Chapter 5

Rule-Based Control

Chapter Contents

5.1 Fuzzy Control 155
5.1.1 Choosing Fuzzy Controller Inputs and Outputs 155
5.1.2 Putting Control Knowledge into Rule Bases 157
5.1.3 Fuzzy Quantification of Knowledge 163
5.1.4 Matching: Determining Which Rules to Use 171
5.1.5 Inference Step: Determining Conclusions 175
5.1.6 Converting Decisions into Actions 178
5.2 General Fuzzy Systems 184
5.2.1 Multiple Input Multiple Output Fuzzy Systems 184
5.2.2 Takagi-Sugeno Fuzzy Systems 186
5.2.3 Mathematical Representations of Fuzzy Systems 187
5.2.4 Relationships Between Neural and Fuzzy Systems 193
5.3 Design Example: Fuzzy Control for Tanker Ship Steering 194
5.3.1 Simulation of a Fuzzy Controller 194
5.3.2 Fuzzy Controller Tuning for the Tanker Ship 201
5.3.3 Design Concerns 205
5.4 Stability Analysis 212
5.4.1 Example: Stability and Limit Cycles in Ship Steering 213
5.4.2 Discussion: Lyapunov Stability Analysis of Fuzzy Control Systems . . . 214
5.5 Expert Control 214
5.5.1 General Knowledge Representations 215
5.5.2 General Inference Mechanisms 216
5.5.3 Stability Analysis of Expert Control Systems 216
5.6 Hierarchical Rule-Based Control Systems 217
5.7 Exercises and Design Problems 217

One relatively complex task that humans can perform is a feedback control task. For instance, driving an automobile is a control task that humans regularly perform. There, the driver senses lane markers, vehicles, obstacles, and other cues to control the direction of travel by steering, and velocity via actuating the throttle and brakes. Humans perform many other control tasks when employed at, for example, chemical processing plants, manufacturing facilities, and in vehicular applications. In this chapter, we study rule-based control by studying the use of fuzzy and expert systems for control. These are probably the most popular intelligent control methods for automating feedback control tasks that have often been performed by humans in the past. The biomimicry here should be thought of as "human-mimicry" as it was explained in Part I, but clearly an accurate model of human reasoning and decision-making processes is neither sought, nor obtained.

The design example for the tanker ship serves to illustrate the heuristic nonlinear control design methodology that fuzzy and expert control allows. Here, there is a particularly important focus on design methodology for fuzzy controllers, as there have been certain problems in the literature with proper design methodology. We discuss effects of disturbances, noise, plant changes, stability, and limit cycles. It is emphasized that sound control engineering methodology, as outlined in Part I, should not be ignored.

5.1 Fuzzy Control

A block diagram of a fuzzy control system is shown in Figure 5.1. The fuzzy controller is composed of the following four elements:

1. *A rule base* (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.

2. *An inference mechanism* (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision-making in interpreting and applying knowledge about how best to control the plant.

3. *A fuzzification interface,* which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.

4. *A defuzzification interface,* which converts the conclusions of the inference mechanism into actual inputs for the process.

We introduce each of the components of the fuzzy controller for the simple problem of tanker ship heading regulation, as was shown in Figure 4.8.

5.1.1 Choosing Fuzzy Controller Inputs and Outputs

Consider a human-in-the-loop whose responsibility is to control the tanker ship (i.e., the ship captain), as shown in Figure 5.2. The fuzzy controller is to be designed to automate how a captain would control the system. First, the captain

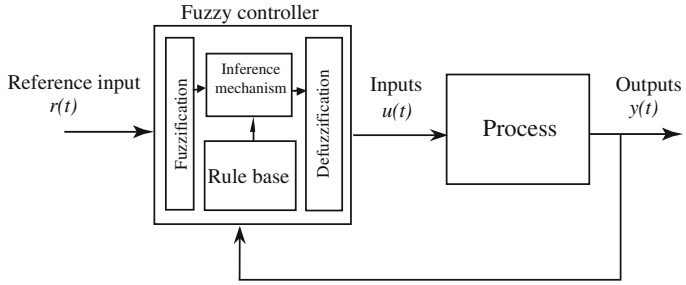

Figure 5.1: Fuzzy controller.

tells us (the designers of the fuzzy controller) what information she or he will use as inputs to the decision-making process. Suppose that for the tanker ship, the expert (this could be you, if you do not have the captain available) says that she or he will use

$$e(t) = \psi_r(t) - \psi(t)$$

and

$$\frac{de(t)}{dt} = \dot{e}(t)$$

as the variables on which to base decisions. Certainly, there are many other choices (e.g., the integral of the error e could also be used) but this choice makes good intuitive sense. Next, we must identify the controlled variable. For the tanker ship, it is assumed that we are only allowed to control the rudder so the input is δ (i.e., we do not consider the use of the ship speed for helping with steering).

Fuzzy controller input/output choice depends on what variables the expert uses and broadly affects the design of the controller.

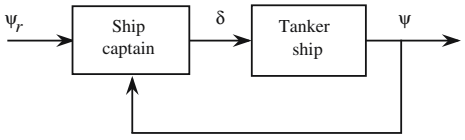

Figure 5.2: Human controlling a tanker ship.

For more complex applications, the choice of the inputs to the controller and outputs of the controller (inputs to the plant) can be more difficult. Essentially, you want to make sure that the controller will have the proper information available to be able to make good decisions and have proper control inputs to be able to move the system in the directions needed to be able to achieve high-performance operation. Practically speaking, access to information and the ability to effectively control the system often cost money. If the designer believes that proper information is not available for making control decisions, he or she may have to invest in another sensor that can provide a measurement of another system variable. Alternatively, the designer may implement some filtering or other processing of the plant outputs. In addition, if the designer

determines that the current actuators will not allow for the precise control of the process, he or she may need to invest in designing and implementing an actuator that can properly affect the process. Hence, while in some academic problems you may be given the plant inputs and outputs, in many practical situations you may have some flexibility in their choice. These choices affect what information is available for making online decisions about the control of a process and hence affect how we design a fuzzy controller. Once the fuzzy controller inputs and outputs are chosen, you must determine the reference inputs. For the tanker ship, we will simply use step changes in ship heading. In general, the specification and generation of the reference input(s) can be more challenging.

After all the inputs and outputs are defined for the fuzzy controller, we can specify the fuzzy control system. The fuzzy control system for the tanker ship, with our choice of inputs and outputs, is shown in Figure 5.3. Now, *within this framework* we seek to obtain a description of how to control the process. We see then that the choice of the inputs and outputs of the controller places certain constraints on the remainder of the fuzzy control design process. If the proper information is not provided to the fuzzy controller, there will be little hope for being able to design a good rule base or inference mechanism. Moreover, even if the proper information is available to make control decisions, this will be of little use if the controller is not able to properly affect the process variables via the process inputs. It must be understood that the choice of the controller inputs and outputs is a fundamentally important part of the control design process for many practical applications.

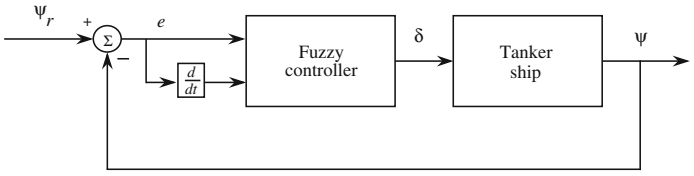

Figure 5.3: Fuzzy controller for a tanker ship steering problem.

5.1.2 Putting Control Knowledge into Rule Bases

Suppose that the human expert (captain) shown in Figure 5.2 provides a description of how best to control the plant in some natural language (e.g., English). Next, we characterize the expert's description with "linguistics."

Linguistic Descriptions

The linguistic description provided by the expert can generally be broken into several parts. There will be "linguistic variables" that describe each of the time-varying fuzzy controller inputs and outputs. For the tanker ship,

"error" describes $e(t)$
"change-in-error" describes $\frac{de(t)}{dt}$
"rudder-input" describes $\delta(t)$

Note that we use quotes to emphasize that certain words or phrases are linguistic descriptions, but emphasize that these variables do change over time. There are many possible choices for the linguistic descriptions for variables. Some designers like to choose them so that they are quite descriptive for documentation purposes. However, this can sometimes lead to long descriptions. Others seek to keep the linguistic descriptions as short as possible (e.g., using "$e(t)$" as the linguistic variable for $e(t)$), yet accurate enough so that they adequately represent the variables. Regardless, the choice of the linguistic variable has no effect on the way that the fuzzy controller operates; it is simply a notation that helps to facilitate the construction of the fuzzy controller via fuzzy logic.

Linguistic variables represent the key variables that the expert uses to make decisions.

Just as $e(t)$ takes on a value of, for example, 0.1 at $t = 2$ ($e(2) = 0.1$), linguistic variables assume "linguistic values." That is, the values that linguistic variables take on over time change dynamically. Suppose for the tanker ship example that "error," "change-in-error," and "rudder-input" take on the following values:

"neghuge"
"neglarge"
"negbig"
"negmed"
"negsmall"
"zero"
"possmall"
"posmed"
"posbig"
"poslarge"
"poshuge"

Note that we are using "negsmall" as an abbreviation for "negative small in size" and so on for the other variables. Such abbreviations help keep the linguistic descriptions short yet precise. For an even shorter description we could use integers:

"−5" to represent "neghuge"
"−4" to represent "neglarge"
"−3" to represent "negbig"
"−2" to represent "negmed"
"−1" to represent "negsmall"
"0" to represent "zero"
"1" to represent "possmall"
"2" to represent "posmed"
"3" to represent "posbig"
"4" to represent "poslarge"
"5" to represent "poshuge"

This is a particularly appealing choice for the linguistic values since the descriptions are short and nicely represent that the variable we are concerned with has a numeric quality. We are not, for example, associating "−1" with any particular number of radians of error; the use of the numbers for linguistic descriptions simply quantifies the sign of the error (in the usual way) and indicates the size in relation to the other linguistic values. We shall find the use of this type of linguistic value quite convenient when it comes to writing computer programs to simulate or implement fuzzy control systems and hence will give it the special name, "linguistic-numeric value."

The linguistic variables and values provide a language for the expert to express her or his ideas about the control decision-making process, in the context of the framework established by our choice of fuzzy controller inputs and outputs. Suppose that for the tanker ship $\psi_r(t) = 45$ deg. ($\psi_r(t) = \frac{45\pi}{180}$ rad.) and $e = r - y$ so that

$$e = \frac{45\pi}{180} - \psi$$

and

$$\frac{de}{dt} = -\frac{d\psi}{dt}$$

since $\frac{d\psi_r}{dt} = 0$. First, we will study how we can quantify certain dynamic behaviors with linguistics. In the next subsection we will study how to quantify knowledge about how to control the tanker ship using linguistic rules.

For the tanker ship, each of the following statements quantifies a different configuration of the ship (refer back to Figure 4.8 on page 117):

Linguistic statements characterize the status of the plant.

- The statement "error is poslarge" can represent the situation where the ship heading is at a significant angle counterclockwise to where it should be heading.
- The statement "error is negsmall" can represent the situation where the ship heading is just slightly clockwise of where it should be heading, but not too close to the reference heading ψ_r to justify quantifying it as "zero" and not too far away to justify quantifying it as "negmed."
- The statement "error is zero" can represent the situation where the ship heading is very near the desired heading (a linguistic quantification is not precise, hence we are willing to accept any value of the error around $e(t) = 0$ as being quantified linguistically by "zero" since this can be considered a better quantification than "possmall" or "negsmall").
- The statement "error is poslarge **and** change-in-error is possmall" can represent the situation where the ship heading is counterclockwise to where it should be and, since $\frac{d\psi}{dt} < 0$, the ship heading is moving *away* from the desired heading (note that in this case, the ship is moving counterclockwise).
- The statement "error is negsmall **and** change-in-error is possmall" can represent the situation where the ship heading is slightly clockwise of

where it should be heading and, since $\frac{d\psi}{dt} < 0$, the ship heading is moving *toward* the desired heading (note that in this case, the ship is moving counterclockwise).

It is important for the reader to study each of the cases above to understand how the expert's linguistics quantify the current situation the ship is in (actually, each partially quantifies the ship's state).

Overall, we see that to quantify the dynamics of the process, we need to have a good understanding of the physics of the underlying process we are trying to control. While for the ship steering problem, the task of coming to a good understanding of the dynamics is relatively easy, this is not the case for many physical processes. Quantifying the process dynamics with linguistics is not always easy, and certainly a better understanding of the process dynamics generally leads to a better linguistic quantification. Often, this will naturally lead to a better fuzzy controller *provided* that you can adequately measure the system dynamics so that the fuzzy controller can make the right decisions at the proper time.

Rules

Next, we will use the above linguistic quantification to specify a set of rules (a rule base) that captures the expert's knowledge about how to control the plant. In particular, for the tanker ship in the three positions shown in Figure 5.4, we have the following rules (notice that we drop the quotes since the whole rule is linguistic):

Linguistic rules represent a description of the rules that the expert uses in control.

1. **If** error is negsmall **and** change-in-error is negsmall **Then** rudder-input is posmed

   This rule quantifies the situation in Figure 5.4(a) where the ship has a heading angle that is clockwise of the desired heading and is moving clockwise; hence, it is clear that we should apply a medium positive rudder angle so that we can get the ship moving in the proper direction.

2. **If** error is zero **and** change-in-error is possmall **Then** rudder-input is negsmall

   This rule quantifies the situation in Figure 5.4(b) where the ship is nearly moving in the proper direction (a linguistic quantification of zero does not imply that $e(t) = 0$ exactly) and is moving counterclockwise; hence, we should apply a small negative rudder angle to counteract the movement so that it moves toward zero (a positive rudder angle could result in the ship heading overshooting the desired angle).

3. **If** error is possmall **and** change-in-error is negsmall **Then** rudder-input is zero

   This rule quantifies the situation in Figure 5.4(c) where the ship is counterclockwise of the desired heading and is moving clockwise; hence, we

apply a near zero rudder angle since the ship is already moving in the proper direction.

Specification of increasingly good rules generally requires increasingly good insights into the physics of the plant.

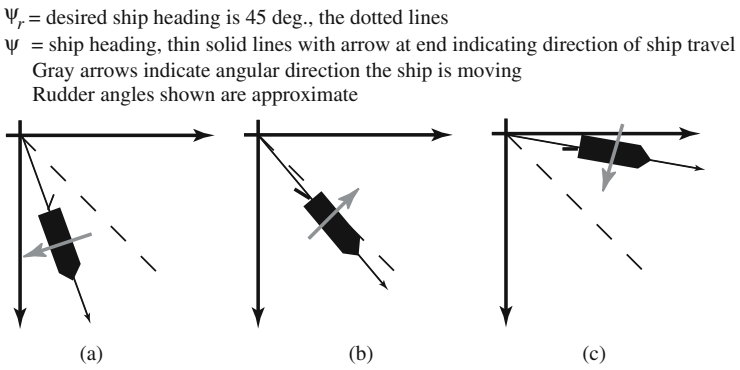

Figure 5.4: Tanker ship in various positions.

Each of the three rules listed above is a "linguistic rule" since it is formed solely from linguistic variables and values. Since linguistic values are not precise representations of the underlying quantities that they describe, linguistic rules are not precise either. They are simply abstract ideas about how to achieve good control that could mean somewhat different things to different people. They are, however, at a level of abstraction that humans are often comfortable with in terms of specifying how to control a process.

The general form of the linguistic rules listed above is

If premise **Then** consequent

As you can see from the three rules listed above, the premises (which are sometimes called "antecedents") are associated with the fuzzy controller inputs and are on the left-hand side of the rules. The consequents (sometimes called "actions") are associated with the fuzzy controller outputs and are on the right-hand side of the rules. Notice that each premise (or consequent) can be composed of the conjunction of several "terms" (e.g., in rule 3 above, "error is possmall **and** change-in-error is negsmall" is a premise that is the conjunction of two terms). The number of fuzzy controller inputs and outputs places an upper limit on the number of elements in the premises and consequents. Note that there does not need to be a premise (consequent) term for each input (output) in each rule, although often there is.

Rule Bases

Using the above approach, we could continue to write down rules for the ship steering problem for all possible cases (the reader should do this for practice, at least for a few more rules). Note that since we only specify a finite number of linguistic variables and linguistic values, there is only a finite number of possible

rules. For the ship steering problem, with two inputs and eleven linguistic values for each of these, there are at most $11^2 = 121$ possible rules (all possible combinations of premise linguistic values for two inputs).

A convenient way to list all possible rules for the case where there are not too many inputs to the fuzzy controller (less than or equal to two or three) is to use a tabular representation. A tabular representation of one possible set of rules for the fuzzy controller for the ship is shown in Table 5.1. Notice that the body of the table lists the linguistic-numeric consequents of the rules, and the left column and top row of the table contain the linguistic-numeric premise terms. Then, for instance, the $(+1, -1)$ position (where the "+1" represents the row having "+1" for a numeric-linguistic value and the "−1" represents the column having "−1" for a numeric-linguistic value) has a 0 ("zero") in the body of the table and represents the rule

If error is possmall **and** change-in-error is negsmall **Then** rudder-input is zero

which is rule 3 above. Table 5.1 represents abstract knowledge that the expert has about how to control the tanker ship given the error and its derivative as inputs.

Table 5.1: Rule Table for the Tanker Ship

δ		$\dot{e}$										
		−5	−4	−3	−2	−1	0	1	2	3	4	5
e	−5	5	5	5	5	5	5	4	3	2	1	0
	−4	5	5	5	5	5	4	3	2	1	0	−1
	−3	5	5	5	5	4	3	2	1	0	−1	−2
	−2	5	5	5	4	3	2	1	0	−1	−2	−3
	−1	5	5	4	3	2	1	0	−1	−2	−3	−4
	0	5	4	3	2	1	0	−1	−2	−3	−4	−5
	1	4	3	2	1	0	−1	−2	−3	−4	−5	−5
	2	3	2	1	0	−1	−2	−3	−4	−5	−5	−5
	3	2	1	0	−1	−2	−3	−4	−5	−5	−5	−5
	4	1	0	−1	−2	−3	−4	−5	−5	−5	−5	−5
	5	0	−1	−2	−3	−4	−5	−5	−5	−5	−5	−5

Note that the other rules are also valid and take special note of the pattern of rule consequents that appears in the body of the table. Notice the diagonal of zeros. Viewing the body of the table as a matrix, we see that it has a certain symmetry to it. This symmetry that emerges when the rules are tabulated is no accident and is actually a representation of abstract knowledge about how to control the ship heading; it arises due to a symmetry in the system's dynamics. Similar patterns will often be found when constructing rule bases for other applications.

5.1.3 Fuzzy Quantification of Knowledge

Up to this point we have only quantified, in an abstract way, the knowledge that the human expert has about how to control the plant. Next, we will show how to use fuzzy logic to fully quantify the meaning of linguistic descriptions so that we may automate, in the fuzzy controller, the control rules specified by the expert.

Membership Functions

First, we quantify the meaning of the linguistic values using "membership functions." Consider, for example, Figure 5.5. This is a plot of a function μ versus $e(t)$ that takes on special meaning. The function μ quantifies the *certainty* that $e(t)$ can be classified linguistically as "possmall." In our discussion in this chapter, do not confuse the term "certainty" with "probability" or "likelihood." The membership function is not a probability density function, and there is no underlying probability space. By "certainty" we mean "degree of truth." The membership function does not quantify random behavior; it simply makes more accurate (less fuzzy) the meaning of linguistic descriptions.

To understand the way that a membership function works, it is best to perform a case analysis where we show how to interpret it for various values of $e(t)$:

Membership functions numerically quantify the meaning of linguistic statements by the expert.

- If $e(t) = -\frac{4\pi}{10}$, then $\mu(-\frac{4\pi}{10}) = 0$, indicating that we are certain that $e(t) = -\frac{4\pi}{10}$ is *not* "possmall" (indeed, it is negative).
- If $e(t) = \frac{2\pi}{20}$, then $\mu(\frac{2\pi}{20}) = 0.5$, indicating that we are halfway certain that $e(t) = \frac{2\pi}{20}$ is "possmall" (we are only halfway certain since it could also be "zero" with some degree of certainty—this value is in a "gray area" in terms of linguistic interpretation).
- If $e(t) = \frac{2\pi}{10}$, then $\mu(\frac{2\pi}{10}) = 1.0$, indicating that we are absolutely certain that $e(t) = \frac{2\pi}{10}$ is what we mean by "possmall."
- If $e(t) = \frac{8\pi}{10}$, then $\mu(\frac{8\pi}{10}) = 0$, indicating that we are certain that $e(t) = \frac{8\pi}{10}$ is not "possmall" (actually, we will soon see that we will quantify it as "poslarge").

