Visual-FIR: A new platform for modeling and prediction of dynamical Systems

Antoni Escobet*, Àngela Nebot, François E. Cellier*****

* Departament ESAII, Universitat Politècnica de Catalunya. EdificiMN2–Campus de Manresa, Av. Bases de Manresa 61-73, Manresa 08240, Spain. *Toni@eupm.upc.es*
** Departament LSI, Universitat Politècnica de Catalunya. Mòdul C6 Campus Nord, Jordi Girona Salgado 1-3, Barcelona 08034, Spain. Angela@lsi.upc.es
*** ECE Department, University of Arizona. P.O. Box 210104, Tucson, AZ 85721-0104, U.S.A. *Cellier@ece.arizona.edu*

ABSTRACT[1]

In this research, a new platform for the Fuzzy Inductive Reasoning (FIR) methodology has been designed and developed under the Matlab environment. The new tool, named *Visual-FIR*, allows the identification of dynamic systems models in a user-friendly environment. FIR offers a model-based approach to modeling and predicting either univariate or multivariate time series. Previous uses of FIR had demonstrated the high potential of this qualitative modeling and simulation methodology in effectively dealing with applications from various areas such as control, biology, and medicine. However, the available implementation of FIR was such that new code had to be developed for each new application studied, reducing considerably the interest in this methodology for the occasional user, and making it tedious even for expert programmers. Visual-FIR resolves this limitation, and offers a high efficiency implementation of the FIR methodology. Furthermore, the Visual-FIR platform adds new features to previous implementations, increasing the overall capabilities of the FIR methodology.

Keywords: Fuzzy Systems, Inductive Reasoning, qualitative modeling of dynamical systems.

INTRODUCTION

FIR is a pattern-based modeling methodology operating on observations of system behavior rather than structural knowledge. It is able to derive causal qualitative relations between the variables of a system, and to infer future behavior of that system from observations of its past behavior. It is therefore a useful tool for modeling and simulating systems, for which no *a priori* structural knowledge is available, including systems from soft sciences [Gómez *et al.,* 2001; Jensen *et al.,* 1999; Nebot *et al.,* 1998].

[1] The research presented in this paper was supported by the DPI2002-030225 CICYT project

The FIR software kernel is coded in C. Since FIR operates on data matrices, it is natural to embed the FIR software in a Matlab toolbox for enhancing the ease, with which the software can be used. In the past, the FIR toolbox was designed around the classical Matlab programming interface. When using FIR, the modeler had to write M-functions, in which the various FIR modules were called one after the other. Since FIR can be used in many different ways, the user interface was kept at a relatively low software level. Consequently, FIR programs often consisted of hundreds of lines of Matlab code. This approach was error prone, and consequently, users found it difficult to make use of the FIR methodology.

In the research effort described in this paper, a new platform for the FIR methodology was designed and developed under the Matlab environment. The new tool, named *Visual-FIR*, allows the identification of dynamic systems models in a user-friendly form-driven environment.

Experiences with the previous user interface have shown that many application programs vary relatively little one from another. They differ in where they draw the data from, how many data files are being used as inputs, and how many output data points are to be predicted. Hence it seemed reasonable to design a table-driven user interface that offers sufficient flexibility to be used in many different applications, yet protects the users from having to design and debug programs of their own.

THE FIR METHODOLOGY

The FIR methodology is composed of four main processes, namely: fuzzification (*recoding*), qualitative modeling (*optimal mask search*), qualitative simulation (*prediction*), and defuzzification (*regeneration*).

Fuzzification

FIR is fed with data measured from the system under study that are then converted to fuzzy information by means of the

recode function. The *recode* function converts quantitative values into qualitative triples, i.e., class, fuzzy membership, and side values. The class value represents a discretization of the original real-valued variable. The fuzzy membership value denotes the level of confidence expressed in the class value chosen to represent a particular quantitative value. Finally, the side value tells us whether the quantitative value is to the left, to the right, or in the center of the peak value of the membership function. The side value, which is a peculiarity of the FIR methodology, since it is not commonly introduced in fuzzy logic, is responsible for preserving the complete knowledge in the qualitative triple that had been contained in the original quantitative data value.

