3 Model-Based Adaptive Critic Designs

SILVIA FERRARI
Duke University

ROBERT F. STENGEL
Princeton University

Editor's Summary: This chapter provides an overview of model-based adaptive critic designs, including background, general algorithms, implementations, and comparisons. The authors begin by introducing the mathematical background of model-reference adaptive critic designs. Various ADP designs such as Heuristic Dynamic Programming (HDP), Dual HDP (DHP), Globalized DHP (GDHP), and Action-Dependent (AD) designs are examined from both a mathematical and implementation standpoint and put into perspective. Pseudocode is provided for many aspects of the algorithms. The chapter concludes with applications and examples. For another overview perspective that focuses more on implementation issues read Chapter 4: Guidance in the Use of Adaptive Critics for Control. Chapter 15 contains a comparison of DHP with back-propagation through time, building a common framework for comparing these methods.

3.1 INTRODUCTION

Under the best of circumstances, controlling a nonlinear dynamic system so as to minimize a cost function is a difficult process because it involves the solution of a two-point boundary value problem. The necessary conditions for optimality, referred to as *Euler-Lagrange equations*, include the final time. Hence, they cannot be accessed until the trajectory has been completed. The most that one can hope for is to solve the problem off line using an iterative process, such as steepest descent [1], before the trajectory begins. Then, the optimal control history can be applied to the actual system. If there are no errors in system description or control history implementation and if there are no disturbing forces, the system will execute the optimal trajectory in the specified time interval. For all other cases, optimality of the actual control system is, at best, approximate.

Of course, the real problem is more complex because there are non-deterministic (or stochastic) effects. The initial condition may differ from its assumed value, the system model may be imperfect, and there may be external disturbances to the dynamic process. At a minimum, closed-loop (feedback) control is required to account for perturbations from the open-loop-optimal trajectory prescribed by prior calculations. Because perturbations excite modes of motion that may be lightly damped or even unstable, the feedback control strategy must either assure satisfactory stability about the nominally optimal trajectory or produce a neighboring-optimal trajectory. If perturbations are small enough to be adequately described by a local linearization of the dynamic model, if random disturbances have zero mean, and if the model itself is not too imprecise, a *linear-quadratic neighboring-optimal controller* provides stability and generates the neighboring-optimal trajectory [1].

If, in addition, there are uncertain errors in the measurements required for feedback control, or if the full state is not measured, then some adjustment to control strategy must be made to minimize the degrading effect of imperfect information. For small perturbations and zero-mean measurement errors, the problem is solved by concatenating an optimal perturbation-state estimator with the feedback control law, forming a *time-varying, linear-quadratic-Gaussian regulator*. Inputs to the optimal estimator include a possibly reduced set of measurements and known control inputs. The algorithm makes predictions from the model of perturbation dynamics and, with knowledge of disturbance and measurement-error statistics, it corrects the predictions using the measurements to form a *least-squares state estimate*. In such case, the *separation* and *certainty-equivalence principles* apply to the perturbation system: the controller and estimator can be designed separately, and the controller itself is the same controller that would have been applied to the deterministic (i.e., certain) system state [2]. With perfect measurements, the neighboring-optimal controller is the same with or without zero-mean disturbance inputs: it is certainty-equivalent.

Thus, we see that real-time, exact optimal control of an actual system in an uncertain environment is not strictly possible, though there are approximating solutions that may be quite acceptable. In the remainder of this chapter, we present an approach to approximate optimal control that is based on dynamic programming, so we briefly relate the prior discussion to this alternative point of view [3, 4].

While there is no stochastic equivalent to the Euler-Lagrange equations, there is an equivalent stochastic dynamic programming formulation that extends to adaptive critic designs [5]. This approach seeks to minimize the *expected value* of the cost function with respect to the control, conditioned on knowledge of the system, its state, and the probability distributions of uncertainties [1]. For simplicity, this chapter only deals with adaptive critic designs that are deterministic.

Dual control is one method that offers a systematic solution to the problem of approximate optimization [6–8]. It optimizes a value function, or cost-to-go, composed of three parts that are associated with nominal optimal control, cautious feedback control, and probing feedback control. The first component represents the cost associated with optimization using present knowledge of the system and its trajectory. The second component represents cost associated with the effects of uncertain inputs and measurements. The third cost is associated with control inputs that improve knowledge of the system's unknown parameters. A numerical search over the present value of the control minimizes a stochastic *Hamilton-Jacobi-Bellman* (HJB) *equation* [6] providing a basis for real-time, approximate optimal control.

If the final time is finite and minimizing a terminal cost is important, the entire remaining trajectory must be evaluated to determine the future cost. If the final time is infinite, then the terminal cost is of no consequence, and the future trajectory is propagated far enough ahead that additional steps have negligible impact on the control policy. In some applications, it may be sufficient to propagate just one step ahead to approximate the future cost, greatly reducing the required computation. Thus, there is a close relationship between dual control and other *receding-horizon* or *predictive-adaptive* control approaches [9, 10].

Adaptive critic controllers provide an alternative, practical approach to achieving optimality in the most general case. A distinguishing feature of this approach is that the optimal control law and value function are modelled as parametric structures (e.g., computational neural networks), whose shapes are improved over time by solving the *Recurrence Relation of Dynamic Programming*. The methods use step-ahead projection to estimate the future cost. When the final time approaches infinity and the system dynamics are unchanging, the methods converge to the optimal control law. In practice, the parametric structures adapt to changing system parameters, including system failures, without explicit parameter identification. This requires an approximate model of the plant dynamics that admits satisfactory estimates of the future cost. In separate work [11, 12], we have shown how to pre-train neural networks to give these parametric structures good starting topologies (or *initializations*) prior to on-line learning by an adaptive critic approach.

3.2 MATHEMATICAL BACKGROUND AND FOUNDATIONS

This section introduces the foundations of model-reference adaptive critic designs, placing them within the framework of optimal control. The on-line solution of infinite-horizon problems, with information that becomes available incrementally over time, is emphasized. In optimal control problems the objective is to devise a strategy of action, or control law, that optimizes a desired performance metric or cost. Two well known solution approaches are the calculus of variations, involving the Euler-Lagrange equations, and backward dynamic programming. They are reviewed here because they both have strong ties with the adaptive critic approach.

Adaptive critic designs utilize two parametric structures called the *actor* and the *critic*. The actor consists of a parameterized control law. The critic approximates a value-related function and captures the effect that the control law will have on the future cost. At any given time the critic provides guidance on how to improve the control law. In return, the actor can be used to update the critic. An algorithm that successively iterates between these two operations converges to the optimal solution over time. More importantly, this adaptive critic algorithm can be used to design control systems that improve their performance on line, subject to actual plant dynamics.

3.2.1 Solution of the Optimal Control Problem

The objective is to determine a stationary optimal control law that minimizes a performance measure *J* expressed by a scalar, time-invariant integral function of the state and controls and by a scalar terminal cost,

$$J = \varphi[\mathbf{x}(t_f)] + \sum_{t_k=t_0}^{t_f-1} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)], \tag{3.1}$$

subject to the dynamic constraint imposed by plant dynamics. This *cost function* represents the cost of operation as it accrues from the initial time t_0 to the final time t_f. The integrand or *Lagrangian* $\mathcal{L}[\cdot]$ is the cost associated with one time increment; it also is referred to as utility or reward in the financial and operation research literature, where the objective typically is to maximize the overall performance (Eq. (3.1)). The plant dynamics are discrete, time-invariant, and deterministic, and they can be modelled by a difference equation of the form,

$$\mathbf{x}(t_{k+1}) = \mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k)], \tag{3.2}$$

with equally spaced time increments in the interval $t_0 \leq t_k \leq t_f$, and initial condition $\mathbf{x}(t_0)$. $\mathbf{x}$ is the $n \times 1$ plant state and $\mathbf{u}$ is the $m \times 1$ control vector. For simplicity, it also is assumed that the state is fully observable and that perfect output measurements are available.

The control law is assumed to be solely a function of the state

$$\mathbf{u} = \mathbf{c}(\mathbf{x}). \tag{3.3}$$

The control functional $\mathbf{c}(\cdot)$ may contain functions of its arguments such as integrals and derivatives, and the optimal form is denoted by $\mathbf{c}^*(\mathbf{x})$. At any moment in time, t_k, the cost that is going to accrue from that moment onward can be expressed by a *value function*,

$$V = \varphi[\mathbf{x}(t_f)] + \sum_{t_k}^{t_f-1} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)] = V[\mathbf{x}(t_k), \mathbf{c}(\mathbf{x})], \tag{3.4}$$

which depends on the present value of the state, $\mathbf{x}(t_k)$, and on the chosen control law $\mathbf{c}(\mathbf{x})$. Therefore, $V[\mathbf{x}(t_k), \mathbf{c}(\mathbf{x})]$ also can be written in abbreviated form as $V[\mathbf{x}_k, \mathbf{c}]$, where $\mathbf{x}_k \equiv \mathbf{x}(t_k)$.

One approach to the optimization of Eq. (3.1) subject to Eq. (3.2) is to augment the cost function by the dynamic equation, recasting the problem in terms of the

Hamiltonian

$$\mathcal{H}(\mathbf{x}_k, \mathbf{u}_k, \lambda_k) = \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)] + \lambda^T(t_k)\mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k)]. \tag{3.5}$$

λ is a *costate* or *adjoint* vector that contains Lagrange multipliers [1] and represents the cost sensitivity to state perturbations on the optimal trajectory; it can be shown that $\lambda(t_f) = \partial\varphi/\partial\mathbf{x}|_{t_f}$ (with the gradient defined as a column vector). When the final time is fixed, necessary conditions for optimality, the Euler-Lagrange equations, can be obtained by differentiating the Hamiltonian with respect to the state and the control [1]. Thus, the dynamic optimization problem is reduced to a *two-point boundary value problem*, where the state and the adjoint vector are specified at the initial and final time, respectively. Ultimately, necessary and sufficient conditions for optimality are provided by Pontryagin's Minimum Principle, stating that on the optimal trajectory $\mathcal{H}$ must be stationary and convex

$$\mathcal{H}^* = \mathcal{H}(\mathbf{x}_k^*, \mathbf{u}_k^*, \lambda_k^*) \leq \mathcal{H}(\mathbf{x}_k^*, \mathbf{u}_k, \lambda_k^*). \tag{3.6}$$

Another approach to solving the optimal control problem consists of *imbedding* [13] the minimization of the cost function, Eq. (3.1), in the minimization of the value function, Eq. (3.4). When the system is in an admissible state $\mathbf{x}_k$, the value function, that is, the cost of operation from the instant t_k to the final time t_f, can be written as

$$V(\mathbf{x}_k, \mathbf{u}_k, \ldots, \mathbf{u}_{f-1}) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{f-1}). \tag{3.7}$$