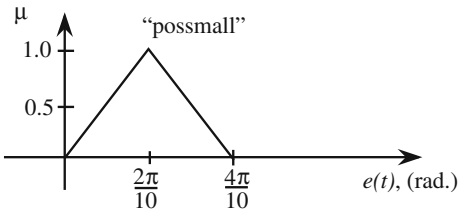

Figure 5.5: Membership function for linguistic value "possmall."

The membership function quantifies, in a continuous manner, whether values of $e(t)$ belong to (are members of) the set of values that are "possmall," and hence it quantifies the meaning of the linguistic statement "error is possmall." This is why it is called a membership function. It is important to recognize that the membership function in Figure 5.5 is only one possible definition of the meaning of "error is possmall;" you could use a bell-shaped function, a trapezoid, or many others, depending on what the expert means by "possmall."

For instance, consider the membership functions shown in Figure 5.6. For some applications someone may be able to argue that we are absolutely certain that any value of $e(t)$ near $\frac{2\pi}{10}$ is still "possmall" and only when you get sufficiently far from $\frac{2\pi}{10}$ do we lose our confidence that it is "possmall." One way to characterize this understanding of the meaning of "possmall" is via the trapezoid-shaped membership function in Figure 5.6(a). For other applications, you may think of membership in the set of "possmall" values as being dictated by the Gaussian-shaped membership function (not to be confused with the Gaussian probability density function) shown in Figure 5.6(b). For still other applications, you may not readily accept values far away from $\frac{2\pi}{10}$ as being "possmall," so you may use the membership function in Figure 5.6(c) to represent this. Finally, while we often think of symmetric characterizations of the meaning of linguistic values, we are not restricted to these symmetric representations. For instance, in Figure 5.6(d) we represent that we believe that as $e(t)$ moves to the left of $\frac{2\pi}{10}$, we are very quick to reduce our confidence that it is "possmall," but if we move to the right of $\frac{2\pi}{10}$, our confidence that $e(t)$ is "possmall" diminishes at a slower rate.

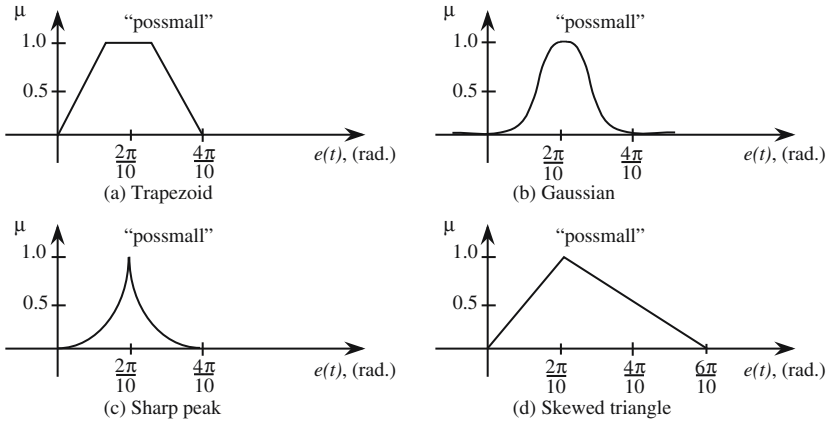

Figure 5.6: Some example membership function choices for representing "error is possmall."

Each of the membership functions in Figure 5.6 has a mathematical representation and these are useful in simulation and implementation of fuzzy controllers. For example, interval checking plus equations for lines can be used to implement the membership function in Figure 5.6(a) and a Gaussian function

with an appropriate center, spread, and scale factor can implement the one in Figure 5.6(b).

In summary, we see that depending on the application and the designer (expert), many different choices of membership functions are possible. It is important to note here, however, that for the most part, the definition of a membership function is subjective rather than objective. That is, we simply quantify it in a manner that makes sense to us, but others may quantify it in a different manner.

The set of values that is described by μ as being "positive small" is called a "fuzzy set." Let A denote this fuzzy set. Notice that from Figure 5.5 we are absolutely certain that $e(t) = \frac{2\pi}{10}$ is an element of A, but we are less certain that $e(t) = \frac{2\pi}{40}$ is an element of A. Membership in the set, as specified by the membership function, is fuzzy; hence we use the term "fuzzy set." A "crisp" (as contrasted to "fuzzy") quantification of "possmall" can also be specified, but via the membership function shown in Figure 5.7. This membership function is simply an alternative representation for the interval on the real line $\frac{2\pi}{20} \leq e(t) \leq \frac{6\pi}{20}$, and it indicates that this interval of numbers represents "possmall." Clearly, this characterization of crisp sets is simply another way to represent a normal interval (set) of real numbers.

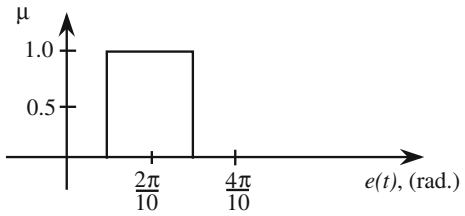

Figure 5.7: Membership function for a crisp set.

While the vertical axis in Figure 5.5 represents certainty, the horizontal axis is also given a special name. It is called the "universe of discourse" for the input $e(t)$ since it provides the range of values of $e(t)$ that can be quantified with linguistics and fuzzy sets. In conventional terminology, a universe of discourse for an input or output of a fuzzy system is simply the range of values the inputs and outputs can take on.

Now that we know how to specify the meaning of a linguistic value via a membership function (and hence a fuzzy set), we can easily specify the membership functions for all 33 linguistic values (eleven for each input and eleven for the output) of our ship steering example. See Figure 5.8 for one choice of membership functions.

For our later convenience, we list both the linguistic and linguistic-numeric values associated with each membership function. Hence, we see that the membership function in Figure 5.5 for "possmall" on the "error" universe of discourse is embedded among several others that describe other sizes of values (so that, for instance, the membership function to the right of the one for "possmall" is the

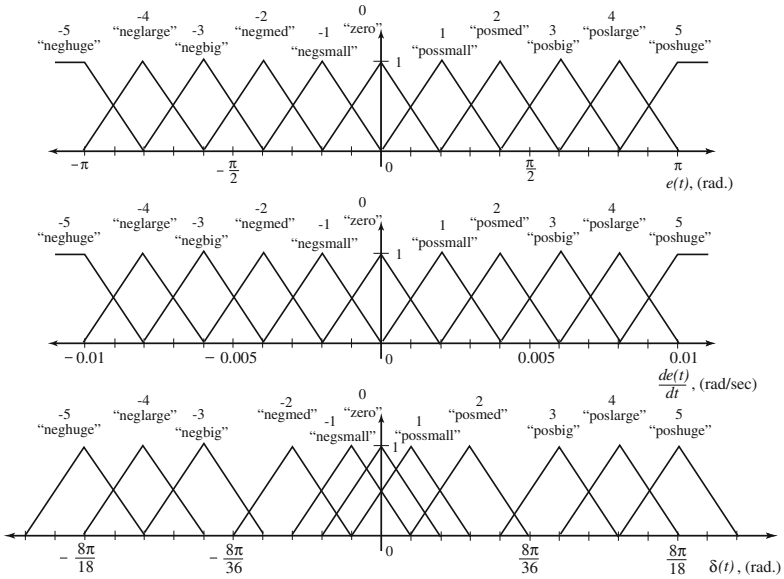

Figure 5.8: Membership functions for a ship steering example.

one that represents "error is posmed"). Note that other similarly shaped membership functions make sense (e.g., bell-shaped membership functions). The scale for the axis was chosen since $|e(t)|$ can be at most π radians since it is the difference between two angles and we use this fact to help define the relative sizes in our membership quantification of our linguistics. Notice that for the $\dot{e}$ universe of discourse, we use a set of membership functions similar to the ones on the e universe of discourse, but that the scale of the $\dot{e}$ axis is different. This scale was chosen since our expert captain felt that $|\dot{e}(t)| \geq 0.01$ radians per second (0.57 degrees per second) was a fast change in the ship heading. Notice then that the meaning of the linguistics on the $\dot{e}$ universe of discourse is different from those on the e universe of discourse.

The membership functions at the outer edges of the e and $\dot{e}$ universes of discourse in Figure 5.8 deserve special attention. For the inputs $e(t)$ and $\dot{e}$, we see that the outermost membership functions "saturate" at a value of one. This makes intuitive sense, as at some point the human expert would just group all large values together in a linguistic description such as "poshuge" (or "neghuge"). The membership functions at the outermost edges appropriately characterize this phenomenon since they characterize "greater than" (for the right side) and "less than" (for the left side). Study Figure 5.8 and convince yourself of this.

It is important to have a clear picture in your mind of how the values of the membership functions change as, for example, $e(t)$ changes its value over time. For instance, as $e(t)$ changes from $-\pi$ to π, we see that various membership functions will take on zero and nonzero values indicating the degree to which the

corresponding linguistic value appropriately describes the current value of $e(t)$. For example, at $e(t) = -\pi$ we are certain that the error is "neghuge," and as the value of $e(t)$ moves toward $-8\pi/10$, we become less certain that it is "neghuge" and more certain that it is "neglarge." We see that the membership functions quantify the meaning of linguistic statements that describe time-varying signals.

Finally, note that often we will draw all the membership functions for one input or output variable on one graph; hence, we often omit the label for the vertical axis with the understanding that the plotted functions are membership functions describing the meaning of their associated linguistic values. Also, we will use the notation μ_{zero} to represent the membership function associated with the linguistic value "zero" and a similar notation for the others.

Next, consider the choice of membership functions for the "rudder-input" $\delta(t)$ universe of discourse in Figure 5.8. The horizontal scale was chosen since, as you may recall, the rudder input can only be moved between ± 80 degrees. Converting to radians, this means that it moves between $\pm 8\pi/18$ radians and this gives us the center values for the membership functions on the outer edges for the δ universe of discourse. Next, note that for the output δ, the membership functions at the outermost edges cannot be saturated for the fuzzy system to be properly defined (more details on this point will be provided at the end of Section 5.1.6 that starts on page 178). The basic reason for this is that in decision-making processes of the type we study, we seek to take actions that specify an exact value for the process input. We do not generally indicate to a process actuator, "any value bigger than, say, $\pm 8\pi/18$, is acceptable."

The Meaning of Membership Functions and Rules

Notice that the pattern of center positions (i.e., where the triangles peak at one) for the output membership functions in Figure 5.8 is not uniform as it is for the input universes of discourse. A uniform distribution (which with proper tuning can work for this ship steering example also) would imply that the captain would roughly make the rudder angle proportional to the error between the heading and desired heading, and the change in the heading error (except when the error and change in error are too big in magnitude, then she or he will simply move the rudder to its maximum deflection). To get a uniform distribution of output membership function centers you can choose the center values, which we denote by b_i where i is the linguistic-numeric index for the corresponding membership function, as

$$b_i = \frac{8\pi}{18}\left(\frac{i}{5}\right)$$

where $i = -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$ (and then use the same base widths and rule base). For practice, draw the resulting membership functions on the output universe of discourse.

The choice of having nonuniformly distributed membership functions in Figure 5.8 represents that for small heading errors, when the change in error is small also, the captain will not put in as big a rudder angle. Why? Through experience the captain has found that if the heading is close to where it should be

and it is not moving away from where it should be fast, then small corrections are more effective in heading regulation. While the captain may have learned this through experience (or training), the basic reason for this arises from the physics of the process. For example, since the heading sensor measurement is noisy, for small heading errors this noise can have a relatively large impact on the error so that for small errors the noise can make it appear that there is a heading error when there is not. Now, for the case where, say, the error is "poshuge" but the change in error $\dot{e}$ is "neghuge" the captain also puts in a zero rudder angle input (see Table 5.1). The expert captain specified this rule since, while the heading is far from where it should be, the heading is moving very fast to correct this condition. If the error is "poshuge" and the change in error $\dot{e}$ is "neglarge," then the rudder input is "negsmall" and by Figure 5.8 this is only a small correction since the captain does not want to expend more rudder movement than necessary; the ship is moving to correct its own error in heading, why expend control energy (and wear out the rudder actuator) trying to do something that is already in the process of happening?

It is important to gain insights into the fuzzy logic quantification of the rule base to clearly understand what control expertise is being implemented by the fuzzy controller.

Next, note that from the pattern of output membership functions in the body of Table 5.1, we see that the captain will saturate the rudder either positive (upper left corner of the rule base) or negative (lower right corner of the rule base) if the error and change in error are too big in magnitude. The choice of when to saturate the rudder (i.e., to move it to its maximum deflection) is made through the captain's experience in heading regulation. If she or he is not willing to saturate it soon enough, larger heading deviations may occur. However, if the captain is too quick to saturate the rudder input, for example, even for relatively small errors, she or he will wear out the rudder actuator faster and will be continually moving the rudder.

With this discussion, it is important to note that the meaning of the linguistic rule base is not clear until the membership functions for the linguistic variables are all defined. The membership function definitions fully specify the meaning of the linguistics. Note that while on the e and $\dot{e}$ universes of discourse the meaning of the linguistics is similar, it is different by a scale factor on the horizontal axes (scaling the horizontal axis changes the meaning of the linguistics). Moreover, the meaning of the linguistics on the output universe of discourse is quite different from meaning of the linguistics on the input universes of discourse (e.g., for the membership functions at the outermost edges and in the nonuniform spacing of the output membership function centers).

Due to the lack of clarity of the meaning of control rules in the linguistic rule base shown in Table 5.1, schemes are often used which include membership function information in the rule base table. While many schemes are possible, a common one is shown in Table 5.2 where rather than listing the indices for the output membership functions, the centers of the appropriate output membership functions are listed, up to a scale factor, which in this case is $8\pi/18$ (i.e., to get the actual center from the rule base table you take the entry and multiply it by $8\pi/18$).

Coupled with our understanding of the meaning of the linguistic-numeric indices for the error and change in error, all the major components of the captain's

Table 5.2: Rule Table for the Tanker Ship (body of table holds the output membership function centers where each element should be multiplied by $8\pi/18$).

		$\dot{e}$										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
e	-5	1	1	1	1	1	1	.8	.6	.3	.1	0
	-4	1	1	1	1	1	.8	.6	.3	.1	0	$-.1$
	-3	1	1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$
	-2	1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$
	-1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$
	0	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1
	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1
	2	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1
	3	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1
	4	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1	-1
	5	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1	-1	-1

knowledge of ship steering are directly evident from Table 5.2 in the following manner:

One good way to gain insights is to characterize the abstract patterns that often emerge when a rule base is constructed.

1. If the heading error and change in error are both too big (upper left and lower right corners of the rule base shown in Table 5.2), then use the appropriate maximum rudder input.

2. For zero e and $\dot{e}$, the rudder angle should be zero, but if e and $\dot{e}$ move positive, then the rudder should move negative (where if $\dot{e}$ moves significantly positive, then the rudder should move even more negative). Similar reasoning is used for e and $\dot{e}$ negative, where we then make the rudder angle positive. For the case where e and $\dot{e}$ have opposite signs and depending on the magnitude of the signals, we will make the rudder input either positive or negative.

3. For small e and $\dot{e}$, be conservative in making changes to the rudder position since such corrections may cause heading deviations instead (i.e., lower the "gain" of the controller near zero so that noise is not amplified). Also, if the ship's angular position is moving sufficiently fast to remove the heading error, then be conservative in using the rudder to help move it since this can require unnecessary control energy.

This provides a summary of the captain's knowledge about ship steering. The above three points can be thought of as "meta-rules," that is, abstract representations of control rules. Some designers use rules that are more abstract in the sense that they describe what control actions should occur whenever $e(t)$ and $\dot{e}(t)$ lie in a certain region. For example, the rule

If (error is neghuge **and** change-in-error is neghuge)
or (error is neghuge **and** change-in-error is neglarge)
or (error is neglarge **and** change-in-error is neghuge)
Then rudder-input is poshuge

represents what action should be taken if $e(t)$ and $\dot{e}(t)$ take on values such that any of the three rules in the upper left corner of the rule base in Table 5.2 are on. In this sense, the above rule represents three of the rules of the form discussed earlier. It achieves this apparently more compact representation via the use of the "disjunction" (or) in the premise of the above rule. If you were to use the above rule in the implementation you would use "maximum" to represent the disjunction; however, if you are concerned with implementation complexity, you must be careful to determine whether this approach is more or less complex than simply treating each rule separately.

Returning to our discussion on the tanker ship, we must emphasize that it is important in rule base construction that the control system designer can clearly list the expertise that is represented in the rule base. Lack of a clear understanding of the rule base is an indication that there is likely to be a later problem in simulation or implementation. Fuzzy control is not a methodology where you can haphazardly construct a rule base and expect in all cases for it to work well; you must put good control knowledge in to get good closed-loop system performance (it is not very often that you can get lucky and get good performance from a poorly constructed rule base).

In summary, the rule base of the fuzzy controller holds the linguistic variables, linguistic values, their associated membership functions, and the set of all linguistic rules (shown in Table 5.1 on page 162), so we have completed the description of the rule base for the ship steering problem. Next we describe the fuzzification process.

Fuzzification

It is actually the case that for most fuzzy controllers the fuzzification block in Figure 5.1 on page 156 can be ignored since this process is so simple. The reader should simply think of the fuzzification process as the act of obtaining a value of an input variable (e.g., $e(t)$) and finding the numeric values of the membership function(s) that are defined for that variable. For example, if $e(t) = 2\pi/10$ and $\dot{e}(t) = 0.001$, the fuzzification process amounts to finding the values of the input membership functions for these. In this case

$$\mu_{possmall}(e(t)) = 1$$

(with all others zero) and

$$\mu_{zero}\left(\dot{e}(t)\right) = \mu_{possmall}\left(\dot{e}(t)\right) = 0.5$$

Some think of the membership function values as an "encoding" of the fuzzy controller numeric input values. The encoded information is then used in the fuzzy inference process that starts with "matching."

5.1.4 Matching: Determining Which Rules to Use

Next, we seek to explain how the inference mechanism in Figure 5.1 on page 156 operates. The inference process generally involves two steps:

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. This "matching" process involves determining the certainty that each rule applies, and typically we will more strongly take into account the recommendations of rules that we are more certain apply to the current situation.

2. The conclusions (what control actions to take) are determined using the rules that have been determined to apply at the current time. The conclusions are characterized with a fuzzy set (or sets) that represents the certainty that the input to the plant should take on various values.

We will cover step 1 in this subsection and step 2 in the next.

Premise Quantification via Fuzzy Logic

To perform inference we must first quantify each of the rules with fuzzy logic. To do this, we first quantify the meaning of the premises of the rules that are composed of several terms, each of which involves a fuzzy controller input. Consider Figure 5.9, where we list two terms from the premise of the rule

If error is zero **and** change-in-error is possmall **Then** rudder-input is negsmall

Above, we had quantified the meaning of the linguistic terms "error is zero" and "change-in-error is possmall" via the membership functions shown in Figure 5.8. Now we seek to quantify the linguistic premise "error is zero **and** change-in-error is possmall." Hence, the main item to focus on is how to quantify the logical "and" operation that combines the meaning of two linguistic terms. While we could use standard Boolean logic to combine these linguistic terms, since we have quantified them more precisely with fuzzy sets (i.e., the membership functions), we can use these.

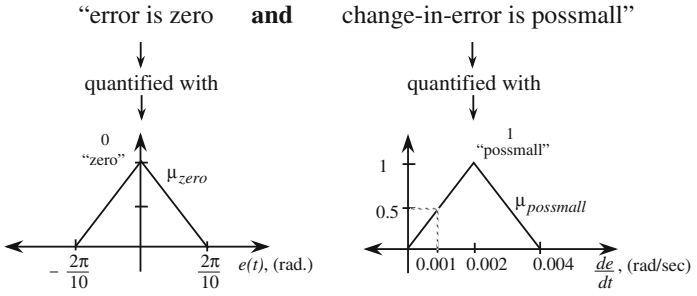

Figure 5.9: Membership functions of premise terms.

