Modified non-uniformly distributed error CMAC Algorithm for Regulation purpose

Abhishek Rawat*, M.J. Nigam**
*Department of Electronics and Communication Engineering, Indian Institute of Technology, Roorkee, India
**Department of Electronics and Communication Engineering, Indian Institute of Technology, Roorkee, India

ABSTRACT
In this paper modifications on CMAC algorithm is proposed to improve the steady state response of CMAC based controllers. CMAC based controller shows improvement in rise time with increase in learning rates but at the same time it may lead to steady state oscillations for very large values of learning rates which is undesirable. To make the CMAC based controllers covering large range of learning rates modification is done on memory update rule of CMAC algorithm. It is further enhanced by distributing the error in the activated hypercubes in non-uniform way. The combination of these two modifications gives much improved response than otherwise basic CMAC in terms of peak overshoots, number of oscillations before die out. Simulation results demonstrate the capability of above modifications.

Keywords– Cerebellar model articulation controller, Continuous Variable Learning Rate CMAC

I. INTRODUCTION

The CMAC known as Cerebellar Model Articulation/Arithmetic Controller was proposed by J.Albus in 1975[1-2]. CMAC performs a multivariable function approximation in a generalized look-up table form. Due to its high learning speed and local generalization it is used in variety of applications [3-7].

In control applications CMAC based controllers are sensitive to learning rates. Increase in values of learning rates is accompanied by the decrease in rise time but it leads to the larger value of peak overshoots. Not only has this, for very high value of learning rated steady state response of the system becomes oscillatory. To eradicate the problem of high overshoots variable learning rate schemes are used. Even if the variable learning rate schemes are employed to curb the peak overshoots it might be possible that there are oscillations in the steady state. Thus it is not possible to curb the peak overshoots after certain chosen values of learning rates used in variable learning rate schemes. In order to remove the drawback of steady state oscillations, a slight modification in memory update rule is done. Distributing the error in non-uniform fashion enhances the response of the system. Simulation results shows that Modified CMAC Algorithm covers larger range of learning rates without any steady state oscillations and hence it can be used to curb the peak overshoots to any level.

II. CEREBELLAR MODEL ARTICULATION CONTROLLER

CMAC is a learning structure which emulates the human cerebellum. It’s an associative neural network[8-10] in which a small subset of the network influences any instantaneous output and that subset is determined by the input to the network as shown in Fig. 1. The region of operation of inputs is quantized say $Q$ inputs i.e. the number of elements in a particular input is $Q$. This quantization determines the resolution of the network [11] and the shift positions of the overlapping regions. If $n$ inputs are presented to the network, then total number of elements in input space is $Q^n$ which is quite large. To reduce this memory, inputs presented are converted into hypercubes or hyper rectangles. A particular input vector is the sum of $G$ nearby inputs i.e. a particular input is the overlapping of $G$ nearby inputs. The number $G$ is called the number of layer of CMAC which is referred to as the generalization width of CMAC. Thus $Q^n$ number of memory is converted into $A$ memory units such that $A << Q^n$. The outline of CMAC algorithm is given below:

(i) Number of inputs = $n$
(ii) Number of elements for a particular input = $Q$, which is also the number of quantized state for input.
(iii) Total memory = $Q^n$.
(iv) Number of layers of CMAC = $G$
(v) Number of hypercubes in $i^{th}$ layer = $k_i$, where $i = 1, 2, 3, \ldots, G$

(vi) Total number of hypercube, $A = \sum k_i^n$, which is the memory of CMAC.

If $Q = kG; k \in \mathbb{I}^+$

$A = k^n + (G - 1)(k + 1)^n$

If $Q = kG + r; k, r \in \mathbb{I}^+ \text{ and } r \in [1, G - 1]$

$A = (r - 1)(k + 2)^n + (G - r + 1)(k + 1)^n$

When an input is encountered, $G$ hypercubes are activated and output of the network is simply the sum of the contents of that hypercube. That is if CMAC is an approximator its basis function can be defined as [11]

$$h_i = \begin{cases} 
1, & \text{for activated hypercubes} \\
0, & \text{if not activated} 
\end{cases} \quad (1)$$

The memory content update rule is given by least square mean as [11]

$$A(t) = A(t - 1) + lr.e/G \quad (2)$$

where $lr$ is the learning rate of CMAC and $e$ is the network error.

