A Modular Fuzzy Logic Expert System for Autonomous Mobile Robots

G. N. Reddy, Vishnudev Vasanthan, Gurpreet Singh, and Sreelatha Maila
Drayer Department of Electrical Engineering, Lamar University
Beaumont, TX, USA, gnreddy@lamar.edu

International Workshop on Artificial Intelligence IWAII-2014, Istanbul, Turkey; August 22-23, 2014: Update 2.1, 7/520/14

Abstract -- This paper presents modular fuzzy logic expert system software, written in C++, for building real-time autonomous mobile robots. The source can be used with any microcontroller that supports C++ such as Arduino-microcontrollers. The software is a fuzzy logic expert system FLES simulator. It can be used to simulate an FLES off-line; or can be downloaded into a microcontroller to operate in real-time. The software is modularized so that it can be used with any controller or estimation type of applications. The embeddable-software has four, three-input and one-output, modules. The three input modules represent FLES subsections in the form its: rule-based knowledge base; input/output variables with their corresponding membership functions; and the sensed input variable values. The output module containing the generated output signals. It also generates text-based output simulation trace illustrating the detailed sequential steps executed by the inference engine. While operating off-line, the three input modules are represented by three input data files and the output module by a generated output file. While operating in real-time, the knowledge-base and the i/o membership functions are merged into the FLES C++ code; the input signals and the outputs signals are mapped to the microcontroller's i/o-ports. In real-time mode the output simulation trace can be disabled. We have embedded this software into Arduino-Uno-Due microcontrollers to built two mobile-RT-FLES: One an FLES for precise estimation of battery state of charge SoC; and the other an FLES to implement the classical controller for balancing an inverted pendulum.

Key Words: Fuzzy Logic Expert Systems; Fuzzy Logic Controllers; Embeddable Software; Micro-controllers; Autonomous mobile robots.

I. INTRODUCTION

Fuzzy logic expert systems have been used for variety applications [1, 2]. This paper is on using FLES to solve control [3, 4] and estimation problems [5]. Fuzzy logic controllers FLCs are attractive because they are robust, multiple in and multiple out, and simpler to implement. FLCs are suitable to run on PCs main-frame computers and on microcontrollers. To make them mobile one should run them on microcontrollers. If microcontrollers are used, the commercial FLES-software packages are difficult use. One needs to write their own FLES source code. The software presented is in this paper is one such code to run on the microcontrollers. It is modularly-structured so it can be used with any application. The following sections describe this microcontroller-embeddable FLES software.

II. EMBEDDABLE FLES SOFTWARE FOR AUTONOMOUS MOBILE ROBOTS

Fig. 1 shows the five basic elements of the FLES: Knowledge base KB, Inference engine IE, User interface UI, Input-fuzzifier, and output-defuzzifier. The Knowledge base KB is a set of empirical rules by which the overall system behavior is summarized. In an empirical rule the input/output values are in linguistic-form such as positive-small ps, negative-large nl, and positive-medium pm. Inference engine IE is the kernel of the FLC that executes all of the sequential steps involved in the overall execution of the FLC, starting from reading the inputs till the control outputs are generated. These steps are described in the following section. Input-fuzzifier converts absolute values into linguistic values. Here the absolute input variable values are expressed as a function of the membership functions of the input variables IMFs. Output-defuzzifier converts fuzzy or linguistic values into absolute values. The computing system is the one on which the FLC-kernel is executed. For off-line applications it is usually a personal computer. For on-line autonomous mobile applications, it should be a microcontroller.

Figure 1. Basic elements of an FLES for control applications.
III. THE FLES SOFTWARE ARCHITECTURE

Fig. 2 shows the overall architecture of the embeddable software. It has four, three-input and one-output, modules. The first input module reads sensor values in absolute form. The second input module has membership functions of the input and the output variables. The third input module has the knowledge base of the system in the form of a rule-set. The output-module generates defuzzified output signals.

![Diagram of FLES software architecture](image)

While running off-line (in simulation-mode) on a personal computer, the inputs are read from three input files and the output is stored into an output file. While running on a real-time autonomous robot’s micro-controller, the input knowledge base and the membership functions are merged into the FLES-source code. Sensory inputs are mapped to the analog-input-ports of the controller; and the outputs are mapped to a controller’s analog output-ports. Overall execution sequence of the inference engine, to implement FLC, is detailed in the following section.

IV. FLC-KERNEL: EXECUTION SEQUENCE OF THE INFERENCE ENGINE

Step-by-step sequences of tasks executed by the inference engine are as follows:

**IE1: Fuzzify input variables**
Express input variables as a percentage of the input membership functions IMFs. Translate inputs into linguistic form.

**IE2: Find the activated rule-set Ra**
Find the rule set Ra in the KB, where their input-variable-value requirements match with the current input-values in their linguistic form.

**IE3: Find effective input membership function eimf-Ri for each of the activated rule-Rai.**
Example: assuming there are two input variables in1 and in2 and the rules are logically ANDed such as:

\[
\text{if (in1 is ns) AND (in2 is pm) then y is nm.}
\]

\[
eimf-Rai = \min(in1v-Rai, in2v-Rai),
\]

\[
eimf-Raj = \min(in1v-Raj, in2v-Raj),
\]

where eimf-Rai is the effective input membership function of the active rule Rai. in1v-Rai and in2v-Rai are the fuzzified values of in1 and in2 in Rai. Similarly, eimf-Raj is the effective input membership function of the rule Raj; in1v-Raj and in2v-Rj are fuzzified values of in1 and in2 in Raj. The fuzzified variable values in1v and in2v are being expressed as a percentage of the input membership functions. To find eimf, select minimum of in1v and in2v.

**IE4: Find the final rule to execute with its eimf and eomf**
Among the active-rules identify the rule to execute:

Rule to fire Rf = R with: max(eimf-Rai, eimf-Raj, ...)

a. Select the rule Rf corresponding to the maximum of the eimfs

b. eimf-Rf = min(in1v-Rf, in2v-Rf)

c. eomf-Rf = omf-Rf (with one output variable)

With multiple output variables eomf should be computed similar to the eimf.

**IE5: Find defuzzified control output y**

\[
\text{output} = \text{eimf} \times \text{eomf}
\]

\[
y = \text{eimf-Rf} \times \text{eomf-Rf}
\]

The embeddable software presented in this paper executes these five steps in sequence. It also generates an output simulation trace corresponding to each of these steps. This output trace is illustrated in the following section for solving a specific control problem.

V. TEST RESULTS: THE FLES OUTPUT SIMULATION TRACE

Test results of the software are presented in the form of solving a specific control problem. The control problem of this case study is to balance the inverted pendulum IP using a cart driven by dc-motors as shown in Fig. 3. The function of the controller is to keep the IP straight-up. When the pendulum tilts away from the center by \( \theta \) (angle a), degrees, at a rate of \( \frac{d\theta}{dt} \) (derivative of the angle da); then the controller must generate a control signal to move cart by x-units in the right direction at right rate of dx/dt. The movement of the cart is proportional to dc-motor current mc. If the motor is controlled at fixed interval then dx/dt is not needed. The problem now has two input variables a and da and one output variable mc. The FLES configuration of this problem is shown in Fig. 4. The input membership functions IMFs for the input variables a and da are shown in Fig. 5. The triangular MFs are represented by three vertices in the output simulation trace as shown in Table 4. For example, imf-nm has vertices p0, p1, p2: -54, -36, -18. The fuzzified or linguistic value of the variable is “nm”. Input variables a and da have min/max values of: -54/+54; and units of degrees and degrees/sec respectively. The range is from -54 to +54 with 7 membership functions each. Fig. 6 shows the output membership function OMF for the output variable mc. It also has 7 membership functions but they are singleton. The knowledge base used for the IP-problem shown is shown in Fig. 7. The final goal now is for a given sensed values of a and da the FLC must generate the appropriate motor current mc.
In the software: the sensed values of a and da are specified through an input file “infile1-SenData.txt” as shown in Table 1. The input/output membership functions are specified through the input file “infile2-IOMFs.txt”, refer to Table 2. The knowledge base is specified through another input file “infile3-KB.txt”, this is shown in Table 3. The generated output is stored into an output file “fles-outfile.txt”, this file is shown in Table 4.

While operating in real-time the knowledge base and the MFs are merged into the FLES-source code. Sensed input values and the generated outputs are mapped to i/o-ports of the microcontroller being used. Depending on the problem the input, the output, and the knowledge base modules can be specified accordingly. It includes the name and units of input and output variables; the IMFs and the number of IMFs, OMFs and the number OMFs, and finally the KB.

The detailed results of execution tasks IE1 through IE5 of the inference engine, specified in section IV, are shown in Table 4. This can be disabled in on-line mode. In this table, the initial sections A, B, and C will display input data to the FLC. This includes the sensor data of the input variables; the input/output membership functions; and the knowledge base of the problem. The task here is to build an FLC to balance an inverted pendulum problem. It is a two-input and one-output problem. The input variable values for the FLC are: the pendulum’s angular displacement (angle a) is -12.0 degrees away from the center, and rate of angular displacement (derivative of angle da) is -3.0 degrees/sec. Input variables in1 and in2 both have 7 membership functions. Minimum and maximum values are -54 to +54 with 18-units between the vertices with units of degrees for in1 and degrees/sec for in2. The output variable mc has 7-MFs with minimum and maximum of -18 ma and +18 ma with 6 ma separation. The mc linguistic values ranging from negative-large (nl) to positive-large (pl). The knowledge base of the IP-problem is specified by 13-rules denoted as R0-to-R12. These are logically ANDed-rules.

The 5-step procedure by which the inference engine generates the control signal from the sensor input data is shown in section-D of Table 4. The five steps are denoted as: IE1 through IE5.

The IE1-Trace: In IE1 the input variables are fuzzified: for in1 = -12 degrees, the fuzzified values are: (12/18)-ns or (6/18)-zr; and for in2 = -3 degrees/sec, the fuzzified values are: (3/18)-ns or (15/18)-zr.

The IE2-trace: In IE2 active rule set Ra is determined: the activated rules are R7 and R8. Ra = {R7, R8}.

The IE3-trace: In IE3 effective input membership functions eIMF-Rai for each active rule Rai is determined:
- eIMF-R7 = (12/18)-ns;
- eIMF-R8 = (6/18)-zr;

The IE4-trace: In IE4 Rule to be fired is determined:
- Rule fired = max(eIMFs) = eIMF-R7 = (12/18);
- Rule fired is R7

The IE5-trace: In IE5 find output – motor current mc:
- output = eIMF * eOMF = (12/18) * ps
= (12/18) * 6.0 = 4.0 milli-amps

VI. CONCLUSIONS

This paper presents embeddable FLC-software-code that can be downloaded into microcontrollers to build autonomous mobile-robots. We have used this software to implement two fuzzy logic systems, one is an estimation problems and the other a control problem. The estimation problem is an application one that is used for precise estimation of the battery’s state of charge SoC [5]. For this we have used Handyboard that supports interactive C. This required partial modification C++ into interactive-C. Recently we have implemented the same application on Arduino-Due microcontroller. The other is a control application that we have implemented is to balance an inverted pendulum. This is again implemented on a Arduino-Due microcontroller. This is an autonomous robot. The current microcontrollers are very powerful yet very inexpensive (under $50). With your own embeddable source-code, such as the one presented in this paper, one can build extremely complex yet inexpensive mobile-FLC.

REFERENCES

Scientific Cooperations International Workshops on Electrical and Computer Engineering Subfields
22-23 August 2014, Koc University, ISTANBUL/TURKEY

Fuzzy-Logic Expert Systems FLES: Inference Engine IE: Case Study
Overall operations by the Inference Engine IE in the RT-FLES

Figure 3. Balancing an Inverted Pendulum.
Figure 4a. Fuzzy Logic Expert System FLES with i/o.

Figure 5. Fuzzy Logic Input Membership Functions IMF's for the input variables: a and da.
Input variable 1: a: Units: degrees; Min / Max: -54 / +54
Input variable 2: da: Units: degrees/sec; Min / Max: -54 / +54

Output variable: mc
Units: milli_amps
Min/Max: -18/18
Member Ship Functions:
0 nl -18
1 nm -12
2 ns -6
3 zero 0
4 ps 6 a \rightarrow angle, 0, theta, angular displacement;
5 pm 12 da \rightarrow derivative of the angle, \frac{d\theta}{dt}, \theta'
6 pl 18

Figure 6. Output Membership Functions OMF's for the output variable mc.

FLES Knowledge-Base: Rule-Set
R0: IF a IS zr AND da IS nm THEN mc IS pm
R1: IF a IS ps AND da IS ns THEN mc IS ps
R2: IF a IS zr AND da IS ps THEN mc IS ns
R3: IF a IS zr AND da IS pm THEN mc IS nm
R4: IF a IS zr AND da IS pl THEN mc IS nl
R5: IF a IS nl AND da IS zr THEN mc IS pl
R6: IF a IS nm AND da IS zr THEN mc IS nl
R7: IF a IS ns AND da IS zr THEN mc IS ps
R8: IF a IS zr AND da IS nm THEN mc IS zr
R9: IF a IS ps AND da IS nm THEN mc IS ns
R10: IF a IS pm AND da IS nm THEN mc IS nm
R11: IF a IS pl AND da IS nm THEN mc IS nl
R12: IF a IS ns AND da IS ps THEN mc IS ps

Figure 7. The FLES Knowledge Base KB for the IP-Problem

Note: da = \frac{d\theta}{dt} is computed: "0" is sampled into the FLES at a constant interval of around 1000-samples/sec (or at a sampling f frequency of 1 kHz) from a photo-sensor. That is, dt = 1/1000 = 1 msec.

Table 1. Sensor Input Data.
// infile1-SenData.txt
// Input values
-12.0 : degrees
-3.0 : degrees/sec

Table 2. Input/Output Membership Functions IOMF's.
// infile2-IO MFs.txt
// Input Membership Functions IMFs: min, max, n
-54.0 : inlmin a: degrees
+54.0 : inlmax a: degrees
7 : inlnum
-54.0 : in2min da: degrees/sec
+54.0 : in2max da: degrees/sec
7 : in2num
-18.0 : in3min mc: milli_amps
+18.0 : in3max mc: milli_amps
7 : in3num

Table 3. The Knowledge Base for the IP-Problem.
// infile3-KB.txt
// Knowledge base for the fuzzy logic expert system
// Example: infile3KB.txt for: 13 rules, 2-inputs, 1-output
// MFS: 0-nl, 1-nm, 2-nz, 3-zr, 4-ps, 5-pm, 6-pl
// Rule: 0 1 2 3 4 5 6 7 8 9 10 11 12
// ----------------------------------------
// zr ps zr zr zr zr zr zr zr zr ps ; kbIn1 MFs
// ps ps ps ps pl ps zr zr zr zr zr zr zr zr ps ; kbIn2 MFs
// ps ps ps ps nl pl ps ps ps ps ps ps ps ; kbOut MFs
Table 4. FLES Output Simulation Trace: FLC Kernel

<table>
<thead>
<tr>
<th>Execution Steps</th>
<th>IE1 through IE5 of the Inference Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUZSY LOGIC EXPERT SYSTEM FLES OUTPUT SIMULATION TRACE:</td>
<td></td>
</tr>
<tr>
<td>Dr. G. Reedy, LUCX, Summer 11, 2013</td>
<td></td>
</tr>
<tr>
<td>Update 2.3: 8/15/2013: 10 am</td>
<td></td>
</tr>
</tbody>
</table>

Input/Output Files:
- infile1-SenData.txt: Sensor inputs: angle & dangle
- infile2-OMPf.txt: Input/Output MFs: Ranges & NumMFs
- infile3-KB.txt: Knowledge Base: Rule Set
- files-outfile.txt: FLES Output Simulation Trace

A. Reading input sensor values from infile1-SenData.txt
   INPUT DATA: infile1-SenData.txt:
   angle = -12.0
   dangle = -3.0

B. Reading input/output MFs from infile2-OMPf.txt
   Rule: Input/Output Variables:
   names, units, min, max, numMFs, deltamf
   1. angle = -54.0 54.0 7.0 18.0
   2. dangle = -54.0 54.0 7.0 18.0

C. Reading the knowledge base from infile3-KB.txt
   Rule: The Knowledge Base is:
   Rule fired:
   Final output:
   End of the output simulation trace: LUXX-FLES

Table 4. FLES Output Simulation Trace: Contd...

<table>
<thead>
<tr>
<th>Rule fired:</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final output:</td>
<td>mc = eIMF * eOMFv</td>
</tr>
<tr>
<td>OMF number, OMF value:</td>
<td>4, 6.00</td>
</tr>
</tbody>
</table>

IB1: Compute membership functions
   Rule fired: |
   Rule fired: |

IE1: Compute membership functions
   Rule fired: |
   Rule fired: |

IE2: Find the activated rule-set
   The activated rule-set is:
   7, 8
   Number of activated rules = 2

IE3: Find eIMF for each the activated rule
   Generate 10-value vector for each rule:
   Active rules & the corresponding 10-value vectors:
   Rule: ilm0: n, str, v; ilm1: n, str, v; eIMF: n, str, v
   7 2 ns 12.00 3 sz 15.00 3 ns 12.00
   8 3 sz 6.00 3 sz 15.00 3 sz 6.00

IE4: Find the rule to fire:
   Rule fired: 7
   eIMF = max(eIMFs): 12.00 / 18.00

IRE: Find Output mc:
   mc = eIMF * eOMFv
   final mc, in ma = 4.00

Note: Membership functions:
- 0-nl: 1-ns: 2-ns: 3-sz: 4-ps: 5-mp: 6-pl