Neural Optimal Control of PEM Fuel Cells with Parametric CMAC Networks

Paulo E. M. Almeida  
DPPG/CEFET – MG  
Av. Amazonas, 7675 – BH – MG  
30.510-000 – BRAZIL  
paulo@dppg.cefetmg.br

M. Godoy Simões  
Engineering Division, Colorado School of Mines  
1610, Illinois St. – Golden, CO  
80.401- 1887 – USA  
msimoes@mines.edu

Abstract — This work demonstrates an application of the Parametric CMAC (P-CMAC) Network — a neural structure derived from Albus’s CMAC algorithm and Takagi-Sugeno-Kang parametric fuzzy inference systems. It resembles the original CMAC proposed by James Albus in the sense that it is a local network, (i.e., for a given input vector); only a few of the networks nodes (or neurons) will be active and will effectively contribute to the corresponding network output. The internal mapping structure is built in such a way that it implements, for each CMAC memory location, one linear parametric equation of the network input strengths. First, a new approach to design Neural Optimal Control (NOC) systems is proposed. Gradient descent techniques are still used here to adjust network weights; but this approach has many differences when compared to classical error back-propagation algorithm. Then, P-CMAC is used to control output voltage of a Proton Exchange Membrane-Fuel Cell (PEM-FC), by means of NOC. The proposed control system allows the definition of an arbitrary performance/cost criterion to be maximized/minimized, resulting in an approximated optimal control strategy. Practical results of PEM-FC voltage behavior at different load conditions are shown, to demonstrate effectiveness of the NOC algorithm.

Keywords — Neural Networks, Control Systems, Fuel Cells, Optimal Control.

I. INTRODUCTION

Parametric CMAC (P-CMAC) artificial neural networks were introduced in the last two years [1], as fast and efficient alternatives to commonly used MLP networks for applications like function approximation and signal processing. P-CMAC architecture is based on the CMAC network and on Takagi-Sugeno-Kang parametric fuzzy systems (TSK-fuzzy systems). Cerebellar Model Articulation Controller (CMAC) networks were proposed in 1972 by James Albus [2], [3] to control robotic manipulators; but, the enormous success of such an algorithm in the control field created its utilization in some other important areas [4], [5], and even some improvements in the original algorithm, to overcome its limitations [6], [7]. Tomohiro Takagi, Michio Sugeno and Geuntaek Kang proposed TSK-fuzzy systems [8], [9]. As shown by some researchers, they present very good function approximation features [10], [11]. P-CMAC network retains the local data flow properties of CMAC and implements parametric fuzzy equations in a hidden layer inside the network, resembling fuzzy-TSK systems.

We demonstrated in previous work that P-CMAC works in a reliable and robust way to approximate nonlinear arbitrary mappings [4], (starting from big and ill-conditioned data sources), while other approaches had difficulties achieving good results. To illustrate this, P-CMAC was successfully used to solve a signal-processing problem at mobile telephony field [12].

Artificial neural networks (ANN) have often been used as basic and complementary tools inside process control systems: mainly in cases where conventional control techniques cannot be applied or do not offer a reasonable performance. Actually, process control researchers have begun replacing these conventional solutions (which demonstrated inefficiency) with alternate algorithms that are based on ANN and other similar techniques. For instance, some tasks like function approximation, dynamic systems modeling, parameter identification, optimization and process regulation are being performed using such algorithms in a simpler, easier and more effective way, when compared to the old techniques used before [13], [14].

This work presents a process control approach, based on P-CMAC networks, which approximates the behavior of an optimal controller. Design (or offline training) and tuning (or online training) steps are described in detail. This method is based on some other techniques used to approximate optimal control behavior in the literature [15], [16], [17]. But, there are clear differences between our approach and these techniques, as this paper will show. We named our approach Neural Optimal Control (NOC), to distinguish it from the existing architectures.

Here, a dynamic model of a proton exchange membrane fuel cell (PEM-FC) – originally developed at Federal University of Santa Maria, Brazil [18] – is adapted to run inside a control scheme. It was used as a test-bed for the proposed control architecture, and its output reversible voltage is controlled by means of a NOC system. Presented results showed that P-CMAC worked very well and achieved the desired objectives.