$\mathbf{x}_{k+1}$ depends on $\mathbf{x}_k$ and $\mathbf{u}_k$ through Eq. (3.2). All subsequent values of the state can be determined from $\mathbf{x}_k$ and from the chosen control history, $\mathbf{u}_k, \ldots, \mathbf{u}_{f-1}$. Therefore, the cost of operation from t_k onward can be minimized with respect to all future values of the control

$$V^*(\mathbf{x}_k^*) = \min_{\mathbf{u}_k,\ldots,\mathbf{u}_{f-1}} \{\mathcal{L}(\mathbf{x}_k^*, \mathbf{u}_k) + V(\mathbf{x}_{k+1}^*, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{f-1})\}. \tag{3.8}$$

It follows that the optimal value function depends on the present state of the system and is independent of any prior history. Suppose a policy is optimal over the time interval $(t_f - t_k)$. Then, by the *Principle of Optimality* [14], whatever the initial state $(\mathbf{x}_k)$ and decision $(\mathbf{u}_k)$ are, the remaining decisions also must constitute an optimal policy with regard to the state $\mathbf{x}_{k+1}$, that is,

$$V^*(\mathbf{x}_k^*) = \min_{\mathbf{u}_k} \{\mathcal{L}(\mathbf{x}_k^*, \mathbf{u}_k) + V^*(\mathbf{x}_{k+1}^*)\}. \tag{3.9}$$

The value function can be minimized solely with respect to the present value of the control, $\mathbf{u}_k$, provided the future cost of operation, $V(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{f-1})$, is optimal (in which case, it only depends on the next value of the state on the optimal trajectory, that is, $\mathbf{x}_{k+1}^*$). Equation (3.9) constitutes the Recurrence Relation of Dynamic Programming. The optimal value function can be interpreted as the minimum cost for the time period that remains after a time t_k in a process that began

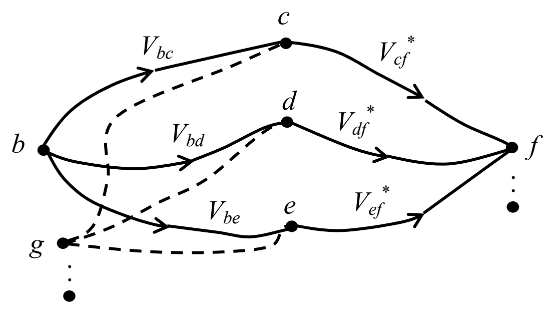


Fig. 3.1 Backward or discrete dynamic programming approach.

at t_0, that is, $(t_f - t_k)$. Alternatively, it can be viewed as the minimum cost for a process that begins at t_k and ends at t_f.

The recurrence relation can be used backwards in time, starting from t_f, to obtain an approximate solution to the exact optimal control history. This approach is referred to as *backward dynamic programming* (BDP). It discretizes the state space and makes a direct comparison of the cost associated with all feasible trajectories, guaranteeing a solution to the global optimal control problem. The space of admissible solutions is reduced by examining a multi-stage decision process as a sequence of one-stage processes. This approach typically is too computationally expensive for higher dimensional systems, with a large number of stages (or time increments). The required multiple generation and expansion of the state and the storage of all optimal costs lead to a number of computations that grows exponentially with the number of state variables. This phenomenon commonly is referred to as the "curse of dimensionality" or "expanding grid" [13].

Forward dynamic programming (FDP) and *temporal difference* methods use incremental optimization combined with a *parametric structure* to reduce the computational complexity associated with evaluating the cost [15–17]. A parametric structure consists of a functional relationship whose adjustable parameters allow it to approximate different mappings. Adaptive critic designs (ACD) derive from the forward-dynamic-programming approach, also called *approximate dynamic programming*.

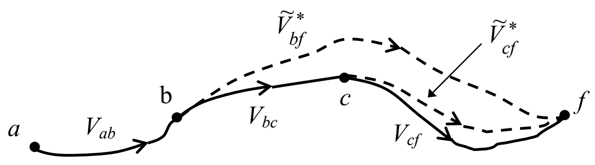


Fig. 3.2 Forward or approximate dynamic programming approach.

For comparison, the backward and forward DP approaches are illustrated for the last two stages of a hypothetical process in Figure 3.1 and Figure 3.2, respectively. The backward approach begins by considering the last stage. Since the optimal final state is not yet known, the following procedure has to be performed for all admissible values of the final state. The optimal paths are computed for all feasible intermediate-state values, c, d, and e, thereby producing the optimal costs V^*_{cf}, V^*_{df}, and V^*_{ef}, respectively. By the principle of optimality these paths also are optimal for the last stage of the optimal trajectories that go from b to f through the respective state values, for example, $V^*_{b(c)f} = V_{bc} + V^*_{cf}$. Thus, if the last-stage's costs, V^*_{cf}, V^*_{df}, and V^*_{ef}, are stored, then the total costs $V^*_{b(c)f}$, $V^*_{b(d)f}$, and $V^*_{b(e)f}$ can be compared to find the optimal path from b to f. For a process with more than two stages, the state b would be unknown. Hence, the same computation would be carried out for all possible state values, such as g in Figure 3.1, in order to determine V^*_{bf}, V^*_{gf}, and so on. The algorithm terminates when the initial stage is reached, where the state is known from the initial conditions.

Unlike backward DP, forward DP algorithms progress forward in time, and they approximate the optimal control and future cost by considering only the current value of the state. Suppose a is the initial state of a two-stage process or, equivalently, the state of an on-going process at the second-to-last stage, as outlined in Figure 3.2. Then, a is known and the cost V_{ab} can be computed from the Lagrangian for a chosen control policy. The optimal cost over all future stages, V^*_{bf}, is predicted by a function approximator or parametric structure. Hence, the sum $(V_{ab} + \tilde{V}^*_{bf})$ can be minimized with respect to the present value of the control, according to the recurrence relation of dynamic programming (Eq. (3.9)). At the next stage b-c this procedure is repeated. The cost approximation, denoted by $\tilde{V}$, has been improved based on the information gathered during the first stage. Therefore, the next path, from c to f, is closer to the optimal trajectory.

Adaptive critic designs reproduce the most general solution of FDP by deriving recurrence relations for the optimal policy, the cost, and, possibly, their derivatives. The goal is to overcome the curse of dimensionality, while ensuring convergence to a near-optimal solution over time. The following section introduces the basic adaptive-critic algorithm and the related proof of convergence.

3.2.2 Adaptive Critic Algorithm

Adaptive critic designs are based on an algorithm that cycles between a *policy-improvement routine* and a *value-determination operation*. At each optimizing cycle, indexed by ℓ, the algorithm approximates the optimal control law (Eq. (3.3)) and value function (Eq. (3.8)) based on the state $\mathbf{x}_k$. We distinguish between iteration over k, which represents the passage of time in the dynamic process, and over ℓ, which relates to the search for an optimal solution. In off-line optimization, an ensemble of state vectors including all possible values of $\mathbf{x}_k$ is used during every cycle. In this case, the entire process can be simulated from t_0 to t_f. In on-line optimization, one cycle is carried out at each time step based on a single value of $\mathbf{x}_k$, that is, the actual state, such that $k = \ell$.

For simplicity, the following discussion is limited to the infinite-horizon optimal control problem, where the final time t_f tends to infinity. In this case, the cost function to be minimized takes the form,

$$J = \lim_{t_f \to \infty} \sum_{t_k = t_0}^{t_f - 1} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)], \tag{3.10}$$

and the terminal cost can be set equal to zero [1].

The projected cost $V(\mathbf{x}_{k+1}, \mathbf{c})$ that a sub-optimal control law $\mathbf{c}$ would incur from t_k to t_f can be estimated from Eq. (3.7). Using the projected cost as a basis, the control-law and value-function approximations can be improved during every optimization cycle, ℓ. Over several cycles, the policy-improvement routine generates a sequence of sub-optimal control laws $\{\mathbf{c}_\ell \,|\, \ell = 0, 1, 2, \ldots\}$; the value-determination operation produces a sequence of sub-optimal value functions $\{V_\ell \,|\, \ell = 0, 1, 2, \ldots\}$. The algorithm terminates when $\mathbf{c}_\ell$ and V_ℓ have converged to the optimal control law and value function, respectively. The proof of convergence [5] is outlined in the Appendix to this chapter.

Policy-improvement Routine Given a value function $V(\cdot, \mathbf{c}_\ell)$ corresponding to a control law $\mathbf{c}_\ell$, an *improved* control law, $\mathbf{c}_{\ell+1}$, can be obtained as follows:

$$\mathbf{c}_{\ell+1}(\mathbf{x}_k) = \arg \min_{\mathbf{u}_k} \{\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}_\ell)\}, \tag{3.11}$$

such that $V(\mathbf{x}_k, \mathbf{c}_{\ell+1}) \leq V(\mathbf{x}_k, \mathbf{c}_\ell)$, for any value of $\mathbf{x}_k$. Furthermore, the sequence of functions $\{\mathbf{c}_\ell \,|\, \ell = 0, 1, 2, \ldots\}$ converges to the optimal control law, $\mathbf{c}^*(\mathbf{x})$.

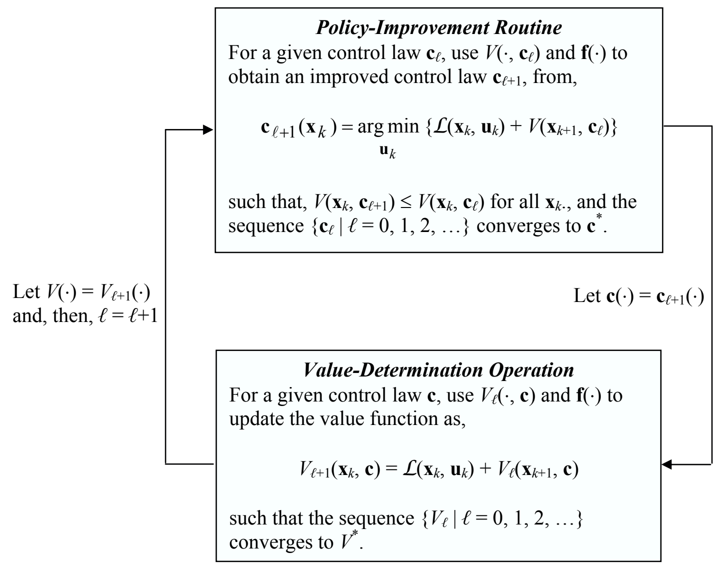
Value-determination Operation Given a control law $\mathbf{c}$, the value function can be updated according to the following rule:

$$V_{\ell+1}(\mathbf{x}_k, \mathbf{c}) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V_\ell(\mathbf{x}_{k+1}, \mathbf{c}), \tag{3.12}$$

such that the sequence $\{V_\ell \,|\, \ell = 0, 1, 2, \ldots\}$, in concert with the policy-improvement update, converges to the optimal value function $V^*(\mathbf{x})$.

The adaptive critic algorithm successively iterates between these two updates, as illustrated in Figure 3.3. The algorithm can begin in either box. The speed of convergence is dependent on the suitability of the initialized control law and value function $\mathbf{c}_0$ and V_0, as well as on the details of the adjustment rules. A simple example is given in Section 3.3.5. More detailed examples are given in [11, 18].

