INCREMENTAL MODEL REFERENCE ADAPTIVE POLYNOMIAL CONTROLLERS NETWORK [1]

Eric Ronco and Peter J. Gawthrop

Technical Report CSC-96013
Centre for System and Control
University of Glasgow
ericr, peterg @mech.gla.ac.uk

Abstract. In this paper we are describing and illustrating the Incremental Model Reference Adaptive Polynomial Controllers Network (IMRAPCN). This algorithm is a polynomial version of the conventional linear controllers network. Two important properties of that system are: (1) its behaviour is clearly understandable because each polynomial controller can be interpreted in linear terms and (2) it is capable of an autonomous construction of its structure which involves the clustering of the validity space. We illustrate its control capability according to the control of two highly non linear systems. The results show that the IMRAPCN enables a very accurate control of the non linear systems over a wide operating range. This system can be used for the control of a possibly discontinuous non linear system and it is not affected by the "stability-plasticity dilemma". The use of polynomial controllers network enables a much more accurate control of a non linear system than the one that can be obtained by using linear controllers network. Finally, a polynomial controllers network of low order (e.g. cubic polynomial) is much more flexible than a single polynomial controller. This flexibility could make the polynomial controllers network adaptable to any kind of system. From these features the IMRAPCN appears very promising as a powerful and general algorithm for the control of non linear systems.

Keywords. Incremental controllers network; Adaptive control; Polynomial controller; Model reference adaptive controller; Polynomial system identification

Introduction

The control of a non linear system is often achieved through the use of a single linear controller; the system is linearised around an equilibrium point and the resulting controller is valid only for a local region of the system. For example figure 1 depicts the non linearity of a system according to a range of state values (e.g. angle values). The linearisation of the system (plotted in dash line in fig. 1) shows that the controller will only be locally valid because the linearisation starts to diverge from the non linear behaviour of the system when the angle is more than 20 degrees. Thus, we can assume that beyond the region of validity of the controller the performance of the controller will be poor. This is a serious problem when the system is highly non linear. One standard way to overcome the problem is to adapt continually the identification (i.e. the linearisation) of the system and thus the controller; this is conventional adaptive control.

[1] Note that this article is submitted to the IJC

Such a method can only be effective if the dynamic of the system is changing smoothly and quite slowly through time. Therefore, if the function is discontinuous adaptive control can not be applied. In addition the slowness of such an adaptation may result in a large transient error (Narendra *et al.*, 1995). Perhaps a more serious problem is related to "the basic design problem" for learning machine emphasised by (Carpenter and Grossberg, 1988): the "stability-plasticity dilemma": while the controller is adapting to an operating region of the system it is forgetting previous adaptations concerning other regions.

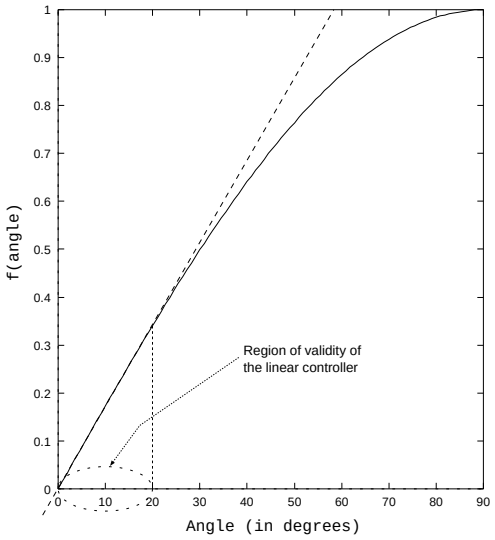

Fig. 1. Local validity of a single linear controller. The function $f(angle)$ is depicted by a plain line and its linear approximation is depicted by a dotted line.

A simple way to have a control system valid for the entire system is to use a certain number of controllers each one locally valid for a different operating region of the system. A clustering of the state space of the system is used in this study in order to select the valid controller concerning a current operating point. In figure 2 six local linearisations of the system are shown. Each one is roughly valid for a different region of the state space (i.e. the validity space) of the system. From these local linearisations of the system arise an accurate non linear model of the system. Hence, if we design a controller for each local identification we are very likely to obtain a multi linear controllers system adapted for the control of the entire non linear system. Following Johansen and Foss (Johansen and Foss, 1992) and Murray-Smith and Hunt (Murray-Smith and Hunt, 1995), we call such a multi controllers system a "Controllers Network" (CN). This controller will not suffer from the stability-plasticity dilemma since different controllers will be adapted for different regions of the system. Hence, and as highlighted by (Narendra *et al.*, 1995), this CN can be adapted for different discontinuities of the system. Another advantage of such a non linear controller is that its behaviour is easily understandable because each controller is linear. The only requirement to apply that algorithm is to know the relative order of the system.

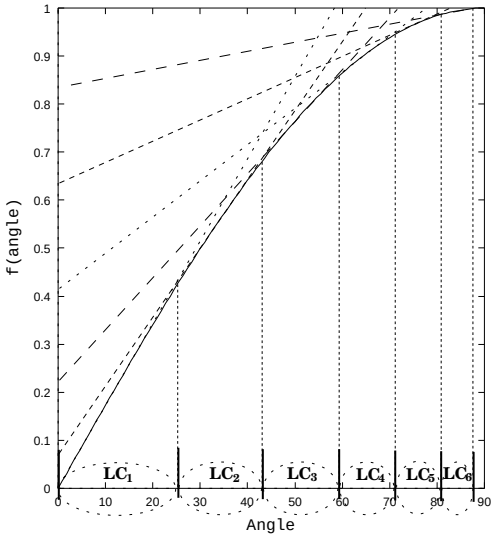

Fig. 2. Entire validity of a controllers network. The function $f(angle)$ is depicted by a plain line and 6 local linear approximations of this function are depicted by different dash lines.

