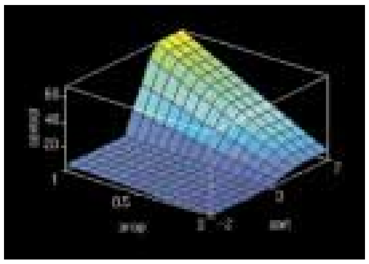
Applications of Fuzzy Logic in Control Design



ABSTRACT

Fuzzy logic can make control engineering easier for many types of tasks. It can also add control where it was previously impractical, as applications such as fuzzy-controlled washing machines have shown. However, fuzzy control need not be a dramatic departure from conventional control techniques such as proportional integral derivative (PID) feedback systems. As this technical brief demonstrates, fuzzy logic can be used to simplify the scheduling of two different controllers.

The MATLAB Environment

MATLAB provides a powerful computing environment for control system design, signal processing, modeling, analysis, and algorithm development. Its accurate numeric computation and built-in visualization make it easy to work with complex systems and data arrays. MATLAB Toolboxes offer specialized functions and easy-to-use graphical user interface tools that speed up the solution of application-specific problems. SIMULINK adds an intuitive block-diagram tool to the MATLAB environment for interactive simulation of nonlinear dynamic systems. With the Real-Time Workshop™, you can generate portable C code from SIMULINK block diagrams for rapid prototyping and implementation of real-time systems. The Fuzzy Logic Toolbox draws upon these capabilities to provide a powerful tool for fuzzy system design, analysis, and simulation. This technical brief describes the use of the Fuzzy Logic Toolbox to solve a typical control design problem.

The Fuzzy Logic Toolbox for use with MATLAB is a tool for solving problems with fuzzy logic. Fuzzy logic itself is a valuable engineering tool because it does a good job of trading off between significance and precision—something that humans have been doing for a very long time.

The Fuzzy Logic Toolbox lets engineers create and edit fuzzy inference systems either by hand, with interactive graphical tools or command-line functions, or by generating them automatically with clustering or adaptive neuro-fuzzy techniques. SIMULINK®, the simulation tool that runs alongside MATLAB, makes it easy to test your fuzzy system in a block diagram simulation environment. In addition, Real-Time Workshop™ can generate portable C code from the SIMULINK environment for use in real-time or non real-time applications.

In this document, we'll discuss some basic concepts behind fuzzy logic, and then we'll look at the implementation of fuzzy logic in a typical control problem.

Basic Concepts

Fuzzy Sets and Membership Functions

In fuzzy logic, the truth of any statement is a matter of degree. Any statement can be fuzzy. A membership function is the curve that defines how true a given statement is for a given input value. It defines how each point in the input space is mapped to a membership value (or degree of membership) from 0 and 1.

One of the most commonly used examples of a fuzzy set is the set of tall people. In this case the input space includes all potential heights, say from 3 feet to 9 feet, and the word "tall" would correspond to a curve that defines the degree to which any person is tall. If the set of tall people is given the well-defined (crisp) boundary of a classical (non-fuzzy) set, we might say that all people taller than 6 feet are officially considered tall. But such a distinction doesn't reflect our experience. Just as importantly, the example is not as practical or as accurate as it could be. It may make sense to consider an abstract concept such as the set of all real numbers greater than six, but when we want to talk about real people, it is unreasonable to call one person short and another one tall when the difference in height between them is only the width of a hair.

