Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

FUZZY SET METHODS FOR QUALITATIVE AND NATURAL LANGUAGE ORIENTED SIMULATION

Paul A. Fishwick

Department of Computer and Information Science
University of Florida
Bldg. CSE, Room 301
Gainesville, Florida 32611

ABSTRACT

Qualitative methodology plays an important role within computer simulation; modeling and analysis of complex systems require qualitative methods since humans think naturally in qualitative and linguistic terms. The critical interface for simulationists exploring qualitative simulation should rely on an unambiguous mathematical formalism or method with foundations in systems theory. Currently, many *ad hoc* formalisms exist for encoding uncertain or qualitative simulation knowledge; however, we have found that fuzzy set theory provides for a formalism where linguistic variables can be encoded as state, parameter, input and output information in the model. Fuzzy numbers, in particular, are useful when population statistics are unavailable — usually due to cost factors. We have constructed fuzzy simulation programs based on our C-based *SimPack* library and we use fuzzy simulation to hypothesize qualitative system models reflecting real system behavior, and to specify qualitative versions of systems.

1. INTRODUCTION

The concept of *quality* in computer simulation and system modelling relates to *invariance* [Fishwick 1990b] and uncertainty [Klir 1990] in modelling. Methods in qualitative simulation [Fishwick and Luker 1990] are a concern in both the simulation and the artificial intelligence (AI) [Fishwick and Zeigler 1989] communities. How are simulation models created over time? What are the engineering methods [Fishwick 1989], pruning approaches [Zeigler 1984,1989] and tools used in the automation enterprise? This represents an extremely complex problem which needs to be addressed if we are to create design approaches for high autonomy systems [Zeigler 1990a,b]. Our goal in this paper is to present a method of computer simulation with fuzzy sets as an integral part of the overall automation process; as complex systems such as mobile robots gain greater degrees of autonomy in the workplace, we will need effective ways of communicating our human concepts to them. Since many of these concepts are linguistic and "fuzzy" in nature, we feel that fuzzy simulation has a potential to provide for an improved robot-human interface. In addition, fully autonomous robots may communicate fuzzy concepts to one another if they are constrained by time.

A human asks a household robot, for instance, "How much of the house will be painted when I return this afternoon." The robot, requiring a natural language interface, may respond "Most of it." Let's examine briefly how the robot could derive such an answer. There are so many fuzzy factors involved in the planning process of painting that the robot can only respond with a relatively fuzzy answer - this fuzzy answer can be the result of having internally simulated (through fuzzy means) a candidate plan for the painting job — candidate plans are hypothesized by the robot, and then the robot simulates an efficient plan that satisfies all constraints. Thus, simulation plays a fundamental role within the overall decision making process.

Another scenario involves a commanding officer on a battlefield who receives a message from a subordinate saying "Two enemy platoons seen crossing the river bed. Should we strike first?" If the general cannot readily converse with the subordinate, then he must make a critical decision based on the fuzzy information he received in the message. Assuming that the general had ready access to a portable field computer, he could run a simulation (with natural language input and output) that is based on fuzzy assumptions to determine possible outcomes. These outcomes can, then, effect his decision. Ideally, we would have data and statistics concerning the number of successful strikes against enemy platoons in specific geographic sites for a select number of troops on each side. Unfortunately, such volumes of data are not often available. In light of this lack of available information, fuzzy methods allow one to directly encode uncertainty to construct a fuzzy simulation for real-time decision making; if one does not have a complete statistical map of domain knowledge then one must use whatever facilities that are available to make a decision. The method of fuzzy simulation provides these facilities.

It is natural to ask how probability theory is situated with regard to fuzzy simulation. While the theory of probability serves the need for modelling uncertainty to a great extent, probability is not sufficient when modeling all phenomena. For instance, it is important for simulationists to model variance without the constraint of normalization defined within probability; there are many instances where gathering data for statistical analysis is too costly in terms of either time or money.

