A Neuro-fuzzy Controller Applying to a Cuk Converter[1]

L. K. Wong* F. H. F. Leung P. K. S. Tam

Department of Electronic Engineering
Hong Kong Polytechnic University
Hung Hom, Hong Kong

* enkin@encserver.en.polyu.edu.hk

Abstract - **A dc-dc power converter is difficult to control due to its non-linearities and parameter uncertainties. To tackle the problem, a neuro-fuzzy controller is proposed. The controller utilizes the error voltage and the change of error voltage as inputs, and outputs the duty cycle of the PWM switch for controlling the converter. Instead of relying on expert knowledge, some heuristic rules are derived with the membership functions of the fuzzy variables tuned by a neural network. After an off-line training, the neuro-fuzzy controller can be applied to regulate a Cuk converter. Simulation results are to be given to demonstrate the performance of the controller.**

I. INTRODUCTION

Switching dc-dc converters are highly non-linear plants. The non-linearities are mainly due to the switching actions and the parameter variations caused by external disturbances. In order to achieve the necessary regulation, a controller in a feedback loop is needed. Conventionally, a small-signal approximation is used to obtain a linearized model of the power converter [1] so that a simple compensator can be designed [3]. However, some models are non-minimum phase systems. Besides, perturbations are often so large that the small signal approximation cannot be valid. Even if the linearized model is accurate, the plant model parameters are varying during operation, making the parameters to be inevitably uncertain. All these factors introduce many difficulties for the problem of controlling switching converters.

In order that the regulated converter has good transient and steady-state responses, a controller with the following properties are desirable:

1) it does not rely on an accurate model of the plant;
2) it is robust to the uncertainties of the plant parameters.

In recent years, fuzzy logic controllers had been used in many areas. Unlike conventional controllers, a fuzzy logic controller does not require an exact model of the plant. Instead, a set of linguistic rules is used to derive the control strategy. These rules come from knowledge and properties of the plant and affect directly the performance of the controller. Hence, a fuzzy logic controller can be more capable of tackling plants with parameter uncertainties and/or undesirable non-linearities than conventional controllers.

Many methods can be used to design a fuzzy controller. The most commonly used method is based on expert knowledge. However, this method is very much dependent on past experience and may not give an optimal performance. An alternative approach is to design the fuzzy controller based on some heuristic rules. A certain learning algorithm is then used to fine tune the membership functions of the fuzzy variables [4]. This approach owns the advantage that a good design can be obtained automatically without requiring expert knowledge.

A fuzzy logic controller which is trained by a neural network is to be applied on a switching mode power converter. The design procedures are to be described and the performance of the controller is illustrated through an application example of a regulated Cuk converter.

II. FUZZY LOGIC CONTROLLER

In this paper, the fuzzy logic controller accepts error $e(k)$ and the change of error $\Delta e(k)$ as inputs and generate a control signal $d(k)$ at the k-th sampling instant. The error $e(k)$ and the change of error $\Delta e(k)$ are defined as:

$$e(k) = \mathrm{Vref} - Vo(k) \quad (1)$$

$$\Delta e(k) = [e(k) - e(k-1)]f \quad (2)$$

where Vref is the reference voltage, $Vo(k)$ is the output voltage, f is the sampling frequency, and $d(k)$ is the duty cycle of the converter. The fuzzy logic controller is composed of three parts: fuzzifier, fuzzy inference engine and defuzzifier. The block diagram of the control system is shown in Fig. 1.

Fuzzification is a conversion of the crisp inputs into fuzzy levels described by membership functions. The membership functions are in bell shapes. They are characterized by two parameters T and w, which are the center value and a measure of the width of the bell respectively [4].