In order to convert quantitative values to qualitative triples, it is necessary to provide to the *recode* function the number of classes into which the definition domain of each variable is going to be divided, as well as the landmarks that separate neighboring classes from each other. Once this information has been provided, the *recode* engine of FIR is capable of automatically fuzzifying the quantitative data values using either Gaussian or triangular fuzzy membership functions.

Qualitative Modeling

The *optimal mask* function of the FIR methodology is responsible for finding causal spatial and temporal relations between variables that offer the best likelihood for being able to predict the future system behavior from its own past, thereby obtaining the best model (called a *mask* in the FIR terminology) that represents the system.

At this point, the continuous trajectory behavior recorded from the system has been converted to an episodical behavior (qualitative data) by means of the *recode* function. In the process of modeling, it is desired to discover causal relations among the variables that make the resulting state transition matrices as deterministic as possible. A mask represents a possible relation among the qualitative variables. Let us introduce the concept of a mask by means of a simple example composed of two inputs, u_1 and u_2, and one output, y.

	u_1	u_2	y
t-2δt	-1	0	-2
t-δt	0	-3	0
t	-4	0	+1

Figure 1: Example of a FIR mask

The negative elements in the matrix of figure 1 are referred to as *m-inputs* (mask inputs), which denote input arguments of the qualitative functional relationship. They can be either inputs or outputs of the system to be modeled, and they can have different time stamps. The above example contains four *m-inputs*. The sequence in which they are enumerated is immaterial. The single positive value denotes the *m-output*, and the zero elements represent unused connections. In the above example, the first *m-input* corresponds to the input variable u_1 two sampling intervals back, $u_1(t-2\delta t)$, whereas the second *m-input* refers to the output variable y two sampling intervals into the past, $y(t-2\delta t)$, etc.

A mask denotes a dynamic relationship among qualitative variables. It has a certain number of rows, the depth of the mask. It represents the temporal domain that can influence the output. Each row is delayed relative to its successor by a time interval of δt representing the time lapse between two consecutive samplings. How is a mask found that, within the framework of all allowable masks, represents the most deterministic state transition matrix? This mask will optimize the predictiveness of the model. In the FIR methodology, the concept of a *mask candidate matrix* has been introduced. A mask candidate matrix is an ensemble of all possible masks, from which the best is chosen by a mechanism of exhaustive search. Some other search strategies, such as genetic and classical search tree algorithms, have also been implemented.

The *optimal mask* function searches through all legal masks of complexity two, i.e., all masks with a single *m-input*, and finds the best one; it then proceeds by searching through all legal masks of complexity three, i.e., all masks with two *m-inputs*, and finds the best of those; and it continues in the same manner until the maximum complexity has been reached. In all practical examples, the quality of the masks will first grow with increasing complexity, then reach a maximum, and then decay rapidly. Each of the possible masks is compared to the others with respect to its potential merit. The optimality of the mask is evaluated with respect to the maximization of its forecasting power that is quantified by means of a quality measure, based mainly on the Shannon entropy.

Qualitative Simulation

Once the best model (mask) has been identified, it can be applied to the qualitative data matrices that were previously obtained in the *recoding* process, resulting in a *pattern rule base* that, in the FIR terminology, is called the *behavior matrix*. Once the behavior matrix and the mask are available, a prediction of future output states of the system can take place using the FIR inference engine. This process is called qualitative simulation.

The FIR inference engine is based on a variant of the *k*-nearest neighbor rule, i.e., the 5-NN pattern matching algorithm is the core of the FIR inferencing process. The forecast of the output variable is obtained by means of the composition of the potential conclusion that results from firing the five rules, whose antecedents best match the actual state.