We often need to incorporate uncertainty in a model just to understand the time-varying sensitivity of specific initial conditions upon system state and output. We use the term "uncertainty" in the widest possible sense, so as to include concepts of uncertainty in belief, perturbations of time varying data and error in physical measurement [Klir 1988]. Even though Monte Carlo methods (frequently used in computer simulation) can be used to accomplish studies in uncertainty, it is useful to "relax" the normalization requirement in probability theory to permit the specification of arbitrary intervals and regions of confidence. Some of this work has been done under the rubric of perturbation theory; fuzzy methods extend this approach to additionally permit levels of uncertainty. We have found that fuzzy set theory, in general, serves as a sound mathematical foundation for incorporating uncertainty and qualitative specification into simulation modeling.

Wenstop [1976] has performed research in the fuzzy simulation of verbal models — discrete time state variables are related using fuzzy relations. Nguyen [1989] has studied fuzzy methods in discrete event simulation. Fuzzy set theory, originated by Zadeh [1965], has a long history (more than two decades) and is a generalization of classical set theory. We feel that fuzzy set theory can provide a necessary formal infrastructure for research in qualitative simulation modeling and analysis [Fishwick and Luker 1990]. Qualitative simulation is normal computer simulation with special emphasis on modeling and analyzing invariance with respect to state/parameter space and model structures. Examples of invariance include identifying qualitative phase features when simulating nonlinear systems (parameters can be varied within ranges without qualitative feature difference), and the use of interval algebra (variable values can be varied without altering interval label values - for certain functional expansions). In the latter case, for instance, we see that the interval equation $C = A + [1,4]$ is invariant with regard to the set of equations where a real value is used: $C = A + \beta$, $\forall \beta \in [1,4]$. Ultimately, invariance in algebraic forms and in model structures is the result of a homomorphic relationship which provides the "lumping" effect. With simulation we are concerned, not specifically with the preservation of the addition relation (as in the previous examples), but with the preservation of the simulation relation which is more complex. By simulation relation, we refer to the relationship between input and output trajectory segments as defined by Zeigler [1976] and Wymore [1977].

Our approach to qualitative simulation is based firmly in system and simulation theory [Zeigler 1976] and uses models at the level of abstraction [Fishwick 1988] at which they are best defined. For instance, with continuous models, where detailed state variable relationships are known, we use differential equation based models to generate quantitative state variable information; however, qualitative inputs and outputs may be assigned using a lexicon containing fuzzy number mappings. These outputs may then be used to identify lumped, abstract models using finite state automata, for instance. Cellier [1987] has recently written about qualitative methods based on highly abstract state spaces. Artificial intelligence researchers such as Kuipers [1986] are investigating the use of lumped, qualitative models for the purposes of simulating human process reasoning.

It is natural to first ask: "What simulation model components can be made fuzzy?" The answer to this question is based on a study of the fuzzy set literature to see what types of structures can be made fuzzy. It is not surprising to find that most non-fuzzy mathematical structures can be made fuzzy by simply extending the appropriate definitions to encapsulate fuzzy, and not crisp, sets. Zadeh and others in fuzzy set theory have specified the *extension principle* [Dubois 1980] which permits the transfer of existing mathematical methods to incorporate fuzzy semantics. Here is a sample of how this relates to simulation. We can make fuzzy:

- A state variable value. This includes both initial conditions and values at a specific time.
- Parameter values. Time variant systems can use fuzzy functions for parameters.
- Inputs and outputs.
- Model structure.
- Algorithmic structure.

In the last item, we note that a simulation model is really just an algorithm at the lowest semantic level, and therefore methods in fuzzy algorithms can be utilized.

In this paper, we first cover basic principles of applying fuzzy set theory to simulation modeling and then we discuss two applications of fuzzy simulation: 1) fuzzy simulation with linguistic variables, and 2) identification of qualitative models from fuzzy simulation runs.

2. FUZZY SIMULATION

2.1 The Fuzzy Number Concept

Considering the wide variety of application of fuzzy set theory to simulation, we have designed and implemented procedures to deal with fuzzy valued variables. We term a first order *fuzzy number* to be one that equals a single real value. As we increase the order, we obtain the following:

- An interval of confidence. Ex: $X = [1,3]$ where X is an *interval fuzzy number* with $\mu(1) = 1$ and $\mu(3) = 1$.
- A triangular number. Ex: $X = [1,1.2,3]$ where X is a triangular fuzzy number with $\mu(1) = 0$, $\mu(1.2) = 1$, and $\mu(3) = 0$.
- A trapezoidal number. Ex: $X = [1,2.5,3,4]$ where X is a trapezoidal fuzzy number with $\mu(1) = 0, \mu(2.5) = 1, \mu(3) = 1$, and $\mu(4) = 0$.
- General discrete fuzzy number. $X = \sum_{i=1}^{n} \mu_i/x_i$.

