IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 18, 2021, accepted March 29, 2021, date of publication April 9, 2021, date of current version April 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072239

Fuzzy Logic Applied to System Monitors

NOEL KHAN[1], DAVID A. ELIZONDO[1], (Senior Member, IEEE), LIPIKA DEKA[1], (Member, IEEE), AND MIGUEL A. MOLINA-CABELLO[2,3]

[1]School of Computer Science and Informatics, De Montfort University, Leicester LE1 9B1L, U.K.
[2]Department of Computer Languages and Computer Science, University of Málaga, 29071 Málaga, Spain
[3]Instituto de Investigación Biomédica de Málaga (IBIMA), 29010 Málaga, Spain

Corresponding author: Miguel A. Molina-Cabello (miguelangel@lcc.uma.es)

This work was supported in part by the Ministry of Science, Innovation and Universities of Spain through the Project Automated Detection With Low-Cost Hardware of Unusual Activities in Video Sequences under Grant RTI2018-094645-B-I00, in part by the Autonomous Government of Andalusia (Spain) through the Project Detection of Anomalous Behavior Agents by Deep Learning in Low-Cost Video Surveillance Intelligent Systems under Grant UMA18-FEDERJA-084, in part by the European Regional Development Fund (ERDF), and in part by the Universidad de Málaga and the Instituto de Investigación Biomédica de Málaga (IBIMA).

ABSTRACT System monitors are applications used to monitor other systems (often mission critical) and take corrective actions upon a system failure. Rather than reactively take action after a failure, the potential of fuzzy logic to anticipate and proactively take corrective actions is explored here. Failures adversely affect a system's non-functional qualities (e.g., availability, reliability, and usability) and may result in a variety of losses such as data, productivity, or safety losses. The detection and prevention of failures necessarily improves a critical system's non-functional qualities and avoids losses. The paper is self-contained and reviews set and logic theory, fuzzy inference systems (FIS), explores parameterization, and tests the neighborhood of rule thresholds to evaluate the potential for anticipating failures. Results demonstrate detectable gradients in FIS state spaces and means fuzzy logic based system monitors can anticipate rule violations or system failures.

INDEX TERMS Autonomous systems, fault tolerance, fuzzy systems, monitoring, software reliability.

I. INTRODUCTION

With increasing system size, complexity, and levels of automation come compounded and disproportionate risks from system failures. Historically, disaster management was both passive and reactive, but the current trend leverages metrics and status information to proactively monitor and address issues before they become failures. A decade ago, ORACLE's flagship product was described as a self-monitoring, self-diagnostic, and self-tuning database [1] and Amazon's CloudWatch monitors performance across entire infrastructures and automates common administrative tasks [2]. More recently, NASA developed a safety-critical autonomy framework that not only supercedes control of a vehicle to avoid external hazards, but tracks its internal health and triggers contingency behaviors such as an emergency landing due to a critical sensor's malfunction or failure [3]. In [4], researchers fed sensor data into a condition monitoring system to evaluate whether scheduled maintenance should be expedited to avoid unsafe conditions or reduced efficacy. System monitors have also been proposed for on-line continuous monitoring of loads on turbine rotor blade of wind turbines [5] and monitoring the deformation of dam slopes [6]. These are examples of systems that utilize some form of self-health monitoring and management.

Meanwhile, the application of fuzzy logic has been considered in a varied wide range of research fields, such as the detection of untrusted nodes in smart grid networks [7], detection of anomalies in computer network segments [8], estimation of the illumination states of the pixels in video surveillance [9], guidance and Control of Marine Surface Vehicles and Underwater Vehicles [10], control of induction motor drives applications [11], or diseases diagnosis [12].

System monitors are part expert system and part controller in that they use metrics to assess a system's status or performance and automatically take corrective action to prevent losses. This paper is a case study of a production system monitor that for 7 years monitored the stability of a custom Geographic Information System (GIS). The custom GIS had a complex business layer comprised of.Net web-services, web-applications, system-level services, support libraries, and Java applets all glued together using an experimental message-bus technology called Cross Request Broker (XRB). There were significant stability issues in the message-bus,

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Elish.

which resulted in data loss, lost productivity, and required almost continuous human intervention.

After a few weeks of observation, the warning signs of a system failure began to materialize and an initial set of rules was captured inequalities with thresholds. Those rules were encoded in a system monitor based on Boolean-logic that checked to see if the XRB service was running, polled memory and CPU usage, and repeatedly sent simple requests designed to make a round trip through the business and data layers to test whether XRB was operational (''ping''). Based on those metrics, the monitoring system was tasked with injecting status information into the GIS system's login page to mitigate an additional load on the system and intervening if necessary by killing and restarting the XRB service. This system monitor averted about 70% of system failures, created the impression of stability, and prevented the loss of data and productivity.

Extending the system monitor based on Boolean logic to fuzzy logic appeared to be a natural extension and is explored in the current work. Fuzzy logic would decouple discrete values from rules (e.g., memory usage) and thereby make the rules platform independent. It also opened up the possibility of making such system monitors proactive if the state space for fuzzy logic rules exhibited gradients that could indicate a trend to or from a failure condition. This paper compares and contrasts the development and performance of a 4-input 2-output system monitor built using Boolean logic, type-1 fuzzy logic, and an adaptive neuro-fuzzy technique.

The challenge in applying fuzzy logic to this application of a system monitor regards the numerous design choices for specifying a FIS, so the first order of business was to evaluate the sensitivity of a FIS to those design choices. Thereafter, using the experimentally determined best FIS design, the question of anticipating failure is explored by evaluating input neighborhoods to simulate the system monitor's response to successive inputs.

The main contributions of this paper are:

1) Exploration and stratification of FIS design options by their impact on performance.
2) Results supporting the use of FIS for proactive system monitors provided the state space monotonically increases from global minima along any dimension.

II. PROPERTIES OF PROACTIVE SYSTEM MONITORS

Like other expert systems, system monitors use IF-THEN rules with thresholds, where the antecedent defines the conditions that must be satisfied before taking the action defined in the consequent. Conditions are Boolean expressions used to evaluate whether certain conditions have been exceeded, i.e., that a failure has occurred. What is desired is a greater opportunity to identify and take corrective action before a failure occurs. This could be addressed by adding more rules. However, the rulebase may become unmanageable or ungrounded from expert opinion. Adding rules also presumes that the correct threshold for every combination of input is knowable and noiseless.

For deeper insight into the qualities required for an expert system to anticipate failure, consider an expert system based on Boolean logic that has two independent Real inputs (x, y) and one dependent output (z) so that the state space can be visualized as a three dimensional landscape. If rules were defined using only the equality operator where only certain input combinations evoked a response, the landscape would be empty except where input combinations were defined. The equality operator would require an infinite number of rules to completely describe a state space and is therefore a poor choice except for combinatorial systems.

A significant improvement resulting in far fewer rules would be the use of inequality operators to define continuous parts the landscape. However, since Boolean logic is only defined on the set $\{0,1\}$, there would be step-wise changes in the landscape at rule boundaries [13] (e.g., akin to the sharp corners and vertical faces in QBERT's world). Disjoint or step-wise changes are undesirable because any sudden change in output fails to provide any indication of a developing problem.

Fuzzy logic is defined on the interval [0,1] and thereby offers intermediate truth values, which can support more gradual transitions in state space [13]. Sequences of those intermediate truth values can be interpreted as a trend toward (or away) from system failures. Graphically, fuzzy inequalities reduce the slope of step-wise transitions. The semantic interpretation of the slope of these transitions is vagueness and the width of the support is uncertainty.

III. THEORETICAL BACKGROUND

A. SET THEORY

In classical set theory, the universal set X uncategorically represents all elements within some domain, such as $\mathbb{R}$, and subsets of X are limited to elements related by common attributes [13]. Elements are discrete objects and an element's membership in subset A is either quantitatively defined in terms of a Boolean valued membership function $\mu : A \rightarrow \{0, 1\}$ such that $\mu(a) = 1$ if $a \in A$ and $\mu(a) = 0$ if $a \notin A$, or qualitatively defined by a characteristic function that specifies all the conditions an element must satisfy to be considered a member of a particular set. In fuzzy set theory, the domain of a universal set X is called a universe of discourse or a ''linguistic variable'' whose elements are fuzzy sets called ''linguistic values'' [14], which represent a set of discrete elements. Put differently, classical set theory assigns discrete values to subsets of X, whereas fuzzy set theory assigns discrete values to fuzzy sets and then assigns fuzzy sets to subsets of X, i.e., fuzzy sets represent a layer of indirection between discrete values and subsets of X. Membership in a fuzzy set is specified in terms of a continuous linear function $\mu : A \rightarrow [0, 1]$, such as triangular, Gaussian, or bell shaped curves [13]. The curves describing fuzzy sets within some universe of discourse customarily overlap near their extremities, which means a particular discrete value can have nonzero membership in a variety of fuzzy sets [15] or

simultaneously have nonzero membership in A and $\bar{A}$ [13]. Linguistic variables and values are typically assigned descriptive labels, e.g., Size = {Small, Medium, Large}.

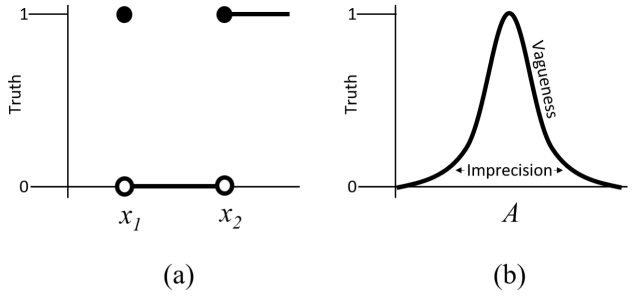

FIGURE 1. Fuzzy sets allow intermediate truth values whereas classical sets do not. Vagueness regards the transition between values. In Boolean membership functions, the change from one value to another is step-wise and has an infinite slope, which means membership in classical sets is not vague. With fuzzy membership functions, the change from one value to another typically has less than infinite slope. Vagueness is that subset of a fuzzy set's domain under the slopes of its membership function, which represent uncertainty about where the boundary between linguistic values actually are. Imprecision regards the size of an fuzzy set's footprint, which is infinitesimally small in a classical set, but covers some contiguous subset of the domain of *X* for a fuzzy set.