The Memory contents of CMAC are used to store the weights and it is also called weight space.

### III. MODIFICATIONS IN CMAC ALGORITHM

Feedback error based CMAC are sensitive to learning rates and it is found that increase in learning rate tends to make the response of the system faster. Its variations in peak overshoot with learning rates is studied in [12]. It is also found that increase in the learning rate to larger values may lead to instability. The value of peak overshoots depends on the transfer function used and if there are higher values of overshoots variable learning rate schemes can be used to decrease the values of peak overshoots.

To widen the range of learning rates to improve the rise time without undesirable oscillations in the steady state a slight modification is done on the error distribution in Memory of CMAC. The activated hypercubes are updated according to the rule given by eqn.1 and eqn.2. While updating the activated hypercubes the contents of the remaining hypercubes are cleared which makes sense when the control objective is regulation because the particular area of that memory which represents the steady state should remain active at that time.

Fig.2 shows the modification in memory update rule in which left one is the modified memory update rule while the right one represents the traditional CMAC memory update rule.

In traditional CMAC algorithm error is divided equally into the number of hypercubes activated. But if one actually looks at the hypercubes activated in each layer it will be easy to see that it is better if the error distributed in the hypercubes in each layer should be different as particular hypercube in each layer represents the different range of values of input it will take. Due to this difference in the ranges represented by hypercubes in each layer, the closeness to the actual input represented by these hypercubes will be different. The current modification in CMAC is based on this logic in which the error is distributed in non-uniform fashion as against the traditional CMAC. The basis of distribution of error is the weightage of particular input in the layers representing the activated hypercubes. The logic behind the distribution of error is described below:

As a particular input is quantized $Q=\text{Quantization}$ $G=\text{Generalization}$

Generalization also represents the number of layers of CMAC. Fig.2 and Fig.3 represents the difference in distribution of error.

**Fig.2.** Modified update rule (left) and original update rule (right) of CMAC

**Fig.3.** Uniform distribution of error (below) Non-uniform distribution of error (up)
error is distributed uniformly among the activated hypercubes which is the case of normal CMAC algorithm used. In the Fig.3, hypercubes activated belonging to the particular input is given different weightage according to the distance of particular input from the center value of activated hypercubes. More the distance from the center of hypercubes lesser will be the weightage given to the hypercubes. From the figure let $w_1,w_2,w_3,w_4,w_5$ and $w_6$ be the weight of the activated hypercubes. The error distributed to the activated hypercubes are $ew_1/P, ew_2/P, ew_3/P, ew_4/P, ew_5/P$ and $ew_6/P$ respectively.

where,

$$P = \sum_{i=1}^{6} w_i$$

(3)

The advantage of this setting is the reduction in the learning interference and hence this setting is fruitful for the CMAC’s having larger value of Generalization.

Following are the outlines of modified algorithm

1. Calculate the total number of hypercubes in each layer.
2. Layer 1: Let the number of hypercubes in first layer be $p$.
   Range of values represented by particular hypercube $=[(i-1)G,iG]$ 
   Thus, Center of hypercubes is given by $c_i = G(i - 0.5)$ for $i = 1$ to $p-1$
   Range of values represented by last hypercube $=[(p-1)G,Q]$ 
   Thus, center of last hypercube is given by $c_p = 0.5G(p-1) + 0.5Q$
3. Layer b (b>1):
   Let the number of hypercube in layer b be $m_b$
   a) Range represented by first hypercube $[0,1]$ and hence center is $c_1 = 0.5(b - 1)$
   b) Range represented by other hypercubes $[(k-2G+1k-1G+1]$ and hence center is given by $c_k = (k - 1.5)G + 1$ for $k = 2 to m_b - 1$
   c) Range represented by last hypercube $=[(m_b-2G+1Q]$ and hence center is given by $c_{m_b} = 0.5(Q + (m_b - 2)G + 1)$

Now one has to vary the values such that the value at the center has the highest value and there are so many ways to do that. For simulation purpose Gaussian function is used in which the center of it is the centers mentioned above and the standard deviation is the product of real number and the width of the hypercubes.

