Three Connectionist Implementations of Dynamic Programming for Optimal Control: A Preliminary Comparative Analysis

Hugues Bersini and Vittorio Gorrini
IRIDIA - CP 194/6- Universite Libre de Bruxelles
50, av. Franklin Roosevelt
1050 Bruxelles - Belgium
bersini@ulb.ac.be gorrini@ulb.ac.be

Abstract

Three optimal control methodologies all relying on neural network for their universal approximation capabilities and on dynamic programming for substituting the time-integral optimization by a succession of time-local optimizations are presented in this paper and applied on the same elementary Rendez-Vous problem. First a simplified version of the Back-Propagation-Through-Time algorithm is presented as the most faithful implementation of dynamic programming when the optimal controller is approximated by neural network (learning by gradient descent) and the process model is available. Relaxing the need for an explicit prior modelling of the process model, Reinforcement Learning (RL) approaches, both for continuous and discrete controllers, are described and tested on the Rendez-Vous problem. The results and the numerous methodological difficulties we met are discussed. The most successful Reinforcement Learning is the connectionist implementation of Q-learning with all Q-values approximated by Radial-Basis-Function networks. However when searching for a continuous optimal controller, the price RL has to pay for the absence of model turns out to be far from negligible in terms of methodological difficulties, lack of robustness, convergence time and quality of the discovered solution.

1. Introduction

Optimal control of a process consists in the discovery of a control law which optimizes a cost given over a certain temporal horizon and thus implicitly accounting for environmental responses delayed in time. When the process is linear and the cost is given in a quadratic form (the usual form), the control law can be mathematically derived, usually as a linear state-feedback time-dependent expression. When the process is not linearizable, instead of laboriously trying to explicitly deriving the control law, a simpler and common approach consists in approximating this state-feedback law by a powerful and easy to tune non-linear approximator. The search for a satisfactory continuous controller (no longer optimal) now has to rely on the use of optimisation algorithm. Neural networks together with backpropagation as the gradient-based optimization algorithm have turned out to be a commonly used method for adaptive non-linear approximation. When gradient-based optimisation algorithms are used, derivatives are required which entails in turn the need for an action-derivable cost and, in the case of model-based control, the knowledge of the process Jacobians.

Back-Propagation-Through-Time (BPTT) [4][5][12][13][17][19][21] algorithm is the temporal extension of backpropagation algorithm when the error to minimize is, like it is indeed the case for optimal control, given over a temporal horizon or delayed in time. A clean way to derive BPTT is by introducing Lagrange multipliers and performing a gradient descent in the synaptic weight spaces [17][5]. This involves a backward recursion necessary to compute the Lagrange multipliers. The resulting algorithm, although described in the literature as working in an easy and satisfactory way (and our results will confirm such assertions), can rapidly consume too much computing time and memory. Complying with the principle of optimality which is at the core of Dynamic Programming (DP) and which justifies the replacement of the exponentially-growing time-integral optimization by a linearly-growing succession of time-local optimization (for each time step), a simplified and more economical version of BPTT is proposed. The strong connection of this simplification with the Bellman-Hamilton-Jacobi functional equation provides a formal support to the proposed simplification. We believe this original simplified BPTT, the first optimal neurocontrol algorithm to be presented in section three, to be the most straightforward way to implement DP principles for optimal neurocontrol, when the control law is continuous, the process Jacobians are known, the learning can be done off-line, and in the case of multistages problems, the time-sequence is given a priori (an extension of BPTT for time-optimal problem is presented in [14]).

Dynamic programming is also intimately related with Reinforcement Learning (RL), a form of trial-and-error learning or learning-by-doing perfectly suited for autonomous agents. As far as we know, reinforcement learning has the same temporally-extended objective as optimal control. However, it tries to maximize this same cost relaxing more and more the type of prior requirement which is necessary to run BPTT. A first important relaxation is to perform the optimal neurocontrol and the approximation of the DP solution on-line, i.e. to perform learning while controlling and controlling while learning. This entails the need, at each time step, of a prediction of the time-integral cost, based, as for DP, on information taken from the next step prediction. Barto et al's adaptive critic [1][2], Werbos' family of Heuristic DP [21][22][23][24] and Sutton's TD [18] are all methods trying on-line to predict the cost, just looking one step ahead, with the hope to first decrease the computational cost by a spread over the entire run but also to speed up the discovery of a satisfactory controller. Since the cost has to be predicted just as a function of the state, a third non-linear approximator (besides the controller and the process model) must be used. Notice that still the easiest way to implement this on-line and forward approximation of DP is in presence of the model, so as to backtrack the predicted cost (just function of the state) through the model, in order to optimize the control action. A further and even stronger relaxation is to perform the same optimal control but now without the process model. The complete architecture is simpler and just composed of two non-linear approximators, one for the controller and one for predicting the cost (but this time the cost must be function of the action and in a way or another implicitly captures some knowledge of the process). Backpropagation is performed through the neural predictor in order to discover a state-feedback control policy which optimizes the cost. Werbos [15][21][22][23][24], Barto [1][2] and Miller and Williams [10] have all developed optimal continuous controller without the need for an explicit prior process model. This second connectionnist implementation of DP aiming at optimizing a continuous neural controller in the absence of the process model will be the object of section four.

