# A Learning Algorithm for the Control of Continuous Action Set-Point Regulator Systems

Augustine O. Esogbue and Warren E. Hearnes II
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205
aesogbue@isye.gatech.edu
wp46268@west-point.org

September 16, 1998

**Abstract**

The convergence properties for reinforcement learning approaches such as temporal differences and $Q$-learning have been established under moderate assumptions for discrete state and action spaces. In practice, however, many systems have either continuous action spaces or a large number of discrete elements. This paper presents an approximate dynamic programming approach to reinforcement learning for continuous action set-point regulator problems which learns near-optimal control policies based on scalar performance measures. The Continuous Action Space (CAS) algorithm uses derivative-free line search methods to obtain the optimal action in the continuous space. The theoretical convergence properties of the algorithm are presented. Several heuristic stopping criteria are investigated and practical application is illustrated on two example problems–the inverted pendulum balancing problem and the power system stabilization problem.

## 1 Introduction

As control problems in real-world applications become more complex, the use of traditional analytical and statistical control techniques requiring mathematical models of the plant becomes less appealing and appropriate. In many cases, the assumption of certainty in the resultant models is made not so much for validity but the need to obtain simpler and more readily solvable formulations.

If a model of the system is known, then traditional well-developed theories of optimal control or adaptive control may be used. Without a reliable analytic model, however, these methods may not be adequate and a *model-free approach* is required. Model-free techniques learn the control law through either supervised or unsupervised means. Supervised learning requires some sort of

a teacher or critic to provide the desired response at each time period. In some cases, however, an expert or quantitative input-output training data may be unavailable. Consequently, model-free control methods that can learn a control policy for a complex system through online experience have recently been proposed [1, 2, 3, 9, 10, 14, 17, 18]. These model-free reinforcement learning methods are increasingly being used as capable and potent learning algorithms in intelligent autonomous systems.

Reinforcement learning systems form effective control policies online through a systematic search of the action space in an environment which is possibly dynamic. Two major approaches to model-free reinforcement learning–specifically Sutton's method of temporal differences (TD) [17] and Watkin's $Q$-learning algorithm [18]–are online versions of classical dynamic programming approximation methods. Convergence properties for these algorithms have been derived for discrete state and action spaces [19, 4, 1]. In practice, however, many processes to be controlled have either continuous action spaces or a large number of discrete elements. Examples include stabilizing a power system under a load, controlling a multi-degree of freedom robot arm manipulator, and retrieving a tethered satellite into a spacecraft.

This paper presents an approximate dynamic programming approach to reinforcement learning for continuous action set-point regulator problems which learns near-optimal control policies based on scalar performance measures. The set-point regulation problem is reviewed in Section 2. The continuous action space (CAS) algorithm, developed in Section 3.1, uses derivative-free line search methods to obtain the optimal action in the continuous space using a dynamic discrete subset of the state space. Theoretical convergence properties and computational aspects of the algorithms are investigated in Section 3.2. The approach is then illustrated in Section 4 on two example set-point regulator systems–the inverted pendulum balancing problem and the power system stabilization problem.

## 2 The Set-Point Regulator Problem

We restrict our attention to the general class of set-point regulator problems. In these problems, a goal state, or set-point $\mathbf{s}^\star$, is defined. The objective is to drive the system from any initial state $\mathbf{s} \in \mathbf{S}$ to the set-point $\mathbf{s}^\star$ in an optimal manner with respect to some scalar return function.

### 2.1 Formulation

Consider a possibly stochastic dynamical system with scalar returns. The set of all possible states is represented by $\mathbf{S}$. In order to control the system $P$, some set of possible decisions or actions must also be available. The set of all possible actions is represented by $\mathbf{A}$. The state of the system at time step $k$ is denoted generally by the $m$-vector $\mathbf{s}(k) \in \mathbf{S}$. The action taken at time step $k$ is the $n$-vector $\mathbf{a}(k) \in \mathbf{A}$. The output state $\mathbf{s}(k+1)$ is defined by the transition

function

$$\mathbf{s}(k+1) = \tau\left(\mathbf{s}(k), \mathbf{a}(k), \omega(k)\right) \tag{1}$$

where $k = 0, 1, \ldots$ and $\omega(k)$ is some random disturbance. Let the probability that $\mathbf{s}(k+1) = j$ when $\mathbf{s}(k) = i$ and $\mathbf{a}(k) = a$ be $p_{ij}(a)$.

We restrict our investigation to policies that are time-invariant. Therefore, for any given control policy $\pi : \mathbf{S} \rightarrow \mathbf{A}$, define the control objective function for an infinite horizon problem as

$$V_\pi(\mathbf{s}) = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R(\mathbf{s}(k), \mathbf{a}(k)) | \mathbf{s}(0) = \mathbf{s}\right] \quad \forall \mathbf{s} \in \mathbf{S} \tag{2}$$

and let

$$V(\mathbf{s}) = \inf_\pi V_\pi(\mathbf{s}) \quad \forall \mathbf{s} \in \mathbf{S} \tag{3}$$

where $\gamma \in [0, 1)$ is a discount factor and $E_\pi$ is the conditional expectation using policy $\pi$. $V_\pi(i)$ represents the expected discounted total return using policy $\pi$ and starting in state $\mathbf{s}$. A policy $\pi^\star$ is $\gamma$-optimal if

$$V_\pi(\mathbf{s}) = V(\mathbf{s}) \quad \forall \mathbf{s} \in \mathbf{S}. \tag{4}$$

Formulating the optimal control problem as a dynamic program, the functional equation becomes

$$V(\mathbf{s}) = \min_{\mathbf{a} \in \mathbf{A}}\left[R(\mathbf{s}, \mathbf{a}) + \gamma \sum_{\mathbf{s}' \in \mathbf{S}} p_{\mathbf{s}\mathbf{s}'}(\mathbf{a}) V(\mathbf{s}')\right] \quad \forall \mathbf{s} \in \mathbf{S} \tag{5}$$

which represents the minimum expected discounted return when starting in state $\mathbf{s}$ and always following an optimal policy.

The existence of an optimal stationary policy is guaranteed for the discounted case if the return function $R(\mathbf{s}, \mathbf{a})$ is bounded below by 0 and above by some number $B$ for all $\mathbf{s}, \mathbf{a}$ [15]. If $\pi$ is the stationary policy that chooses in state $i$ the action minimizing

$$R(i, a) + \gamma \sum_{j=0}^{\infty} p_{ij}(a) V(j) \tag{6}$$

then $\pi$ is $\gamma$-optimal [7].

## 2.2 Assumptions

The above formulation encompasses a variety of performance criteria including the popular performance minimization measures, such as sum of squared error (SSE), for reinforcement learning problems. For the application of the proposed algorithms, the following assumptions are made:

**Assumption 2.1** *The return, $R(i, a)$, for action $a$ taken in state $i$ is determined immediately or in some fixed time period. Further, $R(i, a)$ is bounded below by 0 and above by some finite number $B$.*

**Assumption 2.2** *The system is controllable. For every state* $\mathbf{s}$*, there exists a sequence of actions such that the system reaches* $\mathbf{s}^\star$ *in finite time with probability 1.*

These two assumptions ensure that there exists a policy $\pi$ such that $V_\pi(\mathbf{s}) < \infty$ for all $\mathbf{s} \in \mathbf{S}$.

**Assumption 2.3** *Multiple trials can be run on the system with both success and failure.*

Approximate dynamic programming methods rely on repeated experimental runs. Systems must be allowed to experience both success and failure without damaging the plant.

**Assumption 2.4** *The transition function* $\tau(\mathbf{s}, \mathbf{a}, \omega)$ *is not known. Subsequently, the transition probability,* $p_{ij}(a)$*, from state* $i$ *to state* $j$ *when action* $a$ *is taken is not known.*

The set-point regulation problem is a terminal control problem where the number of stages until the set-point is reached is not fixed. Rather, it is dependent upon the policy $\pi$. Therefore, the objective function is defined as in (5) over an infinite horizon. With an appropriate boundary condition of $V(\mathbf{s}^\star) = 0$, the problem is appropriately formulated.

# 3 Learning Optimal Continuous Actions

Dynamic programming determines the optimal solution to a variety of multi-stage decision problems by taking advantage of recurring imbedded subproblems, commonly referred to as Bellman's principle of optimality. In doing so, dynamic programming solves two related fundamental problems:

1. Determination of the optimal functional value $V(i)$ for each state $i$.
2. Determination of a policy $\pi^\star$ that achieves that value.

In many control problems, the optimal policy is the more important, and oftentimes the more readily available, part of the solution. The value of the functional equation when using an optimal policy may not necessarily be required. Unlike classical dynamic programming, in approximate dynamic programming the optimal policy and the optimal functional values are not necessarily determined in an identical computational effort. This key property is exploited in the Continuous Action Space (CAS) algorithm below.