To see how to quantify the "and" operation, begin by supposing that $e(t) = \pi/10$ and $\dot{e}(t) = 0.0005$, so that using Figure 5.8 (or Figure 5.9) we see that

$$\mu_{zero}(e(t)) = 0.5$$

and

$$\mu_{possmall}\left(\dot{e}(t)\right) = 0.25$$

What, for these values of $e(t)$ and $\dot{e}(t)$, is the certainty of the statement

"error is zero **and** change-in-error is possmall"

that is the premise from the above rule? We will denote this certainty by $\mu_{premise}$. There are actually several ways to define it:

- *Minimum:* Define $\mu_{premise} = \min\{0.5, 0.25\} = 0.25$, that is, using the minimum of the two membership values.
- *Product:* Define $\mu_{premise} = (0.5)(0.25) = 0.125$, that is, using the product of the two membership values.

The premise of a rule is true to a certain degree and we think of rules that are "more true" as being more relevant to the current plant situation.

Do these quantifications make sense? Notice that both ways of quantifying the "and" operation in the premise indicate that you can be no more certain about the conjunction of two statements than you are about the individual terms that make them up (note that $0 \leq \mu_{premise} \leq 1$ for either case). If we are not very certain about the truth of one statement, how can we be any more certain about the truth of that statement "and" the other statement? It is important that you convince yourself that the above quantifications make sense. To do so, we recommend that you consider other examples of "anding" linguistic terms that have associated membership functions.

While we have simply shown how to quantify the "and" operation for one value of $e(t)$ and $\dot{e}(t)$, if we consider all possible $e(t)$ and $\dot{e}(t)$ values, we will obtain a multidimensional membership function $\mu_{premise}\left(e(t), \dot{e}(t)\right)$ that is a function of $e(t)$ and $\dot{e}(t)$ for each rule. For our example, if we choose the minimum operation to represent the "and" in the premise, then we get the multidimensional membership function $\mu_{premise}\left(e(t), \dot{e}(t)\right)$ shown in Figure 5.10 (and if we use product to represent the premise we get the premise membership function shown in Figure 5.11). Suppose that we use minimum to represent the conjunction in the premise. Notice that if we pick values for $e(t)$ and $\dot{e}(t)$, the value of the premise certainty $\mu_{premise}\left(e(t), \dot{e}(t)\right)$ represents how certain we are that the rule

If error is zero **and** change-in-error is possmall **Then** rudder-input is negsmall

is applicable for specifying the rudder input to the plant. As $e(t)$ and $\dot{e}(t)$ change, the value of $\mu_{premise}\left(e(t), \dot{e}(t)\right)$ changes according to Figure 5.10 (or Figure 5.11 if we use product to represent the rule), and we become less or more certain of the applicability of this rule.

In general we will have a different premise membership function for each of the rules in the rule base, and each of these will be a function of $e(t)$ and $\dot{e}(t)$

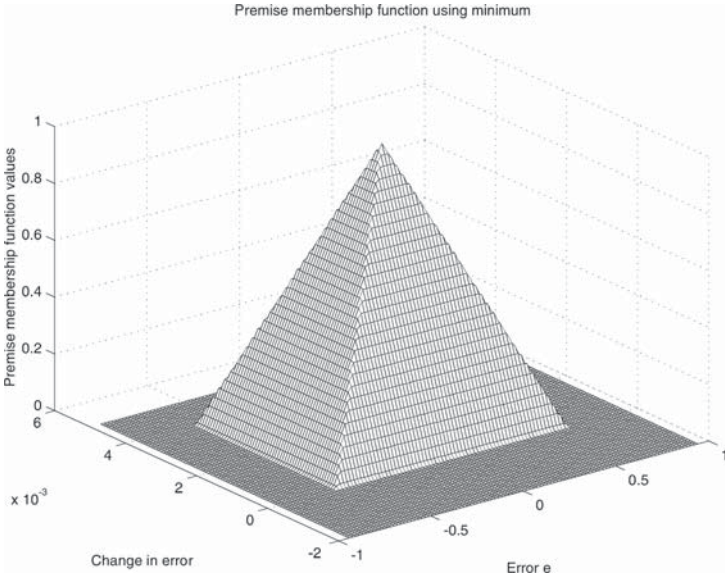

Figure 5.10: Membership function of the premise for a single rule using minimum to represent the conjunction.

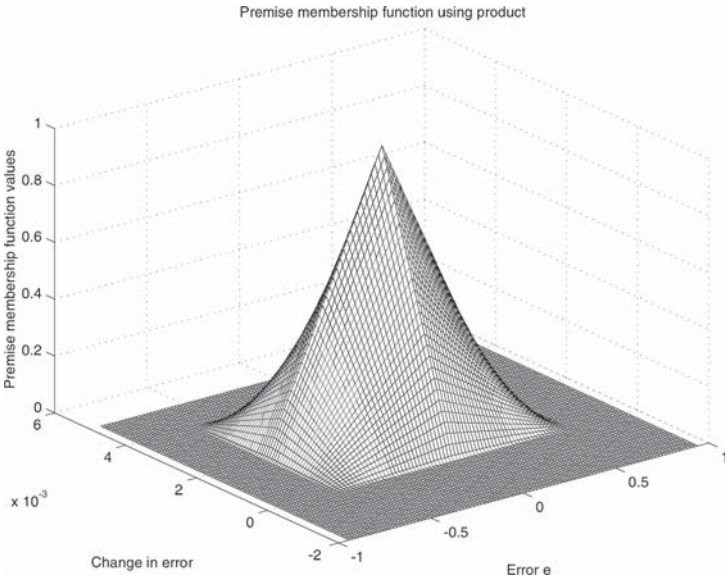

Figure 5.11: Membership function of the premise for a single rule using product to represent the conjunction.

so that given specific values of $e(t)$ and $\dot{e}(t)$, we obtain a quantification of the certainty that each rule in the rule base applies to the current situation. It is important you picture in your mind the situation where $e(t)$ and $\dot{e}(t)$ change dynamically over time. When this occurs, the values of $\mu_{premise}\left(e(t), \dot{e}(t)\right)$ for each rule change, and hence the applicability of each rule in the rule base for specifying the rudder input to the ship, changes with time.

Determining Which Rules Are On

Determining the applicability of each rule is called "matching." We say that a rule is "on at time t" if its premise membership function $\mu_{premise}(e(t), \dot{e}(t)) > 0$. Hence, the inference mechanism seeks to determine which rules are on to find out which rules are relevant to the current situation. In the next step, the inference mechanism will seek to combine the recommendations of all the rules to come up with a single conclusion.

Generally, only a few rules are relevant to choosing the plant input at any one time.

Consider, for the ship steering example, how we compute the rules that are on. Suppose that

$$e(t) = 0$$

and

$$\dot{e}(t) = 0.0015$$

Figure 5.12 shows the membership functions for the inputs and indicates, with thick black vertical lines, the $e(t)$ and $\dot{e}(t)$ values. Notice that $\mu_{zero}(e(t)) = 1$ but that the other membership functions for the $e(t)$ input are all "off" (i.e., their values are zero). For the $\dot{e}(t)$ input we see that $\mu_{zero}\left(\dot{e}(t)\right) = 0.25$ and $\mu_{possmall}\left(\dot{e}(t)\right) = 0.75$ and that all the other membership functions are off. This implies that rules that have the premise terms

"error is zero"
"change-in-error is zero"
"change-in-error is possmall"

are on (all other rules have $\mu_{premise}\left(e(t), \dot{e}(t)\right) = 0$). So, which rules are these? Using Table 5.1 on page 162, we find that the following rules are on:

1. **If** error is zero **and** change-in-error is zero **Then** rudder-input is zero
2. **If** error is zero **and** change-in-error is possmall **Then** rudder-input is negsmall

Note that since for the ship steering example we have at most two membership functions overlapping, we will never have more than four rules on at one time (this concept generalizes to many inputs). Actually, for this system we will either have one, two, or four rules on at any one time. To get only one rule on choose, for example, $e(t) = 0$ and $\dot{e}(t) = 0.002$. In this example, only rule 2 above is on. What values would you choose for $e(t)$ and $\dot{e}(t)$ to get four rules on? Why is it impossible, for this system, to have exactly three rules on?

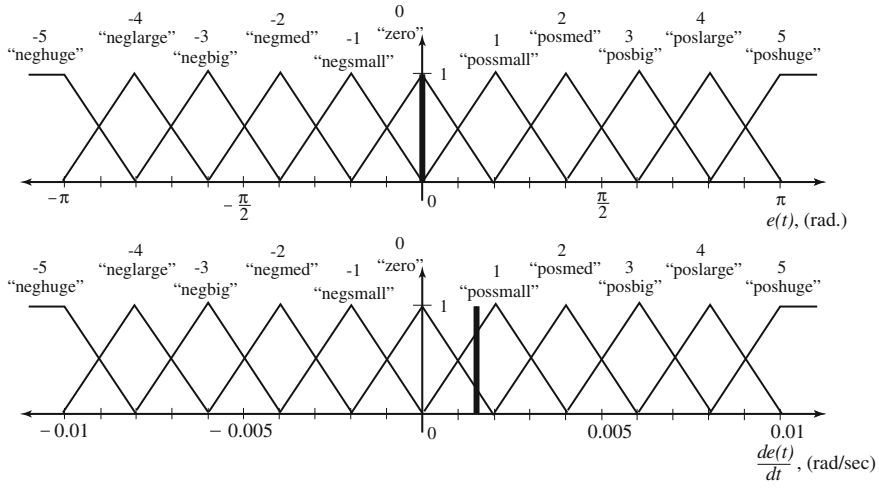

Figure 5.12: Input membership functions with input values.

It is useful to consider pictorially which rules are on. Consider Table 5.3, which is a copy of Table 5.2 on page 169 with boxes drawn around the consequents of the rules that are on (notice that these are the *same* two rules listed above). Notice that since $e(t) = 0$ ($e(t)$ is directly in the middle between the membership functions for "possmall" and "negsmall"), both of these membership functions are off. If we perturbed $e(t)$ slightly positive (negative), then we would have the two rules below (above) the two highlighted ones on also. With this, you should picture in your mind how a region of rules that are on (that involves no more than four cells in the body of Table 5.3, due to how we define the input membership functions) will dynamically move around in the table as the values of $e(t)$ and $\dot{e}(t)$ change. This completes our description of the "matching" phase of the inference mechanism.

5.1.5 Inference Step: Determining Conclusions

Next, we consider how to determine which conclusions should be reached when the rules that are on are applied to deciding what the rudder input to the ship should be. To do this, we will first consider the recommendations of each rule independently. Then later we will combine all the recommendations from all the rules to determine the rudder input to the tanker ship.

Recommendation from One Rule

Consider the conclusion reached by the rule

If error is zero **and** change-in-error is zero **Then** rudder-input is zero

Table 5.3: Rule Table for the Tanker Ship with Rules That Are "On" (highlighted). (Body of table holds the output membership function centers where each element should be multiplied by $8\pi/18$.)

		$\dot{e}$										
		-5	-4	-3	-2	-1	0	1	2	3	4	5
e	-5	1	1	1	1	1	1	.8	.6	.3	.1	0
	-4	1	1	1	1	1	.8	.6	.3	.1	0	$-.1$
	-3	1	1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$
	-2	1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$
	-1	1	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$
	0	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1
	1	.8	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1
	2	.6	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1
	3	.3	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1
	4	.1	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1	-1
	5	0	$-.1$	$-.3$	$-.6$	$-.8$	-1	-1	-1	-1	-1	-1

which for convenience we will refer to as "rule (1)." Using the minimum to represent the premise, we have

$$\mu_{premise_{(1)}} = \min\{1, 0.25\} = 0.25$$

(the notation $\mu_{premise_{(1)}}$ represents $\mu_{premise}$ for rule (1)) so that we are 0.25 certain that this rule applies to the current situation. The rule indicates that if its premise is true, then the action indicated by its consequent should be taken. For rule (1) the consequent is "rudder-input is zero" (this makes sense, for here the ship is headed in the proper direction, so we should not apply a rudder input that is different from zero since this would tend to move the ship heading away from the desired heading). The membership function for this consequent is shown in Figure 5.13(a). The membership function for the conclusion reached by rule (1), which we denote by $\mu_{(1)}$, is shown in Figure 5.13(b) and is given by

$$\mu_{(1)}(\delta) = \min\{\mu_{premise_{(1)}}, \mu_{zero}(\delta)\}$$

(where $\mu_{premise_{(1)}} = 0.25$ as determined above). This membership function defines the "implied fuzzy set"[1] for rule (1) (i.e., it is the conclusion that is implied by rule (1)). The justification for the use of the minimum operator to represent the implication is that *we can be no more certain about our consequent than our*

[1]This term has been used in the literature for a long time; however, there is no standard terminology for this fuzzy set. Others have called it, for example, a "consequent fuzzy set" or an "output fuzzy set" (which can be confused with the fuzzy sets that quantify the consequents of the rules).

premise. You should convince yourself that we could use the product operation to represent the implication also (in Section 5.1.6 we will do an example where we use the product).

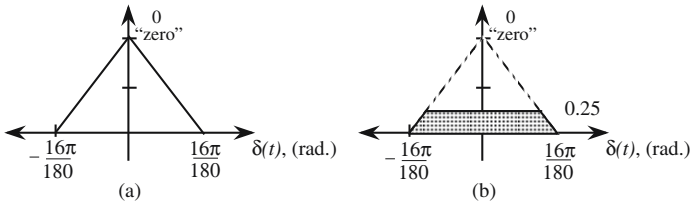

Figure 5.13: (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(1)}(\delta)$ for rule (1).

Notice that the membership function $\mu_{(1)}(\delta)$ is a *function* of δ and that the minimum operation will generally "chop off the top" of the $\mu_{zero}(\delta)$ membership function to produce $\mu_{(1)}(\delta)$. For different values of $e(t)$ and $\dot{e}(t)$ there will be different values of the premise certainty $\mu_{premise_{(1)}}(e(t), \dot{e}(t))$ for rule (1) and hence different *functions* $\mu_{(1)}(\delta)$ obtained (i.e., it will chop off the top at different points).

We see that $\mu_{(1)}(\delta)$ is in general a time-varying function that quantifies how certain rule (1) is that the force input δ should take on certain values. It is most certain that the force input should lie in a region around zero (see Figure 5.13(b)), and it indicates that it is certain that the force input should not be too large in either the positive or negative direction—this makes sense if you consider the linguistic meaning of the rule. The membership function $\mu_{(1)}(\delta)$ quantifies the conclusion reached by only rule (1) and only for the current $e(t)$ and $\dot{e}(t)$. It is important that the reader be able to picture how the shape of the implied fuzzy set changes as the rule's premise certainty changes over time.

Recommendation from Another Rule

Next, consider the conclusion reached by the other rule that is on:

If error is zero **and** change-in-error is possmall **Then** rudder-input is negsmall

which, for convenience, we will refer to as "rule (2)." Using the minimum to represent the premise, we have

$$\mu_{premise_{(2)}} = \min\{1, 0.75\} = 0.75$$

so that we are 0.75 certain that this rule applies to the current situation. Notice that we are much more certain that rule (2) applies to the current situation than rule (1) does. For rule (2) the consequent is "rudder-input is negsmall" (this makes sense, for here the ship is heading in the proper direction but is moving in the counterclockwise direction with a small velocity). The membership function for this consequent is shown in Figure 5.14(a). The membership function

Fuzzy control is "democratic" in that in deciding what input to put into the plant, it listens to the recommendation from each rule, to a degree specified by "how true" the premise of that rule is.

for the conclusion reached by rule (2), which we denote by $\mu_{(2)}$, is shown in Figure 5.14(b) (the shaded region) and is given by

$$\mu_{(2)}(\delta) = \min\{\mu_{premise_{(2)}}, \mu_{negsmall}(\delta)\}$$

(where $\mu_{premise_{(2)}} = 0.75$ as determined above). This membership function defines the implied fuzzy set for rule (2) (i.e., it is the conclusion that is reached by rule (2)). Once again, for different values of $e(t)$ and $\dot{e}(t)$ there will be different values of $\mu_{premise_{(2)}}(e(t), \dot{e}(t))$ for rule (2) and hence different *functions* $\mu_{(2)}(\delta)$ obtained. The reader should carefully consider the meaning of the implied fuzzy set $\mu_{(2)}(\delta)$. Rule (2) is quite certain that the control output (process input) should be a small negative value. This makes sense since if the ship has some counterclockwise velocity, then we would want to apply a negative rudder angle input. As rule (2) has a premise membership function that has higher certainty than for rule (1), we see that we are more certain of the conclusion reached by rule (2).

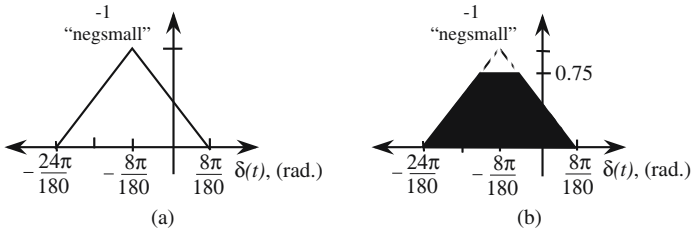

Figure 5.14: (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(2)}(\delta)$ for rule (2).

This completes the operations of the inference mechanism in Figure 5.1 on page 156. While the input to the inference process is the set of rules that are on, its output is the set of implied fuzzy sets that represent the conclusions reached by all the rules that are on. For our example, there are at most four conclusions reached since there are at most four rules on at any one time (and even some of these implied fuzzy sets may have a membership function that is zero for all values of δ so that we may ignore it).

Converting decisions to actions entails combining the recommendations of all the relevant rules.

5.1.6 Converting Decisions into Actions

Next, we consider the defuzzification operation, which is the final component of the fuzzy controller shown in Figure 5.1 on page 156. Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the "most certain" controller output (plant input). Some think of defuzzification as "decoding" the fuzzy set information produced by the inference process (i.e., the implied fuzzy sets) into numeric fuzzy controller outputs.

To understand defuzzification, it is best to first draw all the implied fuzzy sets on one axis as shown in Figure 5.15. We want to find the one output, which we denote by "δ^{crisp}," that best represents the conclusions of the fuzzy controller that are represented with the implied fuzzy sets. There are actually many approaches to defuzzification. We will consider two here.

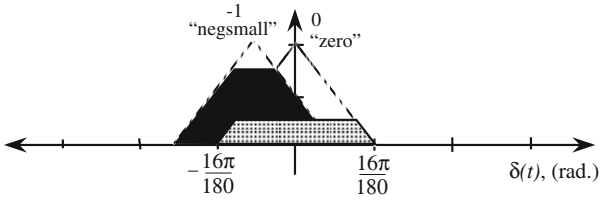

Figure 5.15: Implied fuzzy sets.

Combining Recommendations

Due to its popularity, we will first consider the "center of gravity" (COG) defuzzification method for combining the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function for the implied fuzzy set for the i^{th} rule (i.e., where the membership function for the i^{th} rule reaches its peak for our example since the output fuzzy sets are all symmetric about their peaks). For our example we have

$$b_1 = 0.0$$

and

$$b_2 = -0.1\left(\frac{8\pi}{18}\right)$$

as shown in Figure 5.15. Let

$$\int \mu_{(i)}$$

denote the area under the membership function $\mu_{(i)}$. The COG method computes δ^{crisp} to be

$$\delta^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \tag{5.1}$$

This is the classical formula for computing the center of gravity. In this case it is for computing the center of gravity of the implied fuzzy sets. Three items about Equation (5.1) are important to note:

1. Practically, we cannot have output membership functions that have infinite area since even though they may be "chopped off" in the minimum operation for the implication (or scaled for the product operation), they can still end up with infinite area. This is the reason we do not allow

infinite area membership functions for the linguistic values for the controller output (e.g., we did not allow the saturated membership functions at the outermost edges as we had for the inputs shown in Figure 5.8 on page 166).

2. You must be careful to define the input and output membership functions so that the sum in the denominator of Equation (5.1) is not equal to zero no matter what the inputs to the fuzzy controller are. Essentially, this means that we must have some sort of conclusion for all possible control situations we may encounter.

3. While at first glance it may not appear so, $\int \mu_{(i)}$ is easy to compute for our example. For the case where we have symmetric triangular output membership functions that peak at one and have a base width of w, simple geometry can be used to show that the area under a triangle "chopped off" at a height of h (such as the ones in Figures 5.13 and 5.14) is equal to

$$w\left(h - \frac{h^2}{2}\right)$$

Given this, the computations needed to compute δ^{crisp} are not too significant (note that if w is the same for every output membership function, then it cancels in Equation (5.1)).

