APPROXIMATE OPTIMAL CONTROL APPLIED TO CONSTRAINED CONTINUOS PROCESSES

J. PUCHETA, R. FULLANA, C. SCHUGURENSKY Y B. KUCHEN

Instituto de Automática - Facultad de Ingeniería. Universidad Nacional de San Juan
Av. Libertador 1109 (o), 5400 San Juan, Argentina. E-mail: ipucheta@inaut.unsj.edu.ar

Abstract.
Constrained continuous processes can be optimally controlled through dynamic programming techniques that solve it as a numeric sequence. These techniques are a powerful tool, regardless the nature of the system and its proposed performance index. This technique is a powerful tool, regardless the system's nature and arbitrary performance index. However, some problems may arise in calculations as the problem dimensionality increases -a factor closely related to the desired accuracy for the numeric solution. An alternative is used here to solve the dimensionality problem, both to approach the performance index and the control law. The present work aims at outlining the approximate optimal control applied to infinite horizon continuous processes. A main contribution is to generate an application methodology that ensures the convergence of the algorithm. Upon obtaining the approximate control law, a comparative analysis of the controller performance demonstrates the potential of the proposed control scheme.

Key words: Optimal control, neural network, nonlinear systems.

I. INTRODUCTION

Typically, the optimal control of constrained continuous processes can be described with the classical control theory, which solves it without regarding constrains. Once a solution has been found, if it verifies that the constraints are met, it is accepted as the problem's solution. However, the solution is not always reached with this approach. An alternative technique deals with dynamic programming that renders a numerical solution to the problem in a form of a look-up table. This technique is a powerful tool, regardless the system nature or its arbitrary performance index. Its main limitation is found when increasing the dimension of the problem as, for example, to obtain a greater precision. With an increased problem dimension, the computer and computations requirements grow as well –something difficult to overcome with ordinary computer equipment. This approach is more visible in a real process application, because the controller's implementation demands a sizable memory space in order to store the control action values for each possible system state.

This dimensionality problem has been approached in several ways, in an attempt to decrease the number of table values (Luus, 1989). An alternative way is to generate a compact table through an approaching function. This methodology was formalized in Bertsekas (1996), approaching both the performance index and the control law. This technique will be applied here to solve the optimal control problem for a nonlinear process, with a non-quadratic index.

II. STATEMENT OF THE CONTROL PROBLEM

The process dynamic model, the cost function, and the control and state variable limitations make up the optimal control problem. At times, constrains are omitted, a solution is found and, if under

execution it verifies the restrictions, such a solution is accepted. This methodology solves the problem in several real applications requiring heuristic knowledge, so that the controller performance be efficient. In the following steps, the optimal control problem will be stated by considering all the elements simultaneously, where the problem solution will be the optimal control law itself.

$$J = \sum_{k=0}^{N} g\left(\mathbf{x}_{(k)}, \mathbf{u}_{(k)}, k\right) \tag{1}$$

$$\mathbf{x}_{(k+1)} = f\left(\mathbf{x}_{(k)}, \mathbf{u}_{(k)}, k\right), \qquad k = 0,1,...N-1 \tag{2}$$

$$\mathbf{x} \in S \subset R^{n}$$

$$\mathbf{u} \in U \subset R^{m} \tag{3}$$

where:

J: cost function to be minimized (bounded and positive definite function).
g(·): performance index (bounded and positive definite function).
f(·): process dynamic model.
x: system states; within a closed and bounded subset $S \subset R^{n}$.
u: system inputs; within a closed and bounded subset $U \subset R^{m}$.

We seek for a decision function of "optimal policy", which makes the system -with the dynamic model (2)- evolve from any state and initial time until the evolution's end, i.e. its N-th stage. The decisions sequence must comply with constraint (3) and minimize as well the cost function of Eq.(1). The control scheme is shown in Fig. 1, where the control law is the optimal policy. This outline tries to solve a finite horizon problem, but it can be extended to a problem of infinite horizon by making $N \to \infty$, and by verifying that the proposed cost function (1) is bounded and continuous as regards its arguments.