[1] This work was supported by the Earmarked Grant from Research Grants Council, UGC, for the project of number HKP 62/94E

0-7803-3026-9/95 $4.00 © 1995 IEEE

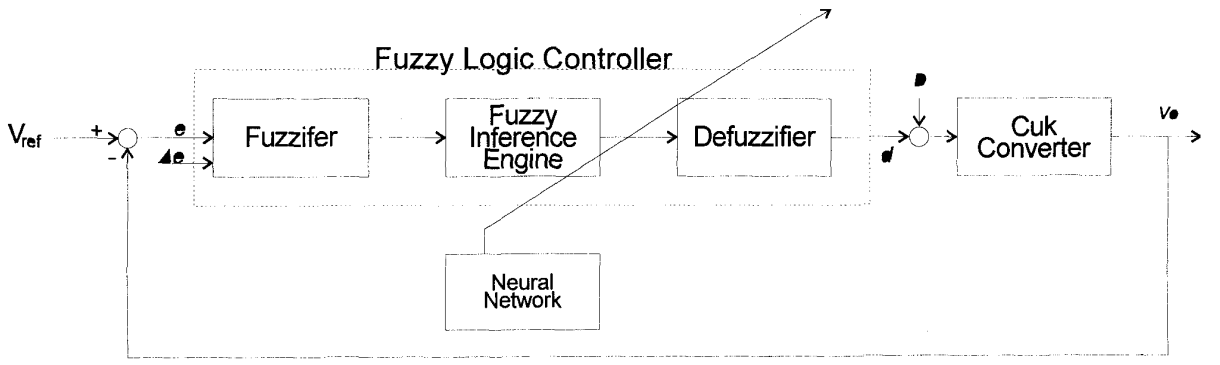

Fig. 1 A block diagram representing the fuzzy control system

The degree of membership μe associated with a fuzzy level of input e is defined as:

$$\mu_e = \exp\left(-\frac{(e-T)^2}{w^2}\right) \tag{3}$$

The range of μe is from 0 to 1, and each input has five fuzzy levels. Similar definitions are applicable to the input Δe. The degree of membership of every fuzzy level is then fed to the fuzzy inference engine based on some defined linguistic rules so that the control action can be determined. There are altogether 25 rules. Each rule accepts one fuzzy level corresponding to each input. An AND operation is realized by taking the minimum of the degrees of membership. A typical fuzzy rule is of the following form:

IF $e(k)$ is *zero* AND $\Delta e(k)$ is *zero* THEN $\Delta d(k)$ is *zero*

The defuzzifier collects the fuzzy outputs from all rules to derive the actual crisp output $d(k)$ based on the following equations:

$$\Delta d(k) = \sum_{m=1}^{25} \mu_m \Delta d_m \tag{4}$$

$$d(k) = \Delta d(k) + d(k-1) \tag{5}$$

where μm and Δdm are the degree of membership after the AND operation and the output fuzzy level of rule m respectively. Since the controller is used as a regulator, it should has an integral effect to ensure a zero steady-state error. This is realized by deriving $d(k)$ based on (5).

III. NEURO-FUZZY CONTROLLER

A 4-layer neural network is formed by sets of nodes connected together. Fig. 2 shows a general structure of a node in the neural network. The inputs to the nodes are $u1$, $u2$, ..., uk with weightings $w1$, $w2$, ..., wk respectively. The total input x and the output y of the node are given by:

$$x = \sum_{i=1}^{k} u_i w_i \tag{6}$$

$$y = \mathrm{f}(x) \tag{7}$$

where f is called the activation function. The desired output d is compared with the actual output y and an error e is generated. This error can be used in training. If the node is the output node, the desired output is given. If the node is in a hidden layer of the network, the error should be back-propagated from the top layer. The back-propagation algorithm will be discussed later.

A neuro-fuzzy controller is formed by implementing the fuzzy logic controller described in section 2 with a neural network which can be tuned by the back-propagation learning algorithm. The architecture of the neuro-fuzzy controller is shown in Fig. 3.

Layer 1 is the input layer which accepts the crisp inputs $e(k)$ and $\Delta e(k)$. The output of each node is the corresponding input multiplied by a constant. It is used to scale the inputs in order to match the universe of discourse.

Layer 2 implements the fuzzification. There are five nodes for each input which associate with five fuzzy levels, making a total of 10 nodes in this layer. All weightings are set to unity. The input-output relation, $y_{mem,n}(k) = \mu e(k) = \mathrm{f}(e(k), T_{mem,n}, w_{mem,n})$, of these nodes are defined in (3) with $e(k) = x_{mem,n}(k)$ where n is from 1 to 5. The same applies to the input $\Delta e(k)$ for n ranges from 6 to 10.