This paper is organized in five sections. The fundamentals of P-CMAC networks are briefly defined in Section II, where NOC architecture is also introduced. Section III addresses the adaptation of a dynamic PEM-FC mathematical model to
simulate a closed-loop control system. Section IV shows results obtained with the proposed control algorithm. Section V concludes the paper by analyzing obtained results and the NOC architecture performance.

II. P-CMAC NETWORKS AND NEURAL OPTIMAL CONTROL

A. P-CMAC Architecture

P-CMAC networks aim to minimize the existing limitations of Albus’s CMAC algorithm. Non-smooth response for smooth inputs, and poor representation capabilities for complex systems (due to binary behavior of CMAC receptive fields) are two of the classical problems with these algorithms.

P-CMAC networks have an important modification in their internal mapping when compared to CMAC. A CMAC internal memory was implemented as a bit memory location connected to the binary input activation functions. Fuzzy-CMAC networks use real valued memories, and each of them corresponds to the membership grade of the input with respect to the fuzzy activation functions [4]. In P-CMAC, these constant valued memories were replaced by parametric equations (depending on the network inputs) and the membership grades of these inputs with respect to fuzzy activation functions. Figure 1 shows a schematic diagram of the P-CMAC architecture:

![P-CMAC Architecture Diagram](image)

In Figure 1, the input mapping transforms input signals into membership grades with respect to the fuzzy activation functions defined within input spaces. Equation (1) illustrates this mapping, \( \mu_i(I_i) \), considering a given input \( I_i \) and a radial basis activation function represented by \( b_i(I_i) \). In this equation, \( c_i \) represents the radial function center and \( r_k \) its curvature index.

\[
\mu_i(I_i) = b_i(I_i) = \exp\left(-\frac{(I_i - c_i)^2}{r_k^2}\right). \tag{1}
\]

In the internal layer, the \( k^{th} \) neuron of a P-CMAC network implements Equation (2), where \( P_{mn} \) are real valued adjustable parameters, \( I_a \) are the inputs to the network and the operator \( \Pi \) represents a fuzzy conjunction operation between the membership grades of the inputs that are connected to this neuron. This operation is represented in Figure 1 by the legend “Nonlinear Mapping.” We used multiplication.

\[
A_i(I_i) = \left[ \prod_j \mu_{a_j}(I_i) \right] \left[ P_{00} + P_{11} I_1 + \ldots + P_{a_n} I_n \right]. \tag{2}
\]

Finally, the output layer executes weighted summations of the \( M \) neurons outputs \( A_i \) from the internal layer, as shown by Equation (3), and generates network outputs \( y \).

\[
y(A) = \sum_{i=1}^{M} w_i A_i. \tag{3}
\]

B. Neural Optimal Control

The main difficulty with using supervised training and ANN to control dynamic systems is to acquire input-output data suitable for the controller network-training phase. In the case of control systems, this is due to the desired controller output signals not being explicitly known by the designer. So, it is necessary to use an auxiliary algorithm to convert the known system desired outputs into the corresponding controller outputs responsible for these systems outputs. Most of the approaches that use neural networks to control dynamical systems employ the back-propagation algorithm, or a variant of it to obtain proper values to controller training.

Starting from an ANN trained as a model of the physical system to be controlled, the plant desired output is back propagated through this model network and generates, at the network input, a signal proportional to the control signal, which should be applied to the plant to drive it to the desired state. One of the first neural control approaches is the so-called direct-inverse neural control [19]. In this case, one tries to teach an ANN to behave like the inverse model of the plant to be controlled. Connected in cascade with the physical system input, this network should minimize the system dynamics and transfer the desired states directly to the system outputs.

This approach can be considered a version of the classical dead-beat controller adapted to the ANN paradigm. The advantages here are the online adjusting properties of the ANN, which make the resulting control scheme much more resistant to physical variations in the controlled system. This was one of the key problems limiting the employment of classical dead-beat controllers in actual control systems.

Common problems with direct-inverse control are the very large control signal amplitudes that result from the controller network [20]. That leads to a control system that depends on a very large amount of energy to control the plant. This occurs due to existing modeling errors, to imperfect poles and zeros canceling, and to the introduction of high frequency poles in the closed-loop system. This last side effect causes the resulting controller to generate highly varying control signals, which are not recommended to practical plants. As every practical system has energy limitations, this approach cannot be fairly employed to actual control systems.