In the ℓ^{th} cycle, $\mathbf{c}_{\ell+1}$ is determined by the policy-improvement routine and can be used as the control law $\mathbf{c}$ in the value-determination operation. Then, in the $(\ell+1)^{\text{th}}$ cycle, the updated value function $V_{\ell+1}$ can be used as $V(\cdot, \mathbf{c}_{\ell+1})$ in the policy-


Policy-Improvement Routine

For a given control law $\mathbf{c}_\ell$, use $V(\cdot, \mathbf{c}_\ell)$ and $\mathbf{f}(\cdot)$ to obtain an improved control law $\mathbf{c}_{\ell+1}$, from,

$$\mathbf{c}_{\ell+1}(\mathbf{x}_k) = \arg\min_{\mathbf{u}_k} \{\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}_\ell)\}$$

such that, $V(\mathbf{x}_k, \mathbf{c}_{\ell+1}) \leq V(\mathbf{x}_k, \mathbf{c}_\ell)$ for all $\mathbf{x}_k$., and the sequence $\{\mathbf{c}_\ell \mid \ell = 0, 1, 2, \ldots\}$ converges to $\mathbf{c}^*$.

Let $V(\cdot) = V_{\ell+1}(\cdot)$ and, then, $\ell = \ell+1$

Let $\mathbf{c}(\cdot) = \mathbf{c}_{\ell+1}(\cdot)$

Value-Determination Operation

For a given control law $\mathbf{c}$, use $V_\ell(\cdot, \mathbf{c})$ and $\mathbf{f}(\cdot)$ to update the value function as,

$$V_{\ell+1}(\mathbf{x}_k, \mathbf{c}) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V_\ell(\mathbf{x}_{k+1}, \mathbf{c})$$

such that the sequence $\{V_\ell \mid \ell = 0, 1, 2, \ldots\}$ converges to V^*.

Fig. 3.3 Adaptive critic iteration cycle, after [5].

improvement routine. The algorithm terminates when two successive cycles produce the same control law, or a suitably close one. If the policy-improvement routine finds two control laws of equivalent performance, such that $[\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}_{\ell+1})] = [\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}_\ell)]$, but $\mathbf{c}_{\ell+1} \neq \mathbf{c}_\ell$, the old one ($\mathbf{c}_\ell$) always is chosen.

The adaptive critic algorithm has the following important properties [5]:

1. The algorithm improves its choice of control law during every cycle ℓ, such that each succeeding control functional has a better overall performance (lower cost) than the previous one.

2. The algorithm improves its choice of value function during every cycle ℓ, such that, for a given state, each succeeding value functional is smaller than the previous one.

3. The algorithm terminates on the optimal control law, provided there exists one. In other words, if there exists a superior control law it is found before the adaptive critic algorithm terminates.

4. The algorithm solves the optimal control problem without backward iteration in time.

Properties 1 through 4 are demonstrated in the Appendix to this chapter. Property 4 is fundamental to the solution approach. The following section discusses on-line implementations of the adaptive critic algorithm.

3.3 ADAPTIVE CRITIC DESIGN AND IMPLEMENTATION

The adaptive critic algorithm can be used to determine the optimal control law for a dynamic process on or off line. In the latter case, control design is based on the model of the system to be controlled, and the required values of the state are produced by simulation. The adaptive critic cycle generates a control law $\mathbf{c}_\ell$ and a value function V_ℓ for an ensemble of state values. The procedure is repeated for $\ell = 0, 1, 2, \ldots$, until the algorithm terminates producing $\mathbf{c}^*$ and V^*. Subsequently, $\mathbf{c}^*$ can be implemented to operate the system in Eq. (3.2) in an optimal fashion.

The adaptive critic algorithm also can be used on-line. Unlike the Euler-Lagrange equations and backward DP, the computation required by each iteration depends only on the present state, $\mathbf{x}_k$, and does not involve the final conditions (e.g., $\lambda(t_f)$). In on-line implementations state-vector values become available progressively in time as $\{\mathbf{x}_k \mid k = 0, 1, 2, \ldots\}$. Then, the indices ℓ and k in Figure 3.3 coincide at every algorithmic cycle and moment in time. Although this procedure can be adopted for systems whose dynamic equation (Eq. (3.2)) is known precisely, it is particularly valuable when the system dynamics are not fully known or subject to change. Then, the adaptive critic algorithm can be implemented on-line, while the system is operating.

In on-line implementations, the control law $\mathbf{c}_\ell$, generated by the ℓ^{th} algorithmic cycle, is applied at time t_k. Assuming that the plant is fully observable, the *actual* value of the present state, $\mathbf{x}_k$, is determined from available output measurements. The next value of the state, $\mathbf{x}_{k+1}$, is *predicted* through the model in Eq. (3.2). Hence, the control law and value function can be improved at every moment in time t_k based on the observed value of $\mathbf{x}_k$. The optimal control law can be determined on-line for systems whose true dynamic characteristics are revealed only during actual operation. Since the adaptive critic algorithm improves upon any sub-optimal control law (say $\mathbf{c}_{\ell-1}$), it usually is convenient to implement $\mathbf{c}_\ell$ as soon as it is generated. The system converges to optimal performance provided its dynamics remain unchanged. Otherwise, the algorithm continues to improve the control and value functions incrementally, subject to the varying dynamics. An underlying assumption is that the dynamic variations occur on a sufficiently large time scale with respect to the algorithmic cycles.

3.3.1 Overview of Adaptive Critic Designs

Adaptive critic designs implement approximate dynamic programming through recurrence relations for the control law, the value function and, possibly, their derivatives. The basic architectures can be classified in four categories [19, 20]:

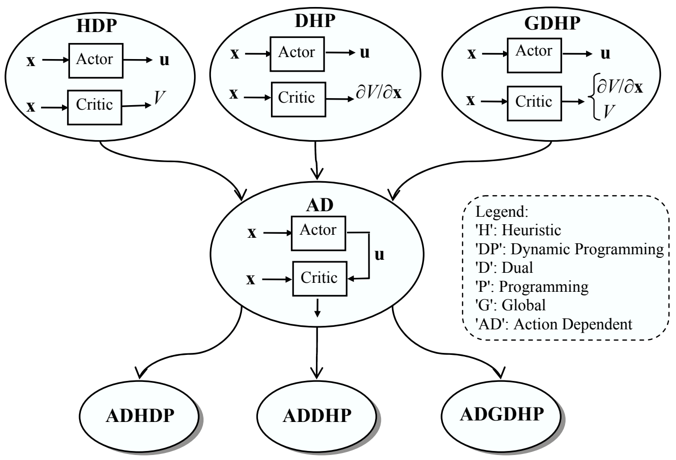


Fig. 3.4 Overview of well known adaptive critic designs.

1. Heuristic Dynamic Programming (HDP)
2. Dual Heuristic Dynamic Programming (DHP)
3. Globalized Dual Heuristic Dynamic Programming (GDHP)
4. Action-dependent (AD) designs

The distinguishing characteristics of each category are described in this section, and are summarized in Figure 3.4.

The salient feature of these designs is that they can be implemented on-line. The exact system dynamics need not be known *a priori* because the computations can be carried out while the system is operating, based on the observable state vector. Initially, the control law generated is sub-optimal. But, it is guaranteed to improve with every iteration cycle by the policy-improvement routine. The goal is to overcome the curse of dimensionality through function approximation, while approaching the optimal solution over time. The main challenge is to achieve satisfactory convergence to the optimal or *near*-optimal solution in a small number of cycles.

The four types of adaptive critic designs have been developed to accelerate convergence to the optimal solution. While they all are based on the adaptive critic algorithm (Section 3.2.2), they differ in what functionals they set out to approximate. The basic iteration cycle (Figure 3.3) involves the optimal control law and value function. Thus the most basic form of adaptive critic design, Heuristic Dynamic Programming (HDP), uses the parametric structure called actor (or action network) to approximate the control law, and another parametric structure called critic (or critic network) to approximate the value function. The role of the critic has been compared with that of "reinforcement" in psychology, and to the "utility function" in utility theory [21]. The critic provides the actor with a performance measure for its present actions by anticipating and evaluating future events. In practice, HDP converges slowly because the parameters of this architecture ultimately are updated based on a scalar evaluation function, that is, V.

An alternative approach referred to as Dual Heuristic Programming (DHP) was first proposed in [21, 22]. DHP uses the critic to approximate the derivatives of the value function with respect to the state, that is, $\partial V/\partial \mathbf{x}_k$. It correlates the parameters of the architecture to a larger number of dependent variables. The actor is used to approximate the control law, as in all other adaptive critic designs. An algorithm for updating the DHP functionals can be obtained from the policy-improvement routine and the value-determination operation, as explained in Section 3.3.3. In applications [23, 24], the DHP algorithm has been shown to find the optimal solution more rapidly (with less iteration cycles) than HDP. However, due to the use of derivative information, the relationships for updating the control and value-derivative functionals are more involved. Furthermore, since the DHP critic approximates a vector functional, the problem of function approximation is more challenging.

Many other methodologies have been proposed to alleviate some of the difficulties mentioned above. For example, Globalized Dual Heuristic Programming (GDHP) combines the advantages of both the HDP and DHP architectures [16, 25, 26]. In GDHP the critic approximates both the value function and its derivatives. Action-dependent (AD) versions of these approaches use a parametric structure, again referred to as critic, to approximate a value function that depends *explicitly* on the control (as shown in Figure 3.4). The motivation behind this [27] and other methodologies (e.g., [28]) is to achieve faster convergence to the optimal solution and to simplify the problem of function approximation.

In summary, an adaptive critic architecture consists of a parametric structure, the critic, which approximates the value function (Eq. (3.4)) or a closely related quantity, and another parametric structure, the actor, which approximates the control law. Once the functions to be approximated are chosen, an algorithm that updates them at every cycle can be obtained from the policy-improvement routine and the value-determination operation in Section 3.2.2. The objective is to produce two sequences of functions that eventually converge to the optimal solution.

3.3.2 Function Approximation

Three functionals play a key role in all adaptive critic designs. They are:

1. The control law
2. The value function
3. The model of the system to be controlled

The objective of the adaptive critic algorithm is to progressively improve its approximation of the optimal control law $\mathbf{c}^*(\mathbf{x}_k)$ and value function $V^*(\mathbf{x}_k)$. A model of the system (such as Eq. (3.2)) is required in order to predict the next value of the state $\mathbf{x}_{k+1}$ at every iteration cycle ℓ (Figure 3.3). Hence, designs based on this algorithm often are referred to as *model-based adaptive critics*. When the system's behavior is partly unanticipated the model provides an approximation to its dynamics [19].

The sequences of functions generated by the adaptive critic algorithm are synthesized by a convenient parametrization or function approximator that captures the relationship between the dependent and independent variables. Without the use of an appropriate parametric structure each function, say $\mathbf{u}_k = \mathbf{c}_\ell(\mathbf{x}_k)$, would need to be represented by a lookup table. For large state and control spaces it is infeasible to carry out the required computation (Eqs. (3.11) and (3.12)) with lookup-table representations of $\mathbf{c}_\ell$, V_ℓ, $\mathbf{c}_{\ell+1}$, and $V_{\ell+1}$. Moreover, it usually is desirable to obtain a functional representation of the optimal control and value functions once the algorithm converges. Therefore, a parametric structure that, ultimately, is capable of approximating the optimal solution must be chosen.