Since DP originally is a method for selecting among an exponentially-growing set of possibilities the optimal control policy, one easy way is to replace the continuous controller by a discrete one and to associate to each possible control action an utility-value function (on which to base the selection) whose update follows the recurrent Bellman's equation, but with the next state obtained on-line and not through the model. Sometimes the optimal controller is known to have a bang-bang profile, like in general for time-minimization problem, so that the discretization of the controller turns out to be a favorable move. For control in discrete world (discrete states and discrete control actions), Q-learning [20] has become a powerful and popular extension of DP which can learn to optimally select an optimal control of the process, on-line, and without the need for any prior model.

Connectionist implementation of Q-learning is necessary when the discretisation of the problem space is responsible for an explosive number of state-action pairs and an unmanageable size of the search space. The Q-values are then estimated by means of neural networks, preferentially one network by action for avoiding interference between the actions [9]. This last connectionist implementation of DP for optimally selecting the discrete control law, still on-line and in absence of the model, will be discussed in the last section

One of the most elementary optimal control application is the “Rendez-Vous” problem. It is a one-dimensional problem in which an immaterial point must reach a certain destination with a zero velocity, in a finite amount of time steps and minimizing the consumption given by the integral of the successive accelerations. Since the process is linear, the optimal solution can easily be derived and compared with the three DP approximations. In all sections, each dedicated to one of the three methods, the way to implement the neurocontroller and the way it learns to optimally resolve the “Rendez-Vous” problem will be presented, compared with the optimal solution and discussed.

2. The Rendez-Vous Problem

The "Rendez-Vous" problem illustrated in figure 1 is one of the simplest optimal control applications characterized by a delayed feedback and a time-integral cost. It is a one-dimensional problem in which an immaterial point (with coordinate y) leaving a departure point y_0 with an initial velocity v_0 must reach the axis origin (the place for "Rendez-Vous") with a zero velocity, in a finite amount of time step kf and minimizing the consumption given by the integral of the successive accelerations a(k).

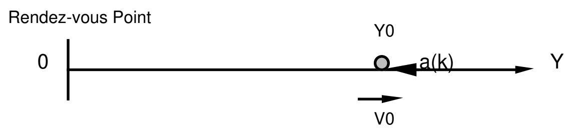

Fig. 1: The "Rendez-vous" Problem

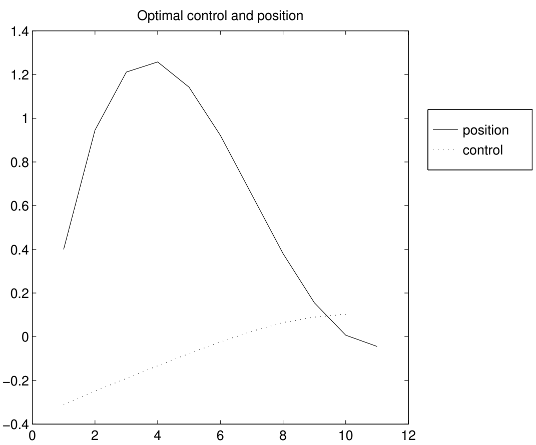

Fig.2: The optimal control law and the optimal position as a function of time for the Rendez-Vous problem.

