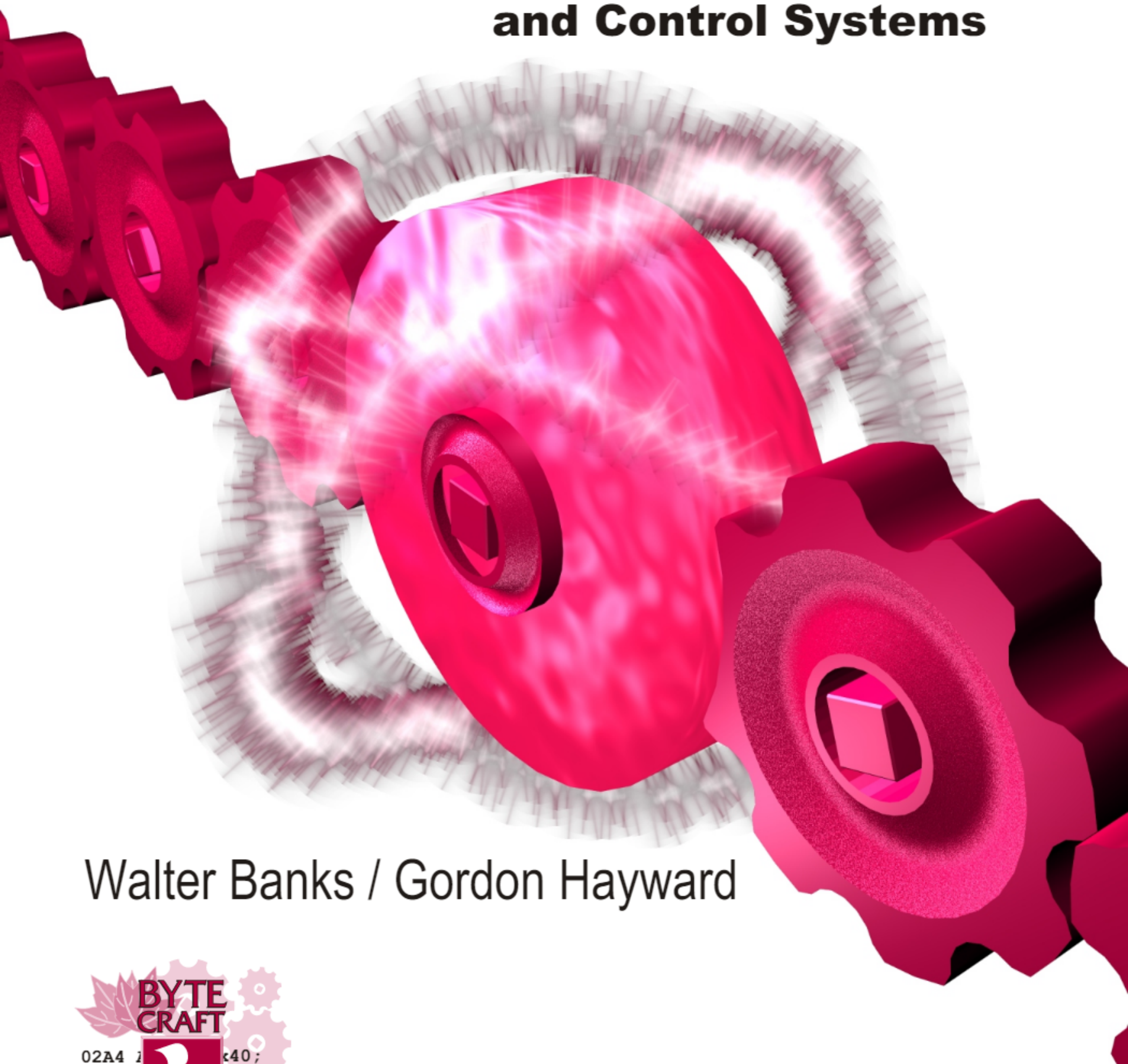
Fuzzy Logic
in Embedded Microcomputers and Control Systems



Walter Banks / Gordon Hayward

BYTE CRAFT LIMITED

Fuzz-C™

Fuzzy Logic Preprocessor for C

```
02A4 
02A6 
02A9 
```

Fuzz-C™ is a stand-alone preprocessor that seamlessly integrates fuzzy logic into the C language. Now you can add fuzzy logic to your applications without expensive, specialized hardware or software. Fuzz-C accepts fuzzy logic rules, membership functions and consequence functions, and produces C source code that can be compiled by most C compilers, including the Byte Craft Limited Code Development System.