The typical method to solve the above-stated problem tries to find a control law $\mu(\mathbf{x}_{(k)})$ with $\mu : R^{n} \to R^{m}$. In real cases, however, it is difficult to find an analytic solution, which leads to resort to numerical methods, where the control law is represented as a look-up table (Luus, 1989; Bellman, 1962).

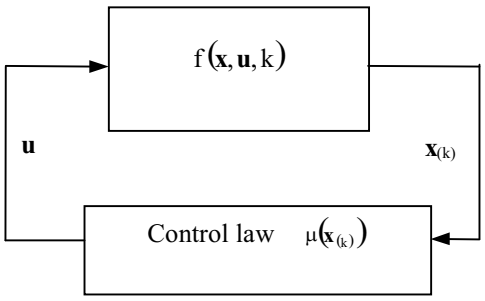

Fig. 1. Optimal control scheme.

III. A SOLUTION PROPOSAL

In this section, we will use a technique called *approximate policy iteration* (Bertsekas, 1996) in order to find a compact approaching expression for the control law $\mu(\mathbf{x}_{(k)})$. After presenting theoretical elements of the technique, a variable transformation will ease the calculations in approaching the control law.

In describing finite horizon problems, i represents each value that the state variable $\mathbf{x}_{(k)}$ may take, both in magnitude and in time, aiming at discerning its value at any given time from that of other times. The new state variable i has a limited numbery of possible values. It belongs to a data set S as shown in Eq. (4). For example, if a single state variable takes two different values at three different instants of time, it will have 6 different values (size of S = 6).

$$i \in S \subset \Re_{+} \times \Re^{n} \tag{4}$$

$$S = \left\{ i \in S : \left\| i - p_0 \right\| \leq r \right\} \tag{5}$$

A difference arises when dealing with infinite horizon control problems, because the problem statement becomes time independent. The data set S will not show any difference between state values at different time intervals; therefore, Eq. (5) will represent the i data set.

A function is intended whose behavior approaches the performance index. Once this approximate cost function is obtained for each process state, together with the system model, the controller for the real process can be implemented. Then, the system can be controlled with a function having an approximate performance to that of the optimal control. The usual practice is to control a real process, which is continuous in nature with respect to the magnitudes that its variables may assume.

A. Introduction of the Approaching Function

A function $\tilde{J} : R^n \rightarrow R$ is sought that, for values of domain (6), generates an image whose magnitude is approximately similar to that of the cost function (1). Let $J^{\mu}_{(i)}$ be the cost value that represents the cost-to-go from state i to the final stage, using the stationary policy control μ. Then, $\tilde{J}^{\mu}(\cdot, \mathbf{r})$ is the approximation of $J^{\mu}(\cdot)$ into domain (3).

$$\tilde{S} = \left\{ i \in S : \left\| i - p_0 \right\| \leq \tilde{r} \right\} \tag{6}$$

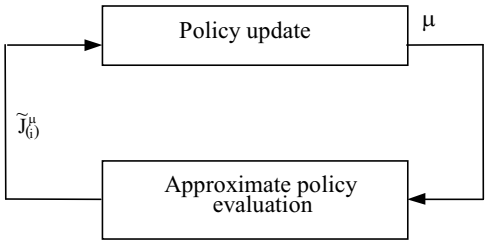

Fig. 2. Optimal policy search process diagram.

The vector **r** contains all the tunable parameters of the approximate cost-to-go function, and the **r** dimension depends on the number of parameters that achieve an appropriate approximating performance.

By adequately tuning the parameter vector **r**, a cost can be approached, needed to arrive at the final state from the current state i, using the stationary policy μ.

In order to adjust the parameters of **r**, the calculation process is divided twofold, as shown Fig. 2. First, an evaluation of a given stationary policy μ (initially random) is carried out, where costs are

calculated for all process states. The second part is for policy updating. Both tasks are done approximately, if compared to the actual system.

