# A C++ library for fuzzy sensitivity analysis and multiple fuzzy linear regression

V. Fortin & B. Bobée
*NSERC/Hydro-Québec Chair on Statistical Hydrology*
*Institut National de la Recherche Scientifique*
*Sainte-Foy, Québec, Canada, G1V 4C7*
*EMail: chaire_hydro@uquebec.ca*

## Abstract

Fuzzy sets theory has proven over the years to be a valuable tool for modeling uncertainty in engineering. It is used extensively in control, in expert systems and in rule-based models. However, applications to sensitivity analysis and regression are still few, mainly because there is no appropriate software available. A C++ library of objects has been developed to easily and efficiently introduce fuzzy sensitivity analysis into new or existing C/C++ code, and to perform multiple fuzzy linear regression. An outline of the library is given, together with examples of applications in hydrologic engineering. For problems involving only fuzzy regression, a more user-friendly interface is currently being developed.

## Introduction

In any modeling task it is vital to study the uncertainty on the inputs and on the model itself, so as to assess its effects on the outputs. For simple models, such as linear models, it is possible using probability theory to perform this sensitivity analysis analytically, given that there is sufficient data. But for more elaborate models, it is necessary to resort to Monte-Carlo simulations, usually with a hypothesis of independence between inputs to reduce complexity. When data is scarce or when the validity of the hypothesis of independence between inputs is not so obvious, probability theory collapses, leaving the scientist in an awkward position: he may feel that there is sufficient physical evidence to justify the model, but cannot defend it on statistical grounds.

An interesting alternative to probability theory in uncertainty analysis is fuzzy sets theory [7,9,15]. It allows for an approximate assessment of uncertainty even when data is scarce or the model very complex by modeling imprecise knowledge, even intuition, and combining it with the available numerical data. It is not a panacea, nor the only way of dealing with the problem. Bayesian analysis [3] or imprecise probabilities [4,12,14] may well prove as efficient and more appealing theoretically, but fuzzy sets theory as the clear advantage of being much simpler. At least as a first approximation, it is a simple solution to a complex problem.

After briefly reviewing and discussing basic concepts of fuzzy sets theory, we discuss the application of fuzzy sensitivity analysis to complex models, and the application of fuzzy regression to simple models when data is scarce. We present a C++ library of objects which can be easily used to build fuzzy sensitivity analysis into new or existing models developed in C or C++, and to perform fuzzy regression. Examples of applications of fuzzy sensitivity analysis and fuzzy regression using the proposed library to hydrologic engineering problems are provided, and possible enhancements are discussed.

# Fuzzy sets - a brief review

Fuzzy sets theory was introduced by Zadeh [15] to model imprecision in language. The concept has since been extended to allow for modeling of imprecision in measurements [5] and in probability assessments [6]. A fuzzy set $A$ is defined by its membership function $\mu_A(x)$, which associates with each element $x$ of the universe $\Omega$ a number between 0 and 1. This number reflects the degree of membership of $x$ to the fuzzy set $A$, with $\mu_A(x)=0$ meaning that $x$ does not belong to $A$ and with $\mu_A(x)=1$ meaning that $x$ belongs completely to $A$. Usual, non fuzzy (or crisp) sets are a particular case of fuzzy sets having membership functions which can only take the values 0 or 1. When there is no ambiguity, we shall write simply $\mu(x)$ for the membership function so as to simplify the notation.

Fuzzy sets theory uses extensively the concept of an $h$-cut, which is defined, for a fuzzy set $A$, as the crisp subset $A(h)$ of all values of $\Omega$ for which $\mu_A(x)\geq h$. A fuzzy set is completely defined by the collection of its $h$-cuts, and is said to be convex if all $h$-cuts are convex, i.e. can be represented by intervals $A(h)=[l_h,u_h]$. A fuzzy set $A$ is said to be normal if $A(1)\neq\emptyset$, or (equivalently) if $\sup\{\mu_A(x)=1\}$. Finally, $A(0)$ is named the support of $A$, and $A(1)$ its kernel.

## Assessing degrees of membership of fuzzy sets

Alone, the preceding definition of a fuzzy set is of little use since it does not associate a clear physical interpretation with degrees of membership. Without such an interpretation, they would be impossible to measure. Just as probabilities cannot be measured (only frequencies, betting rates, or degrees of belief - interpretations of probabilities - can be), degrees of membership cannot be measured. And just as there are many interpretations of probabilities, there are many possible interpretations of degrees of membership.

