ABCM Symposium Series in Mechatronics - Vol. 1 - pp.187-196
Copyright © 2004 by ABCM

INDIMADA: DIRECT INFERENCE OVER A MASS OF DATA, TESTED IN A SIMULATION AS A CONTROL ALGORITHM OF A VEHICLE TRANSMISSION SYSTEM

Leonardo Malheiros Serrano de Cerqueira Leite
UFRJ, Universidade Federal do Rio de Janeiro, Centro de Tecnologia Bloco G - Sala 204
Cidade Universitária - RJ – Brasil, Caixa Postal 68503 - CEP 21945-970
leoleite@petrobras.com.br

Paulo Fernando Ferreira Rosa
IME, Instituto Militar de Engenharia, Praça General Tibúrcio 80, Praia Vermelha, Rio de Janeiro - RJ - CEP 22290-270
rpaulo@de9.ime.eb.br

Max Suell Dutra
UFRJ, Universidade Federal do Rio de Janeiro, Centro de Tecnologia Bloco G - Sala 204
Cidade Universitária - RJ – Brasil, Caixa Postal 68503 - CEP 21945-970
max@serv.com.ufrj.brr

Abstract. *This work proposes a new control algorithm. It merges the system input and output data in the same structure named "data" and organize these data in the system input vectorial space allowing a fast navigation through them. Then the algorithm calculates parameters that characterize the system local behavior for each data collected and store these parameters in the respective data. Finished these two steps, the algorithm is able to navigate through the system vectorial input space and infers the correct command for a new input using the parameters calculated on the second step and stored on the data set. As the number of data inserted in the organized data mass is increased the performance and correctness of the algorithm is increased too.*
The algorithm was tested in the control of the clutch and the change of the gears of a virtual vehicle. A model of a transmission system available nowadays was developed to evaluate the efficiency of the algorithm. Although the results reached are insufficient for an efficient control of the system, they show that the algorithm is able to execute intelligent tasks and show that the improvement of the algorithm can make it able to execute the control task in an efficient way.

Keywords*. artificial intelligence, control algorithm.*

1. Introduction

The robotic technology has been well developed in the last times, having allowed the man to construct robots each time more similar to a human being and a form of walking each more natural, but still with very inferior abilities and versatility to one human being. Even if a robot was developed with flexibility, versatility and physical ability of a human being, it would have its utility very limited if was controlled for algorithms strong dependents of the man to each new task. The available algorithms currently demand, to each new task, great human interference, either defining parameters, effecting adjustments or modifying stretches of the program, for the efficient execution of the new task. This very limits the autonomy of the robots that use these algorithms.

1.1. About this research

The hereafter proposed algorithm is composed of four main components: (a) data generation (utilizing driving conditions, the driver's behaviour simulation and automobile response simulation); (b) data treatment and organization; (c) a direct inference algorithm over a mass of data; and, the evaluation of control performance and stability. The data organization is layed upon a neighbourhood structure; i.e., data are grouped accordingly their proximity in a n-dimensional space of inputs. This struture can be considered as a graph or a new conception for a neural network.

For this research, a hypothetical model of the transmission system of an automobile was developed. Real data from a car manufacturer were used as an input to this model, in order to generate accurate data for our study. These data had been analysed, treated and used in the development of the software that commands the system. With the model of the transmission system and and the control algorithm, it was possible to carry out simulations and to verify the effectiveness of the proposed method.

2. Motivation

The majority of the algorithms of artificial intelligence, or control with learning, posses restricted applications and demands high degree of human interference, besides demanding much try and error procedure and the use of subjective criteria in its structuration and adjust. So, even if we construct a robot that physically held in similar way the human beings, would not be possible make it imitate a human being, using the existing algorithms, with reasonable degree of success, our capacity to learn quickly as to react ahead of a great variety of different situations. We use the same types of "algorithms" to decide an immense gamma of different problems and without needing that nobody external comes to adjust the parameters of our "algorithms" for each new problem that we learn. Thus, if one wants to construct

multipurpose and more independent robots, will have to imitate this characteristic of universality and auto-adjustment. The algorithm considered here is potentially powerful and can be used in different kinds of problems, being, however, sufficiently simple, besides not needing to great part of the interference human being, necessary to the majority of the current algorithms.