We see that the property of membership functions being symmetric for the output is important since in this case no matter whether the minimum or product is used to represent the implication, it will be the case that the center of the implied fuzzy set will be the same as the center of the consequent fuzzy set from which it is computed. If the output membership functions are not symmetric, then their centers, which are needed in the computation of the COG, will change depending on the membership value of the premise. This will result in the need to recompute the center at each time instant.

Using Equation (5.1) with Figure 5.15, we have

$$\delta^{crisp} = \frac{(0)\left(0.25 - \frac{(0.25)^2}{2}\right) + \left(-0.1\frac{8\pi}{18}\right)\left(0.75 - \frac{(0.75)^2}{2}\right)}{\left(0.25 - \frac{(0.25)^2}{2}\right) + \left(0.75 - \frac{(0.75)^2}{2}\right)} = -0.0952$$

as the input to the ship for the given $e(t)$ and $\dot{e}(t)$.

Does this value for a force input (i.e., -5.4545 degrees) make sense? Consider Figure 5.16, where we have taken the implied fuzzy sets from Figure 5.15 and simply added an indication of what number COG defuzzification says is the best representation of the conclusions reached by the rules that are on. Notice that the value of δ^{crisp} is roughly in the middle of where the implied fuzzy sets say they are most certain about the value for the force input. In fact, recall that we had

$$e(t) = 0$$

and

$$\dot{e}(t) = 0.0015$$

so the ship is at the desired heading at this time instant but is moving counterclockwise with a small velocity; hence, it makes sense to apply a small negative rudder input, and the fuzzy controller does this.

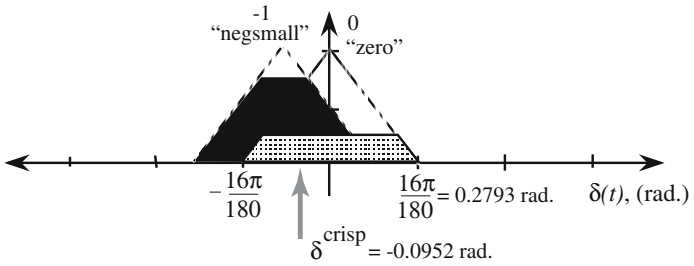

Figure 5.16: Implied fuzzy sets.

It is interesting to note that for our example it will be the case that

$$-\frac{8\pi}{18} \leq \delta^{crisp} \leq \frac{8\pi}{18}$$

To see this, consider Figure 5.17, where we have drawn the output membership functions. Notice that even though we have extended the membership functions at the outermost edges past $-8\pi/18$ and $+8\pi/18$ (see the shaded regions), the COG method will never compute a value outside this range.

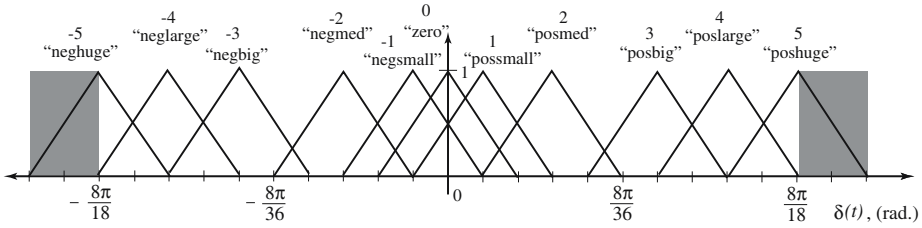

Figure 5.17: Output membership functions.

The reason for this comes directly from the definition of the COG method in Equation (5.1). The center of gravity for these shapes simply cannot extend beyond $-8\pi/18$ and $+8\pi/18$. Practically speaking, this ability to limit the range of inputs to the plant is useful in real applications since all real plant inputs are limited to lie in a specific range. The other conclusion that we would reach from this discussion is that in defining the membership functions for the fuzzy controller, we must take into account what method is going to be used for defuzzification.

Other Ways to Compute and Combine Recommendations

As another example, it is interesting to consider how to compute, by hand, the operations that the fuzzy controller takes when we use the product to represent the implication or the "center-average" defuzzification method.

First, consider the use of the product. Consider Figure 5.18, where we have drawn the output membership functions for "negsmall" and "zero" as dotted lines. The implied fuzzy set from rule (1) is given by the membership function

$$\mu_{(1)}(\delta) = 0.25\mu_{zero}(\delta)$$

shown in Figure 5.18 as the shaded triangle; the implied fuzzy set for rule (2) is given by the membership function

$$\mu_{(2)}(\delta) = 0.75\mu_{negsmall}(\delta)$$

shown in Figure 5.18 as the dark triangle. The computation of the COG is easy since we can use $\frac{1}{2}wh$ as the area for a triangle with base width w and height h (and the factor $\frac{1}{2}w$ cancels in Equation 5.1). When we use product to represent the implication, we obtain

$$\delta^{crisp} = \frac{(0)(0.25) + \left(-0.1\frac{8\pi}{18}\right)(0.75)}{0.25 + 0.75} = -0.1047$$

which also makes sense.

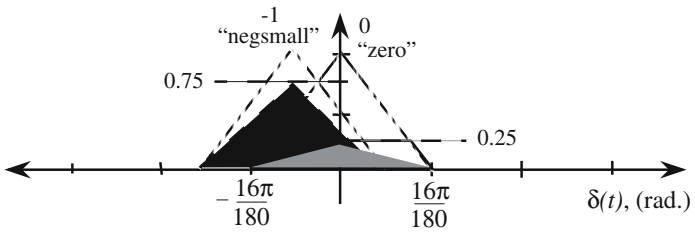

Figure 5.18: Implied fuzzy sets when the product is used to represent the implication.

Next, as another example of how to combine recommendations, we will introduce the "center-average" method for defuzzification. For this method we let

$$\delta^{crisp} = \frac{\sum_i b_i \mu_{premise_{(i)}}}{\sum_i \mu_{premise_{(i)}}} \tag{5.2}$$

where b_i once again denotes the center of the membership function for the implied fuzzy set for the i^{th} rule (i.e., where the membership function for the i^{th} rule reaches its peak for our example since the output fuzzy sets are all symmetric about their peaks). To compute the $\mu_{premise_{(i)}}$ we use, for example, minimum. We call it the "center-average" method since Equation (5.2)

is a weighted average of the center values of the membership functions of the implied fuzzy sets (and output membership function centers). Basically, the center-average method replaces the areas of the implied fuzzy sets that are used in COG with the values of $\mu_{premise_{(i)}}$. This is a valid replacement since the area of the implied fuzzy set is generally proportional to $\mu_{premise_{(i)}}$ since $\mu_{premise_{(i)}}$ is used to chop the top off (minimum) or scale (product) the triangular output membership function when COG is used for our example. For the above example, we have

$$\delta^{crisp} = \frac{(0)(0.25) + \left(-0.1\frac{8\pi}{18}\right)(0.75)}{0.25 + 0.75} = -0.1047$$

which is the same value as above (for this special case). Some like the center-average defuzzification method because the computations needed are generally simpler than for COG because when the output membership functions are symmetric (the usual case), they are easy to store since the only relevant information they provide is their center values (b_i) (i.e., their shape does not matter, just their center value, so this is all that needs to be stored). Moreover, the areas of the implied fuzzy sets do not have to be computed.

Notice that while both values computed for the different inference and defuzzification methods provide reasonable command inputs to the plant, it is difficult to say which is best without further investigations (e.g., simulations or implementation). This ambiguity about how to define the fuzzy controller actually extends to the general case and also arises in the specification of all the other fuzzy controller components, as we discuss below. Some would call this "ambiguity" a design flexibility, but unfortunately there are not too many guidelines on how best to choose the inference strategy and defuzzification method, so such flexibility is of questionable value.

Graphical Depiction of Fuzzy Decision Making

For convenience, we summarize the procedure that the fuzzy controller uses to compute its outputs given its inputs in Figure 5.19. Here, we use the minimum operator to represent the "and" in the premise and the implication and COG defuzzification. The reader is advised to study each step in this diagram to gain a fuller understanding of the operation of the fuzzy controller. To do this, develop a similar diagram for the case where the product operator is used to represent the "and" in the premise and the implication, and choose values of $e(t)$ and $\dot{e}(t)$ that will result in four rules being on. Then, repeat the process when center-average defuzzification is used with either minimum or product used for the premise. Also, learn how to picture in your mind how the parameters of this graphical representation of the fuzzy controller operations change as the fuzzy controller inputs change.

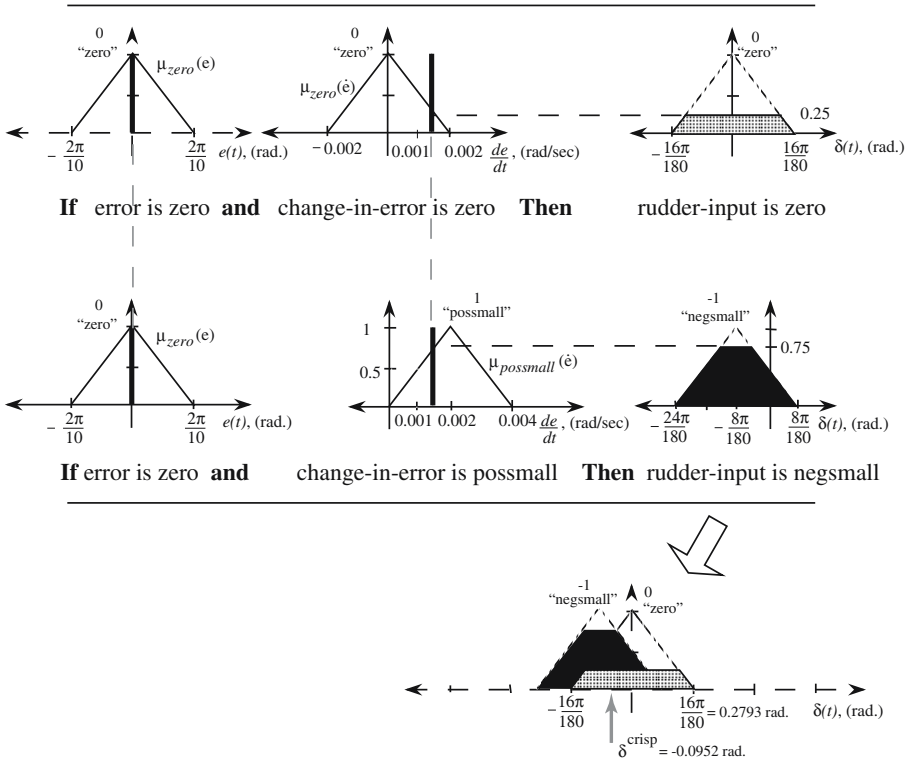

Figure 5.19: Graphical representation of fuzzy controller operations.

5.2 General Fuzzy Systems

In this section we introduce multi-input multi-output fuzzy systems, Takagi-Sugeno fuzzy systems, and then show how to develop mathematical representations for these.

5.2.1 Multiple Input Multiple Output Fuzzy Systems

A fuzzy system is a static nonlinear mapping between its inputs and outputs (i.e., it is not a dynamic system). Some people include the preprocessing of the inputs to the fuzzy system (e.g., differentiators or integrators) in the definition of the fuzzy system and thereby obtain a "fuzzy system" that *is* dynamic. In this book, we adopt the convention that such preprocessing is not part of the fuzzy system, and hence the fuzzy system will always be a memoryless nonlinear map.

A general multiple input multiple output (MIMO) fuzzy system with inputs

u_i, $i = 1, 2, \ldots, n$ and outputs y_j, $j = 1, 2, \ldots, m$ is shown in Figure 5.20. The inputs and outputs are "crisp;" that is, they are numeric values. The fuzzification block converts the crisp inputs to fuzzy sets (i.e., it converts them to "singleton" fuzzy sets, ones that have membership functions with zero width and a unit pulse at the value of the input; an example singleton membership function is shown in Figure 5.21). The inference mechanism uses the fuzzy rules in the rule base to produce fuzzy conclusions (e.g., the implied fuzzy sets), and the defuzzification block converts these fuzzy conclusions into the crisp outputs. In this subsection we explain how to define a MIMO fuzzy controller.

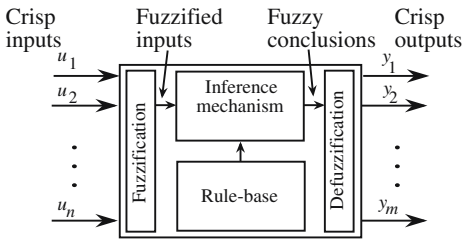

Figure 5.20: Fuzzy system (controller).

First, note that to define a MIMO fuzzy system you simply specify m multiple input single output (MISO) fuzzy systems, where the output of the j^{th} fuzzy system is y_j, $j = 1, 2, \ldots, m$. We already know how to specify a MISO fuzzy system with $n = 2$ inputs so all we need to do is explain how to define a MISO fuzzy system with $n > 2$ inputs (then the case for $n = 1$ will be clear). To do this, note that for each input we define membership functions as we did for the e and $\dot{e}$ universes of discourse for the ship example. You form rules using the n inputs in n premise terms. Next, fuzzification is the same as earlier—you just compute the membership values on all the input universes of discourse. Next, we need to compute the fuzzy logic quantification of the conjunction between n premise terms rather than just two. To do this we take the same approach as before, but take the minimum (or product) of n membership function values to represent the conjunction of n premise terms. This will give us $\mu_{premise_{(i)}}$ for the i^{th} rule and we will compute this for all the rules. From this point on the process is exactly the same as the two-input case since after the matching process, the inference mechanism computations of the implied fuzzy sets and the defuzzification computations only depend on $\mu_{premise_{(i)}}$. Therefore, the effect of additional inputs to the fuzzy system is on the premise and hence the computations needed to find $\mu_{premise_{(i)}}$ for all the rules.

The computations necessary for MISO fuzzy systems will be reviewed in Section 5.2.3 where we explain how to develop mathematical representations of fuzzy systems for n input MISO fuzzy systems.

5.2.2 Takagi-Sugeno Fuzzy Systems

The fuzzy systems discussed in the previous sections will be referred to as a "standard fuzzy system," regardless of the particular choices for premise representation, inference, defuzzification, etc. In this subsection we will define a "functional fuzzy system," of which the Takagi-Sugeno fuzzy system is a special case. For the functional fuzzy system, we use singleton fuzzification and the premise is defined the same as it is for the rule for the standard fuzzy system. The consequents of the rules are different, however. Instead of a linguistic term with an associated membership function, in the consequent we use a *function* $b_i = g_i(\cdot)$ (hence the name "functional fuzzy system") that does not have an associated membership function (or you can think of it as a singleton membership function whose position changes as specified by the function g_i for the i^{th} rule). Notice that often the argument of g_i contains the fuzzy system inputs that are used in the premise of the rule, but other variables may also be used. The choice of the function depends on the application being considered. Below, we will discuss linear and affine functions but many others are possible. For instance, you may want to choose

$$b_i = g_i(\cdot) = a_{i,0} + a_{i,1}(u_1)^2 + \cdots + a_{i,n}(u_n)^2$$

or

$$b_i = g_i(\cdot) = \exp\left[a_{i,1}sin(u_1) + \cdots + a_{i,n}sin(u_n)\right]$$

Virtually any function can be used (e.g., a neural network mapping or another fuzzy system), which makes the functional fuzzy system very general.

Let R denote the number of rules. For the functional fuzzy system we can use an appropriate operation for representing the premise (e.g., minimum or product), and defuzzification may be obtained using

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i(z)}{\sum_{i=1}^{R} \mu_i(z)} \tag{5.3}$$

where $\mu_i(z)$ is the premise membership function (rather than $\mu_{premise_{(i)}}$ which was used in our earlier discussion). It is assumed that the functional fuzzy system is defined so that no matter what its inputs are, we have $\sum_{i=1}^{R} \mu_i(z) \neq 0$. The vector z can be chosen in several ways. One common choice is to use $z = [u_1, u_2, \ldots, u_n]^\top$; however, sometimes z might hold other variables, or only a subset of the u_i values (with only a subset of the values, complexity of the mapping generally decreases since the computations needed to find $\mu_i(z)$ are simplified).

A Takagi-Sugeno fuzzy system is an interpolator between linear mappings.

In the special case where

$$b_i = g_i(\cdot) = a_{i,0} + a_{i,1}u_1 + \cdots + a_{i,n}u_n$$

(where the $a_{i,j}$ are fixed real numbers) the functional fuzzy system is referred to as a "Takagi-Sugeno fuzzy system."

If $a_{i,0} = 0$, then the $g_i(\cdot)$ mapping is a linear mapping and if $a_{i,0} \neq 0$, then the mapping is called "affine." Often, however, as is standard, we will refer to the affine mapping as a linear mapping for convenience. Overall, we see that the Takagi-Sugeno fuzzy system performs a nonlinear interpolation between linear mappings. In control applications, the linear mappings can each represent a different linear controller and the Takagi-Sugeno fuzzy system interpolates between these and applies combinations of the linear controller outputs (similar in some cases to what is called "gain scheduled control" in conventional control).

5.2.3 Mathematical Representations of Fuzzy Systems

Notice that each formula for defuzzification in the previous sections provides a mathematical description of a fuzzy system. There are many ways to represent the operations of a fuzzy system with mathematical formulas. Next, we clarify how to construct and interpret such mathematical formulas for the case where center-average defuzzification is used for n-input MISO fuzzy systems. Similar ideas apply for other defuzzification strategies, MIMO fuzzy systems, and Takagi-Sugeno fuzzy systems.

Two Different Approaches

Rules and Membership Functions: To represent linguistic rules, let $\tilde{u}_i$, $i = 1, 2, \ldots, n$, and $\tilde{y}$ denote the linguistic variables that describe u_i, $i = 1, 2, \ldots, n$, and y, respectively. Let $\tilde{A}_i^j$ denote the j^{th} linguistic value for the i^{th} input universe of discourse (here, suppose that $i = 1, 2, \ldots, n$, but that j can, for instance, take on values that are equal to the linguistic-numeric values). Similarly, let $\tilde{B}^p$ denote the p^{th} linguistic value on the output universe of discourse that has linguistic variable $\tilde{y}$. With this, a linguistic rule may be described mathematically by

$$\begin{gathered}\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^j \\ \textbf{and } \tilde{u}_2 \text{ is } \tilde{A}_2^k \\ \textbf{and } \cdots \\ \textbf{and } \tilde{u}_n \text{ is } \tilde{A}_n^l \\ \textbf{Then } \tilde{y} \text{ is } \tilde{B}^p\end{gathered}$$

Suppose that there are R such rules.

Next, consider the mathematical quantification of membership functions. Clearly, many other choices for the shape of the membership function are possible than the ones discussed so far, and these will each provide a different meaning for the linguistic values that they quantify. See Figure 5.21 for a graphical illustration of a variety of membership functions and Tables 5.4 and 5.5 for a mathematical characterization of the triangular and Gaussian membership functions, including the membership functions that are often used at the outermost edges of the input universe of discourse when the "center" membership functions are used at various positions along the input universe of discourse (other membership functions can be characterized with mathematics using a similar

approach). For practice, you should sketch the membership functions that are described in Tables 5.4 and 5.5. Notice that for Table 5.4, c^L specifies the "saturation point" and w^L specifies the slope of the nonunity and nonzero part of μ^L. Similarly, for μ^R. For μ^C notice that c is the center of the triangle and w is the base width. Analogous definitions are used for the parameters in Table 5.5. In Table 5.5, for the "centers" case note that this is the traditional definition for the Gaussian membership function. This definition is clearly different from a standard Gaussian probability density function, in both the meaning of c and σ, and in the scaling of the exponential function. Recall that it is possible that a Gaussian probability density function has a maximum value at a value other than one; the standard Gaussian membership function always has its peak value at one.

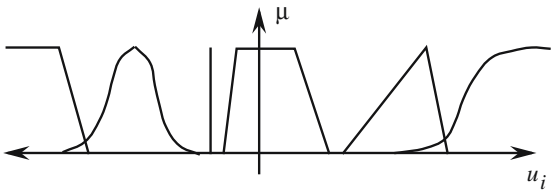

Figure 5.21: Some typical membership functions.

