Adaptive Critic Learning Techniques for Engine Torque and Air–Fuel Ratio Control

Derong Liu, *Fellow, IEEE*, Hossein Javaherian, *Member, IEEE*, Olesia Kovalenko, and Ting Huang, *Student Member, IEEE*

***Abstract*—A new approach for engine calibration and control is proposed. In this paper, we present our research results on the implementation of adaptive critic designs for self-learning control of automotive engines. A class of adaptive critic designs that can be classified as (model-free) action-dependent heuristic dynamic programming is used in this research project. The goals of the present learning control design for automotive engines include improved performance, reduced emissions, and maintained optimum performance under various operating conditions. Using the data from a test vehicle with a V8 engine, we developed a neural network model of the engine and neural network controllers based on the idea of approximate dynamic programming to achieve optimal control. We have developed and simulated self-learning neural network controllers for both engine torque (TRQ) and exhaust air–fuel ratio (AFR) control. The goal of TRQ control and AFR control is to track the commanded values. For both control problems, excellent neural network controller transient performance has been achieved.**

***Index Terms*—Adaptive critic designs (ACDs), adaptive dynamic programming, air–fuel ratio (AFR) control, approximate dynamic programming, automotive engine control, torque control.**

I. Introduction

IN AN effort to design more advanced engine control algorithms with the objectives of reduced emissions and improved performance, we develop and evaluate a learning control technique that originated from dynamic programming. Dynamic programming is a theory developed back in the 1950s [2] for optimal control of nonlinear systems with the objective of minimizing a performance index that is defined as a summation of a utility function from the present time to the future. In general, using dynamic programming, such an optimal control design for nonlinear systems is only theoretically possible. Moreover, in practice, it has been known for years that, due to the so-called "curse of dimensionality" [13], dynamic programming can only be applied to simple small-scale control problems.

Manuscript received August 14, 2007. This work was supported in part by the National Science Foundation under Grant ECS-0355364 and in part by General Motors Corporation. This paper was recommended by Guest Editor F. Lewis.

D. Liu and T. Huang are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: dliu@ece.uic.edu; thuang@cil.ece.uic.edu).

H. Javaherian is with the Powertrain Systems Research Laboratory, General Motors Research and Development Center, Warren, MI 48090 USA (e-mail: hossein.javaherian@gm.com).

O. Kovalenko is with McMaster-Carr, Elmhurst, IL 60126 USA (e-mail: olesia@cil.ece.uic.edu).

Digital Object Identifier 10.1109/TSMCB.2008.922019

Automotive engines are known to be complex nonlinear dynamical systems. The control problem of automotive engines has been investigated for many years by many researchers (see, e.g., [6], [7], [20], [22], [26], [31], [36], and the references cited therein). Almost every branch of the modern and classical control theory has been researched for the control of automotive engines. This paper adds another dimension to the existing rich literature on automotive engine control.

The uniqueness of this paper comes from the following idea: Consider a currently existing control algorithm implemented in a production vehicle. The algorithm is designed according to certain criteria and calibrated for vehicle operation over the entire operating regime. In a sense, the algorithm has been optimized for the engine in terms of its performance, fuel economy, and tailpipe emissions through a significant effort in the research and development, and calibration process. To further improve the engine performance through controller design, one can go through the traditional calibration and control procedures in place today. An alternative to this traditional approach is to use the neural-network-based learning control design approach initiated in this paper.

The final result of our neural network learning process is a controller that has learned to provide optimal control signals under various operating conditions. We emphasize that such a neural network controller will be obtained after a specially designed learning process that performs approximate dynamic programming. Once a controller is learned and obtained (offline or online), it will be applied to perform the task of engine control. The performance of the controller can be further refined and improved through continuous learning in real-time vehicle operations. We note that continuous learning and adaptation to improve controller's performance is one of the key promising attributes of the present approach. Continuous learning and adaptation for optimal individual engine performance over the entire operating regime and vehicle conditions would be desirable for future engine controller designs. For practical reasons, during the initial stage of the controller neural network learning, it is preferable to use offline engine data for initial simulation studies. We will therefore first develop a model of the engine for the purpose of initial neural network controller learning, but such a model is not necessary for the real-time engine operation.