3. Revision of literature

In order to optimize the training of a multilayers artificial neural network, Liang (1995), presents a new algorithm, using Kalman filter, for correction of the weights of neural networks, used for identification on-line and systems of control of adaptative tracking and control with reference model. In the work he used the concept of linearization of input and output of nonlinear systems. In another work, Hsi-Chin (1995) considers a tax of dynamic learning, based in the vectors weight of the current cycle and the previous cycle of training, using the backpropagation algorithm. With this, he obtained a substantial increase in the speed of training of the neural net, used for standard recognition. As much the algorithm of Liang (1995) as of Hsi-Chin (1995) suffers of same males of artificial neural network, therefore it presents considerable a computational work during the training of the net, what it results in delayed training, even so faster of the one than backpropagation, and, mainly, does not have guarantee of that the reached minimum error is the global minimum (optimal point) of the function error, while the algorithm considered in the present work, called INDIMADA, adjusts itself quickly to the new data and does not have the problem of sub-optimization due the local minimums, moreover, algorithm INDIMADA demands minor human interference. In its work Ahson (1995) increased the functionality of the Petri net, in order to use to take advantage of the tools already developed for this type of structure, but it did not demonstrate significant advantage in relation to the fuzzy algorithms or conventional artificial neural networks. Moreover, the algorithm suffers of same males of artificial neural network, while the algorithm considered in the present work adjusts itself quickly to the new data and does not present the problem of sub-optimization due to the local minimums.

In his work on fuzzy algorithms Li (1995) develops a systematic for fuzzy modeling for control problems. The fuzzy controller obtained had a superior performance to the comparative PID. Qiao (1995) compares the algorithm of the weighed distance with its misty algorithm with learning. In its work, Yen (1995) used fuzzy logic to extend the method of fusing of commands of Payton and Rosenblatt for control of mobile robots. Using fuzzy algorithm, however, still the calculation of the center of mass of a complex area is necessary, what it increases the processing time, beyond the importance of the optimized definition of diverse parameters, while the algorithm developed in this work has the majority of its more important parameters automatically determined. Yip (1994) considers a new type of approach for parallel and distributed processing for optimal solution searching, combining evolutive techniques with the Simulated Annealing algorithm. But the evolutive techniques and the Simulated Annealing algorithm depends on randomly generated values, whose distribution itself represents a serious problem to be treated, besides the always present probability of the obtained result not being the optimal one, therefore depends on luck once it is obtained by the evaluation of randomly generated values.

4. Developed algorithm: Direct inference over a mass of data (INDIMADA)

Our objective as scientists and engineers is to develop devices that execute useful functions and bring advantageous solutions on the other options. Thinking about this, an algorithm was created that does not suffer from many males of other algorithms used for control, besides demanding little computational effort during the calculation of the output, once there are necessary only few simple calculations to determine it.

An unknown author already wrote that "the more original the idea, more obvious it seems to be", and is this that occurs with algorithm INDIMADA. In certain way it is already in use, some times in unconscious way, in diverse day-by-day applications, what demonstrates its versatility. But no formalization was found that allows its use in a computer in a more generalized way in the area of artificial intelligence. Once it is understood the algorithm seems obvious, in contrast of the reason for which it still was not found any application of artificial intelligence.

The considered system does not exclude the human interference, but it occurs much less than in the existing control algorithms and demands little specific knowledge on the treated problem once the algorithm is based on the data collected and not on a model of the system.

The algorithm developed here can be used to approach functions in the parts of its domain where they are discontinuous, with discontinuities of the type jump or the type removable, or continuous. The more complex or brusque the surface of the function, more given data will be necessary for a good approach. On the other hand, the more soft the function, less given data are necessary to get a good approach.

The data that the system uses include the input and output values in a way that it can ahead perceive the reaction and behavior of the driver of each situation of the domain of the problem.

INDIMADA algorithm looks in the vectorial space of the domain the data next to the input vector, whose output is desired. The algorithm calculates the image (output) of the input vector based on the local behavior of the function, this behavior is obtained from the points of the organized mass located next of the input vector, whose output is desired, in the domain.