A simple way to avoid the high amplitude and highly varying control signals is to insert a reference model in the
control loop. Now, the controller ANN is trained to respond in closed loop, as a linear system model chosen by the designer.

With these control schemes in mind, and starting from popular dynamic optimization techniques discussed in [13], here an innovative control architecture is proposed to integrate P-CMAC networks into an approximated optimal control system. The resulting architecture is based on the reference model direct-inverse control scheme; however, the training algorithm has been changed. The designer can now define a cost or performance criterion, and the controller network is trained to simultaneously regulate the system and minimize this criterion. An example of a typical cost criterion is shown below in Equation (4):

$$ U = \alpha \left( y_{\text{REF}}(t) - y(t) \right)^2 + \beta u(t)^2. \quad (4) $$

This criterion shows that the error between the desired state and actual state must be minimized and that, at the same time, the resulting control signal must be minimized. The quadratic function of control signal can be interpreted as the amount of energy used to control the physical system. Thus, this criterion is a trade-off between output error and the total energy spent to keep a low output error in the system. $\alpha$ and $\beta$ parameters can be adjusted to fine tune the performance of the resulting controller.

The control architecture hereby proposed was named Neural Optimal Control (NOC), to avoid confusion with other similar schemes existing in the literature. Although NOC is loosely based on the Paul Werbos back-propagation of utility [17], which in turn culminated with the proposition of neural dynamic programming (NDP) techniques [15], NOC does not use the adaptive critic network concept as NDP methods do. Moreover, NOC employs other tools, (like a reference model and a second P-CMAC network trained as a direct model of the plant), to obtain desired control signals and to adjust the controller network. Figure 2 shows a diagram of NOC architecture:

![NOC Architecture Diagram Using P-CMAC Networks](image)

In the above diagram, system output is compared to a reference model chosen by the designer. This offset signal is then fed back to the cost criterion, together with the back-propagated desired control signal from the direct model. Controller network is trained to minimize this criterion, approximating the overall operation to the operation of a conventional optimal controller.

In Figure 2, $\theta$ can be a non-measurable, environmental, constructive or random parameter. If of an environmental nature, it can represent temperature variations that could have influence over the control system. It can also represent a non-modeled constructive aspect as sensor and actuator nonlinearities. In addition, it could take into account low frequency noise existing in the process. Dust in connection pipes and equipment long-term drifts are good examples of such kind of low-frequency noise. At any rate, variation of $\theta$ will, from time to time, determine different operation points; the control system should work well despite the occurrence of any value for this parameter.

C. ANN with Long- and Short-Term Memories

In a sequence of technical papers, James Lo proposed the ANN with long- and short-term memories (LSTM), with application to identification of dynamical systems [10], [21]. In his work, the drawbacks of adaptive adjusting of ANN to system identification are discussed. By adaptive adjusting, James Lo means online training of all weights in the model networks. These drawbacks can be summarized as follows:

1. The existing algorithms to perform adaptive adjusting, (like back-propagation, Kalman Filters and others), spend a lot of memory and CPU time and have very low convergence rates.

2. The criterion to be minimized, in the case of a whole network with nonlinear neurons, presents a quadratic of higher order. This implies the existence of poor quality local minima. Online training allows neither multiple session optimization, nor employment of global optimization methods. Therefore, the probability of finding and getting stuck on a poor local minimum is high.

3. Online adaptation of all the network weights does not intelligently use available, large data sets. Because it does not give priority to identify the system dynamics (or some specific operation points of the plant), the resulting network is not a good approximation of the system to be modeled.

A natural way to avoid these drawbacks is to create an ANN with two different kinds of weights: some related to a long-term memory and others related to a short-term memory. The former must be trained offline from a big data set, which in turn should represent the system to be modeled for a variety of conditions and operating points. The latter must be trained online to fine-tune the network response to the instantaneous operating condition.

In other words, long-term training is focused on getting the main dynamics of the system modeled. After this offline learning phase, the long-term weights or memories must be kept constant, to preserve the grabbed information. The data set used to train the long-term memories in a LSTM neural network must include all operating points of interest and many different values for the environmental parameter $\theta$.