A fuzzy number is defined as a fuzzy set that is both convex and normal. The simple types of fuzzy numbers are piece-wise linear and so they can usually be abbreviated using the interval notation just delineated. With generalized fuzzy numbers we must, though, write down each domain value and its corresponding confidence level. The general form of the discrete fuzzy number (as shown above) is:

$$\sum_{i=1}^{n} \mu_i/x_i$$

and the continuous fuzzy number has the associated denotation for a continuous domain:

$$\int_X \mu(x)/x$$

These two types of fuzzy numbers are similar in concept to the discrete probability mass function and continuous probability density function found in the theory of probability. Note that it is natural to assign lexical values to fuzzy numbers, so that we can assign *high* $= [200.0, 400.0]$, *low* $= [10.0, 200.0]$, and *hardly_anything* $= [0, 10.0]$ for a given application.

The three methods that we have devised for studies in fuzzy simulation are 1) Monte Carlo Method, 2) Uncorrelated Method, and 3) Correlated Method. These methods are defined more completely in a working paper [Fishwick 1990]; however, we will briefly overview them now. The Monte Carlo method has been widely used in simulation and physics to allow multiple simulations based on sampling from a probability distribution. The uncorrelated method assumes that all uncertainties (or errors) are completely uncorrelated. The correlated method assumes that all uncertainty with respect to simulation variables is correlated. Correlated uncertainty fits well with human thinking since people tend to "guess on the low/high side" for the duration of a process. For this reason, we chose to implement a fuzzy simulation capability based on defining uncertainty as being correlated.

3. IMPLEMENTING FUZZY SIMULATION

We now outline the tools necessary to experiment with fuzzy simulation concepts. *SimPack* is a library of C routines and programs for general purpose simulation. The routines and programs contain facilities for creating single purpose or hybrid programs based on the following methodologies:

- Discrete Event (Queuing and Communications Networks)
- Differential Equation (Block Models)
- Difference Equations (Pulse Processes)
- Combined Models (Delay Differential Equations,Discrete/ Continuous)
- State Models (Automata, Markov Models, Petri networks)

Within this library, we have developed a number of routines for dealing specifically with fuzzy simulation. The core routines are as follows:

- `fzvocab(vocabulary)`: specify a lexical and ordinal vocabulary for fuzzy numbers. Different vocabularies may be accessed during the same simulation.
- `nl2fz(key,fuzzy)`: given a lexical key, produce its corresponding fuzzy number.
- `fz2nl(key,fuzzy)`: given a fuzzy number, produce the lexical key that most closely (according to a distance metric) matches existing vocabulary numbers.
- `nl2ord(key,ordinal)`: given a lexical key produce its ordinal mapping. For instance, *very high* = 2, or *very small* = 0. Ordinal mappings are used as lumped fuzzy numbers or as range values for plotting purposes.
- `fzprintf(string,fuzzy)`: format and print a fuzzy number.
- `fznumb(fuzzy,value)`: produce a fuzzy number from a crisp number.
- `fzstore(fuzzy1,fuzzy2)`: fuzzyl $\leftarrow$ fuzzy2.
- `fzadd(fuzzy1,fuzzy2,sum)`: sum $\leftarrow$ fuzzyl + fuzzy2.
- `fzmult(fuzzy1,fuzzy2,prod)`: prod $\leftarrow$ fuzzyl * fuzzy2.
- `fzsub(fuzzy1,fuzzy2,diff)`: diff $\leftarrow$ fuzzyl − fuzzy2.
- `fzdiv(fuzzy1,fuzzy2,div)`: div $\leftarrow$ fuzzyl / fuzzy2.
- `fzinv(fuzzy,inv)`: inv $\leftarrow$ 1 / fuzzy.

