

This is important because the centroid of the solution space must be calculated to produce the crisp output value (in this case ΔSVS). It is much faster and easier to calculate the centroid of a group of isosceles triangles, than the centroid of a group of trapezoids (This embedded system does not have a math coprocessor).

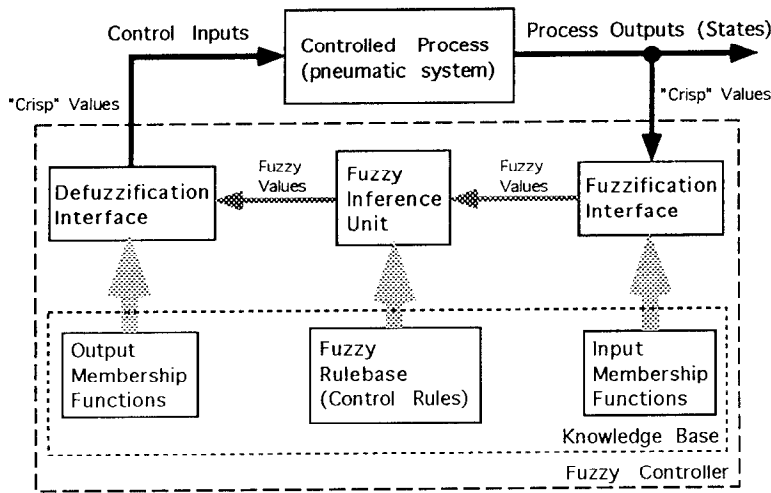


Figure 8. Anatomy of a Fuzzy Controller

The control rules, and input and output membership functions are developed based upon expert knowledge, simulation and testing, and will be discussed in the following paragraphs.

Fuzzy Controller Knowledge Base

The heart of this project was developing the knowledge base used in real-time to determine the amount of control to apply to the Ps servo valve. The knowledge base for a fuzzy controller consists of the rulebase and membership functions (refer back to figure 8). The fuzzy rulebase contains 11 control rules of the form

$$R_{i,j,k} : \text{If error} = A_i \text{ and error rate} = B_j, \\ \text{then control action} = C_k.$$

This form of fuzzy rule is relatively commonly used for control [7]. The design strategy was to choose a default rulebase and "best guess" membership function (mbf) sets, then to test and tune the mbf sets until desired control performance was achieved. The fuzzy rulebase, shown graphically in figure 9, was chosen by intuitively estimating the desired amount of control for combinations of input error and error rates.

Each mbf set contains five mbf's, labeled negative big (NB), negative small (NS), zero (Z), positive small (PS) and positive big (PB). These were chosen since they seemed to

linguistically represent the expected inputs and output well. The initial mbf sets were derived from estimating the maximum likely values of the error and error rate, and the maximum possible range of the servo valve. At this point, no physical response characteristics of the servo were taken into account.

		Change in Error				
		NB	NS	Z	PS	PB
Error	PB			NB		
	PS		Z	NS		
	Z	PB	PS	Z	NS	NB
	NS			PS	Z	
	NB			PB		

Figure 9. Fuzzy Rulebase

Fuzzy Controller Implementation

In order to understand the design and development of this fuzzy controller, it is helpful to be familiar with the environment in which it resides. All of the control of the pitot-static test set is accomplished via software running on a MC68000 microprocessor. In addition to pneumatic control during an actual aircraft test run, the software program, written in C, also performs other tasks such as internal self-testing, venting, pressure rate error calculations and the setting of solenoids for different modes of operation. (The fuzzy controller replaces the PID controller implemented in several software modules).

The fuzzy control software consists of three parts: the control module, the "glue" module, and the membership function tables. The control software is where the control rules are implemented and the inference takes place. The glue module contains code to interface the control module to the rest of the system software, as it performs data format conversions, calls the control module (receiving back ΔSVS) and sets the new servo position. Later in the project, a simple 3-point finite impulse response (FIR) filter was also added to this module in an attempt to reduce jitter. The membership function tables are data tables stored in non-volatile memory and used by the control module for fuzzification and defuzzification. There are four of these tables, one each for the error and error rate inputs, and two for the servo delta (ΔSVS) output (one used for pressure application, the other for vacuum).

Fuzzy Controller Testing and Tuning

Due to the lack of a mathematical model, the primary means of arriving at the desired performance for a fuzzy controller is through testing with the actual hardware, then tuning the input and output membership functions and rulebase as necessary based upon the test results. The testing and tuning procedures for this project are described below.