Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California USA

DETC2005-84668

A ROBUST TOOL FOR TUNING AND EVALUATION OF AUTOMOBILE MOTION CONTROLLERS

Miguel A. Naya
Escuela Politécnica Superior
Universidad de La Coruña
Ferrol, La Coruña 15403. Spain
minaya@cdf.udc.es

Javier Cuadrado
Escuela Politécnica Superior
Universidad de La Coruña
Ferrol, La Coruña 15403 .Spain
javicuad@cdf.udc.es

Keywords: multi-body dynamics, simulation, vehicle control.

ABSTRACT

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 whether 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 coded in Fortran language. Fuzzy logic has been chosen to design the control algorithms, which have been implemented on the 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.

INTRODUCTION

Nowadays modeling and simulation of vehicle dynamics plays a great role in design and evaluation of vehicles and vehicle control systems. Robust and efficient simulations of a vehicle may avoid cost and heavy experiments and construction of prototypes. On the other hand, a real-time simulator of the vehicle dynamics may be included in the onboard equipment of modern vehicles in order to assist the driver either by providing him advice or by directly actuating on the system.

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. 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 maneuvers, like stairs descent.

Now, in order to find out whether 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 to 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 behavior 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 maneuvers are to be considered: straight line and obstacle avoidance.

Copyright © 2005 by ASME

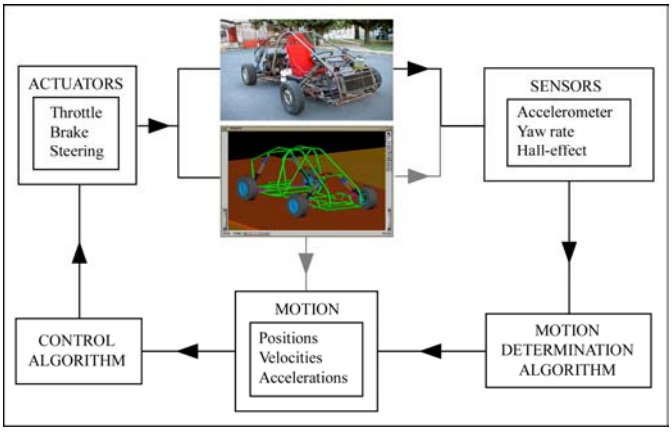


Fig. 1. General context of the work.

Many references can be found in the literature regarding the automatic control of car vehicles. The main area that contains these works is that of *Intelligent Transportation Systems*. The scope of this area is very wide and its motivations come from different disciplines that involve industrial and off-road vehicles, robotics, etc. For example, a great effort has been made in order to increase drivers' security levels. The research carried out in this area until 1995 is well referenced by Shladover [3]. This work states and classifies all issues related to systems that provide the driver with safety warnings or assistance to control the vehicle. In this frame, several research projects appear like California PATH [4] and the European project Prometheus [5]. Frequently, the main focus of this investigation is centered on the recognition of the road and the distance to the previous car [6, 7] or the determination of the optimum actuation over throttle-brake pedals [8]. Several other works are aimed at the determination of the vehicle position, employing inertial sensors and Global Positioning System [9-11]. A common practice in these works is the employment of a very simple model of the vehicle in order to avoid nonlinearities. Another point of view is the approach from the multi-body perspective that pays more attention to the vehicle modelling and its inclusion in the control. Two good reviews of the state-of-the-art concerning the trajectory tracking problem from this perspective were presented by Antos and Ambrosio [12], and Gordon et al [13]. However, the present work aims to address the problem from a more holistic and general point of view.

The paper is organized as follows: Section 1 focuses on the computational model of the car; Section 2 justifies the use of fuzzy logic in the present work; Section 3 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 4 shows the comparison, for the two maneuvers considered, between human and automatic control at simulation level; Section 5 explains the problem concerning the storage of the controllers in order to employ them in real-time applications; finally, Section 6 outlines the conclusions of the work.