The routine `fz2nl` requires that a metric be defined to correctly map a fuzzy number to the most appropriate lexical entry. Let the fuzzy number in question be f defined as a triangular number $[f(1), f(2), f(3)]$ and let the vocabulary file contain n lexical entries with corresponding fuzzy numbers $g_1, \ldots, g_n$. Then, we use the following metric:

$$\| f - g \| = \min_i \sum_{j=1}^{3} (| f(j) - g_i(j) |)$$

In terms of implementation details, we built a fuzzy function library on top of the existing SimPack software. The library is currently encoded in C; however, we have recently created a separate C++ class [Dewhurst 1989] for fuzzy numbers. C++ is an object oriented extension of the traditional C language. In this manner, it is possible to write C++ code with operator overloading that enables arithmetic expressions such as the following:

```
Xnew = X + delta*Xprime;
```

where these variables can be a mixture of fuzzy and real-valued numbers. For instance, variables `Xnew`, `X` and `Xprime` could be fuzzy numbers while `delta` is a real number. This type of class definition, therefore, provides the programmer with a transparency with respect to specific types of numbers.

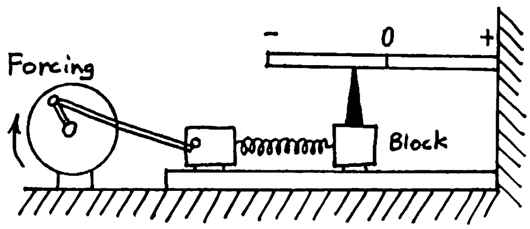

Figure 1. Sliding Block with Periodic Forcing

4. APPLICATIONS OF FUZZY SIMULATION

4.1 A Model: Forced Linear Vibrations

First, we choose a scenario to demonstrate the capabilities of fuzzy methods within simulation. Consider a second order non-homogeneous differential equation for modeling periodically forced vibrations. Fig. 1 shows the system to be modeled: a sliding block (of mass m and friction c) connected to a spring (with constant k). On the opposite end of the block there is a mechanism for enabling periodic forcing.

The equation is represented as a first order system as follows:

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -\tfrac{c}{m}x_2 - \tfrac{k}{m}x_1 + \tfrac{f}{m}cos(ax_3) \\ x_3' &= 1.0 \end{aligned}$$

We now consider two aspects of fuzzy simulation to this physical scenario: linguistic variables [Zadeh 1975a,b,c] and qualitative system identification. In our examples, we will let x_1 equal the triangular fuzzy value $[0.3, 0.5, 0.7]$.

4.2 Linguistic Variables and Values

The overall process of *simulation* involves the following key components:

- *Input*: the type of forcing function. For simple periodic functions (i.e. sinusoidal), we can choose to specify the amplitude as a key parameter to effect the input.
- *State*: the state of the system is a vector. For our system, there are 2 key state variables (x_1 and x_2) and so our state vector is $< x_1, x_2 >$. x_3 simply serves to formalize the system in terms of first order expressions (i.e. x_3 is the independent time variable).
- *Output*: the system output. We let x_1 (horizontal position, displacement) be the output.

- *Parameters*: parameters which are time-invariant for this system, and serve to control the system features (such as spring stiffness and mass).

DEFINITION: FUZZY SIMULATION IS THE PROCESS OF SIMULATING A SYSTEM BY MAKING FUZZY ONE OR MORE OF THE ABOVE SYSTEM COMPONENTS (INPUT, STATE, OUTPUT, PARAMETER).

In our definition of fuzzy simulation, we do not currently cover 1) simulation of a fuzzily defined system structure, or 2) simulation using fuzzy time. In the first instance, a fuzzy structure would be one where the skeletal model structure is not fixed for the duration of the simulation. In the second instance, time would not be totally ordered, but instead, it may be specified by an arbitrary partial ordering (or lattice). Table 1 displays the system components and linguistic variables associated with each component.

Table 1. Linguistic Variables

Component	Sub-Component	Variable
Input	a	amplitude
State	x_1	position
	x_2	velocity
Output	x_1	position
Parameters	c	surface
	k	spring
	m	mass

Table 2 displays the linguistic values associated with the linguistic variables. Linguistic values are encoded using the triangular fuzzy number previously defined. Each linguistic variable has several possible values. Each value is associated with a fuzzy number. Many of the fuzzy numbers also have associated with them a symbol and an ordinal mapping. The symbol is simply a short hand representation which is useful when graphing the system behavior in terms of linguistic values. The ordinal number assigns a total ordering to the fuzzy numbers for a given linguistic variable. This feature is useful if graphing a time series or phase plot of fuzzy values.