Temporal differences and $Q$-learning, which have proved effective in online control problems, are based on finite state and action spaces. However, when either space is infinite their convergence is not necessarily guaranteed. The controller which we have proposed in [8] uses $Q$-learning as the evaluator for a discrete subset of actions within the continuous action space. We note that Watkin's $Q$-learning algorithm is an online version of successive approximations [18] which learns the particular value, termed the $Q$-value, of taking a specific action in a particular state.

The $Q$-values in the $Q$-learning algorithm may be defined so that they either learn the functional values for the optimal policy $\pi^\star$ with

$$Q(i,a) = R(i,a) + \gamma \sum_{j=1}^{S} p_{ij}(a) V(j) \tag{7}$$

and

$$V(i) = \min_j Q(i, \mathbf{a}_j) \tag{8}$$

or for a fixed policy $\pi$ with

$$Q(i,a) = R(i,a) + \gamma \sum_{j \in \mathbf{S}} p_{ij}(a) \sum_h \xi_j^\pi(h) Q(j,h) \tag{9}$$

where $\xi_j^\pi(h)$ is the probability of choosing action $h$ in state $j$ using policy $\pi$. Under the optimal stationary policy $\pi^\star$, (9) reduces to (7). In Assumption 2.4 the transition probabilities $p_{ij}(a)$ are not known. The $Q$-values for each state-action pair are estimated by $Q_n(i,a)$ with the update equation

$$Q_{n+1}(i,a) = \alpha Q_n(i,a) + (1-\alpha)\left[R(i,a) + \gamma \min_h \{Q_n(j,h)\}\right] \tag{10}$$

for (7) and

$$Q_{n+1}(i,a) = \alpha Q_n(i,a) + (1-\alpha)\left[R(i,a) + \gamma \sum_h \xi_j^\pi(h) Q_n(j,h)\right] \tag{11}$$

for (9). The $Q$-learning algorithm converges with probability 1 to within $\epsilon$ of the optimal $Q$-values and, subsequently, the optimal functional values $V$ if the system is controllable and there exists an absorbing state [18]. A number of other convergence proofs for $Q$-learning exist in the literature. Notable examples include [19, 4, 1]. In the sequel, we present an algorithm which exploits the convergence properties of $Q$-learning on discrete sets and nonlinear optimization methods to search for the optimal control in a continuous space.

### 3.1 CAS Algorithm

The Continuous Action Space (CAS) algorithm begins with a representative subset $\mathbf{a}_j, j = 1, \ldots, A$, of the action space $\mathbf{A}$ for each state $i$. This subset spans some interval of uncertainty (IoU) regarding the location of the optimal control action in the continuous action space. The $Q$-learning algorithm determines the optimal control policy given this action subset. Based on this policy, the interval of uncertainty is reduced for selected states, thereby adjusting the locations of the reference actions. As the CAS algorithm continues, the intervals of uncertainty for each state are reduced toward 0, centering on the optimal action in the continuous action space if certain assumptions are maintained.

The general CAS algorithm is as follows:

**Algorithm 3.1**

1 Set boundary condition: $V(\mathbf{s}^\star) = 0$.
2 Initialize $\mathbf{a}_j$ for all states.
3 Set $Q_0(i, \mathbf{a}_j) = M \gg 0 \quad \forall i, j$.
4 $n(i) \leftarrow 0 \; \forall i$
5 **while** $n(i) < N \; \forall i$ **do**
6 Perform an iteration of $Q$-learning.
7 **if** Policy doesn't change for state $i$
8 $n(i) \leftarrow n(i) + 1$
9 **else**
10 $n(i) \leftarrow 0$
11 **fi**
12 **if** Reduction criteria is met for state $i$
13 Reduce IoU by $\beta < 1$ around $\mathbf{a}_{j^\star}, j^\star = \text{argmin}_j Q_n(i, \mathbf{a}_j) \quad \forall i$ **od**

The choice of the reduction parameter $\beta$, learning rate $\alpha$, threshold $N$, and initial $Q$-value $M$ affects the rate of convergence. The properties of the CAS algorithm are examined in the next section.

### 3.2 Properties of the CAS Algorithm

The key to the efficiency of the CAS algorithm is that the optimal policy can, in many cases, be determined before the $Q$-learning algorithm converges to the optimal functional values. Basing the policy improvement procedure on this information is equivalent to waiting for the $Q$-learning algorithm to converge.

The $Q$-values from (7) are equivalent to a positive stochastic dynamic program and, therefore, an optimal stationary policy exists [7]. A stationary policy is one that is nonrandomized and is time-invariant. By Assumption 2.2, the set-point regulation problem is controllable. Defining the set-point $\mathbf{s}^\star$ as an absorbing state, the estimates $Q_n(i, \mathbf{a}_j)$ are guaranteed to converge to $Q(i, \mathbf{a}_j)$ as defined in (10) with probability 1 [18].

**Theorem 3.1** *Given a Markov system with an absorbing state $\mathbf{s}^\star$ and a unique optimal stationary policy $\pi^\star$, there exists an $\epsilon > 0$ sufficiently small such that if*

$$|Q_n(i, a) - Q(i, a)| < \epsilon \quad \forall i, \forall a \tag{12}$$

*for all $n \geq k$, there exists a $k' \leq k$ such that*

$$\pi_n(i) \equiv \min_a Q_n(i, a) = \pi^\star(i) \quad \forall i \tag{13}$$

*for all $n \geq k'$.*

**Proof:** The $Q$-learning algorithm converges for this system. Therefore, (12) is satisfied for each $\epsilon > 0$ and a corresponding $k \in \mathbb{N}$. A unique optimal policy implies

$$m(i) = \min_{a \neq \pi^\star(i)} |Q(i,a) - Q(i,\pi^\star(i))| > 0 \quad \forall i. \tag{14}$$

Choose $\epsilon$ such that

$$0 < \epsilon < \min_i \frac{1}{2} m(i). \tag{15}$$

It follows from convergence that for $n \geq k$

$$|Q_n(i,a) - Q(i,a)| < \epsilon \quad \forall i, a. \tag{16}$$

The choice of $m(i)$ ensures that

$$Q(i,\pi^\star(i)) + \epsilon < Q(i,\pi^\star(i)) + \frac{1}{2} m(i) < Q(i,a) - \frac{1}{2} m(i) < Q(i,a) - \epsilon \quad \forall i, a \neq \pi^\star(i). \tag{17}$$

Therefore, for any estimate $Q_n(i,a)$ for $n \geq k$,

$$Q_n(i,\pi^\star(i)) < Q_n(i,a) \quad \forall a \neq \pi^\star(i), \forall i \tag{18}$$

and

$$\pi_n(i) = \pi^\star(i) \quad \forall i, \tag{19}$$

The optimal policy $\pi^\star$ is found in no more than $k$ iterations. □

Theorem 3.1 provides a weak theoretical upper bound on the number of iterations until the $\epsilon$-optimal policy is found. In practice, the $\epsilon$-optimal policy may be found in *significantly* fewer iterations. Consider a discrete approximation to the inverted pendulum balancing problem with 625 discrete states. Using a full backup for each iteration, 479 iterations are necessary before the $Q$-values converges to within $\epsilon = 0.001$ of the optimal functional values. Yet, the optimal policy is actually determined after only 108 iterations, as shown in Figure 1–a 77.45% reduction in computational effort. Figure 2 illustrates the convergence of the maximum change in the $Q_k(i,a)$ approximation toward 0 as $Q$-learning progresses. The abrupt change in the function at iteration 108 occurs as the last policy change takes place. From that point onward, the standard $Q$-learning algorithm maintains a constant optimal policy but the approximation to the optimal value function is converging to the true values.