NOMENCLATURE

ε_{pos}: Position error of the vehicle.
ε_{att}: Attitude error of the vehicle.
E_1, E_2: Local frame attached to the vehicle.
L: Optical lever to look ahead the intended path.
s: Step of the controllers discretization.
T_p: Preview time.
v_x: Longitudinal velocity of the car.
X, Y: Global frame.
x, y: longitudinal and lateral position of the vehicle at the global frame.

1 COMPUTATIONAL MODEL OF THE CAR

The mathematical model of the car, illustrated in Fig. 2 along with the physical prototype, has been carried out in natural coordinates [14]: 44 points, 7 unit vectors, 5 distances and 1 angle have been used as problem variables, leading to a total problem size of 159.

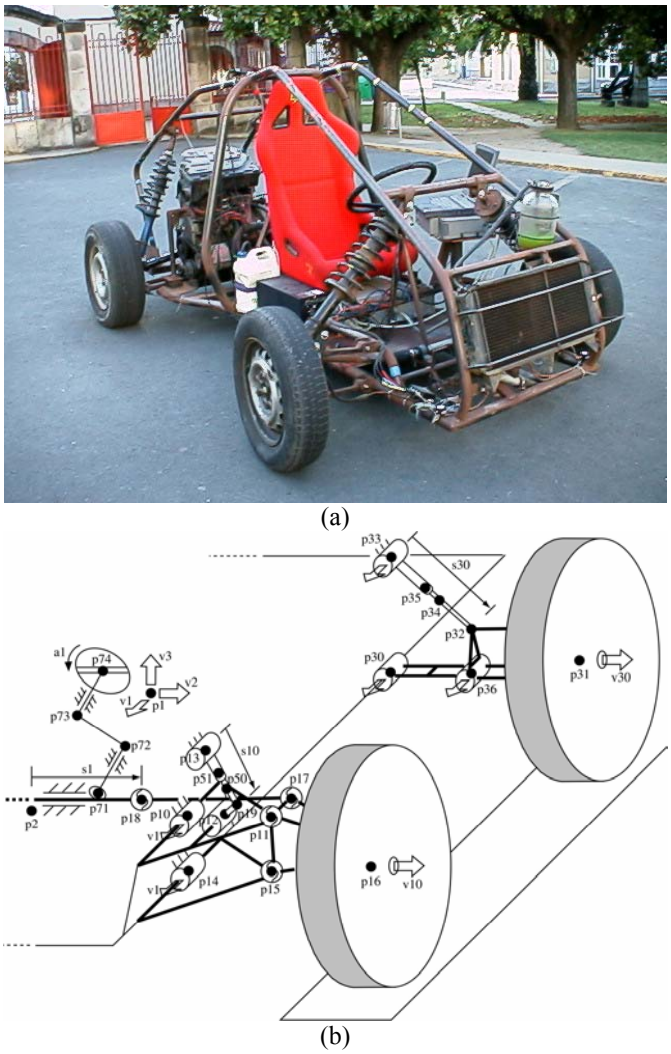


Fig. 2. a) The prototype; b) its model in natural coordinates.

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

Copyright © 2005 by ASME

kinematically guided. Forces which deserve to be described are the following:
- The chassis inertial parameters have been obtained from an I-deas model in which the actual structural elements have been replicated. The engine inertial parameters have been estimated experimentally.
- 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* [15] with coefficients provided by the tire maker. The longitudinal slip has been neglected and no model has been implemented in the simulation of the longitudinal forces. Therefore, forces provided by the brakes are directly applied to the centre of the four wheels and forces provided by the engine are applied to the centre of rear wheels. The contact between tire and ground is supposed to be able to transmit all the demanded force.
- Power transmission forces: from the torque-speed engine relationships 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. The model of power transmission includes engine braking torque at closed throttle position.
- Brake forces: the braking torque has been estimated from disk geometry [16], and applied to the four wheels.

