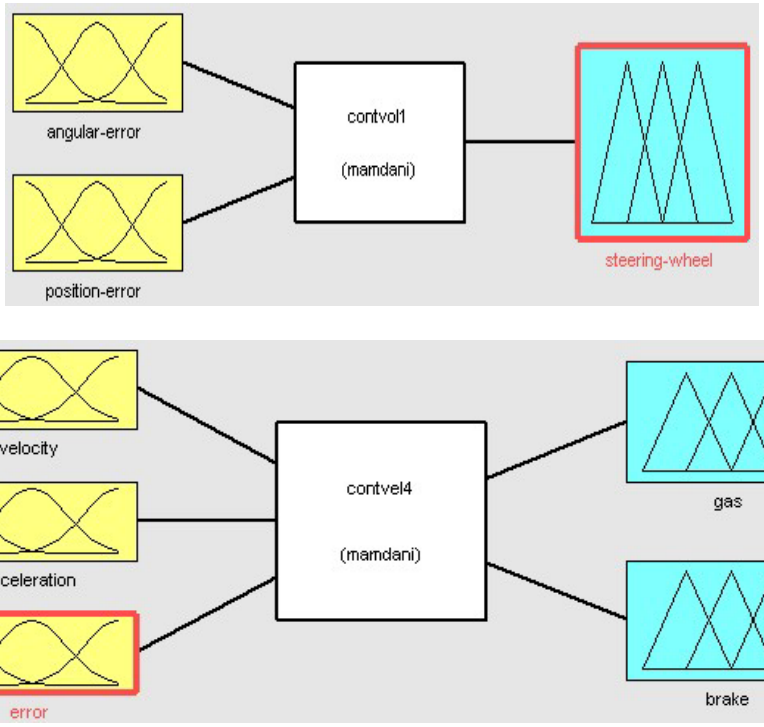

Fig. 6. Control diagrams: a) steering wheel control; b) gas-brake control.

Figure 7 shows the comparison of the velocity and the trajectory carried out by both the human driver and the automatic controller. It can be seen that the human driver tends to smooth the trajectory at the turns. His maximum error is 0.5 m, and occurs at the time of returning to the straight path. It is surprising the fact that the driver does not anticipate the last turn, likely due to an excessive speed.

The controller shows a more moderate trend to smooth the trajectory at the turns. During the curved part of the trajectory, the error is negligible, and the speed reduction is less acute than in the human-driven case. Again, the most difficult point arrives at the time of returning to the straight path, with an error of 0.22 m. For lower speeds, such error is almost zero. However,

as observed when looking at the trajectory performed by the human driver, the manoeuvre has been managed at a quasi-critical speed from a stability point of view.

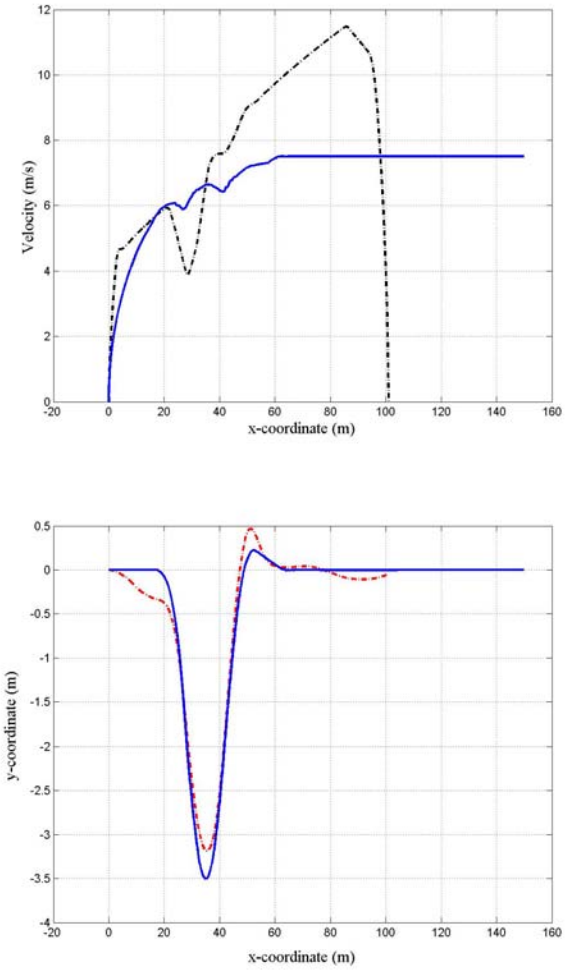

Fig. 7. Comparison between human driving (dashed) and automatic control for the second manoeuvre: a) velocity; b) trajectory.

The CPU-times obtained for the two manoeuvres reported when using the Matlab Engine option on a Pentium IV @ 2 GHz are listed in Table 1. CPU-times* include the time spent in opening a Matlab session, while CPU-times** are obtained when the running program is attached to an already opened Matlab session. It must be said that, when no control is considered, the Fortran program comfortably reaches real-time performance when solving for the dynamics of the vehicle.

It can be seen that the second manoeuvre needs a greater computational effort, due to the evaluation of two controllers, as described above. Evaluation of a fuzzy controller implies carrying out a *fuzzification* (i.e. determining the membership degree in the input fuzzy sets), applying the inference rules and, finally, performing a *defuzzification* (i.e. mapping an output value to its appropriate membership value in the output fuzzy sets). The CPU-times needed when using the MEX file option are slightly higher.

Table 1 Efficiency.

