BULETINUL ŞTIINŢIFIC al Universităţii "POLITEHNICA" din Timişoara, România,
Seria AUTOMATICĂ ŞI CALCULATOARE
SCIENTIFIC BULLETIN of The "POLITEHNICA" University of Timişoara, Romania,
Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, Vol. 58 (72), No. 1, 20vv

# Native Developer Toolbox Library for Sparse Fuzzy Rule Based System

Zoltán Krizsán*, Szilveszter Kovács* and Dong Hwa Kim**

* Department of Information Technology, University of Miskolc, Miskolc-Egyetemváros, H-3515, Hungary
krizsan@iit.uni-miskolc.hu, szkovacs@iit.uni-miskolc.hu

** Department of Electronic and Control Engineering, Hanbat National University, 16-1 Duckmyong dong Yuseong gu Daejeon, South Korea 305-719. kimdh@hanbat.ac.kr

***Abstract*** *– Direct application of classical fuzzy reasoning methods for complex real world tasks are facing the problem of the rule base size. One solution for avoiding the exponentially growing rule base is the adaptation of sparse fuzzy rule-base knowledge representation and the fuzzy rule interpolation methodology. There are numerous implementations of the classical fuzzy reasoning methods can be found on public Internet sources as software products, but there is a shortage of publicly available fuzzy rule interpolation software products. One exception is the publicly available Fuzzy Rule Interpolation Toolbox for Matlab introduced and developed by Johanyák et. al. That special Matlab Toolbox is perfect for research purpose but it is hard to use in real-time application environment. In this paper, we examine the existing FRI methods then clarify a set of common criteria. According to these theoretical demands some practical requirements of the framework structure are defined. Finally a short introduction of the structure and the usage of the Fuzzy Rule Interpolation Developer Toolbox Library are shown. The Developer Toolbox Library provides an efficient way for developing real-time applications of incomplete fuzzy rule base models. The library supports interfaces for most of the popular programming languages.*

***Keywords:*** *Sparse Fuzzy system, Fuzzy Rule Interpolation Developer Toolbox Library.*

## I. Introduction

One of the main practical benefits of the fuzzy system applications is the rule-based knowledge representation. It is quite straightforward to implement a priory heuristic knowledge in rule form. On the other hand classical fuzzy reasoning methods, like the Zadeh-Mamdani-Larsen Compositional Rule of Inference (CRI) (Zadeh [1]) (Mamdani [2]) (Larsen [3]) or the Takagi-Sugeno fuzzy inference (Sugeno [4], Takagi-Sugeno [5]) require complete rule-base definition, i.e. the full coverage of input space by rules. Without the full coverage i.e. if the fuzzy rule-base is sparse, some observation may exists from which no conclusion can be deduced. The need of the complete rule base leads to a rule base size exponentially growing with to the number of the antecedent dimensions.

The reason of sparse fuzzy system can be the incomplete knowledge which means that less information is available about system than it would be required for complete rule base. Or because of simple technical reasons, like in expert systems, where only the important rules are defined without taking care of the completeness, not mentioning the deducible rules [21]. Moreover the full coverage is also not desirable for performance reasons [19].

There are numerous fuzzy reasoning methods which can handle sparse fuzzy rule-bases too. In common they are called Fuzzy Rule Interpolation methods. They can generate the Fuzzy conclusion from the existing rules by fuzzy interpolation.

These methods can be classified into two groups depending on fuzzy conclusion generation strategy: direct methods (also called "one step method"), and "two steps methods". The two steps methods interpolate a temporary fuzzy rule first in the position of the observation, and then apply a single fuzzy rule reasoning method to gain the shape of the final fuzzy conclusion. One step methods generate the fuzzy conclusion directly from the fuzzy observation in one reasoning step.

In spite of the numerous implementations of the classical fuzzy reasoning methods (e.g. Matlab Fuzzy Toolbox), there are not so many publicly available fuzzy rule interpolation software products. One exception is the publicly available Fuzzy Rule Interpolation Toolbox for Matlab introduced and developed by Johanyák et. al. [14], [15].

The original goal of the FRI Toolbox development was the comparison and visualization of the various FRI methods. During the implementation the effective calculation and the possibility of real-time application was a minor demand.

Unfortunately there is no commonly available programming library until now, that can effectively support the elaboration of a sparse fuzzy model based application.