However, a very high (possibly infinite) number of linear controllers can be required to control accurately a highly non linear system. This amount of linear controllers can be very significantly reduced by using polynomial controllers even of as low order as cubic polynomial. This reduction of the amount of controllers will lead to a much more accurate identification and control of the system than the one obtained by using linear functions. Moreover, although polynomial functions are non linear we can apply to them most of the tools developed in linear theory. An important advantage is the possibility to apply a linear regression method (e.g. least squares method) to identify the parameters of a polynomial function. Therefore, using polynomial controllers (instead of using linear controllers) in controllers network, might result in a powerful algorithm for the control of non linear systems. Our purpose is to describe such a polynomial controllers network. This algorithm is the polynomial version of the Incremental Linear Controllers Network we have recently developed (see (Ronco and Gawthrop, 1996) or (Ronco and Gawthrop, 1997*b*)).

In the next section we briefly describe the Incremental Controllers Network (ICN). This system consists of a controllers Network and a Progressive Control Design (PCD) algorithm. The controllers network is the clustering method used to select the valid controller each

time. The PCD is used to construct incrementally the controllers network. The core of our study concerns the description of the Model Reference Adaptive Polynomial Controller (MRAPC) which is the polynomial version of the Model Reference Adaptive Controller (MRAC). The MRAPC is the building block of the ICN. Hence the ICN and the MRAPC together lead to the Incremental Model Reference Adaptive Polynomial Controller Network (IMRAPCN). We will illustrate the control capability of IMRAPCNs of different polynomial order compared to single MRAC and single MRAPC. We will conclude this study by discussing the advantages of this method and our future work in this area.

Incremental Controllers network

Controllers Network

In this section we present a system capable of activating each time the valid controller. Different solutions are possible. For simplicity, we propose here to use a *spatial clustering* approach. As we discuss in (Ronco *et al.*, 1996*a*) and (Ronco and Gawthrop, 1997*a*), this approach is most effective when the clustering space is single dimensional and when only one system variable leads to a non linear behaviour of the system. These are very restrictive features. We use that method here for simplicity and because we will consider in this paper only first order systems. In this case the clustering can be achieved on the single dimension axis shaped by the system output at time $t-1$ y_{t-1}. However, it is vital to generalise our method for systems of any order and thus it will be necessary to use different clustering criteria. In (Narendra *et al.*, 1995) the authors propose to select the controller according to the best current model of the system. The controller linked to that model is the one selected. This can obviously be a valid solution only if there is a mapping from model to controller which preserves the notion of "best". In (Gawthrop and Ronco, 1996) we propose an alternative: to use the a posteriori control error to select the valid controller. This control error is determined according to previous values of that error.

The clustering of a single dimensional Euclidean space can be achieved using a network of very simple basis functions each one corresponding to a segment. You can see in detail the components of such a basis function in figure 3. Each basis function is characterised by a centre and a "single dimension width". In (1) you can see that to determine the activity of a basis function, first the absolute Euclidean distance ('distance 1') between the centre of the basis function and the input pattern 'X' is computed. The distance ('distance 2') between the boundary of the basis function and the pattern can then be determined by subtracting the width of the basis function to the 'distance 1'. The basis function the most activated will correspond to the one having the smallest 'distance 2' between the basis function (boundary) and the pattern.

$$Distance2 = ||centre - X|| - width \qquad (1)$$

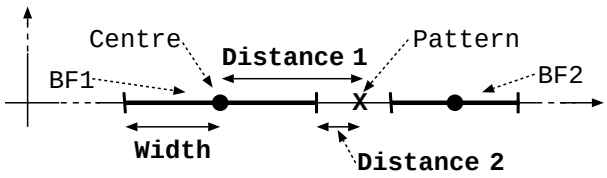

Fig. 3. Description of the components of the basis functions used in that paper for the clustering of a single dimension space. Two basis basis functions are depicted. Each one is composed of a center and a width.

Now, by connecting one controller to each basis function we obtain a network of controllers. In this study we chose to select one controller each time using a "winner takes all" method. For each control error at time t-1, the controller connected to the basis function the most activated is selected . Hence, the output of the controllers network (CN) corresponds to the output of the selected controller.

Note that the interpolation between controllers outputs is very smooth because the use of a polynomial function even of very low order (e.g. cubic polynomial) enables a very accurate local identification and control of the system. Hence it is not necessary to perform a weighted sum of the controllers' output to obtain a smooth interpolation between controllers, as did originally (Johansen and Foss, 1992) when developing the local model network.

Let us now sketch the behaviour of the CN feedback control system (see fig. 4). Note that by controller we are not referring to the MRAC in particular because we believe that other kinds of controllers (e.g. General Predictive Controller) could be used as the building block of the CN. At the first stage of the CN feedback control, the initial value of the system output y_{t-1} is supplied to the CN. We select the controller having its basis function the most activated by the opearating point y_{t-1}. This controller inputs a control signal u_{t-1} to the system according to the control error e. From the new operating point y_{t-1} the valid controller is selected. Thus, a shift between controllers is achieved while the control error is reduced towards zero.

Under some circumstances (as discussed by(Narendra *et al.*, 1995) and (Ronco and Gawthrop, 1997*a*)), it may be

more appropriate to base selection on a weighted average of past values of y; but this is not pursued further here.

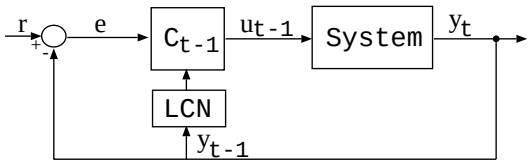

Fig. 4. Block diagram of the controllers network closed loop system.