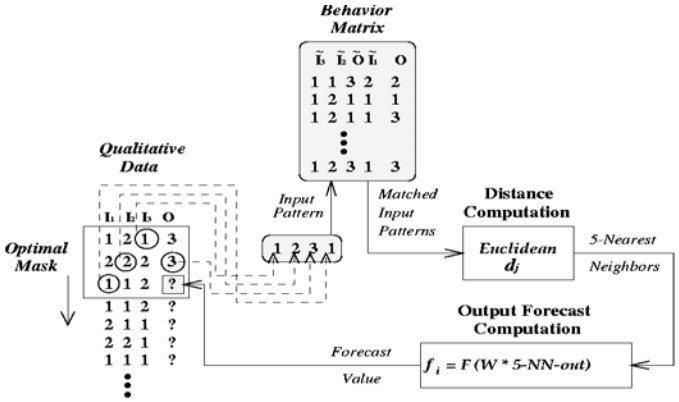

Figure 2: Qualitative simulation process diagram

The *prediction* procedure is presented in the diagram of figure 2 (with an example containing three inputs and one output). The mask is placed on top of the qualitative data matrix in such a way that the *m-output* matches with the first element to be predicted. The values of the *m-inputs* are read out from the mask, and the behavior matrix (pattern rule base) is used to determine the future value of the *m-output*, which can then be copied back into the qualitative data matrix. The mask is then shifted further down by one position to predict the next output value. This process is repeated until all the desired values have been forecast. The qualitative simulation process predicts an entire qualitative triple from which a quantitative variable can be obtained whenever needed. The *prediction* process works as follows, the membership and side functions of the new input state (input pattern in figure 2) are compared with those of all previous recordings of the same input state contained in the behavior matrix. For this purpose, a normalization function is computed for every element of the new input state, and a distance formula is used to select the 5 nearest neighbors, the ones with the smallest distances, that are used to forecast the new output state. Several normalization and distance functions are available in the implementation of the FIR methodology.

The contribution of each neighbor to the estimation of the prediction of the new output state is a function of its proximity. This is expressed by giving a distance-weight to each neighbor, as shown in figure 2. The new output state values can be computed as a weighted sum of the output states of the previously observed five nearest neighbors.

Defuzzification

Regeneration is the inverse function of *recode*. It converts qualitative triples into quantitative values. As has been mentioned earlier, no information is lost in the process of fuzzification. The qualitative triple contains exactly the same information as the original quantitative value, and it is thus possible to regenerate the quantitative value from the qualitative triple precisely. For a deeper insight of the FIR methodology, the reader is referred to [Cellier *et al.,* 1996].

VISUAL FUZZY INDUCTIVE REASONING PLATFORM (Visual-FIR)

In this section, the Visual-FIR platform is described by means of an application from biology, concerning shrimp farming. The goal is to identify growth models for occidental white shrimp (*Penaeus vannamei*) in semi-intensive farming. A growth model is essential to predicting, how the shrimp will grow, and therefore, to plan the best seeding and harvesting strategies that will optimize the profit obtained. In this paper, data collected from a farm in Sinaloa (Mexico) during the summer rainy season is used to identify the FIR models and to validate them.

The main screen of the new platform is invoked by means of the ***Visual-FIR*** command issued from within the Matlab environment. The four main processes of the FIR methodology, i.e. recode, optimal mask, prediction, and regeneration, described earlier, are then displayed as shown in figure 3.

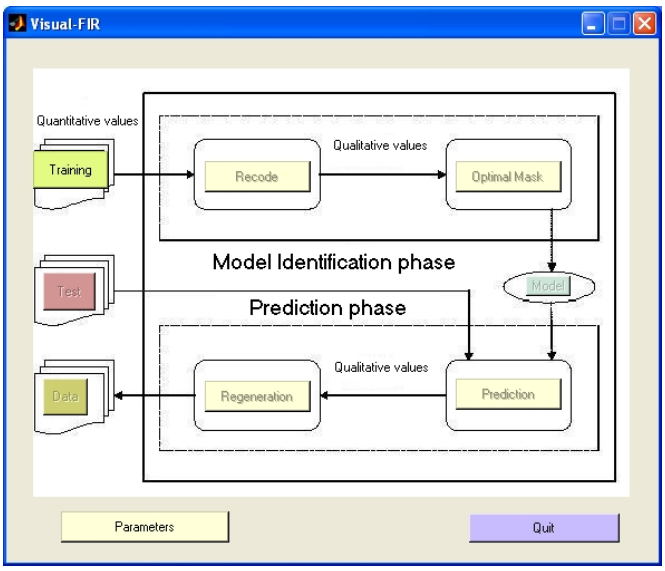

Figure 3: Visual-FIR main screen

The upper half of figure 3 represents the *model identification phase*, whereas the lower half corresponds to the *prediction phase*, during which the model that had previously been identified is used to estimate the future behavior of the system.