The problem is defined as follows: k: current iteration; y(k): the point position; v(k): the point velocity; a(k): point acceleration (and the actual control); x(k): state vector={y(k), v(k)}; x(0): initial state vector and kf: final iteration. The cost to be minimized is given by: $J = \frac{1}{2}\vec{x}(k_f)^T \vec{x}(k_f) + \frac{1}{2} c \sum_{0}^{kf-1} a(k)^2$ with c weighting the consumption saving requirement. The process equations in their discrete-time form are linear: $y(k) = y(k-1) + v(k-1)\Delta t + a(k-1)\Delta t^2/2$ and $v(k) = v(k-1)+a(k-1)\Delta t$. When needed, the two Jacobians ($\partial y/\partial a$ and $\partial v/\partial a$) can be obtained through the analytical equations. The optimal solution is easy to analytically derive [8] and, for the particular case c=1 (we will keep this value in the rest of the paper), is given by: u(t)=-f(t)v(t)-g(t)y(t) with $f(t)=\frac{1}{h(t_f-t)}(1+(t_f-t)^2+\frac{1}{3}(t_f-t)^3)$,

$g(t)=\frac{1}{h(t_f-t)}((t_f-t)+\frac{1}{2}(t_f-t)^2)$ and $h(t)=(1+\frac{1}{3}t^3)(1+t)-\frac{1}{4}t^4$

The profile of the optimal control law and of the optimal position y as a function of time are given in fig.2 for the initial condition: y_0= 0.4 and v_0= 0.7. Also in the rest of paper, for simplicity, the initial conditions are kept fixed.

3. The Simplified BPTT

Let's consider the optimal control problem in its continuous form. The process equation is given by: $\dot{x}=f(x,u)$. In the following, the process will be considered as Single-Input-Single-Output. Like for a reinforcement learning problem, the cost to minimize is given by

$$J=\int_{ti}^{tf} r(x,u)dt \qquad 3.1$$

r(x,u) is the immediate reinforcement received at each time step and t_i and t_f are the initial and final instants of the temporal horizon over which the control is exerted. We suppose the control to be given by a neural net parametrized by the weight matrix W: u=NN(x;W).

We know, by applying the principle of optimality which is at the basis of DP, that the optimal cost $J^*(x,t)$ must satisfy: $J^*(x,t)=\min_u\left\{r(x,u)dt + J^*(x+dx,t+dt)\right\}$. This easily leads to the well known Bellman-Hamilton-Jacobi equation which can be seen as the continuous counterpart of the Bellman's recurrent equation at the core of DP:

$$\min_u\left\{r(x,u)+\frac{\partial J^*(x,t)}{\partial x}f(x,u)+\frac{\partial J^*(x,t)}{\partial t}\right\}=0 \qquad 3.2$$

By assuming u*(x) to be the minimal control law, it follows logically that the expression in bracket must also be minimal with respect to x so that we finally have:

$$\frac{\partial r(x,u^*)}{\partial u}+\frac{\partial J^*(x,t)}{\partial x}\frac{\partial f(x,u^*)}{\partial u}=0 \qquad 3.3$$

$$\frac{\partial r(x,u^*)}{\partial x}+\frac{\partial^2 J^*(x,t)}{\partial x^2}f(x,u^*)+\frac{\partial J^*(x,t)}{\partial x}\frac{\partial f(x,u^*)}{\partial x}+\frac{\partial}{\partial x}\frac{\partial J^*(x,t)}{\partial t}=0 \qquad 3.4$$

The corresponding Lagrange function is introduced in order to find the optimal control sequence u*(x):

$$L=\int_{ti}^{tf} r(x,u)+\lambda(t)\left[f(x,u)-\dot{x}\right]+\mu(t)\left[u-NN(x;W)\right]dt \qquad 3.5$$

where $\lambda(t)$ and $\mu(t)$ are the time functions of the Lagrange multipliers. The different variables can be considered as independent so that, among others, the necessary conditions of optimality with respect to u and x are:

$$\frac{\partial r(x,u)}{\partial u}+\lambda(t)\frac{\partial f(x,u)}{\partial u}+\mu(t)=0 \qquad 3.6$$

$$\frac{\partial r(x,u)}{\partial x}+\lambda(t)\frac{\partial f}{\partial x}-\mu(t)\frac{\partial NN(x;W)}{\partial x}+\dot{\lambda}(t)=0 \qquad 3.7$$

Equation 3.7 is a direct result of applying variational methods to the Lagrangian which is functionally dependent upon x(t) and $\dot{x}(t)$. Finally a gradient descent allows the discovery of a local solution for the unknown optimal weights: $\Delta W=\eta\int_{ti}^{tf}\mu(t)\frac{\partial NN(x;W)}{\partial W}$ 3.8