FIGURE 1 helps visualize the anatomical differences between Boolean and fuzzy truth values in terms of three properties: vagueness, imprecision, and truth value. Fuzzy sets allow intermediate truth values whereas classical sets do not. Vagueness regards the transition between values. In Boolean membership functions, the change from one value to another is step-wise and has an infinite slope, which means membership in classical sets is not vague. With fuzzy membership functions, the change from one value to another typically has less than infinite slope. Vagueness is that subset of a fuzzy set's domain under the slopes of its membership function, which represent uncertainty about where the boundary between linguistic values actually are.

Imprecision regards the size of an fuzzy set's footprint, which is infinitesimally small in a classical set, but covers some contiguous subset of the domain of X for a fuzzy set. For example, a rectangular membership function is imprecise but not vague. The outer boundaries of a trapezoidal membership function are vague, whereas all but a single point on a Gaussian membership function are vague. Motivated by the discussion in [16], we also note that as a membership function's rising and falling slopes approach infinity, vagueness (and possibly imprecision) disappears, but that does not necessarily reduce a fuzzy set into a classical set since the membership may top out at some intermediate value between 0 and 1. A fuzzy set reduces to a classical set only if all forms of uncertainty disappear [17]. The linear functions used to describe fuzzy sets contain parameters that can be tuned to manipulate their shape and position.

From a semantics point of view, fuzzy sets model membership uncertainty using intermediate truth values, imprecise knowledge, and vagueness with regard to the boundaries between different fuzzy sets [13], [18]. The geometrical interpretation is that membership uncertainty is responsible for gradual changes in state space, vagueness describes the gradient at rule boundaries, and imprecision circumscribes a neighborhood affected by interpolated values; all of which how affect gradual transitions in state space are.

Typical operations on sets include union, intersection, complement, and composition. Since sets are defined by membership functions, set operations can also be defined in terms of membership functions [14]. Generic symbols are used for some set operators due to the multiplicity of implementations that differ in their mathematical tractability [14]. Union can be implemented using any t-norm operator $\bigstar$ such as minimum, intersection can be implemented by any t-conorm operator $\oplus$ such as maximum, complement as the difference $1 - \mu(a)$, and composition in terms of a t-conorm of a t-norm.

B. LOGIC THEORY

Classical logic performs deductive reasoning using basic inference and replacement rules to traverse the transitive relationships between both premises and logical statements and either confirms a predetermined conclusion or fails to do so, which is either a proof by contradiction or means the conclusion is not entailed by the available premises [19]. Premises are facts with established truth value and logical statements are premises that are joined together using logical operators such as AND, OR, NOT, or implication. Replacement rules provide logically equivalent transformations of logical statements, whereas inference rules provide templates for logically sound reasoning or arguments [19].

Logical operators have set theoretical corollaries: the corollary for AND is intersection, the corollary for OR is union, and the corollary for NOT is the complement of a set [20]. Logical operators can be implemented exactly the same way as their counterparts. In classical logic, the implication $p \rightarrow q$ can be implemented as $MAX[1 - p, q]$, where p is the antecedent, q is the consequent, and $\mu_{p \rightarrow q}(p, q) \in \{0, 1\}$ [13]. Using Zadeh's extension principle [20] that induces a relationship between nonfuzzy and fuzzy variables by analogy [14], the fuzzy implication $A \rightarrow B$ can be implemented as $MAX[1 - \mu_A(x), \mu_B(y)]$, where $\mu_{A \rightarrow B}(x, y) A \rightarrow B \in [0, 1]$ [20], [21]. Fuzzy logic can perform deductive and inductive reasoning where conclusions are not absolutely supported by the premises [13] and relies on a single inference rule.

In classical logic, the implication $p \rightarrow q$ asserts the truth of q contingent on the truth of p, but asserts nothing about p. The inference rule modus ponens combines the assertion p and the implication $p \rightarrow q$ in order to assert the conclusion q, which is rendered as the logical statement $(p \wedge (p \rightarrow q)) \rightarrow q$, and this combination of logical statements is valid only when each of the variables p and q reference the same objects, respectively. Using the extension principle, the fuzzy corollary to modus ponens is $(A \wedge (A \rightarrow B)) \rightarrow B$ [20], [21], but since variable references in fuzzy logic are also a matter of degree,

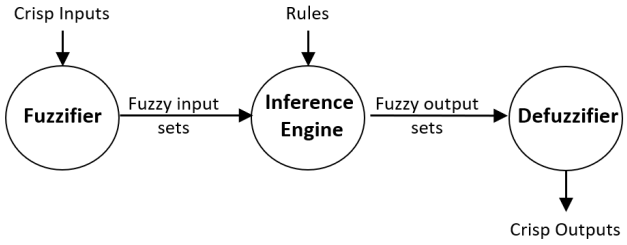

FIGURE 2. Reference design for fuzzy inference system. Adapted from [20].

references to the same object must have the same degree. Generalized modus ponens, $(A' \wedge (A \rightarrow B)) \rightarrow B'$, relaxes this constraint and allows approximate inferences anytime there is a nonzero degree of compatibility between like terms, which is computed as $w = A \cap A'$ [14]. In the case of a single antecedent, the degree of compatibility w truncates the membership function of B to yield the membership for B'. In the case of multiple antecedents, the intersection of all degrees of compatibility, which is called the degree of fulfillment or firing strength, is used to truncate B. Finally, in the case of multiple rules with multiple antecedents, the inference output is the union of all B' [14].

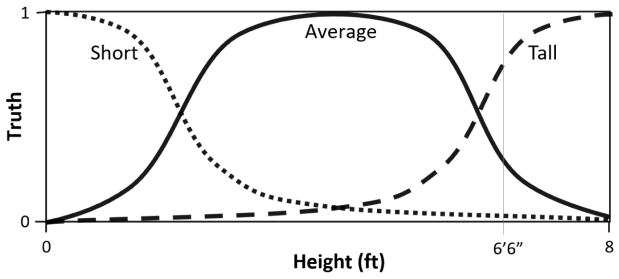

FIGURE 3. Linguistic variable Height with linguistic values Short (dotted), Average (solid), and Tall (dashed). The discrete input 6'6" would simultaneously belong to fuzzy set Short to degree 0.01, Average to degree 0.33, and Tall to degree 0.75.

IV. RULE BASED FUZZY LOGIC SYSTEMS

A. OPERATION

As shown on FIGURE 2, a fuzzy inference system (FIS) takes both a vector of discrete inputs and a rule matrix and produces a discrete output. Internally, a FIS is composed of a fuzzifier, inference engine, and defuzzifier, which will be described in this section. A fuzzy inference system does not assign discrete user inputs directly to variables, but rather assigns them to fuzzy sets that belong to linguistic variables. For example, if a linguistic variable Height with domain [0',8'] contained fuzzy sets Short, Average, and Tall, all defined as partially overlapping Gaussian curves, then the discrete input 6'6" would be assigned to fuzzy set Short to degree 0.01, Average to degree 0.33, and Tall to degree 0.75 as shown in FIGURE 3. This custom form of assignment is called fuzzification to differentiate it from the standard assignment operation.

User defined implication rules establish relationships between certain combinations of linguistic values for inputs and outputs. For example, the rule "If Height = Tall Then SlamDunker = Great" establishes a relationship between the particular fuzzy sets Tall and Great. At a high level, the FIS inference engine combines fuzzified inputs with such rules to determine the degree of truth of the rule's consequent. At a low level, a discrete input intersects the domain of fuzzy set Tall at $x = x'$ and the membership of Tall at x' is y. The truth of the consequent Great can not exceed the truth of y and is therefore truncated by y, which yields a modified version of the consequent fuzzy set. Here, y is called the firing level for this particular rule; each rule may have its own firing level depending on the interaction between user inputs and their corresponding fuzzy sets. In the case of rules with multiple antecedents, the firing level is defined as the $\bigstar$ of all y's obtained from the interaction of user inputs and their corresponding fuzzy sets. In the case of many rules, each rule output may be truncated to a different degree, but the inference output is defined as the $\oplus$ of all rule outputs and is also a fuzzy set.

The defuzzifier component converts the inference engine's fuzzy set into a representative discrete value. The exhaustive approach, called centroid defuzzification, computes a weighted average of membership grades across a discretized output domain. This defuzzifier is considered to be computationally expensive because it is predicated on computing the $\oplus$ of all rule outputs and the discretization of the output domain. Less computationally expensive defuzzifiers replace union with an arithmetic operation or use heuristics that abstract away detail. There are numerous alternatives, but the ones included in this study include: bisector of area, which identifies the point in the output domain that equally divides the total area of the output fuzzy set into two equal parts; and the mean of middle defuzzifier, which identifies the maximum value or, in the case of a plateau at that maximum, identifies the mid-point of that plateau.

B. DESIGN

Even with the same discrete inputs and rulebase, a FIS can produce different outputs by manipulating the definitions of linguistic variables, changing the membership functions or definitions of fuzzy sets, or by using different $\bigstar$ and $\oplus$ operators. The domain of linguistic variables can be changed as can the number of fuzzy sets within the linguistic variable. The curves used to define fuzzy sets can be changed from, for example, triangular to Gaussian curves, and each type of curve has its own embedded parameters that affect its shape and position. The choice of how fuzzy sets are defined is how uncertainty is modeled, but [13] notes that most applications are largely insensitive to variations in curvature and therefore simpler set models are recommended.