Model identification phase

In order to identify the best model from the recorded data, the following steps need to be performed sequentially: configuration of the ***parameters***, loading of the ***training*** data, ***recoding*** the data, and identification of the ***optimal mask***. Each of these steps corresponds to a specific button in the main screen that is enabled sequentially following the

course of the events. Once all these steps have been completed, the full FIR model composed of the optimal mask matrix and the pattern rule base can be displayed pushing the ***model*** button.

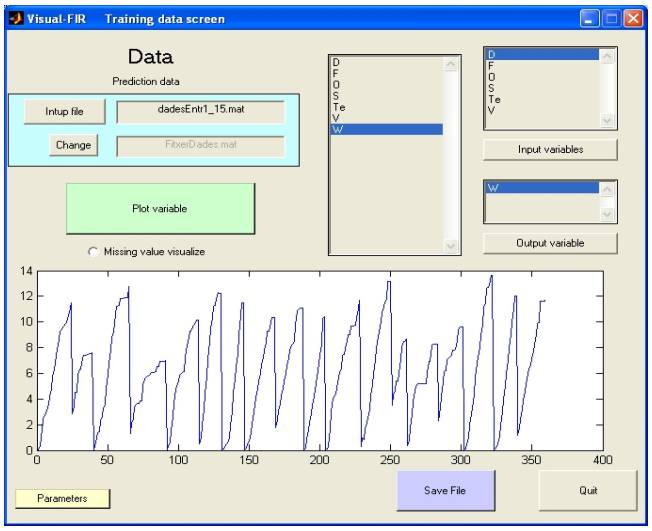

Figure 4: Training data screen

Parameter setting: When the ***parameters*** button is pushed, a screen with a set of 12 global parameters is displayed. All of the parameters can be modified, thereby enabling different execution options of the FIR methodology, e.g. Gaussian *vs.* triangular shaped membership functions, proximity *vs.* similarity confidence measure, different prediction distance measures and weight equations, different ways to compute the quality of the mask, different optimal mask search algorithms, etc. There is also a *miss_data* parameter that allows the user to indicate the presence of missing data in the set of available records. Visual-FIR is able to deal with data that contains missing elements effectively and efficiently. All parameters have associated default values that allow non-expert users to proceed without changing any of them. Notice that the parameters will be accessible from all screens, yet, only those parameters will be enabled at any one time that are related to the stage of the modeling and simulation life cycle, in which the ***parameters*** button has been activated.

Training data: Before starting with the model identification phase, the training data set(s) need(s) to be loaded. This step is accomplished by using the ***training*** button. The data-loading screen presented in figure 4 is then displayed. Initially, the only options enabled are the loading of the training data file (***input file*** button) and ***change*** the name of the output file. In the shrimp farming application, the file "dadesTest1_15.mat" contains the complete set of training data. Seven variables are loaded from that file, i.e., **D**ensity (shrimp/m²), **F**eed (%), **O**xygen (ppm), **S**alinity (ppt), **Te**mperature (°C), **V**isibility (cm), and **W**eight (kg). The user selects the input and output variables that are then displayed at the right-top boxes of the screen. In this case, the input variables are density, feed, oxygen, salinity, temperature and visibility, whereas the output variable is the weight of the shrimp.

Once the input and output variables have been defined, they can be plotted, one by one, in the central graph of the training data screen. In the figure, the output variable is displayed. 18 separate growth periods were used for training. They are presented to FIR as a single data stream with missing data blocks separating the individual growth periods. Before exiting, the data need to be saved using the ***save file*** button.