Each state $i$ has an interval of uncertainty (IoU) in the continuous action space $\mathbf{A}$ which contains the true optimal action $\pi^\star(i)$. The estimates of the $Q(i,\mathbf{a}_j)$ values for each state $i$ serve as a guide for reducing the interval of uncertainty. Initially, this interval is the entire action space $\mathbf{A}$. Each reduction is by a factor of $0 < \beta < 1$. Therefore, the interval can be made arbitrarily small using successive reductions. Let the reference action subset for state $i$ be defined

$$\mathbf{A}_A(i) = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_A\} \subset \mathbf{A}. \tag{20}$$

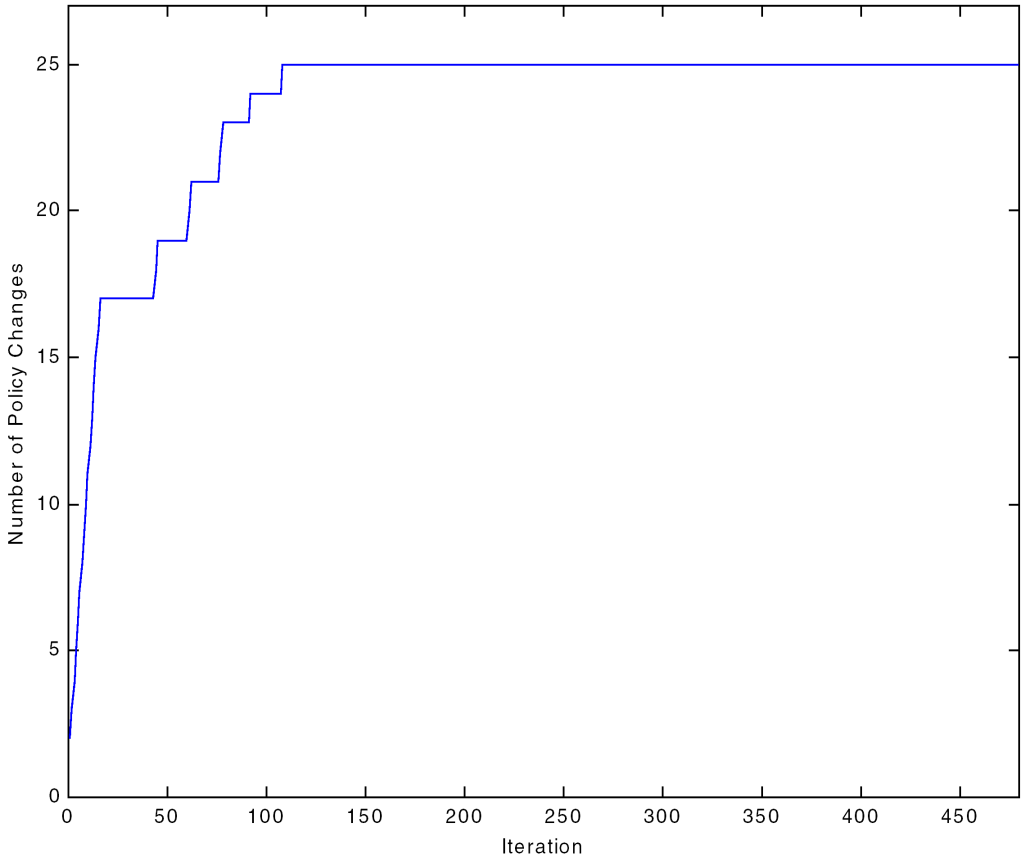


Figure 1: Optimal policy found in considerably fewer iterations than the theoretical optimal functional values.

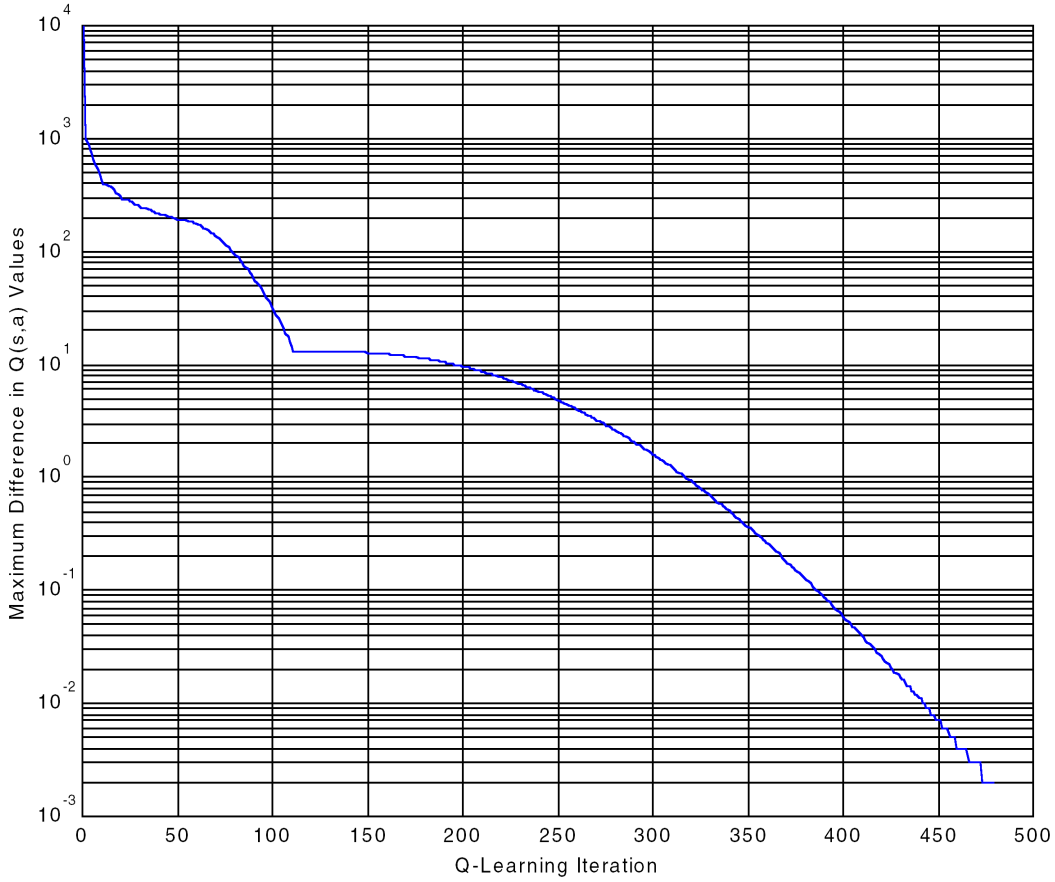


Figure 2: Maximum change in the estimated $Q$-values during each iteration, plotted on semilog paper.

Define the transformation

$$B(\beta, \mathbf{A}_A(i), k) = \{\beta[\mathbf{a}_1 - \mathbf{a}_{j^\star}] + \mathbf{a}_{j^\star}, \ldots, \beta[\mathbf{a}_A - \mathbf{a}_{j^\star}] + \mathbf{a}_{j^\star}\} \tag{21}$$

where $j^\star = \text{argmin}_j Q_k(i, \mathbf{a}_j)$ as the Successive Reduction procedure. Let $\mathbf{A}_A^0(i)$ be the initial set of reference actions. Each successive set of reference actions is defined by the transformation in (21) giving

$$\mathbf{A}_A^{l+1}(i) = B(\beta, \mathbf{A}_A^l(i), k). \tag{22}$$

This reduces the interval of uncertainty for each state $i$ as follows:

**Algorithm 3.2**

1 Determine $\epsilon$-optimal policy for $\mathbf{A}_A^l(i)$ after $k$ iterations.
2 $\mathbf{A}_A^{l+1}(i) \leftarrow B(\beta, \mathbf{A}_A^l(i), k)$

If we denote the initial interval of uncertainty, $IoU_0$, as

$$IoU_0 = \sup_a\{a \in \mathbf{A}\} - \inf_a\{a \in \mathbf{A}\}, \tag{23}$$

then it requires at least

$$\frac{\ln \epsilon - \ln IoU_0}{\ln \beta} \tag{24}$$

successive reductions of the interval of uncertainty to reach any specified accuracy $\epsilon > 0$.

**Theorem 3.2** *The Q-learning algorithm, when used to estimate the $Q(i, \mathbf{a}_j)$ values, generates two possibly distinct times $k'$ and $k$ with*

$$B(\beta, \mathbf{A}_A(i), k') = B(\beta, \mathbf{A}_A(i), k) \quad \forall i. \tag{25}$$

**Proof:** The Successive Reduction procedure is based on $j_{k'}^\star = \text{argmin}_j Q_{k'}(i, \mathbf{a}_j)$ and $j_k^\star = \text{argmin}_j Q_k(i, \mathbf{a}_j)$, respectively, $\forall i$. From Theorem 3.1, it follows that $j_{k'}^\star = j_k^\star \quad \forall i$. Therefore,

$$\beta[\mathbf{a}_1 - \mathbf{a}_{j_{k'}^\star}] + \mathbf{a}_{j_{k'}^\star} = \beta[\mathbf{a}_1 - \mathbf{a}_{j_k^\star}] + \mathbf{a}_{j_k^\star} \quad \forall i, \tag{26}$$

and (25) is proved. □

Theorem 3.2 allows for the application of the Successive Reduction procedure when the optimal policy is found rather than waiting for the optimal functional values to also converge.

**Theorem 3.3** *If $\pi^\star$ represents the $\epsilon$-optimal policy at time $k$ for the system on the reference action subset $\mathbf{A}_A$, then $\pi^\star$ is an allowable policy on the new reference action subset*

$$\mathbf{A}'_A(i) = B(\beta, \mathbf{A}_A(i), k). \tag{27}$$

**Proof:** The optimal policy $\pi^\star$ is defined as $\pi^\star(i) = \mathbf{a}_{j^\star}$, where $j^\star = \text{argmin}_j Q_k(i, \mathbf{a}_j) \quad \forall i$. Therefore, under the successive reduction transformation defined in (21), the element corresponding to $\mathbf{a}_{j^\star}$ becomes

$$\beta[\mathbf{a}_{j^\star} - \mathbf{a}_{j^\star}] + \mathbf{a}_{j^\star} = \mathbf{a}_{j^\star} \tag{28}$$

and $\mathbf{a}_{j^\star} \in \mathbf{A}'_A$, for all $i$. □

Since the current optimal policy over the reference subset is an allowable policy in the revised reference subset, the following Corollary holds.

