Implementation of a probabilistic-fuzzy modelling system in Matlab

Katarzyna Błaszczyk, Opole University of Technology

Abstract

This article is about a toolbox created in the Matlab environment which implements a probabilistic-fuzzy system to model the representation of linguistic knowledge in the form of IF-THEN (1) rules along with weights determining fuzzy events probability [2-4]. A toolbox is based on three different functions: *newmod* (Tab.1), *genreg* (Tab. 2), *infermod*. These functions allow the creation of a new model structure as in Fig. 1. They will also generate a knowledge base along with the usage of empirical data and fuzzy inference on the basis of the created model. The influence of different parameters calling a function on a structure and the complexity of the calculations of the model were studied. The rules of the optimization of the programes code from the point of lasting time of calculations were also described (Tab. 5-6).

1. Introduction

Recently there has been a growing interest in research concerning the development and implementation of fuzzy modelling methods. The proof of this is the growing number of publications and program tools designed for such implementations [1]. The advantage of the fuzzy modelling techniques is the possibility of the implementation in uncertain conditions and imprecise information. On the basis of empirical data appropriate models of non-linear objects are created also in the case when mathematical description is difficult or impossible.

Fuzzy models allow in a comprehensible and characteristic of people way, in the form of IF-THEN rules, to create the activity of a given system. One of the methods of the presentation of linguistic knowledge is the probabilistic-fuzzy rule-based model [2-4]. In the above mentioned model the idea of knowledge base is to define the reliability of the rules which comprise marginal and conditional probability of fuzzy events. The advantage of this model is the possibility of implementing it in the stochastic processes [2,3] for which most of the models is not precise enough. However, there is a possibility to obtain the outputs on the basis of the probable distribution of events. Then, seemingly logical ambiguous rules acquire the meaning in the inference process of the system. The disadvantage of the method is the complexity of calculations especially when a large number of analysed values of the process is conducted and a broad range of defined linguistic variables is present. In order to reduce this disadvantage, in [7] the implementation of one of the data mining – association rules was considered.

This article will present an attempt to implement the described system in the Matlab calculation environment with taking into consideration the time optimization of the program code. In order to conduct the calculations a processor Intel Pentium M 1.73 GHz with 1.48 GB Ram, and Matlab 6.0 were used.

2. Probabilistic-fuzzy models

The basis of the probabilistic-fuzzy modelling of MISO system is the presentation of the knowledge base in the form of file rules as follows [4-6]:

$$\begin{aligned} &w_j(\mathbf{IF}\ \ \mathbf{x}\ is\ A_j^i\ \ \mathbf{THEN}\ \ y\ is\ B_{1/j}\ (w_{1/j}) \\ &\mathbf{ALSO}\ \ y\ is\ B_{2/j}\ (w_{2/j}) \\ &\quad\ldots \\ &\mathbf{ALSO}\ \ y\ is\ B_{m/j}\ (w_{m/j})) \end{aligned} \tag{1}$$

where

$\mathbf{x}=(x_1,x_2,\ldots,x_n)^T$ – vector of input variables, $\mathbf{x}\in X_1\times X_2\times\ldots\times X_n\subset R^n$,

y – output variable of the model, $y\in Y\subset R$,

A^i_j – linguistic value of input variables, $i=1,\ldots,n$, $j=1,\ldots,J$,

$B_{l/j}$ – linguistic value of the output variable, $l=1,\ldots,m$,

w_j – weight of j-th file rule,

$w_{l/j}$ – weight of elementary rule.

Symbols X_i, $i=1,\ldots,n$ and Y state spaces of the input variables and the output variable. The discretization of X_i, Y took place in disjoint intervals of the variable values respectively $\mathbf{a}_i=(a^i_1,\ldots, a^i_K)$ and $\mathbf{b}=(b_1,\ldots, b_K)$.

