Pergamon

Neural Networks, Vol. 8, No. 9, pp. 1367–1372, 1995

0893-6080/95 $9.50+.00

0893-6080(95)00042-9

NEURAL NETWORKS LETTER

Adaptive Critic Designs: A Case Study for Neurocontrol

DANIL V. PROKHOROV,[1] ROBERTO A. SANTIAGO[2] AND DONALD C. WUNSCH II[1]

[1] Texas Tech University and [2] BehavHeuristics, Inc.

Abstract—*For the first time, different adaptive critic designs (ACDs), a conventional proportional integral derivative (PID) regulator and backpropagation of utility are compared for the same control problem—automatic aircraft landing. The original problem proved to contain little challenge since various conventional and neural network techniques had solved it very well. After the problem had been made much more difficult by a change of parameters, increasingly better performance was observed by going from the simplest ACD to more sophisticated designs, dual heuristic programming having been ranked best of all. This case study is of use in general intelligent control problems for it provides an example of the capabilities of different adaptive critic designs.*

Keywords—Adaptive critic, Heuristic dynamic programming, Aircraft autolanding, Neurocontrol, Neural networks for control and optimization.

1. ADAPTIVE CRITIC DESIGNS

ACDs are tools to solve difficult optimization problems (Werbos, 1990). They include heuristic dynamic programming (HDP), dual heuristic programming (DHP) and globalized DHP (GDHP) as well as their action dependent forms, further referred to as having prefix AD. (We will use the four abbreviations: HDP, DHP, GDHP, and ACD, plus the prefix AD on the first two, for the remainder of this paper.) All designs attempt to approximate dynamic programming. This gives ACDs their power to work effectively in a noisy, nonlinear environment while making minimal assumptions regarding the nature of that environment.

A typical ACD consists of three neural nets—critic, action and model. The critic net outputs a function J which is an approximation of the secondary utility function J^* of dynamic programming (as in HDP, ADHDP, and GDHP) or the derivatives of J^* with respect to the state variables R (as in DHP and ADDHP). This function has to be approximated because of intractable computational complexity in finding it. The goal is to maximise or minimize J in the immediate future (in the next time step) which produces an optimum U, the primary utility function, in the long run. This goal is accomplished by the action network which outputs a control vector A that optimizes J.

An adaptation of the action network is based on derivatives of J with respect to the components of the vector A. A straightforward way to get those derivatives is to use backpropagation through the other nets of the design. The use of the backpropagation algorithm to find the derivatives of J is the most crucial distinction between HDP and the well-known adaptive critic element (Barto et al., 1983).

If the environment or an object to be controlled does not allow backpropagation through itself, the model network is used. It mediates propagation of the derivatives of J from the critic network to the action network in HDP and, additionally, provides proper targets for the critic network's adaptation in DHP. The model network forecasts the next states $R(t+1)$, $R(t+2), \ldots, R(t+N)$ of the environment. These forecasts, rather than actual states, are fed into the critic provided that the model is sufficiently accurate. Whenever making one-step-ahead predic-

Acknowledgements: The authors[1] gratefully acknowledge support from the Texas Tech Center for Applied Automation and Research Grant entitled "Applied Computational Intelligence Laboratory", and the National Science Foundation Neuroengineering Program Grant No. ECS-9413120. DHP and related designs are patents pending 1994 by BehavHeuristics, Inc. and Paul Werbos. All rights are reserved. The authors also would like to thank Paul Werbos and Ken Otwell for their support and assistance. Furthermore, we are pleased to acknowledge the work of Chuck Jorgensen et al. in the original development of the autolander problem statement (Jorgensen and Schley, 1990).

Requests for reprints should be sent to Danil Prokhorov, Box 43102, Department of Electrical Engineering, Texas Tech University, Lubbock, TX 79409-3102, USA.

tions is enough, one may simply wait till the next state actually occurs to feed the critic. This technique has been used in all our experiments.

It should be noted that ADHDP does not use an explicit model of the environment which implies extra architectural complexity and more difficult training of the critic net representing such a combined critic-model block.

