Proceedings of the 2007 IEEE Symposium on Approximate
Dynamic Programming and Reinforcement Learning (ADPRL 2007)

Toward effective combination of off-line and on-line training in ADP framework

Danil Prokhorov
Toyota Technical Center
Ann Arbor, MI 48105
E-mail: dvprokhorov@gmail.com

Abstract— **We are interested in finding the most effective combination between off-line and on-line/real-time training in approximate dynamic programming. We introduce our approach of combining proven off-line methods of training for robustness with a group of on-line methods. Training for robustness is carried out on reasonably accurate models with the multi-stream Kalman filter method [1], whereas on-line adaptation is performed either with the help of a critic or by methods resembling reinforcement learning. We also illustrate importance of using recurrent neural networks for *both* controller/actor and critic.**

I. Introduction and background

Our previous work [2] and [3] has concentrated on achieving good performance from neurocontrollers in spite of various uncertainties, especially parametric uncertainties. We demonstrated that it is possible to design a robust neurocontroller from a recurrent neural network (RNN) with fixed weights.

The trained weights of our robust neurocontroller serve as its long-term memory, which is undesirable to change. While robustness is a desirable property, it is occasionally useful to be able to adapt the fixed-weight neurocontroller to a specific plant it operates on, or track changes in the plant on-line. We can change other elements of the neurocontroller to reinforce the robust neurocontroller by adaptivity.

We can offer two options for combining adaptivity and robustness. First, we can use state variables of the recurrent network (i.e., its recurrent node outputs) as adaptive elements [4]. Such state variables act as short-term memory, which is strongly influenced by the weights, or the long-term memory.

Second, we can augment the robust neurocontroller with adaptive structure, e.g., another NN. Such a NN would compensate the remaining performance deficiencies of the robust NN and provide appropriate corrective controls to be added to the robust NN controls.

Both options above add short-term memory to the long-term memory of the robust neurocontroller, although the first option's memory depth is more affected by the long-term memory of the robust neurocontroller than that of the second option.

The adaptive NN could implement just feedforward controls (see, e.g., [5]). However, we think that in general a more effective alternative is to use an RNN in the setting of direct adaptive control.

In this paper, we will elaborate on the second option and propose alternatives for effective combination of the off-line and on-line RNN training. Our pivotal assumption is that all signals in the closed-loop system are bounded, which assures the bounded input bounded output (BIBO) stability. Our assumption holds for all real-world systems.

We discuss our choices for on-line adaptation in the next section. We then describe the example application and illustrate our proposals in simulation. We conclude with our observations and a discussion of outstanding issues.

II. On-line adaptation via additional NN

We examine here the option of augmenting the robust recurrent neurocontroller with another NN. Figure 1 depicts a possible augmentation.

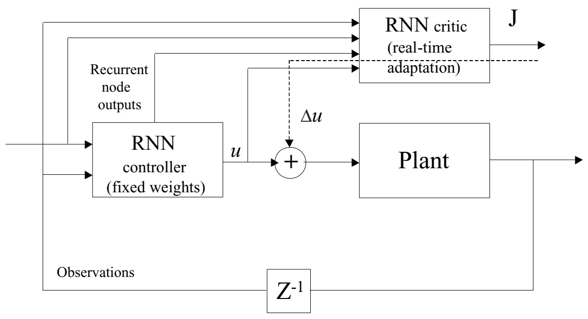

Fig. 1. Proposed augmentation of the robust recurrent neurocontroller with fixed weights by the recurrent critic for improved adaptive control of the plant. The critic network uses the same inputs as the robust neurocontroller, its output u, plus all outputs of the state nodes of the robust neurocontroller. The adaptive correction Δu of control u is computed by backpropagating through the critic.

We employ an adaptive critic to provide bounded correction signals to the controls $\mathbf{u}$ through the critic sensitivity network, i.e., via backpropagation through the critic network with respect to the control inputs $\mathbf{u}$:

$$\Delta \mathbf{u} \quad = \quad -\mu \partial J / \partial \mathbf{u} \tag{1}$$

where J is the critic output – an approximation of the cost-to-go function of dynamic programming.