Progressive control design

It is vital to find out the required number of controllers and their region of validity in order to efficiently apply a controller network (see for further details (Ronco and Gawthrop, 1996) (Ronco and Gawthrop, 1997*b*)). We can find different attempts to solve that clustering problem in the literature (see for details pp43-46 in (Murray-Smith, 1994)). We believe that the incremental approach is the more appropriate. We are here proposing to apply a "progressive control design" (PCD). The underlying idea of this algorithm is to build on line the controllers network according to the current control error. A sketch of the PCD algorithm is presented next (see for further detail about the PCD (Ronco and Gawthrop, 1996) (Ronco and Gawthrop, 1997*b*)).

[1] Initialisation of the system
[2] Selection of the valid controller according to y_{t-1} (i.e. the operating point).
[3] Activation of the system using the control input U determined by the selected controller.
[4] Determination of the error between the actual output of the system and the desired output.
[5] If that error is significant
  (a) The basis function connected to the selected controller is updated in order it covers the current operating point and previous ones for which that controller was valid.
  (b) The parameters of the controller are updated in order to fit the upgraded operating region.
[6] If the error is significant
  (a) Addition of a new controller to the controllers network
  (b) Centreing of the basis function connected to that new controller at the current operating point
  (c) Design of the controller according to the local identification of the system around the current operating point
  (d) Back to stage [1]
[7] Back to stage [2] (while the system has not reached the desired value (i.e. the goal)).

At the end of the PCD the CN will be structured with the required number of controllers concerning the control problem and the region of validity of each controller will be adequate. So, using the CN with the PCD we end up with an Incremental Controllers Network (ICN) capable of an autonomous construction of its own structure and leading to a control adaptation of an entire non linear system.

Model Reference Adaptive Polynomial Controller

Local Polynomial System Identification

Most of the controllers are designed more or less according to the identification of the system. To keep the advantages of using a linear controller (and thus linear theory) a linear identification of the system must be performed. Although polynomial functions are non linear they belong to the class of function underlying a linear combination of M specified functions of x. The general form of a polynomial function is a function p defined for all numbers x by

$$p(x) = a_N x^N + a_{N-1} x^{N-1} + ... + a_1 x + a_0 \quad (2)$$

where N is a non negative integer and $a_0, a_1, ..., a_N$ are fixed real numbers (i.e. the parameters). If $a_N \neq 0$ then $p(x)$ has degree N. We can see that $p(x)$ is simply a sum of non linear functions of x. Thus this function can be rewritten as

$$p(x) = \sum_{j=N}^{0} a_j x^j \quad (3)$$

This characteristic enables any linear regression method (e.g. least square) to be applied in order to identify the parameters of the polynomial function. In fact, most of the linear theory holds for this special case of linear function. Thus, a polynomial function has the advantage of representing non linearly a function without the drawback of other nonlinear functions.

To perform a polynomial identification of a MISO system it is necessary and sufficient to have at one's disposition $(N+1)M$ input-output data. M relates here the number of input variables of the function to identify. The general polynomial equation of a MISO system can therefore be rewritten as

$$p(x) = \sum_{i=1}^{M} \sum_{j=N}^{0} a_{ij} x^j \quad (4)$$

In a control point of view, if we consider only SISO dynamics systems, M equals $O * 2$ the order of the system O time the two system variables (i.e the control input U and the first order system state y). Thus the polynomial equation of a SISO open loop control system could be described by

$$y_k = \sum_{i=1}^{M} \sum_{j=N}^{0} b_i u_{k-i}^j + a_{ij} y_{k-i}^j \tag{5}$$

where k is the discrete time index. The problem of that equation is that it makes the design of the controller rather complex since the control input U is polynomial. To avoid that complication it is much more reasonable to assume that the control input has a linear effect on the system. Although the local identification will be simplified, we will end up with much accurate local identifications than the ones obtained by locally linearising the system. In addition, when the control input U will have effectivly a non linear impact on the system, the local linearisation of the effect of U will only involve an increase of controllers. In the context of the controllers network that increase of controllers should not make much difference to the control quality.

Assuming that the effect of the control input U on the system is linear, the number of data required to perform the polynomial identification of the closed loop system can be reduced to $(N+1) * O + O$ and the polynomial equation of a SISO open loop control system can be rewritten as

$$y_k = \sum_{i=1}^{O} b_i u_{k-i} \sum_{j=N}^{0} a_{ij} y_{k-i}^j \tag{6}$$

Now to perform the polynomial identification it is necessary to know the system order. That can be usually known in advance. The problem in fact is to define a priori the polynomial order O. Although polynomial functions are known to be capable to model most of the functions it is also known that a risk of severe oscillation increases quickly according to the order of the polynomial. To illustrate that phenomena we have ploted on figure 5 the results obtained from different polynomial approximations of the function $f(x) = \frac{1}{1+x^2}$ on the interval $[-4, 4]$. 13 equally spaced points were chosen for each approximation (they are symbolised by a '+' in each sub-plot). Each sub-plot depictes the function to approximate by a plain line and the polynomial approximation by a dotted line. The sub-plots intitle Poly1, Poly3, Poly6 and Poly12 approximation are concerned respectively by an approximation performed by a first order, third order, sixth and twelve order polynomial function of the form 4. You can first note that the approximation performed are not accurate at all. More importantly, you can see that more the order of the polynomial is high more the approximation is oscillating.