In this study, implementations of the $\bigstar$ operator are limited to minimum and product, and the $\bigstar$ operator is used to implement both the logical AND operator for rules with multiple antecedents and to implement implication. Likewise, implementations of the $\oplus$ operator are limited here to maximum, probabalistic-or, and sum, and the $\oplus$ operator is used for the logical OR operation and aggregation. Assuming the

domain of X and curve parameters were held constant, there are still at least 264 FIS variations based on the 24 distinct operator combinations and 11 standard membership function definitions.

Fuzzy inference systems can also be manipulated by selecting the type of rulebase, which has two possible forms. The antecedent in fuzzy implication rules always have the form "If x_1 is A_1 and $\ldots x_n$ is A_n." The consequent for Mamdani FIS has the form "y is $B_1 \ldots y$ is B_n," whereas in Sugeno FIS the consequent has the form "$z = f(x, y)$" [14]. Rule outputs are fuzzy sets that need to be defuzzified to yield a crisp outputs in Mamdani FIS, whereas rule outputs of Sugeno FIS are already crisp numbers from which a weighted average is computed to represent the inference output, which thereby avoids the computational expense of defuzzification.

The remaining way a FIS can be manipulated is by altering the content of the rulebase in terms of its size and individual rule definitions. According to [22], a major stumbling block for expert systems in the past was the requirement that all inputs be provided before an inference could take place, but fuzzy logic systems can produce output with partial information because missing inputs are interpolated. Not having to specify all inputs affords the ability to encode generalizations in fuzzy rules, which is a form of rule abstraction. Furthermore, fuzzification provides a 1:N mapping between discrete inputs and linguistic values, which allows rules to be defined qualitatively rather than be hard-coded with discrete thresholds and this too is a form of abstraction. These forms of rule abstraction give fuzzy logic systems a powerful advantage over classical systems because a manageable set of rules can yield a smooth and continuous control surface, whereas classical systems would require a significantly larger rulebase and still only have step-wise surfaces.

A practical limit exists on rulebase size because of conflicting expert opinions or because the aggregate of expert experience seems to become increasingly diffused or blurred, making it harder for experts to articulate rules. The challenge eventually becomes compensating for rule and threshold uncertainties. Historically, attempts to apply probability theory to rules created derivative problems related to joint probabilities and cascading adjustments in probabilities as the rulebase changed [18]. Fuzzy logic addresses membership uncertainty, but ignores input uncertainty pertaining to measurement, noise, and semantics. Avoiding the burdensome requirement of having to specify additional rules to handle such input uncertainties demands even greater levels of abstraction. Systems with higher order abstraction, such as type-2 fuzzy logic, will have even smoother surfaces resulting in greater system accuracy or precision while maintaining a manageable number of rules [21], but in this study we concern ourselves with type-1 fuzzy logic.

V. ADAPTIVE NEURO-FUZZY INFERENCE SYSTEMS

A. OPERATION

The above discussion makes it clear that although a FIS makes it easier to encode expert knowledge into expressive rules, there are nested parametrization problems that may require considerable effort to tune. For relatively noiseless applications with low semantic subjectivity [14], [23] where a sufficiently representative dataset is available, a FIS can be generated and tuned by a neural network. A prerequisite to understanding how a neural network can be used to generate a fuzzy inference system is to understand the relationship between the two types of systems.

A two-input first-order Sugeno fuzzy inference system (FIS) with its counterparts in a neural network can be found on figure 12.1 in [14]. An adaptive neuro-fuzzy inference system (ANFIS) is an adaptive five-layer feedforward network that is trained using a labeled dataset [14]. Fuzzy sets in the FIS are the layer-1 adaptive nodes in ANFIS architecture, the firing level for each rule corresponds to layer-2 weights, normalized weights are in layer-3, FIS rule outputs correspond to layer-4 adaptive output nodes, and the overall FIS output is the weighted average of rule outputs and their weights, which corresponds to the layer-5 output.

A representative set of input/output tuples can be used to generate a Sugeno FIS using an adaptive feedforward neural network, where a learning process tunes the parameters of each node's output function and parameters of FIS membership functions until the network converges on the expected outputs for each tuple. When membership parameters are fixed, the learning process propagates errors backwards through the network and, for each tuple p in the training dataset, computes the sum of squared differences E_p between the expected and actual values. The total error, $E = \Sigma E_p$, is minimized analytically by computing ordered derivatives with respect to each node and partial derivatives with respect to each parameter. The learning algorithm terminates when the error becomes zero, falls below a threshold, or terminates when a maximum number of iterations is reached. While the error is above a specified threshold, each node's parameters are updated using a steepest descent algorithm that adjusts the node's parameters to reduce that node's error and, in turn, the network's total error. When membership parameters are not fixed, forward passes through the network update output function parameters, while backward passes update membership function parameters [14].

Using MATLAB to generate a FIS from a representative set of input/output tuples is a four step process: datasets are loaded, a FIS is initialized using a training dataset, the FIS is trained using ANFIS, and the FIS is validated using a test or validation dataset [23]. Because MATLAB's ANFIS technique is limited to single output Sugeno FIS [23], MATLAB assumes all but the last element in each tuple represent inputs and that the last element represents the expected output. MATLAB's `genfis1` command produces a skeletal FIS and sets the range for each fuzzy input variable to the minimum and maximum values from each column of input data. The FIS output is defined as a linear first-order polynomial that has N+1 terms, where the first N terms are the products of inputs and coefficients and the last term is a constant. During initialization, the output fuzzy variable is initialized with a

set of vectors representing these coefficients and constants, all initialized to zero, and the number of vectors equals the number of distinct values in the training dataset. This insight into initialization allows the manipulation of input ranges by appending tuples into the dataset with the appropriate bounds. Furthermore, the resolution of the resulting system may be controlled by either binning or splitting the set of output values. The `genfis1` command also allows users to specify the shape and quantity of membership functions used in each fuzzy variable. Although [13] states that FIS are not sensitive to the shape of membership functions, it is worth noting that the slope of the rising and falling edges of the different shapes for membership functions squarely affects how abrupt or smooth the transition is from one fuzzy set to another. MATLAB's `anfis` command produces another FIS by tuning the parameters of all input and output member functions of the skeletal FIS using a back-propagation algorithm such that the trained FIS is able to reproduce the input/output relationships specified in the training dataset.

Upon producing a trained FIS, the final step is validation, which is easy to evaluate using the plots produced by MATLAB's Neuro-Fuzzy Design (NFD) applet. NFD tests how well results from the trained FIS fit against other sets of input/output tuples. Both the training and checking datasets and the trained FIS can be loaded into NFD from the MATLAB workspace and the ''Test FIS'' operations plot actual results against FIS results. Points in both the training and testing datasets should overlap well. A few outliers may not necessarily be a problem since the least squares algorithm makes compromises between conflicting tuples. A large number of outliers in the training data would suggest an insufficient number of epochs, whereas a large number of outliers in the testing dataset would suggest either an inadequate training dataset or over-taining.

B. DESIGN

Although ANFIS automatically tunes membership function parameters, generates rules, and defaults the selection of ★ and ⊕ operators for logical operations and aggregation, those operators can still be manipulated and there are additional ANFIS specific parameters regarding the number of membership functions, their curvature, how rules are partitioned, and a choice of optimization technique. If there was a large number of inputs (e.g., more than 10 or more), then the skeletal FIS can be initialized using clustering, which may reduce the dimensionality of the problem [14]. Otherwise, the ANFIS process partitions input space based on the intersection of M member functions for N universes of discourse (M^N space).

Optimization choices include back-propagation and a hybrid learning rule that combines back-propagation with least squares optimization. Back-propagation can be used to tune the membership function parameters, whereas the hybrid approach additionally tunes output function parameters. Optimization has parameters of its own, which include the number of iterations (epochs) and an early termination threshold.

VI. EXPERIMENTATION

The theoretical premise of this paper is that fuzzy logic can help monitoring systems anticipate failures, but because there are many design choices, the first order of business is to evaluate the sensitivity of a FIS to those design choices. Membership functions were specified graphically using MATLAB's Membership Function Editor so no effort was made to evaluate how best to specify embedded parameters absent a graphical design tool. Finally, using the experimentally determined best FIS design, the question of anticipating failure is explored by evaluating input neighborhoods to simulate the system monitor's response to successive inputs.

To evaluate FIS performance, the outputs from each fuzzy system was compared against known input/output combinations and used to compute a variance, which was reported in aggregate per test as percent error (hereinafter, Error$_{\%}$). The mean and standard deviation for a category represents the relative impact and range of impact from tuning that particular parameter. Attention will also be given to rule boundaries to assess fuzzy logic's ability to gradually transition between states.

To evaluate the ability to anticipate failure, experiments will evaluate a series of inputs chosen on either side of rule boundaries to see whether there is a monotonic trend towards or from failure. If consistently so, that would support the idea that the intermediate truth values in fuzzy expert systems can be exploited to anticipate failures.

As shown on FIGURE 2, rules are a critical part of fuzzy inference systems, but the current work does not evaluate the impact of different rules. Because the dataset and rules for the fuzzy expert system are common to both experiments, the next few sections will describe the fuzzy expert system, its rulebase, and dataset. That will be followed by additional detail on how each experiment was conducted.

A. RULEBASE

A flowchart that documented the operation of the Boolean XRB system monitor was translated into the set of rules listed in TABLE 1. Note that the Boolean rules do not specify all inputs because sometimes only a subset of inputs was sufficient to make an intervention decision. For example, if the XRB service was not running then there is no need to consider resource utilization. These rules were simply fuzzified to create the fuzzy rules shown in TABLE 2, which was certainly convenient. If rules were not already defined, then users would have to establish and collect metrics from which rules could be induced. The researchers in [4] found machine learning required less effort to employ than fuzzy-logic because the latter required data acquisition for each type of failure to support rule definition. Data driven approaches, however, may deter acquiring intimate knowledge and expert insight into a critical system's behavior. Finally, note that ANFIS automatically generates a rulebase.

