# Soft Computing Explains Heuristic Numerical Methods in Data Processing and in Logic Programming

**Hung T. Nguyen**
Department of Mathematical Sciences
New Mexico State University
Las Cruces, NM 88003, USA
hunguyen@nmsu.edu

**Vladik Kreinovich**
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@cs.utep.edu

**Bernadette Bouchon-Meuiner**
LIP6, UPMC, Case 169
4 place Jussieu
75252 Paris Cédex 05
France
bouchon@poleia.lip6.fr

**Abstract**

We show that fuzzy logic and other soft computing approaches explain and justify heuristic numerical methods used in data processing and in logic programming, in particular, M-methods in robust statistics, regularization techniques, metric fixed point theorems, etc.

## Introduction

**What is soft computing good for? Traditional viewpoint.** When are soft computing methods (fuzzy, neural, etc.) mostly used now? Let us take, as an example, *control*, which is one of the major success stories of soft computing (especially of fuzzy methods; see, e.g., (Klir 1995)).

- In control, if we know the exact equations that describe the controlled system, and if we know the exact objective function of the control, then we can often apply the optimal control techniques developed in traditional (crisp) control theory and compute the *optimal* control.

  Even in these situations, we can, in principle, use soft computing methods instead: e.g., we can use simpler fuzzy control rules instead of (more complicated) traditional techniques. As a result, we may get a control that is much easier to compute but that it somewhat worse in quality.

- However, the major application of soft computing in control is to the situations when we only have *partial* knowledge about the controlled system and about the objective functions and in which, therefore, traditional optimal control theory is not directly applicable. Here is where all known success stories come from: utilities like washing machines or camcoders, car parking automation, and other applications all share one thing: they all have to operate in a partially known environment.

From this viewpoint, as we gain more and more knowledge about a system, a moment comes when we do not need to use soft computing techniques any longer: when we have accumulated enough knowledge, we will then be able to use traditional (crisp) techniques.

From this viewpoint, soft computing methods look like a (successful but still) *intermediate* step, "poor man's" data processing techniques, that need to be used only if we cannot apply "more optimal" traditional methods.

**Another possible use of soft computing: it has a great potential for justifying heuristic methods.** The above viewpoint summarizes the existing usage of soft computing techniques: currently, these methods are, indeed, mainly used only when we lack information. However, as we will try to show in this paper, the potential for soft computing techniques is much broader.

Indeed, let us assume that we have the complete information, and so, we can use some crisp data processing algorithms. Where do these algorithms come from? If several data processing algorithms are applicable, which of these algorithms should we choose? In a few cases, these algorithms have a profound mathematical justification, but in most cases, these methods are *heuristic* in the sense that their justification comes from informal arguments rather than from the formal proofs.

Now, "informal" means formulated in a natural language, not in the language of mathematics. So, to justify these methods, we must *formalize* this natural-language description in precise terms. This is exactly what soft computing (especially fuzzy logic) is doing.

So, soft computing has a great potential in justifying heuristic methods.

**What we are planning to do.** In this paper, we will show that soft computing methods can indeed explain heuristic methods, both in more traditional data processing and in intelligent data processing techniques (e.g., related to logic programming).

## Data processing methods: a useful classification

Before we explain how soft computing can be used, let us briefly classify the existing methods of data processing by the complexity of results that we want to achieve:

- In some problems, all we want is one or several numerical values. It may be that we measure some characteristics, or it may be that we know the model, and we want, based on the experimental data, to estimate the parameters of this model. These problems are usually handled by *statistical methods.*
- In other problems, we want to know a function. We may want to reconstruct an image (brightness as a function of coordinates), we may want to filter a signal (intensity as a function of time), etc. These methods are usually handled by different *regularization* techniques.
- Finally, there are even more complicated problems, in which we want to reconstruct a *model* of an analyzed system. Methods that handle such problems are called *intelligent* data processing methods. Many of these methods are based on *logic programming*, a formalism that (successfully) describes complicated logical statements algorithmically, in a kind of programming language terms.

In this paper, we will show that soft computing explains heuristic data processing methods of all three types.

## Soft computing explains heuristic methods in statistics

**Two types of heuristic methods in statistics.** Most statistical methods are well justified. There are only two areas where heuristic methods are still used (see, e.g., (Wadsworth 1990)):

- While we are computing and transforming probabilities, we are on the safe ground of well-justified techniques. Whatever estimate we get, we can always compute the probabilities of different possible errors for this estimates, the probabilities for different errors in estimating the above-mentioned probabilities, etc. However, comes a time when we need to make precise recommendations (make decisions). And here is when we often have to use heuristic methods.