4.1. Functioning of the algorithm

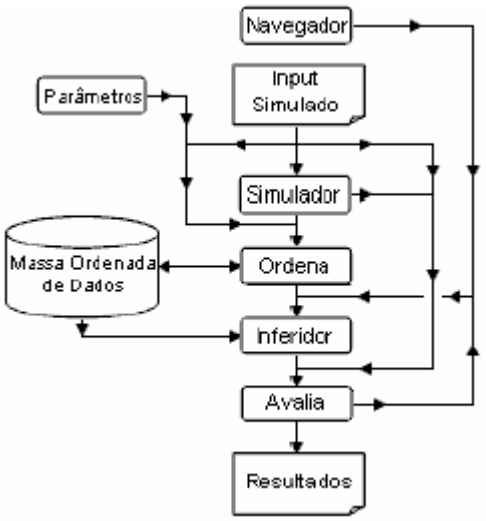

Figure 1. Flowchart of functioning of the algorithm.

The rounded off cornes rectangles of the flowchart are programs, all functioning together define the INDIMADA algorithm subject of this work. The arrows indicate which programs those are necessary for the execution of each item. The cylinder called "Massa Ordenada de Dados" represents information saved in hard disk and the rectangles of folded corner represent input and output of the program. Each item will be explained in the subs-section to follow.

4.2. Sub-routine parâmetros

This is the part of the program where all the used constants and parameters are defined. In this way, we facilitate the alterations and make it faster and safer. These parameters are save in the hard disk to be accessed later by the modules who need its values.

The used parameters and constants are: 1- max_group_size: maximum size of the organized sub-groups of data; 2- child_group_size: size of the organized sub-groups of data derived from a bigger group after to have been divided; 3- in_size: dimension of the domain space of the problem; 4- out_size: dimension of the image space of the problem; 5- centralize_step: number of inserted data in a organized sub-group of data between each operation of centralization; 6- tolerances: vector that indicates the tolerances, that is, the desired maximum error in each output variable. This calculated error is the module of the difference between the correct output and the inferred one.

4.3. Sub-routine navegador

Uses the reference of neighborhood of each data, created by "Ordena" or by "Insere", to browse in the space of the domain of the problem in order to find the data of the organized mass of data that better approaches to the new data to be inserted, or the "Input" vector whose respective "Output" is desired.

4.4. Sub-routine ordena

The Insertion of data, without any organized mass of data, would become the module "Insere" more complex. Thus, the used solution was to store an amount of data k ($k < max\_group\_size$) and to organize them, for then inserting data using "Insere". This program is used when we have a mass of disorganized data and no any organized mass of data. It organizes the data of the sensors of the automobile for proximity in the input space in accordance with the distance and the relative position between them. After the ending of the ordinance, each output variable is locally approached by second degree equations of the input variable, and then the derivatives of each output variable in the direction of each input variable are calculated and, finally, all these information are saved in each one of the data.

4.5. Sub-routine insere

The input and output system data are inserted using “Insere” that keeps the coherence with respect to the information of the neighboring data, so that the navigation is successful and not deviates from the new added data, nor of the old organized data. It establishes the signals of the sensors of the automobile in one only structure called “data”. Using "Inferidor" it infers the output for this data and compares it with the measured real output. If the module of the difference between the inferred and the measured output surpasses the respective value stored in tolerances, then this new data is inserted in the organized mass of data. "Insere" locates the new data in the organized mass of data, using a

structure of neighborhood, in accordance with in the distance and the relative position between it and the data already organized, and adds the new given data to the sub-group that will have its "seed" next to the data to be inserted. Later, it inserts the new data in all the sub-groups in which its more distant elements of the respective "seeds" are more distant of them of whom the new data. Made this, each output variable is locally approached by second degree equations of the input variable and then are calculated the derivatives of each output variable in the direction of each input variable and, finally, all these information are saved in the new data, in all the groups where it appears. The organized mass of data is divided in diverse individually overlapped sub-groups stored in the hard disk in order to speed the recovery of these data during the on-line functioning and also not to overload the memory of the computer.