In engineering, imprecise measurements provide the basis for a useful interpretation of these degrees. Suppose you dispose of a measuring device, which cannot measure precisely the value of a random variable $Y$, but only tell that it is surely inside an interval $X=Y\pm\Delta Y$ (where $\Delta Y$ can vary from one observation to the next). A fuzzy set modeling this imprecision could have as membership function $\mu(x)$ the probability that a given value $x$ would be included in an imprecise observation $X$:

$$\mu(x)=\Pr[x\in X] \qquad (1)$$

There are strong links between probability and fuzzy sets theories, but a membership function is nothing like a probability density function (*pdf*): it does not integrate to one, and it can be interpreted directly both in the discrete and continuous cases. In contrast, to obtain a probability value from a continuous *pdf*, it is necessary to integrate the function between two bounds.

## Possibility theory

A membership function may define a possibility distribution $\Pi$ on a random variable, which is an upper bound of its probability distribution. When all imprecise measurements

$\{X_1,X_2,...,X_n\}$ of a random variable $Y$ are nested (meaning that they can be ordered so that $X_{(1)}\subseteq X_{(2)}\subseteq ...X_{(n)}\subseteq\Omega$), then $\mu(x)=\Pr[x\in X]$ is a possibility assignment of $Y$ in $\Omega$. Indeed, it can be shown [6] that for any subset B of $\Omega$, the maximum value of $\mu(x)$ over that subset is superior or equal to the probability of $Y$ belonging to B:

$$\Pi(Y\in \mathrm{B})=\sup_{x\in \mathrm{B}}\mu(x)\geq \Pr[Y\in \mathrm{B}] \tag{2}$$

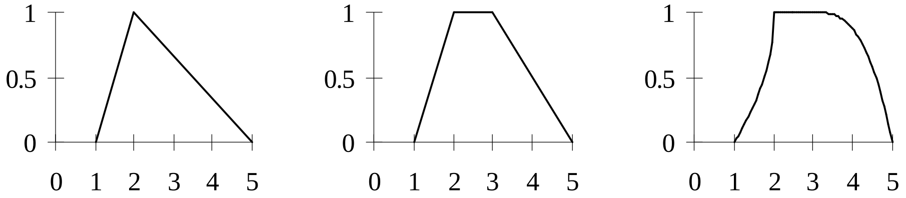
When no imprecise nested data is available (even when almost no information is available), it is nonetheless easy to obtain a possibility assignment, much more easily in fact than a probability assignment. Indeed, even if nothing is known about the probability of $Y\in$ B, a precise possibility assignment $\mu(x)$ can easily be devised to let $\Pi(Y\in \mathrm{B})=1$, so that no value of probability is excluded. Bayesian statisticians have however found that justifying any precise probability assignment in that case is next to impossible [12]. There is of course a price to pay for using possibilities: the resulting inferences are less precise than with probability theory, so that possibility theory should only be used when it is not feasible to obtain a probability distribution, either because of limited knowledge or because of limited resources (such as time and money). However, it may be a good approach to use possibility distributions in first approximation, which could then be refined using probability theory.

## Fuzzy numbers

A fuzzy number is simply a convex and normal fuzzy set. Note that a real number $z$ is a particular case of a fuzzy number having a membership function $\mu_z(x)=1$ if $x=z$ and $\mu_z(x)=0$ if $x\neq z$. Fuzzy numbers are interesting because a complete arithmetic has been defined on them [10], which generalizes interval arithmetic [11]. The result of any operator $\perp$ applied on two fuzzy numbers $A$ and $B$ is a fuzzy set $C$ having the following $h$-cuts:

$$C(h)=A(h)\perp B(h) \tag{3}$$

Notice that $C$ is a fuzzy set but not necessarily a fuzzy number. However, for most operators used in practice, except for set operators such as union and intersection, $C$ is a fuzzy number (if it exists - division by zero is still not allowed). It should also be noted that some basic results of real number arithmetic do not hold for fuzzy arithmetic. For example in general even if $A=B$, $A$-$B\neq 0$ and $A/B\neq 1$.