Let's now make the same substitution as the one which is made in [8] in order to reveal the connections between the Pontryagin principle, which originates from variational methods, and the Bellman-Hamilton-Jacobi equation which originates from DP : $\lambda(t) = \frac{\partial J^*(x,t)}{\partial x}$. By making this substitution, equation 3.4 becomes equivalent to equation 3.7 if the expression multiplied by μ is suppressed. This simplification can be done since indeed by 3.3, μ=0 for the optimal control policy. As a result, the new simplified version of BPTT algorithm, now in a time-discrete form, becomes:

$$\mu_k = -H_k u(k) + \left(\frac{\partial G_{k+1}}{\partial u(k)}\right)^T \lambda_k \qquad 3.9$$

$$\lambda_{k-1} = -F_k x(k) + \left(\frac{\partial G_{k+1}}{\partial x(k)}\right)^T \lambda_k \qquad 3.10$$

The weights are again updated by gradient descent: $\Delta w = -\eta \frac{\partial L}{\partial w} = \eta \sum_{n=0}^{H-1} \left(\frac{\partial N_k}{\partial w}\right)^T \mu_k$.

Besides this theoretical justification, the intuitive reason that allows us to simplify the BPTT algorithm results from the dynamic programming principle of optimality which tells us that, for certain applications, a time-integral optimization procedure can be substituted by a succession of time-local minimization procedures. It is no longer necessary to try to optimize the output of the nth-step controller with respect to the parameters of all previous controllers - a complete temporal separability can be preserved. In comparison with the previous original algorithm, it is indeed the link between the different time step which has been canceled: the derivative of the output of controller k with respect to the output of controller k-1. Therefore the successive optimization can be kept separated and made independent. Practically, the parameter μ_k, although left here for sake of clarity, is actually no longer necessary since its recursive role has been eliminated.

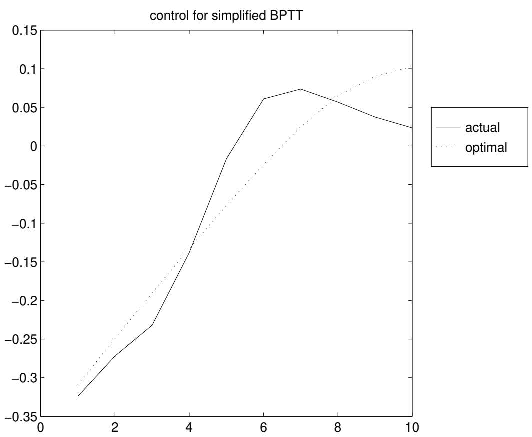

Fig.3 a: The simplified BPTT control law together with the optimal one

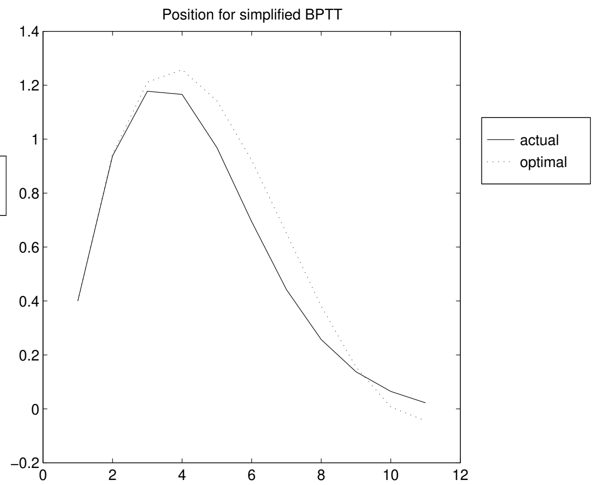

Fig. 3b: The simplified BPTT position profile together with the optimal one

For the Rendez-Vous problem, the acceleration is the only output of a Multilayer Perceptron which receives as inputs the position and velocity and contains two layers of 20 and 10 hidden units. All the Jacobians are derived from the analytical equations. Besides all the Jacobians can be approximated by their sign following a further simplification strategy already discussed in [17]. In their parking of the truck backer-upper, Nguyen and Widrow used a second neural net for emulating the process and deriving the Jacobians [12]. In [5], we have shown that BPTT, when taken in its simplified version, works better and faster than the original one so that, in this paper, we will restrict the use of BPTT to its simplified version. A last important assumption of our problem is that we consider the Rendez-Vous to occur after 10 time steps. Results are indicated in fig.3. This is a solution obtained at the end of 100 learning epochs. Finally, since in principle BPTT assumes from the beginning a fixed number of time steps (10 in our case), an aspect that might be perceived as a limitation, we have proposed in [5] (based on Plumer's article [14]) an extension of BPTT for time-optimal control problems.