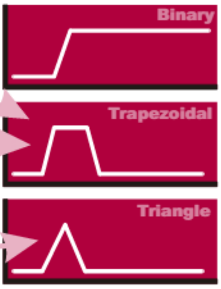
The preprocessor generates C code that is both compact and significantly faster than most current fuzzy logic commercial implementations—all with your favorite C compiler.

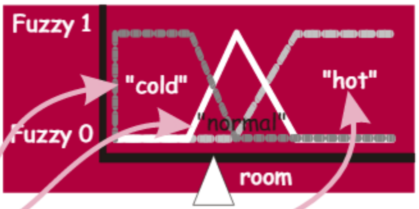
Fuzz-C provides a practical, unified solution for applications that require fuzzy logic control systems. Use your existing C libraries for program management, keyboard handlers and display functions without change; you can implement system control functions using fuzzy rules.

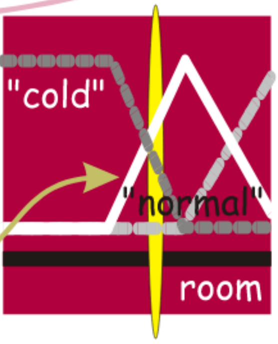
Fuzz-C is a flexible system that allows all data types supported by your C compiler. Standard defuzzification methods, such as *center of gravity*, *max left*, *max right*, and *max average*, are provided in source form. Fuzz-C lets you easily add new defuzzification methods.

```
/* Fuzzy Logic Climate Controller

This single page of code creates a fully
Functional controller for a simple air
conditioning system */

#define thermostat PORTA
#define airCon PORTB.7

/* degrees celsius */
LINGUISTIC room TYPE int MIN 0 MAX 50
{
  MEMBER cold   { 0, 0, 15, 20 }
  MEMBER normal { 20, 23, 25 }
  MEMBER hot    { 25, 30, 50, 50 }
}

/* A.C on or off */
CONSEQUENCE ac TYPE int DEFUZZ CG
{
  MEMBER ON  { 1 }
  MEMBER OFF { 0 }
}

/* Rules to follow */
FUZZY climateControl
{
  IF room IS cold THEN
    ac IS OFF
  IF room IS normal THEN
    ac IS OFF
  IF room IS hot THEN
    ac IS ON
}

int main(void)
{
  while(1)
    {
      /* find the temperature */
      room = thermostat;
      /* apply the rules */
      climateControl();
      /* switch the A.C. */
      airCon = ac;
      wait(10);
    }
}
```

Membership Functions



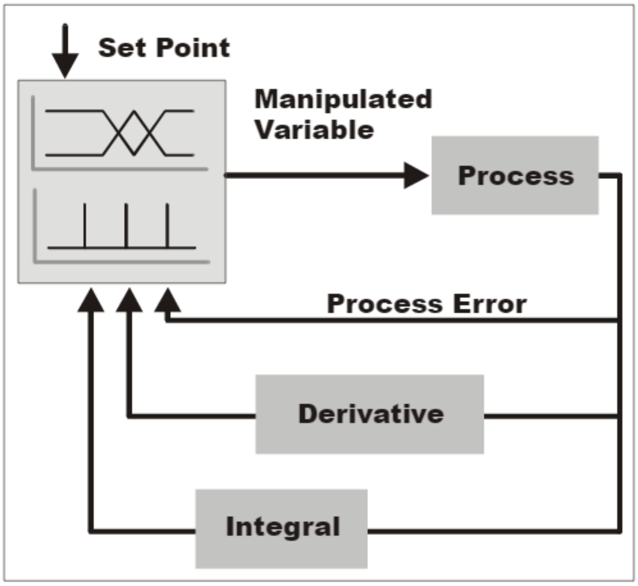




Center of Gravity Calculation

Terms: prepaid American Express, VISA or cheque. Overseas orders prepaid in U.S. funds drawn on a Canadian or U.S. bank only. Please obtain appropriate import documentation. Canadian customers are subject to applicable taxes. Specifications and price information subject to change without notice. Fuzz-C is a registered trademark of Byte Craft Limited. Other marks are trademarks or registered trademarks of their respective holders.