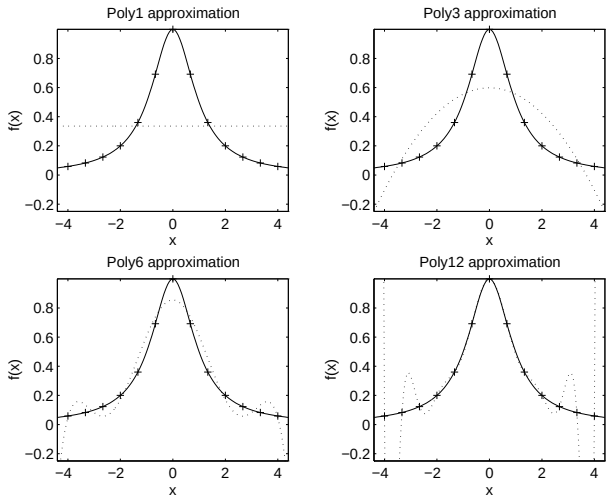

Fig. 5. Plots of polynomial approximations of the function $f(x) = \frac{1}{1+x^2}$ on the interval $[-4, 4]$. Each sub-plot depictes the function to approximate by a plain line and the polynomial approximation by a dotted line. The sub-plots intitle Poly1, Poly3, Poly6 and Poly12 approximation are concerned respectively by an approximation performed by a first order, third order, sixth and twelve order polynomial function of the form 4.

In addition to this oscillation problem, when the functions to identify are not smooth, polynomial functions are not really suitable for the modelisation. Hence, as emphase (Lancaster and Salkauskas, 1986) it is much more efficient to use piecewise polynomial functions of low order to obtain a very accurate and flexible fit of an unknown function, rather than trying to find the single polynomial function fitting the data. That constitutes another argument for the use of a network of controllers or models, since such a network performes a piecewise approximation of a function. To illustrate that fact we have use a Polynomial Model Network (PMN) to perform the identification of the function $f(x) = \frac{1}{1+x^2}$. This PMN is a CN composed of cubic polynomial models. The results are shown in fig. 6. You can see that the graph of $pmn(x)$ matches perfectly the graph of $f(x)$. Only 5 controllers were required to obtain this perfect identification. This identification is significantly better than the one obtained by a 15th order polynomial (see fig. 7). The identifications have been obtained using the 30 points '+' dipected on each figure. The graphes have been obtained using 100 points homogeneously spread within $x[-4; 4]$. This shows as well that the generalisation capability of the PMN is very good. A progressive identification method, similar of the PCD, where used

to perform the clustering depected in the bottom of fig. 6.

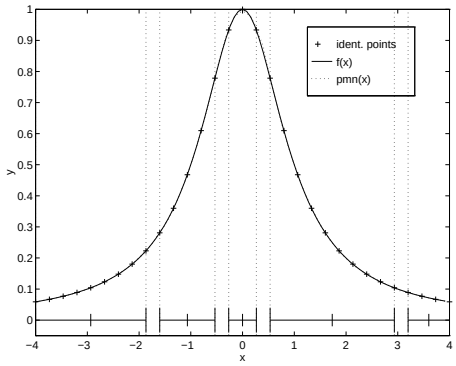

Fig. 6. Plot of the PMN identification of the function $f(x) = \frac{1}{1+x^2}$ on the interval $[-4, 4]$. $f(x)$ is depicted by a plain line and its PMN approximation by a dotted line. The 30 points used for the identification are depicted by a '+'. Finally, in the bottom of that figure you can see the clustering used by the PMN.

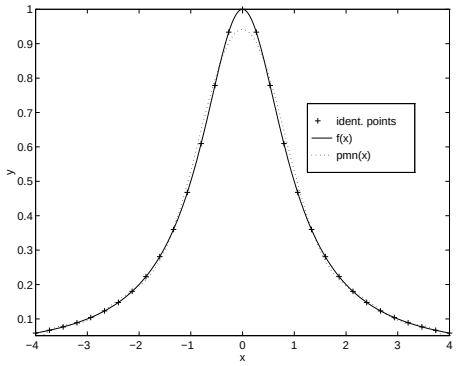

Fig. 7. Plot of the 15th order polynomial identification of the function $f(x) = \frac{1}{1+x^2}$ on the interval $[-4, 4]$. $f(x)$ is depicted by a plain line and its PMN approximation by a dotted line. The 30 points used for the identification are depicted by a '+'.

Note that an other advantage of using local models is that the polynomial functions (that are indeed continuous functions) can be bounded, simply by using bounded basis function as a gaussian function. This bounded property can be of great interest in cases of bounded systems, that are widely spread especially in the field of chemistry.

Model Reference Adaptive Polynomial Controller

We would like to first note that though we are presenting a Model Reference Adaptive Controller (MRAC) version of the Polynomial Controllers Network (PCN) other kinds of controllers could have been used, as for example, the general predictive controller. Note also that we will consider discrete time first order SISO systems during the remainder of this paper. This is only for simplicity because it is straightforward to generalise the current first order discrete time Model Reference Adaptive Polynomial Controller (MRAPC) to a continuous 2nd-3rd order MRAPC.

A MRAC is a very intuitive way to design controllers. The idea is to design the controller in order that it makes the system tracking the output given by a predefined model of the closed loop system i.e. "the model reference output" $\overline{y}$ (see fig. 8). Therefore this transient output corresponds to the desired transient response of the system defined by

$$\overline{y}_k = b_r r_k + a_r y_{k-1} \tag{7}$$

where k is the time indice, r_k is the control goal and y is the system output. $\theta^r = [b_r \quad a_r]$ (with $b_r = 1 - a_r$ to give $y = r$ in the steady state) is the vector of the parameters to set in order to define the desired transient system output. Note that this transient model of the closed loop system is linear rather than polynomial.

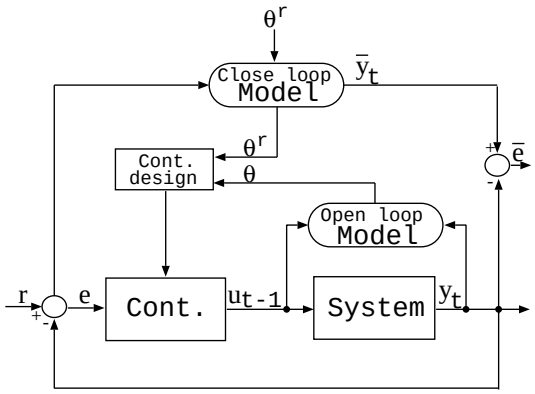

Fig. 8. The model reference adaptive controller. The controller is updated indirectly according to θ (i.e. the vector of parameters referring to the identification of the system) and θ^r (i.e. the given vector of parameters defining the closed loop model).

