Suboptimal Dual Control using Approximation Functions

Mircea Preda
Department of Computer Science
University of Craiova
Al. I. Cuza street, 13, Craiova, 200585
Romania
mirceapreda@central.ucv.ro

Dan Popescu
Department of Automatic Control
University of Craiova
Al. I. Cuza street, 13, Craiova, 200585
Romania
dpopescu@automation.ucv.ro

Abstract – **The adaptive dual controllers are solutions for the control problems that involve uncertainty. Unfortunately, the obtaining of an optimal dual controller is a difficult problem from both analytical and numerical points of view. The paper presents a method to construct a suboptimal controller with dual features, method that involves the Temporal Difference (TD) reinforcement learning algorithms. In order to improve performances in the early stages of the control process, the TD algorithms will work in conjunction with a maximum entropy model of the controlled system. This model will be used to generate simulated experiences for off-line training. The procedure is discussed and evaluated on a well-known nonlinear system.**

I. INTRODUCTION

One way to handle the uncertainties in a control problem is to use an adaptive controller, which to estimate the unknown model of the process. During the control process, a conflict appears between two opposite goals: the information gathering and the control quality. The problem was presented by A. A. Feldbaum [1] who introduced a new type of control system: the dual control system. According with [2], a dual control system is a control system that works in uncertainty conditions and where the control signal has two purposes: to cautiously follow the control goal and to excite the controlled process in order to obtain better information.

Feldbaum proved that the solution to the dual control problem could be obtained using the dynamical programming. The equations obtained by this method are difficult to resolve by analytical or numerical methods. Supplementary, a solution based on the dynamical programming supposes that a perfect model of the system that must be controlled is known and, consequently, such solutions are of little use. We will focus on a set of methods from reinforcement learning domain ([3]), methods that provide approximations of the optimal solutions that can be obtained by dynamical programming. More specifically, we consider the temporal difference (TD) learning, which is a combination of concepts belonging to the Monte Carlo and dynamical programming methods. Like the Monte Carlo methods, the TD learning does not need a model of the environment's dynamic in order to learn and like the dynamical programming it can update estimates based on other estimates without waiting for the end of the experiment.

Our goal is to control a discrete time multi-input single-output nonlinear system described by equation:

$$y(t+1) = f(x(t),u(t)) + e(t+1) \quad (1)$$

where $x(t)$ is the state array at the time step t,

$$x(t) = [y(t),...,y(t-m),u(t-1),...,u(t-p)], \quad (2)$$

$y(t)$ is the output of the system and $u(t)$ is the control array. $e(t)$ represents the noise and let us suppose that we have a Gaussian noise with the average 0 and variance σ^2.

The idea of using approximations to overcome the computational difficulties of the Feldbaum's dual control is well known. Surveys on this topic can be found in [4,5]. We are interested by the nonparametric approach to modeling nonlinear systems and to show that the reinforcement learning can be an alternative to other nonparametric approaches like the Gaussian ones (see [6,7]).

The following sections present the TD(λ) learning algorithm and its possibilities to control a nonlinear system.

II. REINFORCEMENT LEARNING

The reinforcement learning is the name of a set of methods and algorithms for control systems that learn from delayed rewards and automatically improve their behaviors. A typical reinforcement learning problem is described by a quadruple $\{S,U,T,R\}$ where S is a finite set of states, U is a finite set of actions, $T: S \times U \times S \to [0,1]$ is a transition function that specifies the probability to observe a certain state after a specified action is performed in a specified state and R is a reward function, $R: S \times U \to \mathbf{R}$. The policy employed for control is permanently improved by updating a value function $Q: S \times U \to \mathbf{R}$ that approximates the expected long-term return that can be obtained by executing a specified action in a given state and after that following the current policy. If we consider a sequence of n rewards, $r_0, r_1, ..., r_{n-1}$, the expected return will be $\sum_{i=0}^{n-1} \gamma^i r_i$, where $0 < \gamma \leq 1$ is a discount factor that shows how important are the rewards received later during the control process. This parameter provides us a first way to obtain the desired equilibrium between the exploitation and exploration purposes.