4. A First RL for a Continuous Neurocontroller

One important argument justifying the use of RL strategies is the missing prior knowledge of the process model, whatever in an analytical or neuronal form. Once supplied with this model, it is hard to understand why still using any type of RL instead of BPTT in its simple form since, according to the previous chapter, this later is a straightforward implementation of DP. Werbos has proposed a family of connectionist implementations of DP, some using a process model others not. In [15], it is argued that, even in presence of the model, approximate form of DP such as HDP performs better than the original BPTT, above all when the process is subject to a lot of random disturbances and the model hardly precise. However, not enough experiments have been conducted so far to generalize this finding and we still maintain our position (experimentally supported) that RL is an important alternative method for optimal control only in cases where the process model is lacking. The first RL approach we tried is very reminiscent of similar works by Jordan and Jacobs [7], Barto [1][2], Werbos [22]. The control architecture is based on two neural nets: the first one - the control net - receives in input the current state and gives in output the acceleration to apply a(y,v), the second one - the cost predictor net - receives in input the current state and the acceleration applied and gives in output a prediction of the cost Q(y,v,a). The predictor net learns to minimize a quadratic error given by: $E=[(r(t)+QNEXT) - Q(x(t),v(t),a(t))]^2$
where $r = a(t)^2$ when $t<tf$ and $r=a(tf-1)^2 + x(tf)^2+v(tf)^2$ for $t = tf$ and where QNEXT is the output of the predictor net for input x(t+1),v(t+1) and a(t+1).

We decided to first learn the cost on the basis of random series of action (given by the control net for random weights) and, once a good error achieved for the prediction, to then optimize the control so as to minimize the cost (backpropagating through the predictor net to then tuning the weights of the control net by backpropagation). Both networks were MLP with variable number of hidden units. The results were not satisfactory at all since the predictor net never succeeded to converge to a satisfactory cost prediction. If it could learn one series of 10 accelerations, given by a random control net, just an additional series raised enormous difficulties. We never succeeded to perform a satisfactory cost prediction for 20 state-action pairs. These poor results seem to agree with more theoretical papers discussing the difficulty one can encounter when trying to approximate Q-values or utility-value functions. First the mapping to learn is extremely rugged since two slightly different actions (taken from two different series) can lead to very different cost while two successive actions (taken from the same series) can lead to similar cost (at least for the final position). Secondly, the cost to estimate is in turn target for other estimates so that the actual and the desired output are varying together with likely effect to lead to learning instabilities.

5. A Second RL for a Discrete Neurocontroller

A more popular RL approach, and which has been shown to work in a satisfactory way in the literature [9] [18], is still to rely on neural nets for approximating the Q-values but now discretizing the actions in a finite number N (i=1..N) and using as many neural nets as actions. Each neural net approximates $Q_i(y,v)$ for one particular action a_i. A classical alternative is to keep the action as one input of only one net (but still in a discrete form so that no control net is necessary). However using as many Q-values neural approximator as actions helps to avoid the interference between the actions, since what is learned for one action will not interfere with what is learned for another one.

This approach is very closely related with Watkin's Q-learning [20] and consequently shows the same sensitivity to the exploration strategy, even more due to the simultaneous learning of the desired and actual output. The strategy that seems to work the best on this specific application (called CHOICE(Q)) is, at each time step, with probability p to select an action on the basis of a probabilistic Bolzman distribution and with probability (1-p) to take the best action (max Q). p is linearly decreasing between 0 and tmax (this is akin to an annealing strategy) so that after tmax time step, always the action with the maximum Q-value is selected. The complete algorithm runs as follows:

For t=1 to tf-1

> Calculate $Q_i=Q_i(x(t),v(t))$ for each network i
>
> $a_j(t)$=action given by CHOICE(Q) ; Q_j associated with a_j = QACT
>
> Execute the acceleration $a_j(t)$: $y(t+1)$ and $v(t+1) = f(y(t),v(t), a_j(t))$
>
> Find $QNEXT=Max_i(Q_i(y(t+1),v(t+1))$
>
> If t < tf
>
> In the ACT or jth network make one step backprop to minimize $E=[(-a_j(t)^2+QNEXT) - QACT]^2$
>
> Else
>
> In the ACT or jth network make one step backprop to minimize $E=(-a_j(tf-1)^2-y(tf)^2-v(tf)^2)$

