11-20-1991

Pendulum: Controlling an inverted pendulum using fuzzy logic

Scott Houchin J.

Pendulum: Controlling an inverted pendulum using fuzzy logic

by J. Scott Houchin

Thesis committe:

20 NOV 91

Dr. Pratapa V. Reddy (Chariman), Department of Computer Engineering

20 NOV 91

Dr. Peter G. Anderson, Department of Computer Science

Dr. Athimoottil Mathew, Department of Electrical Engineering

Pendulum: Controlling an inverted pendulum using fuzzy logic

I, J. Scott Houchin, hereby **deny** permission to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part.

J. Scott Houchin

11/20/91
Date

Preface

Computers occupy every aspect of society. Despite the cold error sounds of the supermarket cash register and the less than informative "Syntax error" messages output by the latest C compiler, computer scientists are continually trying to teach computers to be more human. The area of computerized control systems has seen significant advances, at least in the interface between system and designer.

In traditional control theory, the system engineer constructs a complex mathematical equation to relate the input of the system to the desired output. In addition to being non-intuitive, this method incurs a lengthy development time; the construction of the equations requires that the entire system be analyzed in great detail before the engineer can begin thinking about the control equations.

Instead of building on traditional control theory, we will start over, throwing out the notion that a control system evaluates a complex mathematical equation to determine the system output. Instead of mathematics, use common sense:

- Describe the state of the system in "human" terms:

  the temperature is very cold

  the pressure is too high

- Decide how the system should behave in each of these situations:

  if the temperature is very cold, turn up the heat

  if the pressure is too high, open the outlet vents a lot

- Teach the computer what we mean by terms such as "very cold," "too high," and "a lot."

- Implement the control engine using these rules and descriptions

This is fuzzy logic: "a way to program computers to mimic the imprecise way people think."

There are three main goals in our mind when developing a fuzzy system. First, by removing the complex mathematics and falling back on basic common sense, we have significantly decreased the development time for the system. Also, we remove the reliance on certain values in the environment, such as gravity and motor torque. The resulting system is thus much more tolerant to fluctuations in these values.

Lastly, by using common sense, we wish to increase the overall performance of the system. For example, fuzzy logic would allow us to design an elevator control unit that significantly decreased the average waiting time, or a train control system that provided a much smoother ride than possible by a human operator.

This thesis will explore fuzzy logic, specifically as it pertains to control systems. However, many of the concepts presented will directly carry over to other areas, such as decision making systems.

Pendulum

Pendulum, a fuzzy logic simulator created specifically for this thesis, will also be presented. Pendulum allows the user to configure a fuzzy logic engine to control an inverted pendulum. By allowing users to modify the system, they can experiment with fuzzy logic and learn first-hand how a system will behave with different rule sets, different fuzzy membership functions, and under different conditions.

Pendulum will be used to demonstrate:

- a methodology for designing fuzzy systems
- the effect of variations in the fuzzy system, such as curve shape
- the effect of a non-ideal environment on the fuzzy system.

The inverted pendulum

The system that will be developed throughout this entire thesis will be an inverted pendulum controller. The system consists of a stick mounted to a motor, and a processing unit of some type to control the torque provided by the motor. The goal is to balance the stick in the upward position with minimum overshoot and settling time. To do this, the system will cause the motor to apply torque to the stick, causing the stick to rotate. The motor will work with or against gravity and the momentum of the stick to stabilize it in the upward position (see figure 0 for several situations the system will encounter).

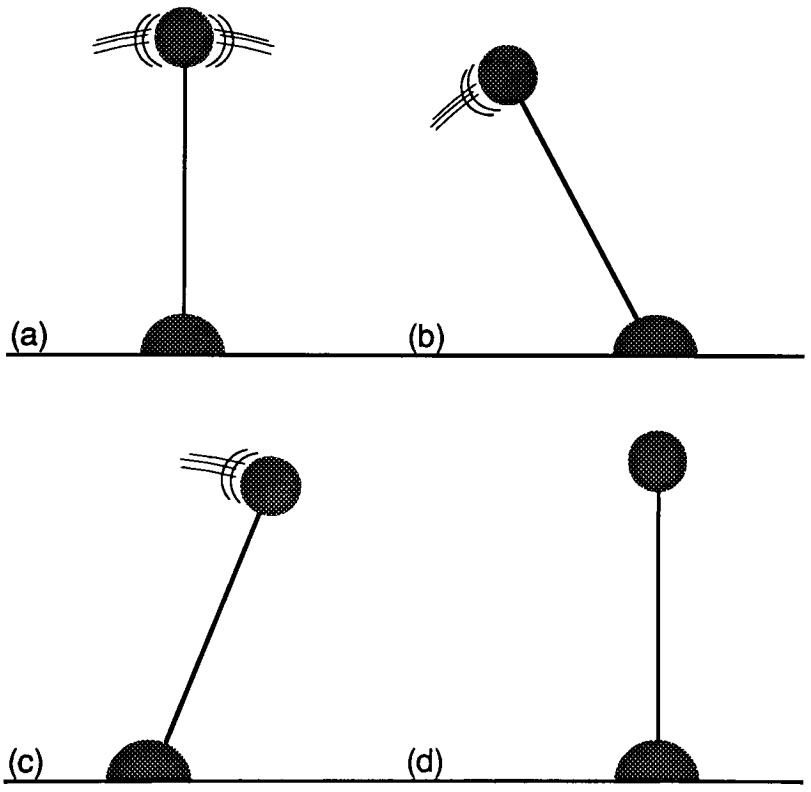

Figure 0—(a) the controller has brought the pendulum to the upright position, but it is oscillating.The controller must now try to bring the pendulum to a complete stop. (b) here the pendulum is off to the left, but it is moving toward the desired angle, so the controller must try to bring it to a stop as it reaches 90°. (c) In this case, the pendulum has passed the desired angle and is moving farther away. The controller must apply enough torque to the pendulum to stop the pendulum and start it moving in the other direction before it can try to balance in the upright position. (d) This is the ultimate goal of the controller: to stabilize the pendulum at 90°

Table of contents

Table of figures

One Fuzzy logic

Fuzzy logic is not an attempt to make computers that can "think." Although it has roots in artificial intelligence, pure fuzzy logic systems (not including self-organizing systems) try only to mimic the human thought process as specifically determined by the human designer. When designing a fuzzy system, two goals are in mind:

- to program the system to describe things the same way we do
- to instruct the system what action should be taken when the system is in a particular state

The later is nothing new to system designers; almost every deterministic computer program ever written has this goal (at least implicitly). It is the former that brings something new to control system theory. What is needed is a new way to describe things—a way that allows for doubt and uncertainty; fuzzy set theory provides this.

Once the system is described, decision-making logic that can deal with the "fuzziness" in the system is needed. In addition to a method (set of rules) for translating the fuzzy descriptions of the system into the desired actions, we also need a method to deal with conflicting decisions made by different rules in the system.

Fuzzy sets

The basis of fuzzy logic is the fuzzy set, an extension of the conventional set. In conventional set theory, a set (S) has as its members objects from the universe of discourse (U). Each object in U is either a member of S or is not a member of S; there is no in-between ground:

$$\neg\exists_x((x \in S) \wedge (x \notin S))$$

Consider, for example, the set *old*. Conventional set theory states that a particular person is either *old*, or that person is not *old*. But we as humans do not think in that manor. When we think of *old*, we think of something that people become gradually, not all at once at some magic age.

Instead of requiring this all or nothing membership rule, we would like to allow an object to be a partial member of a set. This fits our definition of *old* nicely. Different people are (members of the fuzzy set) *old* to different degrees; some people are more *old* (*old* to a higher degree) than others.

Let a fuzzy set F be characterized by a membership function $\mu_F(x)$, which represents the degree of membership of the object x in F. The membership function is defined in the range $0 \leq \mu_F(x) \leq 1$, $\mu_F(x) = 0$ meaning that x is not at all a member of F and $\mu_F(x) = 1$ meaning that x is definitely a member. The terms "measure of belief" or "measure of confidence" are also used for the function $\mu_F(x)$ because the degree of membership in the set is often used to represent the degree to which we believe or the confidence that an object could be a member of an equivalent conventional set.

For example, if $\mu_{old}(joe) = 0.1$ and $\mu_{old}(fred) = 0.7$ then we can say that the measure of confidence in fred being *old* is stronger than the measure of confidence in joe, or fred is *older* (more *old*) than joe.

Partial membership means possibility

This definition of a fuzzy set brings up the question of what we want partial membership in a fuzzy set to represent. The interpretation that will be used in this thesis is that given a conventional set S and an "equivalent (in meaning)" fuzzy set F:

$$\mu_F(x) = \text{possibility}(x \in S) = \pi_x(S)$$

Note that this is not the same as the probability that x is a member of S. For example, it is very possible that a new personal computer released today will not become obsolete for several years; however, it is not very probable. The possibility π_x (S) represents the ease that x may take S as a value. In our case, it represents the ease that x is a member of S.

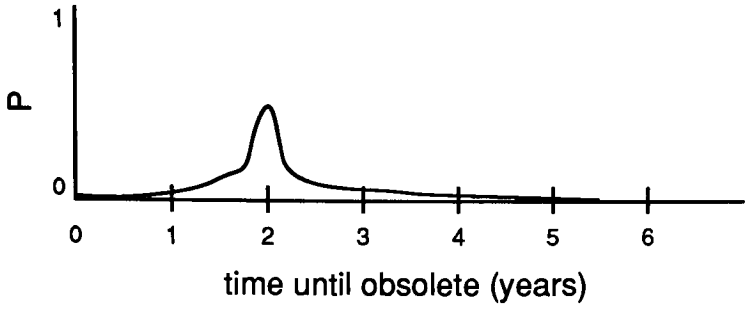

Figure 1a—the probability of a personal computer becoming obsolete vs. time: According to the curve, it is most probable that the computer will become obsolete after two years; however, the probability for the time being shorter or longer is much smaller.

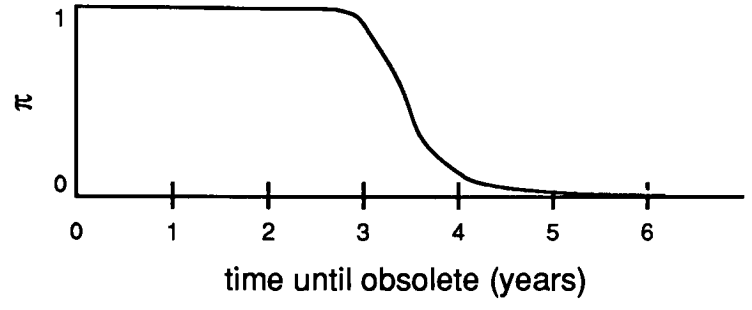

Figure 1b—the possibility of a personal computer becoming obsolete vs. time: According to the curve, it is equally possible for the computer to become obsolete from zero to three years; these cases have already been seen in the history of the computer to date

Notice also the different constraints on probability (P) and possibility (π):

$$\sum_i P(x_i) = 1$$

$$\max_i \pi_{x_i} = 1$$

The possibility function defines an upper bound on the probability function.

Operations on fuzzy sets

Once fuzzy sets have been defined in the universe of discourse, operations on these sets must be defined. To fulfill the purpose of this thesis, three operations must be defined: union, intersection, and inverse. Since fuzzy set theory is an extension of conventional set theory, fuzzy operations should be extensions of the conventional operations.

Union

Conventionally, the union of two sets (A and B) is defined as the set of all x such that x is a member of A, or x is a member of B:

$$A \cup B = \{x | (x \in A) \vee (x \in B)\}$$

To expand this to fuzzy set theory, we must consider the fact that not all objects are members of the sets to the same degree. In addition, we can say that all objects are members of any set to some degree; the degree would be zero for those items that "are not" members of the set. The union operation can be defined by replacing the "or" function with the maximum value function. Thus the degree of membership in the union of two sets (A and B) is the maximum of the degree of membership in A and the degree of membership in B:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Note that the fuzzy union operation still holds for conventional sets. If an object x is in a conventional set A, then $\mu_A(x) = 1$. Since the maximum value of the function $\mu_A(x_i)$ for any x_i is one, the degree of membership of x in the union is also one, indicating that x is in the union of the two sets. If the object x is in neither a member of set A nor B, $\mu_A(x) = \mu_B(x) = 0$; the maximum is 0, indicating that x is not a member of the union.

Intersection

The intersection of two sets (A and B) is conventionally defined as the set of all x such that x is a member of A and x is a member of B:

$$A \cap B = \{x | (x \in A) \wedge (x \in B)\}$$

To expand this definition to include fuzzy sets, we will replace the "and" operation with the minimum value function. Thus the degree of membership in the intersection of two sets (A and B) is the minimum of the degree of membership in A and the degree of membership in B:

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

Again note that the fuzzy intersection operation holds for conventional sets. If an object x is not in a conventional set A, then $\mu_A(x) = 0$. Since the minimum value of $\mu_A(x_i)$ for any x_i is zero, the degree of membership of x in the intersection is also zero, indicating that x is not in the intersection of the two sets. If the object x is a member of both sets A and B, $\mu_A(x) = \mu_B(x) = 1$; the minimum is 1, indicating that x is a member of the intersection.

Inverse

The convention definition of the inverse of a set (A) is all x such that x is not a member of A:

$$\overline{A} = \{x | x \notin A\}$$

To expand this definition of inverse to include fuzzy sets requires a little more thought. We will define the degree of membership of x in the inverse of A to represent the degree to which x is not a member of A. If an object x is a member of a fuzzy set to a degree $\mu_A(x)$, then, logically, the degree to which it is not a member is the upper limit of the function $\mu_A(x)$ minus $\mu_A(x)$:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

This also holds for conventional sets. If an object x is a member of a set A then $\mu_A(x) = 1$. The degree of membership in the inverse is thus $1 - 1 = 0$, indicating

that x is not a member of the inverse. If an object x is not a member of A, $\mu_A(x) = 0$; $1 - 0$ is 1, indicating that x is a member of the inverse.

Fuzzy numbers

Although we can continue to describe objects by giving the degree of membership in a particular fuzzy set, it is often more convenient to describe the object as being equal to something. We will call this "something" a FUZZY NUMBER: an entity that encompasses both the "name" of the fuzzy set and the degree of membership in that set. On a more general level, we can describe the object as being equal to a set of fuzzy numbers. For example, the object *joe* could be described by the following statement:

$$joe = \{(young = 0.1),(middle\ age = 0.8),(old = 0.3)\}$$

This statement is equivalent to:

$$\mu_{young}(joe) = 0.1$$
$$\mu_{middle\ age}(joe) = 0.8$$
$$\mu_{old}(joe) = 0.3$$

Decision making logic

Although fuzzy set theory provides the foundation of a fuzzy logic system, decision making logic is needed to provide the interface between the system inputs and outputs, which have been represented by fuzzy numbers.