This paper is organized as follows: In Section II, the neural network model of the test engine is described. In Section III, adaptive critic designs will be briefly introduced. In Section IV, engine torque (TRQ) and exhaust air–fuel ratio (AFR) tracking control using adaptive critic designs will be presented with

simulation results. In Section V, this paper will be concluded with a few remarks.

II. Neural Network Modeling of the Test Engine

A test vehicle with a V8 engine and four-speed automatic transmission is instrumented with engine and transmission torque sensors, wide-range AFR sensors in the exhaust pipe located before and after the catalyst on each bank, and exhaust gas pressure and temperature sensors. The vehicle is also equipped with a dSPACE rapid prototyping controller for data collection and controller implementation. Data are collected at each engine event under various driving conditions, such as federal test procedure (FTP cycles), as well as more aggressive driving patterns, for a length of about 95 000 samples during each test. The engine is run under closed-loop fuel control using switching-type oxygen sensors. The dSPACE is interfaced with the powertrain control module in bypass mode.

We build a neural network model for the test engine with a structure compatible with the mathematical engine model developed by Dobner [11], [12] and others. Due to the complexity of modern automotive engines, in this paper, we use the time-lagged recurrent neural networks (TLRNs) for engine modeling. In practice, TLRNs have been used often for function approximation, and it is believed that they are more powerful than the networks with only feedforward structures (cf. [27] and [34]).

For the neural network engine model, we choose AFR and TRQ as the two outputs. We choose throttle position (TPS), electrical fuel pulsewidth (FPW), and spark advance (SPA) as the three control inputs. These are input signals to be generated using our new adaptive critic learning control algorithm. We choose intake manifold pressure (MAP), mass air flow rate (MAF), and engine speed (RPM) as reference inputs. The TLRN used for the engine combustion module has six input neurons, a single hidden layer with eight neurons, and two output neurons.

Validation results for the outputs TRQ and AFR of the neural network engine model indicate a very good match between the real vehicle data and the neural network model outputs during the validation phase [14].

III. ADHDP

Adaptive critic designs (ACDs) [1], [4], [5], [8], [9], [15]–[17], [19], [21], [23]–[25], [28]–[30], [32], [33], [35] are defined as *designs that approximate dynamic programming in the general case*, i.e., approximate optimal control over time in noisy nonlinear environments. There are many problems in practice that can be formulated as cost maximization or minimization problems. Examples include error minimization, energy minimization, profit maximization, and the like. Dynamic programming is a very useful tool in solving these problems. However, it is often computationally untenable to run dynamic programming due to the backward numerical process required for its solutions, i.e., due to the "curse of dimensionality" [2], [13]. Over the years, progress has been made to circumvent the "curse of dimensionality" by building a system, called "critic," to approximate the cost function in dynamic programming (cf. [1], [23], [25], [30], [33], and [35]).

A. Dynamic Programming

Suppose that one is given a discrete-time nonlinear dynamical system

$$x(t+1) = F\left[x(t), u(t), t\right] \tag{1}$$

where $x \in R^n$ represents the state vector of the system, and $u \in R^m$ denotes the control action. Suppose that one associates with this system the performance index (or cost)

$$J\left[x(i), i\right] = \sum_{k=i}^{\infty} \gamma^{k-i} U\left[x(k), u(k), k\right] \tag{2}$$

where U is called the utility function or local cost function, and γ is the discount factor, with $0 < \gamma \leq 1$. Note that J is dependent on initial time i and initial state $x(i)$, and it is referred to as the cost-to-go of state $x(i)$. The objective is to choose control sequence $u(k)$, $k = i, i+1, \ldots$, so that the function J (i.e., the cost) in (2) is *minimized*.

Suppose that one has computed optimal cost $J^*[x(t+1), t+1]$ from time $t+1$ onward for all possible states $x(t+1)$ and that one has also found the optimal control sequences from time $t+1$ onward. The optimal cost results when optimal control sequence $u^*(t+1)$, $u^*(t+2), \ldots$, is applied to the system with initial state $x(t+1)$. Optimal control $u^*(t)$ at time t is determined by [2], [3], [13], [18]

$$u^*(t) = \arg\min_{u(t)} \left(U\left[x(t), u(t), t\right] + \gamma J^*\left[x(t+1), t+1\right]\right). \tag{3}$$

The corresponding optimal cost from time t onward is equal to

$$J^*\left[x(t), t\right] = \min_{u(t)} \left(U\left[x(t), u(t), t\right] + \gamma J^*\left[x(t+1), t+1\right]\right).$$

Equation (3) is the principle of optimality for discrete-time systems.