Now, let's consider $s$ different values for $\theta$, given by

$$ \Theta' = \{ \theta_1, \theta_2, \theta_3, ..., \theta_s \}. \quad (5) $$
These are exemplary values of environmental, constructive and low frequency noise referring to the system to be modeled. For each one of these \( \theta_i \), there should exist a subset with \( n \) input-output pairs, where each pair is \((u_{ij}, y_{ij})\), in such a way that \( y_{ij} = f(u_{ij}, \theta_i) \) and \( f \) represents a mapping between the system’s input and output signals. Input values \( u_{ij} \) must be uniformly chosen inside the region of interest (or in a region slightly larger than the actual region of interest), to allow for good approximation of its boundaries. At any rate, it is important that the long-term dataset comprise input-output pairs covering the whole region of operation of the plant.

The value of a long-term memory will be the same for all values of \( \theta_i \) (i.e. long-term training is independent of the various \( \theta_i \) considered in this training phase). This offline training of a LSTM network can be based on a long-term criterion like

\[
J_{LP}(w_{LP}, w_{CP}(\theta)) = \frac{1}{S,H} \sum_{i=1}^{S} \sum_{j=1}^{H} (U_{ij})^2 , \tag{6}
\]

where

\[
U_{ij} = y_{ij} - \hat{f}(u_{ij}, w_{LP}, w_{CP}(\theta)) \tag{7}
\]

is a utility function that measures the error between the desired and the actual output of the modeled system. During long-term training, this criterion is minimized for all possible values of the adjustable parameters \( w \). This is a soft-constrained optimization problem to which many traditional methods can be applied and often achieve exact solutions.

On the other hand, weights related to short-term memory should be trained online through a performance criterion based on the capture of specific environmental/constructive parameters currently active in the plant or control system. Adjustment of these weights will improve the controller performance for these values of \( \theta_i \). For this specialized training, one assumes that the value of \( \theta_i \) varies over time in an unknown or random way; but, there exists a number \( m \) of input/output data measurements between two consecutive changes on it. This data set can then be used to adjust short-term memories of the network so that they take into account this unique value of \( \theta_i \) now active. With general training, a performance criterion can consistently be defined for specialized training, like the expression

\[
J_{CP}(w_{CP}) = \frac{1}{m} \sum_{i=1}^{m} \lambda^{m-i} U_{ij}^2 , \tag{8}
\]

where

\[
U_{ij} = y_{ij} - \hat{f}(u_{ij}, w^{*}, w_{CP}) \tag{9}
\]

and \( w^{*} \) are long-term memory values obtained during general training and \( \lambda \) is a forgetting factor which weights each instantaneous utility function based on its temporal proximity. In other words, the older data will have less contribution during minimization of \( J_{CP} \) criterion.

If an ANN has only linear activation neurons at its output layer, (and the weights of this output layer are chosen to be the short-term memories of this network), then short-term training will be equivalent to a multiple linear regression algorithm. Thus, there will be no local minima, and convergence of training will imply that the global minimum of the short-term criterion has been achieved.

In summary, an ANN with LSTM has: (i) a long-lasting or long-term memory, reflecting basic features of the plant to be modeled and ensuring that further training in other parts of the network will not deviate much from the original knowledge captured during long-term training; and, (ii) a short-term memory with continuous or frequent adjustments occurring when environmental changes are detected, reflecting a response of the network to these changes, (i.e., an adaptation of the network to the newly verified operation conditions).

James Lo demonstrated that this approach leads to networks that exhibit universal approximation properties as well as classical MLP networks. With practical results, he showed that an MLP network with LSTM has a better performance than a classical MLP network with the same number of neurons, and avoids the training drawbacks just discussed [21], [22]. Considering that P-CMAC networks are feed-forward networks with features similar to those of MLP networks, we assumed that those advantages observed with MLP with LSTM would also be observed with P-CMAC. Practical results shown in section IV confirm our expectations.