In order for a fuzzy logic system to make a decision, several steps must be taken. Although three of these steps are also found in conventional decision making systems:

- the determination of the system variables
- the determination of the desired level of "quantization"
- the determination of the rule base

three of the steps are unique to fuzzy logic systems:

- the fuzzification of the system inputs
- the evaluation of the fuzzy rule base
- the defuzzification of the decisions

Determination of the system variables

Like a conventional decision making system, the fuzzy system must gather data and determine the form of the output before a decision can be made. To do this, we must first determine what elements of the system must be measured.

To control the inverted pendulum, for example:

- the control system needs to know the angular position and angular velocity of the pendulum.

- the system would then use this information to determine how much torque (and in what direction) should be applied to stabilize it.

- to adjust the torque output, the system could determine the amount of current to supply to the motor.

In this example, position, velocity and current are LINGUISTIC VARIABLES: a variable aspect of the system that does not not have a number as a value, but a fuzzy number, or a set of fuzzy numbers.

Determination of the quantization level

Once the linguistic variables are chosen, we must choose the quantization level for each one. In a conventional control system, the quantization level is often analogous to the binary representation of the inputs. We might choose, for instance, to quantize the angular position of the pendulum to an eight bit binary value; the position would be read by an analog sensor and converted by an analog to digital converter which would "round" the analog signal to one of 256 different values.

In a fuzzy system, determining the quantization level goes beyond just determining the desired binary representation of the input data. In addition to quantizing the data mathematically, it must be quantized *linguistically* as well. Recall that the input variables are linguistic variables—a set of fuzzy numbers or LINGUISTIC TERMS. Each of these terms should describe the linguistic variable differently, while still relating to each other; these terms are the adjectives in a "fuzzy sentence": the position is *small*, the velocity is *fast*. By quantizing the linguistic variable, we are deciding the number of different linguistic terms (fuzzy numbers) to use to describe it. For example, an initial attempt to stabilize the inverted pendulum might quantize the angular position of the pendulum into three terms: *to the left*, *center*, and *to the right*. Since the angle still has a definite numerical value, we might choose more general terms: *negative*, *zero*, and *positive*. A later revision of the system might decrease the quantization (by

increasing the number of linguistic terms) to seven levels: *negative big, negative medium, negative small, zero, positive small, positive medium,* and *positive big.*

Is less quantization always better?

When determining the quantization level, the designer must keep the application in mind. Although a controller with nine linguistic terms might be able to bring the pendulum to the target angle very quickly with negligible overshoot from any start angle, if the system requirements are much more relaxed, much time and money has been wasted. For example, if the pendulum will never stray outside of ±90°, and a two-degree overshoot is acceptable, the designer may only need to implement a system with three linguistic terms. Not only will this save development time, but the resulting control software will be able to be run on less sophisticated (and thus less expensive) hardware.

As a rule, quantize much as is allowed by the system specifications. It is very easy to add rules and linguistic terms to a simple system to correct for "unacceptable" behavior, but it may be much more difficult to remove rules and terms from a complicated system without introducing unacceptable behavior.

Determination of the rule base

Although not done by all control systems, evaluating a set of rules is a simple way to control an object; rules are written for the different states of the system under control, each rule instructing the controller on how to respond to a certain situation. In our fuzzy logic controller, we will use a set of if-then type rules to determine the system output. The rules will follow the conventional if-then format:

if *antecedent* then *consequence*

The antecedent is a conventional boolean logic expression, using the "and," "or," and "not" operations:

logical value$_1$ op *logical value*$_2$ op...

Logical values are comparisons of the system inputs to the terms of the linguistic variable. For the moment, forget that the value of a linguistic variable is a set of fuzzy numbers. Instead, for this discussion, the value of a linguistic variable is the name of one of its terms. For example, the value of the linguistic variable *angle* mentioned previously might be *zero*. This allows us to isolate the states of the system and generate rules for each of the states independently. To signify the comparison, we will use the "is" operator, denoting that the value of the input on the left side is equal to the right side:

angle is *zero* $\Leftrightarrow$ *angle* = *zero*

The consequence of the rule will be an executable expression, using the operator "should be." This denotes assignment, assigning the value on the right to the output on the left:

current should be *positive* ⇔ *current* ← *positive*

The following are two possible rules for the inverted pendulum controller:

if the *angle* is *positive*
then the *current* should be *negative*

if the *angle* is zero and
the *speed* is *negative*
then the *current* should be *positive*

To list the rules in a fuzzy logic controller, the tabular method (described below) is often used. In many applications, algorithms such as Quine-McKlusky or Espresso would be used to minimize the number of operations needed to evaluate all the rules; however, this is prohibited by the tabular method. Although the number of rules needed may be far from minimized, listing the rules in this form allows the individual states of the system to be scrutinized more independently. This in turn will make the process of adding and deleting rules and linguistic terms to the system simpler. In addition to the above benefit, specifying the rules using this method provides a direct and simple translation to program implementation.

The tabular method for listing rules involves first representing the expression of each rule as a "sum of minterms." In this case, "sum of minterms" is used to mean a sequence of logical expressions combined only using the "or" operation. In turn, each of these smaller logical expressions only contain individual logical values combined using the "and" operation. The rule is then split into

individual rules, one for each "minterm." For example, the rule (this rule is not necessarily meaningful):

if the *angle* is *positive* or
the *angle* is *zero* and the *speed* is *negative* or
the *speed* is *positive*
then the *current* should be *zero*

would be broken down into the following six rules (note that duplicates can be discarded):

if the *angle* is *positive* and the *speed* is *positive*
then the *current* should be *zero*

if the *angle* is *positive* and the *speed* is *zero*
then the *current* should be *zero*

if the *angle* is *positive* and the *speed* is *negative*
then the *current* should be *zero*

if the *angle* is *zero* and the *speed* is *negative*
then the *current* should be *zero*

if the *angle* is *zero* and the *speed* is *positive*
then the *current* should be *zero*

if the *angle* is *negative* and the *speed* is *positive*
then the *current* should be *zero*

By looking at the expanded list of rules, it is easier to see how listing the rules like this makes modifying the rule base and implementing it simpler. For example, if experimentation showed that the first minterm (the first rule in the expanded set) should be deleted, the designer would have to find all logical terms in the original rule set that include that minterm. In this case, two terms of the

original rule would have to be modified. Also, the form of the expanded rule listing is regular; the rules can be stored in a simple two dimensional array.

Instead of listing all of the rules in "longhand" like above, we will list them in an n-dimensional table, with n equal to the number of input linguistic variables. The size of the array is:

$$m_1 \times m_2 \times \ldots \times m_n$$

with m_i equal to the number of linguistic terms of linguistic variable i. For the inverted pendulum example, with two inputs, and three linguistic terms, the rules would be listed in a 3×3 array. The rows of the table will represent the angle, and the columns will represent the speed. The value of each cell will be the value that the current should take if that rule fires. Figure 2 shows how the expanded list of rules would be listed:

		Speed		
		Neg	Zero	Pos
Ang	Neg			Zero
	Zero	Zero		Zero
	Pos	Zero	Zero	Zero

Figure 2—a rule table with two linguistic variables, each with three linguistic terms

A more meaningful set of rules for the inverted pendulum controller would be as follows:

		Speed		
		Neg	Zero	Pos
Ang	Neg	Pos	Pos	Pos
	Zero	Pos	Zero	Neg
	Pos	Neg	Neg	Neg

Figure 3—rules for an inverted pendulum controller using two linguistic variables, each with three terms

Once the rules are defined, we can return to the fact that the value of a linguistic variable is a set of fuzzy numbers. This means that when the rules are evaluated, multiple rules may fire, and different rules may fire to different degrees. Our decision making logic will need to know what to do in these cases. This will be covered later.

Fuzzification of the system inputs

Probably the most important part of the fuzzy logic controller is the fuzzification of the input values. It is in this step that we determine the values of the linguistic variables by converting the numerical value of the input to a set of fuzzy numbers.

Consider again the linguistic variable *angle*, with three linguistic terms: *negative*, *zero*, and *positive*. Common sense comes in to play now, for we must teach the computer what we mean when we say something is *negative*, *zero*, or *positive*; we must determine the functions $\mu_{negative}(x)$, $\mu_{zero}(x)$, and $\mu_{positive}(x)$.

Before defining *zero*, we need to determine what *zero* will mean to the control system. When the *angle* is *zero*, the system should be trying to hold the

pendulum still; the controller will be trying to make small corrections in order to keep the pendulum stable, not to try to bring the pendulum to the target angle.

Next, we must determine what we mean by *zero*; clearly, the value 0.0 is definitely a member of *zero* ($\mu_{zero}(0) = 1.0$). Likewise, $\pm\infty$ could not be farther from *zero* ($\mu_{zero}(\pm\infty) = 0.0$).

Next, we must now consider the values in between. By definition, $\mu_{zero}(x)$ represents the degree to which the value x is *zero*, and that values closer to *zero* should have a higher degree of membership. Thus, the closer the value x is to 0.0, the greater the value of $\mu_{zero}(x)$. Figure 4 represents a first attempt at defining *zero*.

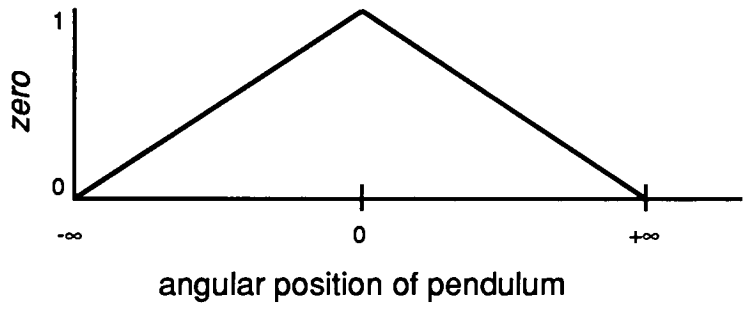

Figure 4—First attempt at defining the linguistic term zero

We must now consider the application of *zero* to the system. Values not in the universe of discourse should be excluded; since the domain of the angular position of the pendulum is $-180 \leq x \leq +180$, zero can be redefined to exclude values not in the domain (figure 5); however, the definition is still not practical.

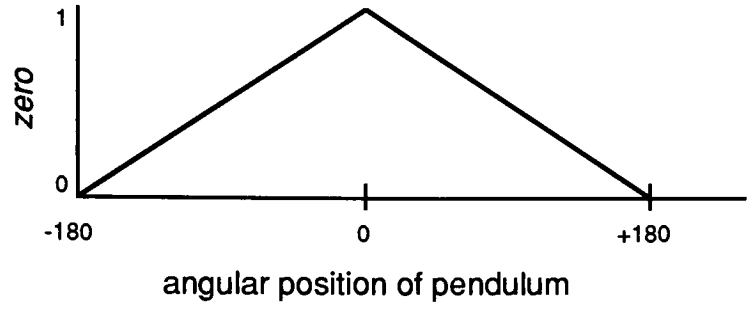

Figure 5—Definition of zero corrected for the domain of the angular position of the pendulum

Remember that these fuzzy definitions are the basis on which the control decisions will be made. Although, -100° is more *zero* than -175°, do we really want the system to behave differently when the pendulum is at either of these two positions? In both cases, it is very far from the destination angle, and the system shouldn't need to worry if the pendulum is close to the target angle yet. We need to modify the definition of *zero* to exclude angles that "do not apply." To do this, we will use the notion of membership in a fuzzy set as a possibility distribution. By asking the question "Could the angle, within reasonable uncertainty, be 0?," we can decide what angles should be excluded from a discussion of *zero*. The following chart (figure 6) shows a much more suitable definition of *zero* for a linguistic variable with three terms:

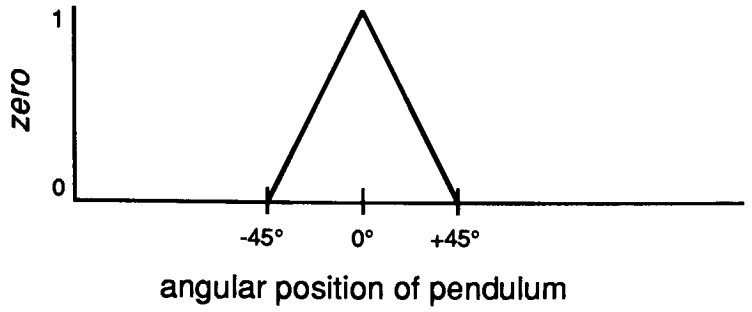

Figure 6—a suitable definition of the linguistic term zero

By setting the cutoff angle to ±45°, we are instructing the system to start trying to stabilize the pendulum when the angle is within this range. Outside this range, other terms will be trying to move the pendulum quickly to the target angle.

If the linguistic variable was less quantized (more linguistic terms), the cutoff angle (±45° in the chart above) might need to be smaller because the angle is being defined more finely; each of our definitions (*zero, positive, negative, ...*) is more specific.

Using the same method, *negative* and *positive* can be defined. The following chart (figure 7) shows the three linguistic terms overlaid:

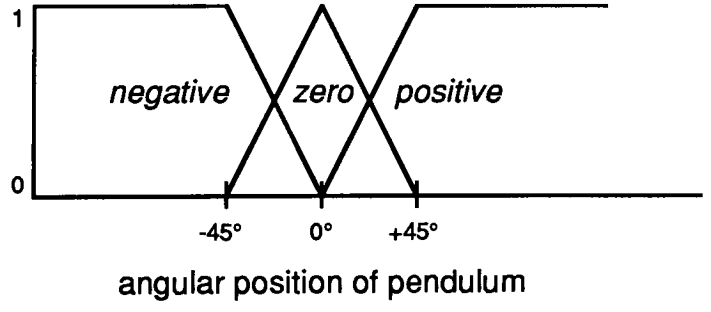

Figure 7—the three linguistic terms: negative, zero, and positive

Notice that the three terms overlap. Although the definition of a linguistic variable does not require that the terms overlap, it is very important to smooth operation of the fuzzy system. Consider each term as a separate case, with separate rules for each case. If the terms did not overlap, there would be only one

rule contributing to the output of the system at one time. As the pendulum rotated, sharp transitions could occur as the angle moves from one case to another. In actuality, the system does not have separate cases; at any one time, multiple cases apply to different degrees, requiring us to design the system to handle the naturally occurring overlap in the states of the system.