TABLE 1. Boolean rules adapted from the XRB system monitor's documentation.

Rule	ServiceUp	MemoryUsage	CpuUsage	Ping	Status	Restart
1	Off	–	–	–	Offline	Yes
2	On	< 500	–	–	OK	No
3	On	> 500	< 20	–	Comatose	Yes
4	On	> 500	> 60	–	Unstable	Yes
5	On	> 500	$(20, 60)$	–	Busy	No
6	On	–	–	None	Comatose	Yes
7	On	–	–	SOS	Unstable	Yes
8	On	–	–	OK	OK	No

TABLE 2. Fuzzy rules adapted from Boolean rulebase.

Rule	ServiceUp	MemoryUsage	CpuUsage	Ping	Status	Restart
1	Off	–	–	–	Offline	Yes
2	On	Low	–	–	OK	No
3	On	High	Low	–	Comatose	Yes
4	On	High	High	–	Unstable	Yes
5	On	High	Med	–	Busy	No
6	On	–	–	None	Comatose	Yes
7	On	–	–	SOS	Unstable	Yes
8	On	–	–	OK	OK	No

B. ENGINEERED MEMBERSHIP FUNCTIONS

All FIS in this study input the following linguistic variables: ServiceUp, MemoryUsage, CpuUsage, and Ping. The output linguistic variables were Status and Restart. Each linguistic variable is comprised of a number of linguistic values. There are a variety of ways to model these values (e.g., analytic, geometric, list [13]) but are modeled graphically here.

The Service linguistic variable contained linguistic values No and Yes, which were modeled in the baseline FIS using trapezoidal functions with almost vertical slope at the domain midpoint since the corresponding input is noiseless and there is no uncertainty in that value.

The MemoryUsage linguistic variable contained linguistic values Low, Med, and High. In the baseline FIS, Low and High were modeled as trapezoidal functions because they could occupy either end of the domain completely and have a slope on the other side to represent the uncertainty of upper and lower boundaries, respectively. Med was modeled as a bell function since the flaring tails of the bell curve seem to capture the greater uncertainty in the meaning of "medium."

The CpuUsage linguistic variable contained linguistic values Low, Med, and High, which were modeled in the baseline FIS as bell curves. Since here the meaning of Med is a more imprecise concept, it occupied a larger part of the domain than the other values. Because the transition between Low and Med appeared to have greater subjectivity than the transition between Med and High, this subjectivity was encoded into the model by using a shallower slope between the former and steeper slope in the latter.

The Ping linguistic variable contained linguistic values None, SOS, and OK, which were modeled in the baseline FIS as trapezoidal curves with near infinite slope because the input was noiseless. Accordingly, an arbitrary domain was partitioned into equally sized segments and listed in order of decreasing severity.

The output linguistic variable Status contained linguistic values Offline, Comatose, Unstable, Busy, and OK. In the baseline FIS, these were modeled using triangular curves and also partitioned an arbitrary domain into equally sized segments in order of decreasing severity. There is significant vagueness in these linguistic values. Trapezoidal curves were not used since trapezoidal boundaries are typically steeper (i.e., imply less uncertainty) whereas curves with shallower sloped boundaries would have conflated those values.

Finally, the output linguistic variable Restart, which controls the XRB monitor's decision to kill and restart the XRB-service, had linguistic values No and Yes modeled as trapezoids that evenly divided an arbitrary domain. Since the boundary between these values has uncertainty, the transition between these two linguistic values should be sloped. These details are shown graphically in FIGURE 4.

C. DATASET

For the system under study, the system monitor's log file offers only a synoptic view of the supervised system's performance because it recorded health metrics only when there was an intervention. In hindsight, metrics during normal operation would also have been informative. TABLE 3 provides a summary of the metrics captured during interventions, which always had an abnormal Ping result. The system monitor also maintained a more detailed error log but that was repeatedly overwritten with information from the last intervention. That detail was intended to support debugging as well as inform additional rules.

Note that any time there was an intervention, the system monitor would immediately retest to ensure the intervention was successful. For half of all interventions, the XRB service only used 4MB RAM and negligible CPU time, which is exactly the resource consumption profile after initialization or a restart. Therefore, these low resource interventions are

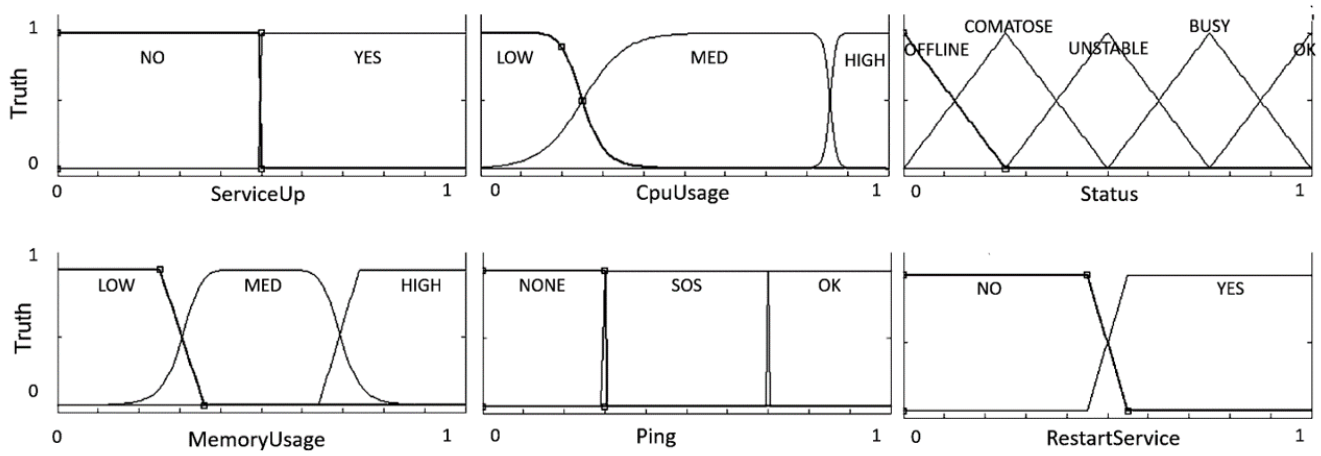

FIGURE 4. Linguistic variables and values of the baseline FIS. See section V.B for the rationale behind the specification of linguistic values as shown here.

TABLE 3. Summary of the system monitor's log file of interventions.

Memory Range	CPU Range	Instances
$[0, 10)$	$[0, 0]$	1053
$[10, 100)$	$[0, 0)$	147
$[100, 200)$	$[0, 0)$	82
$[200, 300)$	$[0, 0)$	89
$[300, 400)$	$[0, 0)$	61
$[400, 500)$	$[0, 0)$	70
$[500, 600)$	$[0, 70)$	311
$[600, 700)$	$[0, 70)$	160
$[700, 800)$	$[0, 80)$	91
$[800, 900)$	$[0, 80)$	46
$[900, 1000)$	$[0, 20)$	9
$[1000, 1100)$	$[0, 20)$	6
$[1100, 1200)$	$[10, 20)$	2
$[1200, 1300)$	$[20, 20)$	2

cascading failures that immediately followed an intervention. These low resource interventions are likely due to startup delays reconnecting the XRB service to business or data layers and if so, half of all recorded interventions could have been avoided by factoring in a longer delay to allow time to reinitialize.

The labeled dataset in TABLE 4 contains the memory, CPU usage, and XRB ping metrics extracted from the Boolean-based expert system's log file, which contained over 2,000 records of interventions. Test class 0 represents cases where the XRB service is offline, so the service needs to be restarted regardless of the other metrics. Class 1 are cases where the XRB service was detected but unresponsive within its timeout period. Again, regardless of other metrics, the service needs to be restarted. Class 2 is when the response from the XRB service was malformed in which case the system is considered unstable. Class 3 are nominal conditions across a variety of load conditions.

D. FIS DESIGN SENSITIVITY ANALYSIS

1) TEST CONFIGURATIONS

The first experimental question regards FIS design. TABLE 5 lists the 67 different tests performed across 8 test categories. Tests within each category typically varied one aspect of the FIS. All tests used the same rulebase listed in TABLE 2 and dataset listed in TABLE 4. Parameter categories were assumed to be independent to avoid a combinatorial number of tests with the exception of categories that paired defuzzifier and logical operators. Categorizing tests were intended to induce a rank ordering of categories that had the greatest impact on results.

Test categories for the manually engineered FIS regarded the choice of: defuzzifier, logic operators, and membership models. The defuzzifier category was comprised of 3 tests using the following defuzzifiers: centroid, bisector, and mean of middle (MOM). The two logic operator test categories each contained 24 tests that used different logic operators for AND, OR, implication, and aggregation. The membership model category of tests contained 3 tests that varied the uncertainty model using triangular, trapezoidal, of Gaussian curves.

The baseline Mamdani FIS configuration used MOM defuzzification, minimum to implement the logical operators AND and implication, and maximum to implement OR and aggregation. For brevity, TABLE 5 lists test configurations in terms of how they deviate from this baseline FIS configuration.

ANFIS specific test categories were: the number of linguistic values within a linguistic variable (hereinafter ''MF quantity''), membership model, optimization technique, and partitioning strategy. Because the automatically generated ANFIS rulesbase has exponential growth with the number of linguistic values per linguistic variable, ANFIS MF quantity tests were limited to 2, 3, and 4 membership functions. The ANFIS membership model category contained 6 tests that evaluated the effect of triangular, trapezoidal, and Gaussian curves with either constant or linear outputs. The ANFIS optimization category contained 2 tests that evaluated the affect of the back-propagation and hybrid optimization techniques. Finally, the ANFIS rule partitioning category contained 2 tests that evaluated the affect of training with the entire dataset versus a partitioned training and test datasets.