D. Neural Optimal Control Synthesis with P-CMAC and LSTM

Two P-CMAC networks with LSTM are employed to control a dynamic system. The synthesis or training scheme is divided into a design phase (when general training is performed), and a tuning phase (when specialized training is performed). From a topological point of view, there is no difference between a conventional P-CMAC network and a P-CMAC network with LSTM. From a training perspective, a P-CMAC with LSTM has the following features:

1. The input activation weights and the parametric memories (i.e., the weights inside the parametric layer) turn into long-term memories of the network. They are trained during control system design phase and, during tuning phase, their adjusting factors are zero.

2. The weights at the linear output layer turn into short-term memories of the network. They are adjusted at design phase, together with long-term memories, and also at tuning phase.

Some differences between the training approach proposed by James Lo and the training approach hereby used are as follows:

1. Training Goal: James Lo proposed a MLP with LSTM to dynamic systems modeling. Here, two P-CMAC networks with LSTM are used to approximate optimal control behavior in a dynamic system. One step of this process does include a model identification phase, but a control goal should be considered an extension of a modeling goal.
2. Optimization Criterion: Lo used a quadratic error criterion at his work. Here, we synthesize a control system capable of approximating its behavior from that of an optimal controller, based on an arbitrary performance or cost criterion chosen by the designer.

An algorithm to synthesize a NOC system starting from P-CMAC networks with LSTM is described below in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Neural Optimal Control Synthesis Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Plant Input/Output data acquisition, to generate a P-CMAC direct model of the system to be controlled. This network is called ( M_{\text{pcmac}} ) (i.e., model by means of a P-CMAC network).</td>
</tr>
<tr>
<td>2.</td>
<td>Design of an initialization control system, able to stabilize the plant at the various operation points of interest and for different values of the environmental non-measurable parameter ( \theta ).</td>
</tr>
<tr>
<td>3.</td>
<td>Controller Input/Output data acquisition, to generate an initial neural controller to the system.</td>
</tr>
<tr>
<td>4.</td>
<td>Controller design phase or general training. Starting from the data collected at step 3, a P-CMAC network with LSTM called ( C_{\text{pcmac}} ) (i.e., controller by means of a P-CMAC network) is trained to function like the initialization controller.</td>
</tr>
<tr>
<td>5.</td>
<td>Tuning phase or specialized training. Online training of ( C_{\text{pcmac}} ) network, to minimize/maximize the chosen cost/performance criterion. Only short-term weights are adjusted at this time.</td>
</tr>
</tbody>
</table>

### III. Dynamic Modeling of PEM-FC

#### A. Basic Functional Principles

A fuel cell (FC) is an electrochemical device that converts chemical energy of hydrogen gas \( (H_2) \) and oxygen gas \( (O_2) \) into electrical energy proper to consumer applications. Today, there are many different technologies associated with this type of energy conversion. The specific technology to be used depends mostly on the adequate amount of energy used to drive an application, (i.e., the power consumption of the application in mind). For low power domestic appliances, the most used technology these days is the proton exchange membrane fuel cell (PEM-FC). A typical generation scheme based on a PEM-FC is shown in Figure 5:

![Typical PEM-FC System](image)

Figure 3: Context Diagram Showing Input and Output Variables of a Typical PEM-FC Generation System [26]

#### B. Simulink Dynamic Model of a PEM-FC

In current literature, there exist various mathematical models for evaluating the operation of PEM-FC. Some are based on curve fitting experiments, starting from acquired data [23]. Others are semi-empirical models that combine experimental data with parametric equations adjusted by comparison with cells physical variables like pressure and temperature [24].

In both cases, the phenomenon of Concentration Over-Potential is unfairly treated, because of the simplifications adopted by some, or because of the static behavior of others. Concentration Over-Potential is crucial in describing the dynamical behavior of such systems. The work developed by Corrêa et al. [18], [25] takes this effect into consideration, (inside a physical variable modeling approach), and achieves a highly accurate model for real world PEM-FC.

Starting from the electrochemical and thermodynamic representations of a PEM-FC, and from the approach described in [26], a detailed dynamical model was created using the software Simulink® by MathWorks. This model will be used as a test bed for NOC architecture proposed in Section II. The model is divided into six main sections according to the physical phenomena described by Corrêa, Farret and Canha in the above-cited works. Figure 4 shows a Simulink® diagram representing the adapted model.