Linguistic values of the model are identified with fuzzy sets according to Zadeh's definition [11]. They are defined by membership functions. In the case of disjoint intervals of the variable values degrees of membership are described as $\mu_{A_j^i}(a_k^i) \in [0,1]$, k=1,…,K for the input variable and $\mu_{B_{l/j}}(b_k) \in [0,1]$ for the output variable. However dependence always takes place:

$$\sum_{j=1}^{J} \mu_{A_j}(a_k) = 1, \text{ k=1,…,K}, \qquad (2)$$

where A_j, j=1,…J determines fuzzy sets specified for the one linguistic variable.

Calculating the values of the rules weights (1) the definition of marginal and conditional probability of fuzzy events has to be accomplished according to Zadeh`s definition [12]. Then, for a SISO model, the probability of the occurrence of the single fuzzy event A_j for the antecedents (e.g. "x is high") is:

$$P(A_j) = \sum_{k=1}^{K} P(x \in a_k)\mu_{A_j}(a_k). \qquad (3)$$

The probability of the simultaneous fuzzy event occurrence for the A_j antecedents (e.g. "x is high") and B_l consequents (e.g. "y is average") is described as follows [2,4]:

$$P(B_l \cap A_j) = \sum_{m=1}^{K}\sum_{k=1}^{K} p_{mk}(x,y) T(\mu_{A_j}(a_k), \mu_{B_l}(b_m)) \qquad (4)$$

where $p_{mk}(x,y)$, as the probability in the sense $P(x \in a_k, y \in b_m)$, determines the relation of the number of observations (in which the variable x achieves the value of a_k range and the variable y achieves the value of b_m range) to the general number of observations in space X×Y.

Symbol T determines any t-norm operation. Probability of fuzzy evens in the case of MISO model is calculated similarly.

The calculations above allow the weight of the rules to be as follows:

- w_j, that is marginal probability of fuzzy events as (3),
- $w_{l/j}$, that is conditional probability of fuzzy events as $P(B_l / A_j) = \dfrac{P(B_l \cap A_j)}{P(A_j)}$.

An example of a different fuzzy modelling method with reliable structures can be found in [6].

3. Construction of probabilistic-fuzzy models in Matlab

In the Matlab environment, a toolbox implemented a probabilistic-fuzzy system according to the estimations presented in chapter two. It is based on three different functions: *newmod*, *genreg*, *infermod*. They enable the creation of a new model, generate knowledge base with the usage of empirical data and fuzzy inference based on a created model.

3.1. Creation of the new model

The creation of the new model is possible due to *newmod* function. Its calling options are described in Tab. 1. Then, the object of the model based on a structure presented in Fig. 1, is generated. Structure stores essential information concerning stages of fuzzyfication, interpretation of the rules base and defuzzyfication of probabilistic-fuzzy system.