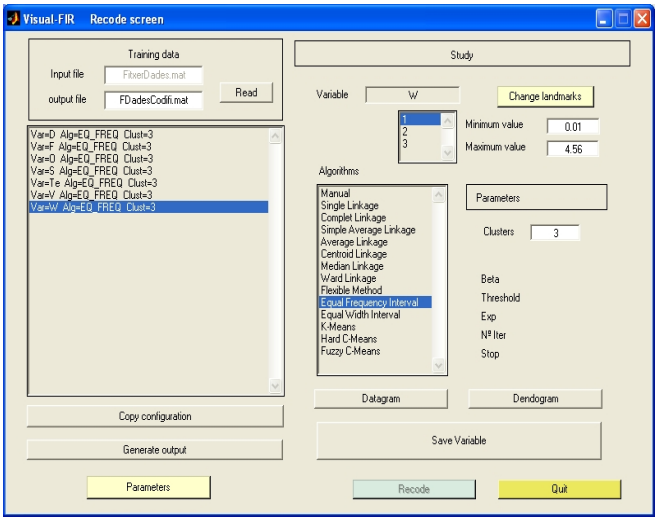

Figure 5: Recode screen

Recode: The data is now ready to be fuzzified (converted from quantitative to qualitative data). The fuzzification box is displayed when the user pushes the ***recode*** button in the Visual-FIR main screen. Figure 5 presents the configuration of the new screen. The first thing to be done is load the data that has been saved in the previous step. In order to do so, the ***read*** button needs to be pressed. The list of all the variables (inputs and output) is then displayed in the left half of the screen. All of these variables have associated, by default, three classes and the equal frequency interval (EQ_FREQ) discretization algorithm. However, a large number of discretization (clustering) algorithms, both hierarchical and non-hierarchical, are offered to the user as shown in the right half of the screen. When an algorithm is selected, the parameters required by that algorithm are enabled in the parameters box. In figure 5, the equal frequency interval algorithm is selected. This algorithm permits only a single parameter, namely that determining the number of classes, to be changed by the user.

Each system variable listed on the left hand side of the recode data screen can be discretized separately, using any of the clustering algorithms listed on the right hand side of

the screen, allowing an *ad hoc* discretization of each variable. The construction of datagrams and dendograms are options offered to the user in order to facilitate the process. Once all the variables have their own clustering strategy defined, the ***generate output*** button computes the number of classes and its corresponding landmarks (borders between neighboring classes), information that is needed by the recode function. Then, the ***recode*** button performs the conversion of the quantitative data into qualitative triples, generating the class, membership, and side value for each of the quantitative data entries in the training data set. The screen can then be closed using the ***quit*** option. For the application at hand, the default values have been used in the recode process.

Optimal mask: Once the data has been recoded, the qualitative modeling process (optimal mask search) can take place. This is accomplished by pushing the ***optimal mask*** button on the main screen. The screen displayed in figure 6 is then shown.

