An Improved Algorithm for Extraction of Fuzzy Logic Rules from Measurement Data

G. N. Reddy, Mohammad Mahmudul Islam, and Samin Sobhani
Drayer Department of Electrical Engineering, Lamar University
Beaumont, TX, USA, gnreddy@lamar.edu

Abstract – This paper presents an improved algorithm to extract the fuzzy logic rules from measurement data, to be used in turn in a Fuzzy Logic Expert System FLES. Fuzzy logic rule extraction from measurement data is a complex task, this algorithm simplifies this task to a considerable degree. In a conventional algorithm, each input variable has its own set of input membership functions. In the algorithm we have developed, all input variables are normalized to the same min-max-ranges and have the same input membership functions IMFs. This means you just have only one set of membership-functions for all of the input variables, simplifying IMF definitions. Further simplification in this single IMF-definition is achieved by choosing suitable min-max values so that MF-vertex values are round numbers for given number of IMFs. We have applied the same normalization technique to simplify the membership function definitions for the output variables. To illustrate the functionality and accuracy of the algorithm three case studies are used: one, measurement of battery state of charge SOC using FLES-based impedance-interrogation method; two, classical balance of inverted pendulum IP problem; and third, KB generated by some other study for the same IP-problem is compared with that generated by our algorithm. For implementing the three case studies, we developed three C++ programs for rule-extraction; and three other C++ programs for corresponding FLES-predictors. The FLES-predictor estimates outputs for a given set of inputs. If the extracted rule-set is correct, for a given measured input the estimated value must match with the corresponding measured value. The number of measurement pairs used in the case studies one, two, and three were: 100, 70 and 70. In case studies one and two the rms-error between the measured outputs and fuzzy-predicted outputs was within 3.3. In case study three, for a given input while KBs were slightly different, the rms-error between the output values predicted by our FLES-predictor the Motorola generated KB were near the same with less than 2.1 percent.

Key Words: Fuzzy Logic Expert Systems FLES; Knowledge Base KB; Fuzzy rule-set; Measurement/Numeric data; inverted pendulum IP; battery SOC; FLES-Predictor.

I. INTRODUCTION

The most difficult task to develop a fuzzy logic expert system, for any given application, is in extracting the fuzzy logic rules from its measured data [1, 2, 3, 4]. Overall rule generation consists of two tools: a rule generation tool and a rule test tool. In this paper, section 2 describes rule generation tool vs the rule test tool; section 3 describes fuzzy rule-extraction/generation-tool; section 4 describes FLES-predictor-tool; section 5 includes test results for the rule extraction process in case study 1: state of charge SOC measurement using FLES; and section 6 includes conclusions.

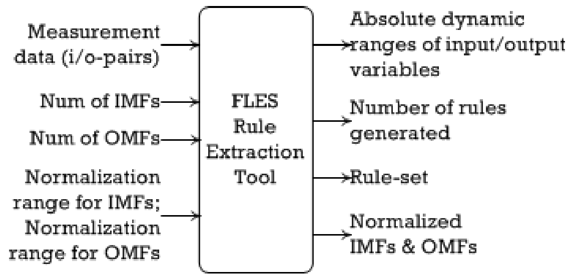

Fig. 1. FLES Fuzzy rule extraction tool

II. FLES RULE GENERATION TOOL Vs FLES-PREDICTOR-TOOL

The rule extraction algorithm consists of two tools: one for the fuzzy rule extraction from measurement data and the other for FLES-predictor tool to test the extracted rule-set. The block schematic diagram of for fuzzy rule-extraction tool is shown in Fig. 1. Input data to the rule extraction tool include: Measured input/output data from an application; desired number of input membership functions IMFs for the input variables; desired number of output membership functions OMFs for the output variables; the normalization-range for the input variables & the normalization range for the output variables.

1.1 The advantages of normalizing input/output variables

The advantages in normalizing input-variables, irrespective of their absolute dynamic ranges, allows us to use just one-IMF-set for all of the input variables. Similarly just one OMF-set for all of the output variables. One can even use just one MF-set for input as well as output variables. The other advantage are: the individual vertices of the IMF-set can be selected such that the vertices are round integer numbers that are easy to understand. One of the application that we have used to verify if the extracted rule set is correct or not was FLES-based measurement of battery state-of-charge SOC. The application has three inputs in1, in2, in3, and one output out1. For a 9-volt battery, the dynamic range of these three input variables were: 8.25-9.59; 8.26-9.62; and 8.26-9.60. With 11 IMFs for in1, the vertices of the IMFs will be: 8.25 + (9.59 – 8.25) * n * / 10, n = 0,..9. The IMF-vertices will be: 8.25; 8.384; .., 9.456, 9.59. The

DOI : 05.IJIS.2016.1.4

span between any two vertices is 0.134. If in1 is normalized 0-100; then the vertices will be 0, 10, 20, .., 90, 100. The span between any two vertices is 10. As one can see, the normalized second set is much easier to deal with than the un-normalized 1st-set. The same thing will apply for the output variables as well.

Fig. 2 shows the FLES-predictor-tool in test or recall or predictive mode or rule-verification mode. Here the outputs generated by the rule-extractor will become the inputs to the predictor. Inputs to the FLES-predictor include three input data files: infile_1.txt, infile_2.txt, and infile_3.txt. The first file infile_1.txt contains: absolute dynamic ranges of each input/output variable; the number of IMFs and OMFs; normalized range for IMFs & normalized range for OMFs; span between any two membership functions or span between any two IMF-vertices; the number of rules. The second input file infile_2.txt contains: the rule set. The third file infile_3.txt contains test inputs. Using the data in infile_1.txt, infile_2.txt, and infile_3.txt the FLES-predictor estimates the output variable value.

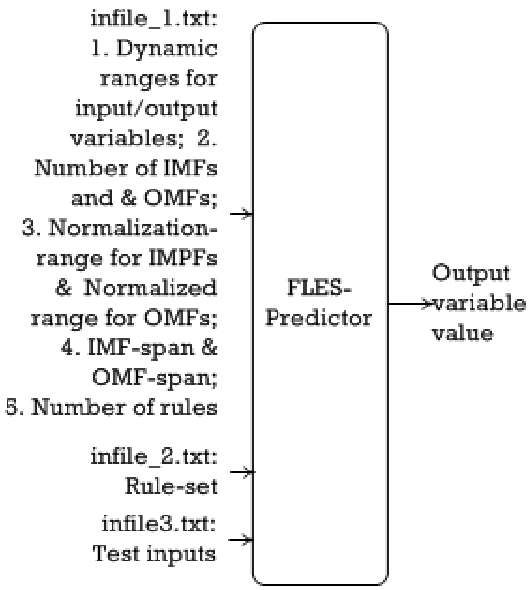

Fig. 2. FLES Predictor

III. FUZZY RULE GENERATION ALGORITHM

The rule-set is generated using the following steps [1-4]:
Step 1: Find dynamic ranges Ri for each variable;
Step 2: Find normalized membership functions MFns;
Step 3: Find input data expressed in normalized form: normalized i/o-pairs;
Step 4: Fuzzify inputs;
Step 5: List fuzzy rules using fuzzified inputs;
Step 6: Find degree of confidence Di for each rule/Resolve the problem of Conflicting Rules;
Step 7: Find the final unique rule-set.

Step 1: Find dynamic ranges Ri for each variable.
Assume we have three-input (x1, x2, x3) and one-output y1 measurement data with m-samples represented as:

s1: x_{11}, x_{21}, x_{31}: y_1;   // measurement sample 1
s2: x_{12}, x_{22}, x_{33}: y_2;   // measurement sample 2
…
sm: x_{1m}, x_{2m}, x_{3m}: y_m; // measurement sample m

Assuming each variable vary from x_{min} to x_{max}; from data search find dynamic ranges Ri for each variable:

Dynamic range R1 for x1 is: $x1_{min}$ to $x1_{max}$;
Dynamic range R2 for x2 is: $x2_{min}$ to $x2_{max}$;
Dynamic range R3 for x3 is: $x3_{min}$ to $x3_{max}$;

Step 2: Find normalized membership functions IMFns & OMFns
Let the specified number of IMFs is N_{imf} and the specified range for the IMFs is R_{imf}. The vertices V_{ini} of the triangular-IMFs are given by:

$$V_{ini} = \left(\frac{R_{imf}}{N_{imf} - 1} \right) * n; \quad n = 0,..,(N_{imf} - 1) \qquad (1)$$

For input IMF-range of 100 and 11 IMFs; the vertices V_{in0}, V_{in1}, Vin2,..,V_{in9}, V_{in10} are: 0, 10, 20,.., 90,100.

Similarly for the output, let the specified number of OMFs is N_{omf} and the specified range for the OMFs is R_{omf}. The vertices V_{outi} of the singleton-OMFs are given by:

$$V_{outi} = \left(\frac{R_{omf}}{N_{omf} - 1} \right) * n; \quad n = 0,..,(N_{omf} - 1) \qquad (2)$$

For output OMF-range of 100 and 11 OMFs; the vertices V_{out0}, V_{out1}, V_{out2},..,V_{out9}, V_{out10} are: 0, 10, 20,.., 90,100.

Step 3: Find input data expressed in normalized form: normalized i/o-pairs.
Normalized input measurements x_n are given by:

$$x_n = \frac{(x - x_{min})}{\Delta x} R_{imf} = \frac{(x - x_{min})}{(x_{max} - x_{min})} R_{imf} \qquad (3)$$

where, x is the un-normalized input data value; x_{min}, x_{max} are the minimum and maximum of x; Δx is the variation-span of x. For x1 = 8.47; x1min = 8.25; x1max = 9.60; Rimf = 100; the value of x1n = (8.47 – 8.25) * 100 / (9.60 – 8.25) = 22/1.35 = 16.30.

Step 4: Fuzzify inputs
Express each normalized input value x_n as function of the IMFs. For any x_n, find between which two-adjacent vertices this value falls-in; then express it as a percentage of those two IMFs; then retain the high-percentage IMF-expression.

DOI : 05.IJIS.2016.1.4
NNGT

For x1n = 39; with 5-IMFs with their vertices at (Vo1:Vo5): 0, 25, 50, 75, 100. The x1n is fuzzified as: the value falls between 25-50 or between IMF1 and IMF2; the value 39 is then expressed as 11/25 of IMF1 or 14/25 of IMF2. In this algorithm we keep 14/25 of IMF2. This is repeated for all thee input variables x1, x2, and x3. In rule extraction algorithms membership function values 11/25 and 14/25 of x1 are represented as $m_1(x1)$ and $m_2(x1)$. Here m represents the membership-function-value. In the above example $m_1(x1)$ is IMF1-value of x1 and m2(x1) is IMF2-value of x1.

Step 5: List fuzzy rules using fuzzified inputs.
Assuming we have 3-input x1, x2, x3 and one output y application. Let the activated-MFs by x1, x2, x3, and y are: IMF2, IMF3, IMF4, and OMF5. Also let the corresponding membership function values are: m2(x1), m3(x2), m4(x3), and m5(y). The rule corresponding to this is written as:

If (x1 is imf2) and (x2 is imf3) and (x3 is imf4) then y is omf5.

The number of rules generated will be equal to the number of measurements. For each measurement there is a rule.

Step 6: Find degree of confidence d_i for each rule / Resolve the problem of Conflicting Rules.
When rules are generated using fuzzified inputs, there will be lots of conflicting rules. Rules considered conflicting if we have the same if-part but with different then-part.

Conflicting rules:
R20: If (x1 is imf2) and (x2 is imf3) and (x3 is imf4) then y is omf5.
R24: If (x1 is imf2) and (x2 is imf3) and (x3 is imf4) then y is omf6.

One way to solve this problem is to find the degree of confidence d_i for each of the conflicting rules then retain the rule with the highest value for d_i. The degree of confidence of a rule is given by the product of the membership function values as follows:
In general if the rule is defined as:
R10: If (x1 is A) and (x2 is B) and (x3 is C) then (y is D); Then the degree of confidence of the R10 is given By:

$$d_{10}=d_{in}*d_{out}=\{m_a(x_1)\ m_b(x_2)\ m_c(x_3)\}*m_d(y) \quad (4)$$

Where d_{10} is the degree of confidence of rule 10; d_{in} is the degree of IMFs (product of input membership function values); d_{out} is the degree of OMF (output membership function value); $m_a(x1)$ membership function value of x1; $m_b(x2)$ membership function value of x2; $m_c(x3)$ membership function value of x3.
With m_a = 14/25; m_b = 19/25; m_c = 20/25; m_d = 8/10:
$d_{10} = d_{in} * d_{out}$ = (14/25) * (19/25) * (20/25) * (8/10) = 0.34 * 0.80 = 0.27

Step 7: Find the final unique rule-set.
Final rule set is a selected list of rules:
Set-A: Select all unique-rules with highest degree of confidence.
Set-B: Pick one rule from each of the conflicting groups with highest degree of confidence.

Combination of the above two rule-sets will become knowledge base of the application's fuzzy logic expert system. This concludes rule generation from measurement data. The next section is to verify if the generated rule set is in fact is the correct rule set representing the application from which the measurement data was obtained. It is called the FLES-Predictor.

IV. FLES-PREDICTOR

The function of this fuzzy logic expert system predictor is to take some test input from the application and estimate corresponding output of the application using an FLES. Overall configuration of an FLES is shown in Fig. 3 [5-7]. It has three elements: Knowledge Base KB, Inference Engine IE, and User interface UI: KB being a systems or an application's knowledge in the form of a rule-set; UI providing real-time i/o signal interface to the application; and IE estimates/infers/computes the output parameter values using the system description in the KB and the inputs from the UI. Numeric-inputs from an application are fuzzified using input membership functions IMFs; and conversely outputs from FLES are defuzzified to generate numeric-outputs to the application using output membership functions OMFs. All of the elements required for developing an FLES for the application are generated by the Rule Generation Tool – knowledge base KB; IMFs; OMFs.

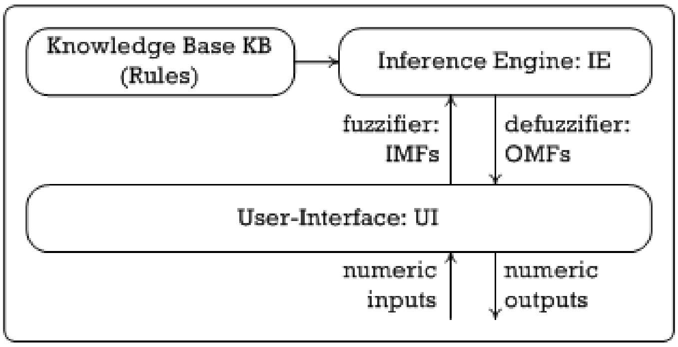

Fig. 3. Overall architecture of an FLES-predictor.

Execution cycle of the Inference Engine:
The Inference Engine senses the input and computes or estimates the output. It accomplishes this using the following sequence of steps:

IE1: Fuzzify inputs
Find input variables as a percentage of the input membership functions IMFs.

IE2: Find activated rule-set Ra
Find all the rules whose if-part is true, i.e., input variable value requirements match with current-input values.

IE3: Find output
Find output using centroid defuzzification formula as follows [3, 4]:

DOI : 05.IJIS.2016.1.4

$$y = \frac{\sum_{i=1}^{K} d_{in\_i} V_{out\_i}}{\sum_{i=1}^{K} d_{in\_i}} \quad (5)$$

where, K is the number of rules activated; d_{in_i} are the input degrees of confidence for each rule (which are the product of the corresponding IMFs); Vout_i are the central-vertices of the output member ship functions OMFs of the activated rules.
Example: Application with three inputs x1, x2 and one out y.
Activated rules:
R1: if (x1 is A) and (x2 is B) then y is C;
R2: if (x1 is D) and (x2 is E) then y is F;
Let membership function values:
$m_A = 0.8$; $m_B = 0.6$; $m_C = 0.7$;
$m_D = 0.5$; $m_E = 0.3$; $m_F = 0.2$;
Let membership function vertices values:
V_{out_1} or $V_{out\_C} = 20$;
V_{out_2} or $V_{out\_F} = 24$;
Here K = 2
Then:
$d_{in\_1} = m_A * m_B = 0.8 * 0.6 = 0.48$
$d_{in\_2} = m_D * m_E = 0.5 * 0.3 = 0.15$
$y = (d_{in\_1} * V_{out\_1}) + (d_{in\_2} * V_{out\_2}) / (d_{in\_1} + d_{in\_2})$
$y = (0.48 * 20) + (0.15 * 24) / (0.48 + 0.15) = 20.95$

We have used this type of FLES-predictor with two applications to verify the generated rule-sets. One, FLES-based State-of-Charge SOC determination; second FLES-based control of an Inverted Pendulum IP problem [3]. Both of them have worked correctly. In this paper we include a short description of the SOC-determination method in the following section; for more detailed description you may refer to [3]. More details on FLES-battery determination one can refer to [7].

V. TEST RESULTS

As described above, one of the application we have used for verifying the rule generation tool is “FLES-based Battery SOC-Determination” [7]. It is an impedance-interrogation method to determine battery SOC. Here you find pulse response of the battery at known SOC-levels. We did 101 measurements at SOC levels of 0, 1, 2, .., 100 using controlled charge/discharge systems. From the pulse response three key features were extracted: x1, x2, x3: min, max, and average. It has one output variable which is battery SOC. So, this application is a 3-input and 1-output application. When rule generation tool is used it generated 12-rule knowledge base KB. We have used this rule-set to develop a FLES-predictor as described in section 3. We ran this FLES-predictor 100-times with inputs with known outputs to see if the predicted values are the same as the measured output values. Eight-IMFs-FLES sytem resulted in best results. The measured values and the predicted values, with extracted rule-set as its KB, are shown in Fig. 4.

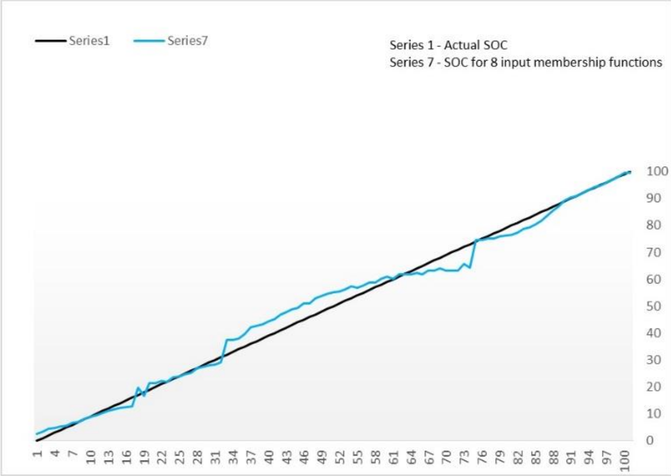

Fig. 4. FLES-Based Battery SOC-Determination.

From the 100-pairs of measured and predicted values, the overall rms-error is estimated as:

$$E_{rms} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (M_i - P_i)^2} \quad (6)$$

Here, E_{rms} is the average rms-error; Mi are the measured values; P_i are the FLES-predicted values, and N is the number of measurement pairs. Statistics of the results are:
Erms: rms-error: 3.33
StDev: Standard deviation: 3.31
Average error: -0.48.

Fuzzy Rule Extraction Tool: Output Simulation Trace
Table A shows the entire output simulation trace of the fuzzy logic-extraction-tool. The output simulation trace of the rule generation tool is listed in Table A. It is implemented in 8-phases. In phase-I, raw measurement data is read into the tool. In phase-II, input variable values are extracted from the raw data. In phase-III, dynamic ranges for the input and the output variables are extracted: in1min, in1max, in2min, in2max, in3min, and in3max. In phase IV, normalized input variable values are computed. In phase V, IMFs and OMF are listed. In phase-VI, for each measurement value the fuzzified values are displayed (absolute values are expressed as function of input membership functions). It also listed as the preliminary-rule set. In phase-VII, the degree of each rule is displayed. In phase-IX, the final and unique fuzzy logic rule set is displayed.

Fuzzy-Predictor-Tool: Output Simulation Trace
Table B shows the entire output simulation trace of the fuzzy-predictor-tool. It has three input data files: measurement data file; normalization data file; and the knowledge base (fuzy-rule-set) file. In phase-I, measurement values are displayed. In phase-II, IMF-set is displayed. In phase-III, the normalized values are displayed. In phase-IV, the measurements are fuzzified. In phase-V, measured (M) and predicted (P) or

DOI : 05.IJIS.2016.1.4

estimated values are displayed. RMS-error is computed from the M-P pairs using (5).

VI. CONCLUSIONS

In this paper an improved fuzzy rule extraction algorithm is presented: 1. It simplifies membership function definitions; 2. It reduces number membership functions needed; and 3. It enables in simplified custom vertex-definitions for the IMFs and OMFs. The rule generation tool & and the fles-predictor-tool set is tested with three case studies with rms-error less than 3.3. While the rms-error is low, the tool-set require further tuning to reduce rms-error even-further.

REFERENCES

[1] Li-Xin Wang and Jerry Mendel. Generating Fuzzy Rules from Numerical Data, with Applications, USC-SIPI-169, University of Southern California, Los Angeles, 1991.
[2] Nikos Pelekis, Babis Theodoulikis, Ioannis Kopanakis, and Yannis Theodoridis. Fuzzy Miner: Extracting Fuzzy Rules from Numerical Patterns, International Journal of Data Warehousing and Mining (IJDWM), ISSN: 1548-3924, 2005.
[3] Mohammad Islam, MS-Thesis: “Extraction of Fuzzy Knowledge Base from Numerical Data”, Drayer Department of Electrical Engineering, Lamar University, Beaumont, Texas, USA, Fall 2015.
[4] G. N. Reddy, Mohammad Mahmudul Islam, and Samin Sobhani. An e-Learning Tool to Extract Rule Set from Measurement Data, Proc. of the 3rd World Congress on Computer Applications and Information Systems WCCAIS-2015, Dubai-UAE, January 9-11, 2016.
[5] G. N. Reddy, Vishnudev Vasanthan, Gurpreet Singh, and Sreelatha Maila. A Modular Fuzzy Logic Expert System for Autonomous Mobile Robots, International Workshop on Artificial Intelligence IWAI-2014, Turkey; August 22-23, 2014: In Proceedings, Scientific Co-operations International Workshops on Electrical and Computer Engineering Subfields: P133-136; ISBN: 978-605-86637-4-9.
[6] G. N. Reddy, Gurpreet Singh, and Vishnudev Vasanthan (2013), Embedding code for a Mobile Fuzzy Logic Controller, Journal of Information Technology Review, Volume 4, Number 4, Pages. 151-156, November, 2013; Print ISSN: 0976-3511 / Online ISSN: 0976-292.
[7] Singh, Gurpreet. 2014. Modern Embedded – Controlled Fuzzy Logic Expert System to Estimate Battery State of Charge (SOC), MS-Thesis, Drayer Department of Electrical Engineering, Lamar University, Spring 2014.

DOI : 05.IJIS.2016.1.4

```
Table A:
Rule Generation Algorithm: Output Simulation Trace:
Application: FLES-Based Battery SOC Measurement
```

```
PHASE I: Raw measurement Data
0.00 8.25 8.25 8.26 8.25 8.26 8.26 8.25 8.26 8.26
8.26 8.26 8.26 8.26 8.26 8.26 8.26 8.26 8.26 8.26
1.00 8.28 8.26 8.28 8.26 8.28 8.28 8.26 8.28 8.28
8.28 8.28 8.28 8.28 8.28 8.26 8.28 8.28 8.28 8.28
…
99.00 9.59 9.59 9.60 9.59 9.60 9.60 9.60 9.60 9.60
9.60 9.60 9.60 9.60 9.60 9.60 9.60 9.60 9.62 9.60
100.00 8.83 8.83 8.83 8.84 8.84 8.84 8.84 8.84 8.84
8.86 8.86 8.86 8.86 8.86 8.86 8.87 8.87 8.87 8.87
```

```
PHASE II: Find input variable values
Computing x1, x2, x3:
x1 = minimum; x2 = maximum; x3 = mean:
x1 = min(pr1, pr2, .. pr20);
x2 = max(pr1, pr2, .. pr20);
x3 = {sum(pr1, pr2, .. pr20)} / 20;
pr: 20-sample battery pulse-response
SOC; x1min; x2max; x3mean:
    0 8.25 8.26 8.26
    1 8.26 8.28 8.28
    2 8.28 8.29 8.29
    3 8.28 8.31 8.29
    4 8.30 8.31 8.30
    5 8.31 8.33 8.31
  ...
   95 9.28 9.29 9.29
   96 9.33 9.37 9.35
   97 9.41 9.42 9.42
   98 9.49 9.53 9.50
   99 9.59 9.62 9.60
  100 8.83 8.87 8.85
```

```
PHASE III: Find dynamic ranges for input variables:
Computing min and max for input variables:
x1 -- minimum; x2 -- maximum; x3 -- mean:
x1_min; x1_max:  8.25 9.59
x2_min; x2_max:  8.26 9.62
x3_min; x3_max:  8.26 9.60
```

```
PHASE IV: Find normalized input variable values:
Normalized xn = int [{(xi - xmin) / delx } * 100];
where, delx = (xmax - xmin);
SOC; x1n_min; x2n_max; x3n_mean; yn:
    0    0    0    0    0
    1    0    1    1    1
    2    2    2    2    2
    3    2    3    2    3
    4    3    3    3    4
    5    4    5    3    5
    6    3    5    4    6
    7    4    6    5    7
   ...
   10    7   10    8   10
   11   10    9    9   11
   12   11   10   10   12
   13   11   11   12   13
   14   12   12   12   14
   15   12   13   13   15
   ...
   99  100  100  100   99
  100   43   44   44  100
```

```
PHASE V: List normalized IMFs:
For all variables: Minimum = 0; Maximum = 100
x1numMFs  = 5;  deltamfx1 = 25
x2numMFs  = 5;  deltamfx2 = 25
x3numMFs  = 5;  deltamfx3 = 25
ynumMFs   = 11; deltamfy  = 10

Corresponding Vertices of the MFs:
Var:       V1      V2      V3
x1mf0       0       0      25
x1mf1       0      25      50
x1mf2      25      50      75
x1mf3      50      75     100
x1mf4      75     100     100

x2mf0       0       0      25
x2mf1       0      25      50
x2mf2      25      50      75
x2mf3      50      75     100
x2mf4      75     100     100

x3mf0       0       0      25
x3mf1       0      25      50
x3mf2      25      50      75
x3mf3      50      75     100
x3mf4      75     100     100

ymf0       0       0      10
ymf1       0      10      20
ymf2      10      20      30
ymf3      20      30      40
ymf4      30      40      50
ymf5      40      50      60
ymf6      50      60      70
ymf7      60      70      80
ymf8      70      80      90
ymf9      80      90     100
ymf10     90     100     100
```

```
PHASE VI: FINDING PRELIMINARY RULE SET
Sample_num; x1MF; x1MFVal; x2MF; x2MFVal; x3MF;
x3MFVal; yMF; yMFVal;
Each of these are max values:
0    0    25   0    25   0    25   0    10
1    0    25   0    24   0    24   0    9
2    0    23   0    23   0    23   0    8
3    0    23   0    22   0    23   0    7
4    0    22   0    22   0    22   0    6
5    0    21   0    20   0    22   1    5
6    0    22   0    20   0    21   1    6
7    0    21   0    19   0    20   1    7
8    0    18   0    19   0    19   1    8
9    0    19   0    17   0    17   1    9
10   0    18   0    15   0    17   1    10
...
90   2    15   2    14   2    14   9    10
91   2    13   3    13   3    13   9    9
92   3    16   3    15   3    16   9    8
93   3    19   3    18   3    19   9    7
94   3    23   3    22   3    22   9    6
95   3    24   3    25   3    24   10   5
96   3    20   3    19   3    19   10   6
97   3    14   3    15   3    14   10   7
98   4    17   4    18   4    17   10   8
99   4    25   4    25   4    25   10   9
100  2    18   2    19   2    19   10   10
```


DOI : 05.IJIS.2016.1.4

NNGT

Table A: Contd..
Rule Generation Algorithm: Output Simulation Trace:
Application: FLES-Based Battery SOC Measurement

Phase VII: Degree of each rule: product of the membership functional values (with 8-IMFs):

0	0	0	0	0	14	14	14	1	0.94
1	0	0	0	1	14	13	13	1	0.81
2	0	0	0	2	12	12	12	1	0.59
3	0	0	0	3	12	11	12	1	0.54
4	0	0	0	4	11	11	11	1	0.46
5	0	0	0	5	10	11	11	1	0.42
6	0	0	0	6	10	9	9	1	0.28
7	0	0	0	7	10	8	9	1	0.25
8	0	0	0	8	7	8	8	1	0.15
9	0	1	1	9	8	8	8	1	0.18
10	0	1	1	10	7	10	8	1	0.19
...									
90	4	4	4	90	11	10	10	1	0.38
91	4	4	4	91	9	8	8	1	0.20
92	5	5	5	92	8	7	8	1	0.15
93	5	5	5	93	11	10	11	1	0.42
94	5	5	5	94	12	13	13	1	0.70
95	5	5	5	95	9	10	9	1	0.28
96	6	6	6	96	10	9	9	1	0.22
97	6	6	6	97	14	13	14	1	0.87
98	6	7	6	98	8	7	8	1	0.15
99	7	7	7	99	14	14	14	1	0.94
100	7	7	7	100	14	14	14	1	0.94

Phase VIII: Sort the degree of the rule: highest deg => lowest deg:
Sorted rule-set:

Sn;	x1MFnum;	x2MFnum;	x3MFnum;	OMF;	DS
1	0	0	0	0.94	
2	7	7	7	99	0.94
3	7	7	7	100	0.94
4	3	3	3	74	0.87
5	3	3	3	75	0.87
6	6	6	6	97	0.87
7	0	0	0	1	0.81
8	1	1	1	16	0.81
9	3	3	3	72	0.81
10	1	1	1	17	0.75
...					
90	1	1	1	23	0.15
91	2	2	2	26	0.15
92	2	2	2	42	0.15
93	5	5	5	92	0.15
94	6	7	6	98	0.15
95	1	2	1	24	0.13
96	4	4	4	85	0.13
97	2	2	2	25	0.12
98	2	2	2	43	0.12
99	3	2	2	44	0.12
100	3	3	3	45	0.12
101	3	3	3	46	0.12

Phase IX: Final unique fuzzy rule set:

Rn;	x1MFnum;	x2MFnum;	x3MFnum;	OMF;	DS
R1	0	0	0	0	0.94
R2	0	1	1	10	0.19
R3	1	1	1	16	0.81
R4	1	2	1	24	0.13
R5	2	2	2	33	0.75
R6	3	2	2	44	0.12
R7	3	3	3	74	0.87
R8	4	4	4	89	0.75
R9	5	5	5	94	0.70
R10	6	6	6	97	0.87
R11	6	7	6	98	0.15
R12	7	7	7	100	0.94

DOI : 05.IJIS.2016.1.4
NNGT

Table B: FLES-Predictor: Test Mode
Application: FLES-Based Battery SOC Measurement

Input data file 1: Normalization data:
// x1min-max; x2min-max; x3min-max; Rnum;
// delta; IMFnum; OMFmin-max
8.25
9.59
8.26
9.62
8.25789
9.59947
12
14.2857
1
8
0
100

Input data file 2: KB-file (Fuzzy rule-set):
Knowledge base infile created from infile_1.txt:

Rule:	0	1	2	3	4	5	6	7	8	9	10	11
	0	0	1	1	2	3	3	4	5	6	6	7
	0	1	1	2	2	2	3	4	5	6	7	7
	0	1	1	1	2	2	3	4	5	6	6	7
	0	10	16	24	33	44	74	89	94	97	98	100

FLES-PREDICTOR: OUTPUT SIMULATION TRACE:

Phase I: Input data:

0	8.25	8.26	8.26
1	8.26	8.28	8.28
2	8.28	8.29	8.29
3	8.28	8.31	8.29
4	8.30	8.31	8.30
5	8.31	8.31	8.31
…			
95	9.28	9.29	9.29
96	9.33	9.37	9.35
97	9.41	9.42	9.42
98	9.49	9.53	9.50
99	9.59	9.62	9.60
100	9.60	9.63	9.61

Phase II: Input data:
Corresponding vertices of the Input Membership functions

IMF0	0.00
IMF1	14.29
IMF2	28.57
IMF3	42.86
IMF4	57.14
IMF5	71.43
IMF6	85.71
IMF7	100.00

Phase III: Normalized input measurements

15	12	13	13
16	12	14	14
17	15	15	15
18	14	16	16
19	17	18	17
20	16	18	18
…			
80	46	45	45
81	46	46	46
82	47	48	47
83	47	49	48
84	49	49	49
85	50	50	51

Phase IV: Fuzzify Inputs: IMFs
...

15	0.16	0.84	0.09	0.91	0.09	0.91
16	0.16	0.84	0.02	0.98	0.02	0.98
17	0.95	0.05	0.95	0.05	0.95	0.05
18	0.02	0.98	0.88	0.12	0.88	0.12
19	0.81	0.19	0.74	0.26	0.81	0.19
20	0.88	0.12	0.74	0.26	0.74	0.26
…						
75	0.99	0.01	0.92	0.08	0.99	0.01
76	0.92	0.08	0.92	0.08	0.92	0.08
77	0.92	0.08	0.92	0.08	0.92	0.08
78	0.85	0.15	0.92	0.08	0.85	0.15
79	0.85	0.15	0.85	0.15	0.85	0.15
80	0.78	0.22	0.85	0.15	0.85	0.15
81	0.78	0.22	0.78	0.22	0.78	0.22
82	0.71	0.29	0.64	0.36	0.71	0.29
83	0.71	0.29	0.57	0.43	0.64	0.36
84	0.57	0.43	0.57	0.43	0.57	0.43
85	0.50	0.50	0.50	0.50	0.43	0.57

Phase V: Measured and Predicted Values

M	P
10.00	9.58
11.00	10.30
12.00	11.02
13.00	11.67
14.00	12.13
15.00	12.52
16.00	12.89
17.00	19.67
18.00	16.79
19.00	21.31
20.00	21.31
…	
80.00	76.60
81.00	77.30
82.00	78.70
83.00	79.40
84.00	80.45
85.00	81.85

DOI : 05.IJIS.2016.1.4