For the design of the controller the closed loop system has to be determined. This can be done indirectly by first identifying the system. Referring to (6), each polynomial local model of the system corresponds to the following first order polynomial model

$$y(k) = bu_{k-1} + \sum_{j=N}^{0} a_j y_{k-1}^j \tag{8}$$

where N is the polynomial order.

Linear regression method can be applied to identify the vector of parameters $\theta = [1 \quad b \quad a_N \quad ... \quad a_0]$. We use here the singular value decomposition (SVD) to identify on line these parameters. To do so an iterative information matrix S_t (9) has to be built on line. The best fit in a least squares sense is the vector corresponding to the smallest singular value.

$$S = S_{k-1} + X_k X_k^T \tag{9}$$

where $X_k = [y_k \quad u_{k-1} \quad y_{k-1}]$ is the vector of the variables of the local model (8) at time k.

Each first order polynomial controller is defined by

$$u_{k-1} = K_1 r_{t-1} + \sum_{j=N}^{0} K_{j+2} y_{k-1}^j \tag{10}$$

where $K_1 \quad ... \quad K_N + 2$ are the parameters determined from the equalisation of the closed loop system (11) and the reference closed loop model (7). From that equalisation we obtain $k_1 = \frac{b_r}{b}$, $k_2 = \frac{a_0}{b}$, $k_3 = \frac{a_1 - a_r}{b}$, $k_4 = \frac{a_2}{b}$, ..., $k_N + 2 = \frac{a_N + 2}{b}$. As you can see, since (7) is a linear equation and (11) a polynomial one, cancellations occur except for the two parameters related to u_{t-1} and y_{t-1}. These cancellations do not disrupt the control of the system.

$$y_k = bK_1 r_{k-1} + \sum_{j=N}^{0} (a_j - bK_{j+2}) y_{k-1}^j \tag{11}$$

Illustration

In order to clarify and illustrate the behaviour of the IMRAPCN we have decided to use systems where the control difficulty is related to their non linearity rather than to their dynamics. Thus we are going to consider the control of first order systems. The first system will be used to demonstrate the advantage of the IMRAPCN compared essentially to single MRAC and IMRACN (i.e. the linear version of the IMRAPCN). The second system will be used to show how low order IMRAPCN are much more suitable than single MRAPC.

IMRAPCN versus single and multi MRAC

In order to show the efficiency of the control IMRAPCN compared to single MRAC and IMRACN, we are going to use the following first order discrete time system

$$y_k = \sin(y_{k-1}) + U_{k-1} \tag{12}$$

This system will have to be controlled in order to reach $y_k = \pi$ being initialised at $y_0 = 0$. The non linearity of that system corresponds to figure 9. As you can see this is a rather difficult problem since this function is non-monotonic.

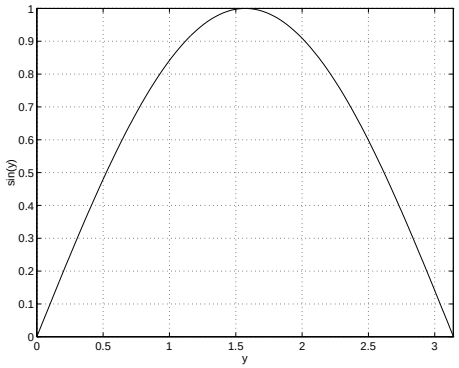

Fig. 9. Plot of the non linearity of the open loop system $f(y) = sin(y)$ according to values of system output y comprised between 0 and π.

The controller will have to track the closed loop model (7) where $a_r = 0.05$ and $b_r = 0.95$. We use a very low slope for the desired transient system output in order that most of the states y_k of the system are covered. Note that this is the simplest method to do so but there are other ways to cover the entire control possibilities. For example this can be achieved by initialising the system at different positions each time (as we did in (Ronco *et al.*, 1996*b*) and (Ronco *et al.*, 1996*c*)) and/or by changing each time the slope of the closed loop model (7).

Before starting the IMRAPCN it is necessary to set the acceptable control error related to the "model reference error" $\overline{e}$.

$$\overline{e}_k = |\overline{y}_k - y_k| \tag{13}$$

This error $\overline{e}$ is used to determine whether or not the controller currently selected is correctly controlling the system. If that error is more than a predefined value a new controller is added to the controllers network (CN). Thus a small error requirement will involve the creation of many controllers. Note that to have many or just a few controllers composing the CN is not an issue. The computing time involved by a CN composed of a huge number of controllers is the same as one involved by a CN composed of few controllers because only one controller is activated each time. In addition, the time involved by the selection of the valid controller is not significant since that selection implies very few computations. Thus it is always better to ask for a very small $\overline{e}_k$ in order that the system output y_k tracks as closely as possible the "real model reference output" $\hat{y}_k$ (10).

$$\hat{y}_k = b_r r + a_r \hat{y}_{k-1} \tag{14}$$

We require for the control of system (2) to never make an error $\overline{e}_k$ up to 0.001 in order to obtained a very accurate control of the system.