Any device capable of approaching the performance of functional one (1) in domain (3) is useful to carry out this solution statement. Generally, **r** -the parameter vector of the approaching function- should be found. This vector will have as many parameters as required by the problem's complexity. In this work, the approaching function will be a neural network of five neurons, with one hidden layer whose neurons operate with hyperbolic tangent functions and one output neuron that operates with a linear function. The network behavior requires to calculate two matrixes W_1 and W_2 whose elements are represented by a parameter vector **r** containing all the tunable parameters of the neural network.

$$\min_{\mathbf{r}} \sum_{i \in \tilde{S}} \left(\tilde{J}(i, \mathbf{r}) - C(i) \right)^2 \tag{7}$$

$$\mathbf{r} := \mathbf{r} + \gamma \nabla \tilde{J}(i, \mathbf{r}) \left(C(i) - \tilde{J}(i, \mathbf{r}) \right) \quad \forall i \tag{8}$$

$$\bar{\mu} = \arg \min_{u \in U(i)} \left(g(i, u) + \tilde{J}(j, \mathbf{r}) \right) \quad \forall i \tag{9}$$

A representative data set $\tilde{S}$ is defined, and for each state $i \in \tilde{S}$ there exist cost values C(i) of the function $J^{\mu}_{(i)}$. The rest of the data set S will be considered by the approaching function as a generalization, because they will be used to validate its performance. By minimizing expression (7) the parameters vector **r** is tuned. The improved policy is obtained as indicated in (9). Therefore, given the stationary policy μ, the values of J^{μ} will not be calculated exactly because an approximation is used through $\tilde{J}^{\mu}(\cdot, \mathbf{r})$. Thus, in an approximate policy iteration an error source arises due to the approximator structure $\tilde{J}^{\mu}(\cdot, \mathbf{r})$, that is not powerful enough (just few neurons). Another error source is found in the inadequate adjustment of **r** parameters (faulty algorithm tuning).

Once the function $\tilde{J}^{\mu}(\cdot, \mathbf{r})$ is available, $\bar{\mu}$ can be obtained from Eq. (9). A method requirement is that the initial policy be as good as possible on the sense of associated costs; else, the parameters vector **r** will be fixed. Usually, initial random values –varying between 0 and 1- are assigned to matrixes W_1 and W_2. Then, the associated costs to each state are evaluated, denoted by $\tilde{J}^{\mu}(\cdot, \mathbf{r})$, and when the associated costs to all states are available, we proceed to the second process stage (see Fig. 2). This stage is a policy update, aimed at improving its performance, and where the parameters of vector **r** and the neural network take this update as new data to start again a new cost calculation for each state. For the present case, the Levenberg-Marquard method was used to tune **r** (Bishop, C. 1995; Nørgaard, M. 1997).

At the real application moment, the above method has a drawback in that Eq.(9) should be solved, which requires a numerical minimization whichever the state the system may be staying at. Because, the method is used to a limited extent in many of the actual processes, a control statement is sought for a more compact scheme (Fig. 1), involving another neural network to generate each control action.

B. Using an action network

To implement any control scheme implies to obtain the control action to be applied to the system from the states vector value.

$$\begin{aligned} \mathbf{x}_{(k+1)} &= f(\mathbf{x}_{(k)}, \mathbf{u}_{(k)}, k) \\ \mathbf{u}_{(k)} &= \mu(\mathbf{x}_{(k)}) \end{aligned} \tag{10}$$

$\mu(\mathbf{x}_{(k)})$ $\mu : R^n \rightarrow R^m$ are the decisions of the optimal policy function, for a given system state $\mathbf{x}_{(k)}$.

The function μ is the analytical solution of the optimal control problem. By means of the approximate policy iteration technique an approach is sought. As before, variable i represents each value that the state variable $\mathbf{x}_{(k)}$ may take, both in magnitude and in time, to make a time instant different from another. Now we look for a function with a form $\tilde{\mu}(\cdot, \mathbf{v})$ as an approximation of the $\mu(\cdot)$ function, where $\mathbf{v}$ is the tunable parameter vector. The dimension of $\mathbf{v}$ depends on the proposed structure for the approaching function.

$$\hat{S} = \left\{ i \in S : \left\| i - p_0 \right\| \leq \hat{r} \right\} \tag{11}$$

The following methodology will try to obtain $\tilde{\mu}(\cdot, \mathbf{v})$, which presents a similar performance to that of the analytic solution $\mu(\cdot)$. The control actions of the improved policy $\bar{\mu}(i)$ are calculated with Eq. (9) , within the data set $\hat{S}$ (11). The decision law is represented by $\tilde{\mu}(i, \mathbf{v})$.

