LETTER Communicated by Andrew Barto

Reinforcement Learning State Estimator

Jun Morimoto
xmorimo@atr.jp
JST, ICORP, Computational Brain Project, 4-1-8 Honcho, Kawaguchi, Saitama, 332-0012, Japan

Kenji Doya
doya@irp.oist.jp
Initial Research Project, Okinawa Institute of Science and Technology, Uruma, Okinawa 904-2234, Japan

In this study, we propose a novel use of reinforcement learning for estimating hidden variables and parameters of nonlinear dynamical systems. A critical issue in hidden-state estimation is that we cannot directly observe estimation errors. However, by defining errors of observable variables as a delayed penalty, we can apply a reinforcement learning framework to state estimation problems. Specifically, we derive a method to construct a nonlinear state estimator by finding an appropriate feedback input gain using the policy gradient method. We tested the proposed method on single pendulum dynamics and show that the joint angle variable could be successfully estimated by observing only the angular velocity, and vice versa. In addition, we show that we could acquire a state estimator for the pendulum swing-up task in which a swing-up controller is also acquired by reinforcement learning simultaneously. Furthermore, we demonstrate that it is possible to estimate the dynamics of the pendulum itself while the hidden variables are estimated in the pendulum swing-up task. Application of the proposed method to a two-linked biped model is also presented.

1 Introduction

Reinforcement learning is widely used to find policies to accomplish tasks through trial and error (Sutton & Barto, 1998). However, standard reinforcement learning algorithms like Q-learning can perform badly when the state variables of the environment are not fully observable. In this letter, we propose a dual reinforcement learning paradigm in which an agent learns to predict the hidden state of the environment and also to take actions to the environment.

There are three major approaches to deal with partially observable Markov decision processes (POMDPs): memoryless stochastic policy

learning, memory-based state augmentation, and model-based state prediction. Memoryless policy gradient methods to achieve better average performance over hidden variables are used in Jaakkola, Singh, & Jordan (1995), Baird & Moore (1999), Meuleau, Kim, & Kaelbling (2001), and Kimura & Kobayashi (1998). However, even when stochastic policies are used, the learning performance of a policy for POMDPs can be significantly worse than that of a policy acquired for Markov decision processes (MDPs). External memory, which stores the history of state transitions and defines an augmented state space to recover the MDP assumption, is used in Meuleau, Peshkin, Kim, and Kaelbling (1999) and McCallum (1995). However, even with external memory, we cannot always estimate hidden variables from observed variables. For example, we need an infinite impulse response filter such as an integrator to estimate position from velocity. Furthermore, a simple integrator does not work well if sensor data include observation noise or if we do not know the initial position.

If we have a dynamic model of an environment, it is possible to use a state estimator for POMDPs. If the dynamics is linear and deterministic, we can design an observer (Luenberger, 1971), which is used in control theory to estimate the hidden states. In the case that the dynamics is linear and stochastic, we can use a Kalman filter, which is widely known as the optimal state estimator for linear dynamics with gaussian noise (Kalman & Bucy, 1961). Where the state transition probability distribution is nongaussian, the particle filter (Doucet, Godsill, & Andrieu, 2000) is used in many studies (Thrun, 2000; Arulampalam, Maskell, Gordon, & Clapp, 2002). Wan and van der Merwe (2000) proposed unscented filters for the nongaussian case.

In this study, we propose a reinforcement learning state estimator (RLSE), a novel learning method for estimating hidden variables of nonlinear dynamical systems to cope with POMDPs.

Evaluation of state estimation policies is difficult because we cannot observe estimation errors for hidden states. However, by defining errors of observable variables as a penalty in the reinforcement learning framework, state estimation problems can be considered as delayed reward problems, and we can apply reinforcement learning to learn policies for hidden state estimation.