The first thing to be made is to choose the first "seed" of the first data group, that is, the only organized group of data so far. For this, the centralization of this group is made: the sum of the distances of each data for all remaining data if the group is calculated, the data that give the minor sum is chosen as the central data of this group, that is, its "seed" of this group. To each n (n = centralize_step) new inserted data in a sub-group, are made a new centralization in this sub-group. As new data are added to a organized data group this group grows and can become very big sized. To prevent the decurrent problems of an exaggerated growth of the organized sub-groups of data, the big size original sub-group is broken in two new lesser sub-groups. In case that the number of data inside of a sub-group reaches max_group_size, then the sub-group is broken in two new sub-groups each one with m given (m = child_group_size). Before dividing the data group it is calculated in the distance of each data of this group until its "seed", the data most distant is called "semente1". Then it is calculated in the distance of each data of the group until "semente1", the data most distant is called "semente2". It is selected, amongst the data of the group to be divided, (p-1) given (p = child_group_size) the next ones to "semente1", these are the data that, together with "semente1", will compose the first one of the new sub-groups of data. It is selected, then, amongst the data of the group to be divided, (p-1) given the next ones to "semente2", these are the data that, together with "semente2", will compose the new sub-groups of data. After to center each one of the new sub-groups, them they are then stored in the hard disk, ande the original group that had originated two sub-groups is eliminated.

Then the inferred output of the new data is recalculated and compared with the real output. In case that the module of the difference beetween any dimension of these two outputs surpasses the foreseen one in tolerances, then the generated data have its output modified in order to coincide with the real output, or else, this generated data are not more necessary, being then eliminated of the mass of data. In both cases the indices of the functions derivatives of all the neighboring data to the generated data are recalculateds.

After the ending of the insertion of the new data, each output variable is locally approached by second degree equations of the input variables, and then the derivatives of each output variable in the direction of each input variable are calculated and, finally, all these information are stored in the new data. The same procedure is made for all the neighbors of the new data.

4.6. Sub-routine inferidor

Has the vector Input as argument and gives the vector Output. The "Inferidor" uses the "Navegador" to find, amongst the data stored in the organized mass of data, the data next to "Entrance" in the input space. The algorithm calculates the image (output) of the input vector based on the local behavior of the function, gotten from the points of the organized mass of data located next in the domain of Input vector, whose output is desired. It has two types of inference, one for discrete outputs and another one for continuous outputs that, in certain cases, also can be used for discrete output.

For the inference of discrete output, for each possible outputs, is calculated the sum of the squares of the distance in the output space, between the output of the next data and the candidate to output of the Inferidor, divided by the distance, or a function of it, between them in the input space. The candidate who present the minor sum is the chosen one.

For the inference of continuous output, the image of the function in the desired point, indicated by Input, is a weighed mean of the images of the neighboring points, where the weight of this average is given by the inverse one of the vectorial distance, in the input space, from each one of these data until the input vector, with conveniently spaced out vectorial components.

4.7. Input & Output

Is the vector composed by the already digitalized signals coming from the sensors of the vehicle. The system input signals are all those important ones to correctly decide on the command to be given. The Input vector is the argument of "Inferidor" for the calculation of the corresponding Output.

4.8. Organized mass of data

The data are organized in a neighborhood structure that can be understood as a graph, where each point points with respect to a certain number of next neighbors, in each felt and direction of the input space. The algorithm simulates a function that maps the space of the input data in the space of the output data. Each given set of input + output constitutes an unit of information used by the algorithm and is simply called data. Each data points with respect to a

certain number of neighbors. The number of pointed neighbors varies inside of the band expressed by the Eq. (1). If, for each input variable we have 2 options, minor and greater, then we have:

$$1 \leq Viz \leq 2 \cdot n_{in} \tag{1}$$

where $2 \cdot n_{in}$ indicates the maximum number of neighbors of data and n_{in} the dimension of the input space (domain). Viz indicates the number of neighbors of data in the input space (domain). It is said here of maximum number of neighbors, therefore the number of neighbors depends on the position of the data considered in relation to the mass of data in the input space. If data will be in the center of the mass of data, it will have the maximum number of neighbors, therefore it will have neighbors in the two felt of all the directions. If the data will be in the "face" of the mass of data, it will have a neighbor to less (corresponding at the dimension in which the data arrived at the extremity). If the data will be in a "edge" or a "meeting of edges" of the mass of data, then the number of neighbors will be equal to the maximum number of neighbors less the number of dimensions where the data arrived at the extremity. All these situations are find illustrated in the Fig. (2) for a domain of dimension 3.