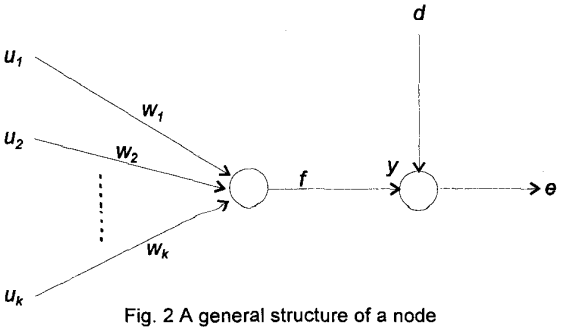

Fig. 2 A general structure of a node

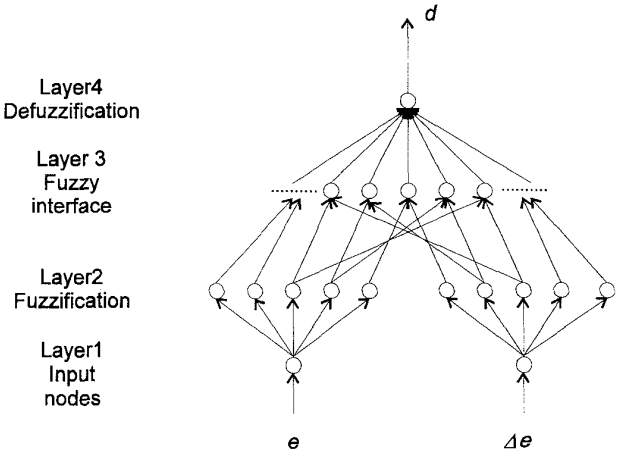

Fig. 3 Structure of the neuro-fuzzy controller

Layer 3 is the fuzzy inference layer. The inputs of the nodes in this layer are two of the output nodes of layer 2, one corresponding to input $e(k)$ and the other corresponding to input $\Delta e(k)$. There are totally 25 nodes in this layer. The weightings are also set to unity and the output, $y_{inf,m}(k)$, is the minimum of the two inputs.

Layer 4 is the defuzzification layer. The total input of the node is $x_{def}(k) = \Delta d(k)$, which is defined in (4). Since μm are the outputs of the nodes in layer 3, the corresponding weightings $w_{def,m}(k)$ are set to Δdm. The output of the node, $y_{def}(k) = d(k)$, is the sum of the input of the node and the previous output.

$$y_{def}(k) = x_{def}(k) + y_{def}(k\text{-}1) \tag{8}$$

IV. TRAINING

Back propagation algorithm [5] is used to train the neuro-fuzzy controller. The parameters to be trained are output fuzzy levels Δdm, center values $T_{mem,n}$ and widths $w_{mem,n}$ of the input membership bell shape functions. However, this training algorithm requires an error, defined as the difference between the desired duty cycle and the actual duty cycle output by the controller, which is not available because the desired duty cycle cannot be obtained directly. The only desired value obtainable is the reference voltage of the converter output Vref. In order to make use of the error between Vref and the actual output voltage Vo, another learning network is used to learn the characteristic of the converter so that the error can be back propagated. This learning network is a three layer network with one node in the output layer, six nodes in the hidden layer and five nodes in the first layer. The inputs of the network are delta duty ratios $\Delta d(k)$, $\Delta d(k\text{-}1)$, $\Delta d(k\text{-}2)$, and output voltages $Vo(k\text{-}1)$, $Vo(k)$. The output of the network is $Vo(k)$. The activation functions of the output layer and the hidden layer are both sigmod functions, while that of the input layer is unity. The initial weightings are assigned in random. The learning network is described by the following equations.

