A New Stability Analysis of a CMAC-Based Adaptive Control System

August 23, 1995

The CMAC adaptive control element is envisaged as being used as a feed-forward component 'piggy-backed' with an existing continuous time conventional feedback controller, as shown below, in what Kawato terms 'Feedback Error Learning'. The CMAC is by its nature a digital element, so

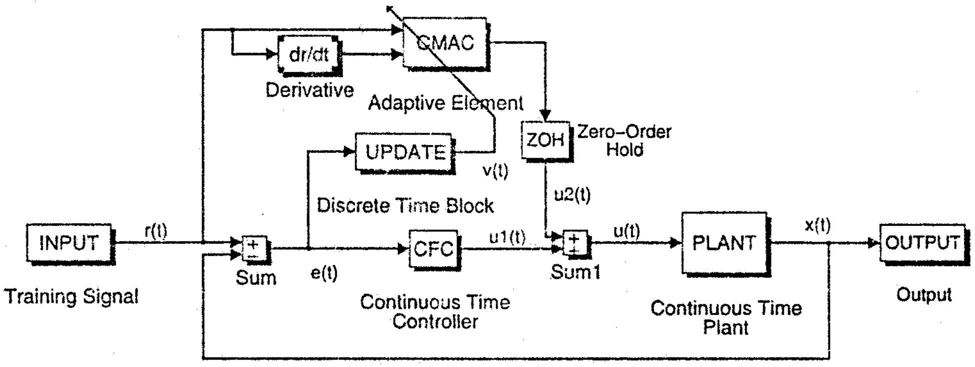

Figure 1: *CMAC adaptive control element used in conjunction with a conventional feedback controller.*

to facilitate easier analysis, the conventional feedback controller (CFC) will be approximated as a digital element followed by a zero-order-hold (ZOH), running at the same clock speed as the CMAC. The continuous time plant preceded by the ZOH can now be exactly represented as a digital plant. For the time being, linear time-invariant single-input single-output systems will be examined. Non-linear systems will subsequently be discussed briefly. The analysis will be carried out in three stages of increasing complexity :

- CMAC without generalization, no CFC.
- CMAC without generalization, with CFC.
- CMAC with generalization, with CFC.

The training of the CMAC is assumed to be via a periodic input signal (period p discrete time steps), which the system output will match when training is complete.

1 CMAC without Generalization, no CFC

The configuration of the system is now as shown below. The discrete transfer

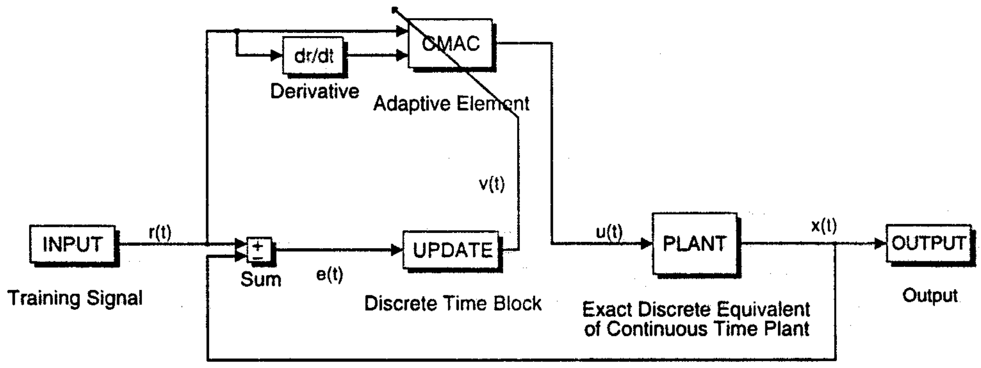

Figure 2: *Previous system with the conventional feedback controller removed for easier analysis.*