On the left in Figure 4, one can see the input gas pressures, temperature level, and electrical current imposed by the externally applied load. Going right, the four main internal generation and consumption sections are shown from top to bottom. They calculate the following potentials:

1. Nernst Potential or Open Circuit Voltage: the basic reversible voltage generated inside an FC, coming from the variation of the free Gibbs energy in the chemical reactions occurring inside the cell.

2. Ohmic Over-Potential: results from resistance to the electrical transfers through the conductor plates and the carbon electrodes, besides the electrical resistance of the composite membrane.

3. Activation Over-Potential: leakage caused by reduction in the speed of chemical reactions in the electrodes surfaces.

4. Concentration Over-Potential: mass transportation existing inside a FC affecting concentration of input gases causes fluctuation in the partial pressures of them and results in voltage drops in the output.
IV. PRACTICAL RESULTS

As a validation test to the adapted model, a simulation session identical to that presented by Corrêa, et al, is performed [18]. In that simulation, a PEM-FC starts running in an open circuit situation (i.e., no load connected to the output terminals). Next, an electrical load that consumes 15A of current is connected to the FC terminals for 10s. In this case, the open loop response of an FC is a high voltage drop until an equilibrium level is reached between generated and consumed power. This occurs because the input gas pressure is maintained constant throughout the simulation time.

A typical result of a control simulation session is shown in Figure 7. Inspecting the graphics, one sees that the PID controller is able to maintain a fairly constant value of the output voltage, for various values of electrical load applied during the simulation time. However, a considerable offset value is present at almost all times, varying between hundreds of milivolts to few volts, (depending of the specific load applied to the FC terminals). This seems to be a limitation of the designed PID controller for the operation conditions tested.

The cost criterion chosen weights the difference between reference voltage and actual output voltage of the PEM-FC model. This function, expressed by Equation (10), is minimized during the specialized training phase of C_{PCMAC} network, as shown in Figure 8:

\[ U(t) = \alpha \left( V_{REF}(t) - V(t) \right)^2 \]  

(10)
Next, input/output data of the PID controller, collected during the control session just described, are used in the general training of the P-CMAC network $C_{PCMAC}$, which will function as the controller network inside the NOC architecture (see Figure 2).

Controller tuning is difficult to accomplish when the plant under control is somewhat complex and exhibits nonlinearities. In that case, an immediate advantage of NOC architecture, over traditional PID control, is that its tuning phase is performed in an automated way, through the minimization of Equation (10) over time.

V. CONCLUSION

This work proposed a new architecture to synthesize approximated optimal control by means of ANN. The NOC architecture employs two simultaneous networks: one functioning as a direct model of the plant under control, and the other functioning as the controller, generating control signals to be applied to the plant. The ANN with LSTM approach (introduced by James Lo for MLP networks) was adapted for use inside NOC architecture and P-CMAC networks. The advantages of using this approach are noticeable, in terms of both computational costs and improved performance viewpoints. An algorithm for synthesis of NOC systems was fully described and on a step-by-step basis.

A control system of PEM-FC was chosen as a test bed for the NOC architecture. Starting from an existing model, a dynamical model was adapted for use inside Simulink®. An NOC system was then synthesized (i.e., trained) to control the PEM-FC output voltage. Obtained results suggest that NOC works well, and has operation advantages when compared to the design drawbacks and worse performance exhibited by a typical PID controller. Specifically speaking, automatic tuning capability and the choice of the control goals by the designer grant NOC architecture great versatility and operation robustness.

Synthesis efforts, and the reliable maintenance of needed topology during execution are bottlenecks not yet completely...
analyzed by the NOC user. In the case of complex nonlinear plants that are difficult to control, and for systems that need accurate regulation behavior, NOC architecture is absolutely justifiable. For simpler control problems that do not require a high degree of performance and cost control, conventional justifiable. For simpler control problems that do not require a high degree of performance and cost control, conventional techniques can continue to be used for many years. It is also important to note that NOC synthesis assumes the existence of an initial controller (which must be able to maintain the plant in stable operation) as a starting point for its design. This assumption is not always true and can limit the practical applicability of this algorithm.

ACKNOWLEDGMENT

The authors are grateful to Jeferson M. Corrêa for his help in modeling a fuel cell system. The National Science Foundation with grant ECS-0134130, and the Brazilian agency CAPES have partially supported this work.

REFERENCES