The J critic attempts to approximate the discounted sum below:

$$J^*(\mathbf{z}(t)) \quad = \quad < \sum_{n=0}^{n=\infty} \gamma^n cost(\mathbf{z}(t+n)) > \tag{2}$$

where $< \cdot >$ is an average taken with respect to all possible continuations of the closed-loop state vector $\mathbf{z}(t)$. The function $cost(\cdot)$ is problem dependent. Our example below will employ the instantaneous quadratic tracking error as $cost(\cdot)$.

The critic approximates $J^*(t)$ via the usual recursion

$$J(t) \quad = \quad cost(t) + \gamma J(t+1) \tag{3}$$

where J is critic output; we suppressed the dependence on the state vector of the closed-loop system for simplicity.

The total control input to the plant is $\mathbf{u} + \Delta\mathbf{u}$. Approximation capabilities of the critic and effectiveness of its learning will clearly affect the performance achieved by the closed-loop system. We can reduce at least large residual errors of the robust neurocontroller by employing a simple critic NN. (This is illustrated in the example section.) To improve the performance significantly, the critic may need to be a sufficiently complex network which is fed by the controller inputs, the controller state variables and its outputs. It is useful to feed into the critic network as much information about the state of the closed-loop system as available for improved performance, as emphasized in [13]. To be especially effective, the critic may have to be an RNN (as shown in Figure 1), for such networks can deal with noise and partial observability better than non-recurrent NN (see, e.g., [6]).

Recurrent critics have been known for quite some time. The reader is referred to [7] and [8] for early publications on the subject and additional information. Others have also been experimenting with recurrent critics with success: see, e.g., [9], [10], [11].

To train *recurrent* adaptive critics on-line, we may need to exercise caution. First, computational constraints are likely to restrict a set of acceptable training methods, especially if training all weights of the critic network is desired. The use of methods scaling linearly with the number of the NN weights may be warranted. Second, the all-weight training in a nonlinear NN may result in output bifurcations, i.e., small changes of the critic weights may cause dramatic changes of the critic output. For continuous on-line adaptation of the critic, its bifurcations may be undesirable. In such a case only an output subset of the critic weights may have to be trained, as it is done in the so-called echo state network [14].

Figure 2 shows another option for on-line adaptation with a special NN. This special NN is an adaptive portion of the combined neurocontroller, which is to be trained by suitable methods. The advantage of this option over Figure 1 is that bounded control corrections $\Delta\mathbf{u}$ are computed directly, rather than via backpropagation through the critic. This option does not preclude the use of the critic NN, although it creates some redundancy between the robust neurocontroller with fixed weights and the adaptive controller.

For training the adaptive NN, at least two possible methods stand out, viz., the algorithm of pattern extraction (ALOPEX) [15] and the simultaneous perturbation stochastic approximation (SPSA) [16]. Both methods belong to the family of direct (model-free) adaptive control methods, and they can also be interpreted as forms of reinforcement learning. The SPSA method described below results in faster training than the ALOPEX method in our example application.

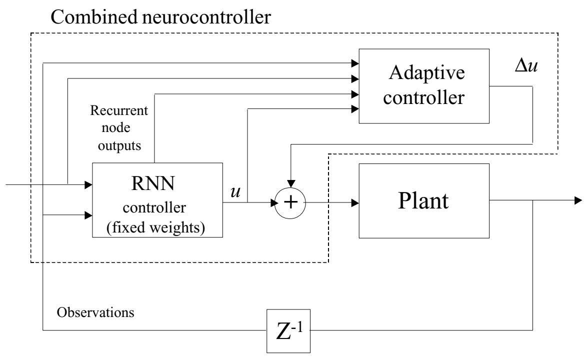

Fig. 2. Proposed combination of the robust recurrent neurocontroller with fixed weights and an adaptive controller for improved control of the plant. The adaptive controller can be another NN which uses the same inputs as the robust neurocontroller, its output u, plus all outputs of the state nodes of the robust neurocontroller. Unlike Figure 1, the adaptive correction Δu of control u is computed directly, thereby avoiding possible arbitrariness in choosing the learning rate μ in (1).