4.3 Running the Simulation

First, we define the ideal execution environment and then provide a sample simulation in our current environment. Ideally, the simulation would be driven by a natural language query such as: *If I give a fast push on the block from just right of center, how fast will the block be going after 15 seconds?*. The processing of such queries is a difficult task; however, some recent work has illustrated some success with limited vocabularies. For instance, we have previously constructed a natural language system [Beck and Fishwick 1989] for a biological systems control domain where a lexical functional grammar was used to break sentences into key components. Within a system such as this on our sample sentence, key phrases would be identified such as *fast push* and *right of center*. These phrases are then mapped to their fuzzy number equivalents and a fuzzy simulation proceeds. The simulation is executed for 15 time units and the state variable x_2 (indicating velocity) is translated first into "speed" and then into its lexical counterpart (using the distance metric). Possible answers may be either *The block is crawling* or *The block is moving slowly* where *crawling* and *slow* are determined according to the fuzzy to linguistic mapping.

Currently a mechanism is set up where the user enters linguistic values for each linguistic variable using a menu structure. The system presents questions (Q) and provides a menu selection, while the user responds (A) with instantiations of linguistic variable values.

```
Q: What is the value of position?
  (1) near right end
  (2) just right of middle
  (3) middle
  (4) just left of middle
  (5) near left end

A: 5
```

Table 2. Linguistic Values

Variable	Value	Symbol	Ordinal	Fuzzy Number
amplitude	*low*	⌒	0	[0.0,0.5,1.0]
amplitude	*medium*	~	1	[1.0,1.5,2.0]
amplitude	*high*	≈	2	[2.0,2.5,3.0]
position	*near right end*	++	2	[0.6,0.8,1.0]
position	*just right of middle*	+	1	[0.2,0.4,0.6]
position	*middle*	0	0	[-0.2,0.0,0.2]
position	*just left of middle*	-	-1	[-0.6,-0.4,-0.2]
position	*near left end*	—	-2	[-1.0,-0.8,-0.6]
speed	*crawling*	0	0	[-0.3,0.0,0.3]
speed	*slow*	+	1	[-1.0,0.0,1.0]
speed	*fast*	++	2	[-3.0,0.0,3.0]
surface	*smooth*	=	N/A	[0.0,0.5,1.0]
surface	*rough*	≅	N/A	[1.0,5.0,10.0]
spring	*relaxed*	≈	N/A	[0.0,0.5,1.0]
spring	*stiff*	=	N/A	[1.0,1.5,2.0]
mass	*light*	□	N/A	[0.0,2.0,4.0]
mass	*heavy*	□□	N/A	[4.0,6.0,10.0]

```
Q: What is the value of speed?
  (1) crawling
  (2) slow
  (3) fast

A: 3

...CONTINUES FOR OTHER LINGUISTIC VARIABLES...

Q: Which variable is to be output?
   (1) Time
   (2) Position
   (3) Speed

A: 3

Q: Do you want:
   (1) Time Series Output
   (2) Steady State Value

A: 2

...FUZZY SIMULATION RUNS...

RESULT: Speed is crawling.
```

We should note here that the choice of fuzzy number is of paramount importance. The choice of triangular numbers should be considered much like the choice of time interval size in digital signal analysis. Specifically, in signal analysis, the choice of an inappropriate sampling time can result in *aliasing* problems – the signal is sampled too infrequently and important signal information is lost. In the same way, fuzzy numbers whose endpoints are widely separated can provide simulation results where the output variable will always equal one lexical value even though the actual numerical system behavior (if one exists) suggests several qualities. Coarse linear interpolation, as demonstrated in the next section, will also result in aliasing problems. As with any good analysis, the analyst must insure that fuzzy number definitions agree as closely as possible with the "expert" that issued those definitions through the usual knowledge acquisition procedure (using expert systems technology) where the fuzzy knowledge is first elicited.

4.4 Qualitative System Identification

System Identification has a long history [Davies 1970; Eykhoff 1974] and there are many methods for identifying both structure and parameter settings through optimization and real system experimentation. The types of models usually identified are either continuous-time or discrete-time systems. To identify a discrete space model (such as a finite state automaton), one finds most of the literature on identification in other areas (especially in language translation and AI) under the heading of "induction." Gold [1967], for instance, was one of the first researchers to investigate the general problem of inducing grammars from examples. Induction can be based on both positive and negative examples; we will use only the positive example obtained from a continuous system as the basis for induced structure.