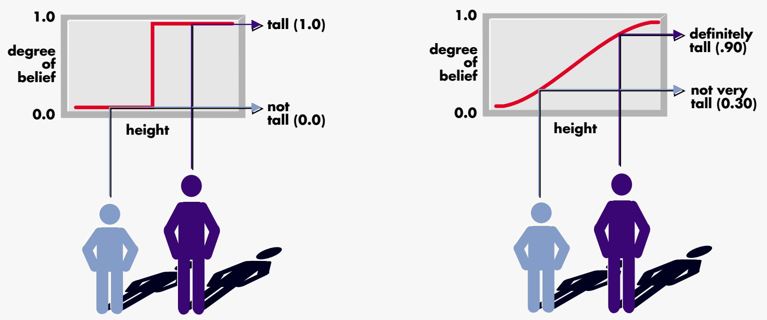


Figure 1. Left: sharp-edged (non-fuzzy). Right: smooth-edged (fuzzy) membership functions.

But if the kind of distinction shown in the left diagram of figure 1 is unworkable, then what is the correct way to define the set of tall people? The right diagram shows a smoothly varying curve that passes from not tall to tall. The output-axis indicates the degree of membership in the set of tall people, which is a value from 0 and 1. The curve is known as a membership function and the degree of membership it defines is often given the designation of μ. The curve defines the transition from not tall to tall. Both people are tall to some degree, but one is significantly taller than the other.

Subjective interpretations and appropriate units are built into fuzzy sets. If I say "She's tall," the membership function "tall" should already take into account whether I'm referring to a six-year-old girl or a grown woman. Similarly, the units are included in the curve. Certainly it makes no sense to say "Is she tall in inches or in meters?"

Application of Fuzzy Rules to Control

The point of fuzzy logic is to map an input space to an output space, and the primary mechanism for doing this is a list of "if-then" statements called rules. All rules are evaluated in parallel, so the order of the rules is unimportant. Before we can build a system that interprets rules, we have to define all the terms we plan on using and the adjectives that describe them. If we want to talk about how hot the water in a boiler is, we need to define the range over which the water's temperature can be expected to vary as well as what we mean by the word *hot*. The diagram in figure 2 is a road map for the fuzzy inference process. It shows the general description of a fuzzy inference process on the left, and a specific fuzzy system (a thermostat problem) on the right.

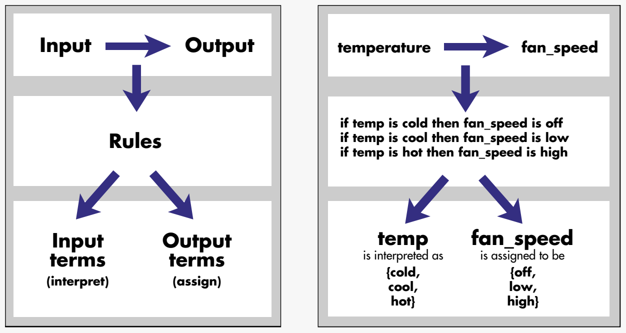

Input
Output
Rules
Input terms (interpret)
Output terms (assign)
temperature
fan_speed
if temp is cold then fan_speed is off
if temp is cool then fan_speed is low
if temp is hot then fan_speed is high
temp is interpreted as {cold, cool, hot}
fan_speed is assigned to be {off, low, high}

Figure 2. Fuzzy inference process (left), specific fuzzy system (right).

The idea behind fuzzy inference is to interpret the values in the input vector (like temperature) and, based on some set of rules, to assign values to the output vector (like fan speed). That's really all there is to it. For our thermostat problem, one of the fuzzy "if-then" rules is:

If temperature is hot then fan_speed is high

The "if" part of the rule *temperature is hot* is called the antecedent or premise, while the "then" part of the rule *fan_speed is high* is called the consequent or conclusion. Interpreting an "if-then" rule involves two distinct steps: evaluating the antecedent (which involves *fuzzifying* the input), and applying that result to the consequent (known as *implication*). In the case of classical binary logic, "if-then" rules don't present much difficulty. If the premise is true, then the conclusion is true. But if we relax the restrictions of binary logic and interpret the antecedent using fuzzy logic, how does this affect the conclusion? The answer is simple: if the antecedent is true to some degree, then the consequent is also true to that same degree. In other words