Fuzz-C™ includes one year technical support via phone or email. Fuzz-C requires modest system resources: DOS or Windows and less than 1 megabyte of memory. Fuzz-C works with *make* and other industry-standard build systems. Complete documentation is included.



Fuzzy Logic in Embedded Microcomputers and Control Systems

Walter Banks / Gordon Hayward

Published by

BYTE CRAFT LIMITED

421 King Street North

Waterloo, Ontario

Canada • N2J 4E4

Sales Information and Customer Support:

BYTE CRAFT LIMITED
421 King Street North
Waterloo, Ontario
Canada N2J 4E4

Phone	(519) 888-6911
FAX	(519) 746-6751
Web	www.bytecraft.com

Printed in Canada October, 2002

First Web Release October 2002

Forward

This booklet started as a result of the rush of people who asked for copies of the overhead slides I used in a talk on **Fuzzy Logic For Control Systems** at the 1993 Embedded Systems Show in Santa Clara.

A fuzzy logic tutorial

There is a clear lack of basic tutorial materials for fuzzy logic. I decided that I did have enough material to create a reasonable tutorial for those beginning to explore the possibilities of fuzzy logic. In addition to the material presented at the embedded systems conference I have added additional chapters.

Clear thinking on fuzzy linguistics

The first chapter essentially consists of the editorial I wrote for *Electronic Engineering Times* (printed on October 4, 1993). The editorial presented a case for the addition of linguistic variables to the programmer's toolbox.

Fuzzy logic implementation on embedded microcomputers

The second chapter is based upon a paper I presented at **Fuzzy Logic '93** by Computer Design in Burlingame, CA (in July of 1993). This paper described the implementation considerations of fuzzy logic on conventional, small, embedded micro-computers. Many of the paper's design considerations were essential to the development of our ***Fuzz-C***® preprocessor. I have created most of the included examples in ***Fuzz-C*** and although you don't need to use ***Fuzz-C*** to implement a fuzzy logic system, you will find it useful to understand some of its design.

Software Reliability and Fuzzy Logic

Originally part of the implementation paper, this chapter presents what is actually a separate subject. The inherent reliability and self scaling aspects of fuzzy logic are becoming important and may in fact be the over riding reason for the use of fuzzy logic.

Appendix

The appendix contains, in addition to copies of the slides, the actual code for a fuzzy PID controller as well as the block diagram of the PID controller used in my Santa Clara talk entitled **Fuzzy Logic For Control Systems**.

Adjusting to fuzzy design

While presenting the paper in Santa Clara, much of the discussion touched on provable control stability. This final issue has discouraged many engineers from employing fuzzy logic in their designs. Despite the great incentive to use fuzzy logic, I found it took me about a year and a half to feel comfortable with the addition of linguistic variables to my software designs.

Fuzzy logic is not magic, but it has made many problems much easier to visualize and implement. Debugging has generally been straight forward in my own code, and I think that most who have implemented fuzzy logic applications share this opinion.

I have tried to make the material presented both in this booklet, and in my presentations in public, as non-commercial as possible. The purpose here is to inform and educate. Some of the slide material came from Dr. Gordon Hayward of the University of Guelph. Gord is a friend and colleague dating back more than twenty years. Gord was the first to look at fuzzy logic through transfer functions. The slides of the actual control system response were generated by a student of Dr. Hayward's in a report (L. Seed 05-428 Project, Winter 1993). I thank both of them for this material.

Much material has been published on fuzzy logic and linguistic variables. Most of the literature available in the English-speaking world was written primarily by and for mathematicians, with few papers and articles written for computer scientists or system implementors. This work started with a paper by Lotfi Zadeh more than a quarter century ago ("Fuzzy Sets", Information and Control 8, pp. 338-353, 1965). Professor Zadeh has remained a tireless promoter of the technology.