Only ADHDP and HDP have reached the level of successful commercial applications as seen in Franklin and White (1992). Surprisingly, despite advantageous performance, particularly for multiple input multiple output (MIMO) systems, DHP has not yet left the experimental stage (Santiago & Werbos, 1994). GDHP, more powerful than DHP, is currently being developed at BehavHeuristics, Inc., and it is not discussed here. A detailed description of ACDs can be found in Werbos (1990, 1992).

2. THE PROBLEM

The autolanding problem we studied is described in Anderson and Miller (1990). It is one of the challenging control problems proposed in Jorgensen and Schley (1990) as interesting benchmarks for neurocontrol. It deals with a linearized two-dimensional model of a commercial aircraft which is to be landed in a specified touchdown region of a runway within limits. The aircraft is subject to wind disturbances. Wind disturbances have two components: wind shear and turbulent wind gusts (horizontal and vertical). Turbulent wind gusts are modeled by a stationary stochastic process. The aircraft model includes an autothrottle and a pitch autopilot. The autothrottle keeps the speed of the aircraft constant. The aircraft descent is controlled by specifying the desired elevator angle to the pitch autopilot.

Inputs to an autolander may include actual values of the altitude and the vertical speed and their desired values given by an Instrument Landing System. This system helps to land the aircraft by specifying an optimal glide path.

3. EXPERIMENTS AND RESULTS

Figure 1 shows ADHDP for aircraft autolanding. The pathways of backpropagation as well as all signals used to adapt the critic network and the action network are shown by dashed lines in all the figures. Both the critic network and the action network are simple feedforward networks, each having one hidden layer of bipolar sigmoid nodes. Their architectures are $3 \times 12 \times 1$ and $4 \times 4 \times 1$, respectively. Outputs of both nets also have bipolar sigmoid neurons whereas inputs are each scaled differently. A scaled difference between the actual altitude h and the desired altitude h_c (from the Instrument Landing System) and that between the vertical speed $\dot{h}$ and the desired vertical speed $\dot{h}_c$ are used. Due to large variations in the altitude h and the horizontal position x of the aircraft, their decimal logarithms are supplied instead to the action network and the critic network.