(1,2,5)(TFN) (1,2,3,5)(TRFN) (2,3,1,2,0.5,3)(LR)

Figure 1: membership function of various fuzzy numbers

If an infinite number of $h$-cuts are in theory necessary to completely specify a fuzzy number, much simpler fuzzy numbers are used in practice. LR numbers, proposed by Dubois & Prade [6], are a very general family of parametric fuzzy numbers. They are completely specified by their kernel $[\underline{m},\overline{m}]$, their support $[\underline{m}-\alpha,\overline{m}+\beta]$ and two functions $L(u)$ and $R(u)$ defined on [0,1] and decreasing from 1 to 0. LR numbers are denoted $(\underline{m},\overline{m},\alpha,\beta)$(LR) and their membership function is given by eq. (4).

$$\mu(x)=\begin{cases} L\left[(\underline{m}-x)/\alpha\right] & \text{if} \quad \underline{m}-\alpha\le x<\underline{m} \\ 1 & \text{if} \quad \underline{m}\le x\le\overline{m} \\ R\left[(x-\overline{m})/\beta\right] & \text{if} \quad \overline{m}<x\le\overline{m}+\beta \\ 0 & \text{elsewhere} \end{cases} \tag{4}$$

The family of functions 1-$u^q$ is often used for *L* and *R* functions, so that the notation ($\underline{m}$, $\overline{m}$ ,$\alpha$,$\beta$,*a*,*b*)(LR) can be used to completely specify an LR fuzzy number if *L*(*u*)=1-$u^a$ and *R*(*u*)=1-$u^b$. Simpler LR numbers for which $\underline{m}$= $\overline{m}$ =*m* are also often used. These will be called LRS numbers and denoted (*m*,$\alpha$,$\beta$,*a*,*b*)(LRS). Triangular fuzzy numbers (TFNs) and trapezoidal fuzzy numbers (TRFNs) are respectively particular cases of LRS and LR numbers having linear *L* and *R* functions. They are consequently completely specified by their support and kernel. A TFN with support [*a*,*c*] and kernel {*b*} is denoted (*a*,*b*,*c*)(TFN). A TRFN with support [*a*,*c*] and kernel [$b_1$,$b_2$] is denoted (*a*,$b_1$,$b_2$,*c*)(TRFN). Figure 1 shows examples of various fuzzy numbers.

## Fuzzy sensitivity analysis

Once a model is devised and calibrated to simulate a physical system, sensitivity analysis consists in studying the effect of small variations of the inputs on the outputs of the model. When the *joint* probability distribution of all inputs is precisely known, the probability distribution of the outputs can be found, either analytically if the model is simple (e.g. linear) or by Monte-Carlo simulation if the model is more complex. However, in most cases it is virtually impossible to determine the probability distribution of the input parameters unless a very large amount of data is available. Furthermore Monte-Carlo simulation involves a fair amount of computing to obtain only reasonably accurate responses.

An alternative is to obtain possibility distributions for each inputs (we have seen that this can be done precisely even when no data is available) and to replace every operation in the model by its fuzzy counterpart. Then, the outputs of the model will be fuzzy numbers, which can themselves be interpreted as possibility distributions. Not only do we then simplify the modeling of uncertainty, but we also eliminate the need for a lengthy simulation. It should however be mentioned that fuzzy operations are more costly than the corresponding operations on real numbers, leading to longer computing times, but still nowhere near the time needed for a Monte-Carlo simulation.

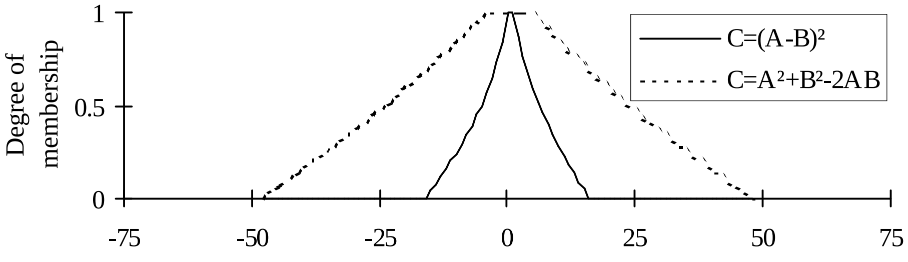


Figure 2: Computing the uncertainty on $(A\text{-}B)^2$ using two different formulae

Using fuzzy sensitivity analysis may seem simple, but there are two major difficulties to be overcome before it can be applied to a model. First, operators used in the model must be replaced by their fuzzy counterparts, which unlike Monte-Carlo simulation involves potentially major modifications to the code if the model is already programmed. The second difficulty is that the fuzzy numbers representing the outputs of the model may vary depending on the way the algorithms are coded, if any input variable appears more than once in the code [6]. Efforts should therefore be made, when coding the algorithms, to refrain from introducing the same input variable more than once. However, introducing needlessly the same variable many times in the model, such as replacing $C$=($A$-$B$)$^2$ by $C$=$A^2$+$B^2$-2$AB$, only increases uncertainty on the outputs, never decreases it, making the possibility assessments more vague, never less. It is consequently not crucial to avoid using the same variables many times, as far as the validity of the results is concerned, but it may help reduce the uncertainty on the outputs substantially.