Before presenting the results obtain with IMRAPCNs of different polynomial orders let us consider the results obtained by two single linear MRACs, one having a forcing term (i.e. a constant parameter) and the other one not. From figure 10 you can clearly see that all the results are very bad since all the MRAC's graphs are far from closely tracking the desired system output graph. In particular, with no forcing term, the MRAC never makes the system reache the desired output (see MRAC's graph). With a forcing term (see MRAC1's graph) it tends to reach the desire output after 2500 sampling times which is around 25 times slower than expected. In addition, the transient output of the system control by MRAC1 is really not smooth. Furthermore, if we reinitialise the system and thus test again the MRAC1 we notice that it behaves in a completely different manner than previously. It is reaching the desired output after very few sampling time (almost 15 times quicker than expected) and then hold the system at that position. This change in behaviour can be interpreted according to the "stability-plasticity dilemma". At the end of the previous control adaptation the MRAC became adapted for the last operating conditions $y[3, \pi]$ and forgot its adaptation to previous operating conditions. This is why, at the second iteration, the MRAC seems only adapted for the very last operating conditions and not at all for the previous ones (for more details about "stability-plasticity dilemma" in adaptive control see (Ronco and Gawthrop, 1996) or (Ronco and Gawthrop, 1997*b*). Those results show clearly that single MRAC can not be applied to systems significantly non linear like (12).

It is now important to see what will be the efficiency of the IMRAPCN for the control of (12). Thus we are now going to show the result obtained by IMRAPCNs of different polynomial orders. Each local close loop system will be polynomials of the form of equation (11). Since the control of the system is completely related to its identification we are first going to present the results concerning the system identification of (12). Those results are depicted in fig. 11. In each sub-plot of that figure the graph of the system is represented by a plain line and the graph of the model of that system is in dotted line. The top left plot depicts the identification of the system performed by the linear IMRAPCN. 35 models were required to obtain this identification. This identification is really not accurate concerning the operating region $y[1, 2.5]$. In that region almost each model

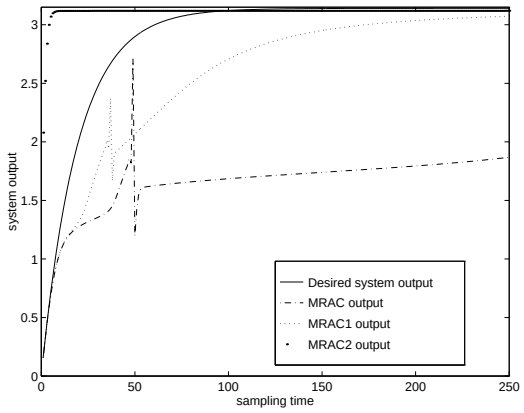

Fig. 10. Graph of the transient system output when the system (12) is controlled by different kind of MRAC: MRAC's graph concerns the result obtained by a single MRAC having no forcing term; MRAC1's graph is the result obtained by a single MRAC having a forcing term; MRAC2's graph is the result obtained after a second control of the system using the same controller (i.e. MRAC1)

fits only one input-output data. This explains why in the clustering (shown in the bottom of that sub-plot) the width of each basis function is so narrow. In fact, to have a perfect identification of this region, it will be necessary to have a rather high number of modeles (possibly infinite if we wanted to obtain a perfect identification). This problem of very local validity of a system linearisation makes clear the necessity of using non linear functions as building blocks of a networks of models or of controllers. This is the main reason why we are using here polynomial instead of linear functions. You can see, from the sub-plot untitled Poly2 model, that using simply a square IMRAPCN significantly reduced the required number of models to approximate that function. Only seven models are required to obtain a very good approximation of the system. What is interesting to notice is that the most non linear part of the system $y[1, 2]$ is accuratly approximated by a single model. This makes clear, that for the identification of non linear systems, polynomials, even of very low order, are much more appropriate than linear models. The bottom left sub-plot untitled Poly3 model shows the same kind of results. In this case the required number of models is reduced to 3. In fact a single model of a polynomial order equal to 6 can accurately approximate the system (see right bottom sub-plot untitled Poly6 model).

Now, concerning the control of system (12), the result are shown in figure 12. You can see that, even for the linear IMRAPCN, the control performed by any IMRAPCN is perfect, since for each case, the actual system output matches perfectly the desired system out-

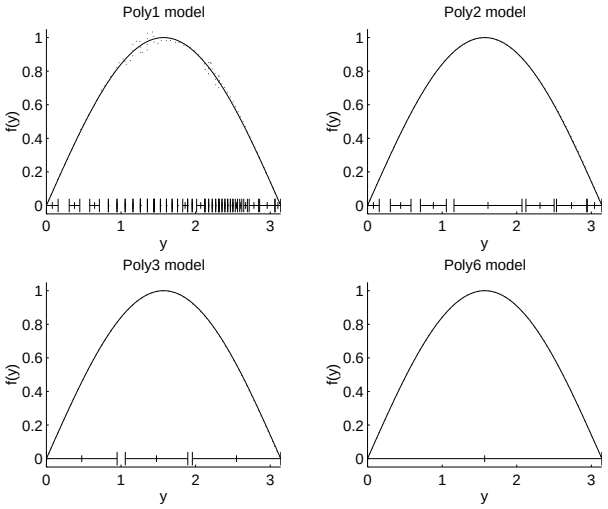

Fig. 11. Plot of the system identifications performed by a linear polynomial model network (see plot Poly1 model), a square polynomial model network (see plot Poly2 model), a cubic polynomial model network (see plot Poly3 model) and a 6th order polynomial model network (see plot Poly6 model). In each of the sub-plots the system is depicted by a plain line and the model of the system is depicted by a dotted line. In the bottom of each sub-plot is depicted the clustering achieved by each polynomial controllers network.

