Software Review

Fuzzy Systems Toolbox—PWS Publishing Company and **Fuzzy Logic Toolbox**—The MathWorks, Inc. *Reviewed by Lawrence O. Hall and Richard J. Hathaway*

MATLAB® (derived from matrix laboratory) is a technical computing environment whose basic data element is a self-dimensioning matrix. It combines fast numerical capabilities with excellent graphics using a command syntax that is quite intuitive. MATLAB® is useful for developing and modifying algorithms, particularly those which are heavy in matrix operations. Many of the MATLAB® commands are based on execution of function programs contained in "M-files," which contain source code. This yields a highly open and extensible environment whereby the user can redefine certain existing commands or create new ones by suitably editing the M-files.

MATLAB® is a product of The MathWorks, Inc., and numerous sets of application specific routines, called "toolboxes," have been developed and marketed by that company for use with MATLAB®. Here, we review the two commercially available MATLAB® toolboxes for creating and using fuzzy inference systems. They are the "Fuzzy Systems Toolbox," by PWS Publishing Company, and "Fuzzy Logic Toolbox," by The MathWorks, Inc.

We examined both the Macintosh and UNIX versions of the two toolboxes, but all the computational experiments described in the following were done using the UNIX version, which was executed on a shared SUN Sparc 20 computer with graphics, transmitted across a reasonably heavily used local ethernet network.

I. The Fuzzy Systems Toolbox (FST)

This toolbox provides a command line approach to building fuzzy sets and fuzzy rule-based systems. A continuous fuzzy set can be represented in either of two ways using FST. The more general option is to specify n support and membership grades for the fuzzy set using a pair of $1 \times n$ arrays. Piecewise linear interpolation between successive (support, grade) points gives the complete membership function. For example, the support array $s = [1\ 2\ 3\ 4]$ and grade array $g = [0\ .2\ .6\ 0]$ describe a fuzzy set that assigns the grade of 0.4 to the support value 2.5. The less general, but more convenient, option of specifying a fuzzy set is through the use of parameters. The six parametric families available allow users to specify singleton or interval crisp sets and fuzzy sets with membership functions in the shape of triangles, trapezoids, "bumps," and "flat bumps." Fuzzy sets corresponding to numbers "near" a given number, or "large" relative to a given support, are easily obtained. It is possible to plot the antecedent fuzzy sets when they are all of the same type. It was unclear how to plot on a single graph a group of fuzzy sets of different types.

FST has a wide variety of operators that can be applied to the fuzzy sets. There are hedge operators, such as "very," "extremely," "exactly," etc., and others for defuzzification (by maximum or minimum grade, or centroid), and α- and β-cuts. The ubiquitous operators min and max are available for conjunction and disjunction of fuzzy sets, as well as Yager's class of intersection/union operators and the algebraic product/sum and bounded difference/sum operators. This provides flexibility in building fuzzy systems. Negation is strong negation. Mamdani [1] style inference is supported with centroid defuzzification available. It was found to be relatively easy to build a fuzzy rule system with FST, and the results from the system were always as expected.

A gradient descent approach to learning is provided. It can be used to tune existing rules. To learn rules from scratch an initial choice of antecedent and consequent fuzzy sets must be made; these choices will be crucial. It is unclear exactly how this learning function works, but it was useful in the single small example done. However, careful choices of the learning rate and initial rule set up are necessary for acceptable performance.

The toolbox provides one fuzzy clustering algorithm, the fuzzy c-means algorithm of Bezdek [2]. We ran fuzzy c-means on several data sets with which we have experience, and the results were as expected. It would be nice to have an easy way to view the value of the functional at each iteration. It would also be nice to have easy access to the membership matrix of elements in each cluster. On the several data sets we tried, the clustering algorithm was an order of magnitude slower than both our C version and the Fuzzy Logic Toolbox (described in the next section) version of fuzzy c-means.

Other procedures included in the toolbox, allow users to: 1) include fuzzy controllers in the SIMULINK® environment and 2) easily apply a fuzzy system to decision problems. Additionally, effective demonstration applications are included for pattern recognition, control, model fitting, and decision making. The provided control example of the inverted pendulum is nice for a classroom demonstration.