(12)

$$\min_{v} \sum_{i \in \hat{S}} \left(\tilde{\mu}(i, \mathbf{v}) - \bar{\mu}(i) \right)^2$$

$$\mathbf{v} := \mathbf{v} + \gamma \nabla \tilde{\mu}(i, \mathbf{v}) \left(\bar{\mu}(i) - \tilde{\mu}(i, \mathbf{v}) \right) \tag{13}$$

The incremental gradient iteration of Eq. (13) adjusts the parameters of $\mathbf{v}$. Two problems are solved at the same time, because, given μ, an approximate policy evaluation is carried out to find $\tilde{J}$ of J^{μ}. Given $\tilde{J}$, the improved policy $\bar{\mu}$ is calculated for some states; then, the new policy $\tilde{\mu}(\cdot, \mathbf{v})$ becomes available.

With function $\tilde{\mu}(\cdot, \mathbf{v})$, it is possible to obtain the control actions starting from any state $i \in S$. The control scheme is shown in Fig. 1, excepting that the control law -instead of being $\mu(\mathbf{x}_{(k)})$- will be $\tilde{\mu}(\mathbf{x}_{(k)}, \mathbf{v})$. Function $\tilde{\mu}(\cdot, \mathbf{v})$ is implemented by a multiplayer perceptron type neural network. It has 5 neurons, one hidden layer whose activation functions are of hyperbolic tangent type, and the output neuron operates linearly. The parameter vector $\mathbf{v}$ has all the neural network elements, which are the weight coefficients, expressed by matrixes W_{1A} and W_{2A}.

C. Stability consideration

In order to assure stability for algorithm convergence of the policy iteration, special care should be taken in the policy update of Eq. (9), because a state j should have been used in the Eq. (7) to generate the cost approximation function. If a policy gets improved with respect to a value generated by the approaching function $\tilde{J}^{\mu}(j, \mathbf{r})$, it is possible that the value be located very far as regards the real value $J^{\mu}(j)$ and, consequently, the policy instead of improving, will worse its associate cost.

$$\delta m = \underset{u \in U(i)}{\text{Max}} \left(\left\| i - g(i, u) \right\| \right) \quad \forall i \in \hat{S} \tag{14}$$

$$\tilde{r} = \hat{r} + \delta m \tag{15}$$

An alternative means to ensure the policy improvement, is to set a relative location for the data set $\tilde{S}$ and $\hat{S}$, verifying that every potentially reachable state j -for a given a policy improvement- will belong into the set $\tilde{S}$. This condition makes that $\hat{s}$ be contained in $\tilde{S}$, thus assuring that $\tilde{r}$ be greater than $\hat{r}$ and, consequently, there will exist a minimum margin δm of difference between them as given by Eq. (14).

IV. APPLICATION EXAMPLE

The above technique will be applied to a linear system with input restrictions, as shown in Fig. 3. The system to be controlled is nonlinear and the performance index is non-quadratic, which hinders the use of classical methods. The implementation methodology can be used in any continuous process, by following the dynamic programming application method detailed in Bellman (1962) and by considering the necessary premises for the approach that are presented in Eq. (14).

A. Two state variables system

For the system modeled by Eq.(16) through dynamic programming with approximate performance index, the decisions sequence u is computed, which complies with constraints (17) and minimizes the defined performance index; with 1.5, the desired value of the system output.

$$F_{(S)} = \frac{Y_{(S)}}{X_{(S)}} = \frac{3}{S^2 + 2 \cdot S + 3} \tag{16}$$

$$0 \le u \le 2 \tag{17}$$

$$\lim_{t \to \infty} y_{(t)} = 1.5 \tag{18}$$

The process model is transformed from the continuous to the discreet domain with sampling time T_0 equal to 6/109 sec.