In the computations in (3), whenever one knows the function J in (2) and the model F in (1), it is a simple problem in function minimization to pick the action $u^*(t)$ that achieves the minimum in (3). However, this procedure requires a backward numerical process, and it is too computationally expensive to determine the solutions due to the so-called "curse of dimensionality" [2], [13].

B. Approximate Dynamic Programming

A typical design of ACDs consists of three modules: 1) critic (for evaluation); 2) model (for prediction); and 3) action (for decision) [23], [25], [32], [33], [35]. When in ACDs, the critic network (i.e., the evaluation module) takes the action/control signal as part of its inputs, the designs are referred to as action-dependent ACDs. We use in this paper an action-dependent version of ACDs that does not require the explicit use of the model network in the design. The critic network in this case

will be trained by minimizing the following error measure over time:

$$\|E_q\| = \sum_t E_q(t) = \sum_t \left[Q(t-1) - U(t) - \gamma Q(t)\right]^2 \quad (4)$$

where $Q(t) = Q[x(t), u(t), t, W_C]$. When $E_q(t) = 0$ for all t, (4) implies that

$$\begin{aligned} Q(t-1) &= U(t) + \gamma Q(t) \\ &= U(t) + \gamma \left[U(t+1) + \gamma Q(t+1)\right] \\ &= \cdots \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} U(k). \end{aligned} \quad (5)$$

Comparing to (2), we can see that, when minimizing the error function in (4), we have a neural network trained, so that its output at time t becomes an estimate of the cost function defined in dynamic programming for $i = t + 1$, i.e., the value of the cost function in the immediate future [21].

The input–output relationship of the critic network is given by

$$Q(t) = Q\left[x(t), u(t), t, W_C\right]$$

where W_C represents the weight vector of the critic network.

We can train the critic network at time $t-1$ with the desired output target given by $U(t) + \gamma Q(t)$. The training of the critic network is to realize the mapping given by

$$C_f : \begin{Bmatrix} x(t-1) \\ u(t-1) \end{Bmatrix} \rightarrow \left\{U(t) + \gamma Q(t)\right\}. \quad (6)$$

We consider $Q(t-1)$ in (4) as the output from the *network to be trained*, and the target output value for the critic network is calculated using its output at time t.

After the critic network's training is finished, the action network's training starts with the objective of minimizing $Q(t)$. The goal of the action network training is to minimize critic network output $Q(t)$. In this case, we can choose the target of the action network training as zero, i.e., we will train the action network, so that the output of the critic network becomes as small as possible. The desired mapping that will be used for the training of the action network in the present action-dependent heuristic dynamic programming (ADHDP) is given by

$$A : \{x(t)\} \rightarrow \{0(t)\} \quad (7)$$

where $0(t)$ indicates the target values of zero. We note that, during the training of the action network, it will be connected to the critic network to form a larger neural network. The target in (7) is for the output of the *whole network*, i.e., the output of *the critic network after it is connected to the action network*.

After the action network's training cycle is completed, one may check the system's performance and then stop or continue the training procedure by going back to the critic network's training cycle again if the performance is not acceptable yet.

C. Tracking Control Problems

Assume that the control objective is to have $x(t)$ in (1) track another signal given by $x^*(t)$. We can define, in this case, local cost function $U(t)$ as

$$U(t) = \frac{1}{2} e^T(t) e(t) = \frac{1}{2} \left[x(t) - x^*(t)\right]^T \left[x(t) - x^*(t)\right].$$

Using the ADHDP introduced earlier in this section, we can design a controller to minimize

$$J(t) = \sum_{i=t}^{\infty} \gamma^{i-t} U(i)$$

where $0 < \gamma < 1$. We note that, in this case, our control objective is to minimize an infinite summation of $U(t)$ from the current time to the infinity future, whereas, in conventional tracking control designs, the objective is often to minimize $U(t)$ itself.