Consider the fuzzy set *old* again, and a related set *young*. As someone ages, they become more *old*, and less *young*. At any one time, we do not consider someone exclusively either *old* or *young*, but a combination of both. It is natural that they overlap.

Curve shape and art

Although piecewise-linear curves were used in the previous examples, other curve shapes might be more appropriate for other situations. For example, *zero* could be defined by using sine and cosine segments (figure 8):

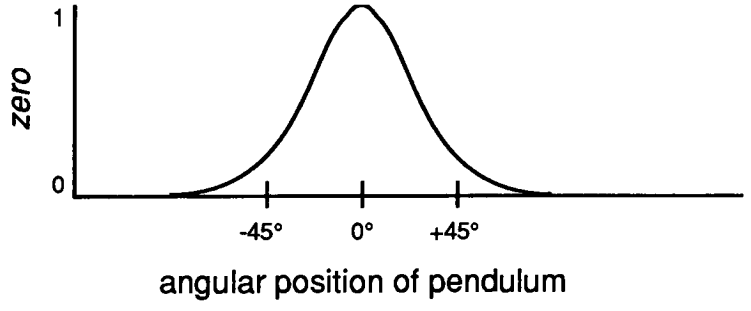

Figure 8—an alternate definition of zero using sine and cosine segments

Although curve shape and cutoff points effect system behavior, there is no "science" in choosing them; if anything, it is an art form. Often, once the basic curve shape is chosen, the designer must use trial and error to find the "perfect" cutoff points and curve shape. In other situations, it may be natural to choose a sinusoidal or exponential curve shape to describe something. Although we can

analyze a curve shape and somewhat determine how the shape will affect the system, it is very difficult to determine if one shape is "better" than another.

Again, there may not be "perfect" cutoff points; after experimentation, we may find that a cutoff point for *zero* anywhere between 40° and 50° provides acceptable performance.

Evaluation of the fuzzy rule base

Once the system has been set-up, by defining the rules and linguistic terms, it can begin processing the input. As in conventional rule-based system, the controller converts the system inputs into the form used by the rule base. Next, it evaluates the antecedent of each rule to determine which rule (or rules) will fire. It then executes the consequence of each rule that does fire. A fuzzy logic controller still follows this basic premise.

Evaluating the logical terms

Once the system reads the inputs of the system, it must first convert them into a form that can be input to the rule base for evaluation. In our fuzzy rule base, we have written rules that take the terms of the linguistic variables as inputs. Although it was not explicitly stated, when we wrote "...the *angle* is *zero*..." in a rule, we were instructing the the rule evaluator to determine the degree of membership of the angle of the pendulum in the fuzzy set *zero*—to **FUZZIFY** the input. This is done by simply evaluating the function $\mu_{zero}(x)$ at the current pendulum angle.

Once each linguistic term in the antecedent has been fuzzified, the controller must evaluate the entire antecedent. It is then represented by a single value. This

value represents the weight of the rule (to what degree we believe that this rule should be executed). To more clearly show the evaluation process, consider the following two rules, when the angle of the pendulum is 10° and the speed is 20°/s:

if	the *angle* is *zero* and the *speed* is *positive*
then	the *current* should be *negative*

if	the *angle* is *zero* and the *speed* is *zero*
then	the *current* should be *zero*

The current state of the pendulum in comparison to the antecedent of the rules is shown by the following figure (figure 9):

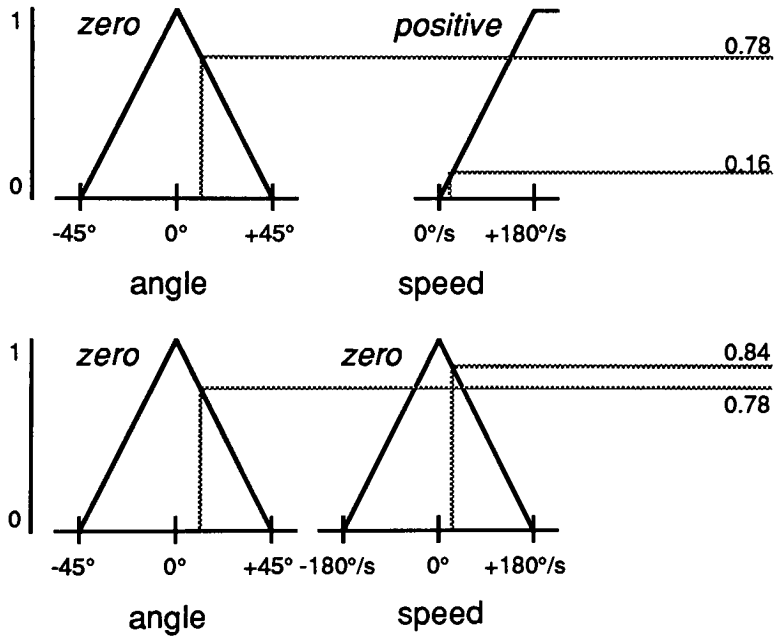

Figure 9—fuzzification of the angle and speed for the two rules

Evaluation of the antecedent

Once fuzzified, the rule evaluator must combine the inputs to the rule using whatever logic operations specified in the rule. To do this, we must define the operations "or," "and," and "not" for fuzzy values. Just as boolean logic

operations are analogous to the conventional set operations (union, intersection, and inverse), fuzzy logic operations are analogous to the fuzzy set operations. The following table (figure 10) shows the boolean operations and their fuzzy logic equivalents:

Bool.	Set	Fuzzy	
OR	union	max	$A \wedge B \equiv \max(\mu_s(a), \mu_s(b))$
AND	intersection	min	$A \vee B \equiv \min(\mu_s(a), \mu_s(b))$
NOT	inverse	1 - val	$\neg A \equiv 1 - \mu_s(a)$

Figure 10—Summary of fuzzy logic operations and the relationship between logic and fuzzy set theory

Using these definitions of "or," "and," and "not," the weights (value of the antecedent) of the two sample rules are 0.16 (the minimum of 0.78 and 0.16) and 0.78 (the minimum of 0.78 and 0.84), respectively.

Defuzzification of the decisions

After the evaluation of the antecedents, we are left with a set of output linguistic terms and corresponding weights (the current should be *negative* with weight 0.16 and *zero* with weight 0.78), each corresponding to particular rule. These weights and terms must now be combined to produce a single value to output from the system. Although (like the determination of curve shape and cutoff values) the combination of the weights and terms is an art, several established methods exist. Two common methods for defuzzification are the Max-Product and the Max-Dot methods. Another method, developed specifically for this thesis is the Nominal Average method.

Max-Product

In the Max-Product defuzzification method, the output linguistic terms are clipped to the weight of the attached rule. A curve is then created by taking the

maximum of the clipped terms at any point. By then finding the centroid (center of gravity) of the resulting curve, we can calculate the desired output value.

Each rule i has a weight w_i and a corresponding output linguistic term T_i. The resulting output membership function for rule i (R_i) is:

$$\mu_{R_i}(x) = \max\left(w_i, \mu_{T_i}(x)\right)$$

These output curves are then overlaid to construct a single output curve. The value of the combined curve at x is the maximum of the values of the individual clipped functions at x:

$$\mu_{out}(x) = \underset{all\ rules}{\mathrm{MAX}}\ \mu_{R_i}(x)$$

The centroid is determined by finding the value x such that half of the weight of the overlaid functions is on each side of x:

$$\int_{x_{min}}^{x} \mu_{out}(v)dv = \int_{x}^{x_{max}} \mu_{out}(v)dv$$

The following figure (figure 11) shows how the Max-Product method would be used to defuzzify the two rules given previously:

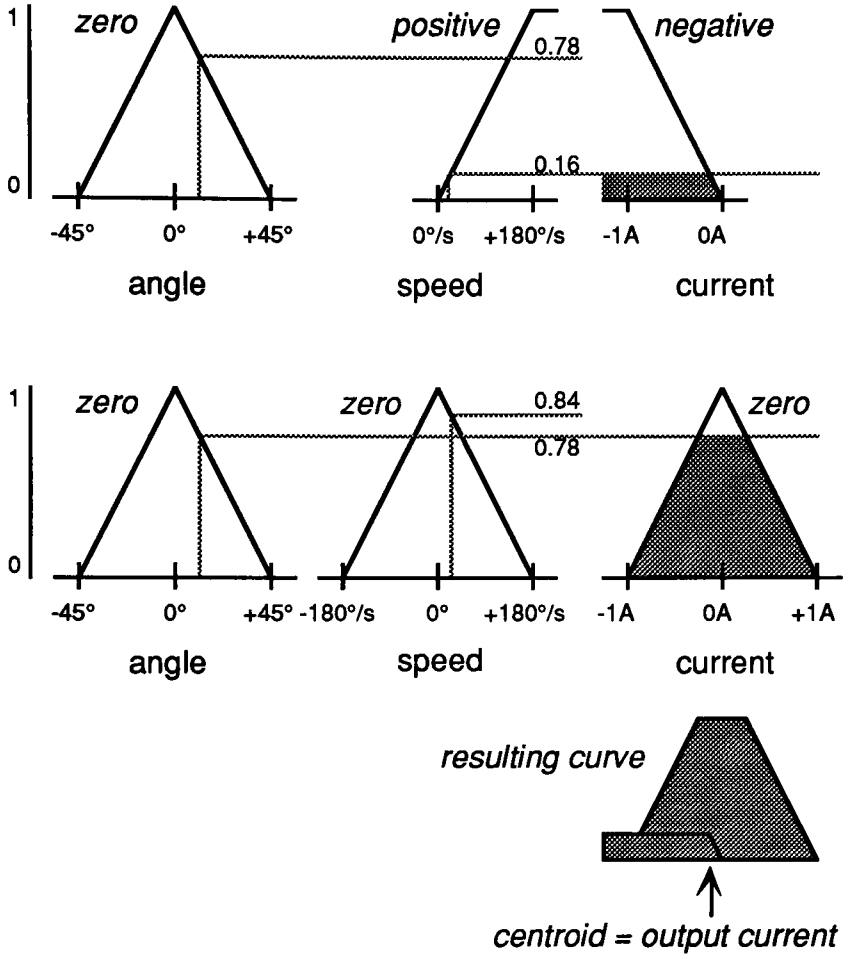

Figure 11—Max-Product method of fuzzification. The antecedent output sets the maximum value of the output linguistic term. The resulting curves are then overlaid, the centroid of the result being the output current

Max-Dot

One apparent problem with the Max-Product method is that most of the detail found in the output linguistic terms is lost when the clipping occurs. To fix this problem, the Max-Dot method calls for the output terms to be scaled instead of clipped. The disadvantage is that Max-Dot requires multiplication, which can be time consuming on small microcontrollers. The membership function for the fuzzy set R_i for Max-Dot is given by the equation:

$$\mu_{R_i}(x) = w_i \mu_{T_i}(x)$$

Aside from this modification, the Max-Dot method is identical to the Max-Product method.

The following illustration (figure 12) shows how the Max-Dot method would be used to defuzzify the two rules given previously:

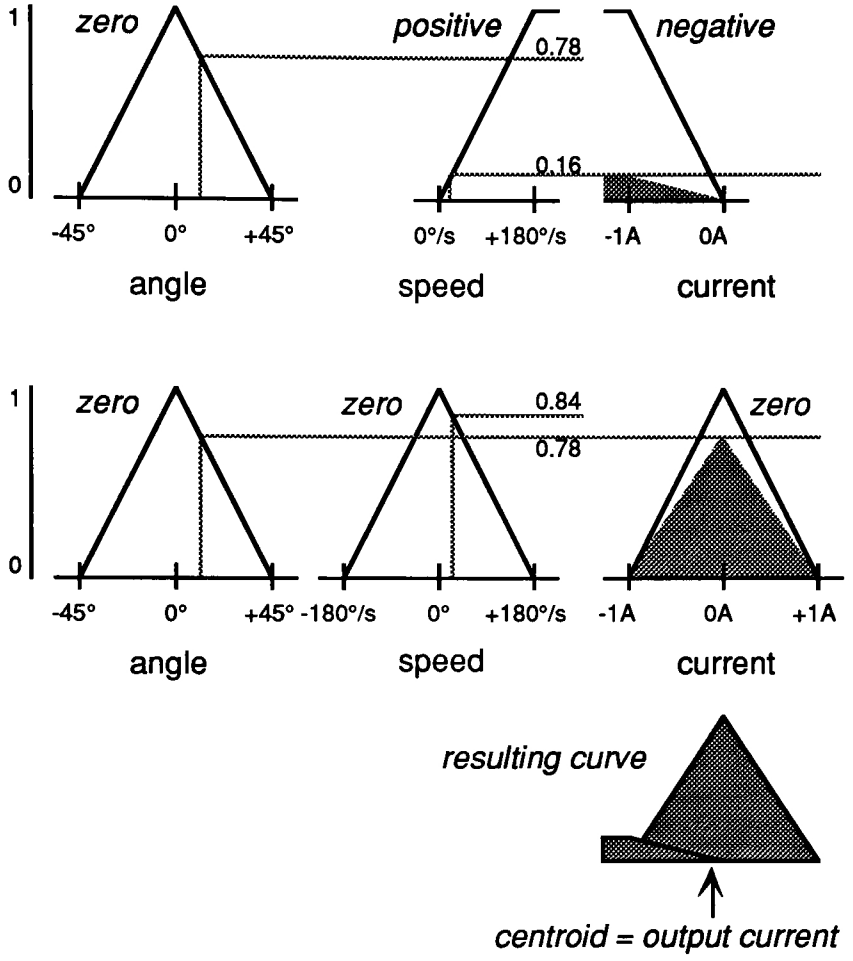

Figure 12—Max-Dot method of fuzzification. The antecedent output sets the scale of the output linguistic term. The resulting curves are then overlaid, the centroid of the result being the output current

Nominal Average

One major disadvantage that both Max-Product and Max-Dot require is the need for the calculation of the centroid, which requires two integrals. If the system inputs are discrete values, such as eight-bit binary values converted by an analog to digital converter, the integrals can be reduced to simple summations; however, in a continuous-valued system, the actual integrals (or approximations) need to be calculated. This is further calculated by the addition of non-piecewise-linear membership functions; the calculation of the approximation must broken down into many more sections for the result to be accurate. To solve this problem, a method is needed that reduces the output linguistic terms to a single, non-fuzzy value—the nominal value.