  Let us give a simple example of what we mean. Measurement errors usually have a Gaussian distribution, with some standard deviation $\sigma$. We usually know this value $\sigma$ after testing the measuring instrument. A natural question is: if we have measured the value $\widetilde{x}$, what are the possible actual values of the measured quantity $x$?

  The literal answer is "any", because for Gaussian distribution, arbitrary large deviations $\widetilde{x} - x$ have non-zero probability: with probability 5%, we can have errors greater than $2\sigma$, with probability 0.1%, errors greater than $3\sigma$, etc. Suppose that we bought an instrument, made a measurement, and the error was greater than $2\sigma$. Well, we would conclude that it is possible. But if an error was greater than $6\sigma$ (which corresponds to the probability $\approx 10^{-6}\%$), we would simply conclude that it cannot be a random error: we would ask the manufacturer to replace the faulty instrument, and he will most probably replace it (or at least repair it).

  This common sense decision is based on the heuristics idea that if some event has a very small probability (like $\le 10^{-6}\%$), then this event cannot occur. If you walk into a casino and a roulette, which is supposed to be random, stops on red 100 times in a row you would conclude that it is broken.

  This idea, however, is very difficult to formalize, because we can always divide the real line into small enough segments so that the probability of bing in each of them is $\le 10^{-6}\%$. Thus, if we simply postulate that such low probability events do not happen, we will have a make a nonsensical conclusion that no value of error is possible at all. Thus, we have to use *heuristic* methods.

- Another situation in which we have to use heuristic methods is *robust statistics* (see, e.g., (Huber 1981) and (Wadsworth 1990)): If we know the exact values of the probabilities, then we can use traditional well-justified heuristic methods. However, often, we do not know these probabilities. What methods should we then use? For example, let us consider the situation in which we want to find the parameters $C_1, \ldots, C_p$ of a model $y = f(x_1, \ldots, x_n, C_1, \ldots, C_p)$ based on the results $(x_1^{(k)}, \ldots, x_n^{(k)}, y^{(k)})$, $1 \le k \le K$, of several ($K$) measurements in which we measure both the inputs $x_i$ and the output $y$. Ideally, we should find the values of $C_i$ for which the model exactly predicts all measurements results, i.e., for which all prediction errors $e_k = y^{(k)} - f(x_1^{(k)}, \ldots, x_n^{(k)}, C_1, \ldots, C_p)$ are equal to 0. This requirement makes perfect sense for an idealized situation in which all the values of $y$ are measured with an absolute accuracy. However, since measurements are never 100% accurate, even if the model is precise, the measured values $y^{(k)}$ will still differ from the actual values, and thus, from the model's prediction. Thus, we cannot require that the errors are exactly equal to 0; we can only require that these errors $e_1, \ldots, e_K$ are *small.*

  When errors are normally distributed, then a natural way to determine the parameters is by using the least squares method $e_1^2 + \ldots + e_K^2 \to \min$. If we do not know the probabilities, we can use a class of heuristic methods called *M-methods*, in which we find the parameters $C_1, \ldots, C_p$ from the condition

  $$\psi(e_1) + \ldots + \psi(e_K) \to \min \qquad (1)$$

  for some function $\psi(x)$. These methods often work fine, but they are still heuristic. How can we justify them?

  The next question is: what function $\psi(x)$ should we choose?

Let us show that both types of heuristic methods can be justified by using soft computing (namely, fuzzy logic).

**Formalization of the requirement that events with small probabilities cannot happen.** We want to describe the requirement that an event with a sufficiently small probability, i.e., with a probability that does not exceed a certain small number $p_0 \ll 1$, cannot occur. The problem with a standard probabilistic justification of this requirement is that we may have many events $E_1, \ldots, E_m$ each of which has a very small probability $p(E_i) \leq p_0$. There is no problem with excluding each of these events. However, when we want to exclude *all* of them, the excluded event $E = E_1 \vee \ldots \vee E_m$ can have a very high probability (all the way up to 1).