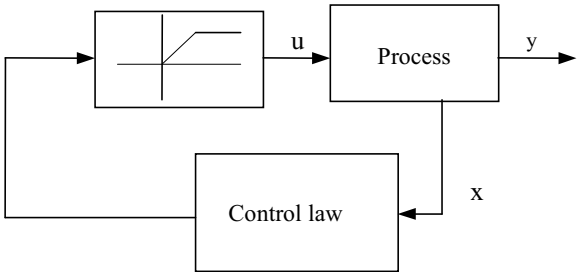

Fig. 3. Control scheme.

The performance index is proposed in such a way as to incorporate the required parameters to control the process.

$$g(y,u) = \theta_1 \cdot \left| y_{(t)} - 1.5 \right| + \theta_2 \cdot P \tag{19}$$

where:

θ_1, θ_2: weigh coefficients.

P: Constraints function.

$$P = \begin{cases} 0 & \text{if} \quad 0 \le u \le 2 \\ (u-2) & \text{if} \quad u > 2 \\ (0-u) & \text{if} \quad u < 0 \end{cases}$$

$$J(y,u) = \int_0^\infty g(y,u)dt \tag{20}$$

The problem formulation of the optimal control is to find a control law function that fulfills the restrictions (17) and minimizes the cost function (20).

B. Solution through neuro dynamic programming

By following the above detailed steps for the approximate optimal control, the statement for the state variables of the system is carried out. The data set S is formed by the attainable values of vector **x**. The data set $\tilde{S}$ is composed by those values that conform the domain space for the approaching function $\tilde{J}(\cdot,\mathbf{r})$, whereas the set $\hat{S}$ corresponds to the domain of the control policy function $\tilde{\mu}(\cdot,\mathbf{v})$. By meeting the constraint set by Eq. (14), a selection -shown in Eq. (21)- is made for $\tilde{S}$ and showin in Eq. (22)- for $\hat{S}$. Fig. 4 shows the graphic representation.

$$-1 \le x_1 \le 1 \tag{21}$$
$$-1 \le x_2 \le 1$$

$$0 \le x_1 \le 0.7 \tag{22}$$
$$0 \le x_2 \le 0.7$$

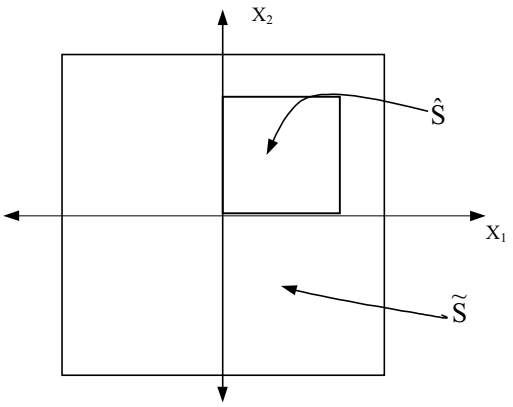

Fig. 4. Spatial location of the approaching function domain.

21,301019385	93,4055549692	-43,7661740917
2,8789199952	14,0757506682	-7,23691854700
2,51827024874	13,0084092351	-6,69509322111
3,34784834103	15,956248038	-8,18651110583
-2,60630689115	-12,6594639291	6,5271312701

Table 1. W_{1A} of the action network.

0,197999145547761
36,2385195503809
-8,59419701523366
-19,7653877925509
9,08449917289846
0,996075106524353

Table 2. W_{2A}^{T} of the action network.

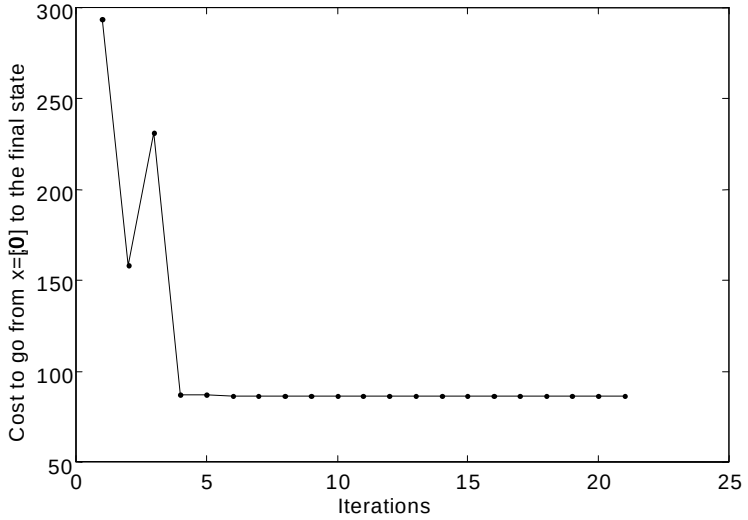

Fig. 5. Policy convergence to the optimal policy.