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

2 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 the controller and evaluating the performance [17]. 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 is 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 [18]. 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 [19]. 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 [20], 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.
In order to mimic the driver behavior, physical attributes of the driver such as preview, adaptation to changing dynamic characteristics of the controlled vehicle, learning, anticipation, and planning abilities cannot be forgotten by the model. In fact, the driver employs a certain "internal", intuitive model or input-output understanding of the vehicle that allows him to compare the time-advanced expectation of the vehicle state at some future time with the directly observed preview input requirements [21]. The boundaries of this "internal model" are not very accurate and this fact fits very well on the Fuzzy Logic concept of membership of a fuzzy set.
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.

3 FORTRAN-MATLAB CONNECTION

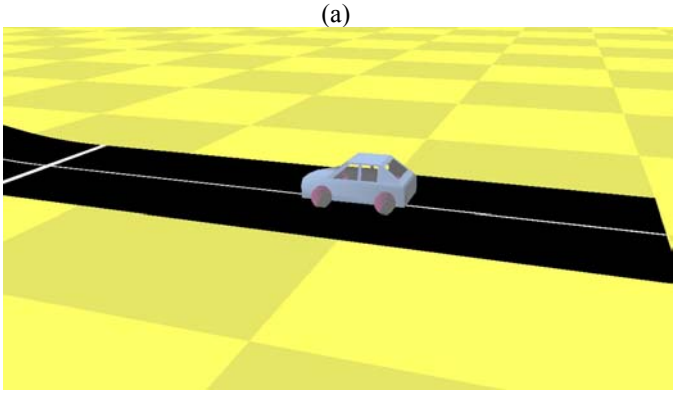
In order to introduce the control on the prototype model, the Fortran code containing the dynamics of the car and the Matlab functions implementing the fuzzy logic control algorithms must be combined. Matlab always employs double-precision variables, so that the different types of Fortran variables must 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 between Fortran and Matlab. Matlab features several compilation functions which enable data transfer between both languages, as well as executing functions in Matlab [22]. Then, when the Fortran program is running, 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 directly executed 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.

4 HUMAN VS AUTOMATIC CONTROL IN SIMULATION

The computational model of the car has been used to create a driving simulator, shown in Fig. 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. The simulator also allows to design maneuvers with realistic parameters. Moreover, the simulator enables the visualization of the maneuvers performed by the controllers.

Copyright © 2005 by ASME

(a)



(b)

Fig. 3. Driving simulator: a) general view; b) first maneuver.

As said in the Introduction, two maneuvers have been performed. In both cases, two controllers have been developed: a first one for the steering wheel and a second one for the actuation of the pedals. Although the steering wheel controller is the same in both cases, the importance of this control is more relevant in the second maneuver. The procedure for development of the controllers starts from an initial model, created by the designer from his perceptions about the maneuver. This model is tuned manually, repeating simulations until the desired accuracy in the performance is achieved.

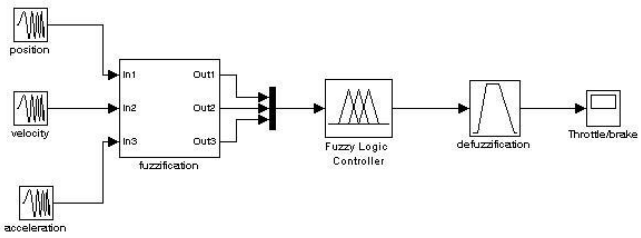


Fig. 4. Control diagram for the actuation of the pedals.

Fuzzy Logic controllers have been developed employing Mamdani-type inference and its typical defuzzification process [23], finding the centroid of a two-dimensional function to determine the value of the output variable from its membership to the output fuzzy set.