To avoid this problem, we can, in situations of partial information, use *degrees of belief* $d(E_i)$ instead of probabilities $p(E_i)$. In this case, the degree of belief $d(E)$ of the entire excluded part $E = E_1 \vee \ldots \vee E_m$ can be determined from the degrees of belief $d(E_i)$ by using a t-conorm $a \vee b$, a fuzzy analogue of "or" (Klir and Yuan 1995), (Nguyen and Walker 1997): $d(E) = d(E_1) \vee \ldots \vee d(E_m)$. The simplest t-conorm is $a \vee b = \max(a, b)$. For this t-conorm, if $d(E_i) \leq p_0$ for all $i$, then $d(E) = \max(d(E_1), \ldots, d(E_m))$ is also $\leq p_0$. Thus, we can safely exclude all these events.

**Justification of M-methods in robust statistics.** Informally, the requirement for choosing the parameters of the model is that all the errors $e_i$ are small, i.e., that $e_1$ is small, $e_2$ is small, ..., and $e_K$ is small. A natural way to formalize this requirement is to use fuzzy logic. Let $\mu(x)$ be a membership function that describes the natural-language term "small". Then, our degree of belief that $e_1$ is small is equal to $\mu(e_1)$, our degree of belief that $e_2$ is small is equal to $\mu(e_2)$, etc. To get the degree of belief $d$ that all $K$ conditions are satisfied, we must use a t-norm (a fuzzy analogue of "and"), i.e., use a formula $d = \mu(e_1) \& \ldots \& \mu(e_K)$, where & is this t-norm.

In (Nguyen, Kreinovich, and Wojciechowski 1997), we have shown that within an arbitrary accuracy, an arbitrary t-norm can be approximated by a strictly Archimedean t-norm. Therefor, for all practical purposes, we can assume that the t-norm that describes the experts' reasoning, is strictly Archimedean and therefore, has the form $a \& b = \varphi^{-1}(\varphi(a) + \varphi(b))$ for some strictly decreasing function $\varphi$ (Klir and Yuan 1995), (Nguyen and Walker 1997). Thus, $d = \varphi^{-1}(\varphi(\mu(e_1)) + \ldots + \varphi(\mu(e_K)))$. We want to find the values of the parameters for which our degree of belief $d$ (that the model is good) is the largest possible. Since the function $\varphi$ is strictly decreasing, $d$ attains its maximum if and only if the auxiliary characteristic $D = \varphi(d)$ attains its minimum. From the formula that describe $d$, we can conclude that $D = \varphi(\mu(e_1)) + \ldots + \varphi(\mu(e_K))$. Thus, the condition $D \to \min$ takes the form (1), with $\psi(x) = \varphi(\mu(x))$.

So, we have indeed justified the M-methods. This justification enables us to answer the second question: what function $\psi(x)$ should we choose. We should base this choice on the opinion of the experts. From these experts, we extract the membership function $\mu(x)$ that corresponds to "small", and the function $\varphi(x)$ that best describes the experts' "and".

*Comment.* In image processing, M-methods are called *generalized entropy methods*, and the function $\psi(x)$ is called a *generalized entropy* function. In (Flores, Ugarte, and Kreinovich 1993), (Mora, Flores, and Kreinovich 1994), and (Flores, Kreinovich, and Vasquez 1997), we have successfully used this method for radar imaging (including planetary radar imaging). For such problems, minimization of the function (1) is a difficult task; to compute the minimum, we used another soft computing technique: *genetic algorithms.*

## Soft computing explains heuristic methods in regularization

**Heuristics.** One of the main problems in reconstructing a signal is that this problem is *ill-defined* in the following sense: if we know the values $x(t_k)$ of the signal $x(t)$ for several consecutive moments of time $t_1, \ldots, t_n$, we can, in principle, get arbitrary values in between. To make meaningful conclusions, we need to restrict the class of signals to signals that smoothly depend on time, i.e., which cannot deviate largely from $x(t_k)$ while $t$ goes from $t_k$ to the next value $t_{k+1}$. Thus, we have two problems here:

- first, how can we limit the smoothness of the signal, and
- second, when we have fixed the smoothness, which extrapolation techniques should we choose.

In both cases, we mostly have to use *heuristics* methods. Let us show that these methods can be justified by using fuzzy logic.

**Justification of smoothness.** Intuitively, the unknown signal must satisfy the property that its values $x(t)$ and $x(s)$ in two close moments of time should be, themselves, close to each other. In other words, we must have the following implication: if $t$ and $s$ are close, then $x(t)$ and $x(s)$ should be close. How can we formalize this requirement?