A popular form of the gradient descent-like SPSA uses two cost evaluations *independent* of parameter vector dimensionality to carry out one update of each adaptive parameter. Each SPSA update is

$$W_i^{next} \quad = \quad W_i - aG_i(\mathbf{W}) \tag{4}$$

$$G_i(\mathbf{W}) \quad = \quad \frac{Cost^+ - Cost^-}{2c\mathbf{\Delta}_i} \tag{5}$$

where $\mathbf{W}$ is a weight vector of the adaptive controller, $Cost^{\pm}$ is a cost function to be minimized, $\mathbf{\Delta}$ is a vector of symmetrically distributed Bernoulli random variables generated anew for every update step (e.g., the $i-th$ component of $\mathbf{\Delta}$ denoted as $\mathbf{\Delta}_i$ is either $+1$ or -1), c is size of a small perturbation step, and a is a learning rate.

Each SPSA update requires that two consecutive values of the $Cost$ function be computed. This means that one SPSA update occurs no more often than once every other time step. It may also be helpful to let the $Cost$ function represent changes of the cost over a short window of some number of time steps τ, in which case each SPSA update would be even less frequent. In our example $Cost$ is the windowed sum of $cost(\cdot)$ values used in (2). This allows the plant additional time to react to changing $\mathbf{W}$.

III. Example application

Our example deals with electronic throttle control (ETC). The ETC is gaining popularity in the automotive industry due to its capabilities for achieving improvements in fuel economy, drivability and other crucial performance factors. In conventional vehicles the driver pedal is linked to the engine throttle mechanically. The ETC vehicles are "drive-by-wire" vehicles, meaning that the throttle is driven by an electric motor controlled electronically through an appropriate interpretation of the driver pedal position.

The ETC is an electromechanical system consisting of a DC motor, a gear mechanism and a throttle valve with a dual spring system. In the neutral position, both springs are relaxed, and throttle valve is slightly open. This is called the "limp-home" position, and it is critical in case of the power failure allowing the engine to operate in a low power mode. The two springs have substantially different stiffness. The first spring affects the throttle valve motion at angles exceeding the "limp-home" angle, whereas the second spring counteracts the motor moment for angles smaller than the "limp-home". The second spring's stiffness must be higher to provide higher angular resolution at small angles. The throttle position is measured by a potentiometer. A sufficiently accurate angular control of the throttle valve's plate is desired.

The ETC model consists of several components (Figure 3). It includes the DC motor dynamics, with the armature time constant T_a, the armature gain K_a, the emf constant K_v, and the torque constant K_t. Our control signal is the motor voltage u, and it is limited between u_{min} and u_{max}. Further, there is a complicated model of friction due to ball bearings and the gear mechanism. The gear ratio is K_l, and the overall inertia is J. Finally, the dual spring system is modeled in the form of a lookup table.

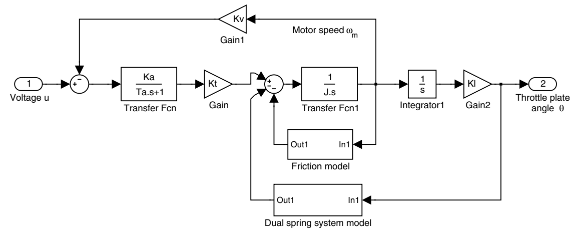

Fig. 3. Block diagram of electronic throttle. The controller (not shown) senses the throttle valve position θ (and, possibly, the motor speed ω_m) and puts out the voltage u.

The ETC model has been validated experimentally and demonstrated to be a very accurate description of the real hardware, provided that the model parameters included as components in the uncertainty vector $\boldsymbol{\theta}$ (15 components including parameters of the friction and spring models) could be estimated with high accuracy (to within a few percent from their true values). It is projected that the massive use of the ETC in automotive industry in the near future will expedite the utilization of relatively cheap components with substantial spread of parameters around their nominal values. For example, the friction model parameters or the spring stiffness may have their true values significantly different from nominal, or they may deviate significantly during the throttle service time. The ETC pervasiveness will also mean that it is too costly to calibrate any model-based control algorithm with fixed parameters. Our previous studies discuss how to obtain a robust neurocontroller for the ETC problem [3]. We illustrate here how to employ a robust neurocontroller (2-5R-3-1 RNN) in combinations with either the critic or the adaptive neurocontroller for improved performance.

First, we verify that the proposed augmentation of the fixed-weight robust neurocontroller by the adaptive critic in Figure 1 is suitable for the ETC problem. Figure 4 shows our typical result obtained with a simple feedforward NN as critic. It has one sigmoidal output node and eight inputs, consisting of two inputs and one output of the robust neurocontroller and five outputs of the recurrent nodes. The critic is trained by the standard gradient descent Heuristic Dynamic Programming (HDP) method [7] with the learning rate of 0.01; $\gamma = 0.5$, $\mu = 0.3$. We can see that the initially large tracking errors are reduced noticeably (by at least 10 percent).