III. TILE CODING

In many practical applications, the states and actions involve continuous values or are very large discrete sets. Consequently, the value function cannot be represented in memory as a table and its learning requires some techniques of function approximation. Until recently, even a lot of successful applications were constructed based on function approximation; there were few theoretical guaranties on the performances of these methods. Fortunately, some recent papers like [8] and [9] discuss the

convergence of TD algorithms as SARSA if linear function approximations are used.

We will use a class of linear functions approximators named Cerebellar Model Articulation Controllers or CMACs (see [10,11]) that are sparse coarse coding memories, which model the cerebellar activity. The name of the CMAC approximations when they are used in reinforcement learning problems is tile coding. These approximations follow the pattern: a state-action input pair activates a specific set of memory locations and the arithmetic sum of the contents of these locations represents the estimated value of the function Q.

A CMAC is composed from a set of N partitions of the input space named tilings: $T_1,...,T_N$. Each tiling T_i is composed by N_i features $f_{ij}, j=\overline{1,N_i}$ that cover all input space. Each feature f_{ij} has attached a weight w_{ij} and an eligibility e_{ij}. The meaning of these two values will be presented below.

An arbitrary input pair (x,u) activates a unique element from each tiling, obtaining thus a set of features

$$F(x,u)=\{f_{ij} \mid (x,u)\in f_{ij}, f_{ij}\in T_i\} \tag{3}$$

used to estimate the value $Q(x,u)$ according to the formula:

$$\overline{Q}(x,u)=\sum_{f_{ij}\in F(x,u)} w_{ij}. \tag{4}$$

TD(λ) updates the weights using formula

$$\Delta w_{ij}=\alpha_t(r_{t+1}+\gamma\overline{Q}_{t+1}-\overline{Q}_t)e_{ij}, \forall f_{ij}\in T_i, \forall T_i \tag{5}$$

where $\overline{Q}_t$ and $\overline{Q}_{t+1}$ represent the estimations of the function Q at the time steps t and $t+1$ during control process. Here $\alpha_t\in[0,1]$ is a parameter named learning rate.

The eligibilities e_{ij} represent the contributions of the different features f_{ij} to the value of the input point (x,u). They allow distributing a reward to the all past state-action pairs that contributed to it. The update formula for eligibilities is

$$e_{ij}=\begin{cases}\dfrac{1}{|F(x,u)|}, if\ f_{ij}\in F(x,u)\\ \lambda\gamma e_{ij}, otherwise\end{cases} \tag{6}$$

and we notice that the parameter λ can be used to control the updating process.

One of the advantages of a CMAC estimator is that the estimated values $\overline{Q}(x,u)$ can be computed in a efficient way. The next section will prove the utility of this feature.

IV. SEARCHING FOR THE OPTIMAL ACTION

The approximation $\overline{Q}$ can be used to identify the best action that can be performed for a certain state of the environment. The action is selected by computing the maximum:

$$u^*=\arg\max_{u\in U}\overline{Q}(x,u). \tag{7}$$

Different algorithms can be employed to obtain the optimal action; most of them are dependent on the method used to approximate the function Q. A simple and largely applicable algorithm is (see [10]):

Input: the state x, the range of the actions $[u_{\min},u_{\max}]$ and a step Δu used to sample the set of actions

Output: the best action u^*

Algorithm:

$u^*\leftarrow u_{\min}$
$best\_value\leftarrow\overline{Q}(x,u_{\min})$
for each u **from** $u_{\min}+\Delta u$ **to** $u_{\max}$ **step** Δu
  if $best\_value<\overline{Q}(x,u)$ **then**
    $best\_value\leftarrow\overline{Q}(x,u)$
    $u^*\leftarrow u$
  endif
endfor
return u^*

The ties are broken randomly. After a maximum is found, same algorithm can be applied on a smaller interval in the vicinity of the maximum in order to find a better approximation for the optimal action.

The computational complexity of this algorithm depends on the action space, the size of the step Δu and the costs required to evaluate $\overline{Q}(x,u)$. The reduced costs involved by a CMAC estimator represents an additional recommendation for using it in conjunction with reinforcement learning methods.