In classical (two-valued) logic, the statement "if $A$ then $B$" is equivalent to the inequality $t(A) \leq t(B)$, where $t(A)$ and $t(B)$ are the truth values of the statements $A$ and $B$ ($t(A) = 1$ if $A$ is true, and $= 0$ otherwise). Similarly, in fuzzy logic, if we have a 100% belief in the implication "if $A$ then $B$", it is usually interpreted as $d(A) \leq d(B)$, where $d(A)$ and $d(B)$ are the degrees of belief in $A$ and $B$, respectively. So, the above statement can be reformulated as the inequality $d(A) \leq d(B)$, where $d(A)$ is the degree of belief that $t$

and $s$ are close, and $B$ is the degree of belief that $x(t)$ and $x(s)$ are close.

Intuitively, the degree of closeness between the two moments of time $t$ and $s$ depends only on the interval between them, i.e., on the difference $t - s$. In other words, $t$ is close to $s$ if and only if the difference $t-s$ is small. Similarly, $x(t)$ is close to $x(s)$ if and only if the difference $x(t)-x(s)$ is small. To describe these degrees of belief in precise terms, we must know the membership functions corresponding to the word "small". Let $\mu(x)$ be the membership function that describe "smallness" of time intervals.

The difference $x$ and $-x$ are of the same size, so the degree of belief that $x$ is small must be the same as the degree of belief that $-x$ is small. Hence, $\mu(x) = \mu(-x)$, i.e., $\mu(x)$ is an *even* function.

The larger $x > 0$, the less reasonable it is to call $x$ small, thus, the membership $\mu(x)$ must be *strictly decreasing* for $x > 0$.

It is natural to assume that smallness of *signal* intervals (of type $x(t) - x(s)$) is described by a similar membership function, with the only difference that signals may be measured in different units. Therefore, we describe "small" for signals as $\mu((x(t) - x(s))/k)$ for some multiplicative constant $k$ that describes the possible difference in units.

Thus, the above inequality takes the form $\mu(t-s) \le \mu((x(t) - x(s))/k)$ for all $t$ and $s$. Since the function $\mu$ is even, we have $\mu(x) = \mu(|x|)$, and the above inequality can be re-written as $\mu(|t-s|) \le \mu(|x(t) - x(s)|/k)$. Since $\mu(x)$ is a strictly decreasing function, this inequality is equivalent to $|t-s| \ge |x(t) - x(s)|/k$, i.e., to $|x(t) - x(s)| \le k \cdot |t-s|$, and

$$\frac{|x(t) - x(s)|}{|t-s|} \le k.$$

Thus, out of a fuzzy informal restriction, we got a crisp restriction on the signal $x(t)$. When $s \to t$, we conclude that the time derivative $\dot{x}(t)$ of the signal $x(t)$ is limited by $k$.

Moreover, from the experts, we can elicit the membership functions that correspond to "small" for time intervals and for signal values, and therefore, extract the value $k$ that limits the time derivative.

**Justification of traditional regularization methods.** In (Kreinovich et al. 1992), we have shown that if we formalize the statements like "the derivative $\dot{x}(t)$ of the signal $x(t)$ should be small for all $t$", and "the value of the signal itself should not be too large", then we arrive at the extrapolation techniques called *regularization*, that chooses a signal $x(t)$ for which $\int (x(t))^2\,dt + \lambda \cdot \int (\dot{x}(t))^2\,dt \to \min$, and that, moreover, thus justification enables us to find the value of the regularization parameter $\lambda$ based on the experts' knowledge.

## Soft computing explains heuristic methods in logic programming

**Heuristic numerical methods in logic programming.** Logic programming is based on traditional, two-valued logic, in which every statement is either true or false (inside the computer: 1 or 0). However, in proving results about logic programs, and in computing the answer sets, it is often helpful to use numbers in between 0 and 1. These intermediate numbers are usually introduced *ad hoc*, without any meaningful interpretation, as heuristic tools.

One of the cases when such numbers are used is the use of *metric fixed point theorems* (Fitting 1994) (see also (Khamsi, Kreinovich, and Misane 1993)). Many methods of logic programming used the idea of a sequential approach to the answer, in which we start with some (possibly incorrect) interpretation $s_0$, and then apply some reasonable correcting procedure $s \to C(s)$ until no more corrections are necessary. An interpretation $s$ is usually described by describing, for each of the atomic properties $p_1, \ldots, p_n, \ldots$ (i.e., properties that are either directly used in the logic program, or that can be constructed based on the ones used), whether each particular property $p_i$ is true or not. In other words, $s = (s_1, \ldots, s_n, \ldots)$, where $s_i = 1$ if $p_i$ is true, and $= 0$ otherwise. In terms of the correcting procedure, the fact that no more corrections are necessary takes the form $C(s) = s$. In mathematical terms, this means that the resulting model $s$ is a *fixed point* of the correcting transformation $C$.