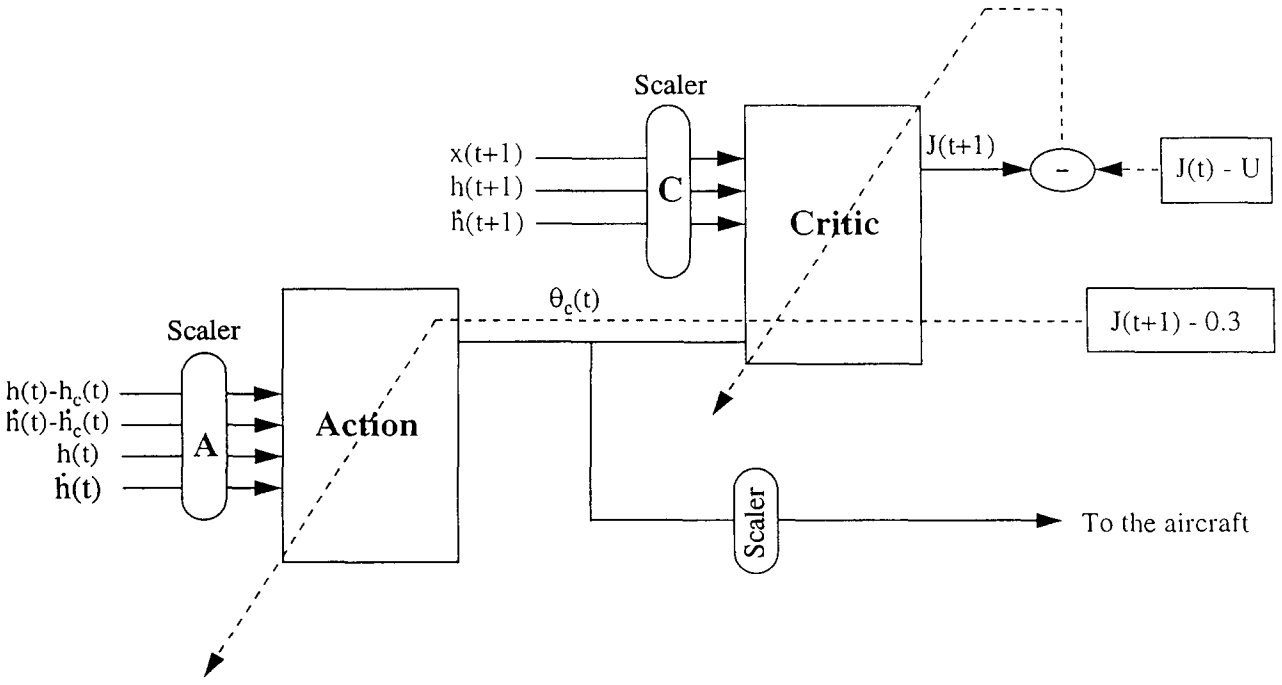

FIGURE 1. ADHDP for aircraft autolanding.

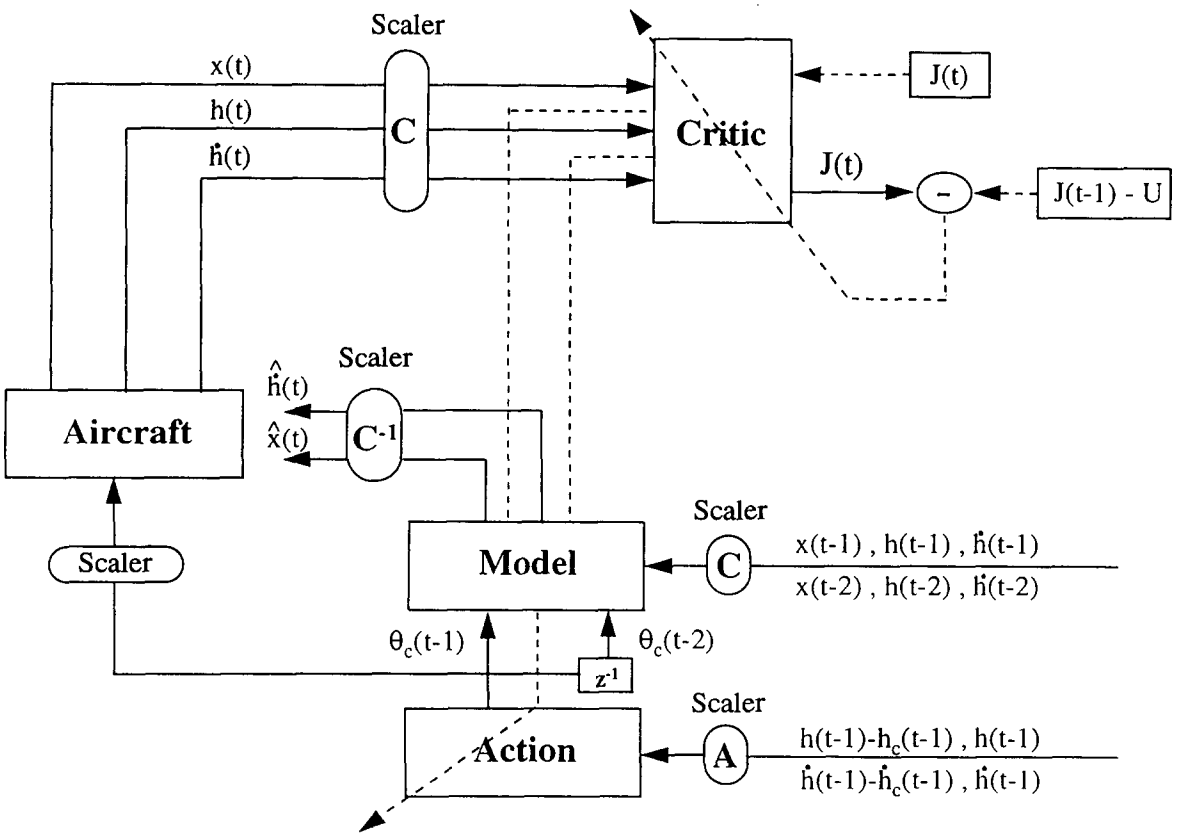

FIGURE 2. HDP for aircraft autolanding.