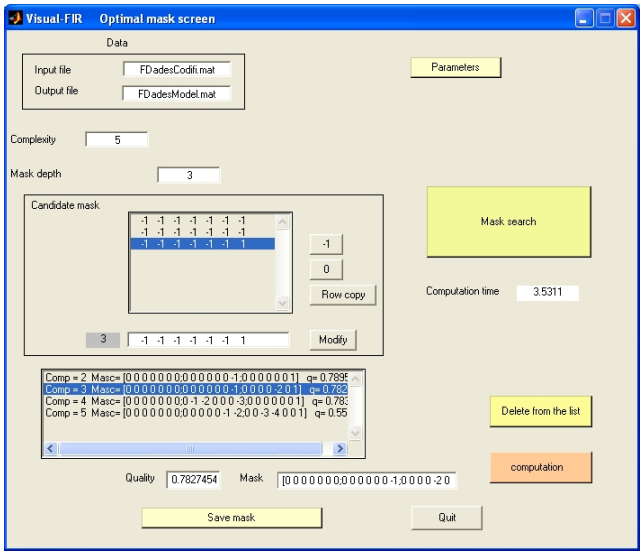

Figure 6: Optimal mask screen

As described previously, the optimal mask search starts from the definition of a mask candidate matrix. In this screen, the maximum complexity of the mask and the mask depth need to be set, in order for Visual-FIR to be able to generate a mask candidate matrix. A complexity of 5 and a depth of 3 are the values chosen by default. Once these parameters have been set, the mask candidate matrix is displayed (as can be seen in figure 6). Forbidden connections (zero values) can be introduced to the mask candidate matrix in suitable places. Once the mask candidate matrix is ready, the mask search starts by pressing the ***mask search*** button. The exhaustive search is the default algorithm used; however, more efficient algorithms (such as genetic algorithms or decision trees) can be chosen using the ***parameters*** screen, if the estimated computational time (see figure 6) is larger than desired.

Once the optimal mask search process has been completed, the list of the best masks found for each complexity together with their quality measures is presented in the lower part of the screen. The user can then delete unsuitable sub-optimal masks by pushing the ***delete from the list*** button. There is also available the option to compute the quality of any mask by introducing it in the "mask" box and pressing the ***computation*** button. Once the mask that the user wants to explore as part of the system model is selected, the ***save mask*** button needs to be used. At this moment, the mask is used to extract the pattern rule base from the recoded training data. The mask selected plus the pattern rule base (behavior matrix) compose the FIR model of the system. To quit the optimal mask search screen, the quit button is available. In the current application, the mask selected by the user is the sub-optimal mask of complexity 3 with two m-inputs, $w(t-\delta t)$ and $Te(t)$, and with an associated quality measure of 0.782.

Prediction phase

Once the FIR model is available, the user can proceed to its validation by using the model to predict a test data set. Once the model is validated, it can be used to predict the future behavior of the system. The prediction phase is composed of four steps using Visual-FIR's main screen (see figure 3), i.e., loading the ***test*** data, ***prediction*** of the system behavior, data ***regeneration***, and result ***visualization***.

Test data: In order to load the test data, the ***test*** button needs to be pressed. The same screen presented in the training data step is then displayed (see figure 4). Now, the input variables and output variable are selected automatically following the selection that the user had made for the training data set. The functioning of this screen is exactly the same as explained earlier in the training data step; however, when the ***save file*** button is pushed, the test data is discretized using the same parameters as defined in the recode process.

Prediction: Once the qualitative test data is available, the prediction can take place. To this end, the ***prediction*** button of the main screen must be pushed. The forecasting algorithm described in the previous section is then executed using the options selected in the global parameters window.

Regeneration: The next step on the Visual-FIR main screen is the regeneration of the predicted test data. This is accomplished by pressing the ***regeneration*** button of the main screen of figure 3. Figure 7 is then displayed, and the defuzzification (inverse process of fuzzification) is computed, when the ***regenerate*** button of this screen is pushed. The user can exit the screen using the ***quit*** button.

The last thing to be done is visualize the results. This is done in the next step.