In the first maneuver, the car starts from rest, covers a distance of 20 m following a straight line (hence the low relevance of the steering wheel control), and stops. The objective is to stop the car as close to the target point as possible. This is why position and acceleration are more weighted than speed in the development of the controller. Anyway, a maximum speed value of 5 m/s is allowed, but the time spent in the maneuver has not been limited. First, a human driver carried out the maneuver. Figure 3a shows the corresponding simulator environment, and Fig. 3b shows the scenario of the first maneuver.

The control scheme for automatic actuation of the pedals has been represented in Fig. 4. It receives position, velocity and acceleration of the car, and acts upon throttle and brake. The domain of the actuation variable is [-1, 1], corresponding respectively to a fully pressed position of the brake pedal (-1), and to the same position of the throttle (1). Values sent to both actuators range from 0 to 1, corresponding to null and maximum displacement of the respective devices.

Such a procedure avoids a simultaneous operation on throttle and brake. Another advantage derived from the use of a unique variable is that, at the end, when controllers must be mapped and stored in a matrix (see below), the memory space required is lower.

No error function has been used, but the position of the car. The total travelling distance has been divided into four intervals – start (0 to 4.5 m), taxi (2.5 to 17.5 m), brake (16.5 to 19.5) and stop (18.75 to 20 m)–, 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 behavior in that phase in the case that the car exceeds the target point. Acceleration ranges from -3 to 3 m/s^2, but severe accelerations or braking are avoided.

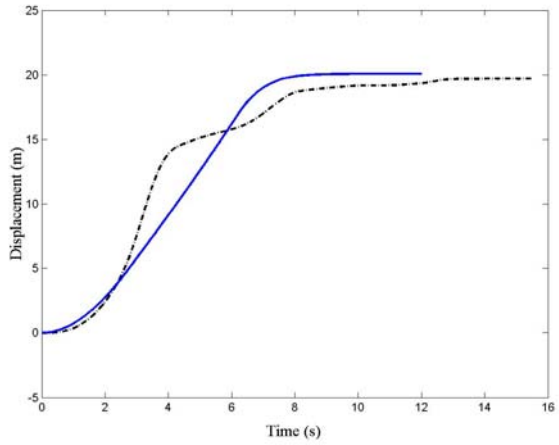
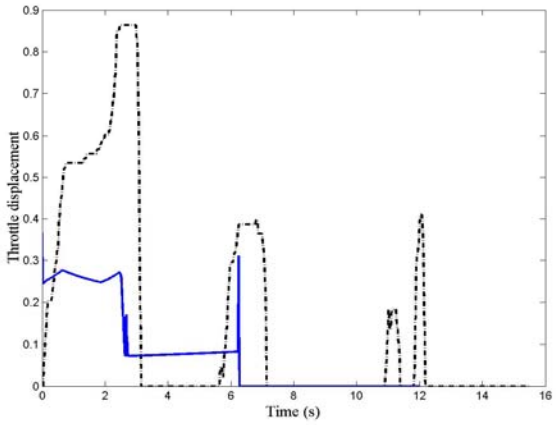


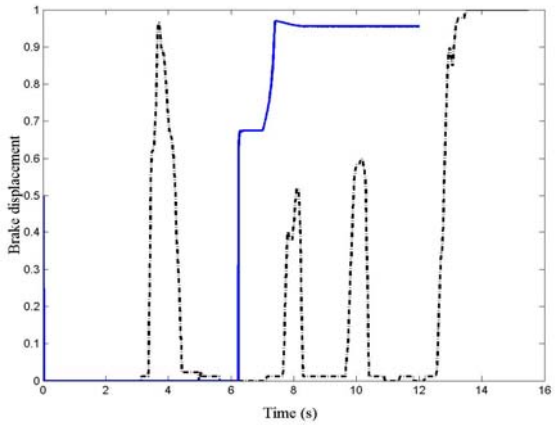
Fig. 5. Position history developed by human (dashed) and automatic control during the first maneuver.