function of the plant is represented as

$$\frac{x(z)}{u(z)} = \frac{PN(z)}{PD(z)}$$

Without generalization, the CMAC behaves as a look-up table. For every point in the discretized phase space, defined by the training signal inputs to the CMAC, there is a memory location with a stored value, representing the control value to be output whenever the training signal trajectory passes through this point in phase space. This corresponding memory will hereafter be referred to as the weight space. The values stored in the weight space locations change during training via addition of the update signal. Because the training signal is assumed to be periodic (period p time steps) the same point in phase space is passed through (and thus the same memory location is used to output the control value) every p time steps. The iterative behaviour of the CMAC block can thus be described as

$$u(k) = u(k-p) + v(k-p+d)$$

where the CMAC memory used to generate the $(k-p)$'th control value is updated with the update value from d time steps later, for reasons that will become clear in a moment. The transfer function of the CMAC element is then

$$\frac{u(z)}{v(z)} = \frac{AN(z)}{AD(z)} = \frac{z^d}{z^p - 1}$$

Let the transfer function of the update element be represented by

$$\frac{v(z)}{e(z)} = \frac{\phi\, UN(z)}{UD(z)}$$

where ϕ is a gain commonly referred to in the literature as the *learning rate.* The overall closed loop transfer function of the system is now

$$\frac{x(z)}{r(z)} = \frac{\phi\, PN(z)\, UN(z)\, z^d}{(z^p - 1)\, PD(z)\, UD(z) + \phi\, PN(z)\, UN(z)\, z^d}$$

Given that ϕ, $UN(z)$, $UD(z)$, and d can be chosen by the designer, the question is how to ensure stability of the above transfer function, in which the period p could be quite large. The poles are defined by the denominator, which is in the form

$$(z^p - 1)\, Q(z) + \phi\, R(z)$$

Obviously if $\phi\, R(z) = Q(z)$ then the denominator reduces to $z^p\, Q(z)$. This is equivalent to setting $UN(z) = PD(z)$, and $UD(z) = PN(z)\, z^d$. Note that the update transfer function $UN(z)/UD(z)$ must be a proper one, ie. the denominator must be of equal or greater order than the numerator, and because it is likely that the order of $PD(z)$ is greater than that of $PN(z)$, the 'update delay' d must be chosen accordingly. This is what could be referred to as the 'ideal update', because the update transfer function is essentially the inverse of the plant transfer function. However this requires exact knowledge of the plant, and in any case one might ask what is gained over just inserting a straight feed-forward element which is the inverse of the plant ? The answer is that if the plant characteristics changed, then assuming stability were maintained, the CMAC would still retrain itself, whereas the straight feed-forward element would not.

The behaviour of the roots of the denominator of the overall transfer function can be examined in more detail by noting that it is already in the correct form to be plotted as a root locus, viz. $(z^p - 1)\,Q(z) + \phi\, R(z) = 0$, where we regard $(z^p - 1)\,Q(z)$ as the denominator (roots of which are represented below as $\times$'s), and $\phi\, R(z)$ as the numerator (roots of which are represented below as o's). Assume all roots of $Q(z)$ are already inside the unit circle. Now it is necessary to ensure that all the roots of $z^p - 1$, sitting on the unit circle, move inside it as ϕ is increased from zero. The best way to ensure this is to place the 'zeros' of $R(z)$ exactly over the 'poles' of $Q(z)$, so that the following root locus results. This is exactly what was done with

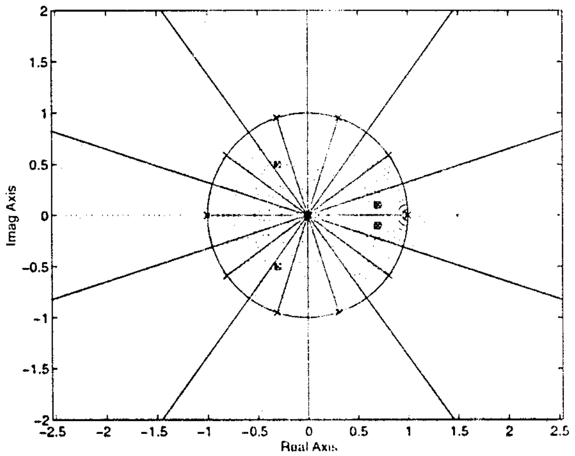

Figure 3: *Root locus of the poles of the overall transfer function as the learning rate ϕ is increased from zero (no generalization, no CFC). For this example, $Q(z)$ has all its roots inside the unit circle, and p was chosen as 10 for easy visualization of results. Here roots of $R(z)$ exactly match those of $Q(z)$. All poles initially move inside the unit circle as ϕ is increased.*