For our identification method, we will specify two sets of parameter settings from the forced vibration model defined in the last section. The two settings are:

1. Setting 1: $a = 1.0$, $c = 0.2$, $f = 0.0$, $k = 1.0$ and $m = 1.0$. (No Forcing, Damping).
2. Setting 2: $a = 3.7$, $c = 0.0$, $f = 10.0$, $k = 16.0$ and $m = 1.0$. (Periodic Forcing, No Damping).

There are several methods that we can use to map specific linguistic values such as "low," "high," "medium," or moderately low" to fuzzy numbers. We have just covered many of these possibilities in tables 1 and 2. Employing the mapping specified in the tables, we used the SimPack tools to produce symbolic/lexical plots. We used an interpolator (`lintp`) to linearly interpolate the ordinal output data. Linear interpolation consists of a filter where the symbolic output values representing sequential states of identical value are removed to yield a more compact plot. The values of x_1 are the ordinal values assigned in the mapping table. The simulation is made fuzzy by assigning a fuzzy initial condition to x_1. We let $x_1 = [0.3, 0.5, 0.7]$ before running the simulation. Using the distance metric we see that the corresponding lexical value for x_1 (position of block) is *just right of middle*. Therefore, the block is let go just right of the middle (i.e. the zero point).

Multiple, identically valued states are shown only where the sequential number of symbolic state values (in an output time series) exceeds a specified threshold. Figure 2 displays two time series solutions for x_1 with parameter settings 1 and 2 respectively. Figure 3 displays a "coarse" linearly interpolated view of the system with parameter setting 1, and shows the stages for identifying a finite state automaton model from a continuous system:

1. Assign linguistic values to fuzzy numbers by building tables 1 and 2.
2. Filter out identical, adjacent linguistic states (linear interpolation).
3. Induce (i.e. identify) a finite state automaton from filtered data.

Fig. 3 has 3 figures which reflect the output of each of these steps. The first figure is the time series for the linguistic state values. The second and third figures represent a finite state automaton (FSA) realized from the linguistic data. The FSA is realized in a straightforward manner by matching a linguistic value with a state. Other related methods for inducing FSA's from data are surveyed by Angluin and Smith [1983] and extensively discussed by Muggleton [1990] who lists several algorithms for this method of identification or induction. The second figure is an automaton where arcs are labelled with the number of traversals for that arc, and states are identified using thicker circles for states more frequently accessed. The third figure uses the linguistic value for each state and thicker arcs for more frequently traversed arcs. The use of thick lines and circles can be considered a visualization technique for allowing an analyst to see, not only the oscillation for the system, but also the tendency of the system to converge (i.e. the damping effect). We note that both plots are the result of continuous simulation. Fig. 3 displays qualitatively interesting characteristics in that it displays the changing state variable with respect to key linguistic concepts. How might such a qualitatively induced model (i.e. a finite state automaton) be used in practice? We see two potential uses:

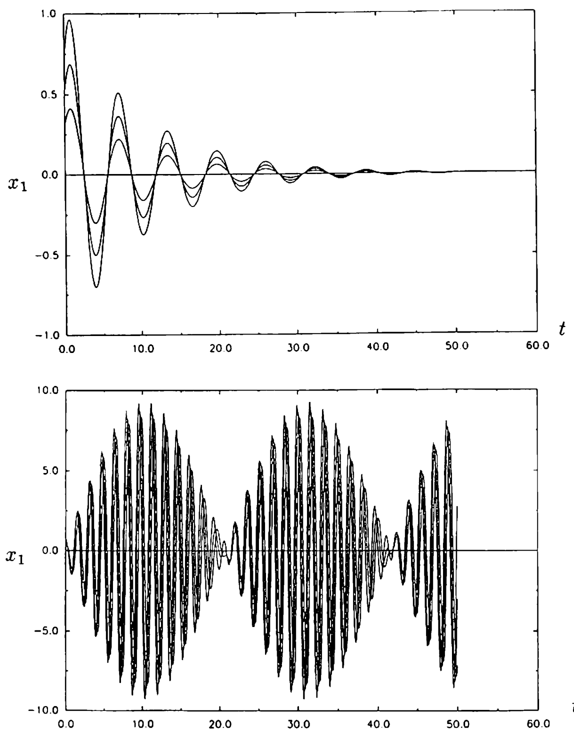

Figure 2. Quantitative Output of x_1 vs t (settings 1 and 2)