$$y_{out}(k) = \mathrm{S}(x_{out}(k)) \tag{9}$$

$$x_{out}(k) = \sum_{j=1}^{6} w_{out,j}(k) y_{hid,j}(k) \tag{10}$$

$$y_{hid,j}(k) = \mathrm{S}(x_{hid,j}(k)) \tag{11}$$

$$x_{hid,j}(k) = \sum_{i=1}^{5} w_{hid,j}(k) y_{in,i}(k) \tag{12}$$

where the subscripts *out*, *hid* and *in* refer to output, hidden and input layer of the network respectively and $\mathrm{S}(x)$ is a sigmod function given by:

$$\mathrm{S}(x) = \frac{1}{1+\mathrm{e}^{-x}} \tag{13}.$$

An off-line training of the learning network is carried out before training the neuro-fuzzy controller. After that, the weightings of the learning network is kept unchanged and the training of the neuro-fuzzy controller proceeds. To train the neuro-fuzzy controller, the sum-squares error E for one epoch is defined:

$$E = \frac{1}{2}\sum_{\text{all k}} e(k)^2 \tag{14}$$

$$\begin{aligned} e(k) &= \text{Vref} - Vo(k) \\ &= \text{Vref} - y_{out}(k) \end{aligned} \tag{15}$$

The training rules are derived as follows. For the defuzzification layer,

$$w_{def,m}(k) = \Delta w_{def,m}(k) + w_{def,m}(k\text{-}1) \tag{16}$$

$$\Delta w_{def,m}(k) = -\eta \frac{\partial E}{\partial w_{def,m}(k)} \tag{17}$$

$$\frac{\partial E}{\partial w_{def,m}(k)} = \frac{\partial E}{\partial y_{in,1}(k)} \frac{\partial y_{in,1}(k)}{\partial x_{def}(k)} \frac{\partial x_{def}(k)}{\partial w_{def,m}(k)} \tag{18}$$

Note that $y_{in,1}(k)$ is the same as $x_{def}(k)$.

Therefore

$$\frac{\partial E}{\partial x_{def}(k)} = \frac{\partial E}{\partial e(k)} \frac{\partial e(k)}{\partial y_{out}(k)} \frac{\partial y_{out}(k)}{\partial x_{out}(k)} \sum_{j=1}^{6} \frac{\partial x_{out}(k)}{\partial y_{hid,j}(k)} \frac{\partial y_{hid,j}(k)}{\partial x_{hid,j(k)}} \frac{\partial x_{hid,j(k)}}{\partial y_{in,1}(k)} \tag{19}$$

From (9) to (15),

$$\frac{\partial E}{\partial w_{def,m}(k)} = -e(k)\mathrm{S}'(x_{out}(k)) \sum_{j=1}^{6} \left[w_{out,j}(k) \mathrm{S}'(x_{hid,j}(k)) w_{hid,1}(k) \right] y_{inf,m}(k) \tag{20}$$

where $\mathrm{S}'(x)$ is the first derivative of the sigmod function.

The weightings of the fuzzy inference layer is set to unity so no training is needed. However, the nodes in this layer will take part in propagating the error to the fuzzification layer. Since the AND operation selects the minimum input

of the node, the corresponding node in the fuzzification layer will be trained while the one not selected will not be trained.

For the fuzzification layer, two parameters, w and T, are to be trained. The training rules are derived as follow:

$$\Delta w_{mem,n}(k) = -\eta \frac{\partial E}{\partial w_{mem,n}(k)} \tag{21}$$

$$\frac{\partial E}{\partial w_{mem,n}(k)} = \frac{\partial E}{\partial y_{mem,n}(k)} \frac{\partial y_{mem,n}(k)}{\partial w_{mem,n}(k)} \tag{22}$$

$$\frac{\partial E}{\partial y_{mem,n}(k)} = \frac{\partial E}{\partial x_{def}(k)} \sum_{m=1}^{25} \frac{\partial x_{def}(k)}{\partial y_{inf,m}(k)} \frac{\partial y_{inf,m}(k)}{\partial y_{mem,n}(k)} \tag{23}$$

$$\frac{\partial x_{def}(k)}{\partial y_{inf,m}(k)} = w_{def,m}(k) \tag{24}$$

$$\frac{\partial y_{inf,m}(k)}{\partial y_{mem,n}(k)} = \begin{cases} 0 & \text{if } y_{mem,n}(k) \text{ is not selected} \\ 1 & \text{if } y_{mem,n}(k) \text{ is selected} \end{cases} \tag{25}$$

$$\frac{\partial y_{mem,n}(k)}{\partial w_{mem,n}(k)} = 2\frac{(x_{mem,n}(k) - T_{mem,n})^2}{w_{mem,n}^3} \exp\left(-\frac{(x_{mem,n}(k) - T_{mem,n})^2}{w_{mem,n}^2}\right) \tag{26}$$