In binary logic:	p => q (p and q are either both true or both false)
In fuzzy logic:	0.5 p => 0.5 q (partially true antecedents imply the same degree of truth in the consequent)

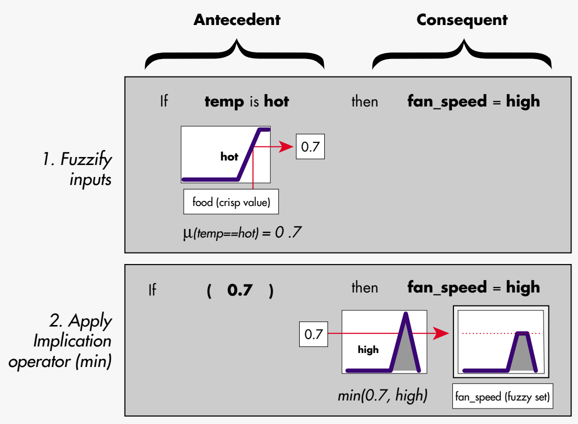


Figure 3. Implementation of a fuzzy rule.

In the example shown in figure 3, we see only one rule—but real-world fuzzy systems may have many. The outputs of each rule are combined and *defuzzified* to return the final output of the system. In fuzzy logic, multiple rules can be active for the same input value. This means that a few rules can interpolatively cover a wide operating space, just as a few poles can support a large tent. As a result, simple fuzzy systems can solve quite complex problems.

An Example Fuzzy Controller

Fuzzy logic control design is somewhat different from conventional control design methods in that it departs from standard analysis tools such as the Bode frequency response plot and the root locus diagram. In some cases, it may be appropriate to use an entirely fuzzy-based approach. But fuzzy logic can also be used in a hybrid approach with conventional control methods, making the most of both worlds. In this section, we examine how fuzzy logic can simplify gain scheduling between two different PID controllers.

The system we'll be looking at here is a simple one: a spring-mass-damper system from Dynamics 101 as illustrated in figure 4. While a basic PID controller will do a fine job of making it behave, fuzzy logic can provide a convenient way to meet stringent control objectives.

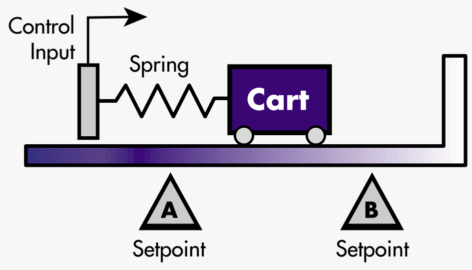


Figure 4. The cart system to be controlled.

In response to a square wave, we want to move the cart back and forth between points A and B. Notice that near point B there is a wall, a hard stop that we want to keep the cart away from. On the other hand, at point A we have considerably more leeway. Let's also assume we want to conserve control power and mechanical wear and tear by using looser, more relaxed control at point A. The design goal is relaxed control at point A, tight control (specifically, fast response with no overshoot) at point B. This situation is similar to the operation of a robot arm in an application where you want precise movement in one position and energy conservation elsewhere.

Because the plant is a simple one, both the precise control and the relaxed control can be implemented with a basic PID controller. But to meet both criteria we need some kind of gain scheduling to alternate between the two controllers, each of which has gain parameters tuned for its specific control objective. We can design a fuzzy controller to handle the gain scheduling for us. Let's start with the SIMULINK model of the system shown in figure 5.