We show that the state estimator can be acquired by using the policy gradient method (Kimura & Kobayashi, 1998) and demonstrate that we can acquire the state estimator for a pendulum swing-up task in which a swing-up controller is also acquired simultaneously by reinforcement learning. Although we can use the extended Kalman filter not only for estimating state variables but also for estimating model parameters of the pendulum dynamics (Wan & Nelson, 1997), the extend Kalman filter requires the form of the target dynamics model. In this study, we show that the swing-up controller can be acquired through estimating the hidden variables even in the case that the dynamics model of the pendulum is

not known. We also apply our proposed framework to a biped walking task.

In section 2, we introduce the nonlinear state estimation problem. It is well known that optimal control and optimal estimation for linear dynamics are dual problems, that is, we can solve an optimal estimation problem as an optimal control problem of a dual system. We present our idea that the parameters of the state estimator can be derived by solving an optimal control problem even for nonlinear dynamics (Morimoto & Doya, 2002). In section 3, we introduce the reinforcement learning state estimator (RLSE), showing our approach to acquire the state estimation policy in a reinforcement learning framework. Section 4 contains our simulation results, obtained by applying the proposed method to a single pendulum dynamics and a two-linked biped robot model.

2 Nonlinear State Estimator

We consider a state estimator for nonlinear stochastic dynamics:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{n}(k), \quad \mathbf{n}(k) \sim \mathcal{N}(\mathbf{0}, \Gamma), \tag{2.1}$$

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k)) + \mathbf{v}(k), \quad \mathbf{v}(k) \sim \mathcal{N}(\mathbf{0}, R), \tag{2.2}$$

where $\mathbf{x} \in X \subset \mathbf{R}^n$ is the state, $\mathbf{u} \in U \subset \mathbf{R}^m$ is the control input, $\mathbf{y} \in Y \subset \mathbf{R}^l$ is the observed output, $\mathbf{n}(k)$ is the system noise where $\mathcal{N}(0, \Gamma)$ represents a normal distribution with zero mean and covariance Γ, and $\mathbf{v}(k)$ denotes the observation noise where $\mathcal{N}(0, R)$ represents a normal distribution with a zero mean and covariance R.

We consider the state estimator that has the form

$$\hat{\mathbf{x}}(k+1) = \mathbf{f}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) + \mathbf{a}(k), \tag{2.3}$$

where $\hat{\mathbf{x}} \in X \subset \mathbf{R}^n$ is the estimated state and $\mathbf{a} \in A \subset \mathbf{R}^n$ is a state estimator input to correct the current estimation by using observation $\mathbf{y}$.

In this study, we define the state estimation problem as the problem of finding a stochastic state estimation policy $\pi(\hat{\mathbf{x}}, \mathbf{y}, \mathbf{a}) = P(\mathbf{a}|\hat{\mathbf{x}}, \mathbf{y})$ to reduce the estimation error. Here we can consider this estimation problem as POMDP because we cannot directly access the state $\mathbf{x}$, and it is necessary to decide actions for this estimation task according to current observation $\mathbf{y}(k)$, where $\mathbf{y}(k)$ is given by the conditional probability distribution,

$$P(\mathbf{y}(k)|\mathbf{x}(k)) = \frac{1}{\sqrt{(2\pi)^l |R|}} \exp\{-(\mathbf{y}(k) - \mathbf{h}(\mathbf{x}(k)))^T R^{-1} (\mathbf{y}(k) - \mathbf{h}(\mathbf{x}(k)))\}. \tag{2.4}$$

The difficulty of finding this policy is that we cannot directly observe the error for the hidden states. For linear dynamics,