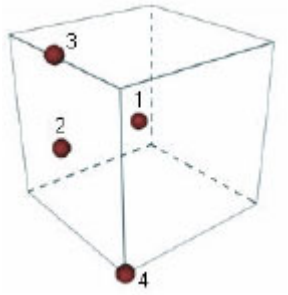

Figure 2. Number of neighbors: (1) data inside the mass of data, (2) data in a face of the mass of data, (3) data in the edge of the mass of data, (4) data in a meeting of edges of the mass of data.

5. Experiment and simulation

As much the attainment of the data how much the evaluation of the algorithm had been made using a simplified computational model of a automachine vehicle.

The algorithm of this work was considered to function using given real, collected of the sensors of the vehicle. As it was not possible in this phase of the study to use a real vehicle, one became necessary the development and use of a simplified model of the system of manual transmission adopted currently.

5.1. Project of the automobile

The automobile, the differential and the train of gears of the gearbox are approached by a rotating mass, with moment of inertia *Ja*, submitted to three torques: torque of internal attrition (*Tμa*, due to the forces and torques of attrition of the transmission components), considered constant and of contrary direction to the angular speed *ωa*; torque of inclination of the ground (*Ti*, due to inclination of the ground in the longitudinal direction of the vehicle); e torque transmitted by the clutch (*Te*).

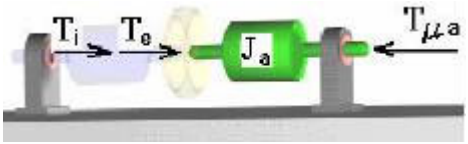

Figure 3. Modeling of the automobile. *Ti* = torque due to inclination of the land; *Te* = torque transmitted by the clutch; *ωa* = torque due to the attrition involving the components of the transmission after the clutch; *Ja* = moment of inertia of the mass that represents the automobile in the model.

5.2. Project of the engine

The engine is approached by a rotating mass, with moment of inertia *Jm*, equivalent to the mobile parts of the engine, submitted to three torques: attrition torque *Tμm* (due to the forces and torques of attrition of the engine components, considered constant and of contrary direction of rotation *ωm* of the mass), torque generated by the engine *Tm* (by means of the burning of the fuel, function of the project of the engine, the rotation and the accelerator position) and external torque *Te* (the load, deriving of the clutch, that the engine is submitted).

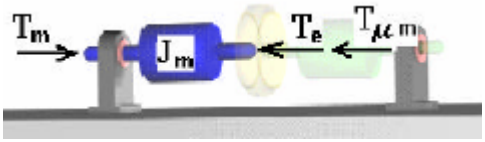

Figure 4. Modeling of the engine. *Tm* = torque generated by the engine; *Te* = torque transmitted by the clutch; *Tμm* = torque due to the attrition involving the mobile components of the engine; *Jm* = moment of inertia of the mobile parts of the engine.

5.3. Time of release of the clutch

a) when the vehicle starts from the rest (either of 1st march, either of march-the-reverse speed, the only difference will be the value of the reduction, t_r),

$$t_f = \omega_m \cdot \frac{J_t + \frac{m_a \cdot r^2}{r_t^2}}{P_a \cdot sin(\beta) \cdot \frac{r}{r_t} - T_{\mu a} + \frac{T_{\max}}{2}} \tag{2}$$

b) when the vehicle is already in movement

$$t_f = \frac{J_m \cdot |\omega_a - \omega_{ml}|}{\frac{T_{\max}}{2} + T_{ml} - T_{\mu m}} \tag{3}$$

Where J_t = moment of inertia of the mobile parts of the automobile, P_a = weight of the automobile, β = angle of inclination of the track, r = ray of the traction wheel, r_t = total reduction of the transmission system (gearbox and differential), $T_{\max}$ = maximum torque transmitted by the clutch (maximum torque of the engine), ω_{ml} = rotation of the engine in march-slow, T_{ml} = torque of the engine in march-slow.

6. Results

Masses of data with three different sizes, one with 150 data, another one with 250 data and the last one with 500 data had been organized. Each mass of data was submitted to evaluations with three distinct types of inference: inference using inferrer for discrete and continuous output, inferrer only for discrete output, and an adaptation of this last one, where the inverse of the distance - used as weight of the weighed mean of the algorithm of the inferrer - was raised the three ($peso \rightarrow peso^3$). To have a base of comparison for the evaluation of the inferrers, it was made the copying that, instead of inferring the result, simply copies the Output, of the present data in the organized mass of data that will be next to the vector v_{in}, whose respective output is desired. All the statistical results obtained with these simulations are expressed in tables to follow. After the tables, come the legend, that clarifies the used acronyms and signals. The commentaries on tables can be find in the next item.