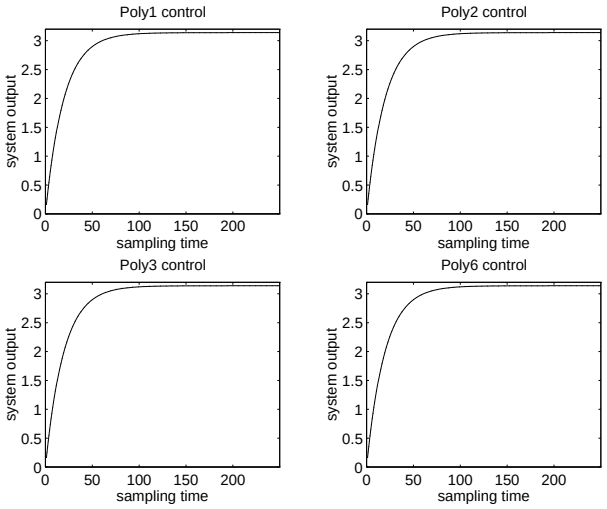

Fig. 12. Plots of the transient system output obtained by using respectively a linear IMRAPCN (see plot Poly1 control), a square IMRAPCN (see plot Poly2 control), a cubic IMRAPCN (see plot Poly3 control) and a 6th order IMRAPCN (see plot Poly6 control). In each of the sub-plots the transient desired system output is depicted by a plain line and the transient actual system output is depicted by a dotted line.

put. Compared to the very poor control performances we obtained with single MRACs (see fig. 10) IMRAPCN appears to be a powerful method for the control of non linear systems.

We must point out that those results were obtained by controlling the same system, always initialised at the same point and with the same closed loop model. Hence no interpolation property was tested, since the same data points were involved each time. It would have been different, if noise was included in the system, or simply, if we had tested the control interpolation of each IMRAPCN by using, at the end of the PCD, other parameters determining the closed loop model. We have done the latter interpolation test by making each controller tracking a new closed loop model (7) defined by $a_r = 0.1$ and $b_r = 0.9$ after having designed each controller for the tracking of (7) with $a_r = 0.05$ and $b_r = 0.95$. We obtained the same perfect control as in figure 12 for all the IMRAPCN but the linear IMRAPCN. For the linear IMRAPCN new controllers were requiered in order to control new situations involved by the new closed loop model (7). This means that the very local validity of each linear controller can involve the creation of a new controller for each new situation. Hence, such a linear controllers network, due to its poor generalisation capability, is not really appropriate for the control of a highly non linear system. This certainly justifies the use of IMRAPCN of higher order than one.

Low order IMRAPCN versus high order MRAPC

In this section we want to illustrate the ability of a IMRAPCN to control accurately a system obviously not suitable for a polynomial approximation. Such a system follows:

$$y_k = |y_{k-1}| + U_{k-1} \tag{15}$$

This system will have to be controlled in order to reach $y_k = 1$ being initialised at $y_0 = -1$. The behaviour of that system corresponds to figure 13. As you can see, this is a difficult approximation problem for a polynomial function, since the system behaves linearly except for a sharp change of sign that occurs at $y = 0$.

The first point to raise is that a polynomial of low order can approximate very accurately a linear system. That is due to the linear regression method used. At least, least square methods and the SVD, are capable of finding out that only the first order polynomial is relevant in the equation and thus, that the rest of parameters, related to higher order polynomials, must be equalised to 0. This is why we have obtained perfect identifications of system (15) using low order IMRAPCN (see

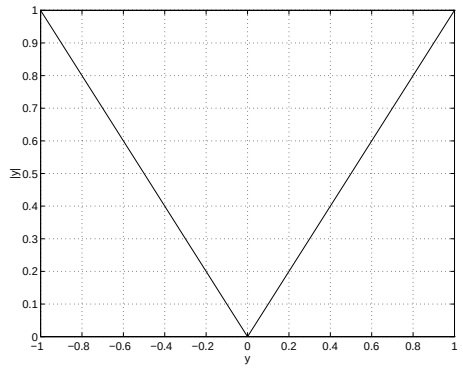

Fig. 13. Plot of the non linearity of the open loop system $f(y) = |y|$ according to values of system output y comprised between -1 and 1.

the sub-plot untitled Poly1 model in figure 14) and cubic IMRAPCN (see sub-plot untitled Poly3 model in figure 14). Each time only two models were required. The clustering achieved by each IMRAPCN is shown on the bottom of each sub-plot in fig. 14.

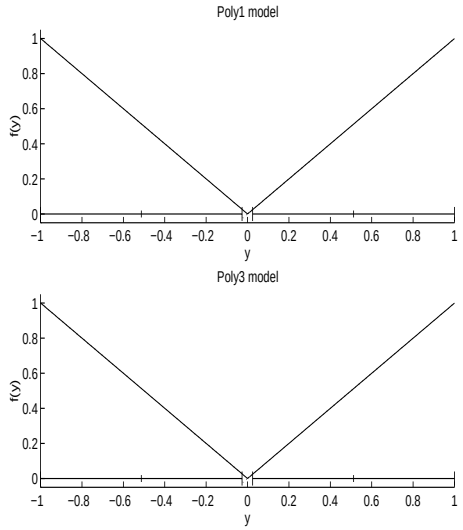

Fig. 14. Plot of the system identifications performed by a linear polynomial model network (see plot Poly1 model) and a cubic polynomial model network (see plot Poly3 model). In each of the sub-plots the system is depicted in plain line and the model of the system is depicted in dotted line. In the bottom of each sub-plot is depicted the clustering achieved by each polynomial controllers network.

The control of the system achieved by the linear and cubic polynomial is perfect as well (see the two sub-plots in the top of fig. 15). This is not the case of the control achieved by a single MRAC (see the bottom left sub-plot of fig. 15). The change of sign is so sharp that the controller makes the system reach 5000000 instead of 0 when the change of sign occurs. It eventually adapts itself to the new situation but it is much too late since the transient system output is far from satisfactory. If you make that same controller (now adapted for the operating region $y[0, 1]$) control again the system initialised at $y_0 = -1$, an important error occurs making the system reach the desired output in almost one step (see the bottom right sub-plot of fig. 15). This shows again that single adaptive controllers are defeated by the "stability-plasticity dilemna".