So far there is no universally accepted common FRI method for reasoning in case of sparse fuzzy knowledge representation. On the other hand there are many FRI methods already developed but none of them is commonly accepted as a universal all purpose FRI method.

Therefore there is a need for a common framework in which any of these methods can be used and can be compared. By this time there were two solutions for this problem. One of them is the direct implementation of the FRI method based on the original source articles. The developer had to implement the appropriate FRI method on his/her own in a chosen programming language. It has more disadvantages. Reimplementation is required when the programming language is changing, more FRI methods have to be implemented for comparison and it can be difficult task because some methods are poorly documented. The other alternative is the application of the FRI Matlab Toolbox which was already implemented by Johanyák at al. This special toolbox is a collection of ".m" Matlab functions which can be run under the Matlab environment. It is an easy to use and handy tool for demonstration and research purposes, but its integration into real application is troublesome. Although it is possible to run Matlab function inside a C++ application, but it requires an embedded interpreter code, and the "real time" reasoning is also troublesome. This interpreted usage of the ".m" Matlab code is not supported in any other programming languages such as the .NET supported ones, like the Java or Python. The function collection of the FRI Toolbox does not work in other mathematical environment and not even supports free simulation engines such as FreeMat or Octave.

As a straightforward solution of the above problem we have designed and implemented a new developer framework for the FRI Toolbox.

The code was written in C++ in order to obtain low reasoning time, however it can be used by any other recent popular programming language such as .NET (C#, VB.NET), Java, Python or Delphi. The C++ language is supported natively and the other languages via dll (Windows) or so (Linux) technology. More applications can use our logic in shared way due to the address space of the dll is shared among applications. Our logic can be used directly as module of application or in assisted mode as component of complex system. In the assisted usage our logic manages the user's fuzzy systems and shares them among more applications. We ship wrapper classes for .NET, Java and Python for the reason of reducing the development period and increasing the productivity. Behind the scene these classes work in assisted mode collaborating with dll.

FRI based automatic fuzzy system identification from training data [21] and embedded FRI model applications [23] are still ongoing research areas. For this reason our primary goal was creating a handy component for developer which can be used in case of sparse fuzzy knowledge representation applications as a tool for creating component configuration file and also for generating the source code of component usage.

Building a common tool, at the first we would like to support the Windows and Linux operating system. For better performance we would like to give a quick native library for C++ and a unified usage model to the other recently used other programming languages. In case of Mac OS the native library is supported but the assisted usage will be migrated later due to the operating system nature.

The rest of this paper is organized as follows. Section II reviews the FRI methods and defines a general set of criteria. Section III presents the structure and the usage of the new Fuzzy Developer Library. Finally the last section concludes the result and outlook the future research and development.

## II. The FRI Methods

In case of sparse fuzzy rule based knowledge representation having a fuzzy observation, the fuzzy conclusion can be gained by an FRI method. There is no universally accepted common purpose FRI method exists at the moment, however there are many FRI methods already developed.

Every FRI methods have different features and different constraints. The developer has to know all of these to be able to make the correct decision for selecting a suitable FRI method. Helping this selection we have to analyze the existing FRI methods and a general set of criteria has to be identified.

According to the classification of the FRI methods, significant members of the direct method are the KH method [6] introduced by Kóczy and Hirota, MACI [7] (Tikk and Baranyi), FIVE [8] (Kovács and Kóczy), IMUL [9] (Wong, Gedeon, and Tikk), and VKK method [10] (Vass, Kalmár and Kóczy). The methods belonging to the "two step method" group are introduced as generalized methodology (GM) defined by Baranyi et al. in [11]. Significant methods of this group are the ST method [12] (Yan, Mizumoto, and Qiao), the IGRV [13] developed by Huang and Shen, and the technique proposed by Jenei [19].