Consider for example the fuzzy numbers $A$=(1,2,5)(TFN) and $B$=(1,2,3,5)(TRFN) (see Figure 1 for their graphical representation). Figure 2 shows the membership function of the fuzzy numbers ($A$-$B$)$^2$ and $A^2$+$B^2$-2$AB$. It is readily seen that computing $C$ as $A^2$+$B^2$-2$AB$ gives a much more imprecise result than computing $C$=($A$-$B$)$^2$, but both results are coherent, i.e. $(A\text{-}B)^2 \subseteq A^2+B^2\text{-}2AB$. In this case, using more than once the same variables increases considerably the uncertainty of the output. However, since fuzzy sensitivity analysis gives only an upper bound on the probability, the result is still valid, only less useful.

While overcoming this second difficulty may imply modifying the structure of the code, simply replacing usual operators by fuzzy ones can be much simpler with an object-oriented language such as C++. Indeed, with an object-oriented language a new type (or class) of objects representing fuzzy numbers can be defined, together with all needed operators, and only the type of the variables used in the model then needs to be changed. We have developed such a new type in C++. Before presenting the main features of this library, we shall discuss the problem of modeling linear relationships when data is scarce.

## Multiple fuzzy linear regression

An important class of physical systems can be represented by linear models. When numerical data is sufficient the problem of taking into account uncertainty can be solved by statistical regression techniques. However, this process requires the verification of a number of hypotheses, including independence between inputs and between each observation of a given input variable, and homogeneous random variations normally distributed about the mean. Usually, these hypotheses can only be verified by looking at the data, before fitting the model (for cross-correlations between inputs) and after (for the distribution of errors). When data is scarce, it is often impossible to determine the validity of those hypotheses. If it is furthermore needed to validate the model itself by analyzing the quality of the fit, even more observations are needed to obtain powerful statistical fitting tests.

Again fuzzy sets theory provides an alternative to probability theory. Fuzzy regression, developed by Tanaka et al. [13] and improved by Bárdossy [1], can often be helpful to model scarce data. It does not require the hypothesis of independence between observations of a same variable nor the independence and identical distribution of errors. In fact, known patterns in the distribution of errors can be explicitly modeled. Instead of adding a white noise to allow for model uncertainty, in fuzzy regression the parameters are taken to be fuzzy numbers. Let $f(\mathbf{x}|\mathbf{A},\bar{\mathbf{x}})$ be the fuzzy output of a linear model of $K$ input parameters $\mathbf{x}=\{x_1,x_2,...,x_k\}$:

$$f(\mathbf{x}|\mathbf{A},\bar{\mathbf{x}})=A_0+\sum_{k=1}^{K} A_k\,(x_k-\bar{x}_k) \tag{5}$$

where $\mathbf{A}=\{A_0,A_1,...,A_K\}$ are parameters of the model, and $\bar{\mathbf{x}}=\{\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_K\}$ is the point in the input space where the model is believed to be most accurate, usually in the middle of the range of interest of the input variables. The basic principle of fuzzy regression is to find the most precise parameters such that a given $h$-cut of $f(\cdot|\mathbf{A},\bar{\mathbf{x}})$ includes all observed output of the system being modeled. Let $\{(Y_j,\mathbf{x}_j),\ j=1,2,...,n\}$ denote a sample of $n$ observations with $\mathbf{x}_j=\{x_{1,j},x_{2,j},...,x_{k,j}\}$ being the $j$th observation of the $K$ inputs, and let $f(\mathbf{x}_j,h=h_0|\mathbf{A},\bar{\mathbf{x}})$ denote an $h_0$-cut of $f(\mathbf{x}_j|\mathbf{A},\bar{\mathbf{x}})$. Note that each $Y_j$ can either be real or fuzzy, allowing for explicit modeling of sampling error structure. Only fuzzy models respecting the following conditions are acceptable:

$$Y_j(h_0) \subseteq f(\mathbf{x}_j, h=h_0 | \mathbf{A}, \bar{\mathbf{x}}),\ j=1,2,...,n \tag{6}$$

where (6) holds for a given level $h_0$. Of all values of **A** compatible with (6), parameters as precise as possible should be chosen. One possibility is to minimize the total width of these parameters, given by (7):

$$H(\mathbf{A}) = \sum_{k=0}^{K} \{\sup A_k(0) - \inf A_k(0)\} \tag{7}$$

but Bárdossy [1] has shown that minimizing prediction vagueness, given by (8), is preferable. It can be interpreted as the total area of the fuzzy model:

$$H(\mathbf{A}) = \int_{x_K^-}^{x_K^+} \int_{x_{K-1}^-}^{x_{K-1}^+} \cdots \int_{x_1^-}^{x_1^+} \int_{-\infty}^{+\infty} \mu_{f(\mathbf{x}|\mathbf{A},\bar{\mathbf{x}})}(y) \cdot dy \cdot dx_1 \cdot dx_2 \ldots dx_K \tag{8}$$

where $[x_k^-, x_k^+]$ defines the range of interest of the variable $x_k$. Minimizing $H(\mathbf{A})$ defined by (7) or (8) subject to the constraints (6) is a linear programming problem, which can be solved by standard techniques. Additional constraints must however be put on the shape of the fuzzy parameters for the solution to be unique [1]. If the parameters are taken to be LRS numbers such that $A_k=(m_k,\alpha_k,\beta_k)$(LRS), with $L$ and $R$ functions specified beforehand, then it is sufficient to impose $\alpha_k=c_k \cdot \beta_k$, where $c_k$ is a constant also chosen beforehand. Usually, the parameters are supposed to be symmetrical, with $\alpha_k=\beta_k$.

The choice of $h_0$, $L$ and $R$ is a controversial issue in fuzzy regression; their interpretation is unclear. The value of $h_0$ influences directly the precision of the fuzzy parameters. Small values of $h_0$ make the model more precise, larger ones make the model less precise. In fact for crisp data (non fuzzy $Y_j$) the only effect of choosing a non-zero value of $h_0$ is to multiply the coefficients $\alpha_k$ and $\beta_k$ respectively by $1/L^{-1}(h_0)$ and $1/R^{-1}(h_0)$. In that case it seems that $h_0$, $L$ and $R$ simply introduce an implicit security factor into the computations. Quite often in the literature, fuzzy regressions are computed with linear $L$ and $R$ functions, and with $h_0$=0.5, leading to an implicit multiplication by two of the regresssion coefficients $\alpha_k$ and $\beta_k$, and thus of the support of the fuzzy model.

As for fuzzy sensitivity analysis, lack of adequate software has limited the application of fuzzy regression. We hope to lift some of these difficulties by offering a simple, easy to use, yet powerful C++ library for performing fuzzy arithmetic, fuzzy sensitivity analysis and multiple fuzzy linear regression.

# Outline of the C++ library

The simplest way of adapting models for fuzzy sensitivity analysis and of programming fuzzy regression is to develop a new data type (or class) representing a fuzzy number, with all associated operators. This is only possible in an object-oriented language such as C++. We have in fact developed a hierarchy of classes modeling different types of fuzzy numbers: intervals, TFNs, TRFNs, LR and LRS fuzzy numbers, and even completely general fuzzy numbers, which are modeled by nested intervals, the number of which can be set by the user. The C++ class names chosen for those new types are respectively `interval`, `tfn`, `trfn`, `lr`, `lrs` and `fuzzy_number`. With this architecture, simpler types of fuzzy numbers can therefore be used whenever possible, minimizing both memory and computing time.

All necessary logic and arithmetic operators, together with a number of useful mathematical functions, have been programmed so that these different types of fuzzy numbers can be manipulated easily. It is also possible to explicitly cast any type of fuzzy number into another one. In particular, a general fuzzy number can be cast into an LR number, with the $L$ and $R$ functions automatically optimizing themselves to obtain a least-square fit. Standard input-output C++ operators have also been overloaded, and a class providing graphical outputs has also been devised. For now however it can only produce text-based graphics. To store collections of fuzzy numbers, a template for a generic array class has been devised, complete with operators allowing for direct arithmetic manipulation of matrix and vectors of fuzzy numbers. The multiple fuzzy linear regression algorithm, using a modified simplex algorithm for linear programming, has been implemented as a functor, which is a C++ encapsulated function. The library should be entirely portable; it has however only been tested with Borland C++™ 4.5 for Microsoft Windows™.