the 'ideal update', as seen by the fact that all the 'poles' of $z^p - 1$ converge at the origin when $\phi = 1$ (if the leading coefficients of $Q(z)$ and $R(z)$ are the same). But what if the plant transfer function is not known exactly? The situation where $Q(z)$ is not 'canceled' exactly by $R(z)$ is shown in the next figure. As long as $Q(z)$ and $\phi R(z)$ approximately match, then the system will be stable for a range of ϕ values. There is one important point to note,

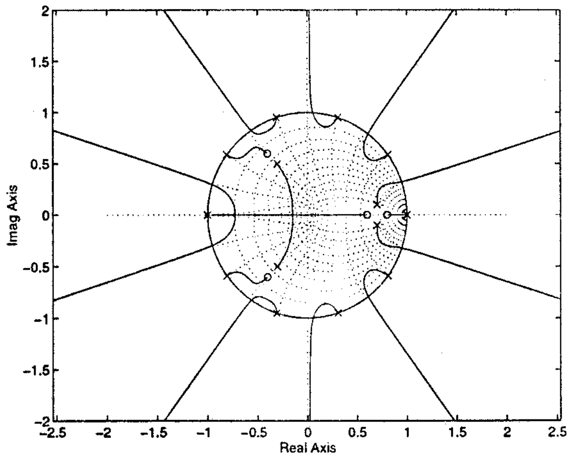

Figure 4: *Same example as previously, but where roots of $R(z)$ do not match those of $Q(z)$ exactly, but $R(z)$ and $Q(z)$ are of the same order. Again, all poles initially move inside the unit circle.*

however. The order of $Q(z)$ and $R(z)$ must be *exactly the same*, or some of the $z^p - 1$ poles will immediately move away from the unit circle as ϕ is increased, rather than first moving inside it and then out. One consequence of this is that if there is some transport delay in the plant, represented by the transfer function

$$\frac{x(z)}{u(z)} = \frac{PN(z)}{z^b PD(z)}$$

then the 'update delay' d must be chosen to take this into account *exactly*. The following figures show this effect.

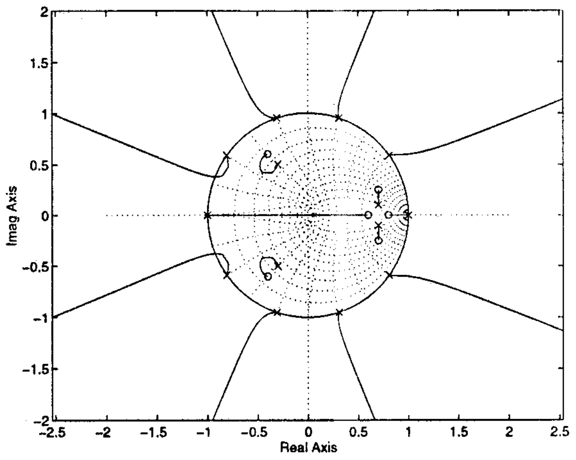

Figure 5: *Same example, where $R(z)$ is of higher order than $Q(z)$. Some poles immediately move outside the unit circle as ϕ is increased.*

