Abstract—In fuzzy applications which run in real environment the complete rule base is not always available due to real manner and performance issue. In case of real application which based on sparse rule base Fuzzy model the conclusion is interpolated using more rules. More programming libraries exist for complete rule base system, but none for sparse rule base. Developer has to implement the FRI method based on articles or the Fuzzy Rule Interpolation Toolbox for MATLAB can be used which is developed by Johanyák. That special MATLAB Toolbox is perfect for research purpose but it is hard to use in real environment. In this paper, we describe the Fuzzy Rule Interpolation Developer Toolbox Library, which is freely available. A real time application that based on incomplete fuzzy rule base model can use this Developer Toolbox Library. The library has more interfaces, which can be used by any popular programming language, for the reason of easier adoption. More FRI methods are implemented in the framework thus more alternatives can be tried and compared.

Index Terms—Fuzzy Rule Interpolation

I. INTRODUCTION

The fuzzy systems determine the conclusion using fuzzy rule base. The characteristic of the system is described by these rules. The classical fuzzy reasoning methods require the full coverage of input space, which means for all the possible observations, at least one rule antecedent exists, and hence a valid conclusion can be gained. The need of the complete rule base leads to exponentially growing rule base size with the number of the observations. When the full coverage of the Fuzzy Rule Base can not be obtained, an observation may exist, which hits none of the rule antecedents and hence no conclusion can be gained. 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. In the other hand this full coverage is not desired for performance reason. There are numerous fuzzy reasoning methods which can gain the fuzzy conclusion in this case too. In common they are called Fuzzy Rule Interpolation methods. They can generate the Fuzzy conclusion from the existing rules by interpolation.

These methods can be separated into two groups depending on generation strategy: direct method (also called "one step method"), and the other which can be called "two steps method". The two steps method interpolates a temporary rule firstly, after that it determines the position and the shape of the conclusion using the previous pre-generated rule.

Usually, in the research papers more figures are shown about incomplete fuzzy systems which are probably created by the MATLAB Fuzzy Rule Interpolation Toolbox. Unfortunately there is not any programming library until now, that can be used for developing a sparse fuzzy model based application.

In most cases the complete rule base is not available due to the real time nature and incomplete knowledge. In this case the conclusion of the sparse system is interpolated by using more rules. More programming libraries exist for complete rule base system, but none for sparse one.

Up to now there is not a common method for sparse fuzzy system. More methods exist parallel which emulate with each others but none of them is accepted as universal. There is a need for a framework in which any of these methods can be used and can be compared.

By this time there was two solution for this problem. One of them is the FRI method implementation based on articles. The developer had to implement the appropriate FRI method in the chosen programming language. It has more disadvantages. The reimplemention 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 FRI MATLAB Toolbox which was implemented by Johanyák. This special toolbox is a collection of MATLAB functions which can be run under MATLAB. It is an easy to use and handy tool for demonstration and research purpose, but the 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 but the "real time" reasoning can not be possible. This interpreted usage of the MATLAB code is not available in case of any other programming languages such as .NET supported ones, Java or Python. This special function collection does not work in other mathematical environment and does not support free simulation engines such as FreeMat or Octave.

We have designed and implemented a new developer framework which includes more FRI methods. We have implemented it 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.

Our primary goal was creating a handy module/component for developer which can be used in case of incomplete fuzzy system as well. 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.

II. RELATED WORK

Significant members of the first group of fuzzy interpolation methods are the KH method [1] introduced by Kóczy and Hirota, MACI [2] (Tikk and Baranyi), FIVE [3] (Kovács and Kóczy), IMUL [4] (Wong, Gedeon, and Tikk), and VKK method [5] (Vass, Kalmár and Kóczy). The methods belonging to the second group are introduced as generalized methodology (GM) defined by Baranyi et al. in [6]. Significant methods of this group are the ST method [7] (Yan, Mizumoto, and Qiao), the IGRV [8] developed by Huang and Shen, and the technique proposed by Jenei [9].

The Fuzzy Rule Interpolation Toolbox (FRI TB) is a collection of MATLAB functions implementing interpolation based fuzzy inference techniques introduced in [10]. 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 [11]. 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. The inputs of the MATLAB toolbox is the extended fis file and obs file of original Fuzzy Logic toolbox. Additional parameter is introduced in fis file for enabling linguistic terms with height smaller than 1. These files are normal text files but there is a graphical editor of Fuzzy toolbox which can be used for easy creation and modification.