## Using the library for fuzzy sensitivity analysis

With the C++ library presented previously, adapting a model coded in C or C++ for fuzzy sensitivity analysis is quite simple. Of course, as mentioned earlier, code optimization to avoid using the same input variable more than once may be needed to decrease the fuzziness of the outputs, but is not necessary to ensure validity of the results. An application of fuzzy sensitivity analysis to a ground water hydrology problem illustrates the use of the library.

Pumping at constant rate in a homogeneous confined aquifer lowers water pressure, producing a steady-state drawdown of *s* meters at a distance of *r* meters from the well, which depends on the pumping rate *Q* ($m^3/s$), the range of action *R* (m) of the well, and the transmissivity *T* ($m^2/s$) of the aquifer [8]:

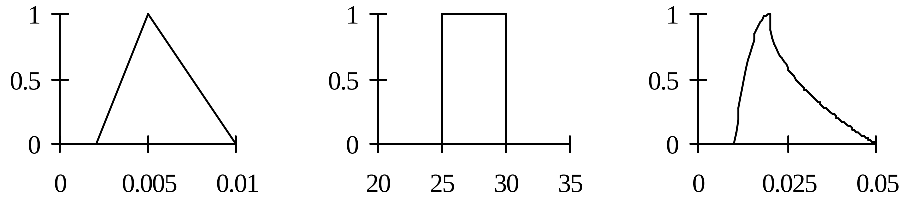
$$s = \frac{Q}{2\pi T} \log(R/r) \tag{9}$$

where log(·) denotes Napierian logarithms. For example, with $T$=0.005 $m^2/s$, $R$=28 km, $Q$=0.02 $m^3/s$ the drawdown can be computed to be $s$=7.5 m at a distance of $r$=20 cm from the well, using the following C++ code (characters in bold identify instructions which will need to be changed to adapt the program for fuzzy sensitivity analysis):

```
#include <math.h>         // LIBRARY FOR log FUNCTION
#include <iostream.h>     // LIBRARY FOR i/o
void main() {
      // INITIALIZATION OF THE INPUT PARAMETERS
      float T(.005);
      float R(28000);
      float Q(.02);
      float r(.2);
      float s;

      // COMPUTING AND PRINTING THE OUTPUT OF THE MODEL
      s = Q/(2*3.1416*T) * log(R/r);
      cout << s;}
```

In daily hydrogeology practice, the main difficulty in solving such problems is the lack of accuracy on input parameters, especially transmissivity (*T*) which depends both on the depth of the aquifer and its hydraulic conductivity. Suppose now that *T* is only known to be between 0.002 and 0.01 $m^2/s$, and is probably around 0.005 $m^2/s$. $T$=(.002,.005,.01)(TFN) could model this information. In the same manner, the following fuzzy numbers could be obtained for *R* and *Q*:



*T*=(.002,.005,.01)(TFN) m$^2$/s  *R*=[25,30] km  *Q*=(.02,.01,.03,2,.5)(LRS) m$^3$/s

Figure 3: membership function of *R*, *Q* and *T*

Modifying the preceding program to take into account uncertainty is straightforward. It involves adding a new library for fuzzy arithmetic and modifying the initialization of the input parameters, but not the actual code responsible for computing and printing the outputs. Here is the same model coded for fuzzy sensitivity analysis, with modifications in bold (note that only one line was added, to include the new library):

```
#include <math.h>         // LIBRARY FOR log FUNCTION
#include <iostream.h>     // LIBRARY FOR i/o
#include "fuzynmbr.h"     // LIBRARY FOR FUZZY ARITHMETIC
void main() {
      // INITIALIZATION OF THE INPUT PARAMETERS
      tfn                 T(.002,.005,.01);
      interval            R(25000,30000);
      lrs                 Q(.02,.01,.03,2,.5);
      float               r(.2);
      fuzzy_number        s;
      // COMPUTING AND PRINTING THE OUTPUT OF THE MODEL
      s = Q/(2*3.1416*T) * log(R/r);
      cout << s;}
```