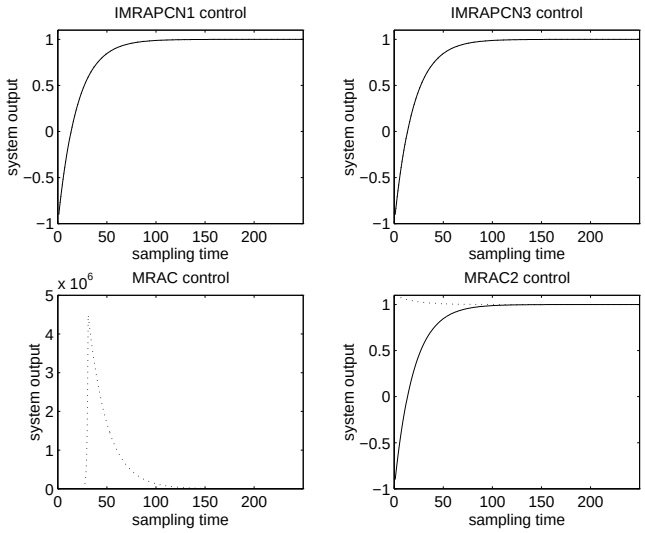

Fig. 15. Plots of the transient system output obtained by using respectively a linear IMRAPCN (see plot Poly1 control), a cubic IMRAPCN (see plot Poly3 control), a single MRAC (see plot MRAC control) and the same MRAC just mentioned but after having already controlled the system 15. In each of the sub-plot the transient desired system output is depicted in plain line and the transient actual system output is depicted in dotted line.

We have tried as well to adapt single polynomial MRACs of high order for the control of system (15). We could not find a solution although perhaps one exists. That shows anyway that, for the control of non linear systems, it is much more suitable to use IMRAPCN of low order than trying to adapt a single polynomial MRAC that perhaps does not even exist.

Conclusion

In this paper we have described and illustrated the IMRAPCN which is the polynomial version of the ILCN we have recently developed (see (Ronco and Gawthrop, 1996) and (Ronco and Gawthrop, 1997*b*)). The result obtained illustrate the powerful capability of the IMRAPCN:

- It enables a very accurate control of a non linear system over a wide operating range.
- Its behaviour is clearly understandable because each polynomial controller can be interpreted in linear terms.
- It is not affected by the "stability-plasticity dilemma".
- It is capable of an autonomous construction of its structure which involves the clustering of the validity space.

Our contribution is to develop a polynomial version of the conventional linear controllers network. We have seen that the use of polynomial controllers network enables a much more accurate control of a non linear system than the one that can be obtained by using linear controllers network. In addition, a polynomial controllers network of law order (e.g. cubic polynomial) is much more flexible than a single polynomial controller. This flexibility makes the polynomial controllers network adaptable to any kind of system. A single polynomial controller does not have this powerful adaptation capability. From these features the IMRAPCN appears very promising as a powerful and general algorithm for the control of non linear systems.

This introductory paper makes two simplifying assumptions:

(1) the system is set in discrete-time and
(2) the system is first order.

We believe that both restrictions can be removed. One promising way forward to remove the second simplification is to cluster on quantities which remain one-dimensional for high-order system; for example reference-model error.

Further work will consider the use of other kinds of controllers than the MRAC as building block of the polynomial controllers network. One potential candidate is the general predictive controller.

1. REFERENCES

Carpenter, G. A. and S. Grossberg (1988). The art of adaptive pattern recognition by a self-organising neural network. *IEEE Computer* **21**(3), 77–88.

Gawthrop, Peter J. and Eric Ronco (1996). Local model networks and self-tuning predictive control. In: *IEEE med.*

Johansen, T. A. and B. A. Foss (1992). A narmax model representation for adaptive control based on local model. *Modeling, Identification, and control* **13**(1), 25–39.

Lancaster, Peter and Kestutis Salkauskas (1986). *Curve and Surface Fitting: An Introduction*. Academic press, Harcourt Brace Jovanovich.

Murray-Smith, R. and K. J. Hunt (1995). Local model architectures for nonlinear modelling and control. In: *Neural Network Engineering in Dynamic Control Systems* (K. J. Hunt, G. R. Irwin and K. Warwick, Eds.). Advances in Industrial Control. Springer-Verlag.

Murray-Smith, Roderick (1994). A LOCAL MODEL NETWORK APPROACH TO NONLINEAR MODELLING. PhD thesis. Dept. of Computer Science: U. of Strathclyde. Glasgow, Scotland, UK.

Narendra, Kumpati S., Jeyendran Balakrishnan and Kemal M. Ciliz (1995). Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems* **3**, 37–51.

Ronco, Eric and Peter J. Gawthrop (1996). Incremental linear controllers network. Technical Report CSC-96008. Centre for system and control, U. of glasgow, UK. Available at www.mech.gla.ac.uk/~ericr/pub/ilcn_pcd.ps.

Ronco, Eric and Peter J. Gawthrop (1997*a*). Gated modular neural networks: a susrvey. *IEEE transaction Neural Networks*. (In preparation).

Ronco, Eric and Peter J. Gawthrop (1997*b*). Incremental linear controllers network. In: *Proceeding of the American Control Conference (ACC'97)*. (Submitted).

Ronco, Eric, Henrik Gollee and Peter J. Gawthrop (1996*a*). Modular neural network and self decomposition. Technical Report CSC-96012. Centre for system and control, U. of glasgow, UK. Available at www.mech.gla.ac.uk/~ericr/pub/mnnsd.ps.Z.

Ronco, Eric, Peter J. Gawthrop and Mohamed Abderrahim (1996*b*). Progressive local control. In: *World Automatic Conference*.

Ronco, Eric, Peter J. Gawthrop and Yasmine Mather (1996*c*). Incremental modular controllers network. In: *Proceeding of the International Conference on Intelligent and Cognitive Systems (ICICS'96)*.