The documentation for the software was good overall, but it could contain more pointers to some of the missing details for the serious developer. Chapters 1–3 (62 pp.) give the background information needed to build and use fuzzy sets and systems; Chapters 4–6 (68 pp.) cover the major application topics of decision-making, control, and pattern recognition systems; Chapter 7 (24 pp.) briefly discusses training, SIMULINK, and other advanced topics such as OWA operators; Chapter 8 (163 pp.) is a reference for the various commands. There were only a small number of errors in the manual. The FST software and documentation were written by M. Beale and H. Demuth.

In summary, this is a nice and reasonably flexible tool for developing fuzzy rule-based systems or doing fuzzy clustering. Everything worked quite robustly.

II. The Fuzzy Logic Toolbox (FLT)

A major difference between FLT and FST is the user interface. FST must be used from the command line (i.e., typing in commands), while FLT can be used from the command line or through a graphical interface. The graphical interface mode greatly simplifies the building and manipulation of fuzzy systems. The FLT stores all the information necessary to represent a fuzzy inference system (fis) in a single matrix called the "fis matrix" and this information, if desired, can be saved to

Manuscript received July 25, 1995; revised August 1, 1995.

L. O. Hall is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA.

R. J. Hathaway is with the Mathematics and Computer Science Department, Georgia Southern University, Statesboro, GA 30460 USA.

Publisher Item Identifier S 1063-6706(96)00634-0.

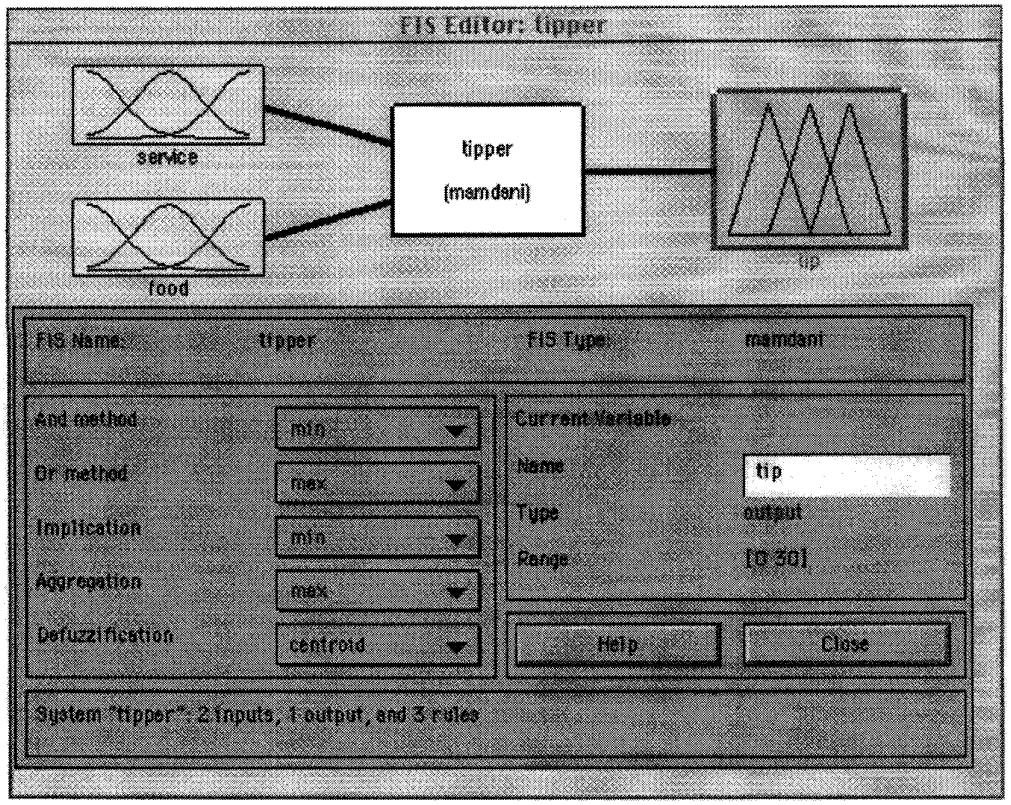

Fig. 1. FLT graphical interface: Main Editor.

disk in a "fis file." Creating new fuzzy systems or changing existing ones can easily be done by editing a fis matrix using the graphical interface.