Let n_R be the nominal value of output linguistic term R; it is the value of the system variable with the greatest degree of membership in the R. The term "nominal" comes from the use of names for the linguistic terms; the nominal value is the value for which the set was named. For example, the nominal value of the linguistic term *zero* is 0.0. For terms that do not have a single value with the highest degree of membership (such as *positive* and *negative*), the nominal value of the linguistic term is the value at which the derivative of the membership function changes, and the degree of membership is a global maximum. For some continuous functions, the nominal value can be more simply defined by the location of a unique global maximum:

$$\left(\left(\frac{d\mu_T}{dx}_{+} \neq \frac{d\mu_T}{dx}_{-}\right) \wedge \left(\max(\mu_T) = \mu_T(x)\right)\right) \vee$$
$$\left(\left(\max(\mu_T) = \mu_T(x)\right) \wedge \left\{x' \neq x \middle| \mu_T(x) > \mu_T(x')\right\}\right)$$

For the linguistic terms *positive* and *negative* defined previously for the linguistic variable *angle*, the nominal values would be +45° and -45° respectively.

Note that this definition may not hold for all membership functions; this definitions assumes that the function is not continuous. Functions using sinusoids, for example (see figure 8), may be continuous despite the plateaus after the cutoff points; there is no instantaneous change in the derivative. For cases such as this, the nominal value is the location of a global maximum with different neighboring values:

$$\left(\mu_T(x) = \max(\mu_T)\right) \wedge$$
$$\left(\left(\mu_T(x+\varepsilon) \neq \mu_T(x)\right) \vee \left(\mu_T(x-\varepsilon) \neq \mu_T(x)\right)\right),\ \varepsilon > 0$$

The rules are then combined and the result converted to real world values by computing the weighted average of the nominal values, where w_i is the weight of rule T_i with nominal value n_{R_i}:

$$v_{out} = \frac{\sum_{rules} w_i n_{R_i}}{\sum_{rules} w_i}$$

The advantages of this method are clear for continuous value systems. The nominal value of each linguistic term can be calculated ahead of time. This leaves only the weighted average to be calculated each iteration of the controller. Although this method requires several multiplications and a division, the time required to calculate the average will most likely be dwarfed by the time to calculate the values of the antecedents the rules, which could conceivably require floating point trigonometric or exponential operations.

The Nominal Average method is not without its problems. It cannot, for example, deal with membership functions with multiple global maximums,

such as *not zero*, the inverse of the linguistic term *zero*. In addition, all detail in the membership function is lost.

Other methods

Just as there is no "right" shape for a membership curve, there is no right choice for the defuzzification method. Often, the choice is made when the hardware for the system is chosen; many methods may be simply too difficult to implement or cannot be performed quickly enough on a particular system. In other instances, the designer may wish to modify an aspect of a method to take advantage of the behavior of the device under control.

How can we allow art?

With all this uncertainty in the system, how can we be sure that the system will be tolerant of external variables, such as gravity and motor torque? The answer to this lies in the system development process. Recall the things considered when developing the inverted pendulum control system. When we constructed the rule base, the only thing considered was the direction that we should push the pendulum for a particular angular position and velocity. When developing the membership functions, the actual input value was the only item discussed. Nowhere in the development of the system did we consider gravity or motor torque.

Of course at some point we must consider these things. For example, when choosing a motor and maximum current output, we must be sure that the motor will be able to overpower gravity (at least at critical points). We must also be sure that the motor chosen does not overpower gravity so much that the pendulum spins out of control.

Even here, though, gravity and motor torque are only being examined in general terms. The system does not rely on a particular value of either of these things, thus small fluctuations in them will go unnoticed by the controller.

Two Pendulum

To demonstrate the principles of fuzzy logic, Pendulum, a fuzzy logic inverted pendulum simulator, was developed. Pendulum allows the user to experiment with fuzzy logic principles by setting up the rules and membership functions used to control the inverted pendulum. The user can then run the simulator (either in single–step or free–running mode) and see how the system behaves by watching strip charts, a graphical display of the pendulum, and the numerical values of important aspects of the system.

Pendulum physics

The pendulum will be simulated on a step by step basis. Each iteration of the simulator, a new value of angular position (θ) and angular velocity (ω) will be calculated. Using these values, the simulator will use fuzzy logic to determine the amount of current to supply to the motor. The simulator will then use this new value of current to determine the new torque output of the motor, and new angular position and velocity of the pendulum a finite (user definable) time later. During this time, the torque due to gravity is assumed to be constant. Because of this, the new status of the pendulum is only an approximation, although the difference between the approximation and the actual value is negligible assuming the change in angle is small. This will be examined more closely later.

In order to simulate an actual inverted pendulum, it was analyzed to determine how both nature and the motor affect its speed and position. The following illustration shows the forces affecting the pendulum (figure 13):

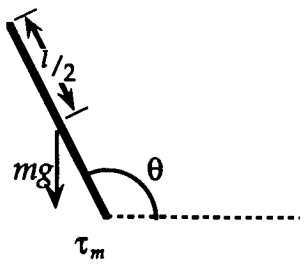

Figure 13—Free body diagram of the inverted pendulum

Two torques affect the pendulum: that due to gravity, and that supplied by the motor. First, consider the simpler of the two: the torque supplied by the motor. The motor being simulated is powered by a variable current source with user definable limits. The output of the motor is linear with respect to current:

$$\tau_m = \tau_{m_{max}} i$$

Determining the torque due to gravity is more complicated. The magnitude of the torque about a point of rotation is equal to the magnitude of the force normal to pendulum, multiplied by the distance from the point of rotation to which the force is applied. With a pendulum of length l and mass m:

$$\tau_g = -\frac{l}{2} mg \cos\theta$$

Note that the torque due to gravity is dependant on the angular position of the pendulum. This is where the simulation becomes an approximation. We can assume that τ_g is constant over the simulation interval (one simulation step) provided that the change in angular position of the pendulum is small over that interval. This can be achieved by keeping the time per interval small.

Recall the general angular motion equations:

$$\theta_n = \theta_o + \omega_o T + \frac{T^2}{2}\alpha$$

$$\omega_n = \omega_o + \alpha T$$

Also recall the equation for angular acceleration (α) where τ is the total torque applied and I is the moment of inertia of the pendulum:

$$\alpha = \frac{\tau}{I}$$

For the inverted pendulum, the equation for α develops as follows:

$$\alpha = \alpha_m + \alpha_g$$

$$= \frac{\tau_m}{I} + \frac{\tau_g}{I}$$

$$I = \frac{ml^2}{3}$$

$$\alpha_m = \frac{3\tau_{m_{max}}}{ml^2} i$$

$$\alpha_g = \frac{-\frac{l}{2} mg\cos}{\frac{ml^2}{3}} = -\frac{3g}{2l}\cos\theta$$

$$\alpha = \frac{3\tau_{m_{max}}}{ml^2} i - \frac{3g}{2l}\cos\theta$$

The general equations for angular motion thus become:

$$\theta_n = \theta_o + \omega_o T + \frac{T^2}{2}\left(\frac{3\tau_{m_{max}}}{ml^2} i - \frac{3g}{2l}\cos\theta_o\right)$$

$$\omega_n = \omega_o + T\left(\frac{3\tau_{m_{max}}}{ml^2} i - \frac{3g}{2l}\cos\theta_o\right)$$

Fuzzy algorithm

The fuzzy logic algorithm chosen for Pendulum is the basic method used to introduce fuzzy logic. To defuzzify the output, the Nominal value method (developed earlier) is used.

The following code fragment shows how the rules are evaluated and the output is defuzzified by Pendulum. `row` and `col` count through the rows and columns of the rule table. `memb[]` is an array of the membership functions. `rules[][]` is the rule table.

```
sum ← 0
weightedSum ← 0

τn ← Normalize(τ,ANGLE)
ωn ← Normalize(ω,VELOCITY)

for row ← 1 to maxRow
   for co ← 1 to maxCol
      begin
         weight ←  min(
                     CurveVal(memb[row],τn),
                     CurveVal(memb[col],ωn)
                   )
         sum ← sum + weight
         weightedSum ← weightedSum + weight

   Nominal(rules[row][col],CURRENT)
           * weight
      end

newCurrent ← (weightedSum / sum)
```

Of course, the actual implementation of this algorithm is optimized. The values of the curves and the nominal values are only calculated once per iteration of the controller, and no values are calculated for rules not in use. Also, the possibility of a divide–by–zero error is eliminated in the calculation of the `newCurrent`.

Displaying information

One of the strongest points of Pendulum is its ability to display tremendous amounts of information to the user, ranging from a graphical representation of the pendulum to a chart of the weight of a rule over time. This information is displayed in five types of windows:

- Pendulum window
- Rule editor
- Membership editor
- Strip charts
- Variable display

Pendulum window

The pendulum window (figure 14) is the "home base" window of the simulation; if a simulation is open, this window will be open. It is in this window that Pendulum shows the graphical representation of the pendulum. Three items are drawn in this window:

- the pendulum
- tolerance indicators
- the target angle indicator

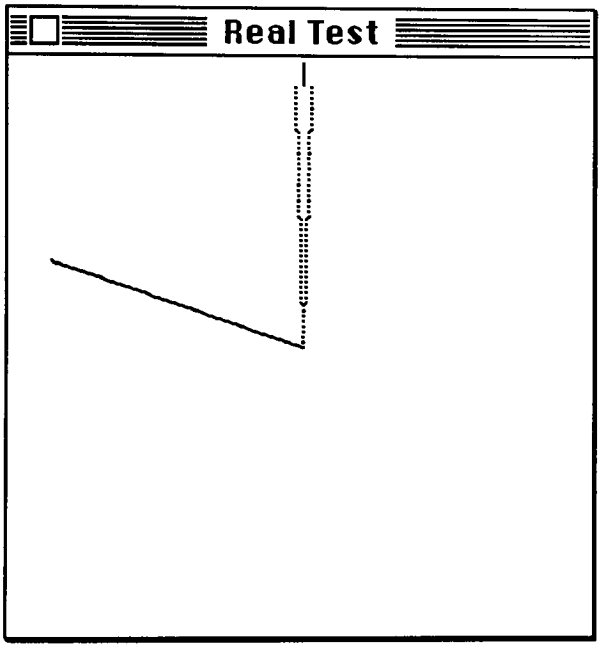

Figure 14—the pendulum window. The pendulum is the long line "rotating" about the center of the window. The target angle indicator, at the top of the window, is bordered by two grey lines. These lines indicate the tolerance chosen by the user for the target angle.

This window is most useful for getting a quick idea of the performance of the system, although the pendulum can only be shown within approximately ±1°. Although tolerance lines are shown, they do not affect system performance in any manor; they only exist to help the designer determine the rough performance of the system.

As this window "is" the simulation, closing this window also closes all other windows associated with this simulation. The pendulum window can be moved around the desktop, but it cannot be resized.

Rule editor

Figure 15 shows the rule editor. It is through this window that the designer can specify the rules of the system. Pendulum uses the table method of rule specification, and allows a maximum of nine linguistic terms (*negative big, negative medium, negative small, negative zero, zero, positive zero, positive small, positive medium,* and *positive big*). Rows of the table represent the angular position, while columns represent the angular velocity. The rules shown in figure 15 are the same rules listed back in figure 3, used to explain the table method.

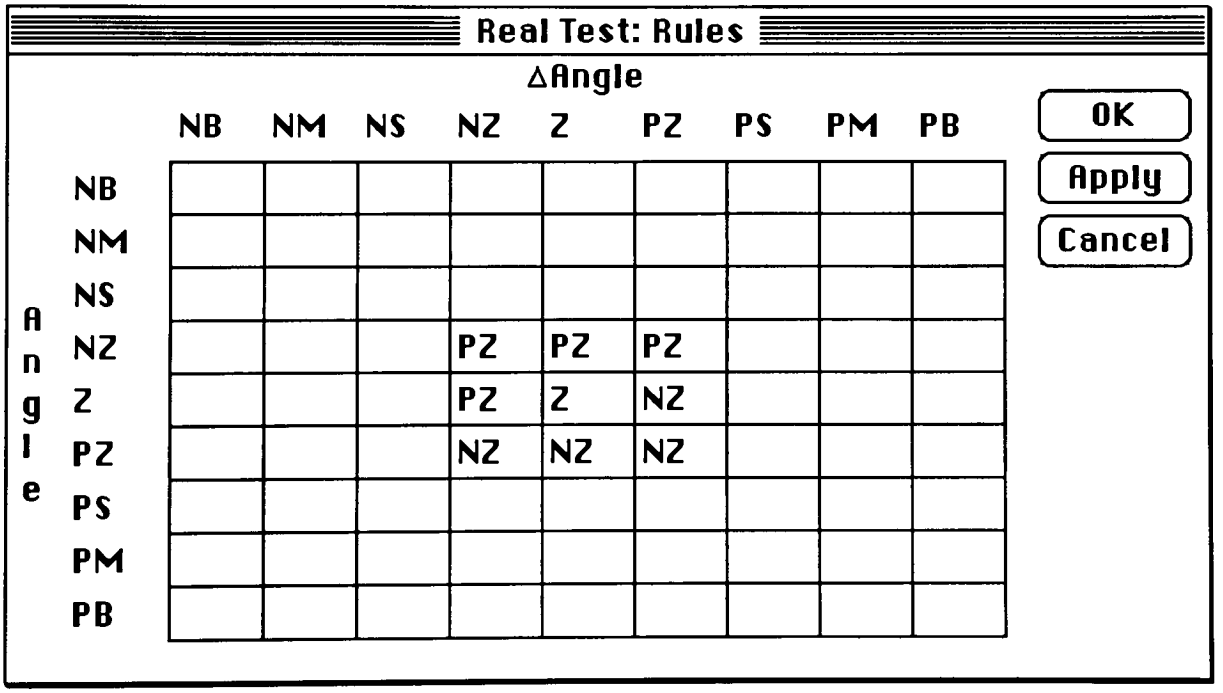

Figure 15—The rule editor. Each cell in the table represents a different state of the system; the rows represent the angular position, and the columns represent the angular velocity. An empty cell indicates that no action is specified for that state.

Pendulum also allows for the rules to be changed while a simulation is running.

Membership editor

To edit a membership function for a particular linguistic term, the membership editor is used (figure 16). Pendulum allows the function to be specified in four independent sections (A, B, C, and D). Each of these sections can take on one of five shapes: straight line, concave or convex sine, or concave or convex arctangent.