Table 5.4: Mathematical Characterization of Triangular Membership Functions

	Triangular and related membership functions
Left	$\mu^L(u) = \begin{cases} 1 & \text{if } u \le c^L \\ \max\left\{0, 1 + \frac{c^L - u}{0.5w^L}\right\} & \text{otherwise} \end{cases}$
Centers	$\mu^C(u) = \begin{cases} \max\left\{0, 1 + \frac{u-c}{0.5w}\right\} & \text{if } u \le c \\ \max\left\{0, 1 + \frac{c-u}{0.5w}\right\} & \text{otherwise} \end{cases}$
Right	$\mu^R(u) = \begin{cases} \max\left\{0, 1 + \frac{u-c^R}{0.5w^R}\right\} & \text{if } u \le c^R \\ 1 & \text{otherwise} \end{cases}$

Approach 1: Given Membership Functions, All Possible Rules: Assume that we use center-average defuzzification so that the formula describing how to compute the output is

$$y = \frac{\sum_{i=1}^{R} b_i \mu_i}{\sum_{i=1}^{R} \mu_i} \tag{5.4}$$

where for convenience we use μ_i to represent the premise certainty for the i^{th} rule (rather than $\mu_{premise_{(i)}}$ which was more descriptive for our earlier discussion, but a bit cumbersome).

Table 5.5: Mathematical Characterization of Gaussian Membership Functions

	Gaussian and related membership functions
Left	$\mu^L(u) = \begin{cases} 1 & \text{if } u \le c^L \\ \exp\left(-\frac{1}{2}\left(\frac{u-c^L}{\sigma^L}\right)^2\right) & \text{otherwise} \end{cases}$
Centers	$\mu^C(u) = \exp\left(-\frac{1}{2}\left(\frac{u-c}{\sigma}\right)^2\right)$
Right	$\mu^R(u) = \begin{cases} \exp\left(-\frac{1}{2}\left(\frac{u-c^R}{\sigma^R}\right)^2\right) & \text{if } u \le c^R \\ 1 & \text{otherwise} \end{cases}$

To be more explicit in Equation (5.4), we need to first define the premise membership functions μ_i in terms of the individual membership functions that describe each of the premise terms. Suppose that we use product to represent the conjunctions in the premise of each rule. Suppose that we use the triangular membership functions in Table 5.4 where we suppose that $\mu_j^L(u_j)$ ($\mu_j^R(u_j)$) is the "left-" ("right-") most membership function on the j^{th} input universe of discourse. In addition, let $\mu_j^{C_i}(u_j)$ be the i^{th} "center" membership function for the j^{th} input universe of discourse. In this case, to define $\mu_j^L(u_j)$ we simply add a "j" subscript to the parameters of the "left" membership function from Table 5.4. In particular, we use c_j^L and w_j^L to denote the j^{th} values of these parameters. We take a similar approach for the $\mu_j^R(u_j)$, $j = 1, 2, \ldots, n$. For $\mu_j^{C_i}(u_j)$ we use c_j^i (w_j^i) to denote the i^{th} triangle center (triangle base width) on the j^{th} input universe of discourse.

Suppose that we use all possible combinations of input membership functions to form the rules, and that each premise has a term associated with each and every input universe of discourse. A more detailed description of the fuzzy system in Equation (5.4) is given by

$$y = \frac{b_1 \prod_{j=1}^n \mu_j^L(u_j) + b_2 \mu_1^{C_1}(u_1) \prod_{j=2}^n \mu_j^L(u_j) + \cdots}{\prod_{j=1}^n \mu_j^L(u_j) + \mu_1^{C_1}(u_1) \prod_{j=2}^n \mu_j^L(u_j) + \cdots}$$

The first term in the numerator is $b_1\mu_1$ in Equation (5.4). Here, we have called the "first rule" the one that has premise terms all described by the membership functions $\mu_j^L(u_j)$, $j = 1, 2, \ldots, n$. The second term in the numerator is $b_2\mu_2$ and it uses $\mu_1^{C_1}(u_1)$ on the first universe of discourse and the leftmost ones on the other universes of discourse (i.e., $j = 2, 3, \ldots, n$). Continuing in a similar manner, the sum in the numerator (and denominator) extends to include all possible combinations of products of the input membership functions, and this fully defines the μ_i in Equation (5.4).

Overall, we see that because we need to define rules resulting from all possible combinations of *given* input membership functions, of which there are three

kinds (left, center, right), the explicit mathematical representation of the fuzzy system is somewhat complicated. To avoid some of the complications, we first specify a single function that represents all three types of input membership functions. Suppose that on the j^{th} input universe of discourse we number the input membership functions from left to right as $1, 2, \ldots, N_j$, where N_j is the number of input membership functions on the j^{th} input universe of discourse. A single membership function that represents all three in Table 5.4 is

$$\mu_j^i(u_j) = \begin{cases} 1 & \text{if } (u_j \le c_j^1, i = 1) \text{ or } (u_j \ge c_j^{N_j}, i = N_j) \\ \max\left\{0, 1 + \frac{u_j - c_j^i}{0.5w_j^i}\right\} & \text{if } u_j \le c_j^i \text{ and } (u_j > c_j^1 \text{ and } u_j < c_j^{N_j}) \\ \max\left\{0, 1 + \frac{c_j^i - u_j}{0.5w_j^i}\right\} & \text{if } u_j > c_j^i \text{ and } (u_j > c_j^1 \text{ and } u_j < c_j^{N_j}) \end{cases}$$

A similar approach can be used for the Gaussian case in Table 5.5.

Suppose we use the shorthand notation

$$(j, k, \ldots, l; p)_i$$

to denote the i^{th} rule shown above. In this notation, suppose the indices in (the "tuple") $(j, k, \ldots, l)$ range over $1 \le j \le N_1$, $1 \le k \le N_2$, ..., $1 \le l \le N_n$, and specify which linguistic value is used on each input universe of discourse. Correspondingly, each index in the tuple $(j, k, \ldots, l)$ also specifies the linguistic-numeric value of the input membership function used on each input universe of discourse.

Let

$$b^{(j,k,\ldots,l;p)_i}$$

denote the output membership function (a singleton) center for the i^{th} rule. Note that we use "i" in the notation $(j, k, \ldots, l; p)_i$ simply as a label for each rule (i.e., we number the rules in the rule base from 1 to R, and i is this number). Hence, when we are given i, we know the values of j, k, ..., l, and p. Because of this, an explicit description of the fuzzy system in Equation (5.4) is given by

$$y = \frac{\sum_{i=1}^{R} b^{(j,k,\ldots,l;p)_i} \mu_1^j \mu_2^k \cdots \mu_n^l}{\sum_{i=1}^{R} \mu_1^j \mu_2^k \cdots \mu_n^l} \tag{5.5}$$

This formula clearly shows the use of the product to represent the premise. Notice that since we use all possible combinations of input membership functions to form the rules there are

$$R = \prod_{j=1}^{n} N_j$$

rules, and hence it takes

$$\sum_{j=1}^{n} 2N_j + \prod_{j=1}^{n} N_j \tag{5.6}$$

parameters to describe the fuzzy system since there are two parameters for each input membership function and R output membership function centers.

For some applications, however, all the output membership functions are not distinct. For example, consider the ship steering example where eleven output membership function centers are defined, and there are $R = 121$ rules. To define the center positions $b^{(j,k,\ldots,l;p)_i}$ so that they take on only a fixed number of given values, that is less than R, one approach is to specify them as a function of the indices of the input membership functions. What is this function for the ship steering example?

Approach 2: Parameterization in Terms of Rules: A different approach to avoiding some of the complications encountered in specifying a fuzzy system mathematically is to use a different notation, and hence a different definition for the fuzzy system. For this alternative approach, for the sake of variety, we will use Gaussian input membership functions. In particular, for simplicity, suppose that for the input universes of discourse we only use membership functions of the "center" Gaussian form shown in Table 5.5. For the i^{th} rule, suppose that the input membership function is

$$\exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)$$

for the j^{th} input universe of discourse. Hence, even though we use the same notation for the membership function, these centers c_j^i are different from those used above, both because we are using Gaussian membership functions here, and because the "i" in c_j^i is the index for the rules, not the membership function on the j^{th} input universe of discourse. Similar comments can be made about the σ_j^i, $i = 1, 2, \ldots, R$, $j = 1, 2, \ldots, n$. If we let b_i, $i = 1, 2, \ldots, R$, denote the center of the output membership function for the i^{th} rule, use center-average defuzzification, and product to represent the conjunctions in the premise, then

$$y = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{u_j - c_j^i}{\sigma_j^i}\right)^2\right)} \tag{5.7}$$

is an explicit representation of a fuzzy system. Note that we do not use the "left" and "right" versions of the Gaussian membership functions in Table 5.5 as this complicates the notation.

There are nR input membership function centers, nR input membership function spreads, and R output membership function centers. Hence, we need a total of

$$R(2n + 1)$$

parameters to describe this fuzzy system.

It is possible to write down the complete mathematical description of the mapping between the input and output of the fuzzy system.

Now, while the fuzzy systems in Equations (5.5) and (5.7) are in general different, it is interesting to compare the number of parameters needed to describe a fuzzy system using each approach. In practical situations, we often

have $N_j \geq 3$ for each $j = 1, 2, \ldots, n$, and sometimes the number of membership functions on each input universe of discourse can be 10 or more. From Equation (5.6) we can clearly see that large values of n will result in a fuzzy system with many parameters (there is an exponential increase in the number of rules). On the other hand, using the fuzzy system in Equation (5.7), the user specifies the number of rules. This, coupled with the number of inputs n, specifies the total number of parameters. There is not an exponential growth in the number of parameters in Equation (5.7) in the same way as there is in the fuzzy system in Equation (5.5), so you may be tempted to view the definition in Equation (5.7) as a better one. Such a conclusion, can, however be erroneous for several reasons.

First, the type of fuzzy system defined by Equation (5.5) is sometimes more natural in control design when you use triangular membership functions since you often need to make sure that there will be no point on any input universe of discourse where there is no membership function with a nonzero value (why?). Of course, if you are careful, you can avoid this problem with the fuzzy system also represented by Equation (5.7). Second, suppose that the number of rules for Equation (5.7) is the same as that for Equation (5.5). In this case, the number of parameters needed to describe the fuzzy system in Equation (5.7) is

$$\left(\prod_{j=1}^{n} N_j \right) (2n + 1)$$

Now, comparing this to Equation (5.6) you see that for many values of N_j, $j = 1, 2, \ldots, n$, and number of inputs n, it is possible that the fuzzy system in Equation (5.7) will require many more parameters to specify it than the fuzzy system in Equation (5.5). Hence, the inefficiency in the representation in Equation (5.5) lies in having all possible combinations of output membership function centers, which results in exponential growth in the number of parameters needed to specify the fuzzy system. The inefficiency in the representation in Equation (5.7) lies in the fact that, in a sense, membership functions on the input universes of discourse are not reused by each rule. There are new input membership functions for every rule.

Generally, it is difficult to know which is the best fuzzy system for a particular problem. In this book, we will sometimes use the mathematical representation in Equation (5.7) because it is somewhat simpler, and possesses some properties that we will exploit. At other times we will be implicitly using the representation in Equation (5.5) because it will lend to the development of certain techniques.

Finally, we would like to recommend that you practice creating mathematical representations of fuzzy systems. For instance, it is good practice to create a mathematical representation of the fuzzy controller for ship steering of the form of Equation (5.5), and then also use Equation (5.7) to specify the same fuzzy system. Comparing these two approaches, and resolving the issues in specifying the output centers for the Equation (5.5) case, will help clarify the issues discussed in this section.

5.2.4 Relationships Between Neural and Fuzzy Systems

There are two ways in which there are relationships between fuzzy systems and neural networks. First, techniques from one area can be used in the other. Second, in some cases the functionality (i.e., the nonlinear function that they implement) is identical. Some label the intersection between fuzzy systems and neural networks with the term "fuzzy-neural" or "neuro-fuzzy" to highlight that techniques from both fields are being used. Here, we avoid this terminology and simply highlight the basic relationships between the two fields.

The multilayer perceptron should be viewed as a nonlinear network whose nonlinearity can be tuned by changing the weights and biases. The fuzzy system is also a tunable nonlinearity whose shape can be changed by tuning, for example, the membership functions. Since both are tunable nonlinearities, it is possible to use the methods of Part III to train either one (e.g., least squares, or gradient methods can be used to train both fuzzy and neural systems). While multilayer perceptron networks can take on a similar role to that of a fuzzy system in performing the function of being a tunable nonlinearity, an advantage that the fuzzy system may have, however, is that it often facilitates the incorporation of heuristic knowledge into the solution to the problem, which can, at times, have a significant impact on the quality of the solution.

Some radial basis function neural networks are *equivalent* to some standard fuzzy systems in the sense that they are functionally equivalent (i.e., given the same inputs, they will produce the same outputs). To see this, suppose that in Equation (4.12) we let $n_R = R$ (i.e., the number of receptive field units equal to the number of rules), let the receptive field unit strengths be equal to the output membership function centers, and choose the receptive field units as

$$R_i(x) = \mu_i(x)$$

(i.e., choose the receptive field units to be the same as the premise membership functions). In this case we see that the radial basis function neural network is *identical* to a certain fuzzy system that uses center-average defuzzification. This fuzzy system is then given by

$$y = F_{rbf}(x, \theta) = F_{fs}(x, \theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)}$$

where θ holds the membership function parameters for the fuzzy system or strengths and receptive field unit parameters for the radial basis function neural network.

The equivalence between this type of fuzzy system and a radial basis function neural network shows that *all the techniques in this book for the above type of fuzzy system work in the same way for the above type of radial basis function neural network.*

Due to the above relationships between fuzzy systems and neural networks, some would like to view fuzzy systems and neural networks as identical areas. This is, however, not the case for the following reasons:

- There are classes of neural networks (e.g., dynamic neural networks) that may have a fuzzy system analog, but if so, it would have to include not only standard fuzzy components but some form of a differential equation component.
- There are certain fuzzy systems that have no clear neural analog. Consider, for example, certain "fuzzy dynamic systems." We can, however, envision how you could go about designing a neural analog to such fuzzy systems.
- The neural network has traditionally been a "black box" approach where the weights and biases are trained (e.g., using gradient methods like backpropagation) using data, often without using extra heuristic knowledge we often have. In fuzzy systems you can incorporate heuristic information and use data to train them. This last difference is often quoted as being one of the advantages of fuzzy systems over neural networks, at least for some applications.

In Part III we will show how to train both neural networks and fuzzy systems and will *try* to provide some insights into which is best to use for a particular application.

5.3 Design Example: Fuzzy Control for Tanker Ship Steering

As there is no general systematic procedure for the design of fuzzy controllers that will definitely produce a high-performance fuzzy control system for a wide variety of applications, it is necessary to learn about fuzzy controller design via examples. Here, we continue with the ship steering example to provide an introduction to the typical procedures used in the design (and redesign) of a fuzzy controller. First, however, we discuss how to code the fuzzy controller for the tanker ship.

5.3.1 Simulation of a Fuzzy Controller

Often, before you implement a fuzzy controller, there is a need to perform a simulation-based evaluation of its performance. To perform a simulation, we will need a model of the plant and a computer program that will simulate the fuzzy control system (i.e., a program to simulate a nonlinear dynamic system). We explained in the last chapter how to simulate a nonlinear system; hence, all we need to do here is explain how to simulate the fuzzy controller.

Fuzzy Controller Arrays and Subroutines

The fuzzy controller can be programmed in C, Fortran, Matlab, or virtually any other programming language. There may be some advantage to programming it in C since it is then sometimes easier to transfer the code directly to

an experimental setting for use in real-time control. At other times it may be advantageous to program it in Matlab since plotting capabilities and other control computations may be easier to perform there. Here, rather than discussing the syntax and characteristics of the multitude of languages that we could use to simulate the fuzzy controller, we will develop a computer program "pseudocode" that will be useful in developing the computer program in virtually any language. For readers who are not interested in learning how to write a program to simulate the fuzzy controller, this section will provide a nice overview of the steps used by the fuzzy controller to compute its outputs given some inputs.

Normalization and Scaling: We will use the ship steering example to illustrate the basic concepts on how to program the fuzzy controller. In particular, we will explain how to simulate the fuzzy control system shown in Figure 5.22. Notice that here we have added the gains g_1 and g_2 at the inputs to the fuzzy controller and g_0 at the output of the fuzzy controller. The reason for adding these is that they are often useful in tuning since they scale the horizontal input and output axes of the fuzzy controller. Hence, to simulate the fuzzy control system developed in the last section, we first "normalize" the input and output universes of discourse. For this example, this means that we simply change the membership functions to those shown in Figure 5.23 (i.e., normalize to an interval ± 1). With the indicated scaling gains in Figure 5.23 (i.e., the ones in the boxes) that are implemented as shown in Figure 5.22, we implement the membership functions shown in Figure 5.8.

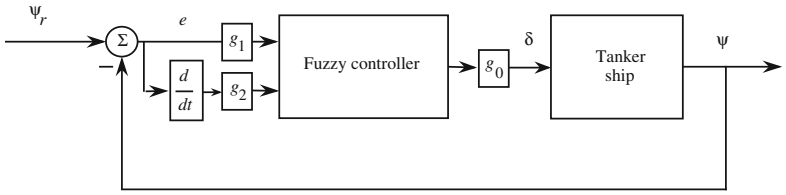

Figure 5.22: Fuzzy controller for tanker ship with scaling gains g_0, g_1, and g_2.

It is important to notice that a scaling gain g_1 on the input is equivalent to scaling the horizontal axis of the e universe of discourse by $1/g_1$ (yes, it is $1/g_1$; think about the fact that increasing g_1 changes, for instance, the meaning of "possmall" so that it quantifies *smaller* values of the error input that is passed through the gain g_1). In more detail, the scaling gain g_1 has the following effects:

- If $g_1 = 1$, there is no effect on the membership functions and there is no effect on the meaning of the linguistic values.
- If $g_1 < 1$, the membership functions are uniformly "spread out" by a factor of $1/g_1$ (notice that multiplication of each number on the e universe of discourse of Figure 5.23 by π which is $1/g_1$, gives you Figure 5.8 on

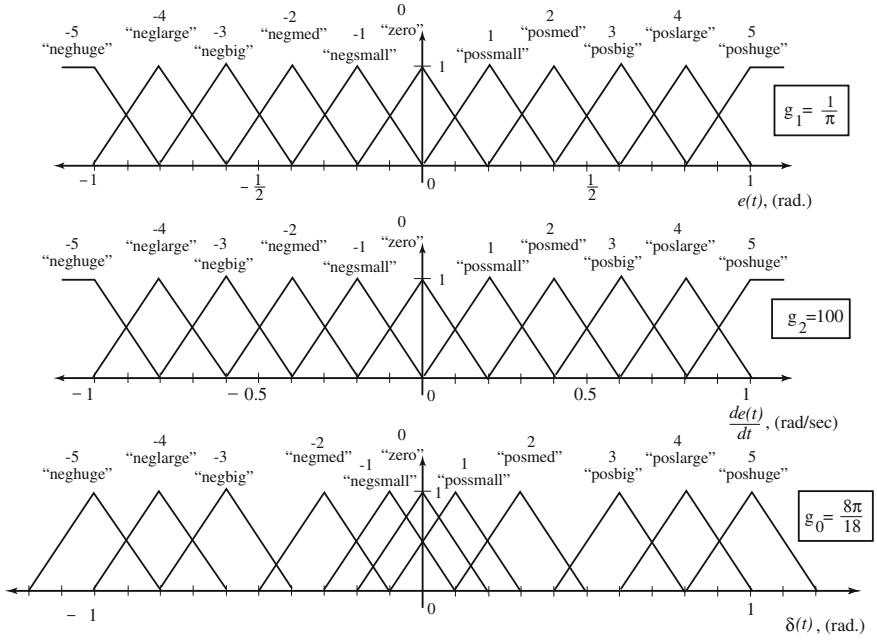

Figure 5.23: Normalized universes of discourse for fuzzy controller for tanker ship (and boxed values of the scaling gains give the original membership functions shown in Figure 5.8).

page 166). This changes the meaning of the linguistics so that, for example, "poslarge" is now characterized by a membership function that represents larger numbers.

- If $g_1 > 1$, the membership functions are uniformly "contracted." This changes the meaning of the linguistics so that, for example, "poslarge" is now characterized by a membership function that represents smaller numbers.

The scaling gain g_2 has similar effects, but for the $\dot{e}$ universe of discourse. However, for the output universe of discourse, the scaling is such that multiplying the output by the gain g_0 is the same as multiplying the horizontal δ axis by g_0.

Here, we will implement the membership functions in Figure 5.23 with the understanding that to get the membership functions in Figure 5.8 on page 166, all we need to do is multiply by scaling gains

$$g_1 = \frac{1}{\pi}, g_2 = 100, g_0 = \frac{8\pi}{18}$$

We will use the minimum operation to represent both the "and" in the premise and the implication (it will be obvious how to switch to using, for example, the product). We will use center of gravity defuzzification. At first we will make

no attempt to code the fuzzy controller so that it will minimize execution time or minimize the use of memory. However, after introducing the pseudocode, we will address these issues.