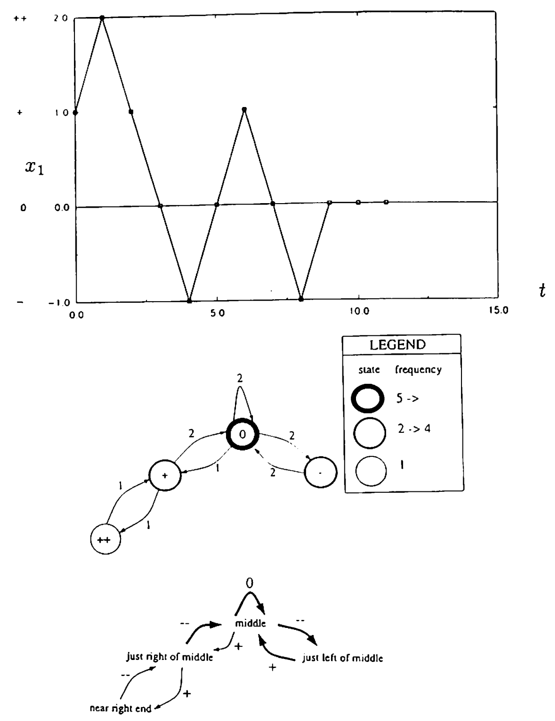

Figure 3. Qualitative Output of x_1 vs t (setting 1)

1. The method automatically produces a finite state machine from a base continuous model. This machine has lexically labelled nodes and serves as a qualitative model.
2. The method can be used to generate a qualitative hypothesis (i.e. the FSA) to compare against qualitative knowledge obtained directly from an expert. In this way, an expert displays his high level, intuitive knowledge about a system and then a fuzzy simulation is used to generate a qualitative hypothesis which is then compared against the expert's model.

5. CONCLUDING REMARKS

We have discussed the approach of using fuzzy set theory to create a formal way of viewing the qualitative simulation of models whose states, inputs, outputs and parameters are uncertain. Simulation was performed using detailed and accurate models, and we showed how input and output trajectories could be reflect linguistic (or qualitative) changes in a system. Uncertain variables are encoded using triangular fuzzy numbers and three distinct fuzzy simulation approaches (Monte Carlo, Correlated, Uncorrelated) are defined. The methods discussed are also valid for discrete event simulation; we have performed experiments on the fuzzy simulation of a single server queuing model even thought this result is not presented here. We augmented our existing C-based simulation toolkit *SimPack* to include the capabilities for modeling using fuzzy arithmetic and linguistic association, and we coded a C++ class definition for fuzzy number types.

Fuzzy simulation, as defined in this paper, provides a method for directly encoding uncertainty into a computer simulation by mapping fuzzy, linguistic values to simulation components. This kind of simulation, in turn, can be utilized to enhance the general decision making abilities of autonomous agents that have qualitative information about a system. Also, it will provide a better interface between autonomous, robotic agents and humans who think naturally using linguistic and fuzzy concepts.

There are other methods for studying both uncertainty in simulation model behavior and qualitative characteristics — in particular, perturbation and sensitivity analysis (for uncertainty) and bifurcation analysis (for global characteristic dynamical behavior). We do not view fuzzy simulation as replacing other approaches but rather we see it as an additional tool that is especially useful when studying the *transient effects* of uncertainty in initial conditions and parameters. We have found, in particular, that graphical study and the linguistic mapping of triangular fuzzy state behavior over time provides a unique understanding of the system.

ACKNOWLEDGMENT

This research is sponsored by a grant from the National Science Foundation IRI-8909152.

REFERENCES

Angluin, D. and C.H. Smith (1983), "Inductive Inference: Theory and Methods," *Computing Surveys*, *15*,3, 237–269.

Beck, H.W. and P.A. Fishwick (1989), "Incorporating Natural Language Descriptions into Modeling and Simulation," *Simulation*, *52*, 3, 102–109.