IV. Simulation Studies of Adaptive Critic Learning Control of a V8 Engine

The objective of the present engine controller design is to provide control signals, so that the torque generated by the engine will track the torque measurement as in the data and the AFR will track the required values also as in the data. The measured torque values in the data are generated by the engine using the existing controller. Our learning controller will assume no knowledge about the control signals provided by the existing controller. It will generate a set of control signals that are independent of the control signals in the measured data. Based on the data collected, we use our learning controller to generate control signals TPS, FPW, and SPA, with the goal of producing exactly the same torque and AFR as in the data set. That is to say, we keep our system in the same requirements as the data collected and build a controller that provides control signals that achieve the torque control and AFR control performance of the engine.

As described in the previous section, the development of an adaptive critic learning controller involves two stages: 1) the training of a critic network and 2) the development of a controller/action network. We describe in the rest of this section the learning control design for tracking the TRQ and AFR measurements in the data set. This is effectively a torque-based controller, i.e., a controller that can generate control signals given the torque demand. The block diagram of the present adaptive critic engine control (including AFR control) is shown in Fig. 1. The diagram shows how adaptive critic designs can be applied to engine control through approximate dynamic programming.

A. Critic Network

The critic network is chosen as a 8–15–1 structure with eight input neurons and 15 hidden layer neurons.

- The eight inputs to the critic network are TRQ, TRQ*, MAP, MAF, RPM, TPS, FPW, and SPA, where TRQ* is

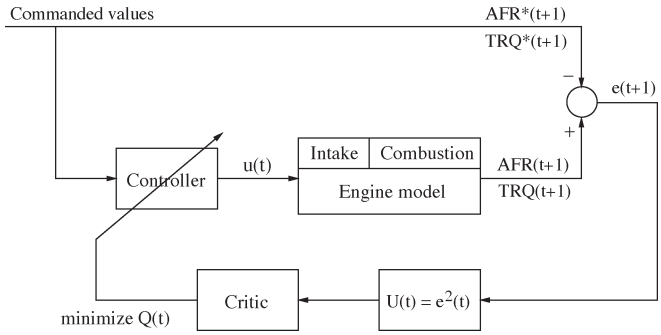

Fig. 1. Structure of the adaptive critic learning engine controller.

read from the data set, indicating the desired torque values for the present learning control algorithm to track.

- The hidden layer of the critic network uses sigmoidal function, i.e., the `tansig` function in Matlab [10], and the output layer uses the linear function `purelin`.
- The critic network outputs function Q, which is an approximation to the function $J(t)$ defined as in (2).
- The local cost function U defined in (2) in this case is chosen as

$$U(t) = \frac{1}{2}\left[\mathrm{TRQ}(t) - \mathrm{TRQ}^*(t)\right]^2 + \frac{1}{2}\left[\mathrm{AFR}(t) - \mathrm{AFR}^*(t)\right]^2$$

where $\mathrm{TRQ}(t)$ and $\mathrm{AFR}(t)$ are the TRQ and AFR generated using the proposed controller, respectively, and TRQ^* and AFR^* are the demanded TRQ value and the desired AFR value, respectively. Both TRQ^* and AFR^* are taken from the actual measured data in the present case. The utility function chosen in this way will lead to a control objective of TRQ, following TRQ^* and AFR following AFR^*.

- Utilizing the Matlab Neural Network Toolbox, we have applied `traingdx` (gradient descent algorithm) for the training of the critic network. We note that other algorithms implemented in Matlab, such as `traingd`, `traingda`, `traingdm`, and `trainlm`, are also applicable. We employ batch training for the critic network, i.e., the training is performed after each trial of a certain number of steps (e.g., 10 000 steps). We choose $\gamma = 0.9$ in the present experiments.

B. Controller/Action Network

The structure of the action network is chosen as 6–12–3 with six input neurons, 12 hidden layer neurons, and three output neurons.

- The six inputs to the action network are TRQ, TRQ^*, MAP, MAF, THR, and RPM, where THR indicates the driver's throttle command.
- Both the hidden layer and the output layer use the sigmoidal function `tansig`.
- The outputs of the action network are TPS, FPW, and SPA, which are the three control input signals used in the engine model.
- The training algorithm we choose to use is `traingdx`. We employ batch training for the action network as well.