Similarly,

$$\Delta T_{mem,n}(k) = -\eta \frac{\partial E}{\partial T_{mem,n}(k)} \tag{27}$$

$$\frac{\partial E}{\partial T_{mem,n}(k)} = \frac{\partial E}{\partial y_{mem,n}(k)} \frac{\partial y_{mem,n}(k)}{\partial T_{mem,n}(k)} \tag{28}$$

$$\frac{\partial y_{mem,n}(k)}{\partial T_{mem,n}(k)} = 2\frac{x_{mem,n} - T_{mem,n}}{w_{mem,n}^2} \exp\left(-\frac{(x_{mem,n} - T_{mem,n})^2}{w_{mem,n}^2}\right) \tag{29}$$

The suffix *def*, *inf* and *mem* are referred to defuzzification, inference, and fuzzification layer.

V. RESULT

Simulations of a regulated Cuk converter under the control of the proposed fuzzy logic controller and a PI-ontroller are performed. The converter is operated at 10 kHz and the sampling time of the controller is set to 5kHz. The gains of the PI-controller are tuned by trial-and-error. The responses due to a load change from 10Ω to 5Ω and back to 10Ω are shown in Fig 4 and Fig 5. It can be seen that the response corresponding to the proposed fuzzy logic controller is better than that of the PI-controller.

VI. CONCLUSION

The difficulties behind the problem of controlling switching dc-dc converters, especially the high order converters such as the Cuk converter, have been addressed. Observing these constraints, a neuro-fuzzy control strategy is proposed. On properly designing the linguistic rules and the membership functions of the fuzzy variables, the performance of the regulated converter promises to be better than that using conventional techniques. Further researches should be directed to an in-depth analysis of the fuzzy controller in order to derive a criteria for determining the linguistic rules systematically based on the properties of the switching converter.

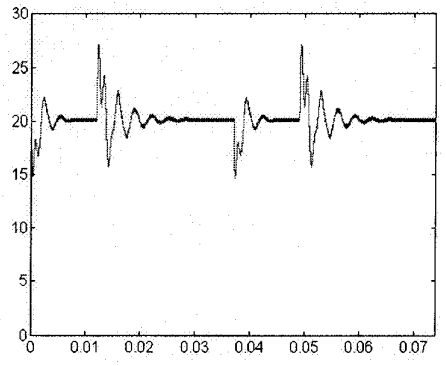

Fig. 4 Step response under fuzzy logic control

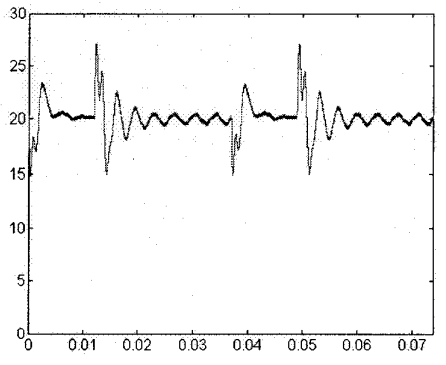

Fig. 5 Step response under PI control

VII. REFERENCES

[1] R. D. Middlebrook and S. Cuk, "A General Unified Approach to Modelling Switching-Converter Power Stages," *IEEE Power Electronics Specialists Conference, 1976 Record*, pp. 18-34, 1976.

[2] S. Cuk and R. D. Middlebrook, "A New Optimum Topology Switching Dc-to-Dc Converter," *IEEE Power Electronics Specialists Conference, 1977 Record*, pp. 160-179, 1977.

[3] P. R. K. Chetty, "Modelling and design of switching regulators," *IEEE trans. on Aerospace Elec. Syst.*, vol. AES-18, no. 3, pp. 333-344, May 1982.

[4] C. T. Lin and C. S. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE trans. on computing*, vol. 40, no. 12, pp. 1320-1336, December 1991.

[5] S. Haykin, *Neural Networks*, Macmillan College Publishing Company, Inc.1994.