Subroutines: First, suppose that for convenience we use a different set of linguistic-numeric descriptions for the input and output membership functions than we used up till now. Rather than numbering them

$$-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5$$

we will renumber them as

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$$

so that we can use these as indices for arrays in the program. Suppose that we let the computer variable `x1` denote (notice that a different typeface is used for all computer variables) $e(t)$, which we will call the first input, and `x2` denote $\dot{e}(t)$, which we will call the second input. Next, we define the following arrays and functions:

- Let `mf1[i]` (`mf2[j]`) denote the value of the membership function associated with input 1 (2) and linguistic-numeric value `i` (`j`). In the computer program, `mf1[i]` could be a subroutine that computes the membership value for the i^{th} membership function given a numeric value for the first input `x1` (note that in the subroutine we can use simple equations for lines to represent triangular membership functions). Similarly for `mf2[j]`.
- Let `rule[i,j]` denote the center of the consequent membership function of the rule that has linguistic-numeric value "`i`" as the first term in its premise and "`j`" as the second term in its premise. Hence `rule[i,j]` is essentially a matrix that holds the body of the rule base table shown in Table 5.2. In particular, for the tanker ship we have `rule[i,j]` as:

$$\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 \\
1 & 1 & 1 & 1 & 1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 \\
1 & 1 & 1 & 1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 \\
1 & 1 & 1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 \\
1 & 1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 \\
1 & 0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 \\
0.8 & 0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 & -1 \\
0.6 & 0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 & -1 & -1 \\
0.3 & 0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 & -1 & -1 & -1 \\
0.1 & 0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 & -1 & -1 & -1 & -1 \\
0 & -0.1 & -0.3 & -0.6 & -0.8 & -1 & -1 & -1 & -1 & -1 & -1
\end{bmatrix}$$

  (recall that we will scale this matrix of centers by $g_0 = \frac{8\pi}{18}$ after we compute the output of the fuzzy controller).
- Let `prem[i,j]` denote the certainty of the premise of the rule that has linguistic-numeric value "`i`" as the first term in its premise and "`j`" as the second term in its premise given the inputs `x1` and `x2`.

- Let `areaimp[c,h]` denote the area under the output membership function with center `c` that has been chopped off at a height of `h` by the minimum operator. Hence, we can think of `areaimp[c,h]` as a subroutine that is used to compute areas under the membership functions for the implied fuzzy sets.

Fuzzy Controller Pseudocode

Using these definitions, consider the pseudocode for a simple fuzzy controller that is used to compute the fuzzy controller output given its two inputs:

```
1. Obtain x1 and x2 values
   (Get inputs to fuzzy controller)

2. Compute mf1[i] and mf2[j] for all i, j
   (Find the values of all membership functions given the values
   for x1 and x2)

3. Let num=0, den=0
   (Initialize the COG numerator and denominator values)

4. For i=1 to 11, For j=1 to 11,
   (Cycle through all areas to determine COG)

    prem[i,j]=min[mf1[i],mf2[j]]

    num=num+rule[i,j]*areaimp[rule[i,j],prem[i,j]]
       (Compute numerator for COG)

    den=den+areaimp[rule[i,j],prem[i,j]]
       (Compute denominator for COG)

5. Next i, Next j

6. Output ucrisp=num/den
   (Output the value computed by the fuzzy controller)

7. Go to Step 1.
```

To learn how this code operates, define each of the functions and arrays for the ship steering example and show how to compute the fuzzy controller output for the same (and some different) inputs used in the previous section. Following this, develop the computer code to simulate the fuzzy controller for the ship steering problem and verify that the computations made by the computer match the ones made by hand.[2]

We do not normally recommend that initially you use only the computer-aided design (CAD) packages for fuzzy systems since these tend to remove you from understanding the real details behind the operation of the fuzzy controller.

[2]One way to start with the coding of the fuzzy controller is to start with the code that is available for downloading at the Web site described in the Preface.

However, after you have developed your own code and fully understand the details of fuzzy control, we do advise that you use (or develop) the tools you believe are necessary to automate the process of constructing fuzzy controllers.

Aside from the effort that you must put into writing the code for the fuzzy controller, there are the additional efforts that you must take to initially type in the rule base and membership functions and possibly modify them later (which might be necessary if you need to perform redesigns of the fuzzy controller). For large rule bases, this effort could be considerable, especially for initially typing the rule base into the computer. While some CAD packages may help solve this problem, it is not hard to write a computer program to generate the rule base, because there are often certain regular patterns in the rule base.

Also notice that since there is a proportional correspondence between the input linguistic-numeric values and the values of the inputs, you will often find it easy to express the input membership functions as a nonlinear function of their linguistic-numeric values. Another trick that is used to make the adjustment of rule bases easier is to make the centers of the output membership functions a function of their linguistic-numeric indices.

Real-Time Implementation Issues

When it comes to implementing a fuzzy controller, you often want to try to minimize the amount of memory used and the time that it takes to compute the fuzzy controller outputs given some inputs. The pseudocode in the last section was not written to exploit certain characteristics of the fuzzy controller that we had developed for the ship; hence, if we were to actually implement this fuzzy controller and we had severe implementation constraints, we could try to optimize the code with respect to memory and computation time.

Computation Time: First, we will focus on reducing the amount of time it takes to compute the outputs for some given inputs. Notice the following about the pseudocode:

- We compute `prem[i,j]` for all values of `i` and `j` (121 values) when for our fuzzy controller for the ship, since there are never more than two membership functions overlapping, there will be at most four values of `prem[i,j]` needed (the rest will have zero values and hence will have no impact on the ultimate computation of the output).
- In a similar manner, while we compute `areaimp[rule[i,j],prem[i,j]]` for all `i` and `j`, we only need four of these values.
- If we compute only four values for `areaimp[rule[i,j],prem[i,j]]`, we will have at most four values to sum up in the numerator and denominator of the COG computation (and not 121 for each).

At this point, from the view of computational complexity, the reader may wonder why we even bothered with the pseudocode of the last section since it

appears to be so inefficient. However, the code is only inefficient for the chosen form for the fuzzy controller. If we had chosen Gaussian-shaped (i.e., or some other bell-shaped) membership functions for the input membership functions, then no matter what the input was to the fuzzy controller, all the rules would be on so all the computations shown in the pseudocode were necessary and not too much could be done to improve on the computation time needed. Hence, if you are concerned with real-time implementation of your fuzzy controller, you may want to put constraints on the type of fuzzy controller (e.g., membership functions) you construct.

It is important to note that the problems with the efficiency of the pseudocode highlighted above become particularly acute when there are many inputs to the fuzzy controller and many membership functions for each input, since the number of rules increases exponentially with an increase in the number of inputs (assuming all possible rules are used, which is often the case). For example, if you have a two-input fuzzy controller with 21 membership functions for each input, you will have $21^2 = 441$ rules, and you can see that if you increase the number of inputs, this number will quickly increase.

To reduce computation time, most of which is used for finding which rules are on, it is important to recognize that only a few rules "near each other" are on at any one time.

How do we overcome this problem? Assume that you have defined your fuzzy controller so that at most two input membership functions overlap at any one point, as we had for the ship example. The trick is to modify your code so that it will compute only four values for the premise membership functions, only four values for areas of implied fuzzy sets, and hence, have only four additions in the numerator and denominator of the COG computation. There are many ways to do this. For instance, you can have the program scan `mf1[i]` beginning at position zero until a nonzero membership value is obtained. Call the index of the first nonzero membership value "`istar`." Repeat this process for `mf2[j]` to find a corresponding "`jstar`." The rules that are on are the following:

```
rule[istar,jstar]
rule[istar,jstar+1]
rule[istar+1,jstar]
rule[istar+1,jstar+1]
```

provided that the indicated indices are not out of range. If only the rules identified by the indices of the premises of these rules are used in the computations, then we will reduce the number of required computations significantly, because we will not be computing values that will be zero anyway (notice that for the ship example, there will be one, two, or four rules on at any one time, so there could still be a few wasted computations). Notice that even in the case where there are many inputs to the fuzzy controller *the problem of how to code efficiently reduces to a problem of how to determine the set of indices for the rules that are on.* So that you may fully understand the issues in coding the controller in an efficient manner, we challenge you to develop the code for an n-input fuzzy controller that will exploit the fact that only a hypercubical block of 2^n rules will be on at any one time (provided that at most two input membership functions overlap at any one point).

Memory Requirements: Next, we consider methods for reducing memory requirements. Basically, this can be done by recognizing that it may be possible to compute the rule base at each time instant rather than using a stored one. Notice that there is a regular pattern to the rule base for the ship; since there are at most four rules on at any one time, it would not be hard to write the code so that it would actually generate the rules while it computes the controller outputs. It may also be possible to use a memory-saving scheme for the output membership functions. Rather than storing their positions, there may be a way to specify their spacing with a function so that it can be computed in real-time. For large rule bases, these approaches can bring a huge savings in memory (however, if you are working with adaptive fuzzy systems where you automatically tune membership functions, then it may not be possible to use this memory-saving scheme). We are, however, gaining this savings in memory at the expense of possibly increasing computation time.

Finally, note that while we focus here on the real-time implementation issues by discussing the optimization of *software*, you could consider redesigning the *hardware* to make real-time implementation possible. Implementation prospects could improve by using a better microprocessor or signal processing chip. An alternative would be to investigate the advantages and disadvantages of using a "fuzzy processor" (i.e., a processor designed specifically for implementing fuzzy controllers). Of course, many additional issues must be taken into consideration when trying to decide if a switch in computing technology is needed. Not the least among these are cost, durability, and reliability.

5.3.2 Fuzzy Controller Tuning for the Tanker Ship

We will start out with the controller that we developed earlier and illustrate some basic ideas (from conventional control) that are often used to tune fuzzy controllers. In particular, note that increasing g_1 is analogous to increasing the proportional gain in a PD controller (i.e., it will often make the system respond faster, but may cause overshoot). Increasing the gain g_2 is analogous to increasing the derivative gain in a PD controller which tends to give the controller a better predictive capability and hence helps it avoid overshooting constant reference set points. Notice, also, that increasing g_0 has an effect of increasing the "gain in the loop" so it can be used to speed up the response.

Performance for the First Guess

First, consider the implementation of the fuzzy controller for ship steering developed in the previous sections which we will refer to as our "first guess." The closed-loop response, using the ship model specified in the previous section, is shown in Figure 5.24 (note that we use $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{1}{\pi}$, and $g_2 = 100$ as scaling gains for our membership functions, which were normalized to the interval ± 1, to implement the membership functions in Figure 5.8). Note that while the response is at least tracking the step changes eventually, there is a significant amount of overshoot.

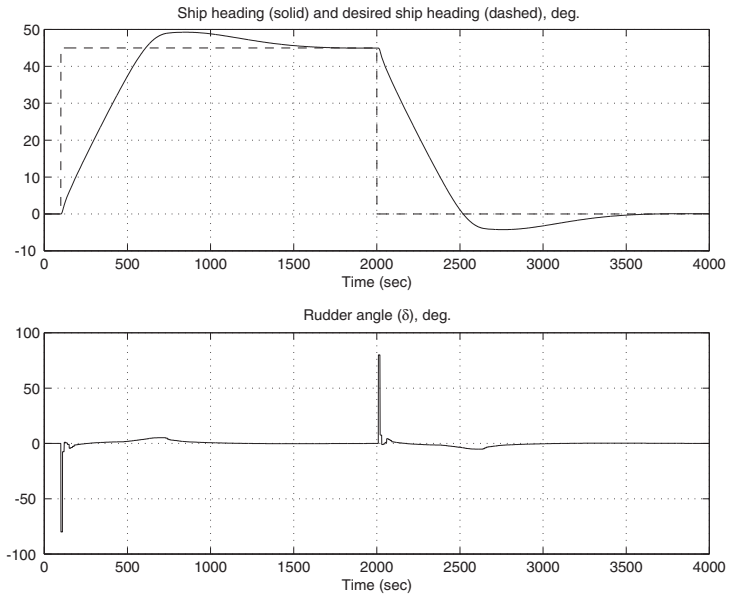

Figure 5.24: Response of fuzzy controller for tanker ship steering, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{1}{\pi}$, and $g_2 = 100$.

Tuning the Derivative Gain to Reduce Overshoot

We often use standard heuristic ideas from tuning conventional controllers for tuning fuzzy controllers.

Using standard ideas from tuning of conventional controllers (e.g., proportional-integral-derivative (PID) controllers), to reduce the overshoot, we should increase the gain on the derivative term (so that the controller gets more capability to "predict where the response is going"). To do this we choose $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{1}{\pi}$, and $g_2 = 200$ and get the response in Figure 5.25, where we see that we have indeed reduced the overshoot. Unfortunately, however, this also reduced the response time of the system (i.e., it "slowed" the system).

Tuning the Proportional Gain to Decrease the Response Time: Finding "Good" Scaling Gains

Next, we seek to choose a good set of scaling gains by speeding up the response from the previous case. To do this we increase the gain on the proportional term so that we increase the speed of the response and hence reduce the response time. When we do this, however, this can cause some overshoot, so we also increase the gain on the derivative term to avoid that. In particular, choose $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$ to get a faster response with very little overshoot as seen in Figure 5.26. We take this set of gains as "good" values in that we consider the response that results from them to be good. Notice that we achieved all our tuning via the scaling gains, although this is certainly not possible in all applications.

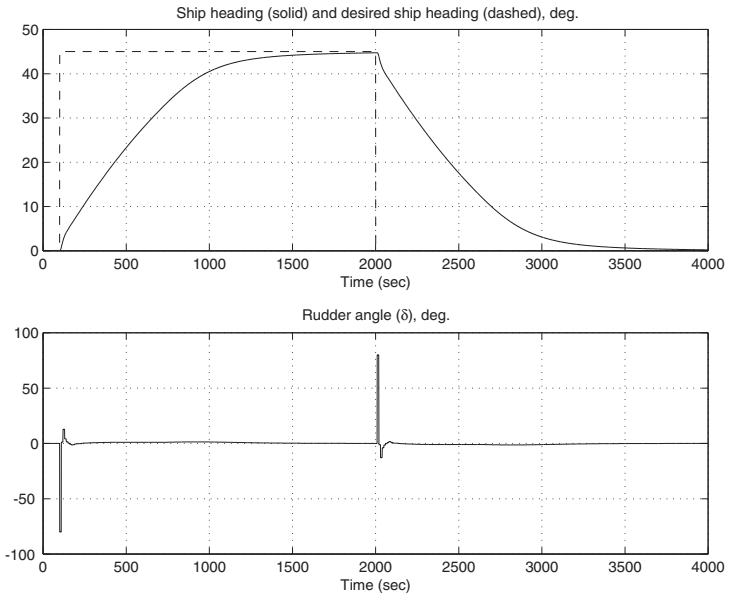

Figure 5.25: Response of fuzzy controller for tanker ship steering, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{1}{\pi}$, and $g_2 = 200$.

The Resulting Nonlinear Control Surface

The fuzzy controller implements a nonlinear input-output map. Rule construction and tuning shapes this map.

To achieve this performance, the fuzzy controller implements a nonlinearity that is shown in Figure 5.27. Notice that this surface is another way to view the captain's expertise in ship steering (compare the list of the captain's steering expertise developed earlier to the shape of the surface; for instance, explain why the slope of the surface changes in the way it does).

Note that the control surface for a simple proportional-derivative (PD) controller is a plane in three dimensions. With the proper choice of the PD gains, the linear PD controller can be made to have basically the same shape as the fuzzy controller near the origin. Hence, in this case the fuzzy controller will behave similarly to the PD controller provided its inputs are small. However, notice that there is no way that the linear PD controller can achieve a nonlinear control surface of the shape shown in Figure 5.27 (this is not surprising considering the complexity difference of the two controllers).

It is useful to notice that there is a type of interpolation that is performed by the fuzzy controller that is nicely illustrated in Figure 5.27. If you study the plot carefully, you will notice that the rippled surface is created by the rules and membership functions. For instance, if we kept a similar nonuniform distribution of membership functions for the input and outputs of the fuzzy system, but increased the number of membership functions, the ripples would correspondingly increase in number and the amplitude of the ripple would decrease. What is happening is that there is an interpolation between the rules. The output is

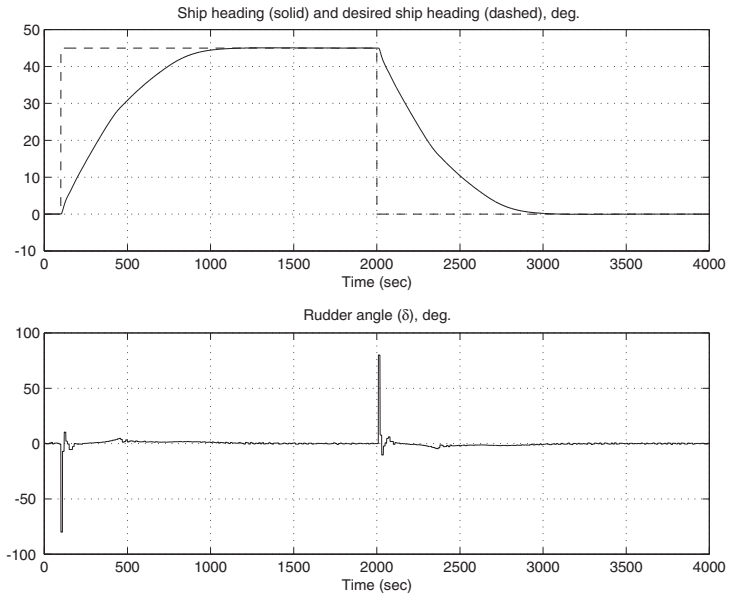

Figure 5.26: Response of fuzzy controller for tanker ship steering, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$.

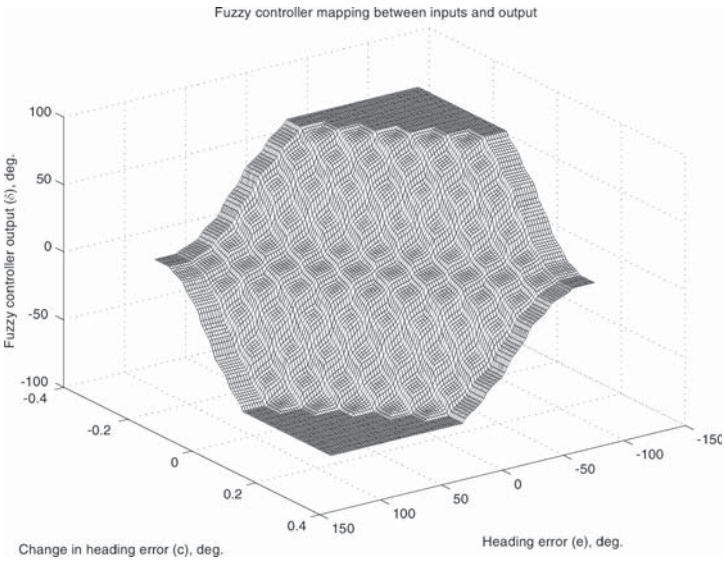

Figure 5.27: Nonlinear control surface implemented by the fuzzy controller, $g_0 = {}^{8\pi}$

an interpolation of the effects of the four rules that are on for the ship's fuzzy controller. For more general fuzzy controllers, it is important to keep in mind that this sort of interpolation is often occurring (but not always—it depends on your choice of the membership functions).

5.3.3 Design Concerns

While designing fuzzy controllers for practical problems you will encounter a whole variety of problems, not the least of which could be pragmatic issues in interfacing and communicating with the plant. In this section we outline some of these and offer ideas on how to solve them. Several of the design concerns we list are not specific to fuzzy control, but apply to any control system, including the ones considered later in this chapter. To illustrate several points we will use the tanker ship steering problem studied in the last section.