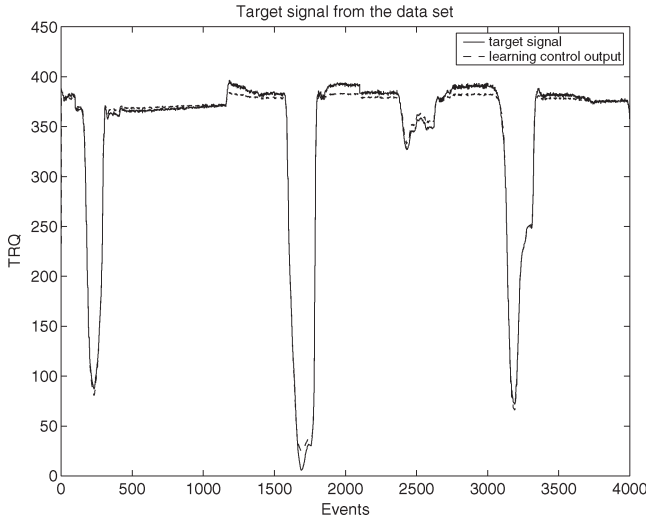

Fig. 2. Torque output generated by the neural network controller.

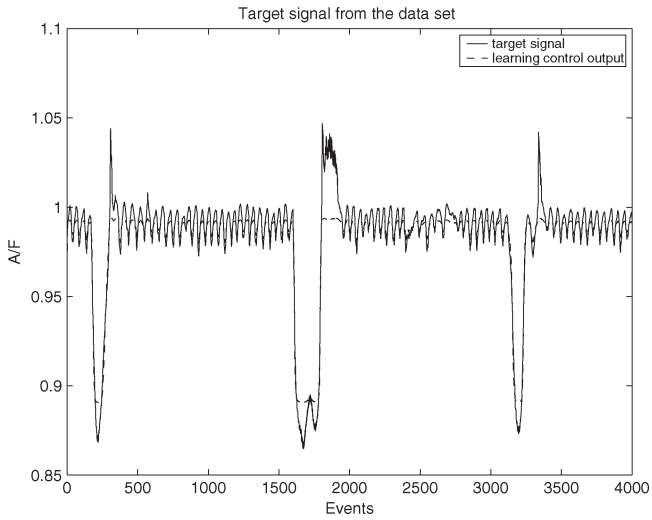

Fig. 3. AFR output generated by the neural network controller.

C. Simulation Results

In the present simulation studies, we first train a critic network for many cycles with 500 training epochs in each cycle. At the end of each training cycle, we check the performance of the critic network. Once the performance is found to be satisfactory, we stop critic network training. This process usually takes about 6–7 h.

After the critic network training is finished, we start the action network training. We train the controller network for 200 epochs after each trial. We check to see the performance of the neural network controller at the end of each trial.

We choose to use 4000 data points from the data (16 000–20 000 in the data set) for the present critic and action network training.

We first show the TRQ and AFR outputs due to the initial training of our neural network controller when TRQ^* and AFR^* are chosen as random signals during training. Figs. 2 and 3 show the controller performance when it is applied, with TRQ^* and AFR^* chosen as the measured values in the data set. The neural network controller in this case is trained for 15 cycles using randomly generated target signals TRQ^* and

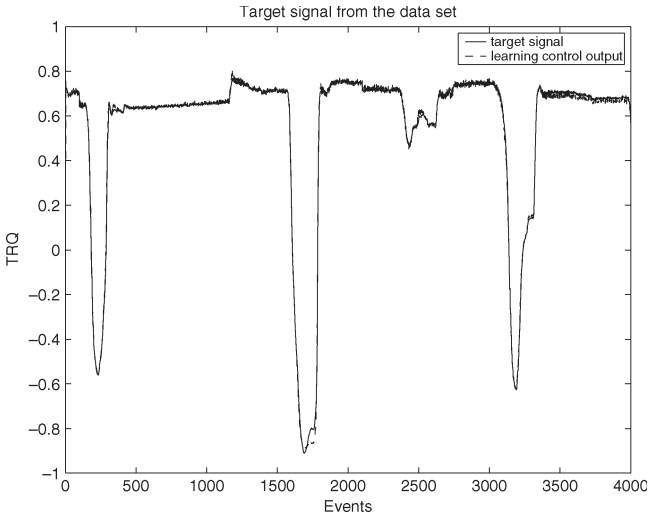

Fig. 4. Torque output generated by the refined neural network controller.