At the 1992 Embedded Systems Conference in Santa Clara, the genie was finally let out of the bottle, and fuzzy logic came into its own with wide interest. Jim Sibigtroth's article in *Embedded Systems Programming* magazine in December, 1991 cracked the bottle, describing for the first time a widely available, understandable implementation of a fuzzy logic control system workable for general purpose microprocessors. Jim Sibigtroth has been working on the promotion of fuzzy logic control systems to the point of personal passion. As developers began to understand the real power of using linguistic variables in control applications, the negative implications of the name *fuzzy logic* have given way to a deep understanding that this is a powerful tool backed by solid mathematical principles.

I thank all those who work with me at Byte Craft Limited for their efforts. A special thanks to Viktor Haag who gets to do much of the hard work for our printed material and far too little credit. For me I accept responsibility for all of the errors and inconsistencies.

Walter Banks
October 28, 1993.

Clear thinking on fuzzy linguistics

I have had a front row seat, watching a computing public finding uses for an almost 30 year-old *new* technology.

Personally, I struggled with finding an application to clearly define what all the magic was about, until I switched the question around and looked at how an ever increasing list of fuzzy logic success stories might be implemented. I then looked at the language theory to see why *linguistic variables* were important in describing and solving problems on computers.

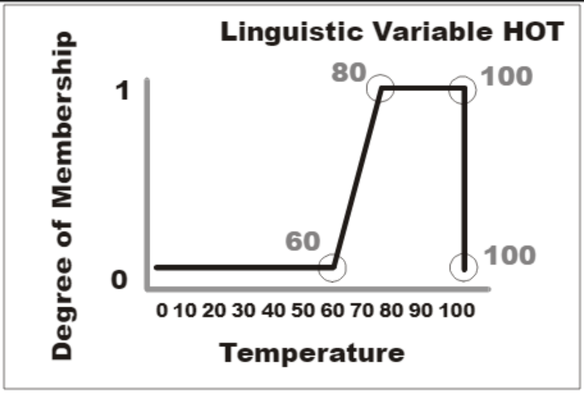
Linguistic variables are central to fuzzy logic manipulations. Linguistic variables hold values that are uniformly distributed between 0 and 1, depending on the relevance of a context-dependent linguistic term. For example; we can say *the room is hot* and *the furnace is hot*, and the linguistic variable *hot* has different meanings depending on whether we refer to the room or the inside of the furnace.

The assigned value of 0 to a linguistic variable means that the linguistic term is not true and the assigned value of 1 indicates the term is true. The "linguistic variables" used in everyday speech convey relative information about our environment or an object under observation and can convey a surprising amount of information.

The relationship between crisp numbers and linguistic variables is now generally well understood. The linguistic variable **HOT** in the following graph has a value between 0 and 1 over the crisp range 60-80 (where 0 is not hot at all and 1 is undeniably hot). For each crisp number in a variable space (say room), a number of linguistic terms may apply. Linguistic variables in a computer require a formal way of describing a linguistic variable in the crisp terms the computer can deal with.

The following graph shows the relationship between measured room temperature and the linguistic term *hot*. In the space between *hot* and *not hot*, the temperature is, to some degree, a bit of both.

The horizontal axis in the following graph shows the measured or crisp value of temperature. The vertical axis describes the degree to which a linguistic variable fits with the crisp measured data.



Most fuzzy logic support software has a form resembling the following declaration of a linguistic variable. In this case, a crisp variable *room* is associated with a linguistic variable *hot*, defined using four break points from the graph.

```
LINGUISTIC room TYPE unsigned int MIN 0 MAX 100
 {
     MEMBER HOT { 60, 80, 100, 100 }
 }
```

We often use linguistic references enhanced with crisp definitions.

Cooking instructions are linguistic in nature: "Empty contents into a saucepan; add 4½ cups (1 L) cold water." This quote from the instructions on a Minestrone soup mix packet shows just how common linguistic references are in our descriptive language. These instructions are in both the crisp and fuzzy domains.

The linguistic variable "saucepan", for example, is qualified by the quantity of liquid that is expected. One litre (1 L) is not exactly 4½ cups but the measurement is accurate enough (within 6.5%) for the job at hand. "Cold water " is a linguistic variable that describes water whose temperature is between the freezing point (where we all agree it is cold) to some higher temperature (where it is cold to some degree).

The power of any computer language comes from being able to describe a problem in terms that are relevant to the problem. Linguistic variables are relevant for many applications involving human interface. Fuzzy logic success stories involve implementations of tasks commonly done by humans but not easily described in crisp terms.