TABLE 4. Test and evaluation dataset. Each class represents a set of inputs that produce the same outputs, with the exception of Class 4 which has miscellaneous tests. The meaning of the status and restart values are defined by the Status and Restart linguistic variable in FIGURE 4, respectively.

Class	ServiceUp	Memory	Cpu	Ping	Status	Restart
0	0	100	Low	None	0	1
0	0	100	Low	SOS	0	1
0	0	100	Low	OK	0	1
0	0	100	Med	None	0	1
0	0	100	Med	SOS	0	1
0	0	100	Med	OK	0	1
0	0	100	High	None	0	1
0	0	100	High	SOS	0	1
0	0	100	High	OK	0	1
0	0	500	Low	None	0	1
0	0	500	Low	SOS	0	1
0	0	500	Low	OK	0	1
0	0	500	Med	None	0	1
0	0	500	Med	SOS	0	1
0	0	500	Med	OK	0	1
0	0	500	High	None	0	1
0	0	500	High	SOS	0	1
0	0	500	High	OK	0	1
0	0	900	Low	None	0	1
0	0	900	Low	SOS	0	1
0	0	900	Low	OK	0	1
0	0	900	Med	None	0	1
0	0	900	Med	SOS	0	1
0	0	900	Med	OK	0	1
0	0	900	High	None	0	1
0	0	900	High	SOS	0	1
0	0	900	High	OK	0	1
1	1	100	Low	None	0.25	1
1	1	100	Med	None	0.25	1
1	1	100	High	None	0.25	1
1	1	500	Low	None	0.25	1
1	1	500	Med	None	0.25	1
1	1	500	High	None	0.25	1
1	1	900	Low	None	0.25	1
1	1	900	Med	None	0.25	1
1	1	900	High	None	0.25	1
2	1	100	Low	SOS	0.50	1
2	1	100	Med	SOS	0.50	1
2	1	100	High	SOS	0.50	1
2	1	500	Low	SOS	0.50	1
2	1	500	Med	SOS	0.50	1
2	1	500	High	SOS	0.50	1
2	1	900	Low	SOS	0.50	1
2	1	900	Med	SOS	0.50	1
2	1	900	High	SOS	0.50	1
3	1	100	Low	OK	1	0
3	1	100	Med	OK	1	0
3	1	100	High	OK	1	0
3	1	500	Low	OK	1	0
4	1	900	Med	SOS	1	0
4	1	900	High	SOS	1	1
4	1	500	Med	OK	0.40	0
4	1	500	High	OK	0.75	0
4	1	900	Low	OK	0.65	1

All ANFIS tests trained their own FIS and used the same set of inputs, which are listed in TABLE 4.

2) EXPERIMENTAL PROCEDURE

Experiments were conducted in MATLAB because it provided both GUI and various capabilities for building FIS declaratively using FIS definition files. These definition files specify which operators to use, the quantity of membership functions, their parameters, and the rulebase. These FIS definition files simplified and facilitated a large number of tests because they were easy to define with scripting. The FIS configuration was also decoupled from the application that ran the tests.

To facilitate testing a large number of cases, the test application used `readfis` to read a FIS definition file and then used `evalfis` to test that configuration with a predefined set of input data. Because correct outputs were known in advance, the test function was able to summarize the performance of the configuration in terms of $\text{Error}_{\%}$.

Since the logic operator tests had combinatorial variations, a SQL database was used to quickly generate all possible operator combinations using a cross product on a table of operators. Cross products are generated anytime there is no join between SQL tables. These logic operator combinations were spooled to a file, which was then parsed by a script that instantiated a template FIS definition file to generate all 24 FIS definition files for each operator test category. Each of these definition files were given enumerated integer filenames to simplify processing FIS files iteratively. The MATLAB script that ran each test configuration also tracked and reported each configuration's $\text{Error}_{\%}$.

E. ANTICIPATING FAILURE

1) TEST CONFIGURATIONS

Evaluating whether fuzzy logic based expert systems can anticipate failures requires testing the neighborhood of values of TABLE 2 rules for a gradient. Because these tests are rule specific, all rule weights will be set to 0 except for the rule under test, which will be weighted 1. Setting rule weights to 0 mutes rule outputs, which impacts the aggregation of rule outputs and the defuzzification of that aggregate to produce a crisp value. There is no need to evaluate rules where the system status is normal or off, so only rules 3 through 7 will be evaluated.

Two neighborhoods of linguistic values will be tested; one near the maximum membership value and another where two linguistic values overlap. The former case is unambiguous and the latter purposely assesses the impact of ambiguity. Tests are also stratified by the number of linguistic variables with a multiplicity of values, i.e., in some tests, certain combinations of variables will be held constant in order to evaluate the neighborhood along specific dimensions. Other tests evaluate the impact of changes to multiple variables. The Case column on TABLE 7 captures these variations: tests near maximum membership have no ambiguity and are labeled 0; tests involving one variable's overlapping values are labeled 1; and tests involving overlapping values across multiple variables are labeled N.

The selection of values for these test cases is driven by plots like FIGURE 4. It is important to note that such plots are different for each FIS configuration so care will be taken to ensure the use of a pertinent set of plots for the highest performing manually engineered FIS.

TABLE 5. Listing of experimental configurations along with the performance metric $\text{Error}_{\%}$.

Test #	Category #	Category Name	Configuration	$\text{Error}_{\%}$
1	1	Defuzzifier	Centroid ("Baseline FIS")	20.53
2			MOM	17.98
3			Bisector	19.59
4	2	Logic Operators	Centroid; min,max,min,max	20.53
5			Centroid; min,max,min,probor	20.62
6			Centroid; min,max,min,sum	20.51
7			Centroid; min,max,prod,max	20.51
8			Centroid; min,max,prod,probor	20.61
9			Centroid; min,max,prod,sum	20.50
10			Centroid; min,probor,min,max	20.53
11			Centroid; min,probor,min,probor	20.62
12			Centroid; min,probor,min,sum	20.51
13			Centroid; min,probor,prod,max	20.51
14			Centroid; min,probor,prod,probor	20.61
15			Centroid; min,probor,prod,sum	20.50
16			Centroid; prod,max,min,max	20.53
17			Centroid; prod,max,min,probor	20.62
18			Centroid; prod,max,min,sum	20.51
19			Centroid; prod,max,prod,max	20.51
20			Centroid; prod,max,prod,probor	20.61
21			Centroid; prod,max,prod,sum	20.50
22			Centroid; prod,probor,min,max	20.53
23			Centroid; prod,probor,min,probor	20.62
24			Centroid; prod,probor,min,sum	20.51
25			Centroid; prod,probor,prod,max	20.51
26			Centroid; prod,probor,prod,probor	20.61
27			Centroid; prod,probor,prod,sum	20.50
28	3	Logic Operators	MOM; min,max,min,max	17.98
29			MOM; min,max,min,probor	17.98
30			MOM; min,max,min,sum	17.72
31			MOM; min,max,prod,max	17.98
32			MOM; min,max,prod,probor	17.98
33			MOM; min,max,prod,sum	17.98
34			MOM; min,probor,min,max	17.98
35			MOM; min,probor,min,probor	17.98
36			MOM; min,probor,min,sum	17.72
37			MOM; min,probor,prod,max	17.98
38			MOM; min,probor,prod,probor	17.98
39			MOM; min,probor,prod,sum	17.98
40			MOM; prod,max,min,max	17.98
41			MOM; prod,max,min,probor	17.98
42			MOM; prod,max,min,sum	17.72
43			MOM; prod,max,prod,max	17.98
44			MOM; prod,max,prod,probor	17.98
45			MOM; prod,max,prod,sum	17.98
46			MOM; prod,probor,min,max	17.98
47			MOM; prod,probor,min,probor	17.98
48			MOM; prod,probor,min,sum	17.72
49			MOM; prod,probor,prod,max	17.98
50			MOM; prod,probor,prod,probor	17.98
51			MOM; prod,probor,prod,sum	17.98
52	4	MF Model	Triangular	9.19
53			Trapezoidal	16.31
54			Gaussian	6.22
55	5	ANFIS MF Quantity	2 Triangular MF, Constant output	2.71
56			3 Triangular MF, Constant output	0.01
57			4 Triangular MF, Constant output	0.01
58	6	ANFIS MF & Output	2 Triangular MF, Constant output	2.71
59			2 Trapezoidal MF, Constant output	2.50
60			2 Guassian MF, Constant output	2.50
61			2 Triangular MF, Linear output	0.70
62			2 Trapezoidal MF, Linear output	0.52
63			2 Guassian MF, Linear output	0.53
64	7	ANFIS Optimization	Back-Propagation, 2 MF, Constant output	49.72
65			Hybrid Optimization, 2 MF, Constant output	2.71
66	8	ANFIS rule partition	No partitioning, train using entire dataset	2.50
67			Partitioned training and test datasets	21.00

Because the aforementioned tests used rule weights to isolate system responses for a single rule, an additional set of tests was needed to evaluate neighborhoods with all rules activated. TABLE 8 puts an asterisk on rules to indicate this difference from tests in TABLE 7. The convention used for the Case column in TABLE 8 also applies to TABLE 8. Here, only Cases 0 and N were evaluated since little value was seen in trying to isolate the affect of a single variable across all rules.

2) EXPERIMENTAL PROCEDURE

MATLAB's Rule Viewer is a GUI that accepts a vector of inputs, but manual entry is error prone. Instead, the linguistic values were entered into a spreadsheet from which a formula was used to generate calls to MATLAB's `evalfis` function along with inputs from each row. That list of calls was then copied into and executed in MATLAB.