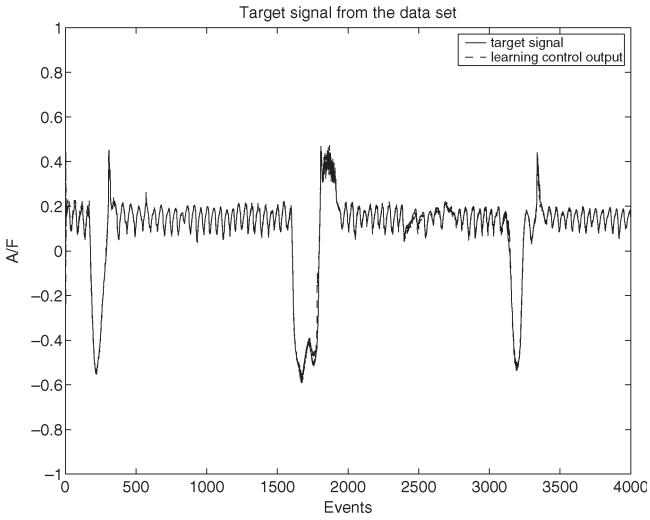

Fig. 5. AFR output generated by the refined neural network controller.

AFR*. Figs. 2 and 3 show that very good tracking control of the commanded torque signal (TRQ) and the exhaust AFR are achieved. We note that, at the present stage of the research, we have not attempted to regulate the AFR at the stoichiometric value but to track a given command. In these experiments, we simply try to track the measured engine-out AFR values, so that the control signal obtained can be directly validated against the measured control signals in the vehicle. In Fig. 3, it appears that better tracking of AFR was achieved on the rich side of the stoichiometric value possibly due to more-frequent rich excursions encountered during model training. This could also have been caused by intentional fuel enrichments (i.e., wall-wetting compensation) during vehicle accelerations.

Figs. 4 and 5 show the TRQ and AFR outputs after refined training when TRQ* and AFR* are chosen as the measured values in the data. The neural network controller in this case is trained for 15 cycles using target signal TRQ* and AFR* as in the data. Figs. 4 and 5 show that excellent tracking control results for the commanded TRQ and AFR are achieved.

The figures shown in this section indicate that the present learning controller design based on approximate dynamic programming (adaptive critic designs) is effective in training a neural network controller to track the desired TRQ and AFR sequences through proper control actions.

In the next phase of our work, we will refine the controller performance through network structure optimization and using longer training process.

V. Conclusion

Our research results have demonstrated that adaptive critic techniques provide a powerful alternative approach for engine calibration and control. The design is based on neural network learning using approximate dynamic programming. After the network is fully trained, the present controller may have the potential to outperform existing controllers with regard to the following three aspects: 1) The proposed technique will automatically learn the inherent dynamics and nonlinearities of the engine from real vehicle data and therefore do not require a mathematical model of the system to be developed. 2) The methods developed will further advance the development of a virtual powertrain for performance evaluation of various control strategies through the development of neural network models of engine and transmission in a prototype vehicle. 3) The proposed controllers can learn to improve their performance during the actual vehicle operations, and will adapt to uncertain changes in the environment and vehicle conditions. This is an inherent feature of the proposed neural network learning controller. As such, these techniques may offer promise for use as real-time engine calibration tools.

Simulation results show that the proposed self-learning control approach is effective in achieving tracking control of TRQ and AFR control through neural network learning.

Acknowledgment

The authors would like to thank Dr. M.-F. Chang for the technical discussions and support, and A. W. Brown and M. P. Nolan (all of General Motors R&D) for the help in the experimental vehicle setup, algorithm implementation, and data collection.

References

[1] S. N. Balakrishnan and V. Biega, "Adaptive-critic-based neural networks for aircraft optimal control," *J. Guid. Control Dyn.*, vol. 19, no. 4, pp. 893–898, Jul.–Aug. 1996.

[2] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[5] J. Campos and F. L. Lewis, "Adaptive critic neural network for feedforward compensation," in *Proc. Amer. Control Conf.*, San Diego, CA, Jun. 1999, vol. 4, pp. 2813–2818.