**Corollary 3.1** *If the $l$-th error bound is defined as*

$$\varepsilon_V^l \equiv \max_i |V_n^l(i) - V(i)| \tag{29}$$

*where*

$$V_n^l(i) = \min_j Q_n(i, \mathbf{a}_j) \quad \forall \mathbf{a}_j \in \mathbf{A}_A^l(i), \tag{30}$$

*then $\{\varepsilon_V^0, \varepsilon_V^1, \ldots\}$ is a non-increasing sequence.*

**Proof:** Determine $\varepsilon_V^0$ from the $Q$-learning algorithm on the initial reference action subset $\mathbf{A}_A^0$. Apply the Successive Reduction procedure:

$$\mathbf{A}_A^1 = B(\beta, \mathbf{A}_A^0, n). \tag{31}$$

From Theorem 3.3, the optimal action at each state is an element of $\mathbf{A}_A^1$. Therefore,

$$V_n^{k+1}(i) = \min_j Q_n(i, \mathbf{a}_j) \leq \min_j Q_n(i, \mathbf{a}'_j) = V_{n'}^k(i) \quad \forall i \tag{32}$$

where $\mathbf{a}_j \in \mathbf{A}_A^0$ and $\mathbf{a}'_j \in \mathbf{A}_A^1$. Thus,

$$\varepsilon_V^1 \leq \varepsilon_V^0. \tag{33}$$

By induction, the result is proven. □

The error bounds are non-increasing, thus the sequence converges to some real number in $[0, \varepsilon_V^0]$. Corollary 3.1 provides that each application of the Successive Reduction procedure produces a policy that is at least as good as the previous policy.

We now examine conditions that ensure convergence to the optimal policy $\pi^\star$. Assume the state of the system is $i$, and the optimal policy over the continuous action space $\mathbf{A}$ is $\pi^\star = \{\mathbf{a}(0), \ldots, \mathbf{a}(k)\}$, where $k+1$ is the iteration when the system enters the absorbing state $\mathbf{s}^\star$. If the transition function $\tau(\mathbf{s}, \mathbf{a}, \omega)$ is not random, then the state space $\mathbf{S}$ can be divided into subsets

$$\mathbf{S} = \mathbf{s}^\star \cup \mathbf{S}^1 \cup \cdots \cup \mathbf{S}^N \tag{34}$$

where $\mathbf{S}^j$ is the set of all states that are $j$ transitions away from the set-point $\mathbf{s}^\star$ under the optimal policy $\pi^\star$. Under Assumption 2.2, $N < \infty$.

Initially, we look at the one-stage case where $k = 0$, or, equivalently, some state $i \in \mathbf{S}^1$. The Successive Reduction procedure guarantees policy improvement when the piecewise-linear approximation defined by the $Q(i, \mathbf{a}_j)$ values for state $i$ is strictly quasi-convex in $j$. See Figure 3. This ensures that if action $a^\star$ is optimal, then actions closer to that optimal action are better

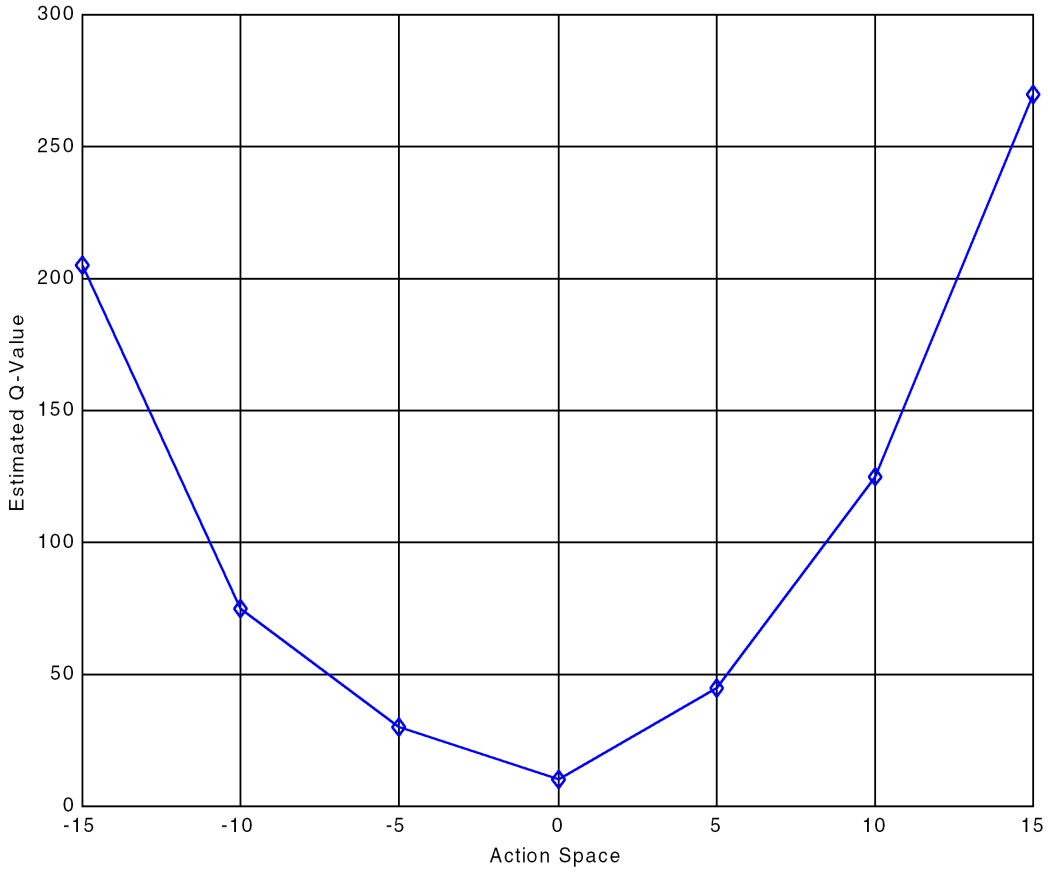


Figure 3: Piecewise-linear approximation of $Q(i, a)$ using reference subset $\mathbf{A}_7 = \{-15, -10, -5, 0, 5, 10, 15\}$.

than actions further away.

**Theorem 3.4** *If the $Q(i, \mathbf{a}_j)$ function is strictly quasi-convex in $j$ for state $i \in \mathbf{S}^1$ using reference subset $\mathbf{A}_A$, then for*

$$(\sup_a\{a \in \mathbf{A}\} - \inf_a\{a \in \mathbf{A}\})(A-1)^{-1} < \beta < 1$$

*the reference subset*

$$\mathbf{A}'_A(i) = B(\beta, \mathbf{A}_A(i), k) \quad \forall i \tag{35}$$

*is such that*

$$\min_j Q(i, \mathbf{a}'_j) \leq \min_j Q(i, \mathbf{a}_j). \tag{36}$$

*Further, for any* $\epsilon > 0$ *there exists an* $N \in \mathbb{N}$ *such that* $N$ *applications of the Successive Reduction procedure satisfies*

$$\max_j \|\mathbf{a}_j - \pi^\star(i)\| < \epsilon \tag{37}$$

**Proof:** Upon convergence of the $Q$-values, the optimal action in $\mathbf{A}_A$ for state $i$ is

$$j_k^\star = \text{argmin}_j Q_k(i, \mathbf{a}_j). \tag{38}$$

Therefore,

$$\beta[\mathbf{a}_{j_k^\star} - \mathbf{a}_{j_k^\star}] + \mathbf{a}_{j_k^\star} = \mathbf{a}_{j_k^\star}, \tag{39}$$

and $\mathbf{a}_{j_k^\star} \in \mathbf{A}'_A$. Thus,

$$\min_j Q(i, \mathbf{a}'_j) \leq Q(i, \mathbf{a}'_{j_k^\star}) = \min_j Q(i, \mathbf{a}_j). \tag{40}$$

Therefore, (36) is satisfied. As a result of the strict quasi-convexity, the optimal action

$$\mathbf{a}_{j_k^\star - 1} < a^\star < \mathbf{a}_{j_k^\star + 1}. \tag{41}$$

The choice of $\beta > (\sup_a\{a \in \mathbf{A}\} - \inf_a\{a \in \mathbf{A}\})(A-1)^{-1}$ ensures that both

$$\mathbf{a}'_1 \leq \mathbf{a}_{j_k^\star - 1}$$

and

$$\mathbf{a}_{j_k^\star - 1} \leq \mathbf{a}'_A.$$

Therefore,

$$\mathbf{a}'_1 < a^\star < \mathbf{a}'_A. \tag{42}$$

□

The strict quasi-convexity assumption does not always hold, but even then an adroit choice of $\beta$ and $A$ can achieve the desired result in practice. This is evident in the example applications that follow.