Training of both networks was performed in batch mode, i.e. after completion of each landing the weights were adapted. Until the instant of landing, the utility U is equal to $+0.3$ if the previous landing was successful, and to -0.3 if the plane previously crashed. The action net is trained by propagating a difference between a current value of J and its desired value (i.e. the value for a successful landing), so that, in essence, the action net tries to maximise U.

HDP is depicted in Figure 2. The critic network and the action network are the same as those used in ADHDP except that the critic network has a $3 \times 4 \times 1$ architecture. The model network is a time delayed feedforward network with a $8 \times 5 \times 2$ architecture. The model network attempts to predict values of x and h at the next time step. All the scalers are like those for ADHDP (Prokhorov & Wunsch, 1994). The model network was trained separately on about 7000 vectors obtained from the aircraft simulator. While the model network's outputs are not used, it provides a feedback path for finding the derivative J with respect to $\theta_c(t)$.

After the first experiments with HDP, it was observed that better results may be obtained by exploiting another utility U which takes into account specifics of the training process:

$$U = \begin{cases} +0.3, & \text{if } (\dot{h} > -1 \text{ ft/s OR } \dot{h} < -3 \text{ ft/s}) \\ & \text{AND } (x \leq 600 \text{ ft}) \\ 0, & \text{if } (-3 \leq \dot{h} \leq -1) \text{ AND } (-300 \leq x \leq 600), \\ -0.3, & \text{if } x > 600 \end{cases} \tag{1}$$

where all ifs are specified for the final x and $\dot{h}$ of the previous landing. Thus, the action network's target is $U = 0$.

DHP is shown in Figure 3. The critic net has the $5 \times 5 \times 5$ architecture with five logistic sigmoid neurons in its hidden layer, and linear outputs. The action and the model are $15 \times 5 \times 1$ and $12 \times 5 \times 5$ time delayed nets, respectively. Like HDP, similar scalers were used for input/output transformation of all the nets.

Finding the proper utility, U is the most important aspect of ACDs. The choice of U greatly affects the speed of training and the chances of getting better results. Unfortunately, it still belongs to the art of experimentation, and a theory for choosing U is yet to be worked out. As for the DHP, our final choice was the following:

$$U(h, \dot{h}, x) = (1 - 1/h)(a_1(h - h_c) + a_2(\dot{h} - \dot{h}_c)) + (a_3/h)\text{Hump}(x), \tag{2}$$