A *suitable function approximator* or parametric structure has the following characteristics:

- It is differentiable;
- It approximates the desired function with less complexity than a lookup-table representation;
- It is supplied with apposite algorithms for computing the parameter values;
- It is capable of approximating the optimal shape of the function within the desired accuracy.

There are many theoretical and practical issues associated with the problem of function approximation. The most important ones pertain to the effect that function approximation has on convergence. In order for the original proof (in the Appendix) to hold, this process must be consistent with the policy-improvement routine and value-determination operation in Section 3.2.2. Additional details can be found in [29–31].

We assume that the actor is a suitable parametric structure $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})$, where $\mathbf{a}$ is a vector of parameters to be optimized by the policy-improvement routine. The function approximator's parameters are updated at every cycle, ℓ, such that $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_\ell) \approx \mathbf{c}_\ell(\mathbf{x}_k)$. Then, the computational complexity of the adaptive critic algorithm is reduced at every cycle. Upon termination of the algorithm (when $\mathbf{c}_\ell \rightarrow \mathbf{c}^*$), the optimal control is readily approximated as $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}^*) \approx \mathbf{c}^*(\mathbf{x}_k)$.

The parameters of the function approximator must be determined at every cycle of the algorithm without compromising convergence. This process, in itself, may require one or more iterations. The following sections describe how function approximation can be combined with the policy-improvement routine and the value-determination operation (Section 3.2.2) to produce heuristic and dual-heuristic adaptive critic designs.

3.3.3 Derivation of the Heuristic and Dual-Heuristic Dynamic Programming Algorithms

In on-line implementations of adaptive critic designs the control and value function approximations are updated incrementally over time. With one available value of the state, $\mathbf{x}_k$, the minimization problem in Eq. (3.11) can be solved by computing the stationary point of the scalar function $V(\mathbf{x}_k, \mathbf{c}_\ell) \equiv [\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}_\ell)]$. This point is defined as the value of $\mathbf{u}_k$ that satisfies the *optimality condition*,

$$\frac{\partial V(\mathbf{x}_k, \mathbf{c}_\ell)}{\partial \mathbf{u}_k} = \frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} + \left[\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\right]^T \frac{\partial V(\mathbf{x}_{k+1}, \mathbf{c}_\ell)}{\partial \mathbf{x}_{k+1}} = 0, \tag{3.13}$$

along with the following *convexity condition*:

$$\frac{\partial^2 V(\mathbf{x}_k, \mathbf{c}_\ell)}{\partial \mathbf{u}_k^2} > \mathbf{0}. \tag{3.14}$$

This condition requires a positive definite Hessian matrix [32]. The gradient of a scalar function [32], for example, $\partial V/\partial \mathbf{x}_k$, is defined as a column vector. The stationary control value can be used to improve upon the control law approximation $\mathbf{c}_\ell$ at the moment in time t_k. Rules for updating the critic on-line are similarly derived, as shown in the following sections.

Heuristic Dynamic Programming (HDP) Algorithm

In Heuristic Dynamic Programming, the actor is a parametric structure $\tilde{\mathbf{c}}$ with parameters $\mathbf{a}$ that is used to approximate the control law. The critic is a parametric structure $\tilde{V}$, with parameters $\mathbf{w}$, that is used to approximate the value function, that is:

$$\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_\ell) \approx \mathbf{c}_\ell(\mathbf{x}_k) \tag{3.15}$$

$$\tilde{V}(\mathbf{x}_k, \mathbf{w}_\ell) \approx V_\ell(\mathbf{x}_k, \tilde{\mathbf{c}}) \tag{3.16}$$

The value function estimate is assumed to be a *composite function* of $\mathbf{x}_k$, since $\mathbf{u}_k = \mathbf{c}(\mathbf{x}_k)$ and system dynamics are known (Eq. (3.2)). The state is its sole input, as this is the action-independent version of HDP. The optimal value function also is a function solely of the state, as shown in Eq. (3.8). When the chosen parametric structures are differentiable, the optimality condition (Eq. (3.13)) generates a sequence of successively improving control laws $\{\mathbf{c}_\ell \,|\, \ell = 0, 1, 2, \ldots\}$. Concurrently, the value-determination operation (Eq. 3.12) generates a sequence of successively improving value functions $\{V_\ell \,|\, \ell = 0, 1, 2, \ldots\}$. An HDP algorithm that iterates between these two operations is shown below. At every cycle ℓ, the actor parameters are updated based on the improved control law $\mathbf{c}_{\ell+1}$, and the critic parameters are updated based on the improved value function $V_{\ell+1}$.

Heuristic-Dynamic-Programming Actor Update Suppose a control-law approximator, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_\ell)$, and a corresponding value-function approximator, $\tilde{V}(\cdot, \mathbf{w}_\ell)$, are given. Then, an improved control-law approximator $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ can be obtained by computing a desired control vector $\mathbf{u}_k^{\mathrm{D}}$, such that,

$$\begin{aligned}\left.\frac{\partial \tilde{V}(\mathbf{x}_k, \mathbf{w}_\ell)}{\partial \mathbf{u}_k}\right|_{\mathbf{u}_k=\mathbf{u}_k^{\mathrm{D}}} &= \left.\left[\frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} + \left(\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\right)^T \frac{\partial \tilde{V}(\mathbf{x}_{k+1}, \mathbf{w}_\ell)}{\partial \mathbf{x}_{k+1}}\right]\right|_{\mathbf{u}_k=\mathbf{u}_k^{\mathrm{D}}} \\ &= 0, \end{aligned} \tag{3.17}$$

and the matrix $[\partial^2 \tilde{V}(\mathbf{x}_k, \mathbf{w}_\ell)/\partial \mathbf{u}_k^2]|_{\mathbf{u}_k=\mathbf{u}_k^{\mathrm{D}}}$ is positive definite.

The improved actor parameters, $\mathbf{a}_{\ell+1}$, can be computed as follows:

$$\mathbf{a}_{\ell+1} = \arg\min_{\mathbf{a}} \{\mathbf{u}_k^{\mathrm{D}} - \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})\}, \tag{3.18}$$

where $\mathbf{u}_k^{\mathrm{D}} \equiv \mathbf{c}_{\ell+1}(\mathbf{x}_k)$, therefore $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1}) \approx \mathbf{c}_{\ell+1}(\mathbf{x}_k)$.

Heuristic-Dynamic-Programming Critic Update Given the improved control-law approximator $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ and the value-function approximator $\tilde{V}(\cdot, \mathbf{w}_\ell)$, an improved value-function approximator $\tilde{V}(\cdot, \mathbf{w}_{\ell+1})$ can be obtained by computing its desired value,

$$V_k^{\mathrm{D}} \equiv V_{\ell+1}(\mathbf{x}_k, \tilde{\mathbf{c}}) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + \tilde{V}(\mathbf{x}_{k+1}, \mathbf{w}_\ell), \tag{3.19}$$

with $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$ and, subsequently, by determining the improved critic parameters,

$$\mathbf{w}_{\ell+1} = \arg\min_{\mathbf{w}} \{V_k^{\mathrm{D}} - \tilde{V}(\mathbf{x}_k, \mathbf{w})\}, \tag{3.20}$$

such that, $\tilde{V}(\mathbf{x}_k, \mathbf{w}_{\ell+1}) \approx V_{\ell+1}(\mathbf{x}_k, \tilde{\mathbf{c}})$.

There exist other approaches for updating the actor based on the optimality condition. One possibility is to obtain the improved actor parameters by minimizing the right-hand side of Eq. (3.17) directly with respect to $\mathbf{a}$:

$$\mathbf{a}_{\ell+1} = \arg\min_{\mathbf{a}} \left\{ \left.\left[\frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} + \left(\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\right)^T \frac{\partial \tilde{V}(\mathbf{x}_{k+1}, \mathbf{w}_\ell)}{\partial \mathbf{x}_{k+1}}\right]\right|_{\mathbf{u}_k=\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})} \right\}. \tag{3.21}$$

Although this method reduces the actor update to one step, it may be computationally less efficient and less reliable.

Dual Heuristic Dynamic Programming (DHP) Algorithm

In Dual Heuristic Dynamic Programming, the critic approximates a sequence of functionals, $\{\lambda_\ell \mid \ell = 0, 1, 2, \ldots\}$, which ultimately converges to the derivative of the optimal value function with respect to the state, defined as:

$$\lambda^*(\mathbf{x}_k^*) \equiv \frac{\partial V^*(\mathbf{x}_k^*)}{\partial \mathbf{x}_k^*}. \tag{3.22}$$

In essence, the role of the critic is to generate the costate or adjoint vector in the *Hamilton-Jacobi-Bellman* (HJB) equation, estimating the cost sensitivity to state perturbations. In the HJB approach, the Hamiltonian (Eq. (3.5)) is defined by adjoining the dynamic equation (Eq. (3.2)) to the Lagrangian by $\partial V/\partial \mathbf{x}_k$.

A parametric structure $\tilde{\mathbf{c}}$ with parameters $\mathbf{a}$, the actor, is used to approximate the control law. Another parametric structure $\tilde{\lambda}$ with parameters ω, the critic, is used to approximate the derivative of the value function with respect to the state:

$$\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_\ell) \approx \mathbf{c}_\ell(\mathbf{x}_k), \tag{3.15}$$

$$\tilde{\lambda}(\mathbf{x}_k, \omega_\ell) \approx \frac{\partial V_\ell(\mathbf{x}_k, \tilde{\mathbf{c}})}{\partial \mathbf{x}_k} \equiv \lambda_\ell(\mathbf{x}_k, \tilde{\mathbf{c}}). \tag{3.23}$$

When the chosen parametric structures are differentiable, the optimality condition (Eq. (3.13)) can be used to generate a sequence of successively-improving control laws $\{\mathbf{c}_\ell \mid \ell = 0, 1, 2, \ldots\}$.

The value-determination operation (Eq. (3.12)) is differentiated with respect to the state to obtain a recurrence relation for the function λ:

$$\begin{aligned} \frac{\partial V_{\ell+1}(\mathbf{x}_k, \mathbf{c})}{\partial \mathbf{x}_k} \quad &\equiv \quad \lambda_{\ell+1}(\mathbf{x}_k, \mathbf{c}) = \frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} + \left[\frac{\partial \mathbf{c}(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right]^T \frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} \\ &+ \quad \left[\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k}\right]^T \frac{\partial V_\ell(\mathbf{x}_{k+1}, \mathbf{c})}{\partial \mathbf{x}_{k+1}} \\ &+ \quad \left[\frac{\partial \mathbf{c}(\mathbf{x}_k)}{\partial \mathbf{x}_k}\right]^T \left[\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\right]^T \frac{\partial V_\ell(\mathbf{x}_{k+1}, \mathbf{c})}{\partial \mathbf{x}_{k+1}}. \end{aligned} \tag{3.24}$$

Equation (3.24) generates the sequence of successively-improving value function derivatives $\{\lambda_\ell \mid \ell = 0, 1, 2, \ldots\}$, as the control law sequence is generated. A DHP algorithm that, at every step ℓ, updates the actor parameters based on the improved control law, $\mathbf{c}_{\ell+1}$, and updates the critic parameters based on the improved value-function derivative, $\lambda_{\ell+1}$, is given below.