The Fuzzy Rule Interpolation Matlab Toolbox (FRI TB) is a collection of Matlab functions implementing interpolation based fuzzy inference techniques introduced in [14]. These functions are implemented in Matlab using internal Matlab types and functions. The current version supports more FRI methods (KH, the stabilized version of the KH, MACI, IMUL, CRF, FIVE, VKK, GM with SCM, FERI, and FPL, and GM with FEAT-p, FERI, and FPL). The whole toolbox is available for download under GNU General Public License from the web site [15]. The FRI Toolbox was developed using Matlab 7 (R14) under Microsoft Windows XP, and it is working under Windows 7 as well as Linux system. It has graphical user interface. The user can set up the details of the system and can choose an FRI method which determines the conclusion. Similarly to the "Matlab Fuzzy Logic Toolbox", the inputs of the FRI TB is an extended ".fis" file for the fuzzy rule and parameter definition and an ".obs" for the fuzzy observation definition. Differently from the ".fis" file of the "Matlab Fuzzy Logic Toolbox", the extended ".fis" file has some additional parameters for enabling subnormal linguistic terms (fuzzy sets with height less than 1). These files are normal text files but there is a graphical editor of the FRI

TB which can be used for easy fuzzy set creation and modification.

For selecting a proper FRI method, based on a general set of FRI criteria, some requirements against the properties of the applied FRI method has to be determined. There are three main criteria sets can be found in the literature, in [19], [16] and [24]. In this paper we recall the FRI criteria introduced in [16] by D. Tikk et. al. These conditions are the following:

*Property 1* Avoidance of the invalid conclusion. *Property 2* Keep the similarity. This means that similar observations should lead to similar conclusions. *Property 3* Preserving the ”in between” relation. If the antecedent sets are between two rules in observation, then the approximated conclusion should be between the two correponding consequent sets. *Property 4* Compatibility with the rule base. This condition requires the validity of the modus ponent reasoning, i.e. if an observation coincides with the antecedent part of a rule, the conclusion produced by the method should correspond to the consequent part of that rule. *Property 5* Keep the fuzziness of the approximated result. There are two opposite approaches in the literature related to this topic. According to the first one in case of a singleton observation the method should produce a singleton consequence. The second approach specifies the fuzziness of the estimated consequent from the nature of the fuzzy rule base. The singleton can be expected only if all the consequents of the rules taken into consideration in the interpolation are singleton. *Property 6* Approximation stability. The estimated rule should approximate the relationship between universes of the antecedent and consequent with the highest possible degree. *Property 7* Preserving the piece-wise linearity. If the fuzzy sets of the rules taken into consideration are piece-wise linear, the approximated sets should preserve this feature. *Property 8* Applicability in case of multidimensional antecedent universe. This condition indicates that an FRI technique should present similar characteristics when being extended and applied to multidimensional input spaces.

*Property 9* Applicability without any constraint regarding to the shape of the fuzzy sets. This condition can be weakened practically to the case of piece-wise linear, and Gauss-bell shaped fuzzy sets, being the most frequently encountered in the applications.

Some methods of the FRI TB are evaluated against the above criteria in [16].

## III. FRI Developer Toolbox Library

There are three main requirements against the FRI Developer Toolbox Library (FRI DTBL):

- *Short reasoning time* to be suitable for real time application environment.
- *Wide application area* has to be supported (at least Matlab, Freemat, Java, and C#).
- *Standardized application interface* for supporting simple interchangeable application of the FRI methods.

From the developers’ point of view the structure of the framework should be easily extended. From the viewpoint of the user, who just uses the existing FRI DTBL, requires a common structure guaranteeing the interchangeability of the FRI methods. The object oriented paradigm with proper class hierarchy can ensure both requirements. In order to meet these demands the existing FRI methods were analyzed and the following criteria were identified:

- Same process steps are required for every FRI method: system creation, initialization, interpolation. Defining a pure virtual method for these steps force the implementation and guarantee the correct process flow.
- Some FRI methods have additional parameters which determine the resolution and precision of calculation (number of α cuts, number of polar cuts, etc.). Introducing an additional method give an opportunity to initialize the given FRI method, but the usage of the FRI methods stay same and universal.
- Every FRI method is a member of a “direct” or “two step” method family, so two super classes are required which will be the base class of any class of FRI method.

For providing wide compatibility a programming library (C++ lib), a dynamic loadable library (dll in case of Windows and so file in case of Linux) and wrapper classes for popular programming languages such as C#, Python and Java were elaborated. In the FRI DTBL the usage of the FRI methods should be unified, which means that any FRI method can be connected via the same interface. The structure of library and the objected oriented paradigm ensure this unified FRI method usage.