Theorem 3.4 ensures convergence to the optimal action for the one-stage problem. By induction, the CAS algorithm converges to the optimal policy $\pi^\star$ under the strictly quasi-convex assumption if the successive reductions are applied to a particular sequence of states.

**Theorem 3.5** *The Successive Reduction procedure determines the* $\epsilon$*-optimal policy* $\pi^\star$ *for all states, for a system with the* $Q(i, \mathbf{a}_j)$ *function strictly quasi-convex in* $j$ *for all* $i$*.*

**Proof:** Consider all states $i \in \mathbf{S}^1$. By Theorem 3.4, there exists a $K$ such that after $K$ successive reductions on each state $i$,

$$\max_j \|\mathbf{a}_j - \pi^\star(i)\| < \epsilon. \tag{43}$$

Assume this is true for subsequent applications to sets $\mathbf{S}^2$, ..., $\mathbf{S}^{N-1}$. Therefore, by induction and Bellman's principle of optimality, for all states $i \in \mathbf{S}^N$, the problem is equivalent to the one-stage case with a terminal cost of $\min_{j'} Q(i', \mathbf{a}'_j)$ where $i' \in \mathbf{S}^{N-1}$ is the successor state to the action $j$ taken in state $i$. Thus, the optimal action is found for all states $i$. □

Theoretical convergence is guaranteed by the appropriate choice of which states to apply the Successive Reduction procedure to first. The constructed sequence, $\mathbf{S}^1$, ..., $\mathbf{S}^N$, accomplishes this but, under Assumption 2.4 this classification of states cannot be made explicitly. As such, heuristics are employed that attempt to approximate these subsets of $\mathbf{S}$.

### 3.3 Heuristic Stopping Criterion

Recall from Figure 1 that the potential for significant computational savings exists. In practice, to ensure that the $\epsilon$-optimal policy $\pi^\star$ is found, the $\epsilon$-optimal $Q$-values must also be determined. This negates any computational savings. This section illustrates that, in practice, near-optimal policies $\pi^*$ may be found in significantly less computational effort. A stopping criterion is used to halt the $Q$-learning algorithm. The stopping criterion investigated is the *$Z$% $K$-stationary stopping criterion.*

**Definition 3.1** *The $Z$% $K$-*stationary stopping criterion *halts the $Q$-learning algorithm at iteration $n$ if the current policy*

$$\pi^*(i) \equiv \min_j Q_n(i, \mathbf{a}_j), \ \mathbf{a}_j \in \mathbf{A}_A(i) \quad \forall i \tag{44}$$

*has remained constant for the last $K$ visits to each state $i$ for at least $Z$% of the reference state subset.*

This prevents states that are infrequently visited during the learning phase from preventing the CAS algorithm to continue with its policy improvement steps.

### 3.4 Order of Successive Reduction Procedure

The theoretical convergence properties of the CAS algorithm rest upon the assumption that the Successive Reduction procedure is performed on the proper sequence of states. In Theorem 3.5, the constructed sequence, $\mathbf{S}^1$, ..., $\mathbf{S}^N$, accomplishes this but, under Assumption 2.4, this classification of states cannot be made explicitly. We propose a heuristic based on temporal differences policy evaluation that approximates these subsets of $\mathbf{S}$.

Determining the number of expected transitions from any state $i$ to the set-point $\mathbf{s}^\star$ is, itself, an approximate dynamic programming problem. The current $\epsilon$-optimal policy $\pi^*$ on reference action subset $\mathbf{A}_A(i)$ was determined via $Q$-learning using a uniform probability exploration strategy. A proper sequence of updates must be determined or the CAS algorithm converges to a suboptimal policy.

The learned $Q$-values determine the optimal action but these values do not necessarily correlate with a measure of the number of expected transitions until the set-point is reached. The online algorithm is used for approximating these $Q$-values may be extended to estimating this new value function:

$$\begin{aligned} V_{\pi^*}(i) &= \text{Expected number of transitions under policy } \pi^* \text{ from state } i \\ &\quad \text{to the set-point } \mathbf{s}^\star. \end{aligned} \tag{45}$$

The functional equation is

$$V_{\pi^*}(i) = 1 + \sum_{j=1}^{S} p_{ij}(\pi^*(i)) V_{\pi^*}(j), \tag{46}$$

with boundary condition $V_{\pi^*}(\mathbf{s}^\star) = 0$. This can easily be determined via approximate dynamic programming with the update equation:

$$V_{\pi^*}^{n+1}(i) = \alpha V_{\pi^*}^{n}(i) + (1-\alpha)(1 + V_{\pi^*}^{n}(j)). \tag{47}$$

The CAS algorithm now becomes a sequence of optimal policy determinations and successive reductions of the interval of uncertainty, IoU. The flowchart is given in Figure 4.

### 3.5 Computational Complexity

Both $Q$-learning and the CAS algorithm possess the same complexity in terms of data storage ($O(SA)$) and computations, but the potential for savings comes through the total number of iterations for each. The cyclical policy determination/policy improvement procedure of the CAS algorithm terminates in a finite number of cycles. Let the desired number of visits for each state-action pair be $D$. In full backups, $D$ iterations accomplishes this goal. In sample backups, if each state is equally probable and each action is chosen according to a uniform distribution, $DSA$ iterations are required for each state-action pair to have $D$ expected visits. If an accuracy of $\epsilon$ is desired, standard $Q$-learning with sample backups requires

$$A > \frac{(\sup_a\{a \in \mathbf{A}\} - \inf_a\{a \in \mathbf{A}\})}{2\epsilon} = \frac{IoU_0}{2\epsilon} \tag{48}$$

reference actions. However, the CAS algorithm requires fewer reference actions for the same $\epsilon$ since the interval of uncertainty is reduced during each policy determination/policy improvement cycle. Under the strict quasi-convexity assumption,

$$\beta = \frac{2}{A-1}, \tag{49}$$

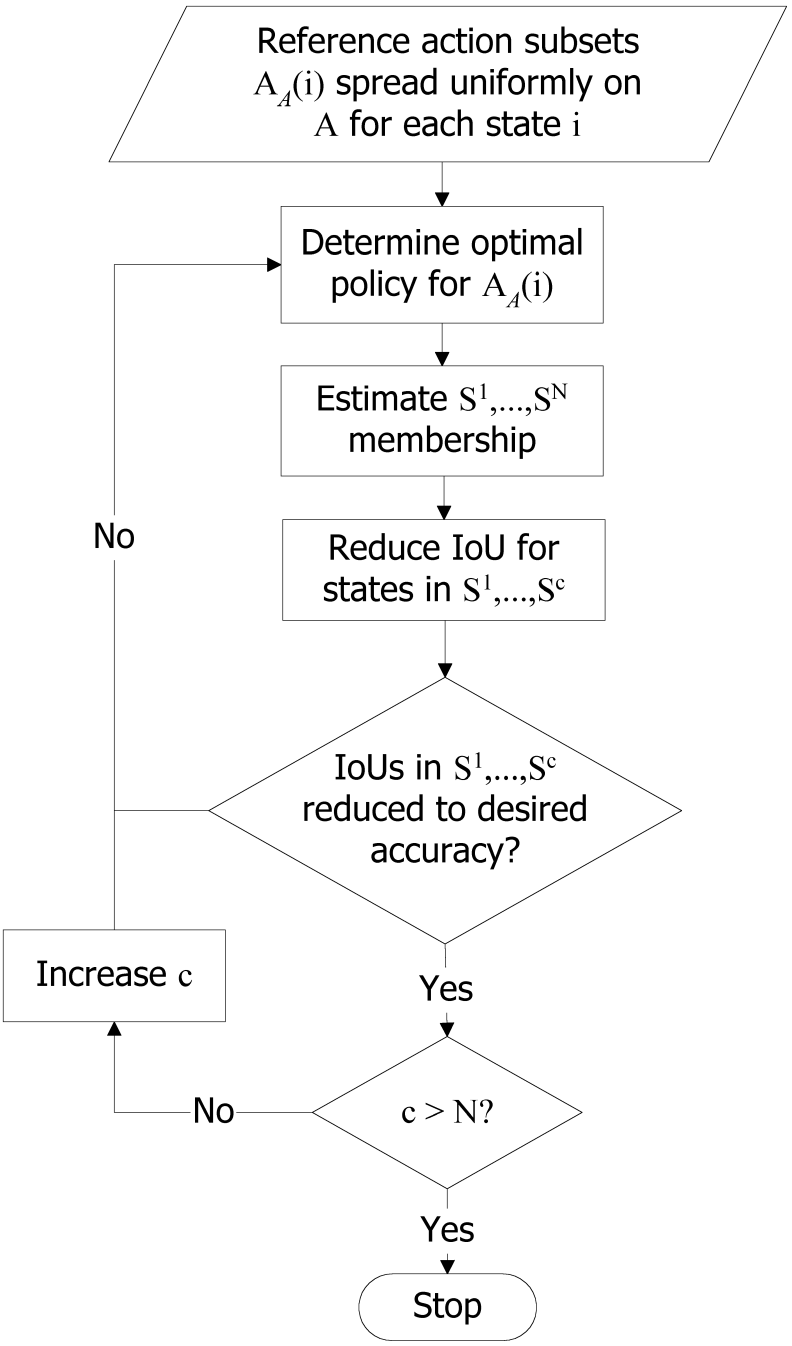


Figure 4: Flowchart for CAS algorithm.