Dual-Heuristic-Programming Actor Update Suppose a control-law approximator $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_\ell)$ and a corresponding value-function-derivative approximator $\tilde{\lambda}(\cdot, \omega_\ell)$ are given. Then, an improved control-law approximator $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ can be obtained by computing a desired control vector $\mathbf{u}_k^{\mathrm{D}}$, such that,

$$\left[\frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} + \left(\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\right)^T \tilde{\lambda}(\mathbf{x}_{k+1}, \omega_\ell)\right]\Bigg|_{\mathbf{u}_k = \mathbf{u}_k^{\mathrm{D}}} = 0 \tag{3.25}$$

and the corresponding Hessian matrix is positive definite.

The improved actor parameters, $\mathbf{a}_{\ell+1}$, are computed as follows:

$$\mathbf{a}_{\ell+1} = \arg\min_{\mathbf{a}} \{\mathbf{u}_k^{\mathrm{D}} - \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})\}, \tag{3.26}$$

where $\mathbf{u}_k^{\mathrm{D}} \equiv \mathbf{c}_{\ell+1}(\mathbf{x}_k)$, therefore $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1}) \approx \mathbf{c}_{\ell+1}(\mathbf{x}_k)$.

Dual-Heuristic-Programming Critic Update Given the improved control-law approximator $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ and the value-function-derivative approximator $\tilde{\lambda}(\cdot, \omega_\ell)$, an improved value-function-derivative approximator $\tilde{\lambda}(\cdot, \omega_{\ell+1})$ can be obtained by computing its desired value,

$$\begin{aligned}\lambda_k^{\mathrm{D}} \equiv \lambda_{\ell+1}(\mathbf{x}_k, \tilde{\mathbf{c}}) = {} & \frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} + \left(\frac{\partial \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})}{\partial \mathbf{x}_k}\right)^T \frac{\partial \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k} \\ & + \left[\frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k}\frac{\partial \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})}{\partial \mathbf{x}_k}\right]^T \tilde{\lambda}(\mathbf{x}_{k+1}, \omega_\ell),\end{aligned} \tag{3.27}$$

with $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$. The improved critic parameters are determined by solving,

$$\omega_{\ell+1} = \arg\min_{\omega} \{\lambda_k^{\mathrm{D}} - \tilde{\lambda}(\mathbf{x}_k, \omega)\}, \tag{3.28}$$

such that, $\tilde{\lambda}(\mathbf{x}_k, \omega_{\ell+1}) \approx \lambda_{\ell+1}(\mathbf{x}_k, \tilde{\mathbf{c}})$.

In the next section, a modular approach for implementing HDP and DHP is presented.

3.3.4 Implementation of the Heuristic and Dual-Heuristic Dynamic Programming Algorithms

Adaptive critic algorithms involve multiple computational levels that are conveniently interpreted by means of a modular approach. Individual modules can be modified independently of one another such that algorithmic changes and debugging can be performed quickly and reliably. The key modules and their characteristics summarized in Table 3.1, are of two types: *functional modules* and *algorithmic modules*.

A functional module is a parametric structure whose inputs and outputs are those of the mathematical function being represented. The module's structure is fixed, and the values of the parameters determine the shape of the function. Hence, each module inherits the subscript of the corresponding parameters (e.g., the actor inherits

Table 3.1 Functional and Algorithmic Modules and Their Characteristics within Heuristic Dynamic Programming (HDP) and Dual Heuristic Programming (DHP) Architectures

Module:	Implementation of:	Inputs:	Outputs:	Architecture:
Actor$_\ell$	$\tilde{\mathbf{c}}(\cdot,\, \mathbf{a}_\ell)$	$\mathbf{x}_k$	$\approx \mathbf{c}_\ell(\cdot)$	Both
Critic$_\ell$	$\tilde{V}(\cdot,\, \mathbf{w}_\ell)$	$\mathbf{x}_k$	$\approx V_\ell(\cdot)$	HDP
	$\tilde{\boldsymbol{\lambda}}(\cdot,\, \boldsymbol{\omega}_\ell)$	$\mathbf{x}_k$	$\approx \boldsymbol{\lambda}_\ell(\cdot)$	DHP
Parameter update	Function approximation	$\mathbf{e}$, $\mathbf{p}$, and $\tilde{\mathbf{g}}\,(\cdot,\, \boldsymbol{\upsilon}_\ell)$	$\tilde{\mathbf{g}}\,(\cdot,\, \boldsymbol{\upsilon}_{\ell+1})$	Both
Actor target	Policy-improvement routine	$\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot,\, \mathbf{a}_\ell)$, and $\tilde{V}(\cdot,\, \mathbf{w}_\ell)$	$\mathbf{u}_k^D$	HDP
		$\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot,\, \mathbf{a}_\ell)$, and $\tilde{\boldsymbol{\lambda}}(\cdot,\, \boldsymbol{\omega}_\ell)$	$\mathbf{u}_k^D$	DHP
Critic target	Value-determination operation	$\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot,\, \mathbf{a}_{\ell+1})$, and $\tilde{V}(\cdot,\, \mathbf{w}_\ell)$	V_k^D	HDP
		$\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot,\, \mathbf{a}_{\ell+1})$, and $\tilde{\boldsymbol{\lambda}}(\cdot,\, \boldsymbol{\omega}_\ell)$	$\boldsymbol{\lambda}_k^D$	DHP

the subscript of the a parameters). The features of the parametric structure can be made accessible to and modified by other modules. In most programming languages this is easily achieved, for example by making these features *global variables* [33]. The actor and the critic constitute the key functional modules.

An algorithmic module may involve one or more iterations whose purpose is to update parameters, solve the optimality condition, and compute the value function or its derivatives. The details of each module depend on the adaptive critic design and parametric structures. Table 3.1 shows how the properties of each module depend on the chosen architecture. Using this notation, both the HDP and DHP algorithms can be described by the diagram in Figure 3.5. The solid lines illustrate the flow of input and output information between the modules. The dashed lines in Figure 3.5 indicate that the updated functional modules replace the preceding ones. The modules' inputs and outputs are listed in Table 3.1.

3.3.4.1 Parameter-Update Module for Function Approximation The actor and the critic functional modules represent the corresponding parametric structures, as illustrated in Table 3.1. These structures may consist of neural networks, polynomials, splines [34], or any other differentiable mappings with adjustable parameters. The

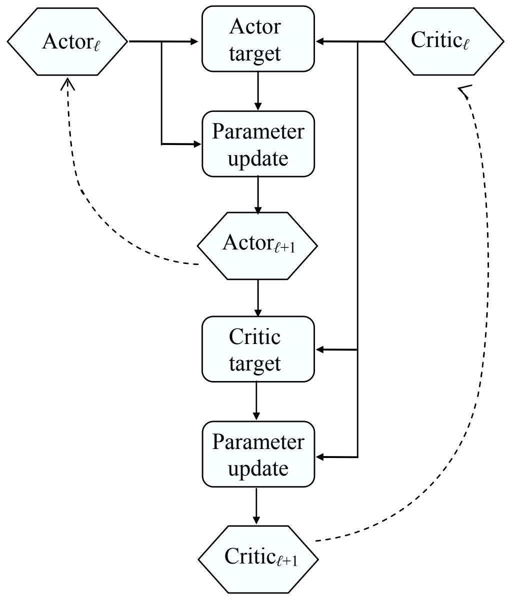
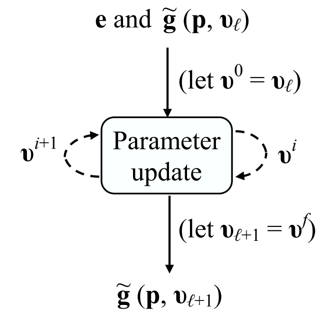


Fig. 3.5 Modular description of the $\ell' th$ iteration cycle in the HDP and DHP algorithms.

values of the parameters are adjusted by the parameter-update module during every cycle, ℓ, of the adaptive critic algorithm. For example, in the case of a neural network this module represents a training algorithm such as backpropagation [35]. In temporal-difference methods it represents the learning algorithm for updating the weight vector of the chosen basis functions.

In general, the functions in a sequence $\{\mathbf{g}_\ell(\mathbf{p}) \mid \ell = 0, 1, 2, \ldots\}$ must be successively approximated by a parametric structure, $\tilde{\mathbf{g}}(\mathbf{p}, \upsilon)$, with input $\mathbf{p}$ and adjustable parameters υ. The functions may comprise control laws or value-related functions, depending on which functional module is being updated. Eventually, the sequence converges to the optimal function $\mathbf{g}^*(\mathbf{p}) \approx \tilde{\mathbf{g}}(\mathbf{p}, \upsilon)$. Prior to this, $\mathbf{g}_{\ell+1}(\mathbf{p})$ represents an estimate of $\mathbf{g}^*(\mathbf{p})$ that has been improved with respect to the preceding function $\mathbf{g}_\ell(\mathbf{p})$ during the iteration cycle ℓ.

The parameter-update module determines the value of the parameters $\upsilon_{\ell+1}$ for which $\tilde{\mathbf{g}}(\mathbf{p}, \upsilon_{\ell+1})$ most closely approximates $\mathbf{g}_{\ell+1}(\mathbf{p})$. The parameters are initialized with either random variables or prior values. In subsequent cycles, an estimate of the parameters already is available in υ_ℓ. This estimate can be improved by minimizing a specified error, denoted in general by $\mathbf{e}$, with respect to the parameter values. The improved parameter values,$\upsilon_{\ell+1}$, replace υ_ℓ in the updated structure $\tilde{\mathbf{g}}(\mathbf{p}, \upsilon_{\ell+1})$ (as illustrated in Figure 3.6). Every error-minimization process may involve one or more



$\mathbf{e}$ and $\tilde{\mathbf{g}}(\mathbf{p}, \boldsymbol{\upsilon}_\ell)$
(let $\boldsymbol{\upsilon}^0 = \boldsymbol{\upsilon}_\ell$)
$\boldsymbol{\upsilon}^{i+1}$
Parameter update
$\boldsymbol{\upsilon}^{i}$
(let $\boldsymbol{\upsilon}_{\ell+1} = \boldsymbol{\upsilon}^f$)
$\tilde{\mathbf{g}}(\mathbf{p}, \boldsymbol{\upsilon}_{\ell+1})$

Fig. 3.6 Conceptual illustration of function approximation by the parameter-update module.

iterations that are referred to as *epochs*. These are indexed by i to distinguish them from the adaptive critic iteration cycles, indexed by ℓ.