[6] F. E. Coats, Jr. and R. D. Fruechte, "Dynamic engine models for control development—Part II: Application to idle speed control," *Int. J. Veh. Des.*, pp. 75–88, 1983. Special Publication SP4.

[7] J. A. Cook and B. K. Powell, "Modeling of an internal combustion engine for control analysis," *IEEE Control Syst. Mag.*, vol. 8, no. 4, pp. 20–26, Aug. 1988.

[8] C. Cox, S. Stepniewski, C. Jorgensen, R. Saeks, and C. Lewis, "On the design of a neural network autolander," *Int. J. Robust Nonlinear Control*, vol. 9, no. 14, pp. 1071–1096, Dec. 1999.

[9] J. Dalton and S. N. Balakrishnan, "A neighboring optimal adaptive critic for missile guidance," *Math. Comput. Model.*, vol. 23, no. 1/2, pp. 175–188, Jan. 1996.
[10] H. Demuth and M. Beale, *Neural Network Toolbox User's Guide*. Natick, MA: MathWorks Inc., 1998. (Version 3).
[11] D. J. Dobner, *A Mathematical Engine Model for Development of Dynamic Engine Control*, 1980. SAE Paper 800054.
[12] D. J. Dobner, "Dynamic engine models for control development—Part I: Non-linear and linear model formation," *Int. J. Veh. Des.*, pp. 54–74, 1983. Special Publication SP4.
[13] S. E. Dreyfus and A. M. Law, *The Art and Theory of Dynamic Programming*. New York: Academic, 1977.
[14] O. Kovalenko, D. Liu, and H. Javaherian, "Neural network modeling and adaptive critic control of automotive fuel-injection systems," in *Proc. IEEE Int. Symp. Intell. Control*, Taipei, Taiwan, Sep. 2004, pp. 368–373. (Invited paper).
[15] O. Kuljaca and F. L. Lewis, "Fuzzy logic/neural network adaptive critic controller design," in *Proc. 41st IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 2002, vol. 3, pp. 3356–3361.
[16] N. V. Kulkarni and K. KrishnaKumar, "Intelligent engine control using an adaptive critic," *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 2, pp. 164–173, Mar. 2003.
[17] G. G. Lendaris and C. Paintz, "Training strategies for critic and action neural networks in dual heuristic programming method," in *Proc. Int. Conf. Neural Netw.*, Houston, TX, Jun. 1997, pp. 712–717.
[18] F. L. Lewis and V. L. Syrmos, *Optimal Control*. New York: Wiley, 1995.
[19] F. L. Lewis, J. Campos, and R. Selmic, "On adaptive critic architectures in feedback control," in *Proc. 38th IEEE Conf. Decision Control*, Phoenix, AZ, Dec. 1999, vol. 2, pp. 1677–1684.
[20] X. Li and S. Yurkovich, "Sliding mode control of delayed systems with application to engine idle speed control," *IEEE Trans. Control Syst. Technol.*, vol. 9, no. 6, pp. 802–810, Nov. 2001.
[21] D. Liu, X. Xiong, and Y. Zhang, "Action-dependent adaptive critic designs," in *Proc. INNS-IEEE Int. Joint Conf. Neural Netw.*, Washington, DC, Jul. 2001, pp. 990–995.
[22] J. J. Moskwa and J. K. Hedrick, "Nonlinear algorithms for automotive engine control," *IEEE Control Syst. Mag.*, vol. 10, no. 3, pp. 88–93, Apr. 1990.
[23] D. V. Prokhorov, "Adaptive critic designs and their applications," Ph.D. dissertation, Texas Tech Univ., Lubbock, TX, Oct. 1997.
[24] D. V. Prokhorov, R. A. Santiago, and D. C. Wunsch, "Adaptive critic designs: A case study for neurocontrol," *Neural Netw.*, vol. 8, no. 9, pp. 1367–1372, 1995.
[25] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 997–1007, Sep. 1997.
[26] P. F. Puleston, S. Spurgeon, and G. Monsees, "Automotive engine speed control: A robust nonlinear control framework," *Proc. Inst. Electr. Eng.—Control Theory Appl.*, vol. 148, no. 1, pp. 81–87, Jan. 2001.
[27] G. V. Puskorius, L. A. Feldkamp, and L. I. Davis, "Dynamic neural network methods applied to on-vehicle idle speed control," *Proc. IEEE*, vol. 84, no. 10, pp. 1407–1420, Oct. 1996.
[28] G. N. Saridis and F.-Y. Wang, "Suboptimal control of nonlinear stochastic systems," *Control Theory Adv. Technol.*, vol. 10, no. 4, pp. 847–871, 1994.
[29] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*, Wiley, New York, 2004
[30] J. Si and Y.-T. Wang, "On-line learning control by association and reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 264–276, Mar. 2001.
[31] G. J. Vachtsevanos, S. S. Farinwata, and D. K. Pirovolou, "Fuzzy logic control of an automotive engine," *IEEE Control Syst. Mag.*, vol. 13, no. 3, pp. 62–68, Jun. 1993.
[32] P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-17, no. 1, pp. 7–20, Jan./Feb. 1987.
[33] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*, W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds. Cambridge, MA: MIT Press, 1990, ch. 3.
[34] P. J. Werbos, T. McAvoy, and T. Su, "Neural networks, system identification, and control in the chemical process industries," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, ch. 10.
[35] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, ch. 13.
[36] M. Won, S. B. Choi, and J. K. Hedrick, "Air-to-fuel ratio control of spark ignition engines using Gaussian network sliding control," *IEEE Trans. Control Syst. Technol.*, vol. 6, no. 5, pp. 678–687, Sep. 1998.