Cellier, F.E. (1987), "Qualitative Simulation of Technical Systems using the General System Problem Solving Framework," *International Journal of General Systems*, *13*, 4, 333–344.

Davies, W.D.T. (1970), *System Identification for Self-Adaptive Control.* John Wiley and Sons, London, England.

Dubois, D. and H. Prade (1980), *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, New York, NY.

Dewhurst, S.C. and K.T. Stark (1989), *Programming in C++.* Prentice Hall, Englewood Cliffs, NJ.

Eykhoff, P. (1974), *System Identification: Parameter and State Estimation.* John Wiley and Sons, New York, NY.

Fishwick, P.A. (1988), "The Role of Process Abstraction in Simulation," *IEEE Transactions on Systems, Man and Cybernetics, 18*, 1, 18–39.

Fishwick, P.A. (1989), "Qualitative Methodology in Simulation Model Engineering," *Simulation, 52*, 3, 95–101.

Fishwick, P.A. (1990a), "Fuzzy Simulation: Specifying and Identifying Qualitative Models," Technical Report TR-90-15, Com puter Science Department, University of Florida, Gainesville, FL.

Fishwick, P.A. (1990b), "Invariance and Nominal Value Mapping as Key Themes for Qualitative Simulation," In *Qualitative Simulation Modeling and Analysis.* P. A. Fishwick and P. A. Luker, Eds. Springer Verlag, New York, NY. (forthcoming)

Fishwick, P.A. and P.A. Luker, Eds. (1990), *Qualitative Simulation Modeling and Analysis*, Springer Verlag, New York, NY.

Fishwick, P.A. and B.P. Zeigler (1990), "Qualitative Physics: Toward the Automation of Systems Problem Solving," In *AI, Simulation and Planning in High Autonomy Systems*, B. P. Zeigler and J. Rozenblit, Eds. IEEE Computer Society Press, Washington, DC. 118–134.

Gold, E.M. (1967), "Language Identification in the Limit," *Information and Control, 10*, 447–474.

Klir, G.J. and T.A. Folger (1988), *Fuzzy Sets, Uncertainty and Information.* Prentice Hall, Englewood Cliffs, NJ.

Klir, G.J. (1990), "Aspects of Uncertainty in Qualitative Modelling and Analysis," In *Qualitative Simulation Modeling and Analysis*, P. A. Fishwick and P. A. Luker, Eds. Springer Verlag, New York, NY.

Kuipers, B. (1986), "Qualitative Simulation," *Artificial Intelligence, 29*,3,289–338.

Muggleton, S. (1990), *Inductive Acquisition of Expert Knowledge.* Addison Wesley, New York, NY.

Nguyen, H. (1989), "Fuzzy Methods in Discrete Event Simulation," M.S. Thesis, Department of Computer Science, University of Florida, Gainesville, FL.

Wenstop, F. (1976), "Fuzzy Set Simulation Models in a Systems Dynamics Perspective," *Kybernetes, 6*, 209–218.

Wymore, A.W. (1977), *A Mathematical Theory of Systems Engineering: The Elements.* Krieger Publishing, Huntington, NY.

Zadeh, L.A. (1965), "Fuzzy Sets," *Information and Control, 8*,338–353.

Zadeh, L.A. (1975a), "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," *Information Sciences, 8*, 199–249.

Zadeh, L.A. (1975b), "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," *Information Sciences, 8*, 301–357.

Zadeh, L.A. (1975c), "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," *Information Sciences, 9*, 43–80.

Zeigler, B.P. (1976), *Theory of Modelling and Simulation.* John Wiley and Sons, New York, NY.

Zeigler, B.P. (1984), *Multi-Facetted Modelling and Discrete Event Simulation.* Academic Press, New York, NY.

Zeigler, B.P. (1989), "DEVS Representation of Dynamical Systems: Event-Based Intelligent Control," *Proceedings of the IEEE, 77*, 1, 72–80.

Zeigler, B.P. (1990a), "High Autonomy Systems: Concepts and Models," In *AI, Simulation and Planning in High Autonomy Systems*, B. P. Zeigler and J. Rozenblit, Eds. IEEE Computer Society Press, Washington, DC, 118-134.

Zeigler, B.P. (1990b), *Object-Oriented Simulation with Hierarchical, Modular Models.* Academic Press, New York, NY.