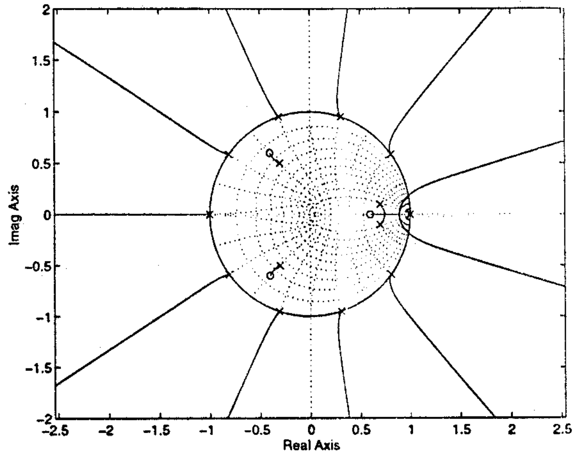

Figure 6: *Same example, where $R(z)$ is of lower order than $Q(z)$. Some poles immediately move outside the unit circle as ϕ is increased.*

2 CMAC without Generalization, with CFC

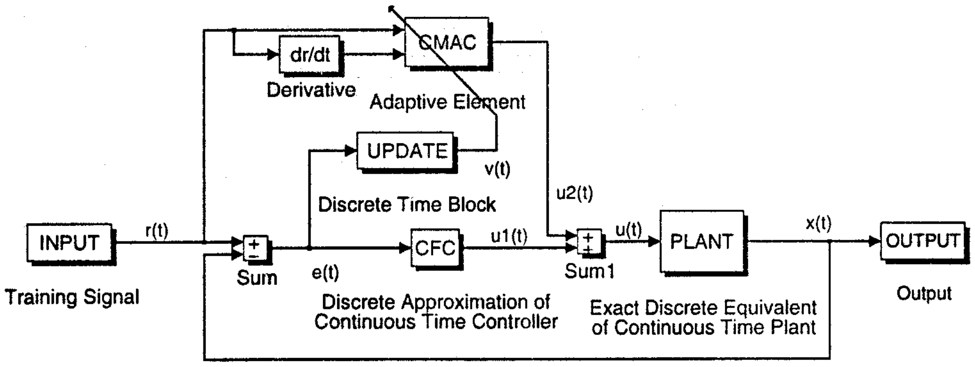

Figure 7: *Continuous time CFC approximated by a discrete time CFC followed by a ZOH. This ZOH is then lumped with the continuous time plant to create an exact discrete time equivalent plant.*

The transfer function of the CFC element is represented as

$$\frac{u_1(z)}{e(z)} = \frac{CN(z)}{CD(z)}$$

and the overall transfer function of the system is now

$$\frac{x(z)}{r(z)} = \frac{(z^p - 1)\,PN(z)\,CN(z)\,UD(z) + \phi\, z^d\, PN(z)\,CD(z)\,UN(z)}{(z^p - 1)\,[PD(z)\,CD(z) + PN(z)\,CN(z)]\,UD(z) + \phi\, z^d\, PN(z)\,CD(z)\,UN(z)}$$

the denominator of which is still in the form

$$(z^p - 1)\,Q(z) + \phi\, R(z)$$

The effect of the CFC has been to modify the $Q(z)$ and $R(z)$ polynomials, so that the 'ideal update' is now

$$\frac{\phi\, UN(z)}{UD(z)} = \frac{PD(z)\,CD(z) + PN(z)\,CN(z)}{z^d\, PN(z)\,CD(z)}$$

where again d must be chosen so that $Q(z)$ and $R(z)$ are exactly the same order. The following is a simulation example which shows that the CMAC does indeed behave as described above, when there is no generalization. The plant and CFC transfer functions were chosen to be

$$\frac{PN(z)}{PD(z)} = \frac{1}{z(z-0.8)}$$
$$\frac{CN(z)}{CD(z)} = 0.5\,\frac{z-0.7}{z}$$

Each of the directions in the the CMAC phase space, $r(t)$,$\dot{r}(t)$ was discretized into 101 levels. The training signal was $r(t) = \sin \frac{2pi}{5} t$ with a time step of 0.025 seconds, giving a periodicity $p = 200$. In accordance with the above principles, the variable parameters were chosen to be

Update transfer function : $\frac{UN(z)}{UD(z)} = \frac{z-0.6}{z}$
Learning rate : $\phi = 0.5$
Update delay : $d = 2$