The main goal of the FRI DTBL is simplifying the standalone application development by providing standardized library functions for the FRI reasoning. *The FRI DTBL provides two facilities for application: direct application and assisted application.*

The *direct application* means that our classes can be directly instantiated and the method of the object can be called without any mediator code. In this case the developer's responsibility is to create, store and destroy the FRI system. The creation and destruction are simple memory operations. The developers can use a vector to store the FRI system instances of classes. This provides methods for getting the size of structures and iterators for the ability to iterate through the elements of that range using operators. These tasks have to be implemented directly in applications, there is no need for a central place for interchanging information among system components.

The *assisted application* means that all operations are indirect. The user asks for a new FRI system with specified parameters from our logic. Then the logic creates the system, which will be stored in an internal list. Once the FRI system becomes unnecessary, then the user has to ask our logic to dispose it. In this case the user does not know the place of system object, the creation and destruction are simple function calls. The assisted usage is available from

any application, but the direct usage is available only from C++ application. Developing the application in C++ and linking our logic as static library is more efficiency and reliable, because the C++ is an object oriented language, which supports complex types.

If the user writes the application in C++, which include our logic, both usages can be accessed, because the library itself, as well as the small manager code are written in C++. If the binary dll is used by any application, only then the assisted usage is supported. If the pre-created wrapper classes are used by any other code, then the assisted mode can be available because the wrappers are using the dll.

The programming languages have many differences. The representation of types in memory and the functional possibilities can be different as well. Avoiding these diversities we have implemented the dll in C because every popular programming languages support the pure C written dll. In our case the global functions of the C dll are adapters, which convert the programming interface of internal classes to simple functions. Behind the scenes these functions are creating instances of internal classes and communicating with the objects. Even if the C language is widely supported, but it it can handle simple types only, moreover it is not object oriented. Because of the usage of the C dll between the user's logic and our internal worker classes, the parameters of the methods have to be simple types, such as double array, integer and pointer (instead of the complex class instance pointer). Handling these differences the adapter function contains additional codes which build the appropriate objects according to the simple parameter requirements. Usage of the dll, which contains functions with simple parameters only, requires precise code, because our logic can not determine the size of array. There is no chance to determine the array size in case of C language (particularly in case of Matlab), so we had to introduce an extra parameter for the array size. In case of wrong array size value, a fatal error can be occurred. Therefore the correct usage is the user's responsibility.

Our logic can handle multiple FRI system in the same time, so it is also possible that the dll can be used side by side from more applications at the same computer at the same time. It is supported, because our logic stores the users' FRI systems in an internal list and each request has an additional parameter which identifies the corresponding FRI system. This solution enables the existence of more applications and more systems simultaneously. In case of low level FRI DTBL usage (.lib file), by a proper C++ source the developer can instantiate our class and can override our implementation. It is easy to extend the FRI methods by deriving a new class from an existing one. For compatibility reasons we have generated the .dll file from a library which provides the programming language independency. The implementation language of the dll is simple C, due to compatibility issues, because it has to be used by other execution contexts too, which can be implemented in Delphi, or C#, or even in Java.

Summing up the comments, the *direct application* gives more possibility as well, but it also requires more responsibility, than the *assisted application*.

In accordance with general criteria related to the FRI methods (enumerated in the previous section) the FRI DTBL also try to check the prerequisites of the actual FRI method and the properties of the fuzzy conclusion. At the moment checking the *Property 1* is only supported. To extend the property evaluation abilities, new algorithms have to be introduced. In case of direct usage and unsupported FRI method prerequisites an exception is thrown. In the assisted mode the exception handling is not supported, so the return value of any method can be an error code, which is a negative integer. We have defined error codes and exceptions for every studied invalid situation. If the method call is finished successfully, then the return error value will be zero.

### *A. The Structure of the FRI DTBL*

The core of toolbox is written in C++. Therefore, a lot of classes are defined, where one class can relate to the other by specialization, or simple usage. Our library supports both FRI method families, the direct (one-step) methods and the two-steps methods. Our system can also use the same .fis file as MATLAB Fuzzy Toolbox does. The content of the extended .fis file can also contains the user's sparse system and the FRI method will determine the conclusion according to this description.

In real circumstances the rules of the system do not change frequently. In most cases more observations are tested against the same FRI system. For the better performance, the simple parameter of function is also supported as an observation beyond the .obs file.