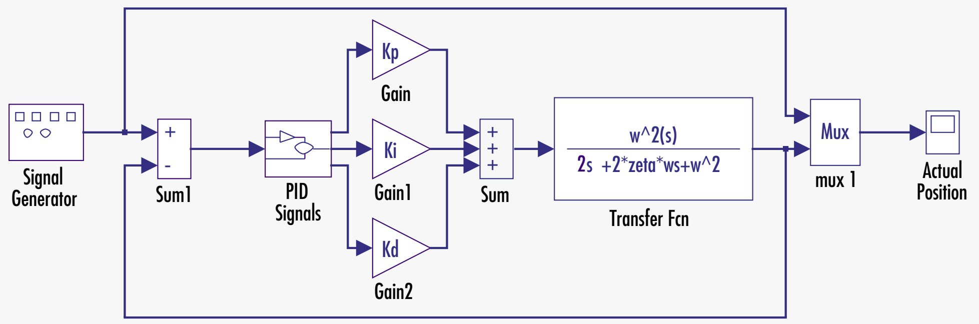


Figure 5. SIMULINK block diagram of cart system with PID controller in place.

The cart system is lightly damped. Its dynamics are described in the transfer function block as a function of the frequency domain variable *s*:

$$G(s) = \omega^2 / (s^2 + 2\zeta\omega s + \omega^2)$$

where the natural frequency $\omega = 1$ rad/sec and the damping is $\zeta = 0.1$.

Let's assume that we have already specified our gains for both the tightly controlled system and the loosely controlled one. There are any number of ways to choose these gains, and the ones we list below aren't necessarily optimal in any sense.

Tight control:	Kp = 60, Ki = 4, Kd = 14
Loose control:	Kp = 5, Ki = 1, Kd = 2

The main design constraint we want to guarantee with the tight control is zero overshoot near setpoint B. On the other hand, the main consideration for the loose control gains is minimizing the control effort (while providing a small degree of damping). Figure 6 shows the closed-loop step response for each set of gains.

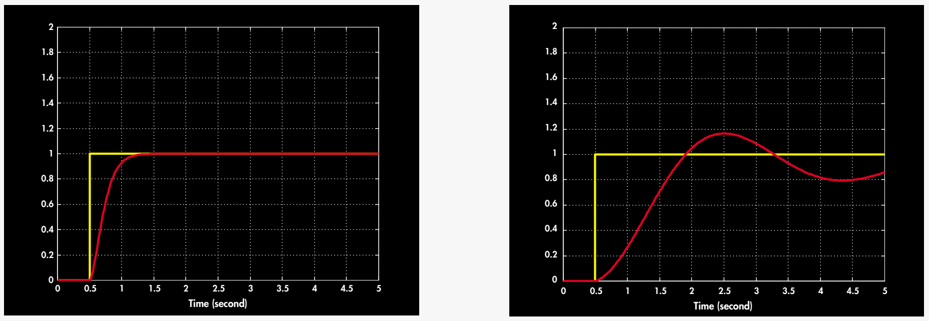


Figure 6. Left: closed-loop step response with tight control gains. Right: closed-loop step response with loose control gains.

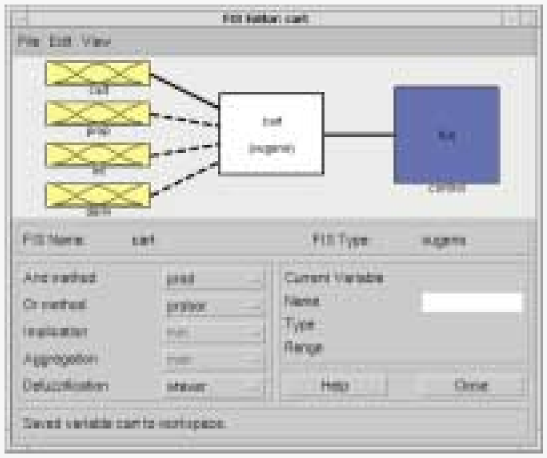
The point of this particular example is to show how we can use fuzzy logic to work hand-in-hand with conventional control. We do this by making use of what is known as Sugeno fuzzy inference system (named for fuzzy logic pioneer Michio Sugeno) to implement a blend of the two different PID controllers. First, we need to make sure we have a good understanding of how a Sugeno system calculates its outputs. In a Sugeno system, the output membership function is a linear function of the inputs. The fuzzy rules for a single input/single output system look like this