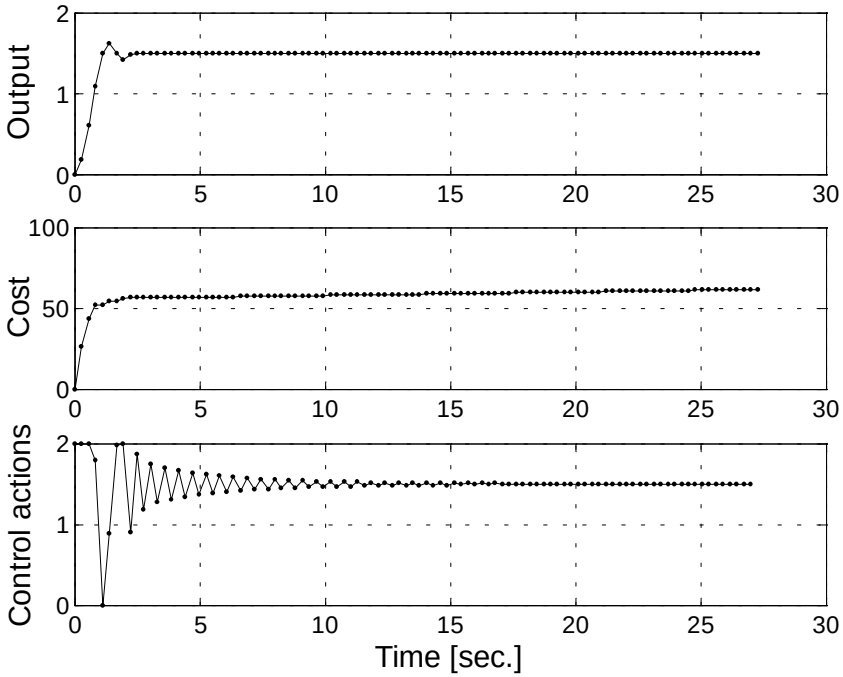

Fig. 6. System evolution using NPD. Cf=64.58.

C. Controller performance

To evaluate the optimal controller performance by means of an approximate index, we make a quantitative comparison with the controller obtained with the classical control theory. This controller is represented by the matrix K=[13.74 11.89]. The introduction of noise into the model turns uncertain its knowledge. Nevertheless, the control objective should be completed, in certain extent. The noise that disturbs the values of system states will be random and short at first; then, they will be long and with a certain amplitude.

%	NDP	DDP	Linear
0	64.58	100.16	118.95
20	253.32	411.52	546.55
40	559.87	777.46	913.28
100	$1.69\ 10^3$	$2.52\ 10^3$	$2.55\ 10^3$

Table 3. Cost values obtained with the controllers with sustained perturbations using neuro-dynamic programming, direct dynamic programming and linear optimal control.

%	NDP	DDP	Linear
0	64.58	100.16	118.95
20	105.14	404.91	603.03
40	1997.28	4488.37	5866.51
100	$7.31\ 10^6$	$8.19\ 10^6$	$1.04\ 10^7$

Table 4. Cost values obtained with the controllers under stepwise perturbations using neuro-dynamic programming, direct dynamic programming, and linear optimal control.