To specify a section, select the section using the section radio buttons. The bars above and bellow the curve display will then change to reflect the x range currently specified for the selected section. The current shape button will also be selected.

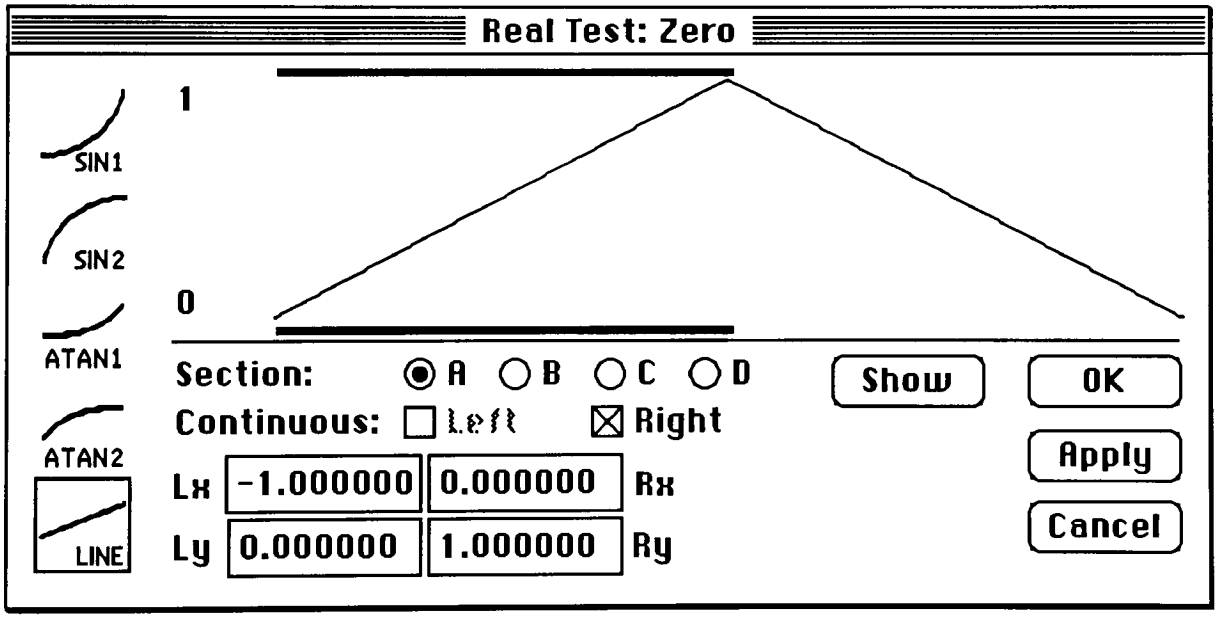

Figure 16—the membership editor (for zero). As shown, section A is currently selected, and has a straight line curve shape. The right continuous button is checked, indicating that the curve does not "jump" between sections A and B.

A separate editor can be opened concurrently for each of the nine linguistic terms, although only one can be active at a time. The editors can also be active while a simulation is running, allowing for the membership functions to be changed in the middle of a simulation.

Strip charts

Often, it is helpful to view how the value of a variable changes over time. To do this, Pendulum allows the user to open strip charts (figure 17).

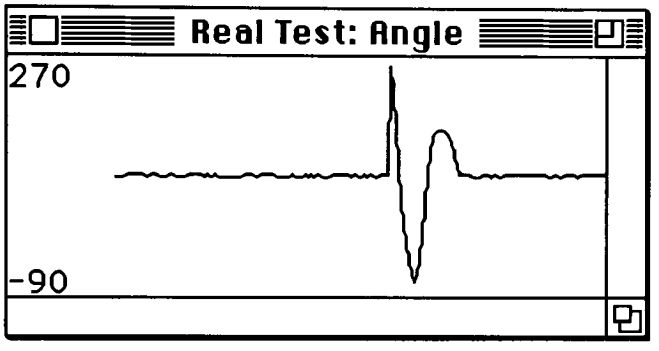

Figure 17—a strip chart

Pendulum allows four different types of variables to be displayed in strip charts:

- the pendulum angular position
- the angular velocity
- the output current
- the weights of the rules

Unlike Pendulum's other window types, strip charts are resizable. Once resized, the window will rescale itself in the y direction. The x scale is always one pixel per simulation step.

Variable display

Much of the system development can be done by watching the graphical displays; however, it is often helpful to be able to view the exact values of the system variables when fine tuning the system. Pendulum allows this through the Variable window (figure 18). This window shows the "exact" value of the pendulum angle, angular velocity, current, and simulation time (in steps). If either steady state noise or spikes are enabled, the value of the noise or spike is also displayed.

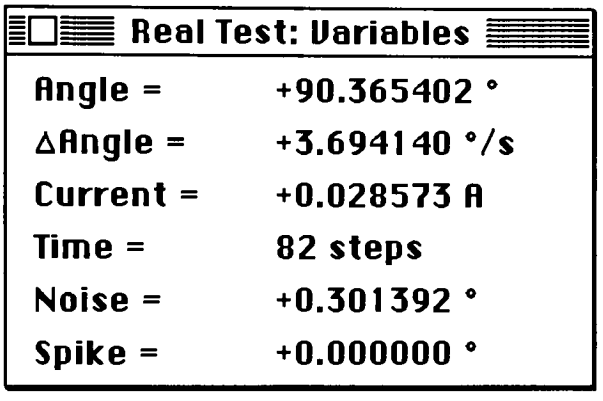

Figure 18—the variable window

Menus

As standard in Macintosh applications, many of Pendulum's featured are accessed through the menus.

Apple Menu

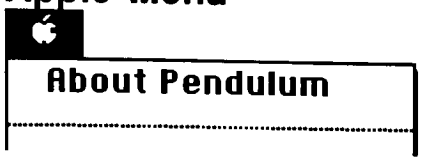

Figure 19—the Apple menu

Aside from desk accessories and other items controlled by the finder, Pendulum only places one item in the Apple menu: About Pendulum. Selecting this items brings up the credits for Pendulum.

File Menu

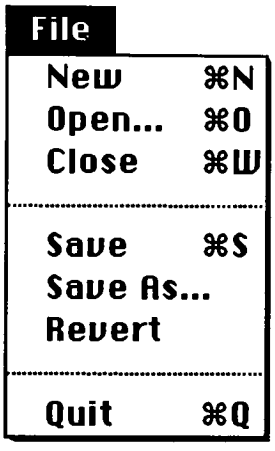

Figure 20—the File menu

All items in the File menu behave as specified by the Apple Human Interface Guidelines. The only limitation placed by Pendulum is a limit of ten open simulations at one time. Also, when the simulation is saved, the window positions are saved in addition to the system values. This allows the user to set up a complex set of strip charts and editors, and not have to re-open each window each time the simulation is opened.

Edit menu

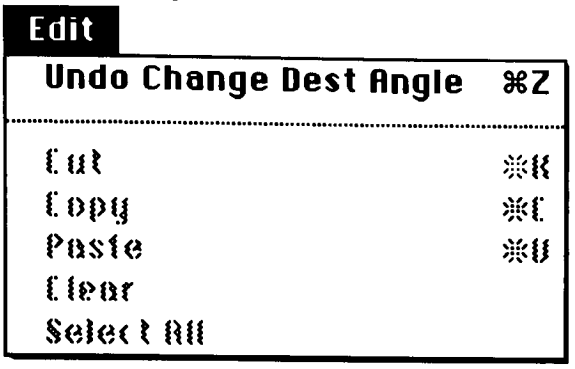

Figure 21—the Edit menu. In this situation, the action that will be undone if the Undo item is chosen is a change in the destination angle

Although the standard Edit menu items are present, they are not available while in Pendulum; To remain compatible with System software earlier than version 7.0, they must be present to allow desk accessories to function when opened from within Pendulum.

Pendulum does offer extensive Undo. Most changes, or sequence of changes can be undone. The action to be undone will be indicated in the text of the Undo item in the Edit menu. If Pendulum cannot undo an action, the item will read "Can't Undo."

Simulation menu

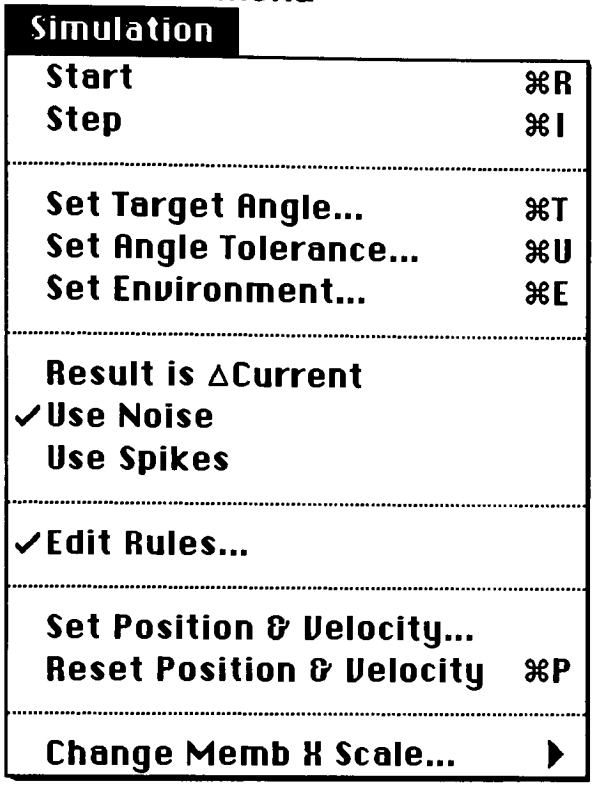

Figure 22—the Simulation menu

Most of Pendulum's functionality is accessed through the Simulation menu.

Start (or Stop)

The Start item allows the simulation to be started or stopped. When the system is started, the text of the menu item will change to "Stop." The simulation then runs on its own until it is stopped.

Step

This item allows the user to take a single simulation step. This is useful for observing critical areas of operation when the free running simulation is too fast.

Set Target Angle...

Through this menu item the user can set the destination angle of the pendulum

Set Tolerance...

This menu item allows the user to choose the tolerance of the system. This only affect the tolerance lines in the Pendulum window, not the behavior of the system.

Set Environment...

Through this menu item, the user can set environment variables, such as motor torque, maximum noise, pendulum length and weight, and the acceleration due to gravity.

Result is ΔCurrent

Because Pendulum allows the target angle to change, it must also allow the interpretation of the current output to be changed. If this item is unchecked, the rule output is interpreted as the new current to be supplied to the motor. This would be used when the current output should be zero when the pendulum is stable at the target angle.

If the item is checked, the current output is interpreted as the amount the current supplied to the motor is changed. This would be used then a constant current output is required to keep the pendulum stable at the target angle.

Use Noise

Use Spikes

If these items are checked, noise and or spikes will be applied to the "measurement" of the angular position of the pendulum, as specified by the settings entered in the Set Environment... dialog box.

Edit Rules...

This item opens the rule editor. A check next to the item indicates that the rule editor is already open. Selecting the item while the rule editor is open brings it to the front.

Set Position and Velocity...

Selecting this item allows the user to set the position, velocity and simulation time of the pendulum. This is useful for watching the pendulum after it starts from a known state.

Reset Position and Velocity

This item resets the pendulum position and velocity, and simulation time back to the last set point. If it was not previously set, the position is set to 0°, the velocity is set to 0°/s and the simulation time is set to 0. This is useful for repeatedly watching the simulation in a particular situation.

Change membership X scale...

This item brings up a the following submenu (figure 23):

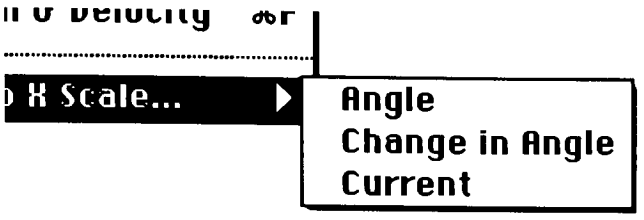

Figure 23—the Change membership X scale... submenu

Through this submenu, the user sets the normalization factors for the three linguistic variables. The value set here is the value of the system variable represented by 1.0 in the membership functions.

For example, to define *zero* for angles as in figure 6, the *zero* membership function would be defined from -1.0 to +1.0, and the angle scale would be set to 45.

Edit Membership menu

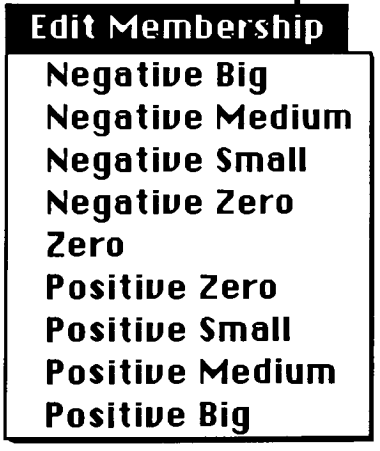

Figure 24—the Edit Membership menu

Selecting an item in this menu opens the respective membership editor. Once opened, the item is checked; if it is reselected, the editor will be brought to the front.

Display menu

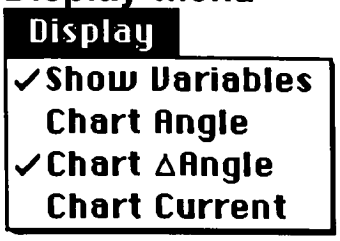

Figure 25—the Display menu

Selecting items from the Display menu brings up the appropriate strip chart or variables window, and checks the selected item. If an open window is selected, it is brought to the front.

Editing the rule base

To edit the rule base, first open the rule editor by selecting "Edit Rules..." from the Simulation Menu. Once you have determined the desired value of a rule, click and hold the mouse button down on the cell in the table corresponding to the rule to be set. A menu of the possible output linguistic terms will pop–up (figure 26); from this menu select the desired term. If you wish no action to be taken on this state, select "unused" from the menu.