*If input is high, then output = q*input + r*

where *q* is a gain operating on the input and *r* is a constant. We need to build a Sugeno system with four inputs: cart position (so we can decide if we are close to point A or point B) and the P, I, and D signals to which we apply the appropriate gains Kp, Ki, and Kd. (The astute reader may notice that the P signal is the same thing as the cart position, but we will continue to refer to both signals for clarity.) Now we can build a rule set with exactly two rules:

1. If cart is near_A then control is loose	[so use the gains Kp = 5, Ki = 1, Kd = 2]
2. If cart is near_B then control is tight	[so use the gains Kp = 60, Ki = 4, Kd = 14]

The antecedents of these rules (e.g. "if cart is near_A") depend on the membership function for the terms "near_A" and "near_B". The consequents of these rules (e.g. "then control is loose") contain the three gains K_p, K_i, and K_d that we've calculated ahead of time. One output membership function implements all three gains at once. The system switches between two different controllers, so there are two output membership functions and two rules.

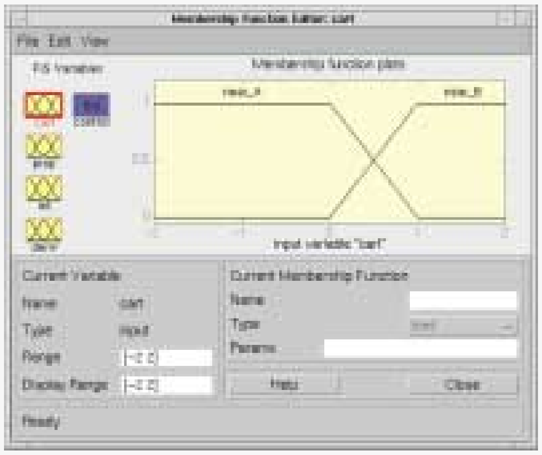


FIS Name: cart
FIS Type: sugeno
And method: prod
Or method: probor
Implication: min
Aggregation: max
Defuzzification: wtaver
Current Variable
Name
Type
Range
Saved variable cart to workspace.

Figure 7. The Fuzzy Inference System Editor in the Fuzzy Logic Toolbox.

The window shown in figure 7 is the Fuzzy Inference System (FIS) Editor, which we use to create our inputs and outputs for the fuzzy controller. We are building a four input/one output system, so we add inputs (cart, prop, int, deriv), and a single output (control action). We'll specify these with the Membership Function Editor.

The Membership Function Editor shown in figure 8 is where we define what we mean by the phrase "cart is near_A." Point A corresponds to the numerical value 0 and point B corresponds to the value 1. We have chosen to make a smooth ramp from one to the other. Notice that this means the statement "cart is near_A" is 100% true when position = 0; it is 50% true when



Current Variable

Name cart

Type input

Range [-2 2]

Display Range [-2 2]

Current Membership Function

Name

Type

Params

Figure 8. Fuzzy Logic Toolbox Membership Function Editor.

position = 0.5; and it is 0% true when position = 1. The converse is true of the statement "cart is near_B."

Once we have built our fuzzy controller using the graphical editors available in the Fuzzy Logic Toolbox, we save its specification in the MATLAB workspace as a memory-resident variable. Alternatively, you may use MAT-files to save it to disk. The fuzzy controller is now available to be used in the Fuzzy Controller block in a SIMULINK diagram.