Copyright © 2005 by ASME

Figure 5 shows the comparison of the position history, throttle and brake actuation carried out by both the human driver and the controller. As it can be observed, the maneuver performed by the controller is cleaner. The controller does not need to make several approximations, as the human driver does. Actuation on both the throttle and brake (see Fig. 6) is more efficient when the car is automatically controlled, and less time is needed to complete the maneuver.



(a)



(b)

Fig. 6. Comparison between human driving (dashed) and automatic control for the first maneuver: a) throttle actuation; c) brake actuation.

The second maneuver consists of 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. Acceleration domain is contained between ±3 m/s^2 and, again, severe actions are avoided. First, a human driver carried out the maneuver. This first trial by the human driver on the simulator ensures that control requirements regarding speed and handling are reachable. Figure 7 shows part of the corresponding path to be followed.

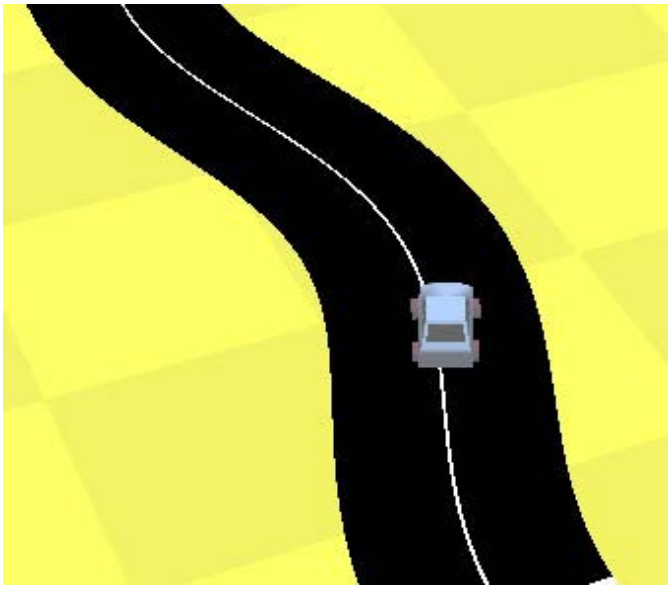
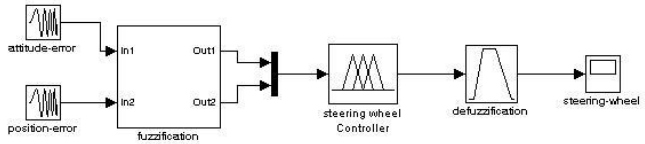
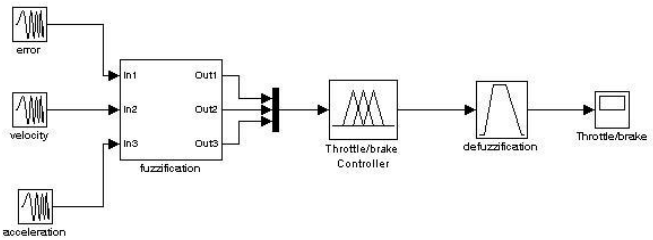


Fig. 7. Second maneuver at the simulator.

Figure 8 shows the scheme of the controllers employed in the second maneuver. Controls of both, the steering and the throttle-brake couple, have been addressed separately as in the first maneuver. For throttle-brake control, both the velocity and the acceleration of the car are taken into account, as well as the path tracking error. In this case, the error is calculated as a weighted mean of the two errors used for the steering control that will be described below, and ranges from 0 to 1. The pursued objective is that the controller slows down the car when the error in path tracking increases.



(a)



(b)

Fig. 8. Control diagrams: a) steering wheel control; b) throttle-brake control.

For steering control, two error functions have been defined. The first one is the position error of the vehicle at the current time, ε_{pos}. This error, shown in Fig. 9, is obtained as the distance from the car to the intended path, measured on the normal to

Copyright © 2005 by ASME

the actual path. This error takes values between -5 and 5 m, but values over 1 m are severely penalized.