V. INDIRECT REINFORCEMENT LEARNING

The TD(λ) does not make full use of the limited available experience. Consequently, a large number of interactions with the environment are necessary to obtain an optimal policy. This is a major drawback for control problems and a solution is to use the so-called indirect reinforcement learning methods, methods that try to establish a model for the environment. The direct and indirect reinforcement learning methods are working together. Between two interactions with the environment, the model will be used to generate simulated experiences, these being used to improve the estimated value function $\overline{Q}$.

A model for the environment is basically a solution to a prediction problem where the values of $p-1$ random variables $X_1,...,X_{p-1}$ are used to predict the value of the random variable X_p. In our case, the input values will be the current state and the action and the predicted value will be the next state. As a framework for this prediction problem we will use the Maximum Entropy Principle ([12]) and in following paragraphs we will build two different models for the environment. For the first one, we will suppose that $X_1,...,X_p$ are continuous, real valued variables.

The maximum entropy based modeling consists in representing the uncertainty regarding the values of the random variable X_p by selecting the maximum entropy probability distributions from all distributions that satisfy the constraints imposed by the a priori information. We will represent the a priori information regarding a probability distribution P as a set of pairs (φ, a) where φ is a function and a a real value. The constraint imposed on the probability distribution P is that $E_P(\varphi) = a$ where E_P denotes the averaged (expected) value regarding the probability law P. The functions φ are called observables and the values a levels.

In the process of identifying of a maximum entropy probability distribution P for the prediction problem $X_1,...,X_{p-1} \rightarrow X_p$ we will use the k degree polynomial regression. This is a method to construct a k degree polynomial $Q_k(X_1,...,X_{p-1})$ starting from a sample of the random variables $X_1,...,X_p$ in such way that, on the sample, the differences between $Q_k(X_1,...,X_{p-1})$ and X_p be minimized relatively to a specified measure.

In the above conditions, as is proved in [12], we have following result:

Proposition 1: Let $X = (X_1,...,X_p)$ be a random vector with values in $\mathbf{R}^p$ and Q_k a polynomial of $p-1$ variables and the degree $\leq k$. Let us consider the observable

$$[x_p - Q_k(x_1,..,x_{p-1})]^2, [Q_k(x_1,..,x_{p-1})]^2, x_i^2, i = 1..p-1. \quad (8)$$

If we choose for X the probability distribution of maximum Shannon entropy that satisfies the constraints defined by the previous observable then the probability distribution for X_p when $X_i = x_i, i = 1...p-1$ is a Gaussian distribution with mean $Q_k(x_1,...,x_{p-1})$ and the variance given by the level of the first observable.

The levels of the observables are constantly updated after each interaction with the environment.

For the second model of the environment, we will suppose that the random variable X_p has discrete real values. The rest of random variables, $X_1,...,X_{p-1}$ keep their real continuous values. We will use following notations: $C = \{c_1,...,c_J\}$ represents the set of allowable values for X_p, Ω denotes a S size training set partitioned in the subsets $\Omega_1,...,\Omega_J$ according with the X_p values, m^j and Σ^j represent the mean and the covariance correlation matrix for the subset $\Omega_j, j = 1...J$ and p_j represents the probability that $X_p = c_j$, probability estimated over Ω, $j = 1...J$. Let us consider the functions $h_j(x_1,...,x_p) = 1$ iff $x_p = c_j$, $j = 1...J$ and the projection functions $x_i(x_1,...,x_p) = x_i$ and $x_{ik}(x_1,...,x_p) = x_i \cdot x_k$, $i,k = 1...p-1$.

In these conditions we can prove the following result:

Proposition 2: If we use for X a probability distribution of maximum Shannon entropy that satisfies the constraints imposed by the a priori information

$$\begin{gathered}(h_j \cdot x_i, p_j \cdot m_i^j), j = 1...J, i = 1...p-1 \\ (h_j \cdot x_{ik}, p_j(\Sigma_{ik}^j + m_i^j \cdot m_k^j)), j = 1...J, i,k = 1...p-1 \\ (h_j, p_j), j = 1...J\end{gathered} \quad (9)$$

then the probability distribution of the random vector $X_1,...,X_{p-1}$ when $X_p = c_j$ is the Gaussian distribution with mean m^j and the covariance matrix Σ^j.