Commonly, all Fuzzy Interpolation Methods can have additional input parameters, which can be used by developer's code, such as number of the alpha levels. The number and the type of parameters depends on the specified FRI method, so method parameters are handled in string format.

In FRI DTBL the *CFRIMethod* class is the direct or indirect base class of all classes (see Fig.1, Fig.2). This base class declares the common interface of all FRI methods. From the user point of view, who wants to utilize our library, same four steps are required for all of the FRI methods:

1. *System creation*, in which the FRI system is created from a string, or based on a .fis file.
2. *FRI method creation*, in which the system, created in the previous step, is assigned to the implementation of the appropriate FRI method. In case of assisted usage the name of the method has to be passed as a simple string parameter. Please note that thereafter this step can be executed more times with the *setup* method, so the interpolation method can be also changed in runtime.
3. *FRI method initialization*, where the method can get specified parameters. They will be used during the determination of the conclusion. The *init*

method can get parameters from user at runtime, which can modify the interpolate method.

4. *Interpolation of the conclusion*, in which the *interpolate* method is called. This is the main step of the FRI DTBL. It can be called more times in case of direct real time applications.

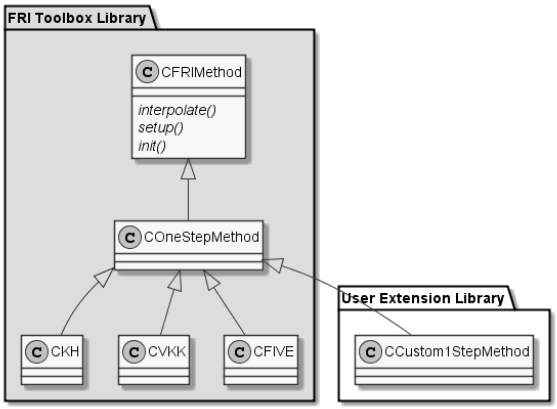


**Figure 1** The structure of the direct FRI method classes

In case of one-step methods (see Fig. 1), the class of the actual FRI method has to be derived from the *COneStepMethod* class. The *COneStepMethod* declares only one pure virtual interpolate method, which has to be implemented, overridden by the class of the new FRI method. This method has to determine the conclusions according to the given observation and the FRI system. The observations are fetched from parameters, which is a vector of Membership functions. The FRI DTBL already contains implementation of the following one-step methods: KH (*CKK* class), VKK (*CVKK* class) and FIVE (*CFIVE* class).

If a new FRI method is implemented, then the class of the new FRI method has to be derived from *COneStepMethod* and the *interpolate* method has to be overridden by the conclusion directly (see *CCustom1StepMethod* on Fig. 1).

The other types of the supported FRI methods are the two-steps methods (see Fig. 2), this case the actual class of the FRI method has to be derived from *CGeneralizedMethod* class. The actual FRI method class has to override three methods: *determineAntecedentShapes*, *determineConsequentPositions* and *determineConsequentShapes* according to the steps of the the General Methology [11]. These methods are called from the *CGeneralizedMethod* class in this strict order every time. The observations are also fetched from the string representation of a vector of Membership functions. The FRI DTBL already contains implementation of the following two-steps methods: LESFRI (*CLESFRI* class), VEIN (*CVEIN* class) and ST (*CST* class). If a new two-steps FRI method is implemented, then the class of the new FRI method has to be derived from *CGeneralizedMethod* and the *determineAntecedentShapes*, *determineConsequentPositions* and *determineConsequentShapes* methods has to be overriden (see *CCustom2StepsMethod* on Fig. 2).

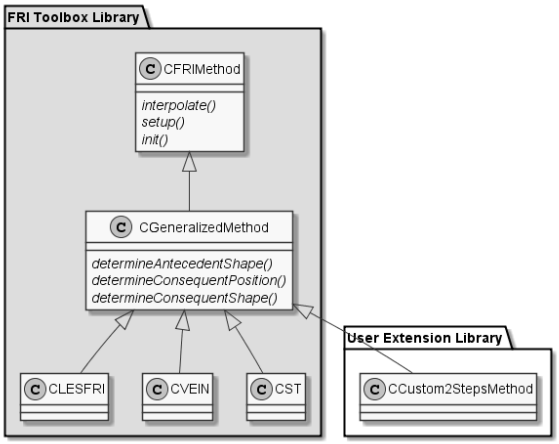


**Figure 2** The structure of two steps FRI method classes

*B. Using the FRI DTBL*

The new native library can be used in C++. This solution produces the best performance, but more efforts are needed. The dynamic library can be used with different types of target languages, including common scripting languages such as Perl, PHP, Python, Tcl and Ruby. The list of supported languages also includes non-scripting languages such as C#, Java, and Delphi. In this case the dll can be used in assisted application.