We will use an illustrative example taken from the documentation to demonstrate the intuitive appearance of the interface of the Macintosh version. The example has to do with deciding on an appropriate tip amount, given certain measured levels of food and service quality. The system is stored in a file called tipper.fis. Entering the command "fuzzy tipper" opens tipper.fis and invokes the main editor portion of the interface shown in Fig. 1 (Figs. 1–3 of the graphical user interface of the Fuzzy Logic Toolbox were obtained as screen captures on a Macintosh computer. The actual interface uses color and is much sharper than what is shown below).

Note that at the level of the system editor in Fig. 1, we can easily select different versions of operators such as "and" (or as it turns out, even easily define new ones). Menu options (not shown) allow additional variables to be easily added. Choices about particular membership functions for particular input or output variables can be made by double clicking on the variable or using a menu option. For example, double clicking on the service variable plot in Fig. 1 brings up the membership function editor shown in Fig. 2.

Using a menu option, additional membership functions can be added to this variable and they can either be custom made or selected from the 11 available parametric families (including families of membership functions built from piecewise linear functions, Gaussian-distribution functions, sigmoid curves, and certain low-order polynomial curves). In Fig. 2, for example, the "poor" membership function for the variable "service" is Gaussian with $\mu = 0$ and $\sigma = 1.5$. Last of all, the rules are edited using the rule editor. This can be invoked from menu options or by suitable double clicking. For this example, the appearance of the rule editor is given in Fig. 3.

Rules can easily be added, deleted, or modified in the rules editor, and normal English can be used (actually, menu options even allow for French or German). Figs. 1–3 should give the reader a clear indication of how intuitive and powerful the graphical interface is. There are also menu options for plots of the fuzzy model surfaces obtained by combinations of any one or two of the input variables, and an informative graphical display that geometrically depicts the evaluation of the fuzzy system for a single input.

Although the graphical interface is usually more convenient, FLT can be used from the command line, although the commands are more aimed at building and manipulating at the fuzzy-system level rather than at the fuzzy-set level. FLT works with fuzzy systems of both the Mamdani and Takagi–Sugeno–Kang types [3], and for the latter, the "anfis" (adaptive network-based fuzzy inference system [4]) routine can be used with suitable data to train and test the parameters of both the antecedent fuzzy sets and the consequent linear models of an existing model. Another routine, "genfis2," uses available data and subclustering techniques to generate, from scratch, a reasonably good fuzzy inference system. We now describe testing of the UNIX version of FLT.

Using the graphical interface, we built up a set of fuzzy rules developed by a prototype of a fuzzy-rule learner. The graphical interface is really a useful tool for building the fuzzy sets of the antecedents, consequents, and the rules themselves. The graphics display might have been faster, but it was coming across the network from a reasonably loaded machine. Overall, the rules were built quickly. Since the domain from which our rules came was unusual we had to resort to a regular editor to get more than nine (maximum allowed via graphical interface) fuzzy sets in some of the antecedent