Proof sketch. We must prove that the density of the marginal probability distribution of the random vector $X' = (X_1,...,X_{p-1})$ when $X_p = c_j$ is

$$f_j(x') = \frac{1}{\sqrt{|\Sigma^j|(2\pi)^{\frac{p-1}{2}}}} \exp\{-(x'-m^j)^t \Sigma^{j^{-1}}(x'-m^j)\} \quad (10)$$

If P is a probability distribution of the random vector $X = (X_1,...,X_p)$ then the Shannon entropy for P is:

$$S(P) = -\sum_{j=1..J} p_j \int f_j(x') \log f_j(x') dx' + \sum_{j=1..J} p_j \log(p_j) \quad (11)$$

To maximize $S(P)$, we apply the Lagrange multipliers method and obtain the Lagrangean function:

$$\begin{aligned}L(f_1,...,f_J,\Lambda) = &-\sum_{j=1..J} p_j \int f_j(x') \log f_j(x') dx' \\ &+ \sum_{j=1..J} p_j \log(p_j) - \\ &\sum_{\substack{j=1..J \\ i=1..p-1}} \lambda_{ij}[\sum_{l=1..J} p_l \int h_j(x',c_l) x_i f_l(x') dx' - p_j m_i^j] - \\ &\sum_{\substack{j=1..J \\ i=1..p-1 \\ k=1..p-1}} \lambda_{ikj}[\sum_{l=1..J} p_l \int h_j(x',c_l) x_i x_k f_l(x') dx' - p_j(\Sigma_{ik}^j + m_i^j m_k^j)]\end{aligned}$$

Λ denotes the set of Lagrange multipliers and from the equality $L(f_1,...,f_J,\Lambda) = 0$ we obtain that the probability densities of the random vector X' are defined by (10). □

In order to apply the second type model to our control problem, the output $y(t)$ of the controlled system must be discretized using an aggregation function, which to map the continuous values $y(t)$ into discrete ones $\bar{y}(t)$. The new framework is depicted in Fig. 1.

There are numerous methods to quantize the output space and thus to convert the outputs to discrete values. The empirical methods that divide the space in pieces with equal sizes provide satisfactory results only for a very small set of discretization problems. Consequently, we will use a non uniform method to quantize the output space, method that will assure that the most important parts of this set are accurately represented.

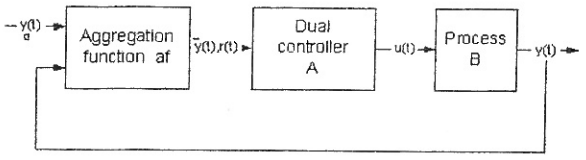

Fig. 1. An adaptive dual controller that uses an aggregation function.

The discretization method uses an unsupervised learning clustering algorithm (see [13]) based on the Euclidian distance. The elements of the continuous output-space will be divided in J groups where the value J will be dynamically updated according to the control's performances and with the memory usage restrictions. We do not need supplementary knowledge about the dynamics of the controlled object and the complexity of the computations is relatively low.

Let us consider $Y \subseteq \mathbf{R}$ and d a metric on $\mathbf{R}$, for example the Euclidian metric. An unsupervised learning clustering method has as purpose to partition a set Y of unlabeled training patterns in J mutual exclusive and exhaustive subsets $Y_1,...,Y_J$ named clusters. In our settings, the set Y will be the output space of our control problem. We consider $C = \{c_1,...,c_J\}$, J evenly distributed centroids from $\mathbf{R}$, which they are used to represent the J clusters and updated according to the following algorithm.