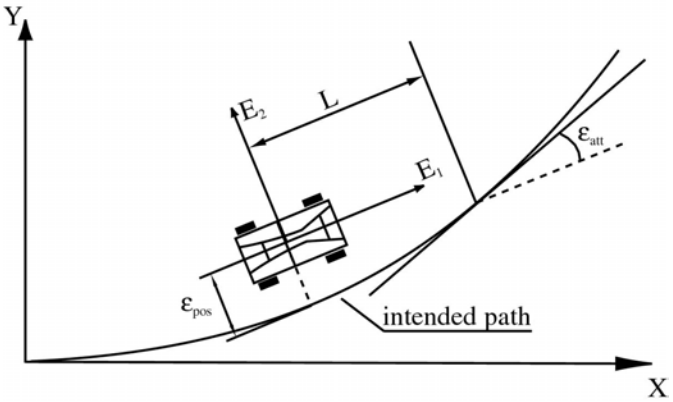


Fig. 9 Employed error functions.

Preview is an essential part of human driver control behavior [24] and, therefore, the second error employed, ε_{att}, aims to preview the car attitude 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 driver monitors the path ahead by projecting forward an optical lever *L*. The distance of the lever represents the extent of the preview available for a preview time T_p. In this way, the distance becomes a function of the vehicle longitudinal velocity:

$$L = v_x \cdot T_p$$

A T_p.value of 0,425 has experimentally shown to be the most appropriate for the maneuver. Values of ε_{att} between ±30º are observed.

Of course, different definitions of the employed errors are possible too. In fact, may be convenient to consider more than one preview point [23], but the the present work focuses more in developing a tool for tuning the controllers than in obtaining the most accurate controller.

Figure 10 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. Driver's maximum error is 0.5 m, and occurs at the time of returning to the straight path. It is surprising 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. This behavior is due to the preview. It is possible to more accurately adjust the controller to the intended path, by allowing a lower attitude error and by considering the attitude error without preview. However, the controller tuned in that way is too sensitive to small disturbances and, consequently, it is not very stable. So, a compromise between these two factors must be found.

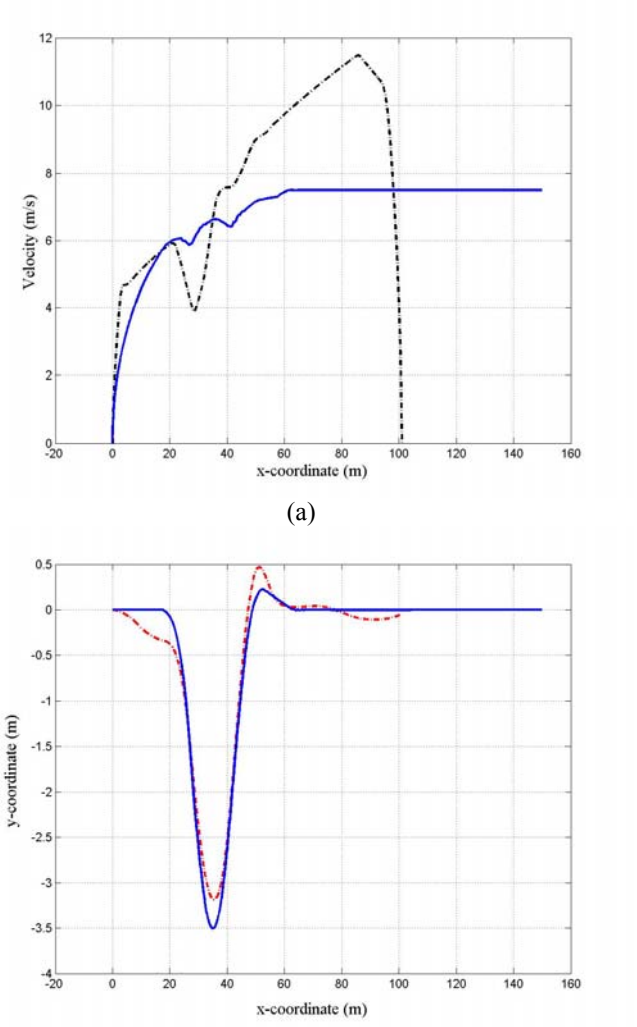


Fig. 10. Comparison between human driving (dashed) and automatic control for the second maneuver: a) velocity; b) trajectory.