Experimentally, we observed that this algorithm shows extreme sensitivity to various aspects: 1) The number and the nature of the discrete actions. In general, increasing the number of actions makes the learning more unlikely to converge 2) The nature of the Q-values neural approximators. We never succeeded to obtain convergence with MLP neural net whatever size we fixed. We had to resort to local type of neural approximator, here precisely Radial Basis Function networks (RBF) 3) The nature of the RBF. The way the Gaussians are distributed in the space is a important factor of success (also discussed in [16]) 4) The nature of the exploration strategy is capital and only practice drives us to use the very ad hoc strategy presented above, with no strong justification (an ad hoc exploration strategy characterizes in the same way the work presented in [10]) 5) The number of time steps. Surprisingly the solution obtained with only four or six time steps instead of ten was better (for instance the point could reach the 0 final position) even if, in principle, it should be worst.

Results are indicated in fig.4, 5 and 6. Fig.4 shows how the error decreases during learning. The exploration was performed during 2000 epochs (= tmax) of learning (this explains the strong variations present in the first part of the curve) and the Q-values approximators converged after approximately 2500 learning epochs. In fig.5, the three control strategies are shown so that it is possible to compare the optimal one with the simplified BPTT and the RL ones. It is easy to see that RL is approximately attempting a discrete fitting of the optimal curve. However, looking at fig.5, which allows a comparison of the performances with respect to the position, the RL is the worst controller since it reaches a final position of 0.6 instead of 0. This could be due to the discretisation of actions (making impossible a better solution) but further analysis convinced us that, even given this discretisation, a much better solution remained possible. The only possible explanation of this unsatisfactory performance is that RL was trapped in a local minimum. This presence of spurious local minima is nothing surprising on account of the importance of the exploration strategy, the simultaneous moving of the desired and actual output, the presentation of the data depending on the on-line control strategy, and the relying on a gradient-based type of learning.

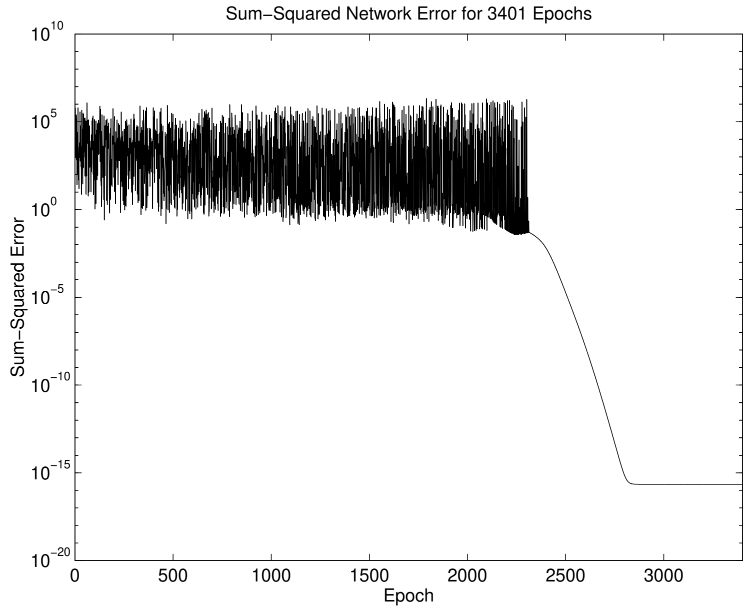

Fig. 4 The Q-value neural networks convergence of error during learning

In a nutshell, and comparing the two solutions obtained by the simplified BPTT and RL, it is clear that BPTT is much more easy to run (i.e. very robust to the nature of the neural approximators and to the nature of the learning algorithm) and leads to better solutions than RL. So far this seems to be the price to pay for the absence of the process model which remains a welcome guide for the running of DP.