For rule specific tests, all other rule weights had to be set to 0, which was done using MATLAB's Rule Editor. This was less prone to human error because the weights were set to 0 or 1 and the update process was always followed by a quick iteration through all rules to ensure only the target rule was weighted 1. However, an additional step was exporting the updated FIS back into the workspace, which was subject to an omission error but easily detectable and correctable as the results were unexpected.

Here the test metric was not Error$_{\%}$, but rather a qualitative assessment of whether there were small monotonic changes in output given small changes in the input. Again, in Boolean based expert systems, the system's output can change from one extreme to another on either side of a rule boundary but - given the state spaces depicted in FIGURE 5 - the expectation here is that the neighborhood of a test point will exhibit a gradient rather than a step-wise change.

VII. RESULTS

In order to establish a quantitative comparison of different FIS performances, the selected measure was the Error$_{\%}$ as was mentioned previously. This measure, also called Mean Absolute Error (MAE), is defined as follows:

$$MAE = \sum_{i=1}^{K} |(\tilde{h}_i - \hat{h}_i)| \tag{1}$$

where K is the size of the test and evaluation dataset from TABLE 4 (so that, $K = 54 \times 2 = 108$ (54 rows $\times$ 2 result variables)) while $\tilde{\mathbf{h}}, \hat{\mathbf{h}} \in \mathbb{R}^K$ are the ground truth and predicted results, respectively. The procedure to compute this measure is listed in Algorithm 1.

FIS performance in terms of Error$_{\%}$ values for each test category are detailed in TABLE 5 and summarized in TABLE 6. TABLE 5 lists the 67 different tests performed across 8 test categories and their performance in terms of Error$_{\%}$. Tests within each category typically varied one aspect of the FIS to induce a rank ordering of categories with the greatest impact on results. The mean Error$_{\%}$ can be used to rank the test category design options by their relative performance. Test categories can be re-interpreted as tuning categories for purposes of discussing the impact of different design options. The minimum and maximum Error$_{\%}$ could be interpreted to indicate the range of control those design options have over FIS performance.

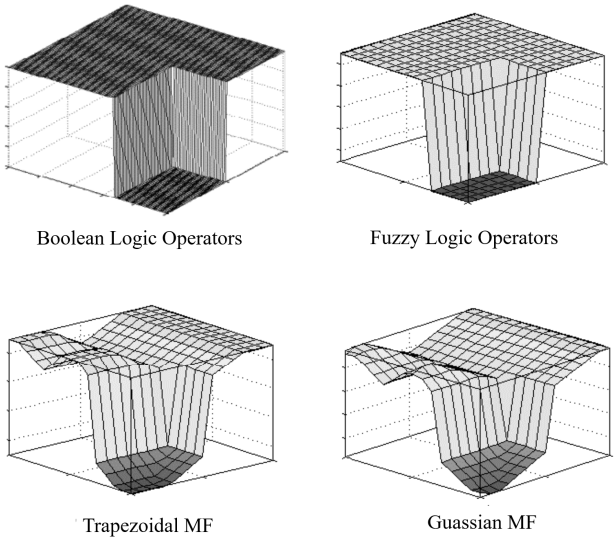

FIGURE 5. The top-left subfigure shows the landscape of a Boolean 2-valued expert system, whose rule boundaries have infinite slope. The top-right subfigure shows the same system using fuzzy logical operators, whose rule boundaries have less than infinite slope. The lower subfigures illustrate the affect of different membership models. Note that although both membership models exhibit an rising slope towards the rear-right, the trapezoidal model exhibits a curl, which is an additional detail not present in the other models. Z-values equal to 1 on these subfigures indicates an intervention. The rear-right edge of the subfigures represents the Offline system status, the top-left corner represents Comatose, and the adjacent hump on the top-left is Unstable. This level of detail on the specific type of failure is not available to the Boolean 2-valued expert system.

Algorithm 1 Procedure for Computing Error$_{\%}$ in TABLE 5

$n = 2$ //number of outputs (status and restart)
$\mathbf{fis_i} \leftarrow readfis(filename)$
for $\mathbf{x_j} = svc, ram, cpu, ping$ in TABLE 4,
  $\mathbf{y_j} = status, restart$ in TABLE 4 **do**
  $\mathbf{p\_result}(j, 1{:}n) \leftarrow evalfis(\mathbf{x_j})$ //predicted result
  $\mathbf{gt\_result}(j, 1{:}n) \leftarrow \mathbf{y_j}$ //ground truth
end for
for i in $rows(gt\_result)$,
  j in $columns(gt\_result)$ **do**
  $\mathbf{error}(i, j) = abs(\mathbf{p\_result}(i, j) - \mathbf{gt\_result}(i, j))$
end for
Error$_{\%}$ $= sum(\mathbf{error}(:, :))$

A. FIS DESIGN SENSITIVITY ANALYSIS

1) MANUALLY ENGINEERED FIS

From TABLE 6, the variety of logical operators and defuzzifier have the greatest impact on FIS performance, but appear to be false choices since those design options do not yield

TABLE 6. Summary $\text{Error}_{\%}$ by test categories. Lower is better. The statistical values were computed from the $\text{Error}_{\%}$ values for each test category listed in TABLE 5. The row in bold-face represents the system configuration with the best performance.

#	Category	Mean ± StdDev	Min	Max
1	Defuzzifier	19.59 ± 01.29	17.98	20.53
2	Operators (w/Centroid)	20.52 ± 00.05	20.5	20.62
3	Operators	17.98 ± 00.10	17.72	17.98
4	MF model	09.19 ± 05.19	6.22	16.31
5	**ANFIS MF Quantity**	**00.01 ± 01.56**	**0.01**	**2.71**
6	ANFIS MF Model	01.60 ± 01.09	0.52	2.71
7	ANFIS Optimization	26.22 ± 33.24	2.71	49.72
8	ANFIS Partitioning	11.75 ± 13.08	2.5	21

perceptible differences in performance. Reference [13] stated that most applications did not show great sensitivity to the shape of membership functions, which we confirm given the membership model had the lowest mean impact on performance. However, the choice of membership model had the largest range of control over performance; affecting $\text{Error}_{\%}$ by 6% to 16%. The smaller lower bound for the minimum $\text{Error}_{\%}$ means this set of design options offers the finest grained control over system performance. This observations are reasonable since membership models represent uncertainty.

FIGURE 5 depicts FIS landscapes distinguished by the operators used. Z-values equal to 1 on these subfigures indicates an intervention. The top-left subfigure shows the landscape of a Boolean 2-valued expert system, whose rule boundaries have infinite slope. The top-right subfigure shows the same system using fuzzy logical operators, whose rule boundaries have less than infinite slope. The other subfigures illustrate the affect of different membership models. Note that although both membership models exhibit an rising slope towards the rear-right, the trapezoidal model exhibits a curl, which is an additional detail not present in the other models.

The triangular membership models appeared to have the smallest slopes at rule boundaries and therefore appeared to be better suited for advanced warnings, but results for test 52 and 54 on TABLE 5 show that Gaussian membership models outperformed triangular models. To understand why, MATLAB's Surface Viewer was used to compare different combinations of dimensions of these two FIS. Since the rulebase was the same, the expectation was that there would only be slight differences between the landscapes and this was largely confirmed. However, 2 sets of dimensions exhibited a significant difference. As shown on FIGURE 6, the CPU x Ping x Status as well as the Memory x Ping x Status landscapes were monotonic only on the Guassian membership model. The corresponding landscape for the triangular membership models appeared to saturate to extreme values near rule boundaries, though it is not clear as to why.

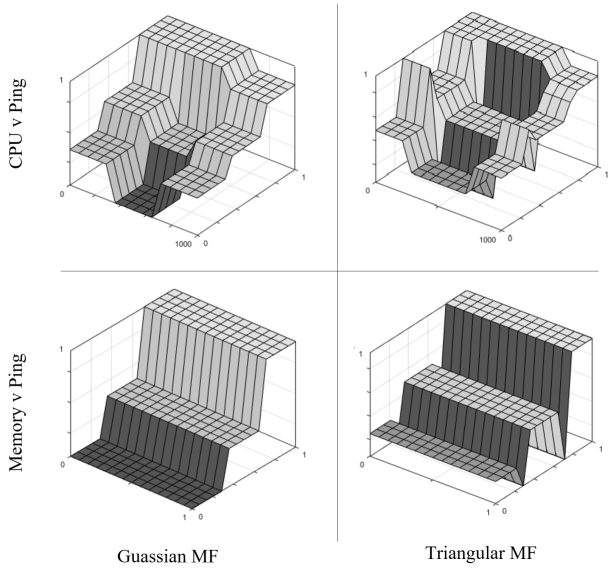

FIGURE 6. Different membership models of the same rulebase may exhibit differences in landscape monotonicity. The state spaces monotonically increased from the global minimum only for the Guassian membership model (left side). The corresponding state space for the triangular membership models (right side) appeared to saturate to extreme values near rule boundaries, though it is not clear why.

2) TRAINED FIS

From TABLE 6, the choice of optimizer had the greatest impact on performance. The large standard deviation suggests a high degree of control, but there were only two options (back-propagation, hybrid optimization) so the large standard deviation is due to stratification. Given the same convergence tolerance and epochs, hybrid optimization out performed back-propagation by a factor of 25.

Partitioning into training and test datasets had the next greatest impact on performance. Unpartitioned training outperformed likely due to over-fitting. Poorer results for the partitioned dataset suggests that performance is proportional to the amount of training, so larger and more representative the training datasets perform better.