Displacement of the steering wheel during the simulation is shown in Fig. 11. A maximum admissible variation between consecutive positions of 15º has been imposed. This value has been chosen having in mind the subsequent implementation of the controller in the prototype. In that context, if the steering wheel handling is effectuated by a step motor, such a displacement is equivalent to an excitation frequency of 833 Hz.

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 maneuver has been managed at a quasi-critical speed from a stability point of view.

Copyright © 2005 by ASME

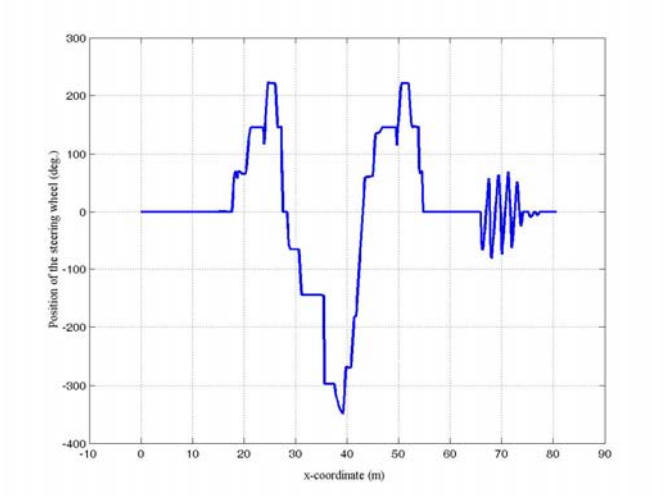


Fig. 11 Displacement of the steering wheel.

The CPU-times obtained for the two maneuvers 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 maneuver 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 transfer 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.

5 MATRICIAL STORAGE OF THE CONTROLLERS

If controllers are to be employed in stand alone applications that require real-time, the previous procedure is not a good option. Time delay is due to Fortran-Matlab connection and the heavy evaluation of the controller in Matlab. 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 (throttle, brake, steering). An evaluation of a matrix containing all the possible combinations of inputs of the controller can be carried out. Matlab provides function *evalfis* to this end. It is possible to employ a matrix as first input of the function in such a way that each row of the matrix contains one of the possible inputs required by the controller. *Evalfis* function will return another matrix whose rows are the corresponding outputs. If the controller is stored in this way, its subsequent evaluation only consists of an interpolation of the outputs matrix.

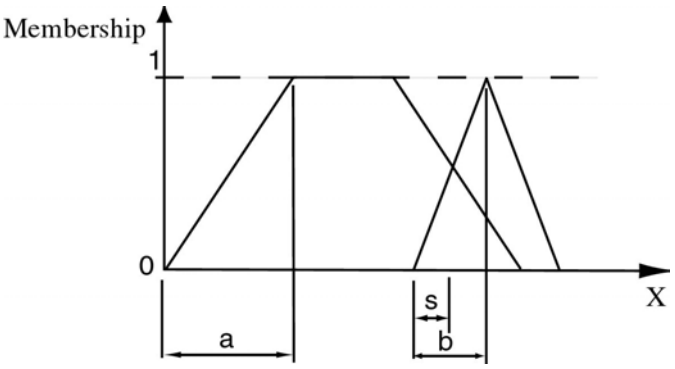


Fig. 12 Fuzzy sets and step of discretization.