Table 1. Statistical measures of the performance of the copying.

	150 organized data						250 organized data						500 organized data					
	M			t(s)			M			t(s)			M			t(s)		
Var	0,3			0,107			0,3			0,095			0,3			0,074		
DM	0,3			0,189			0,3			0,173			0,3			0,146		
DP	0,6			0,327			0,6			0,307			0,5			0,272		
Var_e	1,2			0,107			1,2			0,095			1,2			0,074		
DM_e	1,07			0,189			1,08			0,173			1,06			0,146		
DP_e	1,11			0,327			1,12			0,307			1,09			0,272		
Var % / 100	0,35			2,8			0,34			1,5			0,33			3,4		
DM % / 100	0,38			0,7			0,37			0,6			0,37			0,6		
DP % / 100	0,59			1,7			0,58			1,2			0,58			1,9		
Var_e % / 100	0,6			2,8			0,5			1,5			0,5			3,4		
DM_e % / 100	0,7			0,7			0,7			0,6			0,7			0,6		
DP_e % / 100	0,8			1,7			0,7			1,2			0,7			1,9		
≠ %	27,1			100			25,4			100			23,7			100		
max error	2			1,5			3			1,5			2			1,5		
error % max / 100	2			19			1			18,3			1			21,6		
value of the error in the march	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
march error %	72,9	25,1	2	0	0	0	74,6	23,5	1,8	0,1	0	0	76,3	22,2	1,5	0	0	0

Table 2. Statistical measures of the performance of the continuous inferrer.

	150 organized data						250 organized data						500 organized data					
	M			t(s)			M			t(s)			M			t(s)		
Var	0,3			0,073			0,3			0,063			0,2			0,052		
DM	0,3			0,183			0,3			0,171			0,2			0,149		
DP	0,5			0,27			0,5			0,251			0,5			0,227		
Var_e	1,2			0,073			1,1			0,063			1,1			0,052		
DM_e	1,05			0,183			1,04			0,171			1,03			0,149		
DP_e	1,09			0,27			1,06			0,251			1,05			0,227		
Var % / 100	0,31			0,3			0,3			0,3			0,3			0,2		
DM % / 100	0,34			0,4			0,34			0,4			0,33			0,3		
DP % / 100	0,55			0,6			0,55			0,5			0,55			0,5		
Var_e % / 100	0,5			0,3			0,5			0,3			0,5			0,2		
DM_e % / 100	0,7			0,4			0,6			0,4			0,6			0,3		
DP_e % / 100	0,7			0,6			0,7			0,5			0,7			0,5		
≠ %	24,5			100			23,9			100			20,6			100		
max error	2			1,5			2			1,4			3			1,3		
error % max / 100	1			3,2			1			3,3			1,5			2,7		
value of the error in the march	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
march error %	75,5	23	1,5	0	0	0	76,1	23	0,9	0	0	0	79,4	20,1	6,4	0,1	0	0

Table 3. Statistical measures of the performance of the continuous inferrer with the weight high to three.

	150 organized data						250 organized data						500 organized data					
	M			t(s)			M			t(s)			M			t(s)		
Var	0,3			0,074			0,3			0,065			0,2			0,049		
DM	0,3			0,169			0,3			0,16			0,2			0,134		
DP	0,5			0,271			0,5			0,255			0,5			0,222		
Var_e	1,2			0,074			1,1			0,065			1			0,049		
DM_e	1,05			0,169			1,03			0,16			1,01			0,134		
DP_e	1,07			0,271			1,05			0,255			1,02			0,222		
Var % / 100	0,32			0,5			0,31			0,3			0,3			0,3		
DM % / 100	0,35			0,4			0,34			0,4			0,34			0,3		
DP % / 100	0,56			0,7			0,55			0,6			0,55			0,5		
Var_e % / 100	0,5			0,5			0,5			0,3			0,4			0,3		
DM_e % / 100	0,7			0,4			0,6			0,4			0,6			0,3		
DP_e % / 100	0,7			0,7			0,7			0,6			0,7			0,5		
≠ %	25,7			100			24,5			100			21,5			100		
max error	2			1,5			2			1,4			2			1,4		
error % max / 100	1			8,2			1			6,3			1			5,4		
value of the error in the march	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
march error %	74,3	24,4	1,3	0	0	0	75,5	23,7	0,8	0	0	0	78,5	21,2	0,3	0	0	0

Table 4. Statistical measures of the performance of the discrete inferrer.