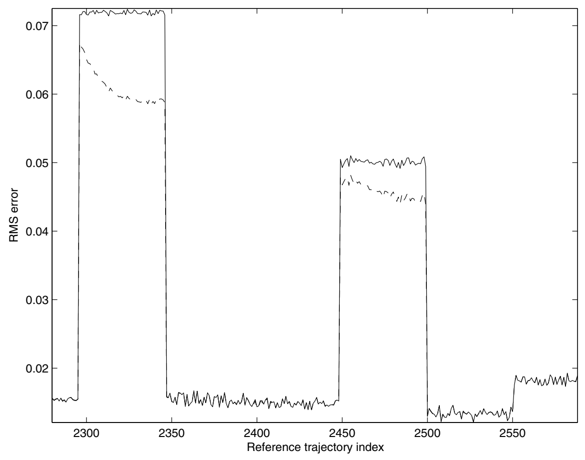

Fig. 4. Typical results of the robust and adaptive controller combination according to Figure 1. A fragment of a longer time series is shown. The plant parameters are changed abruptly every 51 reference trajectories (often shows itself as stepwise changes of the tracking RMS errors). The solid line shows the RMS errors for the robust neurocontroller only, whereas the dashed line shows the RMS errors when the critic training is enabled, producing Δu.

We also verify that the proposed combined neurocontroller in Figure 2 is effective for this problem. We employ 8-2R-1 adaptive RNN with the fully recurrent layer of two nodes and 25 weights. Its eight inputs are the same as in the previous experiment. We update its weights by SPSA with the following parameters: $a = 0.0002$, $c = 0.003$ and $\tau = 5$ for computation of $Cost$ in (4). The histogram in Figure 5 shows that the worst RMS error is reduced substantially for the combined controller than for the robust controller alone, with more than 10% reduction of the mean RMS error. It is worthwhile to point out that 8-5-1 feedforward NN (51 weights) yields a noticeably worse reduction of the mean RMS error.

IV. Discussion

In spite of conceptual simplicity of the proposed control schemes for combining robust and adaptive components, various issues remain to be clarified.

Parameters for training the adaptive part of the combined systems in Figures 1 and 2 should be optimized for best performance. Our choice of the training parameters in both experiments is reasonable, but obtained by trial and error and by no means optimal; a random choice of the parameters is

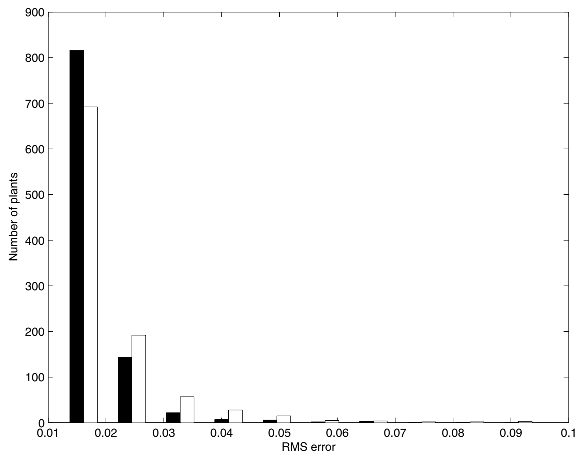

Fig. 5. Comparison of tracking RMS errors over 1000 plants for the same reference trajectory for the robust and adaptive controller combination according to Figure 2. The black bars are for the combination of the two neurocontrollers (the maximum RMS error is 0.071), whereas the white bars are for the robust neurocontroller only (the maximum RMS error is 0.096). The mean RMS error shows 12.5% improvement for the combination.

likely to be poor, resulting in significant degradation of the original robustness.

Ideally, the training parameters should be chosen such that they reduce all RMS errors, i.e., for all plants, disturbances, reference trajectories, etc. The choice of *fixed* parameters, as illustrated here, is likely to fall short of this ideal, at most resulting in reductions of the average RMS errors and the worst among the RMS errors values. If the minimization of the worst RMS errors is the only concern, then it is possible to create an adaptation scheme with a dead zone in which the adaptive part will begin its learning and affecting the robust part only if the errors exceed a specified threshold. However, if improvement of all RMS error values over those of the robust controller is desired, then the training parameters of the adaptive part may need to be adjusted on-line.