Understand the Control Problem

One key to developing a good solution to any problem is to make sure that you clearly understand the problem so that you are sure that you are solving the right problem! For control problems this means that you must do the following:

- *Obtain a good understanding of the plant:* It is critical that you gain a good understanding of the plant you are to control. Yes, this means understanding the physics of the problem and this may demand that you step outside your main area of expertise (e.g., to study thermodynamics, fluid mechanics, mechanics, circuit theory, etc.). Aside from returning to first principles, it may be beneficial to consult others who have operated the plant in the past or who have already developed a controller for it. It may be helpful to develop a simulation of the plant and study the effects of, for example, some inputs or disturbances on the output variables. Now, clearly one of the main advantages of fuzzy and expert control is that you do not explicitly need a mathematical model of a specific form to develop the controller; however, for some plants it is not too hard to develop an approximate mathematical model that can be very helpful in gaining an understanding of how to control the plant. Experience has shown that to develop good control systems you must use all the information you have about how to achieve good control. Some of this information may come in the form of rules from a human operator (or engineer) but other useful information can come from a mathematical model and this should not be ignored. Indeed, such a mathematical model will be needed for the implementation of a planning system.

- *Pay attention to plant constraints:* A particularly important part of the problem of obtaining a good understanding of the plant is to understand those plant characteristics that limit your ability to achieve high performance operation. Some typical limitations include the following:

  - Actuator saturation limits: All actuators have limits in which they can perform and these constrain the ways in which you can affect plant behavior. These limits come in the form of saturation limits on the magnitude of the input, limits on the rate of change of the input, etc.
  - Sensor noise: There is no perfect sensor. Better sensors cost more money. Noise on sensors limits the quality of information you can obtain about the plant and hence your ability to control the plant.
  - Plant dynamics: Unstable, highly nonlinear, nonminimum phase, and highly uncertain plants (i.e., those with significant noise and plant parameter variations), or plants with delays all provide unique challenges in control. The way each of these problems is manifested changes for different applications.

- *Develop appropriate specifications:* Sometimes the boss or customer is the one to provide the specifications of what they want in terms of performance. It is important to have a very clear understanding of the expectations for the plant in terms of typical measures of performance like the following:

  - Rise-time (amount of time for the output to get from 10% of the final value to 90% of the final value when there is a step input), overshoot (the amount the output increases above the final value of a step reference input), steady-state error (error between the plant output and commanded input as time goes to infinity).
  - Stability (e.g., for the ship steering problem, if you start the ship heading near a desired constant heading, will it move toward the reference heading and ultimately reduce the heading error to zero?), limit cycles (oscillations).
  - Performance robustness (e.g., how much can the plant be allowed to change before control system performance degrades significantly?), and stability robustness (e.g., how much can the plant be allowed to change before the system goes unstable?).

  If the requirements are unreasonable, you may have to return to the boss or customer and negotiate a reasonable set of specifications. If you find that more is possible than is being asked, then your company may have a competitive edge with the customer.

- *Consider if it is possible to redesign the plant:* For some control problems (e.g., aircraft control) there are significant efforts to design the plant so that it is easy to control. If you study the control problem (plant dynamics and control specifications) and find that the specifications cannot be met, another option may be to go back and redesign the plant so that the specifications can be met. This may entail adding a sensor, purchasing a better actuator, or even making structural changes to the plant to remove challenging nonlinearities.

- *Study similar problems:* There is an ever-expanding literature on the development of control systems and there may be some similar work done that is in the public domain that could be useful to you.
- *Try the simplest thing:* Trying the simplest thing first is good engineering practice and it may teach you something about the control problem. Fuzzy, expert, or planning systems-based control is probably not the first thing to try when implementing a control system. Even if you do not have a model, it is simple to develop a PID controller, or indeed a P controller (proportional controller), that can be tuned manually. The computations for simple proportional control involve, for instance, forming a difference between the reference input and plant output, and multiplying this difference by a gain. The numerical operations to implement a fuzzy controller are clearly more complex (although it is not always the case that they are more complex than a conventional controller).

Proper Rule Base Construction

Assuming that you are using fuzzy control, one of the most critical steps in the design process is the choice of the rule base. It is therefore very important to pay significant attention to this problem. The main sources of information for rule base construction are the following:

- Interviews of human plant operators (or learning how to operate the plant yourself).
- A good understanding of the plant, the constraints imposed by it, and the closed-loop specifications that you are trying to achieve.
- Modeling and simulation studies.
- Past development of controllers for the same plant (or similar ones).
- Controller implementation studies for controllers that ultimately do not adequately achieve the specifications (e.g., the controller that you are trying to replace in updating a control system to achieve higher performance).

There are several issues to pay attention to in rule base construction, including the following:

- *Conflicting rules:* Most often (but not always), the rules in the rule base should not conflict with one another (e.g., there should not be two rules that apply in the same situation that say to do two very different things). Note, however, that conflicting rules can be used in a fuzzy controller since, depending on how you define the inference mechanism, it will simply interpolate between the two different conclusions (e.g., in the ship steering fuzzy controller four different rules may come on that say to do somewhat different things and defuzzification combines the recommendations that are in a sense conflicting, if only mildly).

- *Completeness:* You must define the rule base so that there is at least one rule that is "on" at each time (and if there is, we will call it a "complete rule base"). This means that there is a sort of complete coverage of the input space of the fuzzy controller so that there is a premise membership function with nonzero certainty for all possible values of the inputs. For an expert controller, it must be the case that there is at least one rule with a premise term that evaluates to true at each time instant. Note that in the ship steering example, no matter what the values of e and $\dot{e}$ are, there is a premise membership function that has nonzero certainty so that there is always at least one rule on. If you do not have a complete rule base then, depending on how you define the fuzzy controller, it can be that the denominator in the defuzzification formula will have a zero value so that you will not be able to compute an explicit output (and your software will return a "divide by zero" error).

Reducing Controller Complexity

For simple academic problems, the complexity of the fuzzy controller is rarely a problem, especially when only simulation examples are considered. The problem is, however, that for real applications there are often limitations on computing power (memory and "throughput"), so it is important to carefully consider how to reduce the computations necessary for implementations. There are two fundamental reasons why complexity arises in fuzzy and expert controllers:

- *Complex nonlinear maps:* For challenging applications where you have spent a significant amount of time tuning the rule base, it is likely that the resulting controller surface has a very interesting and complex shape, and that this shape is critical in meeting the performance specifications. Complex nonlinear maps take significant computations to implement, so to get higher performance control you have to pay for it in controller complexity (you do not get something for nothing).

- *Exponential increase in number of rules:* Recall that in Section 5.2.3, we analyzed the number of parameters needed to define a fuzzy system for a given number of inputs and membership functions. We found that if you define rules for all possible combinations of linguistic values in the premises, then there is an exponential number of rules (similar analyses hold for expert controllers also). For example, for our ship steering problem with two inputs and eleven membership functions on each input universe of discourse there are $11^2 = 121$ possible rules. Hence, increasing the number of linguistic values or inputs causes large increases in the number of rules and hence the complexity of the fuzzy controller (e.g., going from using e and $\dot{e}$ as inputs to also using $\int_0^t e(\tau)d\tau$, with eleven membership functions on the $\int_0^t e(\tau)d\tau$ universe of discourse, would result in $11^3 = 1331$ rules).

Methods to reduce complexity are as numerous as there are applications since each is a special case and there are often methods to simplify the computations. There are, however, some general approaches to reducing complexity and these are listed next:

1. In some cases, upon further study, you may be able to determine that rule base completeness can be achieved with fewer rules simply because it may be the case that certain combinations of the inputs are not possible. In this case the corresponding rules can be removed since they will never be used anyway.

2. You can simply try to reduce the number of linguistic values so that the total number of possible rules is reduced. For example, for the tanker ship it is possible to get reasonably good performance (at least for nominal conditions) using only nine rules (three membership functions on each of the two input universes of discourse). Realize, however, that additional rules allow for the implementation of more complex nonlinear control surfaces, that can then result in higher performance operation. The key is to determine the minimum number of rules that still allows for the implementation of a control surface that can achieve adequate performance. In some cases you may have to go back to the customer and indicate that if you are only allowed a certain amount of computing power, then only a certain performance level is possible.

3. For the case of MIMO fuzzy controllers, study the problem carefully to determine if you truly need all the inputs for each of the fuzzy controllers for each plant input. Elimination of one input, for even one MISO controller, can result in significant savings.

4. Sometimes you may want to use some type of "multi-stage" fuzzy controller where, for example, there are two inputs to each of two controllers and their outputs are combined by a third fuzzy system that provides the input to the plant. In this case we will implement three two-input fuzzy controllers rather than one four-input fuzzy controller (which for some applications can make a big difference). This approach tends to be highly application specific but the principle is valid: try to reduce the number of inputs by cascading fuzzy controllers.

5. Another approach to reduction is to use one fuzzy controller to specify parameters in another. For instance, if you were to develop a controller for the ship that also took as an input the ship speed u, one approach would be to simply use a three-input fuzzy controller (where the rule base would indicate that for faster ship speeds a smaller rudder angle input is needed since the ship is easier to steer when it is moving fast). Another approach, one that avoids the implementation of a three-input fuzzy controller, is to use the two-input fuzzy controller we already developed for the ship but add another single-input single-output fuzzy controller with the ship speed as an input that specifies the amount of correction to the rudder

angle for different ship speeds. Now, this approach does not allow one to make coordinated control actions (e.g., different rudder corrections for different e and $\dot{e}$ and ship speeds), but it may be sufficient to solve the problem. Again, there are many approaches to reducing complexity using this approach since they are application-dependent.

Effects of Disturbances and Plant Changes

It is important to evaluate the performance of the fuzzy control system under adverse conditions.

Plant parameter variations, disturbances, speed changes, and sensor noise all affect our ability to achieve good control. In this section we will use the ship example to illustrate their effect on heading regulation performance when a fuzzy controller is used (this section parallels the simulation studies for the multilayer perceptron and radial basis function controllers for the ship in Sections 4.3 and 4.5; here, we use the same types of variations as we did in those sections). Our intent, however, is to alert the reader to these issues so that they can be taken into account in the design process.

First, we will consider the performance of the fuzzy controller when the ship is under "full" conditions. Figure 5.28 shows how the fuzzy control system, which was tuned for ballast conditions, performs for full conditions. We see that there now is a bit of overshoot in the ship heading since a lighter boat steers easier. We see that plant parameter variations can affect performance.

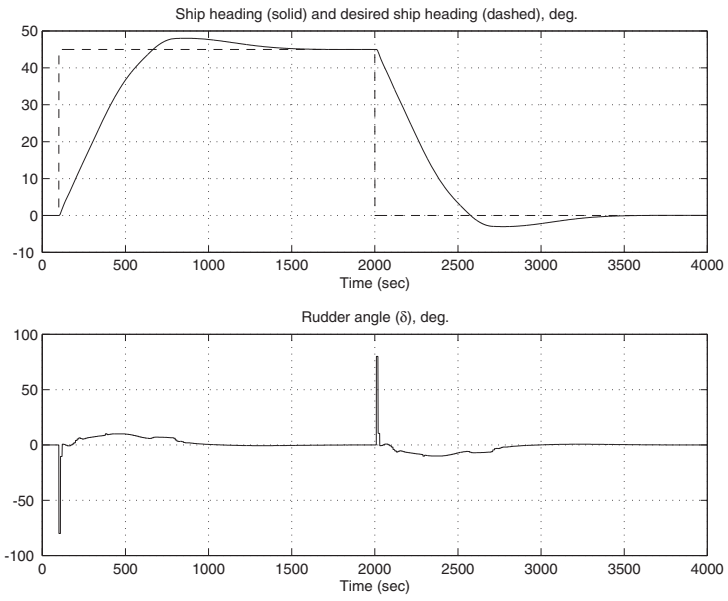

Figure 5.28: Response of fuzzy controller for tanker ship steering, "full" conditions, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$.

Next, consider the effect of a wind disturbance on the ship. If we use $g_0 = \frac{8\pi}{18}$,

$g_1 = \frac{2}{\pi}$, and $g_2 = 250$ (i.e., the good tuned values), we get the response in Figure 5.29. We see that the wind affects our ability to achieve very good regulation of the ship heading since it causes a 1 to 2 degree variation in the tracking of the desired heading.

Adverse conditions generally degrade performance; however, good controller designs minimize such performance degradations.

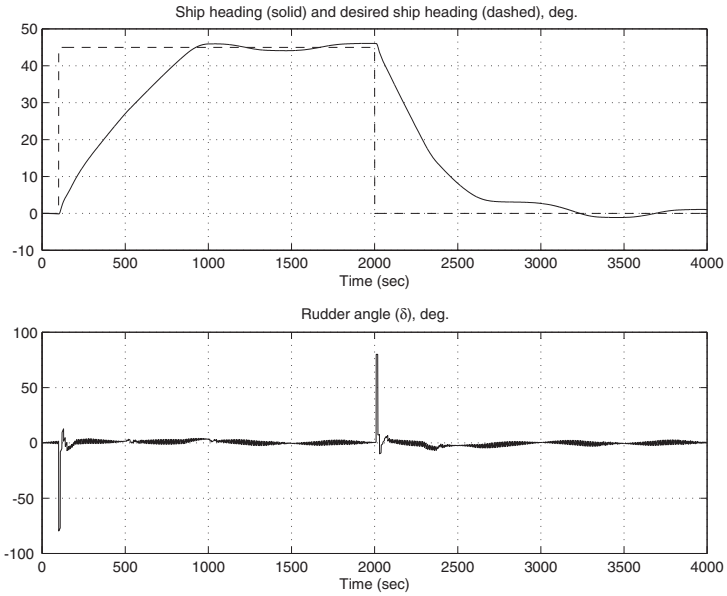

Figure 5.29: Response of fuzzy controller for tanker ship steering, wind disturbance, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$.

If you use, for instance, an additive sensor noise uniformly distributed on $[-0.01, 0.01]$, there is little effect on the response so we do not show the plot (of course, if you get sensors with worse performance characteristics, then you will expect tracking errors to arise in an analogous manner to results for the wind).

Next, consider the effect of a speed change on our ability to steer the ship. If we use $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$ (i.e., the good tuned values), we get the response in Figure 5.30. We see that the speed decrease causes a significant overshoot in the response since the rudder is not as effective.

Tracking Error

Steady-state tracking error is the value

$$\lim_{t \to \infty} e(t)$$

and for most control problems we would like this to be as small as possible or zero when the reference input is, for example, a step change. Adding an integrator to the control loop is one approach that is often successful at reducing or eliminating steady-state error (since if the error is nonzero, the integrator's

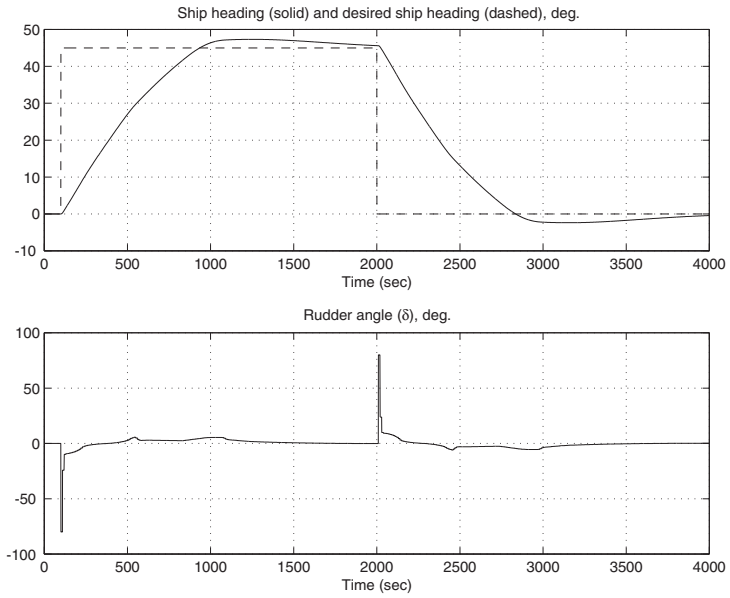

Figure 5.30: Response of fuzzy controller for tanker ship steering, speed decrease, $g_0 = \frac{8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$.

value gets larger, thus causing a larger control input to force the plant to move to reduce the error). In this ship example we did not need to add an integrator to the control loop by putting one in the controller since there is already an integration effect in the plant (note that if you hold the rudder angle input constant, the ship heading will tend off to infinity).

For fuzzy control design, since the plant and controller are nonlinear (you should not be trying to control a linear plant with a fuzzy controller since if the plant is truly linear all that is needed to succeed is a linear controller), we cannot be guaranteed that the addition of an integrator will help reduce steady-state error, but often it does. There are basically two ways to add an integrator to a fuzzy controller: as an input (to achieve, for instance, "PID fuzzy control," i.e., a fuzzy controller with P, I, and D inputs), or by adding an integrator to the output of the plant (which in some discrete time implementations some engineers do inherently by specifying that the output should be a change in the control variable, not an absolute value).

5.4 Stability Analysis

Here, we will be brief by simply providing some examples of how you can encounter limit cycles and instabilities for the tanker ship and a brief explanation of how to conduct stability analysis for fuzzy control systems.

5.4.1 Example: Stability and Limit Cycles in Ship Steering

Stability is often viewed as a fundamental property of a control system since if the system is unstable it is possible that the output response, and hence the tracking error (assuming a bounded reference input), grows without bound. For example, for the ship if you choose $g_0 = \frac{-8\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 250$ (notice minus sign), you can get an unstable response. (In this case, the controller moves the rudder in the wrong direction to try to reduce a heading error and each time it does this, it creates a bigger error.) This is a rather simple mechanism that provides instability but there can be very complex ones.

Improper choices for the rule base can result in a closed-loop system with limit cycles and instability.

Another type of tracking error that can result (that is often considered to be a type of instability) is when $e(t)$ is oscillating, for example, when the reference input is a constant and the output of the plant is a sinusoid. In many applications this is an undesirable characteristic. If you pick the wrong values of the scaling gains in the ship steering problem, you can get such oscillatory behavior. For example, if you pick $g_0 = \frac{2000\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 0.000001$ for the ship, you get the response shown in Figure 5.31 where we see that the oscillation characteristics are dependent on the magnitude of the reference input.

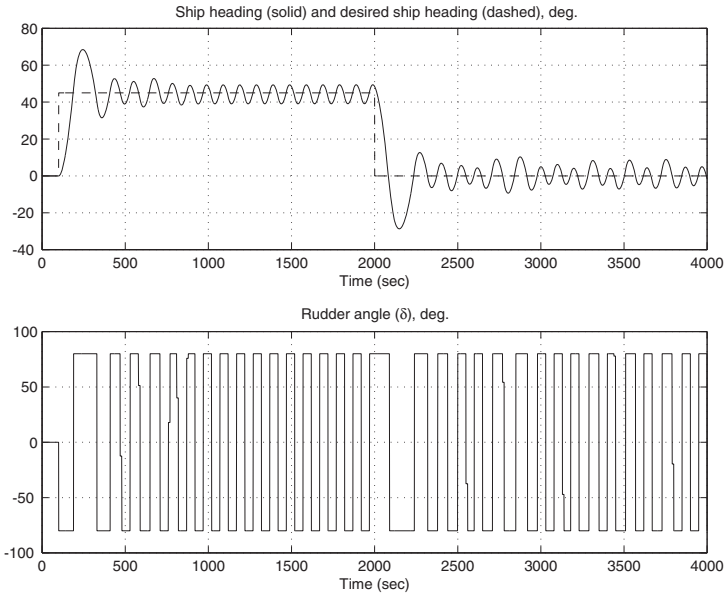

Figure 5.31: Response of fuzzy controller for tanker ship steering, $g_0 = \frac{2000\pi}{18}$, $g_1 = \frac{2}{\pi}$, and $g_2 = 0.000001$.

What is happening in the fuzzy control system to achieve this type of behavior? Usually, it is because some gains are set too large (or small) and the input or output signals are oscillating between their maximum values, forcing

the plant to oscillate also (i.e., this is a "controller-induced oscillation" in this case). Clearly, the designer must be alerted to this possibility and try to avoid it. Methods to avoid this problem typically involve careful choice of rule bases and scaling gains.

5.4.2 Discussion: Lyapunov Stability Analysis of Fuzzy Control Systems

Lyapunov stability analysis can be useful for verification of fuzzy control systems.

The central issue in fuzzy controller design is to obtain good insights into how the plant behaves in order to determine how to shape the nonlinear function that is implemented by the fuzzy controller. Of course, this nonlinear function then affects the closed-loop dynamics. To characterize and analyze exactly how the nonlinearity affects the closed-loop stability properties, you can use mathematical stability analysis just as we did for the neural controller in the last chapter. Moreover, the reader may be interested to know that Lyapunov's first method (via linearization), absolute stability, and describing function analysis can be performed (see the "For Further Study" section at the end of this part for more information).