The following behaviour resulted. The update delay was then changed to

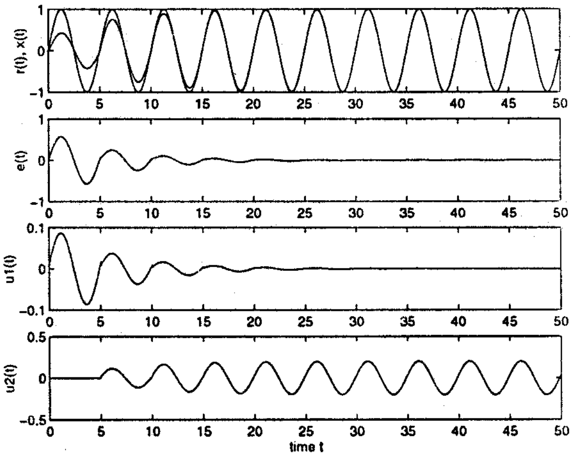

Figure 8: *CMAC without generalization, update delay chosen correctly.*

be $d = 1$, with the results

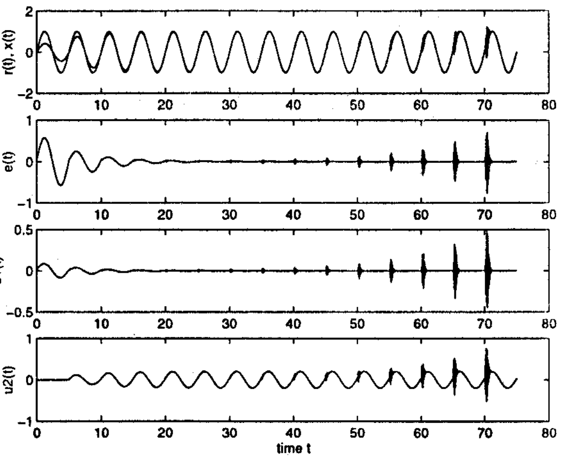

Figure 9: *CMAC without generalization, update delay chosen incorrectly.*

3 CMAC with Generalization, with CFC

With generalization, the CMAC no longer behaves exactly as a lookup table. The amount that each succeeding weight template overlaps the previous ones, and where it overlaps, depends very much on the training signal. As an analysis tool, imagine the following simplified behaviour. Assume that each weight template always overlaps the previous one by the same fixed amount, in the manner shown in the next figure. For the example depicted in the figure, with five weights used at each calculation of the control value, an overlap m of three weights with the previous time step, and and 'update delay' d of one time step, the transfer function of the CMAC becomes

$$\begin{aligned} u_2(k) &= \frac{3}{5}\,u_2(k-1) + \frac{1}{5}\,u_2(k-p+2) + \frac{1}{5}\,u2(k-p+1) \\ &+ \frac{3}{5}\,v(k) + \frac{1}{5}\,v(k-p+3) + \frac{1}{5}\,v(k-p+2) \end{aligned}$$

$$\frac{u_2(z)}{v(z)} = \frac{AN(z)}{AD(z)} = \phi\,\frac{\frac{3}{5}\,z^p + \frac{1}{5}\,z^3 + \frac{1}{5}\,z^2}{z^p - \frac{3}{5}\,z^{p-1} - \frac{1}{5}\,z^2 - \frac{1}{5}\,z}$$

Now the denominator of the overall transfer function becomes $(z^p - \frac{3}{5}\,z^{p-1} - \frac{1}{5}\,z^2 - \frac{1}{5}\,z)\,Q(z)$ instead of $(z^p - 1)\,Q(z)$, and the following figure, plotting

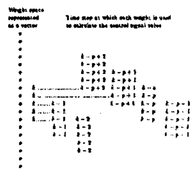
Weight space represented as a vector
Time step at which each weight is used to calculate the control signal value