	150 organized data						250 organized data						500 organized data					
	M			t(s)			M			t(s)			M			t(s)		
Var	0,3			0,075			0,3			0,06			0,2			0,054		
DM	0,3			0,182			0,2			0,165			0,2			0,145		
DP	0,5			0,27			0,5			0,245			0,5			0,233		
Var_e	1,1			0,075			1,1			0,06			1,1			0,054		
DM_e	1,02			0,182			1,04			0,165			1,04			0,145		
DP_e	1,03			0,27			1,06			0,245			1,07			0,233		
Var % / 100	0,33			0,3			0,31			0,2			0,31			0,2		
DM % / 100	0,36			0,4			0,34			0,4			0,34			0,3		
DP % / 100	0,57			0,5			0,56			0,5			0,55			0,4		
Var_e % / 100	0,5			0,3			0,5			0,2			0,5			0,2		
DM_e % / 100	0,7			0,4			0,6			0,4			0,6			0,3		
DP_e % / 100	0,7			0,5			0,7			0,5			0,7			0,4		
≠ %	25			100			22,9			100			21,5			100		
max error	2			1,4			2			1,1			3			1,4		
error % max / 100	1			4,1			1			2,5			1,5			2,9		
value of the error in the march	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
march error %	75	24,5	0,5	0	0	0	77,1	22	0,9	0	0	0	78,5	20,7	0,7	0,1	0	0

Table 5. Comparison between the statistical measures of the performance of the discrete inferrer for two different samples, both of size 1000.

	Amostra 1 (150 organized data)						Amostra 2 (150 organized data)					
	M			t(s)			M			t(s)		
Var	0,3			0,075			0,3			0,073		
DM	0,3			0,182			0,3			0,184		
DP	0,5			0,27			0,5			0,27		
Var_e	1,1			0,075			1,1			0,073		
DM_e	1,02			0,182			1,03			0,184		
DP_e	1,03			0,27			1,05			0,27		
Var % / 100	0,33			0,3			0,33			0,3		
DM % / 100	0,36			0,4			0,36			0,4		
DP % / 100	0,57			0,5			0,57			0,5		
Var_e % / 100	0,5			0,3			0,5			0,3		
DM_e % / 100	0,7			0,4			0,7			0,4		
DP_e % / 100	0,7			0,5			0,7			0,5		
≠ %	25			100			27			100		
max error	2			1,4			2			1,2		
error % max / 100	1			4,1			2			2,6		
value of the error in the march	0	1	2	3	4	5	0	1	2	3	4	5
march error %	75	24,5	0,5	0	0	0	73	26,1	0,9	0	0	0

Legend of tables:

M: Column of the results referring to the march.

t (s.) Column of the results referring to the time of release of the clutch, in seconds.

Var: Measure of the variance of the error.

DM: Measure of the average shunting line of the error.

DP: Measure of the shunting line standard of the error.

Var_e: Measure of the variance of the error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ.

DM_e: Measure of the average shunting line of the error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ.

DP_e: Measure of the shunting line standard of the error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ.

Var % / 100: Measure of the variance of the percentile error divided by 100.

DM % / 100: Measure of the average shunting line of the percentile error divided by 100.

DP % / 100: Measure of the shunting line standard of the percentile error divided by 100.

Var_e % / 100: Measure of the variance of the percentile error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ, divided for 100.

DM_e % / 100: Measure of the average shunting line of the percentile error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ, divided for 100.

DP_e % / 100: Measure of the shunting line standard of the percentile error, calculated only for the points whose result obtained with the inferrer - or the copying - and with the simulator they differ, divided per 100.