According to general conditions related to the fuzzy interpolative methods introduced in [12] our logic has to work considering that properties. 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 results.

Property 3 Preserving the "in between" relation. If the antecedent sets are between two rules in observation, then the approximated conclusion should be between two consequent sets which are derived from previous ones.

Property 4 Compatibility with the rule base. This condition requires the validity of the modus ponent, 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 capability (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.

III. FUZZY RULE INTERPOLATION DEVELOPER TOOLBOX

There are three main requirements of FRI Toolbox library:

1) Short reasoning time to suit for real time environment.
2) Wide application area has to be supported (at least MATLAB, Freemat, Java, and C#).
3) Standardized application interfaced in application the different reasoning method can be easily interchangeable.

In order to meet these demands we ship 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. The usage of the FRI methods should be unified which means that any FRI method can be used via same interface. The structure of library and the objected oriented paradigm ensure this unified FRI method usage.

The main goal of the toolbox is simplifying the standalone application development by providing standardized library functions for the FRI reasoning. It provides two facilities for usage: direct usage and assisted usage.
The direct usage means that our classes can be instantiated and the method of object can be called without any mediator code. In this case the developer’s responsibility is to create, store and destroy the fuzzy system. The creation and destruction are simple memory operations. The developers can use vector to store our instances of classes which 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 in all applications, no central place for interchanging information among systems.

The assisted usage means that all operations are indirect. The user asks a new fuzzy system with specified parameter from our logic. After the logic creates the system, it is stored in an internal list.

Once the fuzzy system becomes unnecessary then the user has to asks 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 call.

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 object oriented language and it supports the complex types.

If the user writes the application in C++ which include our logic, both usage 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 use the dll.

The programming languages are basically different. 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 dll written in pure C.

In our case the global functions of C dll are adapters which convert the programming interface of internal classes to simple function. Behind the scenes these functions create instance of internal classes and communicate the objects. The C language is widely supported but it supports simple types only and it is not object oriented. Because the C dll is inserted 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 to simple type instead of class instance pointer. Handling these differences the adapter function contains additional codes which build the appropriate objects according to simple parameters. Using the dll, which contains functions with simple parameters, requires precise code, because our logic can not determine the size of array. There is no chance to determine this size in case of C language (particularly in case of MATLAB) so we have to introduce a plus parameter for size of array in most cases. In case of wrong value a fatal error can be occurred. The correct usage is the user’s responsibility.

It is possible that our dll can be used side by side from more applications in the same computer at the same time, because our logic allows the existence of more fuzzy systems and handles this situation. It can be because our logic stores the users’ systems in an internal list and each requests have an additional parameter which is the identifier of the fuzzy system. This solution guarantees that more systems can exist in one application and what is more a lot applications can exist with more systems simultaneously. In case of low level usage (.lib file) by the C++ source the developer can instantiate our class and can override our implementation. It is easy to extend the FRI methods deriving a new class from an existing one. For compatibility reason we have generated a .dll file from library which provides the programming language independency. The implementation language of dll is simple C due to compatibility because it has to be used by other execution contexts which can be implemented in Delphi or C# or Java. To sum up the simple usage gives more possibility as well as but more responsibility than the assisted usage.

In accordance with general conditions related to the fuzzy interpolative methods the implementation of FRI methods check the environment and the result. Only the Property 1 can be checked, the other properties just suggestions and the existing method do not fulfill them. In case of direct usage and abnormal situation an exception is thrown. In the assisted mode the exception handling is not available, so the return value of any method can be the error code, which is a negative integer. We have introduced error code and exception for every invalid situation. If the request is finished successfully then the return value will be zero.

A. The Structure of Toolbox library

The core of toolbox is written in C++, therefore, a lot of classes are defined, where one class can related to the other using specialization or simple usage. Our library supports both families of FRI methods which produce the result in one step (one-step method) and the other group in which the method generates the result in two steps (two-steps method).

Our system uses the same fis file as MATLAB Fuzzy Toolbox does. The content of FIS file 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 but the observations usually change according to its nature. For better performance the observations become a simple parameter of function instead of obs file. Commonly, all Fuzzy Interpolation Methods can have additional input parameters which can be used by developer’s code such as number of alpha levels. The number and the type of parameters depends on the specified method so one method parameter is allowed in specified string format.