During the control process, we obtain a sequence of outputs $y_t, t \geq 0$. An output y_t will be associated to the closest centroid $c_j \in C$ according with the metric d and implicitly to the Y_j cluster. We obtain an aggregation function $af_t : Y \to \{c_1,...,c_J\}$ defined by $af_t(y_t) = c_j$ where $d(y_t, c_j) = \min_{i=1..J} d(y_t, c_i)$. When an output y_t is classified to a centroid c_j, we adjust c_j by the following rule:

$$c_j = (1-\alpha_j^t)c_j + \alpha_j^t y_t \text{ where } \alpha_j^t = \frac{1}{1+m_j^t}. \quad (12)$$

The mass m_j^t denotes the number of the outputs $y_i \in Y, 0 \leq i < t$ that were classified to c_j. Thus, we obtain a convergent sequence of aggregation functions $af_t, t \geq 0$, one function for each control step.

VI. SIMULATION RESULTS

In order to show that the systems using reinforcement learning are able to perform nonlinear control even if initially we do not have any information about the process, let us consider the system presented in [6]:

$$y(t+1) = f(x(t)) + g(x(t))u(t) + e(t+1) \quad (13)$$

where $x = y(t)$, $f(x) = \sin(x) + \cos(3x)$ and $g(x) = 2$. $e(t)$ has the variance $\sigma^2 = 0.001$. The reference signal $y_d(t)$ takes values in the range $[0,1]$.

The TD(λ) algorithms are influenced by a large number of parameters and their values can be crucial for the success or failure of the control process. The identification of an optimal set of parameters is basically a trial and error process.

The parameter λ is used to share the current reward with the past state-action pairs that contributed to it. In our case, for any value of the reference signal $y_d(t)$, we can find an optimal value of the command $u(t)$ that is independent of the previous states of the system. Consequently, for the parameter λ, the small values are preferable. In our tests we used the value $\lambda = 0$.

Regarding the configuration of the CMAC approximation (tile coding), [14] provides a detailed discussion about its performances and the optimal parameters' settings when the approximation is used together with a continuous action set. A larger value for the number of tilings N involves better generalization capacities and improved performances in the early stages of the training. Later, the approximation quality is affected and the best solution is to use an adaptive algorithm as recommended in [14]. The largest N value used in our tests was 8.

The reward function R and the initial values of the weights w_{ij} attached to the features of the CMAC approximator have a great influence on the convergence speed to an optimal solution. Some choices for these parameters can favor a deep exploration of the state-action space. Other variants can make an early preference to already explored regions. The reward function used in our tests was $R = -(y_d(t) - y(t))^2 + r_0$ where r_0 is a tuning parameter.

The parameter γ is used to find equilibrium between the exploitation and exploration necessities of the control system. Our experiments show that a small value for γ, which favors an early exploitation of the already achieved knowledge, is preferable for our task.

The ε-greedy policy chooses the action $\arg\max_u Q(x,u)$ with the probability $1-\varepsilon$ and, otherwise, it selects a random action according with a uniform distribution. It assures that there is no state-action pair that is ignored and it is another way to balance the exploration and exploitation. Suitable values for ε are between 0.1 and 0.01.

The learning rate α_t satisfies the conditions $\alpha_t \in [0,1]$, $\sum_t \alpha_t = \infty$ and $\sum_t \alpha_t^2 < \infty$ so it will be decreasing over time.

The employed TD(λ) algorithm is SARSA integrated in a DYNA-Q like architecture [3] with a model of the environment that provides planning capabilities. We used the two types of maximum entropy principle based models previously presented with a little gain of performance in the case of the second one. This one involves also a greater computational complexity because it requires a matrix inversion for the covariance matrix. However, due to the fact that covariance matrix is symmetrical and positively defined there are faster algorithms that can be applied for inversion.

With above settings we are placed in the conditions described in [8] that guarantee us the SARSA algorithm will converge with probability 1 to a fixed bounded region.

The Figs. 2 and 3 present the output of the controlled system and the command for a sinusoidal reference signal $y_d(t) = |\sin(0.1 \cdot t)|$. The output signal has oscillations in the early stages of the training process and after that it closely follows the reference signal. The case of a piecewise constant signal, $y_d(t) = 0$ if $(t \; div \; 100) \bmod 2 = 0$ and 1 otherwise, is presented in Fig. 4 and Fig. 5.