insures that the IoU is reduced to the interval $[\mathbf{a}_{j^\star-1}, \mathbf{a}_{j^\star+1}]$ after the Successive Reduction procedure is applied. Therefore, if a limit of $M$ successive reductions for each state is desired,

$$A > \exp\left[\frac{M \ln 2 + \ln(\sup_a\{a \in \mathbf{A}\} - \inf_a\{a \in \mathbf{A}\}) - \ln \epsilon}{M}\right] + 1 \tag{50}$$

reference actions achieves $\epsilon$-accuracy. For example, if $\epsilon = 0.001$, then for standard $Q$-learning with sample backups for the inverted pendulum requires 15,000 reference actions. With the CAS algorithm, however, the same accuracy can be achieved with 347 actions and 2 successive reductions at each state. Thus, accuracy is linear in $A$ for standard $Q$-learning, but exponential in $A$ for the CAS algorithm.

In the worst case, only 1 state belongs to each $\mathbf{S}^1, \ldots, \mathbf{S}^N$ and, therefore, $(MN)DSA = MDS^2A$ iterations are required. While this is a considerable amount, in practice $N \ll S$. For the average-case in the example problems, the combination of the heuristic stopping criterion coupled with the large reduction in the number of reference actions required generates a near-optimal control policy in 20-40% fewer iterations than standard $Q$-learning.

# 4 Example Applications

The CAS algorithm and the $Z\%$ $K$-stationary stopping criterion are investigated on two set-point regulation problems. The first is the benchmark nonlinear control problem of balancing an inverted pendulum on a cart [13]. This common problem illustrates some of the advantages of the proposed algorithm. The second problem of interest is the more complex task of stabilizing a power system under a load [10, 11, 5, 6, 16]. This exhibits the application of the proposed reinforcement learning algorithm on a practical problem.

## 4.1 Inverted Pendulum Balancing

The inverted pendulum balancing problem is an example of an inherently unstable system [13] and has a broad base for comparison throughout the literature. There are four state variables, $\mathbf{s} = \langle \theta, \Delta\theta, x, \Delta x \rangle$, and one action variable, $\mathbf{a} = F$. In this experiment, the immediate return function for taking action $\mathbf{a}$ while in state $\mathbf{s}$ is defined

$$R(\mathbf{s}, \mathbf{a}) \equiv \left[\theta^2 + x^2\right] \Delta t \tag{51}$$

where $\Delta t$ represents the time interval between samplings of the state vector $\mathbf{s}$. Therefore,

$$\begin{aligned} V(i) \quad = \quad & \text{Expected discounted sum of squared error (SSE) when} \\ & \text{starting in state } i \text{ and following an optimal policy } \pi^\star \text{ thereafter.} \end{aligned} \tag{52}$$

Initially, we examine the effectiveness of our stopping criterion. Specifically, we investigate the percentage of states meeting a particular $K$-stationary criterion after a fixed number of

iterations. Three factors are used–the state space cardinality $S$, the action space cardinality $A$, and the particular $K$-stationary level. When using sample backups, we are not assured that each state-action pair is tried during the learning phase. Therefore, the $K$-stationary criterion is made dependent upon the number of actions $A$. For any level, $xA$, this implies an expected number of visits to each reference action at a particular state is $x$. Table 1 shows the various levels for the three factors in the experiment. Figure 5 plots the main effects of these factors on the percentage of states meeting the $K$-stationary stopping criterion after 1,000,000 iterations.

| Variable\Level | 0 | 1 | 2 |
|---|---|---|---|
| State Space Cardinality | 625 | | 5,625 |
| Action Space Cardinality | 11 | | 45 |
| K-Threshold | 5A | 10A | 20A |

Table 1: Levels of variables in $K$-stationary sample backup inverted pendulum balancing experiment.

Four replications for each factor level combination were run. The mean percentage of states meeting the $K$-stationary stopping criterion is $28.5\% \pm 0.114\%$. The standard error is estimated using a pooled estimate from the replicated runs:

$$s^2 = \frac{\nu_1 s_1^2 + \cdots + \nu_8 s_8^2}{\nu_1 + \cdots + \nu_8} = 0.004\% \tag{53}$$

with 24 degrees of freedom. From the percentage of states that have a stationary policy for $K$ iterations, the following significant inferences can be drawn:

1. All three factors seem to have a significant negative effect on the percentage of states that have a stationary policy for $K$ iterations. The variable $K$ has an effect for obvious reasons. The negative effects of $S$ and $A$ are due to the fixed stopping iteration. Naturally, with more state-action pairs, a fixed stopping point implies fewer expected visits to each pair.

2. The $S \times A \times K$ interaction was significant and, therefore, the separate effects of $S$ and $A$ are difficult to interpret.

3. Failure to balance the pendulum occurred for 3 of the 48 replicate runs. All three of these were at the $K = 5A$ level. The next experiment uses only the $K = 10A$ level for this reason.

We now examine the computational savings based on the iteration that particular $Z\%$ $K$-stationary stopping criterion are met. The state space is kept at a constant size of 625. The two factors are $A$ and $Z\%$. The levels for these two factors are given in Table 2.

The CAS algorithm cycles through a number of policy determination and policy improvement steps until a sufficient approximation of the optimal continuous policy is found. The main

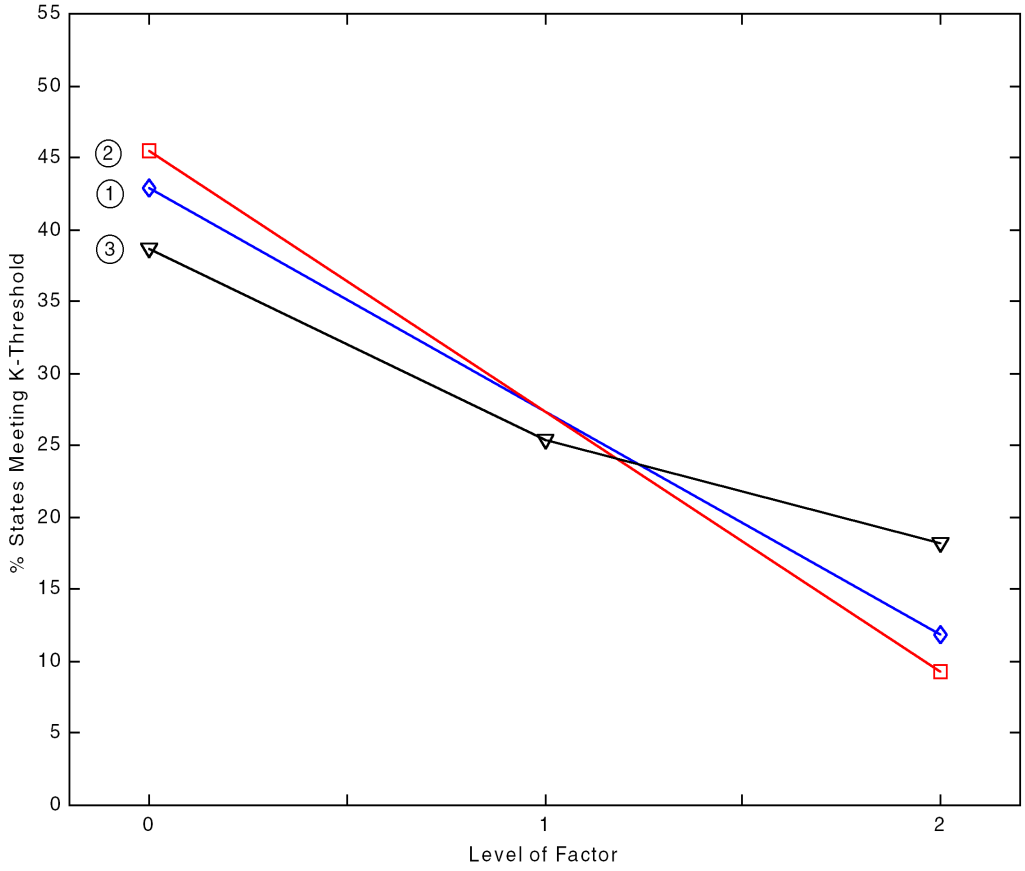


Figure 5: Main effects of factors on the percentage of states meeting $K$-stationary stopping criterion: (1) State space cardinality, (2) Action space cardinality, (3) $K$.