Thus Memory Update rule is given by

$$A_i(t) = A_i(t - 1) + lr . e . w_i/P$$

(4)

where $i = 1 to G$

$$w_i = e^{-0.5(x_i-c_i)/s^2}$$

(5)

$$x_i = \frac{Q(ip - minip)}{(maxip - minip)}$$

(6)

$s$ being the standard deviation of Gaussian function used.

$ip$ = input received by CMAC

$minip$ = minimum value CMAC can take

$maxip$ = maximum value CMAC can take

IV. CONTROL SCHEME

To demonstrate the behavior of variable memory simulations are done on unstable second order plant in which CMAC is used in conjunction with feedback control law to control the plant. Though CMAC has the capability to learn nonlinear functions quickly, linear plant is discussed here for simplicity. The plant can be written in state space form as

$$\begin{cases}
  x_1 = x_2 \\
  x_2 = ax_1 + bx_2 + cu
\end{cases}$$

(7)

It can also be written as

$$\begin{cases}
  \dot{x}_1 = x_2 \\
  \dot{x}_2 = f(x_1, x_2) + cu
\end{cases}$$

(8)

Control law for the state space equation given by (8) can be written as,

$$u = \left[-k_1(x_1 - y_r) - k_2x_2 - \hat{f}\right]/c$$

(9)

where $y_r$ is the input signal, $\hat{f}$ being the function to be estimated online, $k_1$ and $k_2$ is such that the polynomial $s^2 + k_2s + k_1$ is Hurwitz. The Simulink diagram to implement the CMAC is shown in Fig. 4.

V. SIMULATION AND RESULTS

For simulation purpose, Magnetic Levitation plant which is an unstable plant with transfer function as given by eqn. 10 is used,

$$G(s) = \frac{2500}{s^2 - 980}$$

(10)

This plant can be written in state space form

$$\begin{cases}
  \dot{x}_1 = x_2 \\
  \dot{x}_2 = 980x_1 + 2500u
\end{cases}$$

(11)

www.ijsret.org
CMAC parameters are
Q=100
G=20
Total Memory of CMAC=709
Learning rate is varied for different simulation results.
For non-uniform distribution of error
Standard deviation is taken to be 0.425 times the width of activated hypercubes.

From Fig.5, Fig.6 and Table 1 it is quite clear that modified CMAC has an improved response in terms of peak overshoot and there are no steady state oscillations in spite of higher learning rate of 1000. The steady state oscillations of basic CMAC is shown in Fig.6 in which instead of settling at zero error it is switching between two values and these steady state oscillations may lead to instability.

Standard CMAC cannot handle the higher learning rates instead modified CMAC have the capability to handle the higher learning rates. So to derive the benefits of higher learning rate this modification is fruitful. As shown in the Fig.7 and Table 2 peak overshoots also decreases with increase in learning rates although higher values of learning rates results in oscillations which though dies out soon.

So far from Fig.5, Fig.6 and Fig.7 it is concluded that modified CMAC Algorithm is superior than that of traditional ones. But the story doesn’t ends here. Non-uniform distribution of error in CMAC further enhances the improvement in the response. From Fig.8 and Fig.9 for learning rates of 7000 and 10000, the number of oscillations increases which is reasonable but non-uniformly distributed error CMAC is more resistant to oscillations than that of uniformly distributed error CMAC although the peak overshoots of both these CMAC are identical.

Modified CMAC Algorithm with non-uniform distribution of error is applied on Continuous Variable Learning (CVL) rate scheme. Fig.10 and Fig.11 shows...

**TABLE 1:** Comparison of peak overshoot for normal Memory update rule and modified memory update rule

<table>
<thead>
<tr>
<th>Learning rate</th>
<th>Peak Overshoots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal CMAC</td>
<td>18.99</td>
</tr>
<tr>
<td>Modified CMAC</td>
<td>13.83</td>
</tr>
</tbody>
</table>