VII. CONCLUSIONS

This paper presents an adaptive controller based on a non-parametric model. The controller uses methods widely popular in the reinforcement-learning domain and our intention is to show that these methods can be used in the control theory domain in order to build competitive controllers for nonlinear systems when incomplete and uncertain knowledge are involved.

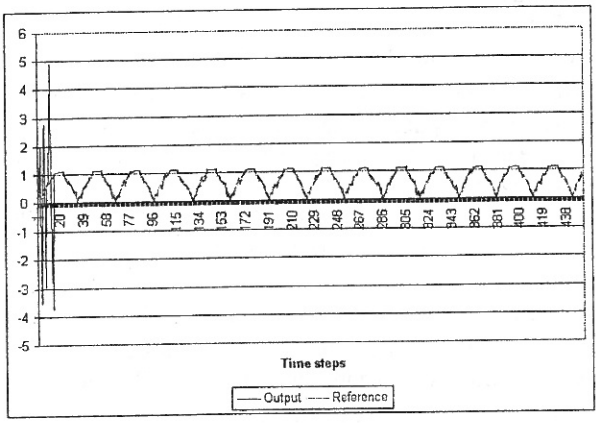

Fig. 2. The output of the adaptive dual controller for the reference signal $y_d(t) = |\sin(0.1 \cdot t)|$. The controller uses the SARSA learning algorithm and a model of the environment based on polynomial regression.

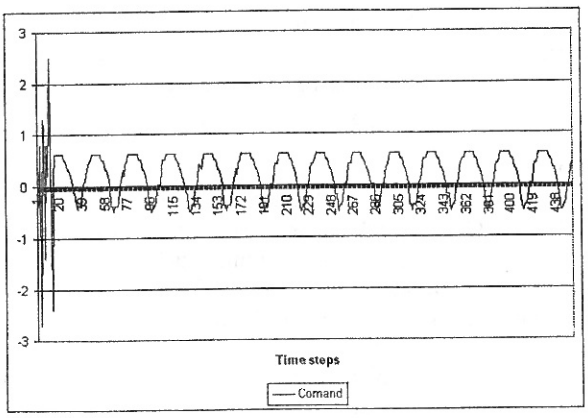

Fig.3. The command of the adaptive dual controller for the reference signal $y_d(t) = |\sin(0.1 \cdot t)|$.

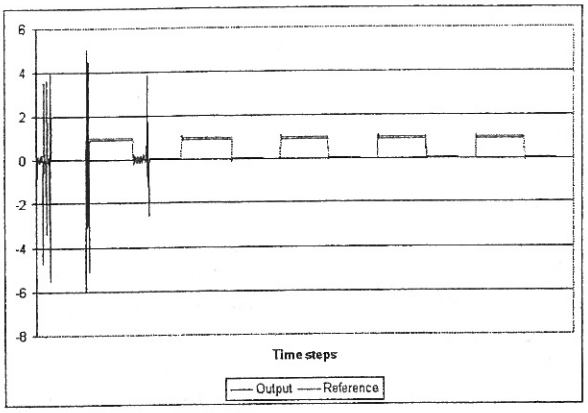

Fig. 4. The output of the adaptive dual controller for a piecewise constant reference signal. The training sequence is presented for 1100 time steps. A model of the environment, which uses covariance matrices, is employed.

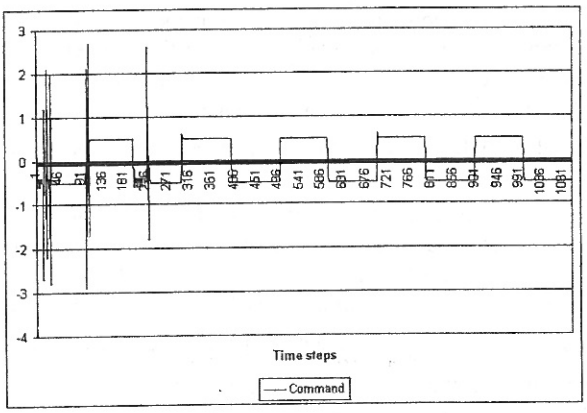

Fig. 5. The command of the adaptive dual controller for a piecewise constant reference signal.