≠ %: Percentile amount of points whose result obtained with the inferrer - or the copying - and with the simulator they differ.

max error: absolute maximum error obtained.

error % max / 100: maximum percentile error obtained, divided for 100.

march error %: Percentile amount of points with the error between the inferred march and the correct one. The error can be null, equal the 1 march, 2 marches, 3 marches, 4 marches, or 5 marches.

7. Conclusion

Comparing the data in the columns of tables, one perceives that, in a general way, as the number of data that compose the mass of data increases, the inferrer performance improves. Comparing tables of the results of the inferrers (Tab. (2), Tab. (3) and Tab. (4)) to the Tab. (1) a general improvement in the indices is perceived. We still evidence a reduction in the biggest errors of inference of the marches in relation to the results displayed in the Tab. (1), besides the reduction of the maximum errors and the maximum percentile errors. As it will be seen ahead, the difficulty in generating random numbers with truly uniform distribution directly affected the quality of the results in tables, so there that was not clear the difference of performance between the 3 tested inferrers, what also hindered the optimization of them. The Tab. (5) compares the results obtained using the discrete inferrer, on the same organized mass of 150 data, tested using 2 different samples, both of size 1000. The results of the Tab. (5) give an idea of the high level of variation of the results, due to low quality of the generator of "random" data used.

The results obtained with the inferrer still are very poor for a good command of the marches and the clutch of a real vehicle. However, the number of data stored in the mass of data still is really very small for a good approach of the function. When distributing the data stored for all the input space, the total amount of 500 data gives an average of $500^{1/4} = 4,7$ data in each direction, what certainly is a very low number of data to satisfactorily approach functions with the level of complexity used here. A so small mass of randomly dispersed data, with distribution uniform, was not capable to catch all the details of behavior of the used functions, however, as the tables of the previous section demonstrate, the developed inferrers are capable to extract relevant information of the behavior of the function and to use them in order to surpass a simple memory. That is, the algorithm demonstrates to execute intelligent tasks once its output surpasses the simple imitation of the stored information. Another advantage of the algorithm, is that the organized mass of data depends only on the input data, that is, once organized a enough amount of data does not matter how many output we will need, the organized and stored information will serve for all the output. The Matlab program, used to implement this algorithm, showed to be not trustworthy, inadequate and of insufficient performance for the perfectioning of the developed algorithm. Thus, in the next stages in the perfectioning of the algorithm it is necessary the migration for another faster, useful, trustworthy and safer system.

8. References

YEN, John, PFLUGER, Nathan, 1995, “A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation”, IEEE Trans. Systems, Man and Cybernetics, vol. 25, no. 6, pp.971-978.

YIP, Percy P. C., PAO, Yoh-Han, 1994, “A guided evolutionary simulated annealing approach to the quadratic assignment problem”, IEEE Trans. Systems, Man and Cybernetics, vol. 24, no. 9, pp.1383-1387.

HSIN, Hsi-Chin, LI, Ching-Chung, SUN, Mingui, SCLABASSI, Robert J., 1995, “An adaptative training algorithm for back-propagation neural networks”, IEEE Trans. Systems, Man and Cybernetics, vol. 25, no. 3, pp.512-514.

LI, Han-Xiong, GATLAND, H. B., 1995, “A new methodology for design a fuzzy logic controller”, IEEE Trans. Systems, Man and Cybernetics, vol. 25, no. 3, pp.505-512.

QIAO, Liu, SATO, Mitsuo, TAKEDA, Hiroshi, 1995, “Learning algorithm of environmental recognition in driving vehicle”, IEEE Trans. Systems, Man and Cybernetics, vol. 25, no. 6, pp.917-925.

AHSON,Syed I., 1995, “Petri Net models of fuzzy neural networks”, IEEE Trans. Systems, Man and Cybernetics, vol. 25, no. 6, pp.926-932.

FAN, Kuo-Chin, LUI, Po-Chang, 1994, “Solving find path problem in mapped anvironments using modified A* algorithm”, IEEE Trans. Systems, Man and Cybernetics, vol. 24, no. 9, pp.1390-1397.

LEITE, Leonardo M. S. C., 2002, “Algoritmo para controle do sistema de transmissão de um automóvel: Inferência direta sobre uma massa de dados”, IME, Master Course Monograph presented to the Course of Mestrado in Systems and Computation of the Military Institute of Engineering.

9. Copyright Notice