**TABLE II:** Comparison of Modified memory update rule for different learning rates

<table>
<thead>
<tr>
<th>Learning rate</th>
<th>Rise Time</th>
<th>Overshoot</th>
<th>Settling Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.0432</td>
<td>13.67</td>
<td>0.1819</td>
</tr>
<tr>
<td>2000</td>
<td>0.0384</td>
<td>12.38</td>
<td>0.1292</td>
</tr>
<tr>
<td>3000</td>
<td>0.0351</td>
<td>11.61</td>
<td>0.1079</td>
</tr>
<tr>
<td>4000</td>
<td>0.0323</td>
<td>11.18</td>
<td>0.1876</td>
</tr>
</tbody>
</table>

Fig.6. Steady State oscillations in Normal CMAC due to high learning rates

Fig.7. Comparison for different learning rates
the response with uniform and non-uniform distribution of error. Overshoots are same but improvement in the undershoots can be seen. Due to this improvement one can use higher values of learning rates in lr4 region to curb the peak overshoots without large value of undershoots.

Fig. 12 uses the CVL CMAC for learning rate of 20000 in lr4 region. It can be easily seen that in uniformly distributed error response have higher value of undershoot than that of non-uniformly distributed error response. It shows that non-uniformly distributed error CMAC are more resistant to undershoots than that of uniformly distributed error CMAC. Due to this tolerance to higher learning rates in lr4 region it can also be inferred that non-uniformly distributed error CMAC are more robust than that of normal ones.

**TABLE III:** Uniform Distribution of error with CVL Scheme for normal memory update rule and modified memory update rule (lr1=5, lr2=5, lr3=50, lr4=5000)

<table>
<thead>
<tr>
<th></th>
<th>Peak Overshoot</th>
<th>Undershoot</th>
<th>ISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVL1</td>
<td>8.81</td>
<td>7.17</td>
<td>0.01555</td>
</tr>
<tr>
<td>CVL2</td>
<td>8.81</td>
<td>4.54</td>
<td>0.01529</td>
</tr>
</tbody>
</table>

**Fig. 9. Response for learning rate=10000 (below one is amplified version of above)**

**Fig. 10. Responses of CVL CMAC and new CVL CMAC with modified memory update rule with uniform distribution of error**

**Fig. 11. Responses of CVL CMAC and new CVL CMAC with modified memory update rule with non-uniform distribution of error**
TABLE IV: Non-uniform Distribution of error with CVL Scheme for normal memory update rule and modified memory update rule
(lr1=5, lr2=5, lr3=50, lr4=5000)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Peak Overshoot</th>
<th>Undershoot</th>
<th>ISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVL1</td>
<td>8.74</td>
<td>5.97</td>
<td>0.01536</td>
</tr>
<tr>
<td>CVL2</td>
<td>8.74</td>
<td>1.13</td>
<td>0.01514</td>
</tr>
</tbody>
</table>

Fig.12 Responses of CVL CMAC and new CVL CMAC with modified memory update rule with non-uniform distribution of error with lr4=20000

TABLE V: Non-uniform Distribution of error with CVL Scheme for normal memory update rule and modified memory update rule (lr1=5, lr2=5, lr3=50, lr4=20000)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Peak Overshoot</th>
<th>Undershoot</th>
<th>ISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVL1</td>
<td>3.74</td>
<td>9.462</td>
<td>0.01548</td>
</tr>
<tr>
<td>CVL2</td>
<td>3.74</td>
<td>1.89</td>
<td>0.01498</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

The rise time of feedback error based CMAC are sensitive to learning rates in which it is directly proportional to the learning rates. However traditional CMAC suffered from oscillations in steady state or unstable for very high value of learning rates. Variable learning methods can be used to improve the response by exploiting different learning rates according to the error. However due to the limitations of using lower range of learning rates one cannot improve the response in terms of peak overshoots while maintaining reasonable settling time to certain level. To widen the range of learning rates slight modification in memory update rule is proposed which shows significant improvement in response as shown by Fig.5, Fig.6 and Fig.7 respectively. This modification is successfully able to curb the steady state oscillations. To further enhance the response of the system error is distributed non-uniformly in the activated hypercubes. Fig.8 and Fig.9 shows that this arrangement is resistant to oscillations although peak overshoots remains the same. The combination of these two modifications is applied on Continuous Variable Learning rate (CVL) scheme which shows significant improvement in the value of undershoots and hence facilitates larger value of learning rate in lr4 region to curb the peak overshoots efficiently without having larger values of undershoots.

REFERENCES