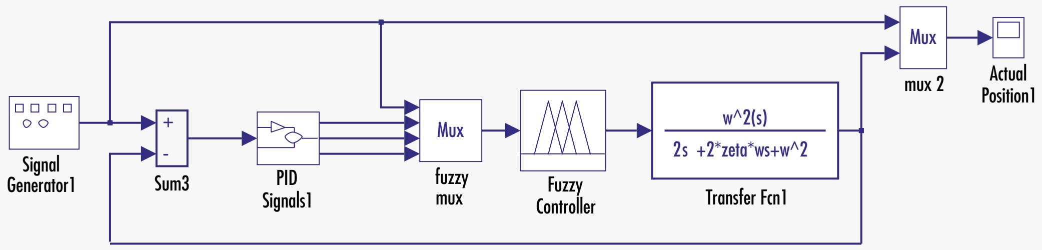


Figure 9. SIMULINK block diagram of cart system with fuzzy controller in place.

Figure 9 contains an updated version of our SIMULINK block diagram. Notice that we have replaced the three PID gains with a fuzzy controller block. Also, we are using the cart position as an extra input so we can blend between the two sets of gains depending on where the cart is.

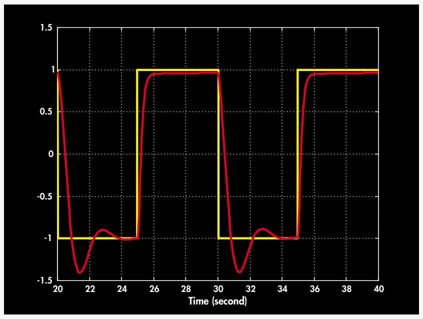
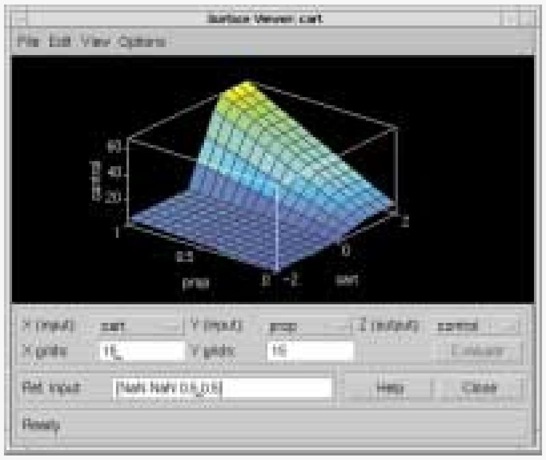


Figure 10. Closed-loop response with the gain-scheduling fuzzy controller.

The plot in figure 10 shows the simulation of our system as it responds to a square wave. Notice that, just as we expected, the control of the cart near point B is tighter and has no overshoot, while the control is much looser near point A. So the Sugeno fuzzy system has successfully implemented a convenient gain scheduler for us.



X (input): cart
Y (input): prop
Z (output): control
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 0.5 0.5]

Figure 11. Fuzzy Logic Toolbox output surface viewer.

In figure 11 we see the surface plot of the fuzzy controller. This is a plot of how the controller's output signal changes as a function of the cart's position and the proportional signal. Where the 3-D shape is mountainous, the control power required is higher and, as expected, corresponds to the region where the cart position is near point B. So we are looking at a map of the required control effort. This kind of visualization can be extremely valuable to a control designer, and it is just one example of the built-in tools available with the Fuzzy Logic Toolbox. Another one of the tools—the Fuzzy Inference Viewer—lets you examine in great detail every step of the fuzzy inference process. Together, the ensemble of graphical tools in the Fuzzy Logic Toolbox gives you enormous flexibility for working with fuzzy systems inside the MATLAB environment.

Additional Applications of The Fuzzy Logic Toolbox

The gain-scheduling problem in this technical brief demonstrates only one of the practical uses of fuzzy logic. Fuzzy systems are also useful in pattern recognition, signal processing, and system modeling applications. Prior to the Fuzzy Logic Toolbox, the principal barrier to broader use of fuzzy logic was the lack of a software environment that enabled users to easily explore fuzzy logic, compare it to known methods, and design practical systems that incorporate fuzzy technology.