Fixed points are well-analyzed in mathematics. Therefore, to prove the existence of answer sets and the convergence of the above iterative procedure, it is reasonable to use the existing mathematical fixed point theorems. Traditionally, in view of the discrete character of traditional logic programming models, in logic programming, only *discrete-valued* fixed point theorems were used. Fitting (Fitting 1994) was the first to successfully apply *continuous-valued* (namely, metric) fixed point theorems in logic programming. The main idea behind these methods is that we introduce a numerical metric to describe the distance $\rho(t,s)$ between two interpretations $t$ and $s$.

Metric methods are used for logic programs in which the atomic properties are naturally stratified:

- the first layer consists of the properties that can be easily deduced directly from the logic program;
- the second layer contains the ones that be determine indirectly, i.e., for which, in addition to the rules, we must know the truth values of the atomic statements from the first layer;
- similarly, we can define third, fourth layers, etc.

The distance $\rho(t,s)$ is then defined as $2^{-L}$, where $L$ is the first layer on which $t$ and $s$ differ.

*Comments.*

- From the viewpoint of metric fixed point theory, the choice of $2^{-L}$ does not really matter; instead, we could use any strictly decreasing sequence $d_L$.
- How can we justify this metric?

**Soft computing justification.** Experts sometimes err. Hence, when an expert assigns the truth values $t_1, \ldots, t_n, \ldots$ to the properties $p_i$, his assignments may differ from the actual (unknown) truth values $T_1, \ldots, T_n, \ldots$ of the properties of the described object.

As a result, even when we have two different interpretations $t \neq s$, it is still possible that these interpretations describe the same object ($T = S$), and the difference is simply due to the experts' errors. Intuitively, the "closer" $t$ and $s$, the larger our degree of belief $d(T = S)$ that the (unknown) actual interpretations $T$ and $S$ coincide. Thus, we can define the metric $\rho(t, s)$ as $1 - d(T = S) = d(T \neq S)$. Let us show that this natural definition leads exactly to the desired metric.

Indeed, $T \neq s$ if and only if $\exists n(T_n \neq S_n)$. An existential quantifier is, in effect, an infinite sequence of "or" operations; so, to make the conclusion meaningful, we must use max as an "or" operation (see the detailed explanation of this choice in (Nguyen et al. 1997)). Hence,

$$d(T \neq S) = \max_n d(T_n \neq S_n).$$

If for some property $p_n$, we have $t_n = s_n$, then for this property, we have no reasons to believe that $T_n \neq S_n$. Therefore, for this $n$, $d(T_n \neq S_n) = 0$, and in the maximum, we can only take into consideration the values $n$ for which $t_n \neq s_n$.

How do the values $d(n) = d(T_n \neq S_n)$ differ?

- If the corresponding property $p_n$ can be directly extracted from the rules, then we are the most confident in the expert estimates. So, if an expert does say that $t_n \neq s_n$, we take $d(n)$ to be equal to 1 (or close to 1). We will denote the corresponding degree of belief by $d_1$.
- For statements from the second layer, we need to use additional rules to justify them. Experts may not be 100% sure in these rules. Hence, our degree of belief in whatever conclusions we make by using these rules should be smaller than in conclusions that do not use these rules. As a result, for the properties $p_n$ from the second layer, our degree of belief $d(n)$ that the expert is right and that $T_n \neq S_n$ is smaller than for the first layer. If we denote the degree of belief for properties from the second layer by $d_2$, we thus conclude that $d_2 < d_1$.
- Similarly, the degrees of belief $d_3, d_4, \ldots$ corresponding to different layers is a strictly decreasing sequence: $d_1 > d_2 > d_3 > \ldots$

For each $n$, the degree of belief $d(T_n \neq S_n)$ is equal to $d_L$, where $L$ is the layer that contains the property $p_n$. The smaller $L$, the larger $d_L$. Therefore, the distance $\rho(t, s) = d(T \neq S)$, which is equal to the maximum of these values $d(n)$ for all $n$ for which $t_n \neq s_n$, is equal to $d_L$ for the smallest $L$ that contains a property $n$ for which $t_n \neq s_n$. In other words, the distance $\rho(t, s)$ is equal to $d_L$ for the first layer $L$ on which $t$ and $s$ differ. *This is exactly the desired metric.*

*Comment.* Similarly to the previous cases, soft computing methods not only *justify* methods of logic programming, but also help to find appropriate algorithms for solving logic programming problems; see, e.g., (Kreinovich 1997), (Kreinovich and Mints 1997), and references therein.