In this procedure, it is not possible to employ an arbitrary value of a variable and, therefore, its range must be divided into a discrete set of values (discretization). This step plays a very important role. Fuzzy logic establishes degrees of membership to fuzzy sets. Discretization must reflect the membership of an input to a fuzzy set. Membership varies between 0 and 1 over an interval. The shape of membership functions must be defined: in the present work, trapezoidal and triangular membership functions have been employed. The intervals of variation of membership are different for different sets, as illustrated in Fig. 12. The step of discretization, *s*, employed, is one tenth of the minimum interval of variation of membership, *b* in Fig. 10. With these values of the step, the behavior of the controller is perfectly reproduced.

Table 2 shows the CPU-times obtained with controllers stored in matrices and loaded in the program as it starts. CPU-time* is the time obtained with Matlab Engine attached to a Matlab previous session (the faster case studied), and CPU-time** is the time consumed when the controllers are stored into matrices. As it can be seen, real-time performance is comfortably achieved.

Table 2. Efficiency employing controllers stored in matrices.

# Maneuver	Time (s)	CPU-time* (s)	CPU-time** (s)
1	12	11.99	4.87
2	24	30.45	11.75

Copyright © 2005 by ASME

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 maneuvers 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 suitable 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) Two maneuvers –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. DPI2003-05547-C02-01) and the Galician SGID (Grant No. PGIDT04PXIC16601PN).

REFERENCES

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

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

[3] Shladover, S.E., 1995, "Review of the State of Development of Advanced Vehicle Control Systems (AVCS)", *Vehicle System Dynamics*, **24**, 551-595.

[4] PATH on the Internet. California PATH Annual Reports 1996-2003.

[5] Dickmanns, E.D., 1997, "Vehicles Capable of Dynamic Vision". 15th International Joint Conference on Artificial Intelligence (IJCA-97). Nagoya, Japan.

[6] Yi, K., Hong, J., and Kwon, Y.D., 2001, "A Vehicle Control Algorithm for Stop-and-Go Cruise Control", *Proc. Instn. Mech. Engrs. Part D: J. Automobile Engineering*, **215**, 1099-1115.

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

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

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

[10] Ryu, J., and Gerdes, J.Ch., 2004, "Integrating Inertial Sensors with Global Positioning System (GPS) for Vehicle Dynamics Control", *Journal of Dynamic Systems, Measurement, and Control*, **126**, 243-254.

[11] Bevly, D.M., 2004, "Global Positioning System (GPS): A Low-Cost Velocity Sensor for Correcting Inertial Sensor Errors on Ground Vehicles", *Journal of Dynamic Systems, Measurement, and Control*, **126**, 255-264.

[12] Antos, P., and Ambrosio, J., 2004,"A Control Strategy for Vehicle Trajectory Tracking Using Multibody Models", *Multibody System Dynamics*, **11**, 365-394.

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

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

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

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

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

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

[19] Zhang, Q., 2003, "A Generic Fuzzy Electrohydraulic Steering Controller for Off-Road Vehicles", *Proc. Instn. Mech. Engrs. Part D: J Automobile Engineering*, **217**, 791-799.

[20] Zadeh, L.A., 1996, "Fuzzy Logic=Computing with Words", *IEEE Transactions on Fuzzy Systems*, **4**, 103-111.

[21] Macadam, C.C., 2003, "Understanding and Modelling the Human Driver", *Vehicle System Dynamics*, **40**, 101-134.

[22] Matlab 6.1 Documentation, 2000, *User's Guide Version 2.1*.

[23] Mamdani, E.H. and Assilian, S., 1975, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", *International Journal of Man-Machine Studies*, **7**, 1-13.

[24] Sharp, R.S., Casanova, D. and Symonds, P., 2000, "A Mathematical Model for Driver Steering Control, with Design, Tuning and Performance Results", *Vehicle System Dynamics*, **33**, 289-326.

 Copyright © 2005 by ASME