One of our goal was to support the Matlab environment. Matlab is a programming environment platform for algorithm development, data analysis, visualization, and numerical computation. The main advantage of its application is the visualization and the strong mathematical support. As an additional component, a Fuzzy Logic Toolbox exists for developing a fuzzy system, but it is not part of base system and it requires complete rule-base for Fuzzy system. When a shared library is used in Matlab it needs a header file, which provides signatures for the functions in the library and the dll itself. A signature, or function prototype, establishes the name of the function and the number and types of its parameters. For the application of the shared library, the full path and its header file have to set up. Matlab accesses the C routines built into external, shared libraries through a command-line interface. This interface loads the external library into Matlab memory and access the functions of the FRI toolbox dll. Although types of the two languages are different, in most cases types can be passed to the C functions without any further conversion, as the Matlab do the type conversion automatically.

During the application first the external C library is loaded by the *loadlibrary* function. To check the success of the loading the *libfunctions* function can be called next. Finally the *callib* function can be used to call external function if the previous steps finished without an error. The argument(s) of the *callib* depends on the actual external function.

The support for any free mathematical environment was also a key element in our FRI DTBL development concept.

In the first step the FreeMat is supported. FreeMat [17] is a free environment for rapid engineering and scientific prototyping. It is similar to commercial systems such as Matlab, but it is licensed as GPL Open Source.
In case of FreeMat the function of shared library can be used by import function. We ship a text file which contains the import for all defined functions. The signature of import function is: *import(libraryname, symbol, function, return, arguments)*, where the argument *libraryname* is the name of the library (as a string). In our case it is the '*fritoolbox.dll*'. The second argument '*symbol*', is the name of the symbol, the internal name of the global function. The third argument of the function is the external name of the function. The fourth argument is a string that specifies the return type of the function.

## III. Conclusion

Fuzzy rule interpolation techniques extend the applicability of fuzzy rule based reasoning methods for the case when the rule base is sparse or incomplete. This paper introduced the design concepts of the FRI Toolbox Developer Library (FRI DTBL) which is freely available public programming library. The main reason of the FRI DTBL development was the direct support for embedded applications of FRI methods in standalone real environment applications.
The FRI DTBL can be used by any recent popular programming languages. It can be extended easily with any kind of new FRI method. Our library has a short reasoning time so it can be also used in any application including Matlab and FreMath mathematical environments as well.
Our framework is suggested for system developers who want to create real environmental applications.
In the future more FRI methods will be integrated and a new benchmark system will be set up within the FRI DTBL framework for testing the fulfillment of the FRI criteria.

## ACKNOWLEDGMENTS

This research was partly supported by the Hungarian National Scientific Research Fund grant no: OTKA K77809, by International Cooporation Program (Sister Univ.) of the Hanbat National University, S. Korea, and by the Hungarian National Development Agency and the TÁMOP-4.2.2.B/10/1-2010-0008 project with support by the European Union, co-financed by the European Social Fund.

## REFERENCES

[1] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", IEEE Trans. on SMC, (3), pp.28-44, 1973.

[2] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", Int. J. of Man Machine Studies, (7), pp.1-13, 1975.

[3] P. M. Larsen, "Industrial application of fuzzy logic control", Int. J. of Man Machine Studies, (12) 4, pp.3-10, 1980.

[4] M. Sugeno, "An introductory survey of fuzzy control", Information Science, (36), pp.59-83, 1985.

[5] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", IEEE Trans. on SMC, (15), pp.116-132, 1985.

[6] L. T. Kóczy and K. Hirota, "Rule interpolation by α-level sets in fuzzy approximate reasoning," BUSEFAL, vol. 46, no. Automne, pp. 115–123, 1991.