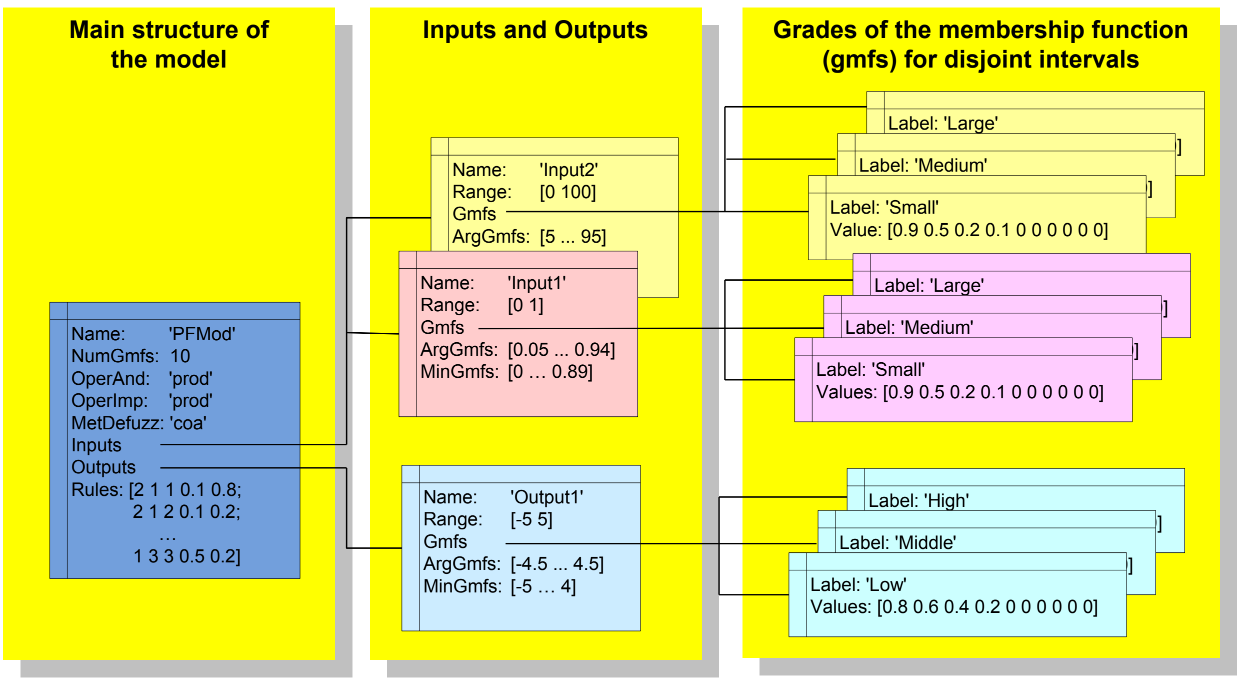

Fig.1. Example of the structure of the probabilistic-fuzzy model in Matlab.

Tab.1.

Specifications of *newmod* function

Syntax	
model=newmod(modName,inX,outX,numGmf,numMf, … typeMf,options)	
Name attribute	**Description**
modName	name of the creating model
inX	matrix of input data
outX	matrix of output data
numGmf	number of disjoint intervals in variable's space (default: 10)
optionMf	vector of membership function options (default: {'trimf' 10})
optionMet	vector of inference options: {'operAnd' 'operImp' 'metDefuzz'} (default: {'prod' 'prod' 'coa'})
optionX	2xN matrix of range variable's values (optional)

The correct choice of membership function depends on the knowledge and experience of experts. The method allows to define the membership degrees for constant intervals of the variable values or it gives the possibility to define them by standard membership functions which are available in Toolbox Fuzzy Logic (Fig. 2) [9]. Function *newmod* creates constant values of membership function (*gmfs*) on the basis of the same membership functions for each variable in the model. In order to differentiate parameters the following functions can be used: *addmod*, *addinp*, *addinpmf*, *addout*, *addoutmf*. Researcher's experiments prove that it is advantageous to use the simplest multiangular membership functions which makes the process of tuning of the fuzzy model easier and they guarantee high accuracy [8].

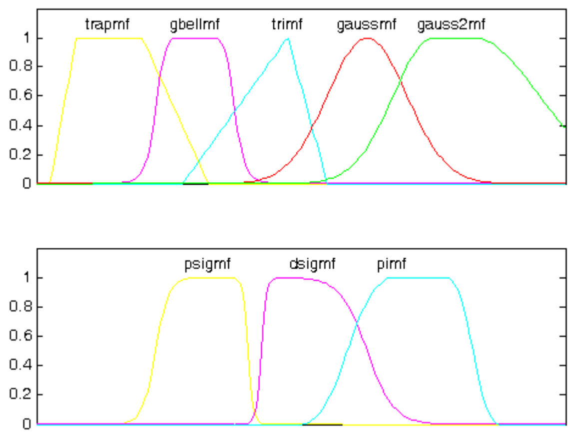

Fig.2. Membership functions in Matlab (cf. [9]).

Fig. 3 presents the transformation mode of membership function into constant degrees for intervals. In each case, fuzzy sets fulfil the conditions of the partition of unity (2) which influences smoothing of the models surface [8].