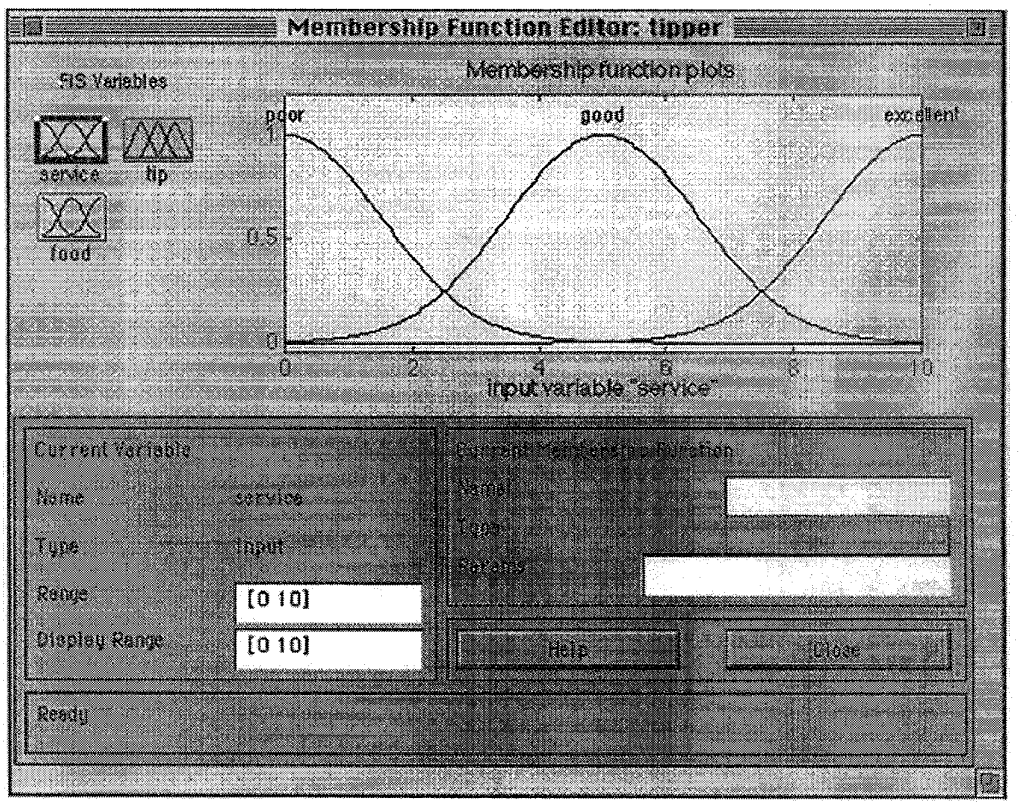

Fig. 2. FLT graphical interface: Membership Function Editor.

fuzzy terms/attributes. However, this presented no real difficulty and all of the sets could be displayed graphically, which was useful in finding a mistake in entering the sets. The rules worked as expected and ran through data quickly. The fact that the Takagi–Sugeno–Kang model of fuzzy inference in addition to the Mamdani model is available is a plus for this system. It allows quite sophisticated fuzzy models to be developed.

We also took the data from which the rules were generated and gave them to genfis2 to attempt to build from scratch a set of fuzzy rules for comparison. With a (genfis2) radius setting of 0.5, acceptable rules were not generated, but with a setting of 0.3 they were. There were six rules generated and they matched our intuition of what "good" rules should look like. Further, when tested on the training data (a difficult set) they had a low residual mean square error, which was impressive. Since you can change the rules after the learning process to "tweak" them toward better performance, the automatic rule generation feature, which did quite well in our test, appears most useful.

There are two clustering algorithms available in the toolbox, including the fuzzy c-means algorithm, which we discuss first. The 2-D interface is really very nice, and many real-world problems contain important projections into 2-D. It is important to be able to access the objective function value, the cluster centers, and the matrix of membership assignments to cluster centers, as well as to influence the fuzziness factor used as an exponent. The builders of the fuzzy logic toolbox made all of them easily available. It was done right. In all tests of fuzzy c-means, the cluster centers found were a very close match to those found by our other implementations. The clustering algorithm was reasonably fast (for a clustering algorithm). It did 100 iterations of a 65 536 pixels $\times$ 3 features magnetic resonance image slice for 10 classes in just over an hour of CPU time. This is on par with our nonoptimized C code.

The other included clustering algorithm does subtractive clustering. Subtractive clustering can be useful in the initialization of other clustering algorithms and for stand-alone clustering when clusters are well separated in feature space. Its use is straightforward and it performed as expected on the well-known Iris data set. There were four clusters found with default values and a radius of 0.5. One of the clusters was the linearly separable class; the two overlapping classes were split among the three remaining clusters. It worked well in conjunction with the anfis routine to build fuzzy rules from data.

Other procedures included in the toolbox allow users to 1) include fuzzy controllers in the SIMULINK environment and 2) build stand-alone code to do fuzzy inference. The set of demonstration applications, all accessible through menu and button options, contains examples from control, time series, signal processing, and clustering. The clustering demo is actually a nice cluster visualization tool that can be used on arbitrary data sets.