How exactly do you perform stability analysis via Lyapunov's direct method? Consider the simple example in Section 4.6.5 on page 147. Note that

$$u = F(x)$$

could be specified as a fuzzy controller so that $F(0) = 0$, $F(x)$ is smooth, and for some scalar $\beta > 0$,

$$\begin{aligned} F(x) &> -\beta x,\ x < 0 \\ F(x) &< -\beta x,\ x > 0 \end{aligned}$$

How do we construct a rule base so that this is the case? We will provide a problem of this type to the reader in Exercise 5.8. Basically, however, when you get familiar with the types of input-output mappings that are generated for certain choices of rule bases and membership functions, you will see how to construct nonlinear control surfaces with different shapes.

5.5 Expert Control

An expert system is a computer program that is designed to emulate a human's skills in a specific problem domain. If it is designed to emulate the expertise of a human in performing control activities, it is called an "expert controller." When the expert controller is connected to a plant, the closed-loop system is called an expert control system (see Figure 5.32). Traditionally, the expert system has been split into two components: the knowledge base and inference mechanism. The knowledge base is simply a generalization of the rule base in a fuzzy system where more general types of information can be characterized. Correspondingly, the inference mechanism is a generalization of the inference mechanism in a fuzzy

controller that can incorporate other reasoning strategies. Hence, conceptually the expert controller is closely related to the fuzzy controller in its structure and function. Moreover, the design philosophy used to construct the expert controller is similar to that of the fuzzy controller. The main differences between the two approaches lie in the details of how the knowledge base and inference mechanism are constructed.

General representations of knowledge and inference can be used for emulating sophisticated control strategies.

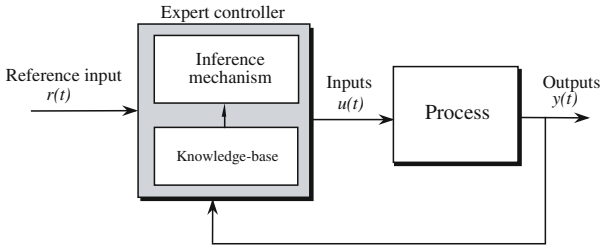

Figure 5.32: Expert control system.

5.5.1 General Knowledge Representations

The knowledge base in the expert controller could be a rule base, but is not necessarily so. It could be developed using other knowledge-representation structures, such as frames, semantic nets, causal diagrams, and so on (see the "For Further Study" section at the end of this part for information on these). Here, we will simply show how the form of the rules used in fuzzy controllers can be generalized and used in expert control.

The rule premises can be defined in much more general ways. For instance, any type of predicate logic can be used that can include any kind of Boolean logic, functions, relations, and existential quantifiers ("for all," and "there exists"). For example, a rule may have the form:

$$\begin{gathered}\textbf{If } e(t) > 2 \textbf{ or} \text{ there exists a time over the last} \\ 10 \text{ sec. where } \tfrac{de(t)}{dt} \leq 0.5 \\ \textbf{Then } u(t) = 2.\end{gathered}$$

Testing the validity of the premise can be defined in many ways, but normally the standard rules of logic are used (similar to how the premise part of a standard computer "if-then" statement is tested). Moreover, degrees of matching the premises to the current situation can be used in an analogous way to how it is in fuzzy systems.

The specific types of rules needed for control depend on the application being considered and often it requires significant expertise with the plant to develop an effective set of rules. Indeed, for practical applications this is typically an iterative trial-and-error process and may involve a team of process experts to test and develop the rule base. Conceptually, however, the synthesis of the rule base proceeds in basically the same way as for the fuzzy control methodology.

Finally, it is interesting to note that in practical control systems there are often rules used for "exception handling" and special situations. These rules sometimes override a currently operating controller (e.g., a PID controller) to take appropriate actions under special situations. The inclusion of such rules in controllers should not be viewed as a rare occurrence in the development of a practical control system; control rules are often present, and sometimes are significantly more difficult to develop than the "conventional" (e.g., PID) part of the overall control system.

5.5.2 General Inference Mechanisms

The inference mechanism in the expert controller is more general than that of the fuzzy controller. It can use more sophisticated matching strategies to determine which rules should be allowed to fire. Also, it can use more elaborate inference strategies. For instance, some expert systems use

- "Refraction," where if a rule has fired recently, it may not be allowed back into the "conflict set" (i.e., the set of rules that are allowed to fire).
- "Recency," where rules that were fired most recently are given priority in being fired again (sometimes a valid approach since such rules may be most relevant to the current situation).
- "Priority schemes" where certain rules are a priori given higher priority to fire if they are both in the conflict set. It is also possible to dynamically assign priority.

Verification of correct behavior of general reasoning systems used as feedback controllers is important and challenging.

It is in fact the case that an expert system is in a sense more general than a fuzzy system since it can be shown that a single rule in an expert controller can be used to represent an entire fuzzy controller. To see this, note that a single fuzzy controller can be represented with a single static input-output map. Then, a single rule in an expert controller can represent that mapping. If an entire set of fuzzy controllers is represented as a set of such rules, then the resulting expert controller will reason about how to successively apply fuzzy controllers at each time step.

5.5.3 Stability Analysis of Expert Control Systems

Just as for neural and fuzzy control systems, it is possible to analyze qualitative properties of expert control systems. For instance, a discrete time formulation can be used to study the following properties:

- Stability in the sense of Lyapunov that may characterize how well the expert system can stay focused on (attend to) a control task, or boundedness of plant variables in the closed-loop when an expert controller is used.
- "Reachability" properties where, for instance, search algorithms can be used to test if the expert controller can drive the plant into some state (e.g., the goal state).

- Cyclic properties where the expert system may get stuck in an infinite loop (circular reasoning) and hence not be able to achieve its goal.

Here, we will not develop the mathematical models and show explicitly how to conduct stability analysis for expert control systems since it basically follows the same conceptual approach as for neural or fuzzy control systems. We do, however, provide more references in the "For Further Study" section at the end of this part for the interested reader, and Design Problem 5.3 where the reader is asked to conduct stability analysis of a simple expert control system.

5.6 Hierarchical Rule-Based Control Systems

There are a variety of ways to construct hierarchical fuzzy or expert control systems. For instance, you could use the following approaches:

Knowledge and inference are sometimes conveniently represented via hierarchies.

- You could use a rule-based (fuzzy or expert) system as a supervisor for the operation of a rule-based controller. This supervisor could monitor certain plant conditions and modify the rules to try to maintain good performance. It may be more convenient to implement the system as two rule-based systems, rather than a single one that takes in all the inputs that the two systems do, and outputs the input to the plant (e.g., it may be more computationally efficient, or this may be the way that the human operator thinks about controlling the plant).
- You could use a rule-based system to supervise the operation of an adaptive control system. This possibility will be discussed in more detail in Section 9.4.5.
- Sometimes multiple layers of such supervision could be needed.

There are still other possibilities. For instance, you can think of the hierarchy in Figure 1.11 and suppose that each block is a fuzzy or expert system. The blocks at the low level may be standard fuzzy controllers. The blocks at the coordination level may contain fuzzy systems with rules about how to coordinate the operation of the fuzzy controllers at the execution level, and an expert system at the management level could supervise both the levels below it. In this context you may think of using rules at the higher levels to turn on appropriate rules at the low levels (some would think of this as pruning the rules at the lower levels).

5.7 Exercises and Design Problems

Exercise 5.1 (Defining Membership Functions: Single Universe of Discourse): In this problem you will study how to represent various concepts and quantify various relations with membership functions. For each part below, there is more than one correct answer. Provide one of these and justify your choice in each case.

(a) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all people of medium height.

(b) Draw a membership function that quantifies the statement "the number x is near 10."

(c) Draw a membership function that quantifies the statement "the number x is less than 10."

Exercise 5.2 (Defining Membership Functions: Multiple Universes of Discourse): In this problem you will study how to represent various concepts and quantify various relations with membership functions when there is more than one universe of discourse. Use minimum to quantify the "and." For each part below, there is more than one correct answer. Provide one of these and justify your choice in each case. Also, in each case, provide the three-dimensional plot of the membership function.

(a) Draw a membership function (and hence define a fuzzy set) that quantifies the set of all people of medium height who are "tan" in color (i.e., tan *and* medium-height people). Think of peoples' colors being on a spectrum from white to black.

(b) Draw a membership function that quantifies the statement "the number x is near 10 and the number y is near 2."

Exercise 5.3 (Fuzzy Sets): There are many concepts that are used in fuzzy sets that sometimes become useful when studying fuzzy control. The following problems introduce some of the more popular fuzzy set concepts that were not treated earlier in the chapter.

(a) The "support" of a fuzzy set with membership function $\mu(x)$ is the (crisp) set of all points x on the universe of discourse such that $\mu(x) > 0$ and the "α-cut" is the (crisp) set of all points on the universe of discourse such that $\mu(x) > \alpha$. What is the support and 0.5-cut for the fuzzy set shown in Figure 5.5 on page 163?

(b) The "height" of a fuzzy set with membership function $\mu(x)$ is the highest value that $\mu(x)$ reaches on the universe of discourse on which it is defined. A fuzzy set is said to be "normal" if its height is equal to one. What is the height of the fuzzy set shown in Figure 5.5 on page 163? Is it normal? Give an example of a fuzzy set that is not normal.

(c) A fuzzy set with membership function $\mu(x)$ where the universe of discourse is the set of real numbers is said to be "convex" if and only if

$$\mu(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu(x_1), \mu(x_2)\} \tag{5.8}$$

for all x_1 and x_2 and all $\lambda \in [0, 1]$. Note that just because a fuzzy set is said to be convex does not mean that its membership function is a convex function in the usual sense. Prove that the fuzzy set

shown in Figure 5.5 on page 163 is convex. Prove that the Gaussian membership function is convex. Give an example of a fuzzy set that is not convex.

(d) A linguistic "hedge" is a modifier to a linguistic value such as "very" or "more or less." When we use linguistic hedges for linguistic values that already have membership functions, we can simply modify these membership functions so that they represent the modified linguistic values. Consider the membership function in Figure 5.5 on page 163. Suppose that we obtain the membership function for "error is very possmall" from the one for "possmall" by squaring the membership values (i.e., $\mu_{verypossmall} = (\mu_{possmall})^2$). Sketch the membership function for "error is very possmall." For "error is more or less possmall" we could use $\mu_{moreorlesspossmall} = \sqrt{\mu_{possmall}}$. Sketch the membership function for "error is more or less possmall."

Exercise 5.4 (Fuzzy Logic): There are many concepts that are used in fuzzy logic that sometimes become useful when studying fuzzy control. The following problems introduce some of the more popular fuzzy logic concepts that were not treated earlier in the chapter or were treated only briefly.

(a) The complement ("not") of a fuzzy set with a membership function μ has a membership function given by $\bar{\mu}(x) = 1 - \mu(x)$. Sketch the complement of the fuzzy set shown in Figure 5.5 on page 163.

(b) There are other ways to represent the conjunction "and" using fuzzy sets, different from the minimum and product that were introduced in the chapter. Let μ^1 and μ^2 denote two specific membership function values. Then, to represent "and," we could use the "bounded difference" (i.e., $\max\{0, \mu^1 + \mu^2 - 1\}$) and "drastic intersection" (where its value is μ^1 when $\mu^2 = 1$, μ^2 when $\mu^1 = 1$, and zero otherwise). Consider the membership functions shown in Figure 5.8 on page 166. Sketch the membership function for the premise "error is zero **and** change-in-error is possmall" when the bounded difference is used to represent this conjunction (premise). Do the same for the case when we use the drastic intersection. Compare these to the case where the minimum operation and the product were used (i.e., plot these also and compare all four).

(c) Fuzzy logic can be used to represent the disjunction ("or") of, for example, two premise terms. While there are many ways to represent "or" in fuzzy logic, the most popular one seems to be to simply use the maximum of the membership values. Consider the membership functions shown in Figure 5.8 on page 166. Sketch the membership function for "error is zero **or** change-in-error is possmall" when the maximum is used to represent this disjunction.

Exercise 5.5 (Matching, Inference, and Defuzzification: Hand Calculations): Suppose that for the tanker ship you use the membership

functions in Figure 5.8 on page 166 and the rule base in Table 5.1 on page 162. Also, suppose that we have

$$e(t) = \frac{\pi}{2}$$

and

$$\dot{e}(t) = -0.0045$$

at some time t. Assume that we use the rule base shown in Table 5.1 on page 162 and minimum to represent both the premise and implication.

(a) On Table 5.1, draw boxes around the centers of the output membership functions in the body of the table that correspond to the rules that are on.

(b) Draw all the implied fuzzy sets on the output universe of discourse.

(c) Find the output of the fuzzy controller using center-average defuzzification.

(d) Find the output of the fuzzy controller using COG defuzzification.

(e) Assume that we use the product to represent both the premise and implication. Repeat (b)–(d).

(f) Write a computer program to solve (b) and (c).

Exercise 5.6 (Graphical Depiction of Fuzzy Decision Making): Develop a graphical depiction of the operation of the fuzzy controller for the tanker ship similar to the one given in Figure 5.19 on page 184. For this, choose $e(t) = \frac{\pi}{2}$ and $\dot{e}(t) = -0.0045$, which will result in four rules being on. Be sure to show all parts of the graphical depiction, including an indication of the values for $e(t)$ and $\dot{e}(t)$, the implied fuzzy sets, and the final defuzzified value.

(a) Use minimum for the premise and implication and COG defuzzification.

(b) Use product for the premise and implication and center-average defuzzification.

Exercise 5.7 (Takagi-Sugeno Fuzzy Systems): In this problem you will study the way that a Takagi-Sugeno fuzzy system interpolates between linear mappings. In particular, as an example, suppose that $n = 1$, $R = 2$, and that we have rules

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^1 \textbf{ Then } b_1 = 2 + u_1$$

$$\textbf{If } \tilde{u}_1 \text{ is } \tilde{A}_1^2 \textbf{ Then } b_2 = 1 + u_1$$

with the universe of discourse for u_1 given in Figure 5.33 so that μ_1 represents $\tilde{A}_1^1$ and μ_2 represents $\tilde{A}_1^2$. We have

$$y = \frac{b_1\mu_1 + b_2\mu_2}{\mu_1 + \mu_2} = b_1\mu_1 + b_2\mu_2$$

We see that for $u_1 > 1$, $\mu_1 = 0$, so $y = 1 + u_1$, which is a line. If $u_1 < -1$, $\mu_2 = 0$, so $y = 2 + u_1$, which is a different line. In between $-1 \leq u_1 \leq 1$, the output y is an interpolation between the two lines.

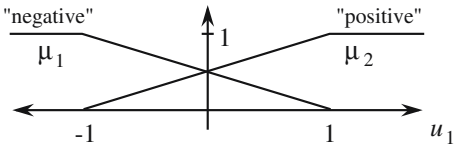

Figure 5.33: Membership functions for Takagi-Sugeno fuzzy system example.

(a) Show that the nonlinear mapping induced by this Takagi-Sugeno fuzzy system is given by

$$y = \begin{cases} 1 + u_1 & \text{if } u_1 > 1 \\ 0.5u_1 + 1.5 & \text{if } -1 \leq u_1 \leq 1 \\ 2 + u_1 & u_1 < -1 \end{cases}$$

(Hint: The Takagi-Sugeno fuzzy system represents three lines, two in the consequents of the rules and one that interpolates between these two.)

(b) Plot y versus u_1 over a sufficient range of u_1 to illustrate the nonlinear mapping implemented by the Takagi-Sugeno fuzzy system.

Exercise 5.8 (Lyapunov's Direct Method for Fuzzy Control Systems): Consider Exercise 4.3 but now suppose that you design $F(x)$ to be a fuzzy controller.

(a) Repeat parts (a)-(c) in Exercise 4.3.

(b) From the perspective of stability analysis, for this simple example, do you see any advantage of neural control over fuzzy control, or vice versa?

Design Problem 5.1 (Design of a Fuzzy Controller for Cargo Ship Steering): In this problem we study the development of fuzzy controllers for a cargo ship steering problem. Use the nonlinear model of the tanker ship provided in Equation (4.5) but with $K_0 = -3.86$, $\tau_{10} = 5.66$, $\tau_{20} = 0.38$, $\tau_{30} = 0.89$, and $l = 161$ meters [30]. Assume the rudder is saturated at ± 80 degrees as in the tanker case. Also, we will assume that the cargo ship is traveling in the x direction at a velocity of 5 meters/sec. Similar to the tanker ship, you should seek to get as good a steering response as possible.

(a) Develop a fuzzy controller for the cargo ship steering problem and simulate the closed-loop system to demonstrate its performance. Test the cases where there is a wind disturbance (assume it is modeled in the same way as for the tanker ship) and speed change.

(b) Develop a proportional-derivative (PD) controller for the cargo ship and test it under the same conditions as in (a).

(c) Compare the results in (a) and (b). Discuss.

Design Problem 5.2 (Design of a Fuzzy Controller that Balances an Inverted Pendulum): Consider the simple problem of balancing an inverted pendulum on a cart, as shown in Figure 5.34. Here, y denotes the angle that the pendulum makes with the vertical (in radians), l is the half-pendulum length (in meters), and u is the force input that moves the cart (in Newtons). We will use r to denote the desired angular position of the pendulum. The goal is to balance the pendulum in the upright position (i.e., $r = 0$) when it initially starts with some nonzero angle off the vertical (i.e., $y \neq 0$). This is a very simple and academic nonlinear control problem, and many good techniques already exist for its solution. Indeed, for this standard configuration, a simple PID controller works quite well, even in implementation. Here, you will develop a fuzzy controller for the inverted pendulum simply to gain practice in fuzzy control design.

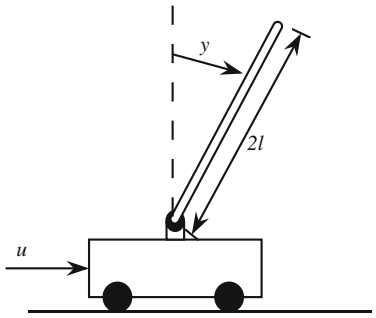

Figure 5.34: Inverted pendulum on a cart.

One model for the inverted pendulum shown in Figure 5.34 is given by

$$\begin{aligned} \ddot{y} &= \frac{9.8\sin(y) + \cos(y)\left[\frac{-\bar{u}-0.25\dot{y}^2\sin(y)}{1.5}\right]}{0.5\left[\frac{4}{3} - \frac{1}{3}\cos^2(y)\right]} \\ \dot{\bar{u}} &= -100\bar{u} + 100u. \end{aligned} \tag{5.9}$$

The first order filter on u to produce $\bar{u}$ represents an actuator. In the simulations of the fuzzy control system for balancing the inverted pendulum, be sure to use an appropriate numerical simulation technique for the nonlinear system and a small enough integration step size (e.g., a fourth-order Runge-Kutta method with an integration step size of $h = 0.001$). In your simulations, let the initial condition be $y(0) = 0.1$ radians ($= 5.73$ deg.), $\dot{y}(0) = 0$, and $\ddot{y}(0) = 0$ (this translates into an initial condition on the actuator state).

(a) Develop a fuzzy controller that uses $e = r - y$ and $\dot{e}$ as inputs, the minimum operator to represent both the "and" in the premise and the implication, and COG defuzzification. Simulate the closed-loop system and plot the output y and input u to demonstrate that your fuzzy controller can balance the pendulum. You should add scaling gains and tune the fuzzy controller as we did for the tanker ship steering problem.

(b) Repeat (a) for the case where you use product to represent the premise and implication and center-average defuzzification.

(c) Study the performance of the controllers in (a) and (b) for different initial conditions.

Design Problem 5.3 (Design and Stability Analysis of Expert Control Systems)*: This problem is based on a chapter in [410] that you should first obtain and read carefully before answering the following questions. You may also want to consult [338] for a related study.

(a) First, for the model of the tank provide a state transition diagram (circles for states, directed arrows between circles to represent plant changes for certain inputs) that represents the dynamics of the plant. Next, specify the state-transition diagram for the closed-loop system when the "seven-rule controller" is used. Also, draw the diagram for the case where the "three-rule" controller is used.

(b) Simulate the closed-loop system. In simulation, demonstrate that for each initial condition in the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (initial liquid height) that the liquid level will converge to an appropriate set of values.

(c) Explain what "reachability" is, provide a mathematical definition for it, and analyze a reachability property of the tank system via simulation.

(d) Repeat the stability analysis shown in the chapter, providing full explanations at every step of the derivation, to illustrate mathematically that the closed-loop system processes the indicated stability properties (do this for both the seven-rule and three-rule controllers).