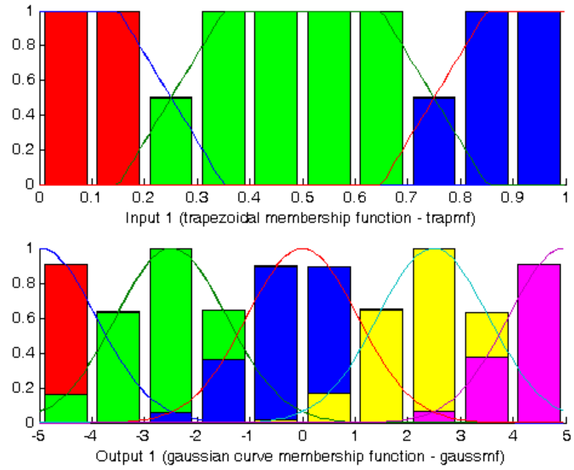

Fig.3. Grades of membership from the standard membership function.

3.2. Generation rules

The above model is deprived of the main component of the structure – rules base. *Genreg* function (Tab. 2) allows to generate the rules in the form of (1) on the basis of experimental data from matrix *inX* and *outX* together with the usage of a chosen t-norm operator.

Tab.2.

Specifications of *genreg* function

Syntax	
model=genreg(model,inX,outX,tNorm)	
Name attribute	**Description**
modName	name of the created model
inX	matrix of input data
outX	matrix of output data
tNorm	operator tNorm used to create rules

In order to use Matlab properties of environment calculations were conducted with the usage of the vector record and multidimensional matrix. Fig. 4, as an example, presents: the scheme of algorithm for the calculations of joint probability of fuzzy events $P(B_l \cap A_j^2 \cap A_i^1)$ for the model with 2 inputs, the product as t-norm operator and discretion of spaces variables for 3 disjoint intervals.

The time of the function execution was reduced several times in relation to calculations done with the usage of the loop. Unfortunately, time complexity of the algorithm is still exponential dependence in relation to the number of the models parameters. Then calculations become ineffective for the models of many variables.

The following Tab. 3-4 and Fig. 5-6 present dependence on the number of generated rules and the lasting time of calculations to different parameters of calling function.

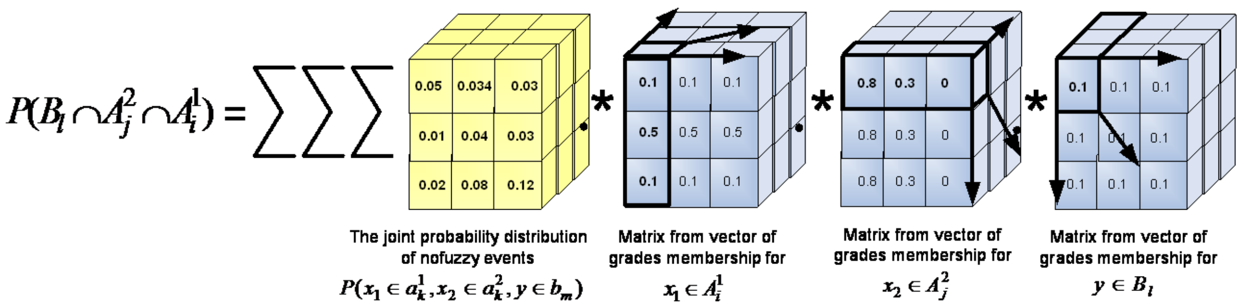

Fig.4. Schematic computation joint probability of three fuzzy events.

Tab.3.

Influence of the membership function of the number of rules and lasting time of calculations (numInputs: 3, numOutputs: 1, numMfs: 10, numGmfs: 10, t-norm: product)

Membership function	Number of elementary rules in the model	Time of the rules generating [s]
Gaussian curve (gaussmf)	10000	45.00
Generalized bell curve (gbellmf)	10000	51.61
Pi-shaped curve (pimf)	2894	42.86
Triangular function (trimf)	7105	49.48
Trapezoidal function (trapmf)	2894	38.83
Difference of two sigmoids function (dsigmf)	10000	53.01
Product of two sigmoids function (psigmf)	10000	52.24

Tab.4.