The documentation is very well written with only a few minor typos. Chapter 1 (14 pp.) is an interesting, elementary introduction to fuzzy sets and logic. The second chapter (100 pp.) is an effective tutorial on using the software, and Chapter 3 (67 pp.) is the reference for the various commands. The description of the software is done by means of tutorial examples, and we found this to be a helpful approach. The on-line help is also very useful. It is usually, but not always, equal to the manual. For example, the on-line help of the UNIX version does not show you how to display a fuzzy rule set in the graphical interface from the command line, a very minor difference. With the online help one can probably hack their way to a fuzzy rule set without opening the manual, and this is a feature we

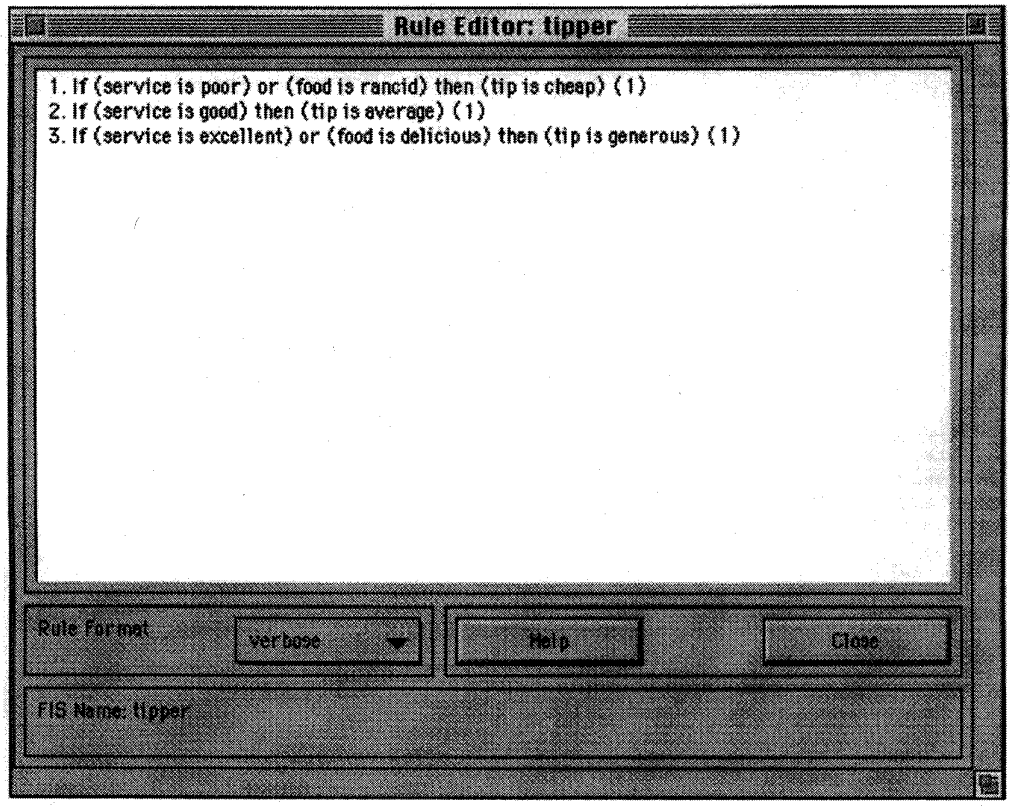

Fig. 3. FLT graphical interface: Rules Editor.

like. The FLT software and documentation were written by J.-S. R. Jang and N. Gulley.

In the process of testing out FLT, we found a couple of bugs and features that were restrictive. The MathWorks was notified and promptly replied with fixes and workarounds. They plan to have a WorldWide Web page for this toolbox up very soon with downloadable bug fixes. Their attitude was most impressive and they have solid fixes for anything that was found problematic. The fuzzy logic toolbox is a very useful and robust addition to MATLAB®. If you want to do development of fuzzy rule-based systems or fuzzy exploratory analysis, the fuzzy logic toolbox is a great vehicle to accomplish such tasks. FLT is the first choice of both reviewers.

REFERENCES

[1] E. H. Mamdani, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Machine Studies,* vol. 7, no. 1, pp. 1–13, 1975.

[2] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum, 1981.

[3] M. Sugeno, *Industrial Applications of Fuzzy Control.* Amsterdam: Elsevier Science, 1985.

[4] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man, Cybern.,* vol. 23, no. 3, pp. 665–685, 1993.