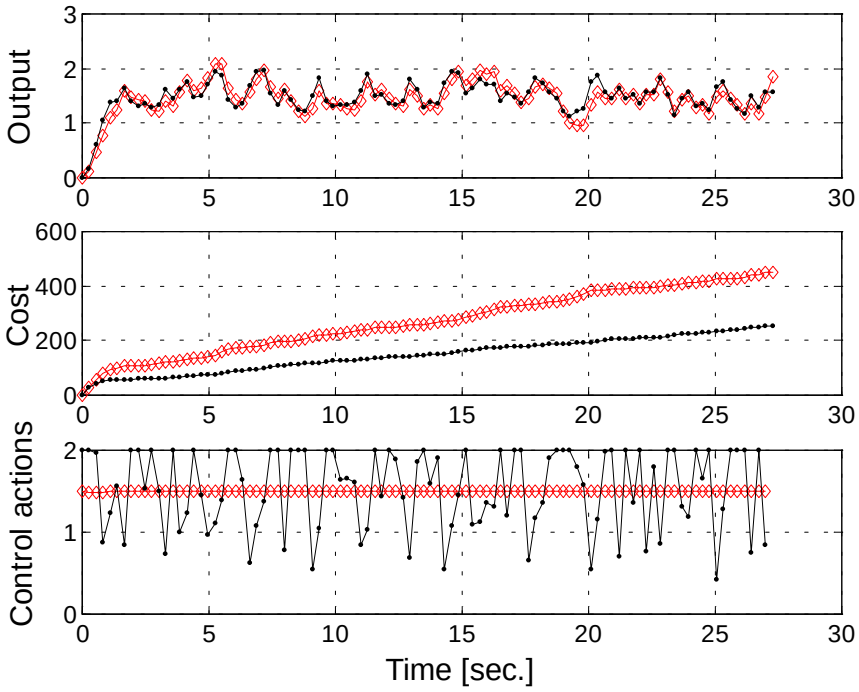

Fig. 7. System evolution using linear controller (◊-) and the approximated optimal controller (.-). Perturbations equaling 20% of state measurement.

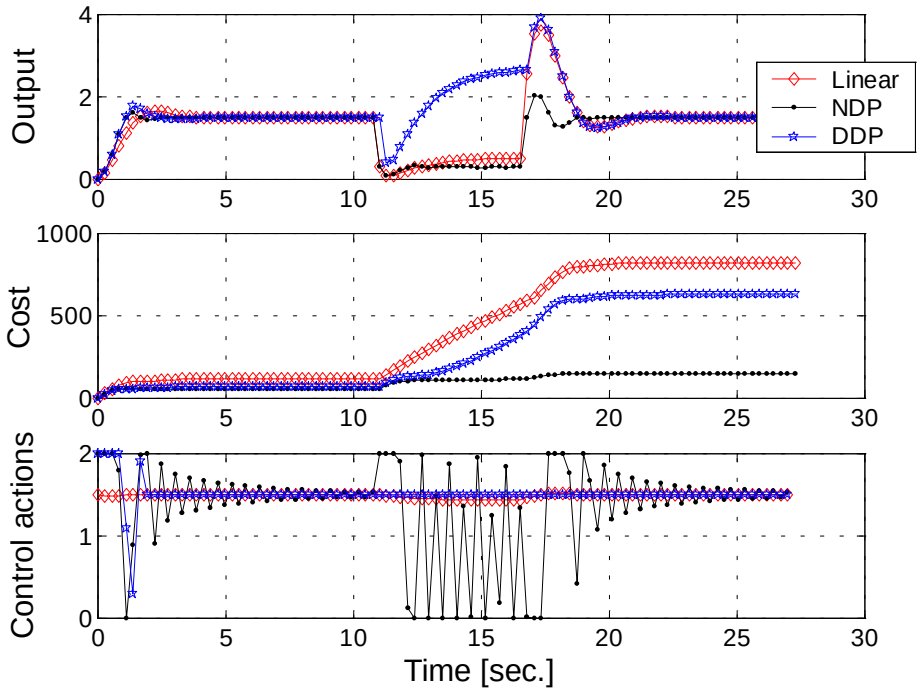

Fig. 8. System evolution using the linear controller (◊_), approximated controller(.-), and by mean of direct dynamic programming (✭-). Perturbations of 80% of state measurements.