[7] D. Tikk and P. Baranyi, "Comprehensive analysis of a new fuzzy rule interpolation method," IEEE Trans. On Fuzzy Systems, vol. 8, no. 3, pp. 281–296, 2000.

[8] S. Kovács and L. T. Kóczy, "Application of an approximate fuzzy logic controller in an agv steering system, path tracking and collision avoidance strategy," Tatra Mountains Math. Publ., vol. 16, pp. 456–467, 1999.

[9] K. W. Wong, T. D. Gedeon, and D. Tikk, "An improved multidimensional α-cut based fuzzy interpolation technique," in Proc. of the Int. Conf. on Artificial Intelligence in Science and Technology (AISAT'00), V. Karri and M. Negnevitsky, Eds., Hobart, Tasmania, Australia, December 2000, pp. 33–38.

[10] G. Vass, L. Kalmár, and L. T. Kóczy, "Extension of the fuzzy rule interpolation method," in Proc. of the Int. Conf. on Fuzzy Sets Theory and its Applications (FSTA'92), Liptovsk´ y Jan, Slovakia, 1992, pp. 1–6.

[11] P. Baranyi, L. T. Kóczy, and T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," IEEE Trans. on Fuzzy Systems, vol. 12, no. 6, pp. 820–837, December 2004.

[12] S. Yan, M. Mizumoto, and W. Z. Qiao, "An improvement to Kóczy and Hirota's interpolative reasoning in sparse fuzzy rule bases," Int. J. of Approximate Reasoning, vol. 15, pp. 185–201, 1996.

[13] Z. H. Huang and Q. Shen, "Fuzzy interpolation with generalized representative values," in Proc. of the UK Workshop on Computational Intelligence, Loughborough, UK, September, 2004, pp. 161–171.

[14] Z. C. Johanyák, D. Tikk, S. Kovács, and K. W. Wong, "Fuzzy rule interpolation Matlab toolbox – FRI toolbox," in Proc. of the IEEE World Congress on Computational Intelligence (WCCI'06), 15th Int. Conf. on Fuzzy Systems (FUZZ-IEEE'06). Vancouver, BC, Canada: Omnipress, July 16–21, 2006, pp. 1427–1433.

[15] Z. Johanyák. Fuzzy Rule Interpolation Matlab Toolbox website. [Online]. Available: http://fri.gamf.hu

[16] D. Tikk, Z. C. Johanyák, S. Kovács, and K. W. Wong, "Fuzzy rule interpolation and extrapolation techniques: Criteria and evaluation guidelines," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 15, pp. 254–263, 2011.

[17] Freemat website. [Online]. Available: http://freemat. sourceforge.net

[18] Kóczy, L. T., Hirota, K.: Size reduction by interpolation in fuzzy rule bases, IEEE Trans. on SMC, 1997, 27:14–25.

[19] S. Jenei, "Interpolating and extrapolating fuzzy quantities revisited – an axiomatic approach", Soft Comput., vol. 5., pp. 179-193, 2001.

[20] S. Jenei, E. P. Klement and R. Konzel, "Interpolation and extrapolation of fuzzy quantities – The multiple-dimensional case", Soft Comput., vol. 6., 258-270, 2002.

[21] Sz. Kovács, "Interpolative Fuzzy Reasoning in Behaviour-based Control", Advances in Soft Computing, Vol. 2, Computational Intelligence, Theory and Applications, Bernd Reusch (Ed.), Springer, Germany, ISBN 3-540-22807-1, pp.159-170, 2005.

[22] Zs. Cs. Johanyák, "Sparse fuzzy model identification Matlab toolbox - RuleMaker toolbox", Proceedings of IEEE 6th International Conference on Computational Cybernetics ICCC, Stara Lesná, Slovakia, pp. 69-74, 2008.

[23] D. Vincze, Sz. Kovács, M. Gácsi, P. Korondi, Á. Miklósi, P. Baranyi: A Novel Application of the 3D VirCA Environment: Modeling a Standard Ethological Test of Dog-Human Interactions, Acta Polytechnica Hungarica, Vol. 9, No. 1, ISSN 1785-8860, pp. 107-120, 2012.

[24] I. Perfilieva, et al., "Interpolation of fuzzy data: Analytical approach and overview", Fuzzy Sets and Systems, doi:10.1016/j.fss.2010.08.005, 2010.