The approach is simple, elegant and does not suppose any knowledge about the controlled system. It also has some drawbacks, most notably, the larger number of interaction required to obtain an optimal solution and the relatively lack of the mathematical guaranties on the performances of the TD(λ) methods when function approximations are involved. The first problem can be partially solved by using an estimated model of the environment and simulated training steps. In some cases, this model can add significant complexity on the computations involved during control. Another problem is that sometimes the recommended sequence of commands is improper, as example the amplitude of some commands or the differences between successive commands are too large. In this case, hybrid architectures must be used, with command validation components that can override the commands recommended by reinforcement learning. Supplementary work is needed to reveal the full potential of the reinforcement learning methods for the control theory.

Further work will go in more directions. First, we want to address the usage of the actor-critic methods (see [15]) in conjunction with models of the environment to control nonlinear systems. These methods are particularly

attractive because the selection of the optimal action based on the value function $Q(s,u)$ involves a very low complexity. Another interesting issue is to investigate the possibilities provided by the dynamical programming applied in conjunction with an estimated model of the environment and the influence of the full backups on the convergence speed to an optimal solution. The dynamical programming approach for the dual control systems was introduced and extensively studied in [1] and we should see if the discussion could be carried in domains where approximations are used. In a more distant future, we are interested in investigating the capacity of the reinforcement learning methods to control continuous time non-linear systems. The reinforcement learning in domains with continuous time received relatively little attention and one of the prominent papers is [16]. Other interesting future topics are the issues regarding the efficiency in order to obtain a real time implementation and the control of the non minimum-phase systems

VIII. REFERENCES

[1] A. A. Feldbaum, *Optimal control systems*, Academic Press, New York, 1965.

[2] N. M. Filatov and H. Unbehauen, "Survey of adaptive dual control methods," *IEEE Proc. Control Theory Appl.*, vol. 147, no. 1, January 2000.

[3] R. S. Sutton, A. Barto, *Reinforcement learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[4] B. Wittenmark, "Adaptive dual control," *In Unbehauen, H. (Ed.): Control Systems, Robotics and Automation, Encyclopedia of Life Support Systems (EOLSS)*, Developed under the auspices of the UNESCO, Eolss Publishers, Oxford, UK, 2002.

[5] B. Wittenmark, "Adaptive dual control methods: An overview," in *5th IFAC symposium on Adaptive Systems in Control and Signal Processing*, 1995, pp. 67-73.

[6] R. Murray-Smith, D. Sbarbaro, "Nonlinear Adaptive Control using Nonparametric Gaussian Process Prior Models," In *15th IFAC World Congress on Automatic Control*, Barcelona, 2002.

[7] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, A. Girard, "Adaptive, Cautious, Predictive Control with Gaussian Process Priors," In *16th IFAC World Congress on Automatic Control*, 2003.

[8] G. J. Gordon, "Reinforcement Learning with Function Approximation Converges to a Region", *NIPS*, 2000.

[9] T. J. Perkins, D. Precup, "A Convergent Form of Approximate Policy Iteration," *NIPS*, 2002.

[10] J. C. Santamaria, R.S. Sutton, A. Ram, "Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces", *Adaptive Behavior*, 6:2, 1998.

[11] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding", *Advances in Neural Information Processing Systems*, 8, 1996, pp. 1038-1044.

[12] C. Robert, *Modéles statistiques pour l'intelligence artificielle*, Masson, Paris, 1991.

[13] A. K. Jain, M. N. Murty, P. J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, No. 3, 1999.

[14] A.A. Sherstov, P. Stone, "On Continuous-Action Q-Learning via Tile Coding Function Approximation," *under review*, 2004.

[15] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, "Policy Gradient Methods for Reinforcement with Function Approximation," *Advances in Neural Information Processing Systems*, 12, 2000, pp. 1057-1063.

[16] K. Doya, "Temporal Difference Learning in Continuous Time and Space," *Advances in Neural Information Processing Systems*, vol. 8, The MIT Press, 1996, pp. 1073-1079.