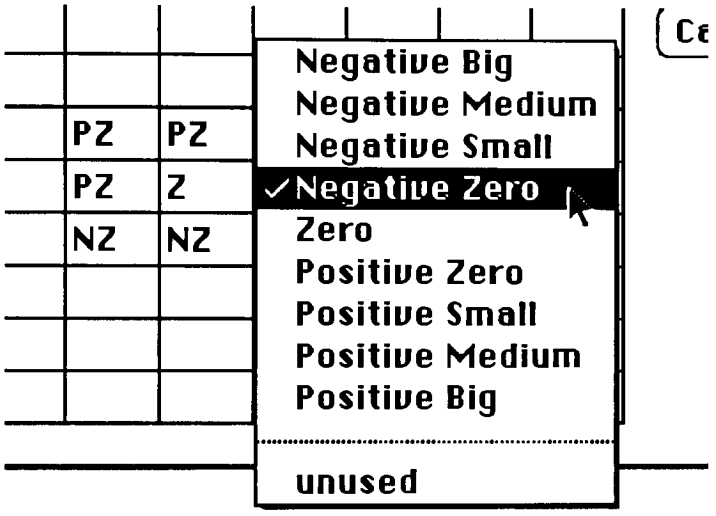

Figure 26—setting a rule

Once all the rules have been set, hit either the "OK" or the "Apply" button. Both buttons will copy the rule choices back to the simulation and place them in use. "OK" will also close the rule editor. Hitting the "Cancel" button reverts back to the rules in use then the editor was opened. All changes will be discarded, even if they had been previously been applied to the simulation.

Strip charts of the rule weights are also opened through the rule editor. To open a chart for a particular rule, click and hold the mouse button down on the cell for that rule *while holding down the option key*. A menu will then pop–up, allowing

you to select the chart (figure 27). If the chart is already open, the item will be checked. Re–selecting it will bring the chart to the front.

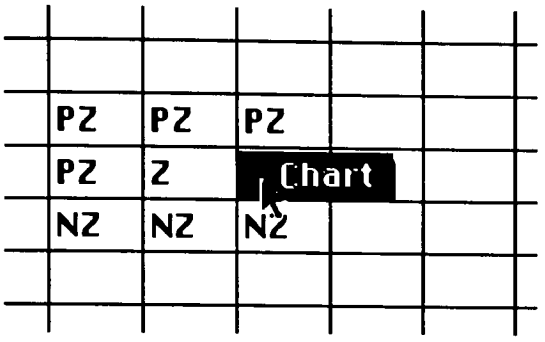

Figure 27—opening a strip chart for a rule

Editing membership functions

In addition to specifying the rule base, all membership functions that are to be used must be defined. To open the membership editor, select the function to define from the "Edit Membership" menu. All four curve sections must be defined. Also note that normalized curves should be defined. These curves will be used for all three linguistic variables (*angle, speed,* and *current*).

Choose the section to define by clicking on its radio button in the "Section" section of the editor. The bars above and below the section will then change to reflect the current x axis range of the section. When specifying the section, changes will not be reflected in the graph of the curve unless you click on the "Show" button, or change sections. To fully specify the selected curve section, the following items must be specified:

- the curve shape: this is specified by selecting the icon of the desired shape.
- the endpoints of the section: these are specified by entering the numeric values in the text boxes. Although usually not practical, the right x value may

be less than the left *x* value. This is useful if you only want the particular section of the sine curve, for example, with the large slope (figure 28). When specifying the *x* values, remember that Pendulum determines the section for a particular value from left to right.

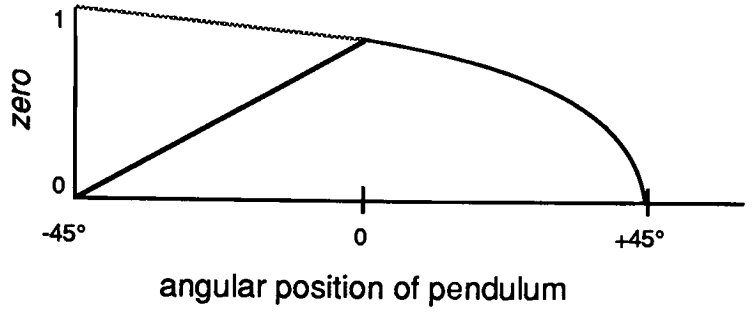

Figure 28—use of overlapping sections to specify a curve

Also note that Pendulum does not allow gaps in the x-axis. Changes in the left X of C, for example, will be reflected in the right X of B.

The following code fragment illustrates how the curve is evaluated:

```
if x <= curve[A].leftX
   then return curve[A].leftY
else if x >= curve[D].rightX
   then return curve[D].rightY
else for i ← A to C do
   if x <= curve[i].rightX
      then return Evaluate(x,curve[i])
```

Note that if the left *x* value is set less than the right, that curve section is not available for real data

- the left and right continuousness: if these check boxes are checked, Pendulum will restrict the endpoint so that the section is piece-wise continuous with its left and/or right neighbors. If the right continuous box is checked for section A, for example, a value entered in the right Y box will be copied to the left Y value of section B.

Once all sections are defined, the curve is copied back to the simulation by clicking on the "Apply" or "OK" button. In addition to copying back the membership curve, the "OK" button also closes the membership editor. Clicking on the "Cancel" button discards all changes since the editor was opened (even if they had been applied back to the simulation) and closes the editor.

In addition to defining the normalized curves, the scales of each variable must be entered. These are entered by selecting the variable from the "Change membership X scale" submenu of the "Simulation" menu.

- NOTE: in addition to setting the scale for normalizing the current, the current scale setting also functions as the maximum magnitude of current (in amperes) that the controller will output.

Changing the environment

In addition to defining the fuzzy system, the non-fuzzy environment must also be specified. This is done by selecting the "Set Environment..." item in the "Simulation" menu. The following dialog box will appear (figure 29):

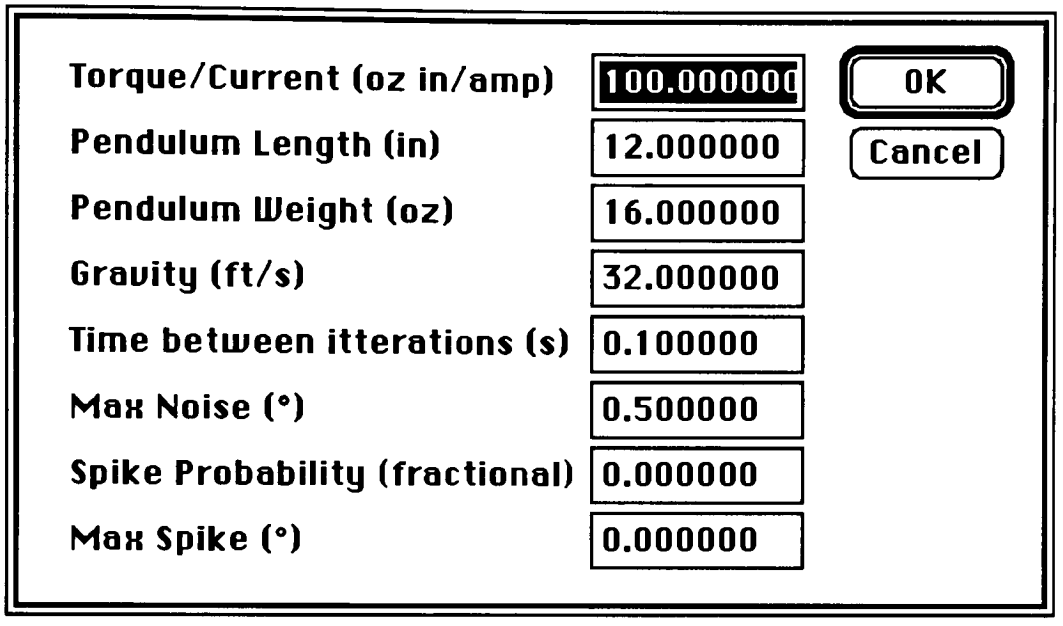

Figure 29—the Set Environment... dialog box

All values should be specified as stated, although the weight of the pendulum should be entered assuming gravity is 32 ft/s^2, not the value entered for gravity; this allows the two values to be modified independently.

Running the system

Once the entire system is specified, the simulation can be started by selecting "Start" from the "Simulation" menu, or run step by step with the "Step" command. Note that the system does not run in "real" time. One simulation step is taken approximately every tenth of a second. This allows reasonable response while still allowing for application running in the background to

request CPU time. Pendulum can also run simulations in the background, but only approximately four steps are taken every second.

Saving the system

To prevent the system from having to be setup each time Pendulum is started, Pendulum allows the system to be saved. In addition to saving the rules, membership functions, and environment, Pendulum also saves the positions of the open windows. The current pendulum angle, speed, and simulation time are not saved.

Three System creation

Now that we have an understanding of the fuzzy logic principles and how to use Pendulum to develop the system, we can begin creating a fuzzy system to control the inverted pendulum. An overall design methodology will be presented which will simplify the design process, minimizing the amount of "redesign" while developing the fuzzy system.

Desired environment

Before we can design a system to control and inverted pendulum, the pendulum's environment must be set. For this design example, we will use the following values:

- Pendulum length = 12 in
- Pendulum weight = 16 oz
- Motor strength = 100 oz in/A
- Acceleration due to gravity = 32 ft/s
- Time interval = 0.01s

Design goals

Two goals will be in mind while designing the inverted pendulum controller:

- minimize settling time: we want the pendulum to settle at the target angle as soon as possible. For this thesis, settling time will be defined as the number of simulation steps needed to bring the pendulum permanently (without the influence of noise and spikes) within the tolerance region.

- minimize the overshoot: ideally, we want the pendulum to stop at the target angle without passing it and having to turn around. Practically, we want the

to minimize the distance that the pendulum overshoots the target angle. For this thesis, overshoot will be defined as the greatest angular displacement between the target angle and a point at which the pendulum turns around after passing the target angle.

First attempt

Once the goals are defined, we can make a first design attempt. The design methodology we will be using is simple: implement the minimum system to perform the task. For a first attempt, we will try to create the simplest system possible that could control the inverted pendulum; we will use the minimum number of linguistic terms and rules. Only after we have a system that can control the inverted pendulum will we worry about minimizing the overshoot and settling time.

The first attempt system will use three linguistic terms: *negative, zero,* and *positive.* These terms will be mapped to the following terms in Pendulum: *negative small, zero,* and *positive small.* This will allow expansion in both directions if it is later needed.

With only three linguistic terms, few possibilities for the rule base exist. To simplify the process of creating rules, one more decision must be made. Although it is important that we consider both the angular position of the pendulum as well as the speed, are both linguistic variables equally important? In this case, since the angular position is more important, more emphasis should be placed on the angle when determining the rules; when creating a rule for a particular situation, the angle should be considered first. The speed is then considered only if we cannot make a decision based upon the angle alone.

Although this methodology simplifies the determination of the rule base, it cannot be used in all systems, even if one linguistic variable is much more important than the rest. For example, if Pendulum did not place strict limits on the current output, it would be probable that ignoring the speed might allow the controller to keep increasing the speed until the system becomes unstable.

By examining the problem, we can determine that we will need the following complex rules:

if the *angle* is negative *small*
then the *current* should be *positive small*

if the *angle* is positive *small*
then the *current* should be *negative small*

if the *angle* is *zero* and the *speed* is *negative small*
then the *current* should be *positive small*

if the *angle* is *zero* and the *speed* is *zero*
then the *current* should be *zero*

if the *angle* is *zero* and the *speed* is *positive small*

then the *current* should be *negative small*

These rules map into the following rule table (figure 30):

	NB	NM	NS	NZ	Z	PZ	PS	PM	PB
NB									
NM									
NS			PS		PS		PS		
NZ									
Z			PS		Z		NS		
PZ									
PS			NS		NS		NS		
PM									
PB									

Figure 30—rule set for the first attempt. Only three linguistic terms will be used, requiring at most nine rules.

Zero, negative small, and *positive small* must now be defined. For a first attempt, the membership functions will be defined as follows:

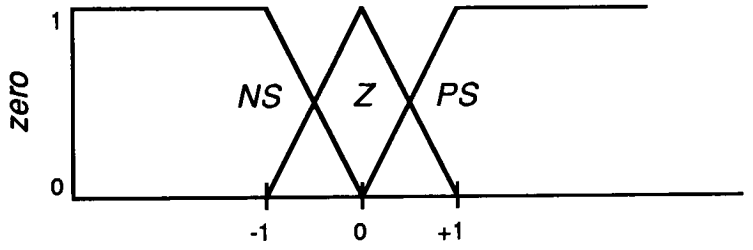

Figure 31—membership functions for the first attempt

The normalization factors will be set as follows:

- angle = 45°
- Δangle = 180°/s
- current = 1A

First attempt performance

Despite the simple nature of the system, our first attempt performed very well, as summarized by the following tables (figure 32):

Overshoot

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.797°	0.349°	0.774°
	270°	51.614°	46.816°	47.206°

Settling Time

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.43s	1.10s	0.61s
	270°	1.46s	1.58s	1.55s

Figure 32—performance of first attempt system

If the real pendulum will be constrained from 0° to 180°, our first system is all that is needed. In all three cases the overshoot was less than half of the tolerance, and the pendulum settled very quickly. If we cannot make this restriction on the pendulum angle, the system needs to be improved. Although the system still settles very quickly in the other three cases, the overshoot should be decreased significantly. Since our first system only has three linguistic terms, the system cannot begin to slow the pendulum until it is *zero*; by adding another linguistic term on each side of *zero*, we can add rules that will begin to slow the pendulum sooner, thus decreasing the overshoot.

Second attempt

We have established one goal for the second revision of the inverted pendulum controller: to significantly decrease the overshoot from a 270° starting angle. Before we can begin modifying the system, some analysis of the current system must be done.

We must decide if the undesirable behavior (large overshoot) is due to a rule in the current system, or because no rules exist to prevent it. Because of the simplicity of the first system, we can easily determine that the later is the case; the system does not have any rules to prevent a large overshoot if the pendulum approaches the target angle at a high speed. Although the controller starts decreasing the current once the pendulum comes within 45° of the target angle, it is not smart enough to decrease it (to allow gravity to slow the pendulum) fast enough to prevent the large overshoot.

This decision tells us that we should only add to the existing system, not remove rules currently in the rule base. Remember that this is only a heuristic, not a design algorithm; after experimenting with the system, we may find that one or more rules should be removed.

When modifying the controller (both the rule base and the membership functions), change as little as possible; by keeping the amount changed between revisions minimal, we hope to prevent negative progress from revision to revision.