| Variable\Level | 0 | 1 | 2 |
|---|---|---|---|
| Action Space Cardinality | 7 | 11 | 45 |
| Z% | 25% | 50% | 75% |

Table 2: Levels of variables in $Z\%$ $K$-stationary sample backup inverted pendulum balancing experiment.

effects on computational savings from a particular $Z\%$ $K$-stationary stopping rule is shown in Figure 6. With a small number of reference actions, $A$, the computational savings can be quite

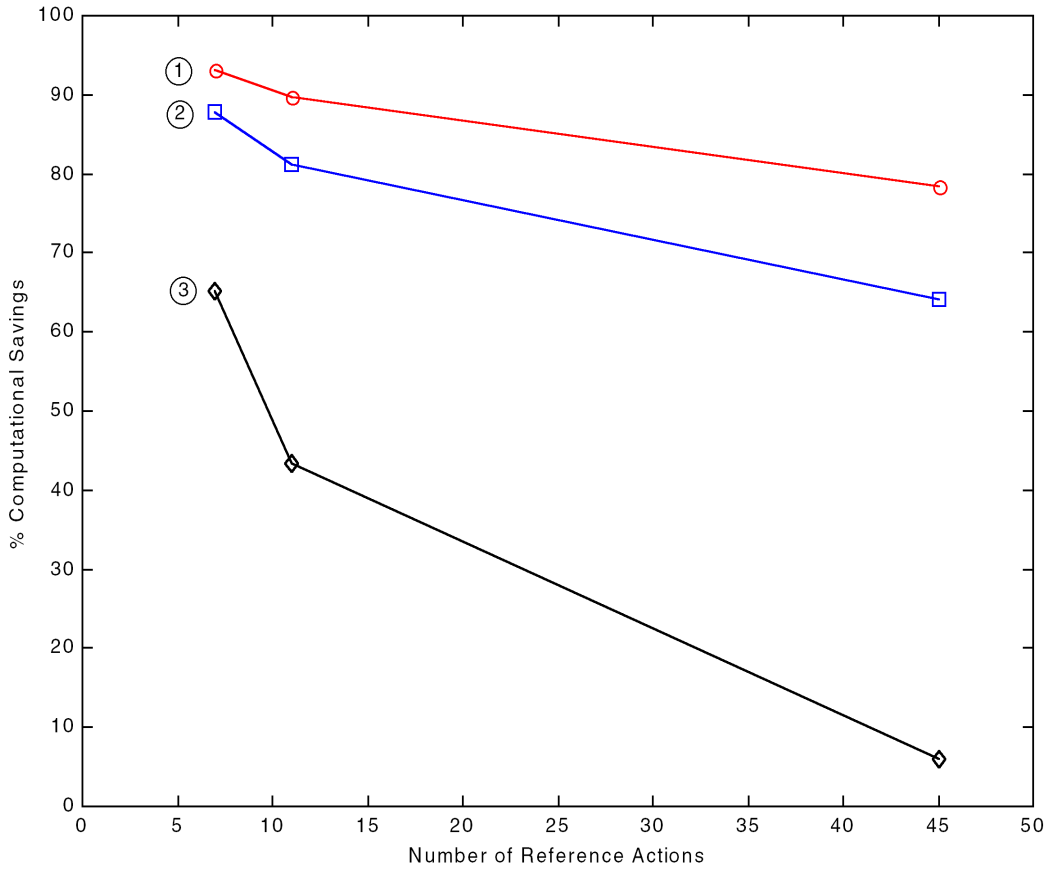


Figure 6: Main effects of factors on the number of iterations required before meeting various $Z\%$ $K$-stationary stopping criterion: (1) $Z\%$ set to 25%, (2) $Z\%$ set to 50%, (3) $Z\%$ set to 75%.

large. Specifically, for this problem the average computational savings of the CAS algorithm over standard $Q$-learning is 82.1%, 71.4%, and 49.6% respectively for $A = 7$, $A = 11$, and $A = 45$. Furthermore, there was no statistically significant difference in the optimal value function between the CAS algorithm and $Q$-learning for various initial state vectors.

For the inverted pendulum balancing problem, Figure 7 plots the trajectory of the learned optimal control for a particular initial point compared with a benchmark optimal control. The first trajectory is the benchmark optimal trajectory assuming the model is known. The second is a benchmark trajectory for an unknown model using sample backups. The third trajectory is the learned trajectory for a discrete action space approximation of $A = 7$. Finally, the fourth trajectory is the result of the CAS algorithm using $A = 7$. Corollary 3.1 ensures that the result

of the CAS algorithm can be no worse than this third trajectory. The figure highlights the advantage of searching a continuous action space, as the CAS algorithm does, over searching over a fixed discrete subset of the action space like standard $Q$-learning. In this case, the optimal control is *not* part of this discrete subset and the learned control law for $Q$-learning, while optimal over the discrete subset of reference actions, is sub-optimal for the continuous space.

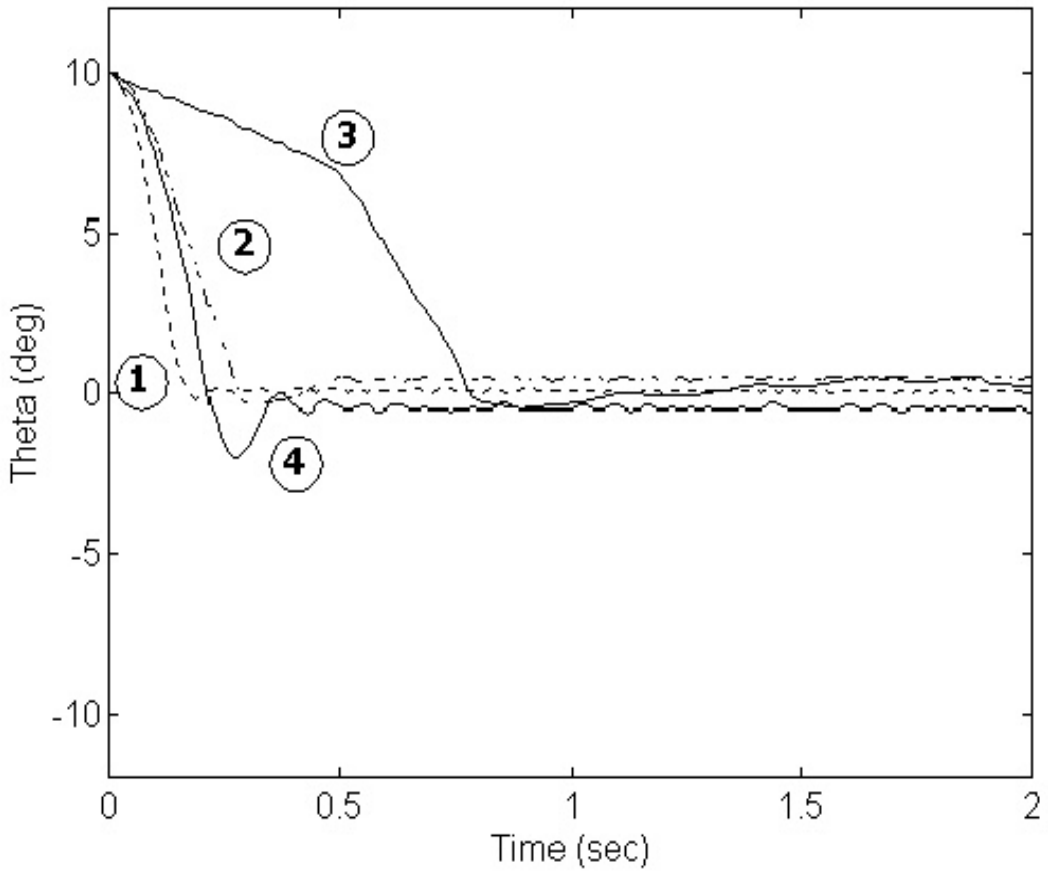


Figure 7: Learned control trajectories for various algorithms: (1) Benchmark with full backup, (2) Benchmark with sample backup, (3) $Q$-learning, (4) CAS algorithm.

### 4.2 Power System Stabilization

The power system stabilization problem represents an underdetermined system that can be approached as either a multiple-input/single-output (MISO) or multiple-input/multiple-output (MIMO) control problem [10]. For illustrative purposes, the MISO approach is taken here. The system considered is composed of a synchronous machine with an exciter and a stabilizer

connected to an infinite bus. The dynamics of the synchronous machine can be expressed using the linearized incremental model with two measurable inputs, $\mathbf{s} = [\omega, \Delta\omega]$, and one output, $\mathbf{a} = [u]$ [12]. In contrast to the inverted pendulum system where the dynamics are assumed unknown but the complete state of the system can be measured, the power system stabilization problem has only two state variables, $[\omega, \Delta\omega]$, that can readily be measured. Therefore, the unknown values for the remaining state variables create a stochastic process with unknown transition probabilities which the algorithms will implicitly learn.