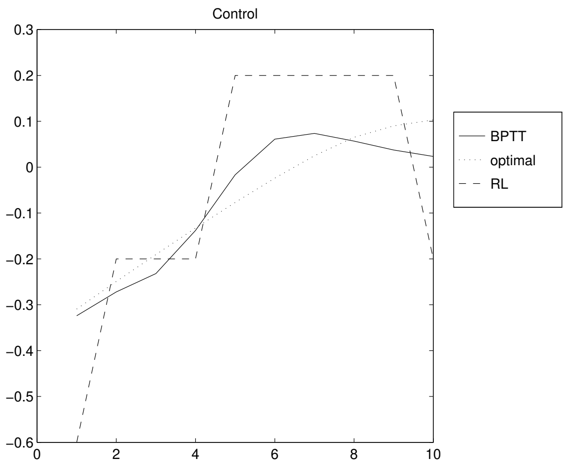

Fig.5 Comparing the three control strategies

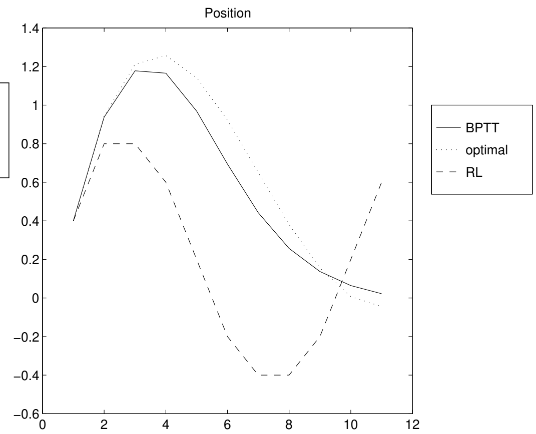

Fig.6 Comparing the three position profiles

6. Conclusions

In this work, we have compared three optimal control strategies relying on the use of neural net for their non-linear universal approximation capacity and complying with DP time separability principle. The first one is a simplification of the Back-Propagation-Through-Time algorithm. We first have shown that this simplification makes BPTT to be the most faithful and direct neurocontrol implementation of DP principles. However it can only work provided a lot of requirements are satisfied: the process Jacobians are known, the cost is action-derivable, the backward time-sequence is given a priori (for instance, how many time steps), and the control is performed off-line. Some of these requirements have already been relaxed. For instance, it has been shown by Saerens, Renders and Bersini [17] that the backward recursive computation that allows the computation of the gradient can be transformed into a forward algorithm which can be carried out on-line, assuming that the adaptation rate of the weights is slow. Besides, despite the non-derivability of time when it appears into the cost to minimize, some extension of BPTT for time-minimization problems, which again compares very favorably with RL (like for instance for the mountain-car application [11][18]), has been proposed in [5].

We believe all forms of RL to be promising and appealing alternatives mainly when the optimal control has to be performed without knowing anything a priori about the process. Accordingly, all forms of RL and DP approximation which rely on a process model is likely to raise more skepticism, since the prior presence of the model allows in many cases the strict application of DP (it must be clear that this skepticism does not involve methods like Dyna where the model is learned while acting and not given from the very beginning). The only RL approach which run successfully for the Rendez-Vous problem needed as many RBF Q-value approximators as actions. In comparison with the simplified BPTT, this RL approach shows an extreme sensitivity to a lot of aspects rendering thus its correct behaving more a question of art, feeling and practice than anything else. BPTT shows much less degree of freedom to play with and accordingly much more robustness to architectural and algorithmic options. Also simple BPTT performs faster (100 learning epochs vs 2500 learning epochs) and better (the final cost is lower than for RL). Similar results are presented in [5] for the time-minimization mountain-car problem despite the fact that, for this problem, the optimal control shows a bang-bang profile. The better performance obtained by the simplified BPTT can be explained by two main reasons: the availability of the process model (even if this model can itself be the result of a neural approximation) and the backward and off-line form of the optimization.

A fundamental point then to be clarified before any use of RL strategy is whether or not a model can be available (even in an approximated form like a neural form). If it is obvious that such a model is very unlikely for real robot acting in the real world (since according to some roboticians, the world is its only and best model), this unavailability is much more questionable in a lot of process control applications for which RL is frequently applied with apparent success like the cart-pole, the acrobot, the mountain-car, robotic arms, peg-in-hole, the bioreactors and the aircraft autolanding problem.

References

[1] Barto A., Sutton R. & Anderson C., 1983, "Neuron-like adaptive elements that can solve difficult learning control problems". *IEEE Transactions on Systems, Man and Cybernetics*, 13 (5), pp. 834-846.

[2] Barto, A.G. 1992. Reinforcement Learning and Adaptive Critic Methods. *Handbook of Intelligent Control*, D.A.White and D.A.Sofge (Eds.), Van Nostrand Reinhold: New York, 469–491.