The CFRIMethod class is the direct or indirect base class of all classes of the FRI methods. This class declares the common interface of all FRI methods. In point of user, who wants to utilize our library, same four steps are done while using any FRI method:

1) System creation, in which the fuzzy system is created from string or from FIS file.
2) FRI method creation, in which the system, created in previous step, is assigned to implementation of appropriate FRI method. In case of assisted usage the name of the method has to be passed as simple string parameter. Please note that this step can be executed more times thereafter with setup method, so the interpolation method can be changed at runtime.

3) FRI method initialization, where the method can get specified parameters. Them will be used during determination of conclusion probably. The init method can get parameters from user at runtime which will be able to used by interpolate method.

4) Interpolation of conclusion in which the interpolate method is called. This is the main step of the FRI method. Probably it will be called more times in case of real time applications.

In case of one-step method, which is shown in Fig. 1, the class of concrete FRI method has to be derived from 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 given observation and fuzzy system. The observations are get from parameter which is the vector of Membership functions. The toolbox contains implementation of some significant one-step methods: KH (CKK class), VKK (CVKK class) and FIVE (CFIVE class). If the new FRI method creates the conclusion directly then the class of new FRI method has to be derived from COneStepMethod and the interpolate method has to be overridden (CCustom1StepMethod in the Fig. 1).

The other group of supported methods is called two-steps method, which is shown in Fig. 2, where the particular class of the FRI method has to be derived from CGeneralizedMethod class and it has to override three methods: determineAntecedentShapes, determineConsequentPositions and determineConsequentShapes as the General Methology suggests in [6]. These methods are called from CGeneralizedMethod class in this rigid order in every case. Here the observations are get as parameter as well which is string representation of the vector of Membership functions. The toolbox contains implementation of some significant two-steps methods: LESFRI (CLESFRI class), VEIN (CVEIN class) and ST (CST class). If the developer want to implement a new two-steps FRI method then it has to be derived from CGeneralizedMethod and has to override the determineAntecedentShapes, determineConsequentPositions, determineConsequentShapes methods (CCustom2StepsMethod in the Fig. 2).

B. Using the Framework Library

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

Our primary goal was to support MATLAB environment. MATLAB is a programming environment platform for algorithm development, data analysis, visualization, and numerical computation. The main advantage of the application is the visualisation and the strong mathematical support. A Fuzzy Logic Toolbox exists for developing a fuzzy system, but it is not part of base system and it requires complete 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. The full path of the shared library and its header file have to set up.
MATLAB accesses C routines built into external, shared libraries through a command-line interface. This interface lets to load the external library into MATLAB memory and access the functions of the FRI toolbox dll. Although types of the two language differ, in most cases types can be passed to the C functions without converting, MATLAB does it. The loadlibrary function has to be called first. To check the success loading the libfunctions function can be called next. Finally the calllib function can be used to call external function if the previous steps run without error. The argument(s) of the calllib depends on particular external function. The codes of these steps are shown in Algorithm 1.

Algorithm 1 Sample code of FRI Developer Toolbox Library in MATLAB

```matlab
loadlibrary( 'fritoolbox.dll' , fritoolbox.h )
list = libfunctions( 'fritoolbox' , '-full' )
calllib( 'fritoolbox' , 'funcname' , arg1 , ....argN)
```

Supporting any free mathematical environment was also a key element of our conception. In the first step the FreeMat is supported. FreeMat is a free environment for rapid engineering and scientific prototyping which can be found at [13]. It is similar to commercial systems such as MATLAB, but it is Open Source. FreeMat is available under the GPL license. 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 'fritoolbox.dll'. The second argument symbol is the name of the symbol. This is the internal name of 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.

IV. 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 FRI Toolbox Developer Library which is freely available public programming library that can be used in real environment by standalone application. The library can be used by any recent popular programming language. It can be extended easily with any new FRI method. Our solution has short reasoning time so it can be used by any application including MATLAB and FreMath mathematical environments as well. Our framework is suggested to developers who want to create an application which can be used in real environment. In the future more FRI methods will be integrated and a new benchmark system will be developed which cooperating this framework.

ACKNOWLEDGMENT

This research was partly supported by the Hungarian National Scientific Research Fund grant no: OTKA K77809. This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES