A SELF-ORGANISING FUZZY LOGIC CONTROLLER

Paul N. Ekemezie
Department of Electronic Engineering
University of Nigeria, Nsukka.

Charles C. Osuagwu
Department of Electronic Engineering
University of Nigeria, Nsukka

Abstract

One major drawback of fuzzy logic controllers is the difficulty encountered in the construction of a rule- base that is suitable for the controlled process. In this paper we tackle this problem by proposing an algorithm that allows a designer to initially specify a possibly inaccurate rule-base, which is then made more and more accurate in the course of operation of the control system. The effectiveness of the proposed self-organizing procedure has been investigated by means of computer simulation. The results of the simulation studies indicate that the proposed algorithm is effective.

1. Introduction

The greatest limitation of fuzzy logic control is the lack of a systematic methodology for developing fuzzy rules. The rule-base of a fuzzy logic controller (FLC) often needs to be manually adjusted on a trial-and-error basis in order for the control system to reach the desired level of performance. This tuning process could be quite complicated, and could be time consuming for a first-time FLC developer [**1**].

Apart from the initial tuning problem, there is this general problem in process control, namely, that changes in the operating conditions of a process plant are difficult to predict and adjust for. This means that a FLC needs to be continually tuned if it is to be practically relevant. The on-the-job tuning process is no less cumbersome than the initial tuning of the system. Hence it is desirable to develop a FLC that can adapt its response in relation to variations in the process dynamics.

A lot of work has gone into the development of self-organizing FLCs. Procyk and Mamdani [**2**] proposed a self-organizing FLC in which the rules are deciphered from a predefined performance index table. Pedrycz [**3**] proposed a method of constructing the rule-base of a FLC in which it is viewed as a relational matrix that can be derived by solving pre-identified relation equations that govern the controlled process. Takagi and Sugeno [**4**] developed an approach for extracting control rules from skilled operators, and used the approach to identify rule-based process models from which the rule-base of a FLC can be constructed. Today, none of these methods is popular, which is an indication that they have not b been satisfactory.

Recent researches into adaptive intelligent controllers have resorted to marriage of Fuzzy Logic with other methods of Soft Computing, principally Neural Networks and Genetic Algorithms. The general practice has been to import the ideas developed in Fuzzy Logic setting, with their attendant advantages, into these other technologies. The self-organizing FLC strategy we present in this paper operates in a purely fuzzy logic setting. The advantage of operating in a purely fuzzy logic setting is the ease of design and realization. Since this setting has exported a lot of ideas to other technologies, we want to bring back ideas from one of these technologies that have fuzzy logic orientation. In particular, ideas developed by D. A. Linkens and J. Nie in [**5**], [**6**] and [**7**] under neuro-fuzzy control setting are reshaped and fitted into fuzzy logic setting.

Linkens and Nie [**6**] proposed some self-learning fuzzy controller structures, with the assumption that neither control experts nor teacher signals are available for the control problem. Each of the fuzzy controller structures adopted an architecture similar to that used by traditional model reference adaptive control systems (MRAC), and is basically implemented as a neural network. They claimed that the advantages of the neural network implementation are computational efficiency

and trainable capability of the network paradigm. However, it is known that system modeling by neural networks does not resolve the dimensional problem, i.e., the number of required weights may be large [**8**].

Careful study of the report given by Likens and Nie in [**5**] reveals that the utilization of the simplified fuzzy control algorithm (SFCA) which they developed, in a purely fuzzy logic setting, could make the rule-based paradigm to possess the computational efficiency and trainable capability that is usually associated with the network paradigm. In particular, it seems that the adoption of an architecture similar to that used by MRAC would enable a fuzzy controller based on the SFCA to self-learn its rule-base. In this paper, we want to formulate an adaptive FLC strategy based on these ideas. We want to modify the concepts developed in [5] for neuro-fuzzy controllers and fit them into traditional FLC.

The paper is organised as follows. We define the research problem in section 2. The self-organising FLC algorithm is presented in section 3. Simulation studies of the self-organising FLC algorithm are described in section 4. Section 5 presents the results of the simulation studies and then discusses them. The paper ends with conclusive remarks.

2.THE PROBLEM

Consider a FLC in which there are m error signals, and hence 2m inputs to the fuzzy reasoner. Assume also that each input is decomposed into n overlapping fuzzy regions, so that the fuzzification of the inputs X_i, i = 1, 2, ... , 2m, can be expressed by

$$X_i = \{A_i^1, A_i^2, \ldots, A_i^n\} \quad (1)$$

Where $A_i's$ are the fuzzy subsets corresponding to the ith input. The premise of the rules is formed by the Cartesian product of the input fuzzy sets [the $X_i's$ of (1) above]. The conclusion of each rule is formed by the Cartesian product of the output fuzzy sets U_k. Consequently, each rule has the following form:

$$\text{IF } X_1 \text{ is } A_1^j \text{ and } X_2 \text{ is } A_2^j \text{ and } \ldots \text{ and } X_n \text{ is } A_n^j$$

$$\text{THEN } U_1 \text{ is } B_1^j \text{ and } U_2 \text{ is } B_2^j \text{ and } \ldots \text{ and } U_p \text{ is } B_p^j \quad (2)$$

or, more compactly,

$$(A_1^j \times A_2^j \times \ldots \times A_n^j) \rightarrow (B_1^j \times B_2^j \times \ldots \times B_p^j) \quad (3)$$

(2) and (3) refer to the jth rule; there are p control outputs, and B_i^j is the output fuzzy subset corresponding to the i'th control output.

There is no clearly defined way of tying the Cartesian products of the input fuzzy subsets to the Cartesian products of the output fuzzy subsets. This situation makes the design process a difficult task for an inexperienced designer, since a lot of heuristics is obviously involved. This is the main reason for requiring self-organising ability.

3. THE SELF-ORGANISING FLC

Fig. 1 shows the overall structure of the adaptive FLC. It consists of a conventional FLC plus a reference system and an adaptation mechanism. The fuzzy reasoner is given an *a priori* rule-base that may not be accurate, and a set of input and output fuzzy sets, at the design stage. The function of the adaptation mechanism is to tune the parameters of the fuzzy reasoner so as to attain the desired control objectives. The reference model represents the desired transient and steady-state performances; it provides a prototype for use by the adjustment mechanism.

3.1. Modified representation of the fuzzy system

Let the membership functions for A_i (B_j) in equation (2) be triangular with the apex existing at the middle (or modal point) $M_i(M_j)$ of the support set and width δ_i (δ_j) being the distance from the modal point to either of the left and right edges. Consequently, the membership value at each point $x_i(u_i)$ is given by

$$A_i(x_i) = \begin{cases} 1 - \left[\dfrac{|M_i - x_i|}{\delta_i}\right], & \text{if } |M_i - x_i| \le \delta_i \\ 0, & \text{if } |M_i - x_i| > \delta_i \end{cases} \quad (4)$$

$$B_j(u_j) = \begin{cases} 1 - \left[\dfrac{|M_j - u_j|}{\delta_j}\right], & \text{if } |M_j - u_j| \le \delta_j \\ 0, & \text{if } |M_j - u_j| > \delta_j \end{cases} \quad (5)$$

Only two items of data are required in order to store the fuzzy sets $A_i(B_j)$ in memory; $M_i(M_j)$ and $\delta_i(\delta_j)$. The rule represented by equation (2) can be expressed in the following form:

$$\text{IF } X_1 \text{ is } M^j_{x,1} \text{ and } X_2 \text{ is } M^j_{x,2} \text{ and } \ldots \text{ and } X_n \text{ is } M^j_{x,n} \quad (6)$$

$$\text{THEN } U_1 \text{ is } M^j_{u,1} \text{ and } U_2 \text{ is } M^j_{u,2} \text{ and } \ldots \text{and } U_p \text{ is } M^j_{u,p}$$

In equation (6), the width of the fuzzy sets is not included in the rule because it can always be obtained by accessing the information stored in connection with the appropriate fuzzy set. The fuzzy sets can also be considered to be fuzzy numbers, so that all that need be included in the rule is the ideal or centre value of the fuzzy number. With these representations of the fuzzy system, it now remains to specify how the inference process should be implemented.

The inference process begins with matching the fuzzified input patterns with the condition part of the rules in the rule-base. The truth value of the predicate of the rule is determined by means of the standard minimum rule for conjunction of fuzzy sets. The inferred output fuzzy sets are then correlated to the truth value of the predicate by means of either the correlation minimum method or any other means, such as the correlation product method **[9].** The modified output fuzzy sets are then copied into the output variables' fuzzy sets. Where the output fuzzy sets overlap, or two or more rules infer the same fuzzy set for the same control output, the standard rule for disjunction of fuzzy sets will be applied to determine the surface of the output variables' fuzzy sets. In the adaptation strategy being proposed, the inference procedure will be modified, because a defuzzification technique that is not exactly the standard centroid method will be employed. Since the ultimate result of the fuzzy reasoning process is the defuzzified output, it is necessary first of all to choose a defuzzification method that is suitable to the proposed strategy.

The weighted averaging method of defuzzification described in [5], which is similar to the centroid method, is particularly suitable. In this method, the jth control output is given by

$$u_j = \frac{\sum_{q=1}^{Q} \mu^q M_u^q}{\sum_{q=1}^{Q} \mu^q} \quad (7)$$

In equation (7), Q rules have been triggered in the inference process; μ^q is the truth value of the predicate of the qth rule and M_u^q is the centre value of the consequent fuzzy set.

Having selected a method of defuzzification, the issue of how to determine the surface of the control output's fuzzy set where the output fuzzy sets overlap, or if more than two rules infer the same fuzzy set, no longer arise, because the surface of the fuzzy set need not be known for the selected defuzzification method.

3.2. The reference model

The desired performance of the control system is designated by the reference model, which specifies what the process responses should be when it is subjected to the same command signal as the reference model [**6**]. The model should be as simple as possible so that its output can be readily determined by adjusting a few parameters in the model. The other requirement in selecting the model is that the desired performances should be achievable. For example, there is no sense in demanding an instant response for a dynamic process with pure time delay subject to a step command signal. For a single-input-single-output process, the following linear first-order and second-order models would normally be used.

$$H(s) = \frac{Ke^{-\varpi}}{s + \frac{1}{T}} \qquad (8)$$

$$H(s) = \frac{\omega_n^{\,2} e^{-\varpi}}{s^2 + 2\xi\omega_n + \omega_n^{\,2}} \qquad (9)$$

The models given above are linear, but do not suggest that the process itself must be linear. Yet they do suggest that some knowledge about the process should be available. With some prior knowledge of the process and well-known linear control theory, it is not difficult to determine the parameters in the above models by specifying the desired time domain indices or by allocating the pole positions in the s-plane [**6**].

In equation (8), the term T is the time constant, an indication of how fast the system tends to reach the final value, and τ is the delay time. In equation (9) the positive coefficient ω_n is the *undamped natural frequency*, the coefficient ε is the *damping ratio* of the system and τ is the delay time.

Control systems are generally designed with damping less than one, that is, oscillatory step response [**10**]. This type of response is characterised by several performance indices one of which is the settling time. The settling time is most often defined as the time required for the response to a unit-step function input to reach and remain within a specified percentage (frequently 2 or 5 percent) of its final value [**11**]. It is a measure of speed of response of the control system. For a 2 percent tolerance band the settling time is given by the approximate relationship.

$$t_s = \frac{4}{\xi\omega_n} \qquad (10)$$

3.3. Adaptation mechanism

In normal operation of the adaptive FLC shown in Fig.1, the fuzzy reasoner infers the current control action every sampling instant. The reference model and the learning mechanism are activated only when there is a step change in error, of sufficient magnitude. Each activation forms an iteration of the adaptation mechanism. This assumes that both the process output and the reference model output were in steady state before the step change occurs. Both the closed-loop control system and the reference model are given the same set point. During each sampling period k within an iteration, the adaptation

mechanism compares the process output y_p and reference model output y_r as follows

$$e_L(k) = y_r(k) - y_p(k) \qquad (11)$$

where e_L is the learning error. If the process response does not exhibit any dead time, the adaptation mechanism also obtains from the fuzzy reasoner the rules triggered during the last sampling period. Since every rule in the rule-base is represented in the form of equation (6), the adaptation mechanism then calculates a new centroid for each rule that contributed to the learning error e_L according to the following:

$$M_u(k) = M_u(k-1) + \rho.e_L(k).\mu \qquad (12)$$

where $M_u(k)$ is the new centroid μ, is the predicate truth value of the winner rule and p is the learning rate.

Suppose there is dead time in the process response, and the estimated delay time is λ sampling periods. In this case the adaptation mechanism stores an array of previously triggered rules obtained from the fuzzy reasoner with their associated predicate truth values. In each sampling period it recalculates the centroid for every rule that was triggered λ sampling periods before the present sampling period, but makes use of the current learning error. The updating of each rule's centroid can be expressed as follows:

$$M_u(k) = M_u(k+\lambda) + \rho.e_L(k)\mu \qquad (13)$$

4. Simulation studies

The objective of the simulation studies is to investigate the performance of the self-organising algorithm. Actually, the objective is to see how well the adaptive control strategy forces the control system to approximate the reference model. A simulation program that implements the algorithm was created for this purpose. The simulation program used a learning rate $\rho = 0.05$.

4.1. The control system

Here, we want to specify the control system being simulated. The specifications given here were incorporated in a computer simulation program.

The Process:

The controlled process is assumed to be described by the nonlinear differential equation

$$\frac{d^2y(t)}{dt^2} = -1.6\frac{dy(t)}{dt} - 0.1y^2(t) + u(t) \qquad (14)$$

where y(t) is the process output and u(t) is the control signal. The differential equation model of the process is solved numerically in the computer by means of the Runge-Kutta-Nystrom technique [**12**].

Input Formation:

The process represented by Eq. (14) is single-input-single-output (SISO). Therefore, there is only one input (error signal) to the controller. The control error is extended into two signals, *error* and *change-in-error.* The *error* signal is calculated in the following manner

$$e(kT) = SP - y(kT) \qquad (15)$$

where SP(=1) is the set point for the control system and y(kT) is the numerical solution of equation (14) in the kth sampling period. The *change –in-error* signal is computed as follows.

$$c(kT) = e(kT) - e((k-1)T) \qquad (16)$$

e(kT) and c(kT) serves as the inputs to the fuzzy reasoned.

Fuzzification:

The universe of discourse for the two inputs, *error* and *change-in-error*, is normalised and decomposed into seven fuzzy regions namely, *negative large* (NL), *negative medium* (NM), *negative small* (NS) *zero* (ZR), *positive small* (PS), *positive medium* (PM) and *positive large* (PL). The normalization of an input or output variable involves multiplication with a suitable gain factor g_i. The scale factors are fixed at constant values, as follows:

$g_{error} = 1$

$g_{errorchange} = 2$

$g_{output} = 2$

The fuzzification of a crisp input value is done by means of Eq. (4). For the input linguistic variables, the centroid of the fuzzy regions and the span are as follows:

M_j^i = {-1, -2/3,-1/3, 0,1/3, 2/3, 1}

$\delta_i = 1/3$ (18)

Rule-base:

Having specified the fuzzy sets, the next task in the FLC design process is to elicit the control rules. Since the fuzzy reasoner has only two inputs, each of which can fall into any of seven fuzzy regions, writing the rules simply involves deciding what the output fuzzy set should be for each possible input combination. From the interaction of the two inputs, a seven-by-seven matrix can be constructed showing the output for each input combination (Table 1). This matrix is the rule-base for the fuzzy controller.

Definition of the initial rule-base makes use of Table 1. The rule-base is stored as in Table 1, but with the control output linguistic labels (NL, NM, NS, ZR, PS, PM, PL) replaced by the centroids of the implied fuzzy sets. For purposes of defining the initial rule-base, the control output is also decomposed into seven fuzzy regions, with centroids given as

M_j =={-1, -2/3, -1/3, 0, 1/3, 2/3, 1} (19)

The initial rule-base is represented as follows.

rulebase [7][7]=
{-1, -1, -67, -67, -33, -33, 0},
{-1 -67 -67 -33 -33 0 0},
{-67 -33 -33 0 0 .33 .33},
{-.33 -.33 0 0 .33 .67 .67},
{-.33 0 0 .33 .33 .33 .67 .67},
{0 0 .33 .33 .67 .67 1},
{0 .33 .33 .67 .67 1 1}

(20)

Defuzzification: The final step in the design process is the selection of a method of *defuzzification*. In the present case, the weighted averaging method [Eq.(7)] is employed.

Reference model:

Two sets of parameters were used for the reference model during the investigations, namely

ω_n ==2, ξ ==0.75.

ω_n ==3, ξ ==0.75.

In selecting the sets of parameters, some care was taken to ensure that their values are such that the adaptation mechanism can force the control system to approach the reference model output. The settling times (about 2.67 s in the first set and 1.78 s in the second set) were put into consideration.

4.2. Simulation procedure

The simulation program was run for fifty cycles, so as to enable the learning process to possibly stabilize. By stabilization of the learning process is meant that there will not be any appreciable adjustment of the rule-base from cycle to cycle. In each cycle the control system is given a step input and the cycle time is such that the system settles to a steady state response. The cycle time used for the simulation study was 5 seconds, even though any time greater than the settling time could

have served equally well. (The simulation program was designed in such a way that the rule-base resulting from a cycle is available for use in the next cycle.) The response of the system was recorded after the first, twentieth and fiftieth cycles. The resulting rule-base after the first, twentieth and fiftieth cycles were also recorded, to find out how the adaptation mechanism modified the rule-base

5. Results and discussion

Fig. 2 shows the output responses of the process in comparison with the reference model after the first, twentieth and fiftieth cycles of the adaptation procedure, for the reference model parameters $\omega_n = 2$, $\xi = 0.75$. Table 2 shows the adjustment of the rule-base after the first, tenth, twentieth and fiftieth cycles of the adaptation procedure for the reference model parameters, $\omega_n = 2$, $\xi = 0.75$.

Fig. 3 shows the output responses of the process in comparison with the reference model after the first, twentieth and fiftieth cycles of the adaptation procedure, for the reference model parameters $\omega_n = 3$, $\xi = 0.75$.

In Fig. 2, it can be noticed that as the number of iterations increases, the output of the process approaches the reference model output. In fact, the curve for the fiftieth iteration indicates a system that is faster than the reference model in terms of the rise and settling times, even though the speed enhancement is accompanied by an increase in overshoot. The increase in overshoot, however, is quite tolerable. Comparing the rise times of the curve for the fiftieth iteration with that of the reference model, it can be seen that whereas for the reference model $t_r \doteq 1.4$ s, the curve for the fiftieth iteration indicates that $t_r \doteq 1.2$ s. Comparing the settling times of the curve for the fiftieth iteration with that of the reference model, it can be seen that whereas for the reference model $t_s \doteq 3$s, the curve for the fiftieth iteration indicates that $t_s \doteq 2.4$s.

In Fig. 3, it can also be noticed that as the number of iterations increases, the output of the process approaches the reference model output. However, the curve for the fiftieth iteration indicates that the system could not follow the reference model as well as did the responses in Fig. 2. Comparing the rise times of the curve for the fiftieth iteration with that of the reference model, it can be seen that whereas for the reference model $t_r \doteq 0.9$ s, the curve for the fiftieth iteration indicates that $t_r \doteq 1$s. Comparing the settling times of the curve for the fiftieth iteration with that of the reference model, it can be seen that whereas for the reference model $t_s \doteq 2$ s, the curve for the fiftieth iteration indicates that $t_s \doteq 3.2$s. This means that the choice of a reference model should be done realistically. The reference model should be such that the self-organizing controller could possibly force the process to follow it.

In Table 2, it can be seen that the adaptation mechanism adjusts its rule-base in response to the learning error. Starting with the initial rule-base shown in Eq. (20), the adaptation mechanism continually adjusted the fired rules as control progressed. Rules that were not fired were not affected by the adaptation mechanism. Hence, the self-organizing controller was able to adjust itself in response to the prevailing conditions of the controlled process.

Taking a closer look at table 2, one can see which of the rules that were adjusted by the adaptation mechanism, and which were not. All the rules represented by the first and second columns of the rule-base matrix were not influenced by the adaptation mechanism. In the third column, only R_{31} and R_{32} were not adjusted. The adjustment of the rules in the third column started right from the first cycle, and considerable adjustment was done. In the fourth column, only R_{41} and R_{42} were not adjusted. Also, the adjustment of the rules in the fourth

column started right from the first cycle, and considerable adjustment was done. In the fifth column, three of the rules are adjusted, R_{53}, R_{54} and R_{55}. There was no adjustment of the rules in the sixth column. In the seventh column, only R_{77} is adjusted, and by a significant amount.

The adjustment pattern indicated in table 2 is not surprising, when one considers that a control system maintains the controlled variable mostly close to the set point. That is why columns three and four of the F AM experienced the greatest levels of adjustment. The fact that R_{77} is adjusted might look as a surprise, but the surprise goes when one remembers that R_{77} is activated when the system begins from an inert (zero) initial conditions. In this state the system is not really within control. The adjustment mechanism increased the value of R_{77} in such a way as to quickly bring the system under control. Overall, one can say that 14 out of the 49 rules were adjusted. This implies that only about 29 percent of the rules really participated in the control process. This is why in systems with large number of inputs to the fuzzy reasoner, due to unwieldy number of rules that would be implied, the self-organising controller may be given the ability to determine which of the rules to retain in memory.

CONCLUSION

A self-organising fuzzy logic control algorithm has been proposed. The algorithm utilizes a modification of the standard fuzzy logic controller structure. The performance of the proposed self- organising procedure has been investigated by means of computer simulation. The results obtained from the simulation studies indicate that the algorithm achieves the desired objective. However, care must be taken when selecting the parameters of the reference model, since the self- organising FLC can only force the system to approximate a realistic model.

The great advantage of the proposed self-organizing FLC is that the designer does not have to worry so much about the accuracy of the initial (untuned) rule-base. The control system can start up with a rule-base that may be quite unsuitable for the controlled process. Self-adaptation can then be used to tune it up to a good level of performance.

References

[1] W. C. Daugherty et al, "Performance evaluation of a self- tuning fuzzy controller," Proc. IEEE Int. Conf. on Fuzzy Sys., pp. 389-397, March 1992.

[2] T. Procyck and E. H. Mamdani, A linguistic self-organising controller," Automatica, Vol. 15, pp. 15-30, 1979.

[3] W. Pedrycz, "Design of fuzzy control algorithms with the aid of fuzzy models", It in Industrial Applications of Fuzzy control, ed. by M. Sugeno. Amsterdam: North_Holland, 1985.

[4] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Trans. Sys. Man and Cyber. Vol. SMC-15, pp. 116- 132, 1985.

[5] J. Nie and D. A. Linkens, "Fast self-learning multivariable fuzzy controllers constructed from a modified CPN network," Int. J. Control. vol. 60, no. 3, pp 369- 393, 1994.

[6] J. Nie and D. A. Linkens, "A hybrid neural-network-based self- organizing controller", Int. J. Control. vol. 60, no. 2, pp 197- 222, 1994.

[7] D. A. Linkens and J. Nie, "Back- propagation neural-network based fuzzy controller with a self learning teacher," Int. J. Control. vol. 60, no. 1, pp 17-39, 1994.

[8] M. S. Ahmed and M. F. Anjum, "Neural-net-based direct self-tuning control of nonlinear plants," Int. J. Control, Vol. 66, no. 1, pp. 85-104, 1997.

[9] B. Kosko, "Addition as fuzzy mutual entropy," Information Sciences, vol. 73, pp. 273-284, Oct. 1993.

[10] I. J. Nagrath and M. Gopal, Control Systems Engineering, 2 ed. New Delhi: Wiley Eastern, 1981.

[11] J. J. DiStefano, III, Schaum's Outline of Feedback and control Systems. New York: McGraw-Hill, 1976

[12] E. Kreysig, Advanced Engineering Mathematics, 8 ed. New York: John Wiley & sons, 1999.

Table 1. Rulebase matrix.

Change-in-error / Erro	NL	NM	NS	ZR	PS	PM	PL
NL	NL	NL	NL	NM	NM	NS	NS
NM	NL	NL	NM	NM	NS	NS	ZR
NS	NL	NM	NS	NS	ZR	ZR	PS
ZR	NM	NS	ZR	ZR	PS	PS	PM
PS	NM	NS	ZR	PS	PS	PM	PL
PM	NM	ZR	PS	PS	PM	PL	PL
P	ZR	PS	PS	PM	PL	PL	PL

Table 2. **Adjustment of the rule-base after (a) first, (b) twentieth, and (c) fiftieth simulation cycles.**

(a)

-1	-1	-0.667	-0.667	-0.333	-0.333	0
-1	-0.667	-0.667	-0.667	-0.333	-0.333	0
-0.333	-0.667	-0.335	-0.339	0	0	0.333
-0.333	-0.333	0.002	0.045	0.336	0.333	0.667
0	-0.333	0.017	0.378	0.336	0.667	
0	0	0.357	0.364	0.667	1	1
0.333	0.333	0.342	0.684	1	1	1

(b)

-1	-1	-0.667	-0.667	-0.333	-0.333	0
-1	-0.667	-0.667	-0.667	-0.333	-0.333	0
-0.333	-0.667	-0.459	-0.881	-0.039	0	0.333
-0.333	-0.333	-0.217	0.101	0.341	0.333	0.667
0	-0.333	0.101	0.686	0.37	0.667	1
0	0	0.633	0.679	0.667	1	1
0.333	0.333	0.459	0.922	1	1	1.0

(c)

-1	-1	-0.667	-0.667	-0.333	-0.333	0
-1	-0.667	-0.667	-0.667	-0.333	-0.333	0
-0.333	-0.667	-0.685	-1.546	-0.077	0	0.333
-0.333	-0.333	-0.809	0.119	0.412	0.333	0.667
0	-0.333	-0.134	0.907	0.449	0.667	1
0	0	0.787	0.834	0.667	1	1
0.333	0.333	0.565	1.152	1	1	1.02