[3] Barto, A.G., Bradtke, S.J. and S.P. Singh, 1993. Learning to Act using Real-Time Dynamic Programming. *CMPSCI Technical Report 93-02*. - Department of Computer Science - University of Massachusetts of Amherst

[4] Beaufays F., Abdel-Magid Y. & Widrow B., 1994, Application of neural networks to load-frequency control in power systems. *Neural Networks*, 7 (1), pp. 183-194.

[5] Bersini, H. 1996. A simplification of the back-propagation-through-time algorithm for neurocontrol - in revision for *IEEE Trans. on Neural Net.* - a shorter version has appeared in the *Proceedings of the 1995 World Congress on Neural Networks* - vol.2 p.428-432

[6] Jordan M.I. & Rumelhart D., 1991, Internal world models and supervised learning. *Proceedings of the eighth International Workshop on Machine Learning*, Illinois, pp. 70-74.

[7] Jordan, M.I. and R.A. Jacobs 1990. Learning to control an usntable system with forward modelling; In D.S. Touretzky, editor, in Porceedings of NIPS 2 - Morgan Kaufman, San Mateo, CA.

[8] Kirk, D.E. 1970 *Optimal Control Theory - An Introduction*. - Prentice Hall Networks Series - Robert W. Newcomb, Editor - Inc. Englewoods Cliffs, New Jersey.

[9] Lin L-J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, **8**, 3-4, 293–322.

[10] Miller, S. and R.J. Williams, 1995. Temporal Difference Learning: A Chemical Process Control Application - in *Applications of Neural Networks* - Murray (Ed.) - Kluwer Academic Publishers

[11] Moore, A. W, 1992, Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In Birnbaum, L.A. & Collins, G.C. (Eds.), *Machine Learning: Proceedings of the eighth internatioal workshop*, pp. 333-337, San Mateo, CA. Morgan Kaufmann.

[12] Nguyen D. & Widrow B., 1990, Neural networks for self-learning control systems.*IEEE Control Systems Magazine*, 10 (3), pp. 18-23.

[13] Parisini T. & Zoppoli R., 1991, Backpropagation for n-stage optimal control problems. *Proceedings of the IEEE International Joint Conference on Neural Networks*, Singapore, pp. 1518-1529.

[14] Plumer E., 1993, Time-optimal terminal control using neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, pp. 1926-1931.

[15] Prokhorov, D.V., Santiago, R.A. and D.C. Wunsch II, 1995. Adaptive Critic Designs: A Case Study for Neurocontrol - in *Neural Networks*, Vol. 8, No 9, pp. 1367-1372.

[16] Sabes, P. 1993. Approximating Q-values with Basis Function Representations - in *Proceedings of the 1993 Connectionnist Models Summer School*, Hillsade, NJ - Mozer, M., Smolensky, P., Touretzky, D. , Elamn, J. and A. Weigend (eds.).

[17] Saerens M., Renders J-M. & H. Bersini, 1995, Neural Controllers based on Backpropagation Algorithm - in *IEEE Press Book on Intelligent Control Systems* -- M. Gupta & N. Sinha (Eds) IEEE Press - pp. 292-326.

[18] Sutton, R., 1995, Reinforcement Learning with Replacing Eligibility Traces. To appear in *Machine Learning*.

[19] Suykens J., De Moor B. & Vandewalle J., 1994, Static and dynamic stabilizing neural controllers, applicable to transitions between equilibrium points. In *Neural Networks*, 7 (5) - pp. 819-831.

[20] Watkins C. J. C. H. and Dayan P., 1992. Technical Note: Q-learning. *Machine Learning*, **8**, 3-4, 279–292.

[21] Werbos P., 1990, Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78 (10), pp. 1550-1560.

[22] Werbos, P. 1992 Approximate Dynamic Programming for real-time control and neural modelling. In D. White & D. Sofge (Eds.) - Handbook of intelligent control. New York: Van Nostrand.

[23] Werbos P., 1993. *The Roots of BackPropagagtion: From Ordered Derivatives to Neural Networks and Political Forecasting*, Wiley.

[24] Werbos, P., 1995 Optimal Neurocontrol: Practical Benefits, New Results and Biological Evidence - In *Proceedings of the 1995 World Congress on Neural Networks* - Washington DC - vol. II - pp. 318-326.

[25] White D. A. & Sofge D. (editors), 1992, *Handbook of intelligent control. Neural, fuzzy, and adaptive approaches*. Van Nostrand Reinhold.