## Acknowledgments

This work was partially supported by NSF Grants No. EEC-9322370 and DUE-9750858, by NASA under cooperative agreement NCCW-0089, and by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, under grant number F49620-95-1-0518.

This work was partially done when V. Kreinovich was an invited professor at Laforia, University Paris VI, Summer 1996.

## References

Fitting, M. 1994. Metric methods: Three examples and a theorem. *Journal of Logic Programming* 21(3): 113–127.

Flores, B. C., Ugarte, A., and Kreinovich, V. 1993. Choice of an entropy-like function for range-Doppler processing. In Proceedings of the SPIE/International Society for Optical Engineering, Vol. 1960, Automatic Object Recognition III, 47–56.

Flores, B. C., Kreinovich, V., and Vasquez, R. 1997. Signal design for radar imaging and radar astronomy: genetic optimization, In: Dasgupta, D., and Michalewicz, Z. eds. *Evolutionary Algorithms in Engineering Applications*, Berlin-Heidelberg: Springer-Verlag 406–423.

Huber, P. J. 1981. *Robust statistics.* N.Y.: Wiley.

Khamsi, M. A., Kreinovich, V., and Misane, D. 1993. A new method of proving the existence of answer sets for disjunctive logic programs: a metric fixed point theorem for multi-valued maps. In Proceedings of the Workshop on Logic Programming with Incomplete Information, Vancouver, B.C., Canada, October 1993, 58–73.

Klir, G., and Yuan, B. 1995. *Fuzzy sets and fuzzy logic: theory and applications.* Upper Saddle River, NJ: Prentice Hall.

Kreinovich, V., Chang, C.-C., Reznik, L., and Solopchenko, G. N. 1992. Inverse problems: fuzzy representation of uncertainty generates a regularization. In Proceedings of NAFIPS'92: North American Fuzzy Information Processing Society Conference, Puerto Vallarta, Mexico, December 15–17, 1992, 418–426. Houston, TX: NASA Johnson Space Center.

Kreinovich, V. 1997. S. Maslov's Iterative Method: 15 Years Later (Freedom of Choice, Neural Networks, Numerical Optimization, Uncertainty Reasoning, and Chemical Computing), In: Kreinovich, V., and Mints, G. eds. 1997. *Problems of reducing the exhaustive search.* Providence, RI: American Mathematical Society, 175–189.

Kreinovich, V., and Mints, G. eds. 1997. *Problems of reducing the exhaustive search.* Providence, RI: American Mathematical Society (AMS Translations — Series 2, Vol. 178).

Mora, J. L., Flores, B. C., and Kreinovich, V. 1994. Suboptimum binary phase code search using a genetic algorithm. In: Udpa, S. D., and Han, H. C. eds. *Advanced Microwave and Millimeter-Wave Detectors.* Proceedings of the SPIE/International Society for Optical Engineering, Vol. 2275, San Diego, CA, 1994, 168–176.

Nguyen, H. T., Kreinovich, V., Nesterov, V., and Nakamura, M. 1997. On hardware support for interval computations and for soft computing: a theorem. *IEEE Transactions on Fuzzy Systems* 5(1): 108–127.

Nguyen, H. T., Kreinovich, V., and Wojciechowski, P. 1997. Strict Archimedean t-Norms and t-Conorms as Universal Approximators, Technical Report No. UTEP-CS-97-3, Department of Computer Science, University of Texas at El Paso; the LaTeX file can be downloaded via a link in the second author's homepage http://cs.utep.edu/csdept/faulty/kreinovich.html.

Nguyen, H. T., and Walker, E. A. 1997. *A first course in fuzzy logic.* Boca Raton, Florida: CRC Press.

Wadsworth, H. M. Jr. ed. 1990. *Handbook of statistical methods for engineers and scientists.* N.Y.: McGraw-Hill Publishing Co.