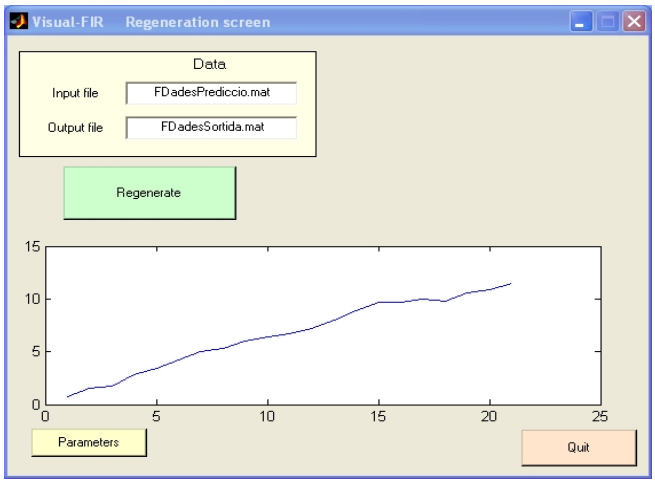

Figure 7: Regeneration screen

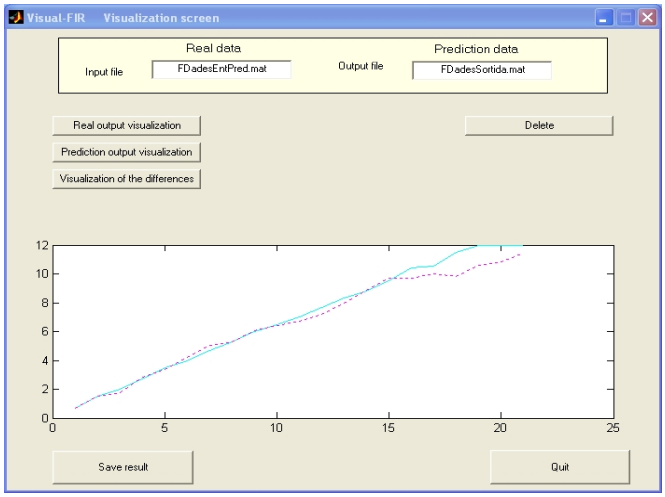

Figure 8: Visualization screen

Result visualization: At this point all of the prediction and regeneration processes have been completed, and therefore, the user can visualize the results and check the error obtained when using the FIR model for forecasting future behavior of the system. To this end, the ***visualization*** button on the Visual-FIR main screen needs to be pushed. The visualization screen presented in figure 8 is then displayed. As can be seen from figure 8, the user can visualize the real output signal, the predicted signal and the difference between the two signals. Each time the user pushes one of these options, the corresponding signal is added to the display area. If needed, the overall plot can be deleted using the ***delete*** button. The mean square error (mse) in percentages computed between the real and the predicted values is also shown. For the application at hand, the real (continuous) and predicted (dashed) weight signals are displayed. The mse error obtained with the FIR model is 2.897%, an excellent result if compared with the 20% obtained for the same problem with classical statistical techniques [Carvajal and Nebot, 1998].

CONCLUSIONS

In this paper, the Visual-FIR platform has been presented. This new tool offers access to the Fuzzy Inductive Reasoning methodology in an effective and user-friendly manner. The FIR methodology offers a model-based approach to predicting dynamical systems with very good results when applied to soft sciences applications. Visual-FIR has the advantage, in comparison with previous FIR implementations, that no code needs to be developed for each new application studied. Visual-FIR offers a table-driven environment that provides an intuitive and easy-to-use access to the FIR methodology. Clearly, a price has to be paid for this enhanced user comfort. A table-driven user interface can never offer the complete flexibility of a programming interface. Yet, the Visual-FIR platform should be flexible enough for most applications. It is hoped that the new interface will open up the FIR methodology to new users, as it makes it much easier to apply the methodology to new applications, and as it makes it much faster to try out different algorithms on one and the same application, something that hitherto always required a substantial amount of reprogramming.

References

Carvajal, R. and A. Nebot (1998), "Growth Model for White Shrimp in Semi-intensive Farming using Inductive Reasoning Methodology," *Computers and Electronics in Agriculture*, **19**:187-210.

Cellier, F.E., A. Nebot, F. Mugica, and A. de Albornoz (1996), "Combined Qualitative/Quantitative Simulation Models of Continuous-Time Processes Using Fuzzy Inductive Reasoning Techniques," *Intl. J. of General Systems*, **24**(1-2):95-116.

Gómez, P., A. Nebot, F. Mugica, and F. Wotawa (2001), "Fuzzy Inductive Reasoning for the Prediction of Maximum Ozone Concentration," *Proceedings of the 13th European Simulation Symposium (ESS'01)*, Marseille, France, October 18-20, 2001, p. 535-542.

Jensen, E.W., A. Nebot, P. Caminal, and S.W. Henneberg (1999), "Fuzzy Inductive Reasoning Applied to Identify a Causal Relationship among Haemodynamic Parameters, Auditory Evoked Potentials and Isoflurane," *British Journal of Anaesthesia*, **82**(1):25-32.

Nebot, A., F.E. Cellier, and M. Vallverdú (1998), "Mixed Quantitative / Qualitative Modeling and Simulation of the Cardiovascular System," *Computer Methods and Programs in Biomedicine*, **55**:127-155.