# Maneuver	Time (s)	CPU-time* (s)	CPU-time** (s)
1	12	18.00	11.99
2	24	36.14	30.45

The Matlab Engine option seems to be preferable, since the computational effort required is lower, and since less code must be added to the original program. Indeed, only commands for the opening and closure of the communication channel along with those relative to data transference must be added (when the MEX file alternative is chosen, a new heading file must be added too, in order to enable the program to be called from Matlab).

Therefore, Matlab Engine represents a good solution when, as in the present case, only some parts of the program need to be executed on Matlab. Furthermore, it is adequate for the stage of controller tuning, since real-time is not required. However, for the use of an already tuned controller on a real application, real-time performance must be achieved. At the view of the CPU-times shown in Table 1, it is obvious that another alternative must be searched. In order to find a solution, it must be taken into account that the fuzzy logic generates, by means of rules of membership and actuation, a hyper-surface which relates the input (error, velocity, etc.) and output variables (gas, brake, steering). Then, it is possible to evaluate the controller by sweeping a mesh of the different input values, and storing the resulting output values on a matrix. The mesh will be more refined at those points in which the controller behaviour is more nonlinear. Matlab provides function *evalfis* to this end. For those elements which are not in the matrix, fast interpolation can be done, thus assuring real-time performance.

6 CONCLUSIONS

Based on the results previously described, the conclusions can be drawn as follows:

a) By applying an authors' method for the dynamics of multibody systems, the computer model of an actual prototype car has been generated, and a Fortran program to determine its motion has been implemented.

b) Algorithms for the automatic control of the car during two manoeuvres have been developed by using fuzzy logic functions -which mimic human strategies-, provided by the corresponding Matlab toolbox.

c) Different alternatives to connect the Fortran and Matlab programs have been studied. Matlab Engine seems to be the best option for controller tuning, while encapsulation of the controller hyper-surface in a matrix appear to be more convenient for real-time applications.

d) Simulator capabilities have been given to the program by means of a realistic graphical output and game-type driving peripherals (steering wheel and pedals), so that comparison may be established between human and designed automatic control.

e) The two manoeuvres –straight line and obstacle avoidance– have been performed by both a human driver and the automatic controllers developed, and the obtained results have been compared, showing an excellent behaviour of the controllers.

In a future work, the authors intend to address the experimental validation of their formulation for the dynamics of multibody systems, by implementing the developed control algorithms onboard the actual prototype car and verifying whether a good behaviour is still achieved.

ACKNOWLEDGMENTS

This research has been sponsored by the Spanish CICYT (Grant No. DPI2000-0379) and the Galician SGID (Grant No. PGIDT01PXI16601PN).

REFERENCES

[1] J. Cuadrado, J. Cardenal and E. Bayo, "Modeling and Solution Methods for Efficient Real-Time Simulation of Multibody Dynamics", *Multibody System Dynamics*, **1**, 259-280 (1997).

[2] J. Cuadrado, J. Cardenal, P. Morer and E. Bayo, "Intelligent Simulation of Multibody Dynamics: Space-State and Descriptor Methods in Sequential and Parallel Computing Environments*", Multibody System Dynamics*, **4**, 55-73 (2000).

[3] P. Antos and J.A.C. Ambrosio, "A Control Strategy of Vehicle Multibody Model for Trajectory Tracking", *Multibody Dynamics 2003*, CD Proceedings, Lisbon, Portugal (2003).

[4] T.J. Gordon, M.C. Best and P.J. Dixon, "An Automated Driver Based on Convergent Vector Fields", *Proc. Instn. Mech. Engrs. Part D: J. Automobile Engineering*, **216**, 329-347 (2002).

[5] K. Yi, S. Lee and Y.D. Kwon, "An Investigation of Intelligent Cruise Control Laws for Passenger Vehicles", *Proc. Instn. Mech. Engrs. Part D: J. Automobile Engineering*, **215**, 159-169 (2001).

[6] N.D. Matthews, P.E. An, J.M. Roberts and C.J. Harris, "A Neurofuzzy Approach to Future Intelligent Driver Support Systems", *Proc. Instn. Mech. Engrs. Part D: J. Automobile Engineering*, **212**, 43-58 (1998).

[7] D.M. Bevly, J.C. Gerdes and C. Wilson, "The Use of GPS Based Velocity Measurements for Measurement of Sideslip and Wheel Slip", *Vehicle System Dynamics*, **38** , 127-147 (2002).

[8] J. Garcia de Jalon and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems –The Real-Time Challenge–*, Springer-Verlag, New York (1994).

[9] E. Bakker, and H.B. Pacejka, "The Magyc Formula Tyre Model", *1st International Colloquium on Tyre Models for Vehicle Dynamics Analysis*, Proceedings 1-18, Delft, Netherlands (1991).

[10] J.E. Shigley and C.R. Mischke, *Mechanical and Engineering Design*, McGraw-Hill, 6th edition, Singapore (2001).

[11] G. Ellis, *Control System Design Guide*, Academic Press, San Diego (1991).

[12] A. Hopgood, *Intelligent Systems for Engineers and Scientists*, CRC Press, New York (2001).

[13] Q. Zhang, "A Generic Fuzzy Electrohydraulic Steering Controller for Off-Road Vehicles", *Proc. Instn. Mech. Engrs. Part D: J Automobile Engineering*, **217**, 791-799 (2003).
[14] L.A. Zadeh, "Fuzzy Logic=Computing with Words", *IEEE Transactions on Fuzzy Systems*, **4**, 103-111 (1996).
[15] Matlab 6.1 Documentation, *User's Guide Version 2.1* (2000).
[16] M.A. Naya and J. Cuadrado, "Real-Time Determination of the Position of a Car with Tri-axial Accelerometers", *Multibody Dynamics 2003*, CD Proceedings, Lisbon, Portugal (2003).