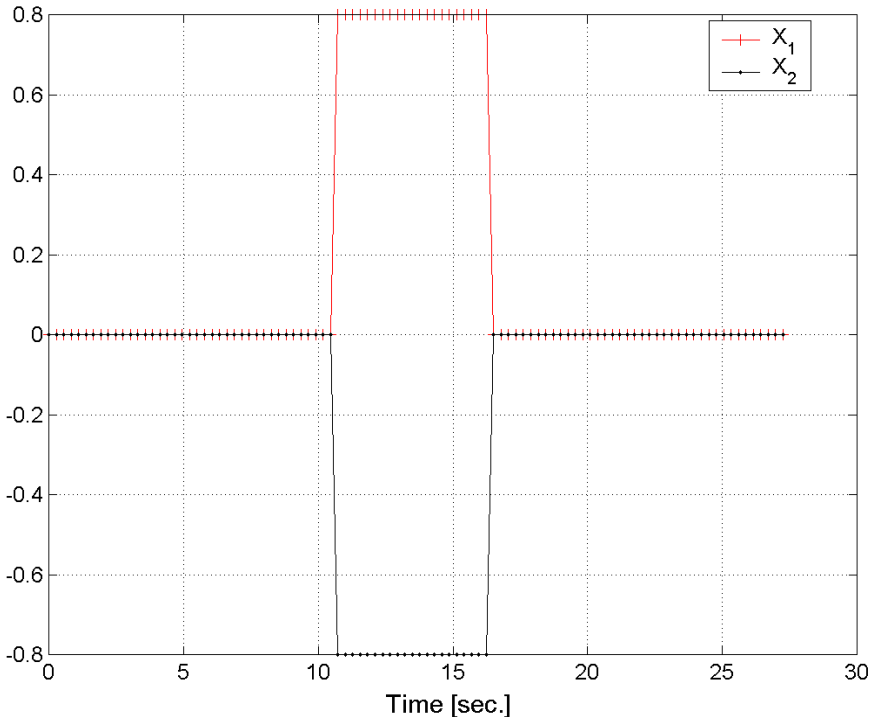

Fig. 9. 80% Perturbations that modifies the evolution of the system.

V. CONCLUSIONS

The optimal control approach offers an attractive methodology for the designer. However, this general solution becomes hindered when the process is nonlinear or the performance index is non quadratic.

The classical theory of optimal control generates a function that renders the optimal policy according to a specified performance index. This function –called $\mu(\cdot)$- has the control actions as image to apply to the system according to the state i where it is staying at. Thus, the search for this function is not always analytically viable. In this sense, the alternative approach that we use here with dynamic programming solves the optimal control problem for any system with any performance index. As a result from this, a table of values is created that links the control actions with the states. To obtain the optimal policy function, the approximate policy iteration method is known as neuro dynamic programming. This technique was detailed and applied in a case example to study the potential use in continuous constrained processes. It is necessary to assure an approximate cost function that guarantees its image for the entire space where the policy gets improved, showing a quick convergence and without further oscillations.

The results obtained -starting from the comparative analysis of controllers- shows the power of the technique, because a robust controller is obtained that is capable of operating under noise and interference conditions.

In some cases, only the performance index can be approached, and a controller is obtained in which a minimization should be carried out as regards the control actions. Other times, this approximate performance index is enough, and it can be considered the problem solution. On the other hand, if the application does not admit a minimization expression, on account of time or technological problems, an extra neural network can be used so that it may behave as the control law. In this work, it was called the action network.

REFERENCES

Bellman R. and S. Dreyfus (1962). *Applied dynamic programming*. Princenton university press.

Bertsekas D. and J. Tsitsiklis (1996). *Neuro-dynamic programming*. Chapters 5 and 6. Athena scientific. MIT.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Pp. 290 292. University Press. Oxford.

Larson and Casti. (1978). *Principles of dynamic programming (Part I)*. Marcel Dekker Inc.

Luus R. (1989). *Optimal Control By Dynamic Programming Using Accesible Grid Points And Region Reduction*. Hung. J. Ind. Chem. 17, 523-543.

Nørgaard M. (1997). *Neural Network Based System Identification Toolbox*. Tech. Report. 97- E-851, Department of Automation, Technical University of Denmark.