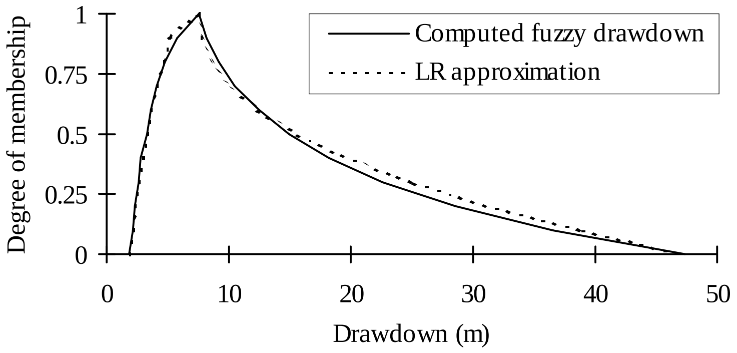


Figure 4: Uncertainty on computed drawdown

Notice that various types of fuzzy numbers can be freely mixed in a same formula. The output is of type `fuzzy_number` since no simpler parametric type can model it exactly, given the operators used in the model. Except for very simple models, such as linear one, this is always the case. However, if optimizing memory is of concern, *s* could be modeled as an LR number, with minimal loss of accuracy. Figure 4 shows the output of the fuzzy model, both if *s* is forced to be an LR number, and if it is modeled as a general fuzzy number, represented by 10 nested intervals (the default choice - which can be overridden). In most cases, using LR numbers to model uncertainty seems sufficient. Note that the mode of the fuzzy number, which corresponds to the most possible value of *s* is around 7,5 m, but *s* could be anywhere between 2 and 47 m. An *h*-cut of *h*=0.5 shows that values less than 3 m and larger than 15 m have a possibility of 0.5, meaning that the probability of these extreme events could be as much as one half (since the possibility measure is an upper bound on probability).

With little efforts, a model coded in C++ has been adapted for fuzzy sensitivity analysis using the proposed library, allowing for a quick estimation of uncertainty on the output of the model. The resulting fuzzy number gives both an idea of the most likely value and of worst possible cases. Modeling of uncertainty on the inputs is much simpler than using probability theory, since only an upper bound on the probability is needed, and computations are by far simpler and faster.

## Using the library for fuzzy linear regression

The advantages of fuzzy regression over statistical regression are not simplicity and speed. Compared with statistical regression, fuzzy regression is complex and slow, especially for large sample sizes. Fuzzy regression is however helpful precisely when data is scarce, since statistical regression cannot be applied, for it is then impossible the verify its underlying hypotheses or to assess the quality of the fit. As an example of application of fuzzy regression, we discuss the problem of finding the relationship between average annual flow at two measuring points (SM2 and SM3) located on the Sainte-Marguerite river, on the north shore of the Saint-Lawrence river in Québec.

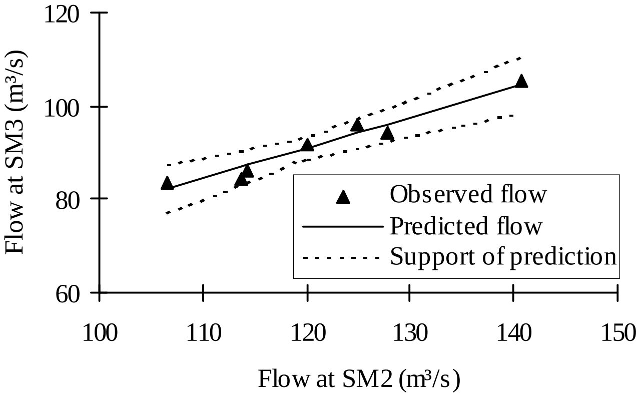


Figure 5: Fuzzy linear regression predicting flow at SM3 from flow at SM2

The sample size of only 7 years (1987-1993) would be considered too small for a meaningful statistical regression. However, for fuzzy linear regression it is quite reasonable. Let $Q_2$ be the flow at SM2, and $Q_3$ be the flow at SM3. To estimate the flow at SM3 from SM2, we propose the following fuzzy model, where the parameters $\mathbf{A}=\{A_0,A_1\}$ are symmetrical TFNs, and $\bar{x}$ is the average interannual flow at SM2:

$$Q_3=f(Q_2|\mathbf{A},\bar{x})=A_0+A_1(Q_2-\bar{x}) \tag{10}$$

With the degree of belief *h* in the fuzzy model set to *h*=0,5 and prediction vagueness being minimized over the range of observed flow, the parameters are $A_0$=(89.3,91.7,94.0)(TFN) and $A_1$=(.46,.65,.83)(TFN). Figure 5 shows the fuzzy regression obtained.