The quantities represented by $\mathbf{e}$, $\mathbf{g}$, and $\mathbf{p}$ depend on the context in which the parameter-update module is used. Therefore, the error to be minimized, $\mathbf{e}$, and the parametric structure to be updated, $\tilde{\mathbf{g}}(\mathbf{p}, \upsilon_\ell)$, are inputs to the parameter-update module. For example, when the module is implemented to update the parameters $\mathbf{a}$ of the actor $\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})$ according to Eq. (3.18), the error to be minimized is $\{\mathbf{u}_k^{\mathrm{D}} - \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a})\}$. If the actor parameters were updated according to Eq. (3.21), the error to be minimized would be given by $\partial\tilde{V}/\partial\mathbf{u}_k$. When the module is used to update the parameters of an HDP critic, $\mathbf{e}$ consists of the term inside the curly brackets in Eq. (3.20).

3.3.4.2 Actor-Target Module for Policy Improvement The actor-target module produces an improved control policy by solving the optimality condition for a desired control vector $\mathbf{u}_k^{\mathrm{D}}$. The calculations to be performed depend on the chosen architecture. They are illustrated here for the Heuristic Dynamic Programming (HDP) and the Dual Heuristic Dynamic Programming (DHP) algorithms derived in Section 3.3.3. Depending on the form of the dynamic equation (Eq. (3.2)) and on the chosen parametric structures, the optimality condition (Eq. (3.13)) leads to a set of simultaneous equations that are either linear or nonlinear. In the latter case, an iterative approach for solving nonlinear equations (e.g., a least-squares method) can be employed. The inputs to this module are the present state of the system and the latest actor and critic functionals (Table 3.1).

At every iteration ℓ, the HDP optimality condition reduces to Eq. (3.17) for the given control-law and value-function approximations, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_\ell)$ and $\tilde{V}(\cdot, \mathbf{w}_\ell)$. These approximations are provided by the latest actor and critic functional modules, as shown in Figure 3.5. In on-line implementations, $\mathbf{x}_k$ is known from the actual state of the system observed at the present time, t_k. Then, the optimality condition can be implemented by the following actor-target module:

```
**HDP actor-target module** {
    Given $\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_\ell)$, and $\tilde{V}(\cdot, \mathbf{w}_\ell)$
    Initialize $\mathbf{u}_k^{\mathrm{D}}$ by a guess or previous estimate
    **while** $\partial\tilde{V}/\partial\mathbf{u}_k|_{\mathbf{u}_k=\mathbf{u}_k^{\mathrm{D}}} \neq 0$, update $\mathbf{u}_k^{\mathrm{D}}$ and evaluate Eq. (3.17) by computing:
        $\partial\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k|_{\mathbf{u}_k^{\mathrm{D}}}$, from the derivative of the Lagrangian
        $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)|_{\mathbf{u}_k^{\mathrm{D}}}$
        $\partial\tilde{V}(\mathbf{x}_{k+1}, \mathbf{w}_\ell)/\partial\mathbf{x}_{k+1}|_{\mathbf{u}_k^{\mathrm{D}}}$, from the derivative of $\tilde{V}(\cdot, \mathbf{w}_\ell)$ and from $\mathbf{x}_{k+1}$
        $\partial\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k|_{\mathbf{u}_k^{\mathrm{D}}}$ (from the derivative of Eq. (3.2))
    **end while**
    Check that Hessian is positive definite at $\mathbf{u}_k^{\mathrm{D}}$
    }
```

The derivatives in the *while* loop can be obtained analytically or numerically, depending on the form of the governing equation (Eq. (3.2)), on the Lagrangian $\mathcal{L}(\cdot)$ (in Eq. (3.1)), and on the critic $\tilde{V}(\cdot)$. They are evaluated at $\mathbf{x}_k$ using the latest estimate $\mathbf{u}_k = \mathbf{u}_k^{\mathrm{D}}$. Finally, $\mathbf{u}_k^{\mathrm{D}}$ must satisfy the convexity condition in Eq. (3.14).

In the DHP algorithm the actor-target module implements the same policy-improvement routine, Eq. (3.13). However, the derivative of the value function with respect to the state is approximated by the critic, so the critic module need not be differentiated.

```
**DHP actor-target module** {
    Given $\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_\ell)$, and $\tilde{\lambda}(\cdot, \omega_\ell)$
    Initialize $\mathbf{u}_k^{\mathrm{D}}$ by a guess or previous estimate
    **while** Eq.(3.25) is not satisfied, update $\mathbf{u}_k^{\mathrm{D}}$ and re-evaluate Eq. (3.25) by computing:
        $\partial\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k|_{\mathbf{u}_k^{\mathrm{D}}}$, from the derivative of the Lagrangian
        $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)|_{\mathbf{u}_k^{\mathrm{D}}}$
        $\tilde{\lambda}(\mathbf{x}_{k+1}, \omega_\ell)$, from the critic $\tilde{\lambda}(\cdot, \omega_\ell)$ and $\mathbf{x}_{k+1}$
        $\partial\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k|_{\mathbf{u}_k^{\mathrm{D}}}$ (from the derivative of Eq. (3.2))
    **end while**
    check that Hessian is positive definite at $\mathbf{u}_k^{\mathrm{D}}$
    }
```

The static minimization problem solved by the actor-target module (Eq. (3.13)) has the same dimension (m) as the control vector. Its computational difficulty is problem dependent. In the presence of multiple minima, an appropriate numerical search must be performed to ensure that $\mathbf{u}_k^{\mathrm{D}}$ is the *global* minimum of $\tilde{V}(\cdot)$.

3.3.4.3 Critic-Target Module for Value Determination The critic-target module computes an improved value function or value-function derivative for the present state, $\mathbf{x}_k$, according to the value-determination operation. The value of the improved function, denoted by the superscript $(\cdot)^{\mathrm{D}}$, can be used to update the critic parameters by means of the parameter-update module. Since the input/output characteristics of the critic depend on the chosen architecture, so do the computations performed by the critic-target module. In this section, the critic-target module is illustrated for the HDP and DHP algorithms derived in Section 3.3.3.

In the HDP algorithm, a critic-update routine is obtained by combining the original value-determination operation (Eq. (3.12)) with function approximation. The critic-target module computes the improved value V_k^{D} based on $\mathbf{x}_k$ and on the latest actor and critic approximations, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ and $\tilde{V}(\cdot, \mathbf{w}_\ell)$.

```
HDP critic-target module {
    Given $\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$, and $\tilde{V}(\cdot, \mathbf{w}_\ell)$
    Evaluate $V_k^{\mathrm{D}}$ from Eq. (3.19) by computing:
        $\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)$, using $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$
        $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$, using $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$
        $\tilde{V}(\mathbf{x}_{k+1}, \mathbf{w}_\ell)$, from the critic $\tilde{V}(\cdot, \mathbf{w}_\ell)$ and $\mathbf{x}_{k+1}$
    }
```

The above algorithm implies that the actor parameters already have been updated from $\mathbf{a}_\ell$ to $\mathbf{a}_{\ell+1}$ according to the HDP actor update (Section (3.3.3)).

Similarly, in DHP the critic-target module computes the desired value-function derivative, λ_k^{D}, according to the critic-update operation. As before, λ_k^{D} is computed from the present state of the system, $\mathbf{x}_k$, and from the latest actor and critic structures, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$ and $\tilde{\lambda}(\cdot, \omega_\ell)$. Then, the critic parameters can be updated from ω_ℓ to $\omega_{\ell+1}$ by the parameter-update module. The DHP critic-target module implements a recurrence relation (Eq. (3.27)) that is obtained by differentiating the original value-determination operation (Eq. (3.12)).

The derivatives in the critic-target module can be obtained analytically or numerically, depending on the form of the governing equation (Eq. (3.2)), on the Lagrangian $\mathcal{L}(\cdot)$ (in Eq. (3.1)), and on the actor $\tilde{\mathbf{c}}(\cdot)$. All of these quantities are evaluated at the present state $\mathbf{x}_k$ and at the control value produced by the updated actor, that is, $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$.

3.3.5 Example: Linear Quadratic Problem

Linear-quadratic (LQ) problems [1] involve a linear dynamic equation and a quadratic cost function. For this class of optimal control problems the solution can be computed

```
DHP critic-target module {
    Given $\mathbf{x}_k$, $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$, and $\tilde{\lambda}(\cdot, \omega_\ell)$
    Evaluate $\lambda_k^{\mathrm{D}}$ from Eq. (3.27) by computing:
        $\partial\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k$ and $\partial\mathcal{L}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{x}_k$, from the derivatives of the Lagrangian
        Using $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$
        $\partial\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{u}_k$ and $\partial\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)/\partial\mathbf{x}_k$ (from the derivatives of Eq.(3.2)),
        Using $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$
        $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$, using $\mathbf{u}_k = \tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})$
        $\tilde{\lambda}(\mathbf{x}_{k+1}, \omega_\ell)$, from the critic $\tilde{\lambda}(\cdot, \omega_\ell)$ and $\mathbf{x}_{k+1}$
        $\partial\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_{\ell+1})/\partial\mathbf{x}_k$, from the derivative of the actor $\tilde{\mathbf{c}}(\cdot, \mathbf{a}_{\ell+1})$
    }
```

by means of a *matrix Riccati Equation* [1]. Their solution by an adaptive critic approach is demonstrated to illustrate the concepts introduced in the previous sections.

A quadratic cost function of the form,

$$J = \lim_{t_f \to \infty} \frac{1}{2} \sum_{t_k = t_0}^{t_f - 1} [\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k], \tag{3.29}$$

is to be minimized subject to the *linear time-invariant* (LTI) dynamic equation

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k. \tag{3.30}$$

By assuming that the optimal value function is quadratic,

$$V^*(\mathbf{x}_k^*) = \frac{1}{2} \mathbf{x}_k^{*T} \mathbf{P} \mathbf{x}_k^*, \tag{3.31}$$

it can be proven [13] that the LQ optimal control law has the form,

$$\mathbf{u}_k^* = -\mathbf{C}(\mathbf{x}_k^* - \mathbf{x}_R), \tag{3.32}$$

where $\mathbf{x}_R$ represents the reference state value. The problem's objective is to determine the matrices $\mathbf{C}$ and $\mathbf{P}$ that bring the state to the value $\mathbf{x}_R$, while optimizing Eq. (3.29) subject to Eq. (3.30).

Typically, the gain matrix $\mathbf{C}$ is obtained from the Riccati matrix $\mathbf{P}$ after solving a Riccati equation [1]. Here, we determine the optimal control and value function using the DHP approach. The functions to be calculated are the control law, $\mathbf{c}$, and the value function derivative, λ. From Eqs. (3.31) and (3.32), it can be deduced that polynomials constitute suitable approximating structures for these functions. By considering the elements of $\mathbf{C}$ as the actor parameters ($\mathbf{a}$) and the elements of $\mathbf{P}$ as the critic parameters (ω), the DHP actor and critic are given by:

$$\tilde{\mathbf{c}}(\mathbf{x}_k, \mathbf{a}_\ell) = -\mathbf{C}_\ell(\mathbf{x}_k - \mathbf{x}_R), \tag{3.33}$$

$$\tilde{\lambda}(\mathbf{x}_k, \omega_\ell) = \mathbf{P}_\ell \mathbf{x}_k. \tag{3.34}$$

$\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{P}_\ell$ are symmetric, positive-definite matrices. The DHP actor and critic updates (Section 3.3.3) are used to update the elements of $\mathbf{C}_\ell$ and $\mathbf{P}_\ell$ at every cycle, ℓ, of the adaptive critic algorithm.