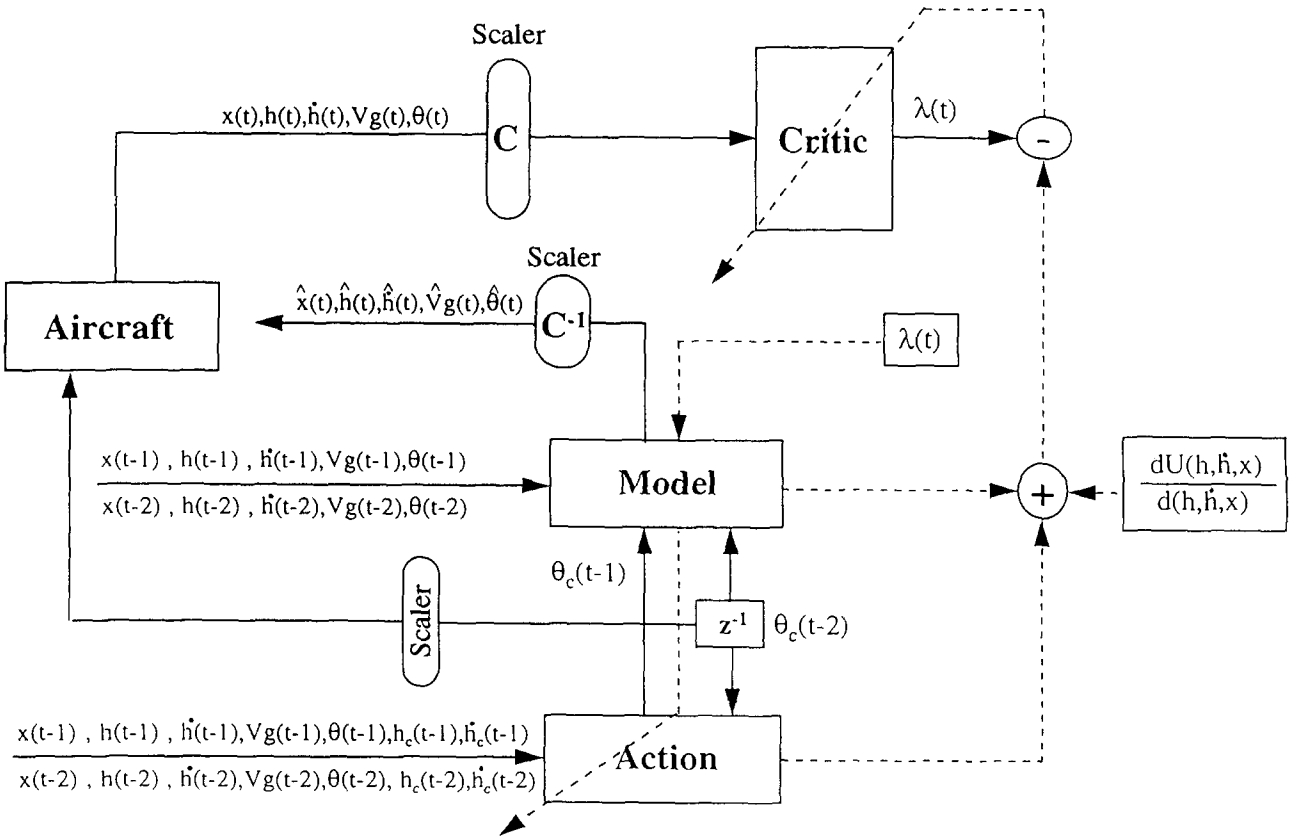

FIGURE 3. DHP for aircraft autolanding.

where Hump (x) = $\{-x/450+4/3$, if $x \geq 150$ ft, and $x/450+2/3$, if not$\}$.

First, we performed a series of simulation studies with ADHDP. For the case where there are standard constraints on x, $\dot{h}$, the actual pitch angle of the aircraft and its actual horizontal speed V_g (as in Anderson and Miller (1990)), a conventional PID regulator was proposed in Jorgensen and Schley (1990) for a comparison. This regulator is able to land the aircraft regardless of wind gusts. Moreover, when wind gusts were increased, the PID regulator performed well. The aircraft crashed in only two trials out of 400.

ADHDP steadily showed surprisingly similar performance in all the conditions the PID regulator was tested. First successful landings are obtained fast, after about 50 trials. We observed, however, that both the PID regulator and ADHDP were landing the aircraft close to the far edge of a touchdown region of the runway. In contrast, a human pilot may readily land the plane near the beginning or in the middle. Thus, it was decided to restrict the constraint on x (to shorten the touchdown region by 400 ft), thus complicating the problem.

Both the PID and ADHDP did not meet the new constraint on x both with and without wind gusts. Nevertheless, we noticed that if a simple rule of keeping the pitch angle command $\theta_c(t)$ constant and equal to $-2°$, is applied for the flight time from 0 to 32 s, ADHDP occasionally (about 3% of all trials) lands the aircraft successfully in the presence of wind gusts (neither this rule nor adjustments of parameters helped to obtain successful landings with the PID) (Prokhorov and Wunsch, 1994). This rule is the only trick that has been borrowed from manual landing experience, and it is also used further for HDP experiments below.

For the complicated problem, the HDP demonstrated a stable success in landing the plane subject to wind shear only. Learning was fast with the first success achieved in 50–100 attempts. To accelerate the weights convergence in the action network we used a high learning rate. HDP trained with and without wind gusts of various strengths, showed almost the same performance, which proved its robustness. About 40% of all trials met the artificially tightened constraints mentioned above. HDP met the original weaker constraints in another 30–40% of attempts.