Dependence of the number of elementary rules of the model and the lasting time of calculations to the type of implemented t-norm operator (numInputs: 2, numOutputs: 1, mf: gaussmf, numMfs: 10, numGmfs: 10)

Operator t-norm	Number of elementary rules in the model	Time of the rules generating [s]
Min	1000	2.38
Product	1000	2.33
Hamacher product	1000	2.53
Drastic product	0	2.44
Einstein product	1000	2.45
Bounded difference	728	2.30

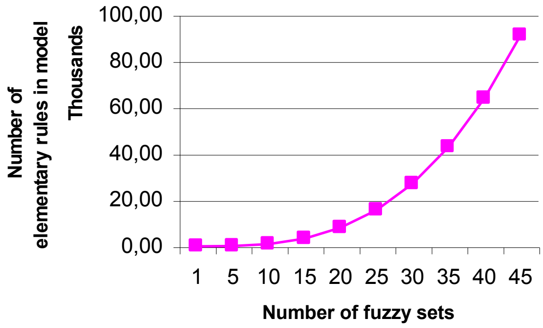

Fig.5. The number of elementary rules depending on number of fuzzy sets for variables.

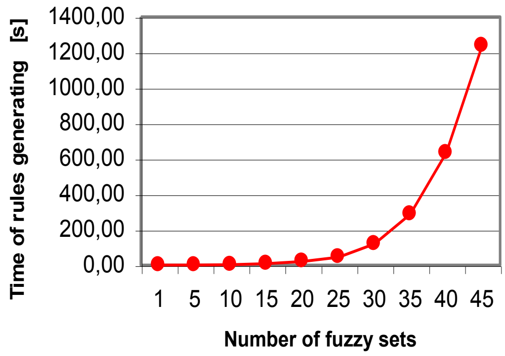

Fig.6. The time of generating rules depending on number of fuzzy sets for variables.

4. Time optimization of the program code

The fuzzy system described was implemented with the usage of Matlab structural programming. In order to optimize the time of calculations the following programming rules were applied:

Tab.5.

Principle of the time optimization in Matlab (cf. [10])

The usage of functional m-files instead of script files	
Proper usage of data	
Description	**Example**
Allocation of the variables with known sizes.	InOut=zeros(size([inX outX]));
Creating only essential variables.	h=h(find(h>0));
Correct choice of the type of variables and their keeping.	Implementation of integers instead of floats
Correct choice of pre-defined functions.	Implementation, where it is possible, of the function *num2str* instead of *in2str* (time for *genreg* function execution for 1 input variable with the usage of *int2str* – 0.17s, *num2str* – 0.30s).

Counting of the table elements according to columns not lines.	vec=zeros(1,numAtr);
	for **i**=1:numAtr
	for **j**=1:numMf
	if ~isempty(find(mf(**j**,**i**)))
	vec(1,**i**)=vec(1,**i**)+1;
	end
	end
	end

Tab.6.

Principle of the time optimization in Matlab cont. (cf. [10])

<table>
<tr><th colspan="2">Vectorized the code</th></tr>
<tr><td>The usage of the operator colon in reference to e.g. whole lines or columns.</td><td rowspan="4">Example of calculations of activating degree of the rules with the usage:
* product
h=valuMfIn(:,1);
for j=2:numIn
  h=h.*valuMfIn(:,j);
end;
(time execute: 0.015 s for 12 inputs and 10000 measurements)
Instead of:
for i=1:numM, h(i)=valuMfIn(i,1); end
for j=2:numIn
   for i=1:numM
      h(i)=h(i)*valuMfIn(i,j);
   end
end;
(time execute: 0.235 s for 12 inputs and 10000 measurements)
* drastic product
h=valuMfIn(:,1);
for j=2:numIn
  d = zeros(size(h));
  b=valuMfIn(:,j);
  d(find(max(h,b)==1)) = ...
  min(h(find(max(h,b)==1)), ...
  b(find(max(h,b)==1)));
  h=d;
end
(time execute: 0.032 s for 10 inputs and 10000 measurements)
Instead of:
for i=1:numM
   h(i)=valuMfIn(i,1);
end
for j=2:numIn
   for i=1:numM, b(i)=valuMfIn(i,1); end
   for i=1:numM
      if max(h(i),b(i))==1
         h(i)=min(h(i),b(i))
      else h(i)=0; end
   end
end
(time execute: 0.453 s for 10 inputs and 10000 measurements)</td></tr>
<tr><td>The usage of the standard functions operating on the tables e.g. find, min, max, sum, prod, all, repmat, etc.</td></tr>
<tr><td>Usage of the array arguments: ./, .*, .^</td></tr>
<tr><td>Usage of the index logic in reference to elements of matrix.</td></tr>
</table>