The LQ problem can be solved on-line by using the observed value of the state $\mathbf{x}_k$ during the ℓ^{th} cycle of the DHP algorithm (letting $\ell = k$). After the actor and critic parameters are initialized to $\mathbf{C}_0$ and $\mathbf{P}_0$ the iterative process begins. Every cycle comprises an actor update followed by a critic update. During the ℓ^{th} cycle the DHP actor update is carried out for the given $\mathbf{x}_k$, $\tilde{\mathbf{c}}_\ell$, and $\tilde{\lambda}_\ell$. The DHP actor-target module (Section 3.3.4.2) computes $\mathbf{u}_k^{\text{D}}$ from Eq. (3.25), which takes the form

$$\mathbf{R}\mathbf{u}_k^{\text{D}} + \mathbf{G}^T \mathbf{P}_\ell(\mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{u}_k^{\text{D}}) = 0. \tag{3.35}$$

The new actor parameters $\mathbf{C}_{\ell+1}$ are determined by the parameter-update module (Section 3.3.4.1). According to Eq. (3.26), this can be achieved by minimizing the error $\{\mathbf{u}_k^{\text{D}} + \mathbf{C}(\mathbf{x}_k - \mathbf{x}_R)\}$ with respect to $\mathbf{C}$.

During the same cycle, ℓ, the DHP critic update is carried out for the given $\mathbf{x}_k$, $\tilde{\mathbf{c}}_{\ell+1}$, and $\tilde{\lambda}_\ell$. The DHP critic-target module (Section 3.3.4.3) computes λ_k^{D} from Eq. (3.27), which takes the form

$$\lambda_k^{\text{D}} = \mathbf{Q}\mathbf{x}_k + \mathbf{C}_{\ell+1}^T \mathbf{R}\mathbf{C}_{\ell+1}\mathbf{x}_k + [\mathbf{F} + \mathbf{G}\mathbf{C}_{\ell+1}]^T \mathbf{P}_\ell(\mathbf{F} - \mathbf{G}\mathbf{C}_{\ell+1})\mathbf{x}_k. \tag{3.36}$$

The new critic parameters $\mathbf{P}_{\ell+1}$ are determined by the parameter-update module by minimizing the error $\{\lambda_k^{\text{D}} - \mathbf{P}\mathbf{x}_k\}$ with respect to $\mathbf{P}$.

In this simple example the parameter-update module performs a linear minimization to adjust the actor and the critic functionals (Eqs. (3.33) and (3.34)). In both cases, an error with the form $\{\mathbf{b} - \mathbf{A}\mathbf{x}\}$ is to be minimized with respect to $\mathbf{A}$. Since there are more unknowns (elements in $\mathbf{A}$) than there are equations (error elements), the problem is underdetermined. Hence, even in this simple case there exist multiple alternatives for the design of the parameter-update module (including least-squares methods). This usually is the most challenging aspect of the implementation and is likely to affect convergence greatly.

LQ problems are a class of control problems for which there exist more effective methods of solution than adaptive critic designs. This example is offered mainly for illustrative purposes. Nevertheless, if the actual system dynamics were to differ from Eq. (3.30) while retaining this linear form, the adaptive critic approach described here could be implemented to perfect the control law (Eq. (3.33)) on-line. Thus, a similar method has been applied to aircraft optimal control in [40, 41].

3.4 DISCUSSION

This chapter focuses on on-line implementations of adaptive critic designs, where the actor and the critic are updated incrementally every time a state value is observed. Clearly, this approach is not always the most effective way to solve an optimal control problem. For example, when uncertainties and plant dynamics satisfy appropriate assumptions, other designs can be implemented more easily and effectively. Adaptive critic designs present many challenges, including intricate implementation details and lack of performance guarantees. Nevertheless, they remain an important area of research with many promises and accomplishments. Their most significant feature is that they can be used to develop control systems that can *learn* to improve their performance over time subject to the actual plant dynamics.

Because the behavior of most plants cannot be fully anticipated *a priori*, adaptive critic designs have been investigated for a wide range of applications. The DHP method has been successfully implemented to prevent cars from skidding when driving over unexpected patches of ice [36]. DHP-based controllers have been developed for missile interception [37] and for aircraft auto-landing [38]. Recently, adaptive critic designs have been used to replace the automatic voltage generator and the turbine governor of a turbogenerator connected to the power grid [24]. In this research, DHP is shown to converge faster than HDP, especially in cases where the system's dynamics and operating conditions change unexpectedly. Also, DHP has been used to improve the performance of an aircraft controller on-line in the presence of unmodelled dynamics, control failures, and parameter variations [39]. These numerical simulations show that learning can prevent loss of control during critical maneuvers while they are experienced by the controller for the first time. Another possible application of adaptive critic designs is on-line state estimation, which can be formulated as an optimal control problem (Section 3.2.1), where the control, $\mathbf{u}$, is replaced by the estimate of the state.

3.5 SUMMARY

This chapter provides an introduction to the field of model-reference adaptive critic designs. These methods seek optimality in the most general case. Their use for the on-line approximate solution of infinite-horizon optimal control problems is emphasized here. Section 3.2 presents basic background material, including Howard's adaptive critic algorithm or iteration cycle, which forms the basis of most ACD. The cycle comprises a policy-improvement routine and a value-determination operation that can be used in sequence to improve the approximate solution over time. By combining this basic iteration cycle with function approximation a variety of adaptive critic designs can be obtained, as described in Section 3.3. The Heuristic and Dual-Heuristic Dynamic Programming (HDP and DHP) algorithms are derived in this section. Both algorithms can be implemented by a streamlined modular approach. This is based on two functional modules, the actor and the critic, and on three algorithmic modules, the parameter update, the actor target, and the critic target. Adaptive critic designs have been applied to a variety of problems, ranging from control of automobiles to regulation of turbogenerators. The DHP approach is demonstrated in Section 3.3.5 for a simple linear-quadratic problem. A purpose of this chapter is to motivate the further exploration of ACD for the control and estimation of complex systems.

Appendix: Proof of Convergence for the Adaptive Critic Algorithm

The following proof has been adapted from [5]. Given a generic control law $\mathbf{u}_k = \mathbf{c}(\mathbf{x}_k)$, the following recurrence relation can be found by inspection from Eqs. (3.4) and (3.7)

$$V(\mathbf{x}_k, \mathbf{c}) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}), \tag{3.37}$$

where the value function V is not necessarily optimal. For a process with a very large number of stages (or time increments), the total cost J in Eq. (3.1) can grow without limit as t_f tends to infinity. Therefore, it is convenient to define an *average total cost* corresponding to the control law $\mathbf{c}$, that has the same optimality conditions as Eq. (3.1)

$$\bar{J}_{\mathbf{c}} = \lim_{t_f \to \infty} \frac{1}{t_f} \sum_{t_k = t_0}^{t_f - 1} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k)]. \tag{3.38}$$

Also, as t_f approaches infinity in the limit, it is reasonable to let the terminal cost $\varphi[\mathbf{x}(t_f)]$ equal zero [1]. Subsequently, a *relative value function* that has value in proving convergence of the adaptive critic algorithm can be defined as

$$\upsilon_{\mathbf{c}}(\mathbf{x}_k) = \lim_{t_f \to \infty} \sum_{t_k = t_0}^{t_f - 1} [\mathcal{L}(\mathbf{x}_k, \mathbf{c}) - \bar{J}_{\mathbf{c}}], \ \ \mathbf{x}(t_0) = \mathbf{x}(t_k), \tag{3.39}$$

such that,

$$\bar{J}_{\mathbf{c}} + \upsilon_{\mathbf{c}}(\mathbf{x}_k) = \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) + V(\mathbf{x}_{k+1}, \mathbf{c}). \tag{3.40}$$

The relative value function differs from the original one by a constant that, under a given policy, can be eliminated by considering the difference in relative values, for example, $\upsilon_{\mathbf{c}}(\mathbf{x}_d) - \upsilon_{\mathbf{c}}(\mathbf{x}_e)$, representing the additional cost caused by starting at state d, rather than at state e.

Suppose that a control law $\mathbf{c}_A$ has been evaluated for the operation of the system, but the adaptive critic algorithm has produced a control law $\mathbf{c}_B \neq \mathbf{c}_A$. Then, property 1 of the algorithm can be proven by showing that the new control law has a smaller average cost, that is, $\bar{J}_B \leq \bar{J}_A$, or $\bar{J}^{\Delta} \equiv (\bar{J}_A - \bar{J}_B) \geq 0$. Since, Eqs. (3.37) and (3.40) can be written solely in terms of $\mathbf{x}_k$ and $\mathbf{u}_k$ through Eq. 3.2, the subscript k can be omitted and the equations written with respect to $\mathbf{x}$, implying a generic moment in time. If the control law $\mathbf{c}_B$ was chosen over $\mathbf{c}_A$ by the policy-improvement routine, then the following relationship applies:

$$\mathcal{L}(\mathbf{x}, \mathbf{u}_B) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_A] \leq \mathcal{L}(\mathbf{x}, \mathbf{u}_A) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_A), \mathbf{c}_A], \tag{3.41}$$

and the following positive function can be defined for all $\mathbf{x}$:

$$\gamma(\mathbf{x}) \equiv \mathcal{L}(\mathbf{x}, \mathbf{u}_A) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_A), \mathbf{c}_A] - \mathcal{L}(\mathbf{x}, \mathbf{u}_B) - V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_A] \geq 0. \quad (3.42)$$

Equation (3.40) is applied to both control laws individually, as if they were each used to control the process from t_0 to t_f:

$$\begin{aligned} \bar{J}_A + \upsilon_A(\mathbf{x}) &= \mathcal{L}(\mathbf{x}, \mathbf{u}_A) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_A), \mathbf{c}_A], \quad (3.43) \\ \bar{J}_B + \upsilon_B(\mathbf{x}) &= \mathcal{L}(\mathbf{x}, \mathbf{u}_B) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_B]. \quad (3.44) \end{aligned}$$

When the above relations are subtracted from one another they produce

$$\begin{aligned} \bar{J}_A - \bar{J}_B + \upsilon_A(\mathbf{x}) - \upsilon_B(\mathbf{x}) &= \mathcal{L}(\mathbf{x}, \mathbf{u}_A) - \mathcal{L}(\mathbf{x}, \mathbf{u}_B) \\ &\quad + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_A), \mathbf{c}_A] - V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_B]. \end{aligned} \quad (3.45)$$

which simplifies to

$$\bar{J}_A - \bar{J}_B + \upsilon_A(\mathbf{x}) - \upsilon_B(\mathbf{x}) = \gamma(\mathbf{x}) + V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_A] - V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_B], \quad (3.46)$$

by means of Eq. (3.42). By defining the quantities $\upsilon^{\Delta}(\mathbf{x}) \equiv \upsilon_A(\mathbf{x}) - \upsilon_B(\mathbf{x})$ and $V^{\Delta}[\mathbf{f}(\mathbf{x}, \mathbf{u}_B)] \equiv V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_A] - V[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_B]$, Eq. (3.46) can be written as

$$\bar{J}^{\Delta} + \upsilon^{\Delta}(\mathbf{x}) = \gamma(\mathbf{x}) + V^{\Delta}[\mathbf{f}(\mathbf{x}, \mathbf{u}_B)]. \quad (3.47)$$

This relationship takes the same form as Eq. (3.40) with the average cost $\bar{J}_{\mathbf{c}}$ given by Eq. (3.38). Thus, Eq. (3.47) can be solved similarly for $\bar{J}^{\Delta}$,

$$\bar{J}^{\Delta} = \lim_{t_f \to \infty} \frac{1}{t_f} \sum_{t_k = t_0}^{t_f - 1} \gamma(\mathbf{x}_k), \quad (3.48)$$

where $\gamma(\mathbf{x}_k) \geq 0$ for all $\mathbf{x}_k$. It follows that $\bar{J}^{\Delta} \geq 0$ and that the adaptive critic algorithm has improved the control law producing a functional that has better overall performance. This proves property 1 of the algorithm.