After analyzing the system, the rules and membership functions were changed as follows (figure 33):

	NB	NM	NS	NZ	Z	PZ	PS	PM	PB
NB									
NM									
NS			PS	PS	PS	PS	PS		
NZ			PS	PZ	PZ	PZ	Z		
Z			PS	PZ	Z	NZ	NS		
PZ			Z	NZ	NZ	NZ	NS		
PS			NS	NS	NS	NS	NS		
PM									
PB									

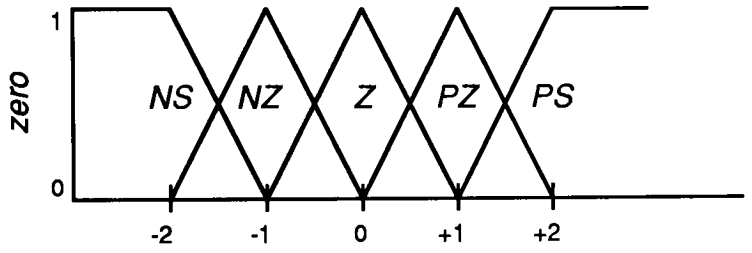

Figure 33—rules and membership functions for second attempt

Normalization factors were not changed. Note that although the maximum current output (current scale) is still one ampere, the controller may ask for two amperes to be output. Although the output of the system will seem as course as the first attempt, it is not. The decreased quantization is present all the way up to the final current output; the nominal average is taken at the decreased level of quantization.

Second attempt performance

As shown in the following figure (figure 34) we were able to meet our goals with the second revision. Even with the pendulum starting at 270° and 100°/s, the controller was able to keep the overshoot within the system tolerance. However,

notice that the decrease in overshoot did exact a cost; the settling times for the case where the pendulum starts at 0° were slightly greater.

Overshoot

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.100°	0.094°	0.100°
	270°	1.951°	1.581°	1.596°

Settling Time

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.54s	1.17s	0.73s
	270°	0.54s	.60s	.58s

Figure 34—overshoot and settling times for second attempt

After only two revisions, we have designed a system that meets our needs. Had this attempt not fully succeeded, two options exist for the next attempt:

- The normalization factors could be adjusted. This option would be most practical if the new system failed because it was slowing down too soon or too late.

- The quantization could be decreased again. This option would be better suited if we could determine that we still did not create rules to provide the good behavior needed.

There are no hard, fast rules for developing fuzzy systems; it is in a way a form of art; we only have heuristics for doing things, not algorithms.

Four *Fixing what ain't broke*

Although we were able to create a working inverted pendulum control system easily, several questions are still up in the air. We have made several design decisions, such as cutoff points and curve shape, in the name of "art." The system worked beautifully despite these decisions, but what would the result have been had different decisions been made? In this sections, we will explore the result of making the following modifications to the second revision of the inverted pendulum controller:

- changing the normalizations scales
- changing the curve shape
- removing rules that don't seem to be "needed"

Changing the normalization scales

When designing the control system, we set the normalization scales by picking cutoff points that "seemed right," without really trying to analyze the system to determine the best choice. Despite our ignorance, we succeeded in developing a working system. Now it is time to analyze the cutoff points.

Angle normalization scale

When designing the system, we picked 45° to be normalized to one, but what would be the effect of choosing a different value. This analysis is simplest if we only consider the linguistic term *zero*; We have chosen ±45° as the limits of *zero*. By increasing this value, we will include more values in our definition of zero. Because the angle will start becoming *zero* sooner, rules designed to stop the pendulum will come into effect sooner; the pendulum will approach the target angle at a slower speed, which should decrease the overshoot. Of course, since the approach is slower, the settling time should increase.

We can verify our analysis by changing the values in Pendulum; we are indeed correct. By increasing the angle normalization scale, the settling time increases, and the overshoot decreases. By 52°, the overshoot from the worst case (starting at 270° and -100°/s) has been reduced to 0°; however, the settling time increased by 0.17s (to 0.73s). If a zero overshoot is needed, it may be wise to make this change. Although the percent increase in settling time is large, it is very small compared to "human" time. Although we can benefit from increasing the normalization factor, we must be wary of trying for a too perfect system. Although "perfect" results are achieved from the worst case with a scale of 52°, the performance is worse if the pendulum starts at 0° and 0°/s. As we increase the normalization factor, we instruct the system to begin slowing the pendulum down sooner. If the pendulum is not travelling very quickly as it reaches the *zero* cutoff point, it may not be able to reach the target angle because the system decreased the speed too soon. This is the case if the angle normalization factor is increased to 60°.

Next, we should consider the effect of decreasing the angle normalization factor. This would, in effect, decrease the limits of *zero*, causing the rules designed to stop the pendulum at the target angle to fire later. This should bring the pendulum to the target quicker (smaller settling time), although since it will be moving more quickly, the overshoot should be greater.

We can again verify our analysis by modifying the values in Pendulum; again, we are correct. Although the overshoot does increase and the pendulum comes to the target angle more quickly, we do not see the expected decrease in settling time. This is because of our definition of settling time: the time required to bring the pendulum permanently into the tolerance region. Because 45° is the smallest *zero* cutoff angle that can keep the pendulum within the tolerance region on the

first pass, the settling time jumps as we decrease the angle normalization factor from 45° to 44°. A jump like this will occur every time the overshoot increases the enough that the pendulum must make another pass over the target angle before stabilizing. The controller did perform correctly with angle normalization factors as small as 3°; however, the performance was dismal. The overshoot had increased to almost 180° and the settling time was over 18s.

Changing the speed normalization scale

In addition to changing the angle normalization scale, we can change the speed normalization scale. Again, by analyzing the system, we can determine that if we increase the scale, the settling time should decrease and the overshoot should increase since the *zero* rules will not fire as soon; however, this is not the case. If we enter new values into Pendulum, we find that the opposite is the case: the settling time increases and the overshoot decreases.

This can be explained by recalling the rule base for the second revision, and reconsidering the other linguistic terms. As we decrease the normalization factor, the speed falls into the *negative zero* and *negative small* terms more strongly. When the speed is mostly *zero* (as is the case with the original normalization factor, the rules call for the current to progress smoothly from *positive/negative small* to *zero*; however, when the speed stays more *negative*, the output current stays *positive* longer, slowing the pendulum down sooner. Thus the settling time increases and the overshoot decreases; the overshoot can be eliminated by decreasing the scale to 150° (with the settling time increasing to 0.61s).

From this analysis, we can assume that the opposite with thus occur if we increase the speed normalization scale. As expected, Pendulum confirms our

hypothesis. Even after increasing the speed normalization factor to 8000°/s, in effect eliminating the speed from control decisions (the maximum speed the pendulum achieves under motor control is about 450°/s), the system still can balance the pendulum, although the overshoot is again over 180° and the settling time is several minutes.

Changing the curve shape

Changing the curve shape can be explored next. Although both systems were designed using piece-wise linear membership functions, we should try to determine the effects of other curve shapes. Consider the bell curve first (made of sine segments), and again only consider the linguistic term *zero*. By keeping the slope small at the edges of *zero*, the current output will be higher longer, resulting in faster speeds as the pendulum approaches the target angle. The membership function then quickly reaches the maximum value as the angle approaches 0.0; this may result in a larger overshoot.

To determine the actual effect of this curve shape, the curves were entered into Pendulum. As expected, the overshoot increased. In addition, because of the very small amount of current output at slow speeds near the target angle, the system was unable to bring the pendulum back to the target angle.

The problem with this curve shape seems to stem from the quick transition from low to high degree of confidence; to fix this, we can invert each curve segment (figure 35):

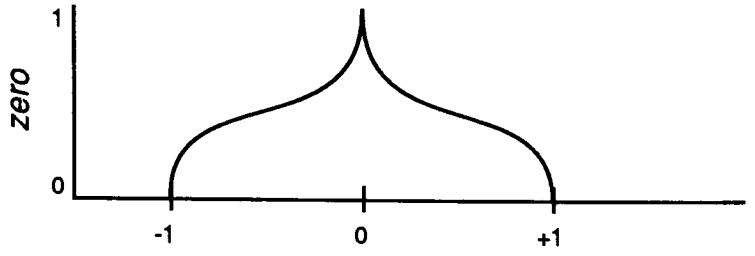

Figure 35—Backwards sine curve shape

With this curve shape, the transition occurs much more slowly. By trying this shape out in Pendulum, we find that it does solve the problems of the bell curve, although it didn't perform quite as well as the piece-wise linear membership functions (although tuning could eliminate the difference). Figure 36 shows how this system performed:

Overshoot		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.306°	0.269°	0.305°
	270°	3.460°	3.177°	3.179°
Settling Time		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.50s	1.13s	0.69s
	270°	0.52s	0.57s	0.55s

Figure 36—Performance of the controller using the backward sine membership functions

Changing the rule base

Although the rule base was the first part of the system designed, it is logically the last part to be modified. Often, especially in small systems, there is not a lot that can be changed around for improve system performance. Sometimes the output progression as one of the linguistic variables changes from its "minimum" term

to its "maximum" can be adjusted (becoming *zero* sooner as the *angle* changes from *negative small* to *zero*, for example).

A more interesting adjustment is to eliminate "unneeded" rules for the rule base. For example, in the second revision of the pendulum, we have placed the following rule:

if	the angle is positive small and the *speed* is *negative small*
then	the *current* should be *positive big*.

Examine the reasoning behind this rule. In other terms, the rule states "if the pendulum is far from the target angle and is moving very quickly towards the target, then push it a lot towards the target." If the pendulum is already moving very quickly toward the target angle, do we really want to push it even more, further increasing the speed. In a system such as Pendulum, where the output current is limited, this rule is probably having little overall effect, compared to the entire rule base; the current will only go so high, so trying to increase it further is futile.

In other systems, this rule may cause the pendulum to further speed up; the stabilization rules will then have a more difficult time slowing the pendulum down as it approaches the target angle. The effect will be an increase in overshoot, and possibly (if the overshoot does not increase past the tolerance) a decrease in settling time. Since we already have found three other stable methods (changing the angle normalization factor, changing the speed normalization factor, and changing the curve shape) to tune the settling time and overshoot, it is probably in our benefit to remove this rule from the rule base.

In addition to greater stability, removing rules will also decrease the complexity of the rule evaluation, and thus decrease the number of CPU cycles required to evaluate the rule base. This would allow the use of slower (and thus less expensive) or simpler hardware in the control system.

In addition to evaluating the usefulness of rules by analyzing the rules, we can use Pendulum to do the analysis. Strip charts can be opened for the rules governing a particular situation, and the simulation can be repeatedly run for the situation in question. By examining these strip charts, we can see how a particular rule weighs with respect to other rules.

Sometimes, the analysis methods will contradict, such as with the example rule above. Although logically the rule is not needed, in many situations it is the only rule that fires and is crucial to keeping the settling time small. In situations such as these we must reexamine our design goals; which is more important, system performance or execution speed?

After completing the analysis of the second revision rule base, it was reduced to the following:

	NB	NM	NS	NZ	Z	PZ	PS	PM	PB
NB									
NM									
NS			PS		PS				
NZ				PZ	PZ	PZ			
Z			PS	PZ	Z	NZ	NS		
PZ				NZ	NZ	NZ			
PS					NS		NS		
PM									
PB									

Figure 37—Reduced rule set

The system performs as follows:

Overshoot		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.133°	0.099°	0.112°
	270°	0.153°	0.157°	0.156°

Settling Time		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.54s	1.17s	0.72s
	270°	1.11s	1.13s	1.12s

Figure 38—Performance of reduced-rule system

For the case where the starting angle is 0°, the result are very close to the results with the full rule set, although the overshoot is slightly greater; however, for the case where the starting angle is 270°, the overshoot is much smaller and the settling time is much greater. This must be considered when deciding on the final system; which is most important: settling time, overshoot, or system simplicity?

Five Non-ideal worlds

At the onset of our endeavor, we decided that one of our goals was to create a system that would be tolerant of changes in the environment. Two items, the acceleration due to gravity, and the motor torque, are sure to fluctuate during the lifetime of the inverted pendulum controller; we should verify that our system is indeed tolerant of these fluctuations. Also, many sources are sure to introduce noise into the system; our system should also be tolerant of this.

Changes in gravity

As the inverted pendulum controller is moved from location to location on the planet, or even from planet to planet, it will see the acceleration due to gravity rise and fall. We need to determine the extent that the controller can deal with these changes.

By experimenting with Pendulum, we can show that for values of gravity between 31 and 33 ft/s^2, the deviation in performance of the system was negligible. In fact, the acceleration due to gravity can drop as far as 16 ft/s^2 if the pendulum starts at 0°, with the performance still being acceptable (the pendulum settles quickly, and the overshoot stays within the tolerance). Above 33 ft/s^2, the motor is too weak to overpower gravity. Although the performance is worse in the lower part of the range of gravity when the pendulum starts at 270° (because the pendulum gets moving too fast to prevent a large overshoot), the system is still able to stabilize the pendulum quickly.

Although the system would not be expected to perform well in a range of environments as great as this, it is comforting that a system as simple as the controller is this tolerant to changes in gravity. To create a conventional system that is this tolerant would be considerably more difficult, if not impossible.

Changes in motor torque

Unfortunately for system designers, it is impossible to build parts, both electrical and mechanical, that do not wear. For example, as the inverted pendulum controller is used, the motor will wear. It will be able to provide less and less torque as time passes. Eventually, it will need to be replaced. Unfortunately, the new motor may have a slightly different torque output than the original motor. Our control system needs to be tolerant to these changes.

Again by experimenting with Pendulum, we can test this tolerance. Like with changes in gravity, the inverted pendulum controller was very tolerant to changes in torque. With the pendulum starting at 270°, acceptable performance was achieved with torque ranging from 80 to 120 oz in/Amp; from starting at 0°, torques ranging from 98 to 120 oz in/Amp were acceptable (Note: the angle normalization scale was increased to 50° for these tests).

Again, our system is far more tolerant than is needed. If we start with a motor of torque 120 oz in/Amp, it can take wear that would reduce the power by over 18% before it would have to be replaced.

Noise in sensor measurements

In additions to fluctuations in the acceleration due to gravity and the deterioration of the motor, there will always be inherent noise (uncertainty) in the measurement of the angular position of the pendulum. The source of this uncertainty could be line noise, uncertainty in the analog to digital converter, error in the actual analog measurement of the angle, or many other sources. We need to test our inverted pendulum control system for tolerance of this noise;

the system should still be able to control the pendulum with acceptable precision despite the noise.

Pendulum can model this noise in two forms:

- constant random noise resulting in an uncertainty in the angular position of at most $\pm N_{max}°$.
- larger spikes in the measurement of the angle resulting in an uncertainty in the angular position of at most $\pm S_{max}°$. The probability that a spike will occur is P_s.

We need to determine the maximum value for these parameters that our system can tolerate. To better emphasize the benefits of fuzzy logic when it comes to noise, Pendulum implements noise in the angular position of the pendulum only; noise in this value does not carry through to the calculation of the angular velocity.