The number of connections and rules in an ANFIS neural network grows exponentially with the number of linguistic values but with depreciating benefit. There was an appreciable improvement from 2 to 3 linguistic values, but a negligible improvement from 3 to 4 linguistic values. Therefore, although the quantity of linguistic values per variable has the greatest impact on the size of the ANFIS neural network, the number of linguistic values had a diminishing impact on performance.

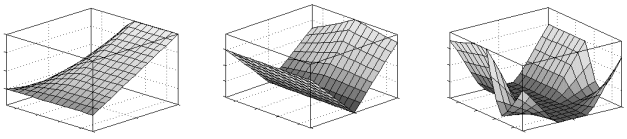

FIGURE 7. Notional illustration highlighting the increasing resolution of landscape features with 1, 2, and 3 linguistic values from left to right.

The graphical interpretation of an increasing number of linguistic values is shown in FIGURE 7. The complexity of

TABLE 7. Specification and results for isolated rule boundary-tests.

Test			Inputs				Expected		Actual		Assessment			
#	Rule	Case	Service	Memory	CPU	Ping	Status	Restart	Status	Restart	Status	Restart	$\mathcal{N}$	Overall
1					0.05					0.925				
2	3	0	1	950	0.07	0	0.25	1	0.25	0.925	1	1	1	1
3					0.10					0.910				
4				700						0.750				
5	3	1	1	725	0.00	0	0.25	1	0.25	0.770	1	1	1	1
6				750						0.790				
7					0.23					0.820				
8	3	1	1	950	0.24	0	0.25	1	0.25	0.810	1	1	1	1
9					0.25					0.805				
10				700	0.23					0.750				
11	3	N	1	725	0.24	0	0.25	1	0.25	0.770	1	1	1	1
12				750	0.25					0.790				
13					0.90					0.825				
14	4	0	1	950	0.93	0	0.50	1	0.50	0.880	1	1	1	1
15					0.97					0.925				
16				700						0.750				
17	4	1	1	725	1.00	0	0.50	1	0.50	0.770	1	1	1	1
18				750						0.790				
19					0.84					0.730				
20	4	1	1	950	0.85	0	0.50	1	0.50	0.745	1	1	1	1
21					0.86					0.765				
22				700	0.84					0.730				
23	4	N	1	725	0.85	0	0.50	1	0.50	0.745	1	1	1	1
24				750	0.86					0.765				
25					0.55									
26	5	0	1	950	0.60	1	0.75	0	0.75	0.075	1	1	0	0
27					0.65									
28				700						0.245				
29	5	1	1	725	0.60	1	0.75	0	0.75	0.230	1	1	1	1
30				750						0.210				
31					0.23					0.330				
32	5	1	1	950	0.24	1	0.75	0	0.75	0.320	1	1	1	1
33					0.25					0.310				
34				700	0.23					0.330				
35	5	N	1	725	0.24	1	0.75	0	0.75	0.320	1	1	1	1
36				750	0.25					0.310				
37						0.05				0.925				
38	6	0	1	1000	0	0.10	0.25	1	0.25	0.915	1	1	1	1
39						0.15				0.870				
40						0.23				0.810				
41	6	1	1	1000	0	0.25	0.25	1	0.25	0.790	1	1	1	0
42						0.28				0.770				
43						0.45								
44	7	0	1	1000	1	0.50	0.50	1	0.50	0.925	1	1	0	0
45						0.55								
46						0.65				0.810				
47	7	1	1	1000	1	0.68	0.50	1	0.50	0.780	1	1	1	1
48						0.70				0.750				

the FIS landscape increases with the number of linguistic values. Despite their differences, landscapes involving fewer linguistic values appear to abstract away details seen on landscapes with more. The likely explanation by way of analogy would be that if ANFIS was a curve-fitting process, then the number of linguistic values would be the order of the curve-fitting polynomial.

Finally, note that although the ANFIS generated system was able to correctly detect failures with better accuracy than any manually designed FIS, it utilized far more rules. Therefore although a trained system may outperform a manually designed FIS, the excess number of rules may simply represent overfitting and otherwise not be understandable. However, an ANFIS expert system requires far less effort to engineer and yields far greater performance provided training data is available.

B. ANTICIPATING FAILURE

Test 54 on TABLE 5 shows that the baseline FIS modified to use only Gaussian membership models outperformed all other manually engineered configurations, so that was the FIS used for these tests. Results from the linguistic neighborhood near maximum membership values are shown in TABLE 7 and results where two linguistic values overlap are shown in TABLE 8.

TABLE 8. Specification and results for rule boundary tests with all other rules activated.

Test			Inputs				Expected		Actual		Assessment			
#	Rule	Case	Service	Memory	CPU	Ping	Status	Restart	Status	Restart	Status	Restart	$\mathcal{N}$	Overall
49				925	0.05	0.05				0.925				
50	3*	0	1	950	0.10	0.10	0.25	1	0.25	0.915	1	1	1	1
51				975	0.15	0.15				0.875				
52				700	0.20	0.05				0.925				
53	3*	N	1	725	0.25	0.10	0.25	1	0.25	0.915	1	1	1	1
54				750	0.30	0.15				0.870				
55				925	0.93	0.45								
56	4*	0	1	950	0.95	0.50	0.50	1	0.50	0.925	1	1	0	0
57				975	0.97	0.55								
58				700	0.93	0.05				0.925				
59	4*	N	1	725	0.95	0.10	0.50	1	0.25	0.915	0	1	1	0
60				750	0.97	0.15				0.870				
61				925	0.55	0.95								
62	5*	0	1	950	0.60	0.98	0.75	0	1	0.075	0	1	0	0
63				975	0.65	1.00								
64				700	0.33	0.95								
65	5*	N	1	725	0.35	0.98	0.75	0	1	0.075	0	1	0	0
66				750	0.37	1.00								
67				1000	0.00	0.05				0.925				
68	6*	0	1	950	0.05	0.10	0.25	1	0.25	0.925	1	1	1	1
69				925	0.10	0.15				0.910				
70				700	0.23	0.20				0.830				
71	6*	N	1	725	0.24	0.25	0.25	1	0.25	0.790	1	1	1	1
72				750	0.25	0.30				0.790				
73				1000	0.95	0.48								
74	7*	0	1	950	0.98	0.50	0.50	1	0.50	0.925	1	1	0	0
75				925	1.00	0.53								
76				700	0.89	0.65				0.810				
77	7*	N	1	725	0.91	0.68	0.50	1	0.50	0.780	1	1	0	0
78				750	0.93	0.70				0.790				

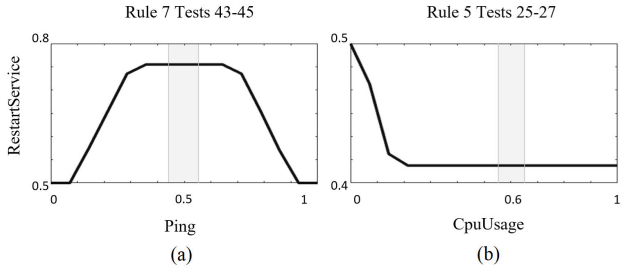

FIGURE 8. Tests that did not exhibit a gradient had neighborhoods with the same input values, i.e., landscape plateau.

All results from TABLE 7 exhibit a gradient for the output Restart with 2 exceptions. As why tests 25-27 and tests 43-45 did not exhibit a gradient, the suspicion was the test values were on a plateau. This was confirmed using MATLAB's Surface Viewer to plot the relationship between the test variables and Restart, as shown in FIGURE 8. This observation means that plots such as FIGURE 4 are insufficient to identify rule boundaries. In both of these cases, the test inputs were taken in the neighborhood of peaks from the middle of three linguistic values modeled as Gaussian curves, which is all the more unexpected that there are plateaus at those points. Upon further investigation, it appears insufficient to base test values on linguistic values alone since the operation of FIS resolves combinations of membership values to a firing level that truncates the output defined in the rule. Therefore, a better approach to understanding the affect inputs have on the outputs require a broader view of the system, which is provided by either MATLAB's Surface Viewer or Rule Viewer.

That largely also explains why the Status output variable was quiescent regardless of inputs across all tests. For tests isolating a particular rule, the rule itself specifies what the output linguistic value should be. Irrespective of how that linguistic value is modeled, the firing level only truncates the magnitude of that specified output, which is then defuzzified to the model's central value.

Yet even with all rules activated, a detectable gradient in the neighborhood of test points was found 60% of the time. This appears largely due to the defuzzification of aggregated rule outputs in order to produce crisp output values. That process essentially discretizes the output into one predefined linguistic value for each output. Therefore, the application of fuzzy logic to system monitors should avoid defuzzification. This reveals a gap in the current study. Namely, that all the manually engineered FIS were based on Mamdani FIS. Sugeno FIS yield a weighted average that may have avoided discretized defuzzification. ANFIS uses the Sugeno output model and that may explain why ANFIS significantly outperformed the manually engineered FIS.

Finally, both test cases for rule 5 on TABLE 8 had unexpected results. Specifically, these were the only tests that had

an incorrect Status output. The most probable explanation is that those test revealed an error in the rulebase. Rule 5 only specifies values for Service, Memory, and CPU variables in order to make a determination about the system's Status and decision to intervene. The thought was that limiting the number of linguistic variables in rules had greater generality, which is why Ping was undefined. The FIS correctly decided there was no need to intervene, but was unable to discriminate Busy and OK. Upon investigation, this was partly confirmed. Adding a value for Ping in rule 5 diminished the Status output from 1 to as low as 0.83, but was still higher than the expected 0.75. It appears that the concept of Busy and OK are being conflated, which biases the Status output towards OK, possibly due to aggregation. Translating the Busy linguistic away from OK resolved the ambiguity between Busy and OK, but conflated Busy with the Unstable. This suggests that semantically overlapping linguistic values should be avoided.