During every cycle, the value-determination operation updates the value functional from V_ℓ to $V_{\ell+1}$, for a given control law produced by the policy-improvement routine. Suppose during the last cycle the policy-improvement routine has chosen the control law $\mathbf{c}_B$ over $\mathbf{c}_A$, the control law from the previous cycle. Then, the following relationship must apply for all $\mathbf{x}$:

$$V_\ell(\mathbf{x}, \mathbf{c}_B) \leq V_\ell(\mathbf{x}, \mathbf{c}_A). \quad (3.49)$$

The value functional is updated by the value-determination operation according to the relationship:

$$V_{\ell+1}(\mathbf{x}, \mathbf{c}_B) = \mathcal{L}(\mathbf{x}, \mathbf{u}_B) + V_\ell[\mathbf{f}(\mathbf{x}, \mathbf{u}_B), \mathbf{c}_B]. \tag{3.50}$$

Thus the following inequality can be found to hold from Eqs. (3.37) and (3.49):

$$V_{\ell+1}(\mathbf{x}, \mathbf{c}_B) = V_\ell(\mathbf{x}, \mathbf{c}_B) \leq V_\ell(\mathbf{x}, \mathbf{c}_A). \tag{3.51}$$

The above equation shows that the value functional obtained during this last cycle, $V_{\ell+1}(\mathbf{x}, \mathbf{c}_B)$, is improved with respect to the one obtained during the previous cycle, $V_\ell(\mathbf{x}, \mathbf{c}_A)$. The equality represents the case in which the policy improvement routine has found two equally-good control laws, $\mathbf{c}_B$ and $\mathbf{c}_A$, such that $V_\ell(\mathbf{x}, \mathbf{c}_B) = V_\ell(\mathbf{x}, \mathbf{c}_A)$, but $\mathbf{c}_B \neq \mathbf{c}_A$. When this situation arises, the algorithm keeps the old control law, $\mathbf{c}_A$, and value functional, $V_\ell(\mathbf{x}, \mathbf{c}_A)$, as mentioned in Section 3.2.2. This proves property 2 of the adaptive critic algorithm. This property also implies that if the latest control law, $\mathbf{c}_B$, were used for the remainder of the process, the cost that would accrue over all future times, $V_{\ell+1}(\mathbf{x}, \mathbf{c}_B)$, would be lower than the cost that would have accrued by using the results from the previous cycle, that is, $\mathbf{c}_A$ and $V_\ell(\mathbf{x}, \mathbf{c}_A)$.

Property 3 of the adaptive critic algorithm is proven by contradiction. Suppose a superior control law exists, call it $\mathbf{c}_A$, but the algorithm has converged on a different functional, $\mathbf{c}_B$, with a larger average cost, that is, $\bar{J}_A \leq \bar{J}_B$. It follows from Eq. (3.42) that $\gamma(\mathbf{x}) \leq 0$ for all $\mathbf{x}$. But, then, Eq. (3.48) implies that $\bar{J}^\Delta \leq 0$, which contradicts the assumption $\bar{J}_A \leq \bar{J}_B$. Thus, if the algorithm converges on a control law, it means that it is optimal, for if there were a superior control law it would be discovered before termination.

Property 4 can be deduced by observing that the converging sequences can be generated successively over time, by letting $\ell = k$.

Bibliography

1. R. F. Stengel, *Optimal Control and Estimation,* Dover Publications, New York, 1994.

2. Y. Bar-Shalom and E. Tse, Dual effect, certainty equivalence, and separation in stochastic control, *IEEE Trans. Automatic Control,* vol. 19, no. 5, pp. 494–500, 1974.

3. R. Bellman, *Dynamic Programming,* Princeton University Press, Princeton, NJ 1957.

4. R. Bellman and R. Kalaba, *Dynamic Programming and Modern Control Theory,* Academic Press, New York, 1965.

5. R. Howard, *Dynamic Programming and Markov Processes,* MIT Press, Cambridge, MA, 1960.

6. A. A. Feldbaum, *Optimal Control Systems,* Academic Press, New York, 1965.

7. E. Tse, Y. Bar-Shalom, and L. Meier, III, Wide-sense adaptive dual control for nonlinear stochastic systems, *IEEE Trans. Automatic Control,* vol. 18, no. 2, pp. 98–108, 1973.

8. Y. Bar-Shalom, Stochastic dynamic programming: caution and probing, *IEEE Trans. Automatic Control,* vol. 26, no. 5, pp. 1184–1195, 1981.

9. D. Q. Mayne and H. Michalska, Receding horizon control of non-linear systems, *IEEE Trans. Automatic Control,* vol. 35, no. 5, pp. 814–824, 1990.

10. G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control,* Prentice-Hall, Englewood Cliffs, NJ, 1984.

11. S. Ferrari and R. F. Stengel, Classical/neural synthesis of nonlinear control systems, *Journal of Guidance, Control and Dynamics,* vol. 25, no. 3, pp. 442–448, 2002.

12. S. Ferrari and R. F. Stengel, Algebraic training of a neural network, *Proc. 2001 American Control Conference,* Arlington, VA, pp. 1605–1610, 2001.

13. D. E. Kirk, *Optimal Control Theory: An Introduction,* Prentice-Hall, Englewood Cliffs, NJ, 1970.

14. R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming,* Princeton University Press, Princeton, NJ, 1962.

15. R. Bellman, *Methods of Nonlinear Analysis: Volume II,* Academic Press, New York, 1973.

16. P. J. Werbos, Building and understanding adaptive systems: a statistical/numerical approach for factory automation and brain research, *IEEE Trans. Syst., Man, Cybern.,* vol. 17, no. 1, pp. 7–20, 1987.

17. D. P. Bertsekas, Distributed dynamic programming, *IEEE Trans. Automatic Control,* vol. 27, pp. 610–616, 1982.

18. S. Ferrari and R. F. Stengel, An adaptive critic global controller, *Proc. American Control Conference,* Anchorage, AK, 2002.

19. P. J. Werbos, Neurocontrol and Supervised Learning: An Overview and Evaluation, *Handbook of Intelligent Control,* D. A. White and D. A. Sofge (eds.), pp. 65–86, Van Nostrand Reinhold, New York, 1992.

20. V. Prokhorov and D. C. Wunsch, II, Adaptive critic designs, *IEEE Trans. Neural Networks,* vol. 8, no. 5, pp. 997–1007, 1997.

21. P. J. Werbos, A Menu of Designs for Reinforcement Learning Over Time, *Neural Networks for Control,* W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), pp. 67–96, MIT Press, Cambridge, MA, 1990.

22. P. J. Werbos, Advanced Forecasting Methods for Global Crisis Warning and Models of Intelligence, *General Systems Yearbook,* 1997.

23. G. G. Lendaris and T. Shannon, Application considerations for the DHP methodology, *Proc. International Joint Conference on Neural Networks,* Anchorage, AK, 1998.

24. G. K. Venayagamoorthy, R. G. Harley, and D. C. Wunsch, Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neurocontrol of a turbogenerator, *IEEE Trans. Neural Networks,* vol. 13, no. 3, pp. 764–773, 2002.

25. A. Barto, R. Sutton, and C. Anderson, Neuronlike elements that can solve difficult learning control problems, *IEEE Trans. Systems, Man, and Cybernetics,* vol. 3, no. 5, pp. 834–846, 1983.

26. P. J. Werbos, Applications of advances in nonlinear sensitivity analysis, *System Modeling and Optimization: Proc. of the 10th IFIP Conference,* R. F. Drenick and F. Kozin (eds.), Springer-Verlag, New York, 1982.

27. C. Watkins, Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, Cambridge, England, 1989.

28. T. H. Wonnacott and R. Wonnacott, *Introductory Statistics for Business and Economics,* 2nd Ed., Wiley, New York, 1977.

29. R. E. Bellman and S. E. Dreyfus, Functional Approximation and Dynamic Programming, *Math. Tables and Other Aids Comp.,* Athena Scientific, Belmont, MA, 1995.

30. G. J. Gordon, Stable Function Approximation in Dynamic Programming, Technical Report CMU-CS-95-103, Carnegie Mellon University, Pittsburgh, 1995.

31. D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control,* Van Nostrand Reinhold, New York, 1992.

32. G. Strang, *Linear Algebra and Its Applications,* 3rd Ed., Harcourt, Brace, Janovich, San Diego, 1988.

33. The MathWorks, Inc., *Getting Started with MATLAB,* http://www.mathworks.com, Version 5, September 1998.

34. J. N. Tsitsiklis and B. Van Roy, An analysis of temporal-difference learning with function approximation, *IEEE Trans. Automatic Control,* vol. 42, no. 5, pp. 674–690, 1997.

35. P. J. Werbos, Backpropagation through time: what it does and how to do it, *Proc. of the IEEE,* vol. 78, no. 10, pp. 1550–1560, 1990.

36. G. G. Lendaris, L. Schultz, and T. T. Shannon, Adaptive critic design for intelligent steering and speed control of a 2-axle vehicle, *Proc. International Joint Conference on Neural Networks,* Como, Italy, 2000.

37. D. Han and S. N. Balakrishnan, Adaptive critic based neural networks for control-constrained agile missile control, *Proc. American Control Conference,* San Diego, pp. 2600–2604, 1999.

38. G. Saini and S. N. Balakrishnan, Adaptive critic based neurocontroller for autolanding of aircraft, *Proc. American Control Conference,* Albuquerque, NM, pp. 1081–1085, 1997.

39. S. Ferrari, Algebraic and adaptive learning in neural control systems, Ph.D. Thesis, Princeton University, Princeton, NJ, 2002.

40. S. N. Balakrishnan and V. Biega, Adaptive-critic-based neural networks for aircraft optimal control, *Journal of Guidance, Control, and Dynamics,* vol. 19, no. 4, pp. 893–898, 1996.

41. K. KrishnaKumar and J. Neidhoefer, Immunized adaptive critics for level-2 intelligent control, *Proc. IEEE Int. Conf. Systems, Man and Cybernetics,* vol. 1, pp. 856–861, 1997.