Landing experiments were also made with back-propagation of utility (BU) described in Nguyen and Widrow (1990) as well as Werbos (1992). Our design for it was like HDP, except without the critic network and with the same model network. Studies performed

Gusts N(0,1)		DHP	HDP	ADHDP	BU	PID
Trained with wind shear only	**tight success**	74+	40	3	2	0
	loose success	12	20	97	98	100
	near misses	8	22.7	0	0	0
Trained additionally with wind gusts	**tight success**	72+	37	2	4	0
	loose success	8	21.5	98	96	100
	near misses	5	21.5	0	0	0
Gusts N(0,1.5)		DHP	HDP	ADHDP	BU	PID
Trained with wind shear only	**tight success**	71+	38.2	n/a	1	0
	loose success	10	15.3	97.5	96	99.5
	near misses	4	22	n/a	2	n/a
Trained additionally with wind gusts	**tight success**	70+	35.3	n/a	2	0
	loose success	7	17.5	97.5	95	99.5
	near misses	7	24.8	n/a	1	n/a

FIGURE 4. Results of all experiments.

for various desired x and $\dot{h}$ at the instant of landing showed that the backpropagation of utility is able to emulate the results obtained with the PID and ADHDP for the original problem while failing for the complicated problem. Backpropagation of utility would require a very accurate model network to match the performance of HDP in the latter case, and it is difficult to get such a model network due to the randomness of the problem.

Finally, we applied DHP to the complicated case of autolanding. Initial training of the action network alone was performed to emulate the PID regulator. Then the action network and the model network were trained together in 3000 landings. Unlike ADHDP and HDP, where common gradient descent with constant learning rate was exploited to adapt the weights, an adaptive learning rate was applied to speed up convergence. Training took longer than HDP. However, this paid off. More than 70% of all trials ended up within the tightened constraints, and another 10% of landings satisfied the original weaker constraints.

Figure 4 shows the results of all our experiments. "Tight success" means landings which met the tightened constraints. Landings within the weaker constraints are denoted as "loose success". "Near misses" mean that either $\dot{h}$ is 0.6 ft/s higher or 0.8 ft/s lower than required, or the constraint on x was broken by ± 50 ft. There is a marked improvement from HDP to DHP. DHP also demonstrates robustness in that the results for wind gusts and no wind gusts conditions are very similar. Such results are very promising because they are obtained using a rough model network (an average error per output is about 20%) and with simple network architectures. Various techniques in

existing literature can improve either of these conditions.

4. CONCLUSION

We have compared various ACDs and other techniques for the aircraft autolanding. Superiority of DHP over other control methods has been successfully demonstrated. We hope that this publication will promote further applications of DHP.

REFERENCES

Anderson, C. W., & Miller, W. T. (1990). Challenging control problems. In W. Miller, Sutton R. & P. Werbos (Eds.), *Neural networks for control* (Appendix A), Cambridge, MA: MIT Press.

Barto, A., Sutton, R., & Anderson, C. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics,* **13,** 834–846.

Franklin, J., & White, D. (1992). Artificial neural networks in manufacturing and process control. In D. White & D. Sofge (Eds.), *Handbook of Intelligent Control.* New York: Van Nostrand.

Jorgensen, C., & Schley, C. (1990). A neural network baseline problem for control of aircraft flare and touchdown. In W. Miller, R. Sutton & P. Werbos (Eds.), *Neural networks for control.* Cambridge, MA: MIT Press.

Nguyen, D., & Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine* April, 18–23.

Prokhorov, D., & Wunsch, D. (1994). An aircraft autolander based on backpropagated adaptive critic. *Proceedings of the 1st NASA Symposium on Neural Nets for Flight Control.* NASA Ames Research Center.

Santiago, R., & Werbos, P. (1994). A new progress towards truly brain-like control. *Proceedings of the WCNN-94,* **1,** 27–33.

Werbos, P. (1990). A menu of designs for reinforcement learning over time. In W. Miller, R. Sutton & P. Werbos (Eds.), *Neural networks for control.* Cambridge, MA: MIT Press.

Werbos, P. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. White & D. Sofge (Eds.), *Handbook of intelligent control.* New York: Van Nostrand.

Werbos, P. (1992). Neurocontrol and supervised learning: an overview and valuation. In D. White & D. Sofge (Eds.), *Handbook of intelligent control.* New York: Van Nostrand.