Derong Liu (S'91–M'94–SM'96–F'05) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 1994.

He is currently a Full Professor of electrical and computer engineering with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL. He has published seven books (four research monographs and three edited volumes). He is also an Associate Editor for *Automatica*.

Dr. Liu is currently the Editor for the IEEE Computational Intelligence Society's ELECTRONIC LETTER, an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS, an Associate Editor for the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE, and an Associate Editor for the IEEE CIRCUITS AND SYSTEMS MAGAZINE. He is the General Chair for the 2008 IEEE International Conference on Networking, Sensing and Control in China and the Program Chair for the 2008 International Joint Conference on Neural Networks. He is an elected AdCom member (2006–2008) of the IEEE Computational Intelligence Society. He was the recipient of the Michael J. Birck Fellowship from the University of Notre Dame (1990), the Harvey N. Davis Distinguished Teaching Award from the Stevens Institute of Technology (1997), the Faculty Early Career Development (CAREER) Award from the National Science Foundation (1999), and the University Scholar Award from the University of Illinois (2006–2009).

Hossein Javaherian (M'89) received the M.Sc. and Ph.D. degrees in control systems from Imperial College, London, U.K., in 1974 and 1978, respectively.

In 1979, he held a postdoctoral position at the Imperial College and was an Assistant Professor with the Department of Mechanical Engineering, Tehran University of Technology, Tehran, Iran. In 1985, he joined the General Motors (GM) Research Laboratories, Warren, MI. He is currently a GM Technical Fellow with the Powertrain Systems Research Laboratory, GM R&D Center, Warren, working on engine control algorithm development for future propulsion systems, with emphasis on emissions and fuel economy. He is the holder of 16 patents. He has published a number of journal and conference papers. He serves as an Editor for the *Journal of Optimal Control, Methods and Applications*. His research interests are adaptive and learning systems, nonlinear systems control, embedded systems, and computational intelligence.

Dr. Javaherian was the recipient of two McCuen Awards (1996 and 1998) and the prestigious "Boss" Kettering Award (2000) at General Motors for innovations in engine control and diagnostics.

Olesia Kovalenko received the M.S. degree in electrical and computer engineering from the University of Illinois, Chicago, in 2005.

She is currently a Software Developer with McMaster-Carr, Elmhurst, IL.

Ting Huang (S'06) received the B.S. degree in electrical engineering from Xiamen University, Xiamen, China, in 2000 and the M.S. degree in aerospace engineering from the Chinese Academy of Space Technology, Beijing, in 2003. He is currently working toward the Ph.D. degree in electrical and computer engineering in the Department of Electrical and Computer Engineering, University of Illinois, Chicago.

From 2003 to 2004, he was a Software Engineer with Beijing Infrared Technology Company, Beijing Institute of Control Engineering. His current research interests include swarm intelligence, neuroscience, and automotive engine control.