After introducing JIT-Accelerator, codes vectoring is not so advantageous unless it retains proper programming rules [10]. However, while operating on multidimensional matrixes JIT-Accelerator does not assist time optimization. In cases when algorithms do not allow vectoring of the code and to operate on matrix, then recoded C language and usage of mex-files may be considered.

5. Conclusion

The tool described in this article helps to build a probabilistic-fuzzy model on the basis of unrestricted empirical data. Thanks to the possibility of the implementation of different parameters while building the structure there is a possibility of matching the model to the analysed process. Unfortunately calculations are only efficient for a small number of variables and when the number of fuzzy sets is relatively small. The aim of future research is the method of identification of the model which will lessen calculation input and the number of rules but at the same time it will retain clarity and accuracy of the model.

Bibliography

[1] Korbicz J., Kościelny J.M., Kowalczuk Z., Cholewa W.: Diagnostyka procesów. Modele. Metody sztucznej inteligencji.. Zastosowania, WNT, Warszawa, 2002.

[2] Walaszek-Babiszewska A.: Linguistic Knowledge Representation for Stochastic Systems, proceedings of Internat. Multiconference on Computer Science and Information Technology, 2007, pp. 141-150.

[3] Walaszek-Babiszewska A.: IF-THEN liguistic fuzzy model of a discrete stochastic system. In:. Artificial Intelligence and Soft Computing, EXIT, Polish Neural Network Society, Warsaw, 2006, pp.169-174.

[4] Walaszek-Babiszewska A.: Construction of fuzzy models using probability measures of fuzzy events, proceedings of MMAR 2007, pp. 661-666.

[5] Rutkowski L.: Metody i techniki sztucznej inteligencji, PWN, Warszawa, 2005.

[6] Yager R.R., Filev D. P.: Podstawy modelowania i sterowania rozmytego, WNT, Warszawa, 1995.

[7] Walaszek-Babiszewska A., Błaszczyk K., Czabak A.: Budowa rozmytych modeli procesów stochastycznych przy użyciu reguł asocjacji, W: Sterowanie i automatyzacja: aktualne problemy i ich rozwiązania, Malinowski K., Rutkowski L (red.), EXIT, Warszawa, 2008.

[8] Piegat A.: Modelowanie i sterowanie rozmyte, EXIT, Warszawa, 2003.

[9] Fuzzy Logic Toolbox 2, User's Guide, The MathWorks, Release 2008a.

[10] Programowanie w języku Matlab, Optymalizacja czasowa kodu, www.rose.aei.polsl.pl/~darekc/inst/pjm/PjMw2.pps.

[11] Zadeh L.A.: The concept of a linguistic variable and its application to approximate reasoning, Information Sciences, Part I: pp. 199-240, 8, 1975.

[12] Zadeh L.A.: Probability measures of fuzzy events, Journal of Mathematical Analysis and Applications, vol. 23, pp. 421-427, 2, 1968.

Author:
MSc. Eng. Błaszczyk Katarzyna
Opole University of Technology
ul. Ozimska 75
45-370 Opole
tel. (077) 423-40-35

email: *k.blaszczyk@po.opole.pl*