In our illustrations above we chose to apply the corrections Δu at every time step. Determining *when* to apply Δu for maximum effect may be important, especially for the scheme of Figure 1 because changes of controls may act as disturbances to the critic, as noted in [12].

The optimal representation for the adaptive part in Figures 1 and 2 is another issue, stemming from the well known stability-plasticity dilemma. The fastest adaptation may require the simplest possible representation, i.e., the adaptive bias for each of the components of $\mathbf{u}$. However, such memoryless representation might not be the optimal choice for many problems.

V. Conclusion

We propose two combinations for the off-line and the on-line training in the ADP framework and demonstrate their performance on an industrial application. They feature a robust neurocontroller with weights fixed after off-line training and adaptive elements for on-line training. Explicit adaptivity originates from a special NN which may need be recurrent for improved performance. Such a NN can be either an adaptive critic, or an adaptive controller which augments the robust controller and is trained via forms of reinforcement learning. This special (adaptive) NN and the associated training algorithm may need to have the complexity simpler than that of the robust NN to reduce computational load of the on-line implementation.

References

[1] L. A. Feldkamp, D. V. Prokhorov, C. F. Eagen, and F. Yuan, "Enhanced multi-stream Kalman filter training for recurrent networks," in J. Suykens and J. Vandewalle (eds), *Nonlinear Modeling: Advanced Black-Box Techniques*, Kluwer Academic Publishers, 1998., pp. 29–53.

[2] D. V. Prokhorov, G. V. Puskorius, and L. A. Feldkamp, "Dynamical neural networks for control," see in *A Field Guide to Dynamical Recurrent Networks*, J. Kolen and S. Kremer (Eds.), IEEE Press, 2001, pp. 257–289.

[3] D. Prokhorov, "Training Recurrent Neurocontrollers for Robustness with Derivative-Free Kalman Filter," *IEEE Trans. Neural Networks*, November 2006, pp. 1606–1616.

[4] D. Prokhorov, "Training Recurrent Neurocontrollers for Real-Time Applications," *IEEE Trans. Neural Networks*, to appear.

[5] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control Systems Magazine*, vol. 8, no. 2, April 1988, pp. 8–15.

[6] J. Suykens, J. Vandewalle, and B. De Moor, *Artificial Neural Networks for Modeling and Control of Non-Linear Systems*, Kluwer Academic, 1996.

[7] Publications of Paul J. Werbos at $www.werbos.com$.

[8] L.-J. Lin and T. Mitchell, Memory Approaches To Reinforcement Learning In Non-Markovian Domains, Technical Report CMU-CS-92-138, School of Computer Science, Carnegie Mellon University, May 1992.

[9] P. Eaton, D. Prokhorov, and D. Wunsch, "Neurocontroller Alternatives for Fuzzy Ball-and-Beam Systems with Nonlinear, Nonuniform Friction," *IEEE Trans. on Neural Networks*, March 2000, pp. 423-435.

[10] K. Bush and C. Anderson, "Modeling Reward Functions for Incomplete State Representations via Echo State Networks," *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada, August 1-4, 2005.

[11] B. Bakker, "Reinforcement learning by backpropagation through an LSTM model/critic," *Proceedings of the 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, Honolulu, Hawaii, April 1-5, 2007.

[12] L. Feldkamp and D. Prokhorov, "Observations on the Practical Use of Derivative Adaptive Critics," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Orlando, FL, October 1997, pp. 3061-3066.

[13] D. Prokhorov, "Backpropagation through time and derivative adaptive critics: a common framework for comparison," Chapter 15 in *Handbook of Learning and Approximate Dynamic Programming*, J. Si et al. (eds), IEEE Press, 2004.

[14] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunications," *Science*, April 2, 2004, pp. 78–80.

[15] E. Micheli-Tzanakou, *Supervised and Unsupervised Pattern Recognition: Feature Extraction in Computational Intelligence*, CRC Press, 2000.

[16] J. C. Spall and J. A. Cristion, "Model-free control of nonlinear stochastic systems with discrete-time measurements," *IEEE Trans. Automatic Control*, vol. 43, no. 9, September 1998, pp. 1198–1210.