Constant random noise

Pendulum calculates constant random noise using the following formula:

$$-1 \le random_{\pm 1}() \le +1$$
$$N = N_{max} \times random()$$

This noise is added to the angle that is passed to to the rule evaluator. It does not affect the actual angular position of the pendulum. Although we can use Pendulum to make qualitative guesses as to the tolerance of the control system, because of the random nature of the noise, we cannot make a quantitative determination of the maximum noise value.

Using pendulum, we can start with small values of noise an increase until the system "seems" unstable. Since our tolerance of pendulum angle is ±2°, 0.5° seems like a good place to start. Watching the graphical windows (not the variable display) in Pendulum, the effect of the noise could not be noticed until the maximum noise was increased to 5°. In fact, the system did not become "unstable" until the maximum value was increased to 13°. As stated before, these measurements are only qualitative. Although the pendulum did occasionally sway outside the tolerance region with a maximum noise value of 12°, it did not seem to get "stuck" outside the region. Even with the noise as high as ±20°, the pendulum remained stable provided the noise moved between positive and negative values often; if the actual noise value stayed in the high positive values for more than ten or so simulation steps, the pendulum would get "stuck" outside the tolerance region.

Spikes

In addition to constant random noise, Pendulum can also add spikes to the measurement of the angular position of the pendulum. Pendulum calculates spikes using the following formula:

$$-1 \le random_{\pm 1}() \le +1$$

$$0 \le random_{0}() \le 1$$

$$S = \begin{cases} P_s \ge random_0() & S_{max} \times random_{\pm 1}() \\ otherwise & 0 \end{cases}$$

After our great success with constant random noise, we can be more trusting in the ability of the control system. Using Pendulum to simulate the spikes, the inverted pendulum control system was able to keep the pendulum "stable" even with a 5% chance of spikes resulting in an error in the measurement of the angle

of 150°. This level of spikes performed about as well as a constant noise maximum of 20°. As the frequency of spikes increases, the maximum spike value tolerated will decrease.

Why so tolerant

Although we introduced rather high noise values (with respect to the tolerance), the system was still able to keep the pendulum stable. Recall the definitions of the membership curves. When we defined *zero*, for example, we took this noise into account without even knowing it. Remember that one of the definitions of *zero* was the set of angles that we would consider placing in an "equivalent" non-fuzzy set. In effect, we instructed the system to ignore these small deviations in the measurements and continue processing as we had before.

∞ ***That's all folks!***

Although any serious control systems engineer would have problems using something called "fuzzy logic," we have seen in this thesis how it is something that the engineering community must take seriously to move into the future.

What we learned

Although we haven't changed our overall system design goal (in the inverted pendulum case, to design a system that will bring the pendulum to the target position as quickly as possible and stabilize it with as little overshoot as possible), we have redefined the methods for achieving this goal. Instead of creating a complex mathematical equation to relate the system inputs to the outputs, we make a list of rules that we would use to control the system if we had to do it manually. Instead of describing the system with precise mathematical terms, we use simple linguistic terms that make sense to us, not the computer. Once we know how we would control the system, we convert our common sense description of the system into terms a standard computer can understand.

Fuzzy sets provide this conversion. For each "description" we use, we create a fuzzy set. By analyzing what we mean by each description, we can then define each membership function; for each "element" we consider the possibility that it "is" the description, or the degree of confidence in its membership in an "equivalent" conventional set.

During system operation, the fuzzy logic control system evaluates each rule. Using the fuzzy set membership functions, the weight of each rule is determined (the degree to which each rule should be executed). The system then uses an heuristic to combine the desired output from all the rules into a single system output.

By developing a fuzzy logic system to control an inverted pendulum, we have seen how we can use a very simple methodology to create a working system. By starting with the minimum possible system, we were able to "sketch out" the key features. We were then able to build upon the simple system, filling in features until we had sufficient detail such that the control system met our design goals. Figure 39 shows the performance of our system.

Overshoot

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.100°	0.094°	0.100°
	270°	1.951°	1.581°	1.596°

Settling Time

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.54s	1.17s	0.73s
	270°	0.54s	.60s	.58s

Figure 39—Performance of the Inverted pendulum controller

By expanding and contracting the membership functions (increasing and decreasing the normalization factors), we were able to shift the balance between settling time and overshoot. By changing the curves one way, the new system provided a shorter settling time; by changing the curves the other, overshoot decreased. By selectively changing the normalization factors, we can tune the system to provide just the behavior we want, whether it be fast response or small overshoot. Figure 40 lists the amount the normalization factors could be varied:

	Withing Tol.		Correct Perf.	
Item Changed	Min	Max	Min	Max
Ang normalization	45	52	5	60
ΔAng normalization	150	185	<25	>8000

Figure 40—Min and max values of the angle and speed normalization factors for both acceptable and "correct" behavior

We also have seen how we can tune the system behavior by altering the shape of the membership curves; changing the slope of sections of a membership function alters how the degree of confidence changes. By using the bell curve we saw that the angle became *zero* to quickly for the system to balance the pendulum; reversing the curve illustrated that it is just as important that the angle gradually "become" *zero*; we cannot just consider how *zero* the angle is when tuning the curve shape. Figure 41 shows how the system using backward sine membership functions:

Overshoot

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.306°	0.269°	0.305°
	270°	3.460°	3.177°	3.179°

Settling Time

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.50s	1.13s	0.69s
	270°	0.52s	0.57s	0.55s

Figure 41—Performance of the controller using the backward sine membership functions

When doing the original design of the system, we blindly created rules to cover every situation; we did not consider if the system would need a particular rule. By removing rules from the system, we illustrated that the rule grid does not have to be "complete" for the system to operate. When creating the list of rules, we need only to list specific rules to set specific behavior, although we can always add more rules to help a "crucial" rule (one that, with out it, the system would lack a particular behavior, such as stopping at the target angle with minimal overshoot). Although the stripped-down rule set didn't settle as quickly as the complete set, the system designer has another option when considering the physical implementation; the simple set can be implemented on simpler and

cheaper hardware, and be executed in fewer CPU cycles. Figure 42 shows the performance of the reduced-rules controller:

Overshoot

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	0.133°	0.099°	0.112°
	270°	0.153°	0.157°	0.156°

Settling Time

		Starting speed		
		-100°/s	0°/s	100°/s
Start Ang	0°	1.54s	1.17s	0.72s
	270°	1.11s	1.13s	1.12s

Figure 42—performance of reduced-rules controller

We also explored how the fuzzy logic control system is affected by a non-ideal environment. Despite large changes in gravity and torque output of the motor, the system still performed within our design goals; we implicitly designed this into the system by leaving details like the value of gravity and motor torque out of the design process. The only reason that limits exist for these values is that they must work together. Gravity and motor torque must "balance". A motor that is too strong will cause the pendulum to spin out of control; a motor that is too weak will not be able to overcome gravity when attempting to lift the pendulum to the target position. Figure 43 lists the limits of gravity and motor torque tolerated by the controller:

Item Changed	Min	Max
Acceleration due to gravity (ft/s^2)	16	33
Torque output of motor (oz in/A)	98	120

Figure 43—Min and max values of gravity and motor torque

In addition to how changes in the environment affect the performance of the system, Pendulum illustrated the effects of noise in the measurement process. Although noise was only modeled in a part of the system, we saw that the

uncertainty in the angular position of the pendulum could be rather large before the system performance would become unacceptable. Again, this was implicitly designed into the system; one of the definitions of the membership function used the uncertainty in our own mind to set membership values. Figure 44 lists the maximum amounts of noise tolerated by the system:

	Max
Constant random noise (°)	12
Spikes (° @ % time)	150 @ 5%

Figure 44—Maximum noise tolerated by the controller

Applying the solution to other problems

When we designed the system to control the inverted pendulum, the actual system was only considered at a very general level. This would cause us to believe that our solution to the inverted pendulum should be able to control anything else that behaves like an inverted pendulum, like a machine-gun turret. Unfortunately, this is not the case. We were able to design the inverted pendulum controller because we were experts at it (our fuzzy logic system is a type of expert system); we are not necessarily experts at controlling other similar systems. Our system most likely will not be able to control other things; however, because of the generality of the design process, it is a good starting point. We can then use knowledge specific to the new system to modify our controller.

Is fuzzy logic for everyone?

Clearly, fuzzy logic is a very powerful tool designers can use to implement control systems and make decisions, but should we blindly throw out everything

we already knew and totally embrace fuzzy logic? Although many systems will clearly benefit by the introduction of fuzzy logic, it lacks the mathematical "precision" needed in some applications.

For example, if we wanted to move a robot arm along a very specific path, fuzzy logic would be inappropriate. It would be much easier to generate the mathematical equation that relates the position of the arm to the desired path than to "fiddle" with a fuzzy logic control system until the system matches the path. Fuzzy logic doesn't provide a method to achieve a long series of goals (reaching each critical point on the path at a particular time). In effect, we would have to create several fuzzy systems, each taking over after the previous system meets its goal. Fuzzy logic is best when the only goal is the final result (stabilizing the pendulum at the target angle, for example).

"Whoever dies with the most toys wins" sums up the real benefits of fuzzy logic. No matter what fuzzy logic may be used for, it is another tool in the toolbox; another option when trying to solve a problem. The more tools in the toolbox, the more options we have; we can then analyze the advantages and disadvantages of each tool with respect the the problem at hand and choose the best tool for the job. We should not try to know "everything" about one particular tool, such as fuzzy logic, but instead try to learn enough about a wide variety of tools such that we can use our knowledge to make wise decisions in everything we do.

Appendix: Bibliography

Abdelrahman, Mona. "Fuzzy Sensors for Fuzzy Logic." *Control Engineering*, Dec 1991, v37 n 15, pp. 50–51.

Akin, H. Levent and V. Altin. "Rule–Based Fuzzy Logic Controller for a PWR–Type Nuclear Power Plant." *IEEE Transactions on Nuclear Science*, Apr 1991, v38 n2, pp. 883–889.

Ezawa, Yoshinori, and A. Kandel. "Robust Fuzzy Inference." *International Journal of Intelligent Systems*, Mar 1991, v6, pp. 185–197.

Blonda, P. N., and others. "An experiment for the interpretation of multitemporal remotely sensed images based on a fuzzy logic approach." *International Journal of Remote Sensing*, Mar 1991, v12 n3, pp. 463–476.

Daley, S. and K. F. Gill. "Comparison of a Fuzzy Logic Controller With a P+D Control Law." *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Jun 1989, v111, pp. 128–137.

Godo, L., J. Jacas and L. Valverde. "Fuzzy Values in Fuzzy Logic." *International Journal of Intelligent Systems*, Mar 1991, v 6, pp. 199–212.

Hassan, M. A. M., O. P. Malik and G.S. Hope. "A Fuzzy Logic Based Stabilizer for a Synchronous Machine." *IEEE Transactions on Energy Conversion*, Sep 1991, v6 n3, pp. 407–413.

Infelise, Nick. "A Clear Vision of Fuzzy Logic." *Control Engineering*, Jul 1991, v38 n9, pp. 28–30.

Ishida, T. "Fuzzy Control." *Medical Electronics*, Dec 1990, v21 n6, pp. 83–87.

Kandel, Abraham. *Fuzzy Mathematical Techniques with Appications*, copyright 1986, Florida State University.

Kader, Victoria. "Japanese Companies are Incorporating 'Fuzzy Logic' In a Growing Number of Products and Services." *Business America*, Sep 24, 1990, v11 n18, p. 7.

Lee, Chuen Chien. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part I." *IEEE Transactions on Systems, Man, and Cybernetics*, Mar 1990, v20 n2, pp. 404–418.

Lee, Chuen Chien. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller—Part II." *IEEE Transactions on Systems, Man, and Cybernetics*, Mar 1990, v20 n2, pp. 419–435.

Levary, Reuven R. and C. Y. Lin. "Modelling the Software Development Process Using an Expert Simulation System Having Fuzzy Logic." *Software—Practice and Experience*, Feb 1991, v21 n2, pp. 133–148.

Linkens, D. A. and S. B. Hasnain. "Self–organising Fuzzy Logic Control and Application to Muscle Relaxant Anaesthesia." *IEE Proceedings–D*, May 1991, v138 n3, pp. 274–284.

McCusker, Tom. "Neural Networks and Fuzzy Logic, Tools of Promise for Controls." *Control Engineering*, May 1990, v37 n6, pp. 84–85.

Nafarieh, Asghar, and J. M. Keller. "A Fuzzy Logic Rule–Based Automatic Target Recognizer." *International Journal of Intelligent Systems*, Jun 1991, v6, pp. 295–312.

Novák, Vilém, and W. Pedrycz. "Fuzzy Sets and T–norms in the light of Fuzzy Logic." *International Journal of Man–Machine Studies*, Dec 1989, v29 n6, pp. 113–127.

Otto, Matthias. "Fuzzy Sets: Applications to Analytical Chemistry." *Analytical Chemistry*, Jul 15, 1990, v62 n14, pp. 797–802.

Reddy, V. C. V. Pratapa. "Hardware Implementation of Fuzzy Logic Circuits." *Z. elektr. Inform. –u. Energietechnik*, Leipzig, 1979, v9, pp. 85–90.

Sombré, Léa. "Reasoning Under Incomplete Information." *International Journal of Intellignent Systems*, Sep 1990, v5 n4, pp 394–401.

Togai Infralogic, Inc. *Strategic and Practical Use of Fuzzy Logic Technology*. August 27, 1991, Eastman Kodak Co, Rochester, New York.

Wakileh, B. A. M., and K. F. Gill. "Robot Control Using Self–Organising Fuzzy Logic." *Computers in Industry*, Nov 1990, v15 n3, pp. 175–186.

Waller, Larry. "Fuzzy Logic: It's Comprehensible, It's Practical—And It's Commercial." *Electronics*, Mar 1989, v62 n3, pp. 102–103.

Williams, Tom. "Fuzzy Logic Simplifies Complex Control Problems." *Computer Design*, Mar 1991, v30 n5, pp. 90–105.

Yamakawa, Takeshi. "Stabilization of an Inverted Pendulum by a High–Speed Fuzzy Logic Controller Hardware System." *Fuzzy Sets and Systems*, n32, pp. 161–180.

Zadeh, Lotfi A. "Knowledge Representation in Fuzzy Logic." *IEEE Transactions on Knowledge and Data Engineering*, Mar 1989, v1 n1, pp. 89–100.

Zadeh, Lotfi A. "The Birth and Evolution of Fuzzy Logic." *International Journal of General Systems*, 1991, v17, pp. 95-105.