The Fuzzy Logic Toolbox provides a comprehensive, intuitive software environment that streamlines the development of intelligent and adaptive products and processes. In addition to the Sugeno-style fuzzy system shown in this technical brief, the toolbox supports both conventional Mamdani fuzzy inferencing and state-of-the-art algorithms such as adaptive neuro-fuzzy learning and two methods of fuzzy clustering. An in-depth tutorial and demonstrations included with the software help you learn and apply fundamental concepts and advanced techniques.

The Fuzzy Logic Toolbox complements the advanced control, signal processing, and modeling techniques already available in MATLAB and the MATLAB Toolboxes. With the addition of SIMULINK and Real-Time Workshop, the Fuzzy Logic Toolbox offers a comprehensive, open system for simulation and code generation.

Additional Tools and Resources

Matlab Toolboxes

The MathWorks offers more than 20 application-specific toolboxes that build on the computational and graphical capabilities of MATLAB. All MATLAB Toolboxes are implemented in the high-level MATLAB language so that you can modify the source code for functions, or add your own. You can easily combine the techniques in all of the toolboxes to design custom solutions for your specific problems. The MATLAB toolboxes represent the work of some of the world's top researchers in their particular fields.

Products of interest to fuzzy logic technology users include:

Control System Toolbox
Automatic control system design and analysis tools, including linear system state-space modeling

Signal Processing Toolbox
Tools for spectral analysis, spectrum estimation, filtering, and specialized DSP operations

Neural Network Toolbox
Design and simulation tools for neural networks and adaptive systems

Optimization Toolbox
Optimization tools for general linear and nonlinear functions

System Identification Toolbox
Signal processing tools for parametric modeling and time-series analysis

Nonlinear Control Design Toolbox
Tools for design of optimal controllers for nonlinear systems

SIMULINK®
Dynamic system simulation in an intuitive block-diagram environment

Real-Time Workshop™
Generates portable, real-time C code from SIMULINK block diagrams

References and Additional Reading

The Fuzzy Logic Toolbox provides easy-to-use access to a variety of standard and advanced methods for designing and implementing fuzzy systems. For in-depth treatment of these topics, consult any of the following references:

Advanced Methods Used in the Fuzzy Logic Toolbox

Bezdek, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

Bonissone, P.P. et. al., "Industrial Applications of Fuzzy Logic at General Electric". *Proceedings of the IEEE*, March 1995, pp.450-465.

Jang, J.-S. R., "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm," *Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91)*, pp. 762-767, July 1991.

Jang, J.-S. R., "ANFIS: Adaptive-Network-based Fuzzy Inference Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, May 1993.

Jang, J.-S. R. and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, March 1995, pp. 378-406.

Jang, J.-S. R. and C.-T. Sun, "Neuro-Fuzzy and Soft Computing," 1995, (submitted for publication).

Jang, J.-S. R.and N. Gulley, "Gain scheduling based fuzzy controller design," *Proc. of the International Joint Conference of the North American Fuzzy Information Processing Society, Biannual Conference, the Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic*, San Antonio, Texas, Dec. 1994.

Sugeno, M., "Fuzzy measures and fuzzy integrals: a survey," (M.M. Gupta, G. N. Saridis, and B.R. Gaines, editors) *Fuzzy Automata and Decision Processes*, pp. 89-102, North-Holland, New York, 1977.

Sugeno, M., *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.

Background on Classical Fuzzy Logic Methods

Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

Wang, L.-X., *Adaptive fuzzy systems and control: design and stability analysis*, Prentice Hall, 1994.

Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.

Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

Zadeh, L.A., "The concept of a linguistic variable and its application to approximate reasoning, Parts 1, 2, and 3," *Information Sciences*, 1975, 8:199-249, 8:301-357, 9:43-80

For more information on the MATLAB Toolboxes, a list of papers in the MATLAB Technical Library, or a list of MATLAB based books, please return the attached reply card or contact your account representative at The MathWorks, Inc. at (508) 647-7000.