VIII. CONCLUSION

A. FIS DESIGN SENSITIVITY ANALYSIS

Although there are a fair number of design options involved when designing fuzzy inference systems, the KISS principle is advised. There was no significant variation in results based on different defuzzifiers or logical operators. The choice of membership model for linguistic values was the single best way to tune the performance of a FIS, but should be graphically checked for monoticity. Here, Guassian models had a monotonic landscape and outperformed both triangular and trapezoidal models.

ANFIS require the least amount of effort to build provided training data is available. As is often the case for supervised systems, the quality of the system is proportional to the size of training and test datasets. MATLAB's hybrid optimization significantly outperformed back-propagation and is therefore the recommended approach for MATLAB users. Increasing the number of linguistic values per linguistic variable will result in an exponentially complex neural network and has diminishing returns.

B. ANTICIPATING FAILURE

System monitors that use fuzzy logic can anticipate failures because the boundaries between rules have a gradient. The key to the application of Mamdani fuzzy logic to system monitors is to avoid defuzzifying. Sugeno FIS yield weighted values, which may have more gradual changes in output and may produce even better results, but needs to be experimentally confirmed in follow-on work. Although we have not evaluated type-2 fuzzy logic in the current work, [21] notes in the context of control systems that type-2 may offer even smoother state spaces, which may further enhance a system monitor's ability to proactively anticipate failure. Similar to the comparison of type-1 and type-2 fuzzy logic for a particular control system in [24], the extension of this work to type-2 fuzzy logic would similarly emphasize added value in a very controlled way.

Regardless of how a fuzzy system monitor is designed, the boundaries between rules need to be assessed more broadly than simply examining linguistic values at rule variables. This is because fuzzy inference systems combine variable values to find a firing level, which is used to truncate the rule's output variables and that interplay is not apparent by examining only plots of linguistic values. Another key observation was that linguistic values should be sufficiently stratified semantically to avoid conflating the meaning of adjacent values.

ACKNOWLEDGMENT

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Supercomputing and Bioinformatics (SCBI) Center of the University of Málaga.

REFERENCES

[1] Oracle Corporation. *Oracle Database 10g: The self-Managing Database*. Accessed: Jan. 10, 2021. [Online]. Available: https://www.oracle.com/technetwork/articles/sql/twp-manage-self-managing-database-128245.pdf

[2] Amazon Web Services. *Amazon Cloudwatch User Guide*. Accessed: Jan. 10, 2021. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/acw-ug.pdf

[3] M. Skoog. *Trustworthy Autonomy Development & Flight Demonstration; Multi-Monitor Run Time Assurance Research Update*. Accessed: Jan. 10, 2021. [Online]. Available: https://ntrs.nasa.gov/api/citations/20180002830/downloads/20180002830.pdf

[4] K. S. Pawar, S. Sondkar, S. Dattarajan, and N. Fernandes, "Comparative analysis of fuzzy logic and machine learning algorithm for predictive analysis of control valve," in *Proc. 5th Int. Conf. Comput., Commun., Control Automat. (ICCUBEA)*, Sep. 2019, pp. 1–6.

[5] K. Schroeder, W. Ecke, J. Apitz, E. Lembke, and G. Lenschow, "A fibre Bragg grating sensor system monitors operational load in a wind turbine rotor blade," *Meas. Sci. Technol.*, vol. 17, no. 5, pp. 1167–1172, May 2006.

[6] L. Tianjie, D. Liuqian, C. Liu, J. Jia, C. Zheng, X. Li, J. Wang, and C. Zhao, "System and method for monitoring deformation of dam slope," U.S. Patent 2021 0 048 294 A1, Feb. 18, 2021. [Online]. Available: https://patents.google.com/patent/US20210048294A1/en

[7] A. Alnasser and H. Sun, "A fuzzy logic trust model for secure routing in smart grid networks," *IEEE Access*, vol. 5, pp. 17896–17903, 2017.

[8] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network anomaly detection system using genetic algorithm and fuzzy logic," *Expert Syst. Appl.*, vol. 92, pp. 390–402, Feb. 2018.

[9] M. A. Molina-Cabello, E. López-Rubio, R. M Luque-Baena, E. Domínguez, and E. J. Palomo, "Foreground object detection for video surveillance by fuzzy logic based estimation of pixel illumination states," *Log. J. IGPL*, vol. 26, no. 6, pp. 593–604, Sep. 2018.

[10] X. Xiang, C. Yu, L. Lapierre, J. Zhang, and Q. Zhang, "Survey on fuzzy-logic-based guidance and control of marine surface vehicles and underwater vehicles," *Int. J. Fuzzy Syst.*, vol. 20, no. 2, pp. 572–586, Feb. 2018.

[11] N. Farah, M. H. N. Talib, N. S. M. Shah, Q. Abdullah, Z. Ibrahim, J. B. M. Lazi, and A. Jidin, "A novel self-tuning fuzzy logic controller based induction motor drive system: An experimental approach," *IEEE Access*, vol. 7, pp. 68172–68184, 2019.

[12] H. Ahmadi, M. Gholamzadeh, L. Shahmoradi, M. Nilashi, and P. Rashvand, "Diseases diagnosis using fuzzy logic methods: A systematic and meta-analysis review," *Comput. Methods Programs Biomed.*, vol. 161, pp. 145–172, Jul. 2018.

[13] G. Klir, U. Clair, and B. Yuan, *Fuzzy Set Theory: Foundations and Applications* (Bibliyografya Ve Indeks). Upper Saddle River, NJ, USA: Prentice-Hall, 1997. [Online]. Available: https://books.google.com/books?id=DNxQAAAAMAAJ

[14] J. Jang, C. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence* (MATLAB Curriculum Series). Upper Saddle River, NJ, USA: Prentice-Hall, 1997. [Online]. Available: https://books.google.com/books?id=vN5QAAAAMAAJ

[15] A. Hopgood, *Intelligent Systems for Engineers and Scientists*. Boca Raton, FL, USA: CRC Press, 2016. [Online]. Available: https://books.google.com/books?id=UxXYCwAAQBAJ
[16] G. Yen and D. Fogel, *Computational Intelligence: Principles and Practice*. Piscataway, NJ, USA: IEEE Computational Intelligence Society, 2006. [Online]. Available: https://books.google.com/books?id=A3YLPQAACAAJ and https://www.cs.utexas.edu/users/ai-lab/?miikkulainen:ci06
[17] J. M. Mendel, "Type-2 fuzzy sets: Some questions and answers," in *Proc. IEEE Connections*, Jan. 2003, pp. 10–13.
[18] N. Nilsson, *The Quest for Artificial Intelligence*. Cambridge, U.K.: Cambridge Univ. Press, 2009. [Online]. Available: https://books.google.com/books?id=nUJdAAAAQBAJ
[19] I. Copi and C. Cohen, *Introduction to Logic*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001. [Online]. Available: https://books.google.com/books?id=CbU7PgAACAAJ
[20] J. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001. [Online]. Available: https://books.google.com/books?id=j3NQAAAAMAAJ
[21] J. Mendel, H. Hagras, W. Tan, W. Melek, and H. Ying, *Introduction to Type-2 Fuzzy Logic Control: Theory and Applications* (IEEE Press Series on Computational Intelligence). Hoboken, NJ, USA: Wiley, 2014. [Online]. Available: https://books.google.com/books?id=C1HcAwAAQBAJ
[22] I. Bratko, *Prolog Programming for Artificial Intelligence* (International Computer Science Series). Reading, MA, USA: Addison-Wesley, 2011. [Online]. Available: https://books.google.com/books?id=-8BPPgAACAAJ
[23] The MathWorks. (2021). *Fuzzy Logic Toolbox: User'S Guide*. [Online]. Available: https://www.mathworks.com/help/pdf_doc/fuzzy/fuzzy_ug.pdf
[24] H. Bagua, M. Guemana, A. Hafaifa, and A. Chaibet, "Gas turbine monitoring using fuzzy control approaches: Comparison between fuzzy type 1 and 2," in *Proc. Int. Conf. Appl. Smart Syst. (ICASS)*, Nov. 2018, pp. 1–6.

NOEL KHAN is currently a Software Engineer working on autonomous systems in the U.S. aerospace industry. That pursuit was inspired during the 2007 DARPA Urban Challenge in his hometown and the works of Dr. Hod Lipson. His current research interests include robotic architecture that decouples the controller and body so that robots can discover their own capabilities, learn or relearn their body's topology, accommodate new forms and functions, applications of computational optimization, and natural language processing.

DAVID A. ELIZONDO (Senior Member, IEEE) received the B.A. degree in computer science from Knox College, Galesbourg, IL, USA, the M.S. degree in artificial intelligence from the Department of Artificial Intelligence and Cognitive Computing, University of Georgia, Athens, GA, USA, and the Ph.D. degree in computer science from the University of Strasbourg, France, in cooperation with the Swiss Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP). He is currently a Professor in intelligent transport systems with the Department of Computer Technology, De Montfort University, U.K.

LIPIKA DEKA (Member, IEEE) received the Ph.D. degree in computer science and engineering with a focus on concurrency control within file systems. She is currently a Computer Engineer, conducting research within the field of intelligent transport systems and connected autonomous cars. She is also working on the software engineering, software update, positioning accuracy, and path planning aspects of connected autonomous cars. She has interdisciplinary research experience of applying computer science algorithms (mainly AI techniques) in the fields of chemical engineering, healthcare infrastructure, and intelligent transport systems.

MIGUEL A. MOLINA-CABELLO received the M.Sc. and Ph.D. degrees in computer engineering from the University of Málaga, Spain, in 2015 and 2018, respectively. In 2015, he joined the Department of Computer Languages and Computer Science, University of Málaga, where he held a teaching and researching position. His current research activities are in collaboration with other universities. His research interests include visual surveillance, image/video processing, computer vision, neural networks, and pattern recognition.