Due to the additional uncertainties in the dynamics of the system, more learning iterations are needed in both $Q$-learning and the CAS algorithm. Not all experiments ended in success, though the majority did. Four replications of each experiment were run. For each, the desired accuracy, $\epsilon$, is set to 0.001. For standard $Q$-learning, this implied that

$$A > \frac{+0.12 - (-0.12)}{2 \cdot 0.001} = 120. \tag{54}$$

For the CAS algorithm, $A = 11$ was chosen. Under the strict quasi-convexity assumption, $\beta = 0.2$ and three iterations of the Successive Reduction procedure will obtain the $\epsilon$-accuracy. The number of discrete states was set to 625 for both algorithms.

Our experiments on this practical problem of interest highlight some of the advantages and disadvantages of using the CAS algorithm on an underdetermined system. Of the twenty random experiments run for both $Q$-learning and the CAS algorithms, both learned to keep the system from failing in 19 of 20 trials. By varying the number of learning iterations to insure that convergence to the optimal value function is obtained in $Q$-learning, it was determined that in some instances this system cannot be controlled simply by measuring $\omega$ and $\Delta\omega$ and varying $u$. In the 19 trials where success was obtained, there were varying degrees of "goodness" of the learned control law. For the $Q$-learning algorithm, the range of settling times varied from 4.21 seconds to 11.38 seconds. Similarly, the range of settling times for the CAS algorithm varied from 4.53 seconds to 9.76 seconds. There was no statistically significant difference in the paired settling times for the same random trial for each algorithm. As an example of a learned control trajectory, see Figure 8. The CAS controller successfully learned to stabilize the plant in approximately 4.5 seconds after a load is applied.

The advantage of using the CAS algorithm over standard $Q$-learning is in the computational savings. For the power system stabilization problem, the average computational savings is 23.4% for the 20 trials. This is due to the ability of the CAS algorithm to concentrate its learning effort on a particular region of the action space by refining the interval of uncertainty (IoU) through the Successive Reduction procedure.

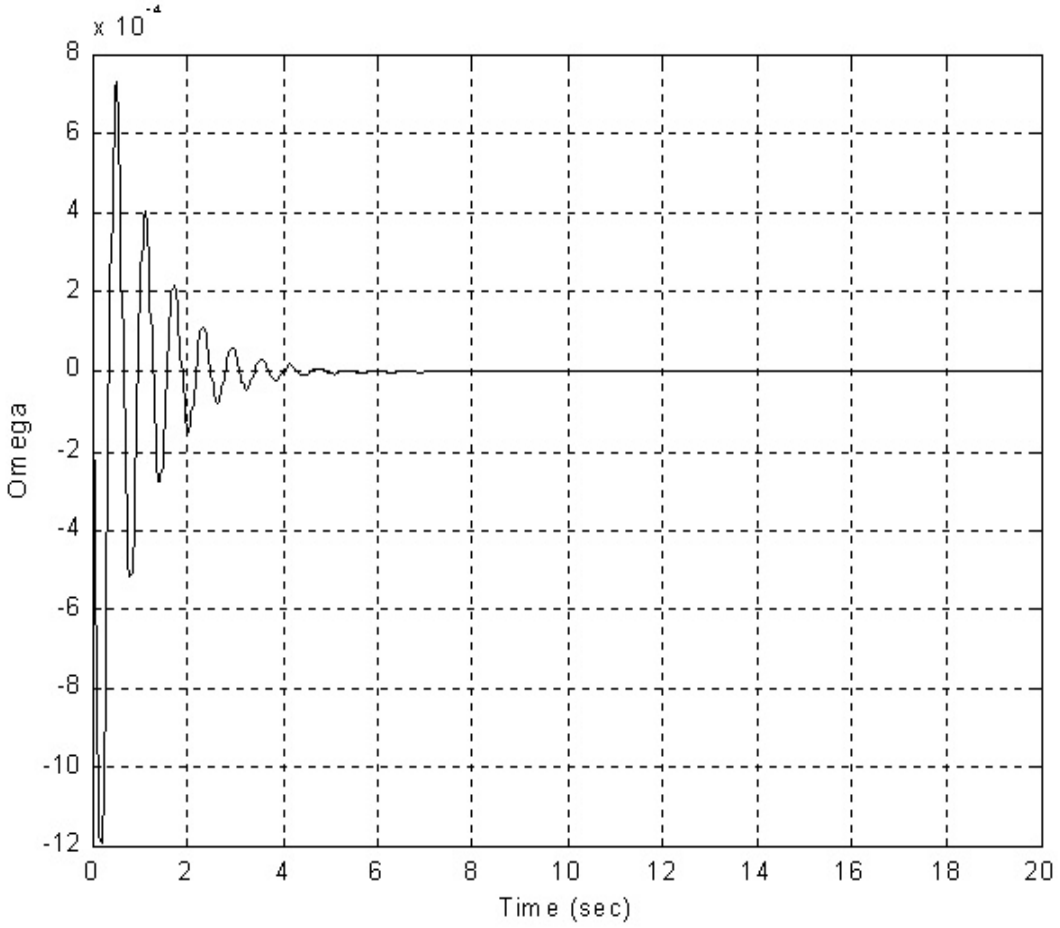


Figure 8: Learned control trajectory for the power system stabilization problem using the CAS algorithm.

## 5 Summary

This paper proposes a reinforcement learning algorithm for set-point regulation problems that have continuous action spaces. It is based on Watkin's $Q$-learning algorithm and derivative-free line search methods in optimization. The computational savings over traditional $Q$-learning have been illustrated. The CAS algorithm has also been shown to efficiently learn an $\epsilon$-optimal control law for two example problems. Its theoretical convergence properties have been established under moderate assumptions, though deviation from these assumptions in practice is not necessarily detrimental to learning a good control law.

We are currently investigating methods to generalize these learned control laws to continuous state spaces through procedures based on fuzzy set theory. Application to more diverse problems of interest is also underway, especially those problems that require the use of a hierarchical reinforcement learning approach.

## 6 Acknowledgments

This research is sponsored in part by the National Science Foundation under Grant ECS-9216004, the Electric Power Research Institute under Grant RP 8030-16, and the National Aeronautics and Space Administration under Grant NAG 9-726.

## References

[1] A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1-2):81–138, 1995.

[2] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[3] H.R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.

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

[5] S.-J. Cheng, Y.S. Chow, O.P Malik, and G.S. Hope. An adaptive synchronous machine stabilizer. *IEEE Transactions on Power Systems*, PWRS-1(3):101–107, August 1986.

[6] S.-J. Cheng, O.P Malik, and G.S. Hope. Self-tuning stabilizer for a multimachine power system. In *IEE Proceedings*, volume 133-C, pages 176–185, May 1986.

[7] S.E. Dreyfus and A.M. Law. *The Art and Theory of Dynamic Programming*. Academic Press, New York, 1977.

[8] A.O. Esogbue and W.E. Hearnes. Approximate policy improvement for continuous action set-point regulation problems. In *Proceedings of the 1998 World Congress on Compuational Intelligence*, pages 1692–1697, Anchorage, AK, May 4-9 1998.

[9] A.O. Esogbue and J.A. Murrell. A fuzzy adaptive controller using reinforcement learning neural networks. In *Proceedings of Second IEEE International Conference on Fuzzy Systems*, pages 178–183, March 28–April 1 1993.

[10] A.O. Esogbue, Q. Song, and W.E. Hearnes. Application of a self-learning fuzzy-neuro controller to the power system stabilization problem. In *Proceedings of the 1995 World Congress on Neural Networks*, volume II, pages 699–702, Washington, DC, July 17-21 1995.

[11] A.O. Esogbue, Q. Song, and W.E. Hearnes. Defuzzification filters and applications to power system stabilization problems. In *Proceedings of the 1996 IEEE International Conference on Systems, Man and Cybernetics Information, Intelligence and Systems*, Beijing, China, October 14 - 17 1996.

[12] Y.-Y. Hsu and C.-Y. Hsu. Design of a proportional-integral power system stabilizer. *IEEE Transactions on Power Systems*, PWRS-1(2):46–53, May 1986.

[13] Thomas III MIller, Richard S. Sutton, and Paul J. Werbos, editors. *Neural Networks for Control*. MIT Press, Cambridge, MA, 1990.

[14] James A. Murrell. *A Statistical Fuzzy Associative Learning Approach To Intelligent Control*. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, December 1993.

[15] S. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, San Diego, CA, 1983.

[16] J. Shi, L.H. Herron, and A. Kalam. A fuzzy logic controller applied to power system stabilizer for a synchronous machine power system. In *Proceedings of IEEE Region 10 Conference, Tencon 92*, Melbourne, Australia, November 11-13 1992.

[17] R.S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[18] C.J.C.H. Watkins. *Learning from delayed rewards*. Ph.d. thesis, Cambridge University, Cambridge, England, 1989.

[19] C.J.C.H. Watkins and P. Dayan. $q$-learning. *Machine Learning*, 8:279–292, 1992.