Since the observed flow fall quite on a straight line, the width of the predicted fuzzy number is relatively small, and the predicted flow, which corresponds to the value of the fuzzy prediction with maximum possibility (the mode), seems sensible. It should be stressed that the support of the predictions, $f(Q_2,h=0|\mathbf{A},\bar{x})$, is not a confidence interval, neither for the prediction line nor for the observations. It rather limits the space containing the set of possible relationships between $Q_2$ and $Q_3$, given the data and the degree of belief in the model ($h$=0,5 in this case). The shape of these limits has been shown [2] to be a good indicator of the quality of the fit. Indeed, when the model is not adequate, the support of the prediction becomes clearly out of proportion since the parameters must be very vague for the model to include all observations. A consequence of that is of course a sensitivity of fuzzy regression to outliers. Care should therefore be taken to avoid including erroneous data when using fuzzy regression.

## Discussion and Conclusion

Fuzzy number arithmetic provides a simple framework for sensitivity analysis in complex models, but has until now found few applications for lack of adequate software tools. We developed a C++ library of object for easily incorporating fuzzy sensitivity analysis into new or existing models coded in C or C++. The library also has capabilities for multiple fuzzy linear regression, and can be used to help study uncertainty in linear models when data is scarce. Examples of applications show that adapting a model for fuzzy sensitivity analysis requires minimal modifications to existing code, and that using fuzzy regression is relatively simple.

While C++ classes are adapted tools for fuzzy sensitivity analysis, since they need to be incorporated into the code, it would be useful to possess a more user-friendly tool for fuzzy regression. Fuzzy regression could be needed in a more complex model, but most applications only require the fuzzy relationship to be established from the available data. We are currently devising a software dedicated to fuzzy linear regression which will use the available libraries, and should be available by the end of the year.

## Acknowledgment

The financial support of the Natural Science and Engineering Research Council of Canada and of Hydro-Québec is gratefully acknowledged.

## References

1. Bárdossy, Á. Note on fuzzy regression, *Fuzzy Sets and Systems*, 1990, **37**, 65-75.
2. Bárdossy, Á., Bogárdi, I. & Duckstein, L. Fuzzy Regression in Hydrology. *Water Resources Research*, 1990, **26**(7), 1497-1508.
3. Berger, J.O. *Statistical Decision Theory and Bayesian Analysis*, 2nd ed., Springer-Verlag, New York, 1985.
4. Caselton, W.C. & Luo, W. Decision Making with Imprecise Probabilities - Dempster-Shafer Theory and Application, *Water Resources Research*, 1992, **28**(12), 3071-3083.
5. Dubois, D. & Prade, H. Fuzzy sets and statistical data. *European Journal of Operational Research*, 1986, **25**: 345-356.
6. Dubois, D. & Prade, H. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York, 1988.
7. Ferson, S. & Kuhn, R. Propagating Uncertainty in Ecological Risk Analysis Using Interval and Fuzzy Arithmetic, in Envirosoft/92 (ed. P. Zanetti), pp. 387-401, *Proceedings of the 4th Int. Conf. on Computer Techniques in Environmental Studies*, Computational Mechanics Publications, Southampton, 1992.
8. Heath, R.C. *Basic Ground-Water Hydrology*, U.S. Geological Survey Water-Supply Paper 2220, 1983.
9. Kaufmann, A. *Introduction to the Theory of Fuzzy Subsets*, Academic Press, New York, 1975.

10. Kaufmann, A. & Gupta, M.M. *Introduction to fuzzy arithmetic: Theory and applications*, Van Nostrand Rheinhold, New York, 1991.
11. Moore, R. *Methods and Applications of Interval Analysis*, SIAM Studies on Applied Mathematics, Vol. 2, Philadelphia, 1979.
12. Shafer, G. *A Mathematical Theory of Evidence*, Princeton University Press, New Jersey, 1976.
13. Tanaka, H., Uejima, S. & Asai, K. Linear regression analysis with fuzzy model, *IEEE Trans. Syst. Man Cybern.*, 1982, SMC-12, 903-907.
14. Walley, P. *Statistical Reasoning with Imprecise probabilities*, Chapman and Hall, London, 1990.
15. Zadeh, L.A. Fuzzy sets. *Information and Control*, 1965, **8**, 338.