Figure 10: *Simplified CMAC behaviour. Weights used at each time step to calculate the control value.*

the poles and zeros of the CMAC transfer function, shows that the poles of this new polynomial, rather than sitting on the unit circle, are all on or inside the unit circle. The net effect of this change is to make the system much easier to stabilize. The reader may verify that the general form of the CMAC transfer function with the preceeding simplification in behaviour is

$$\begin{aligned} u_2(k) &= \frac{a}{g}u_2(k-j_a) + \frac{b}{g}u_2(k-j_b) + \frac{c}{g}u2(k-j_c) \\ &+ \frac{a}{g}v(k-j_a+d) + \frac{b}{g}v(k-j_b+d) + \frac{c}{g}v(k-j_c+d) \end{aligned}$$

with the indices and coefficients

$$\begin{aligned} j_a &= 1 \text{ if } m > 0 \\ j_b &= p - n_b,\ n_b \text{ the largest such that } n_b(g-m) < g, \text{ if } g - m > 0 \\ j_c &= p - n_c,\ n_c \text{ the largest such that } n_c(g-m) <= m, \text{ if } g - m > 0 \end{aligned}$$

$$a = \begin{cases} m/g \text{ if } d = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$b = \begin{cases} 1/g \max([\min\{(p-j_b+1)(g-m), g\} - \max\{(p-j_b)(g-m), m\}], 0) \text{ if } j_b - d >= 0 \\ 0 \text{ otherwise} \end{cases}$$

$$c = \text{as for } b \text{ (but do not duplicate } b \text{ if } j_c = j_b)$$

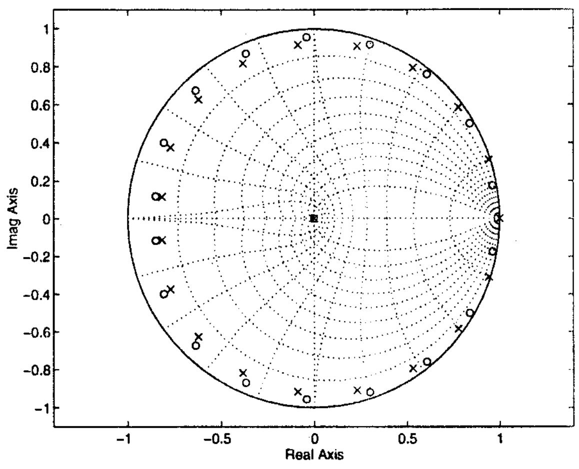

Figure 11: *Poles and zeros of the CMAC transfer function, generalization* $g = 5$, *overlap* $m = 3$, *update delay* $d = 1$, *period* $p = 20$

How closely does this simplified behaviour represent the actual behaviour of the CMAC? In practice, because the weights chosen as training proceeds are so dependent on the training signal, the overlap varies. It becomes possible that even with generalization, there are periods in the training cycle where there is no overlap, and this situation is exactly equivalent to no generalization. Thus it becomes necessary to design the update block and the learning rate such that the system is stable for any overlap. However the no-generalization case, because the poles of the CMAC transfer funtion are initially on the unit circle rather than inside it, is the most restrictive case.

The following figure shows the same simulation depicted previously, with a generalization $g = 5$ selected.

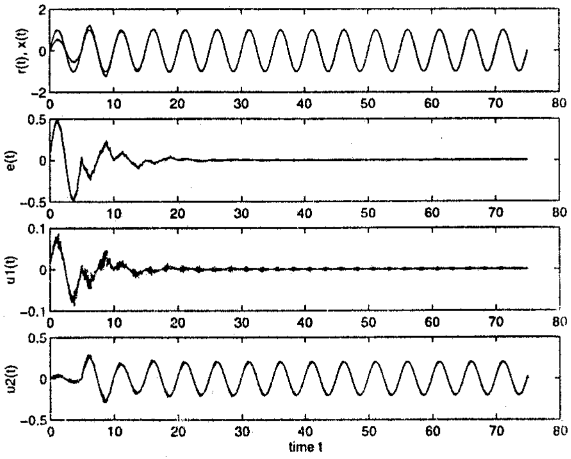

Figure 12: *CMAC with generalization $g = 5$, update delay chosen correctly based on the no-generalization case.*