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + \mathbf{n}(k), \tag{2.5}$$

$$\mathbf{y}(k) = C\mathbf{x}(k) + \mathbf{v}(k), \tag{2.6}$$

we can design an observer,

$$\hat{\mathbf{x}}(k+1) = A\hat{\mathbf{x}}(k) + B\mathbf{u}(k) + L(\mathbf{y}(k) - C\hat{\mathbf{x}}(k)), \tag{2.7}$$

where observer gain L is selected to stabilize the error dynamics,

$$\mathbf{e}(k+1) = (A - LC)\mathbf{e}(k), \tag{2.8}$$

which can be derived by subtracting equation 2.7 from equation 2.5 without considering noise input $\mathbf{n}(k)$ in equation 2.5, where $\mathbf{e}(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$. In this case, the estimation policy is represented as a deterministic policy: $\mathbf{a}(k) = L(\mathbf{y}(k) - C\hat{\mathbf{x}}(k))$, that is, $\pi(\hat{\mathbf{x}}(k), \mathbf{y}(k), \mathbf{a}(k)) = P(\mathbf{a}(k)|\hat{\mathbf{x}}(k), \mathbf{y}(k)) = \delta(\mathbf{a}(k) - L(\mathbf{y}(k) - C\hat{\mathbf{x}}(k)))$, where $\delta()$ denotes the Dirac delta function.

Our solution to this estimation problem is to define the accumulated errors of observable variables as the objective function and solve the optimal control problem using a reinforcement learning framework. In the next section, we introduce our proposed method to learn the estimation policy.

3 Reinforcement Learning State Estimator

Here we introduce the reinforcement learning state estimator (RLSE), a new method to acquire a state estimation policy based on a reinforcement learning framework. We consider a discrete-time formulation of reinforcement learning with the following system dynamics:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{n}(k), \tag{3.1}$$

$$\mathbf{y}(k) = \mathbf{h}(\mathbf{x}(k)) + \mathbf{v}(k), \tag{3.2}$$

$$\hat{\mathbf{x}}(k+1) = \hat{\mathbf{f}}(\hat{\mathbf{x}}(k), \mathbf{u}(k); \mathbf{p}(k)) + \mathbf{a}(k), \tag{3.3}$$

where $\mathbf{a} \in A \subset \mathbf{R}^n$ is the output of a state estimation policy (see section 3.2), which is input to the state estimator; $\mathbf{p}$ denotes the parameter vector of the approximated plant model $\hat{\mathbf{f}}$ and is the output of a model estimation policy (see section 3.3). The reward is defined as

$$r(\hat{\mathbf{x}}, \mathbf{y}, \mathbf{a}) = e(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})) - c(\mathbf{a}), \tag{3.4}$$

where $e(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}))$ denotes reward function for correct estimation, and $c(\mathbf{a})$ denotes cost function for changing the estimator dynamics (Morimoto & Doya 2002). The function e has the maximum value when the size of the error $|\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})|$ equals zero, and if the direction of the error vector is the same, the value of the function e monotonically decreases when the size of the error grows. The cost function $c(\mathbf{a})$ corresponds to the amplitude of process noise. The state estimator tends to be strongly affected by the observed output $\mathbf{y}(k)$ with a smaller cost for changing the estimator dynamics. For example, we can use a negative log-likelihood function as the reward and the cost functions (Crassidis & Junkins, 2004).

The basic goal is to find a policy $\pi_{\mathbf{w}}(\hat{\mathbf{x}}, \mathbf{y}, \mathbf{a}) = P(\mathbf{a}|\hat{\mathbf{x}}, \mathbf{y}; \mathbf{w})$ that maximizes the expectation of the discounted accumulated reward,

$$E\{V(k)|\pi_{\mathbf{w}}\} = E\left\{\sum_{i=k}^{\infty} \gamma^{i-k} r(\hat{\mathbf{x}}(i+1), \mathbf{y}(i+1), \mathbf{a}(i)) \middle| \pi_{\mathbf{w}}\right\}, \tag{3.5}$$

where $V(k)$ is the actual return, $\mathbf{w}$ is the parameter vector of the policy $\pi_{\mathbf{w}}$, and γ is the discount factor. Therefore, we essentially try to solve a smoothing problem rather than a filtering problem because we consider future estimation error represented in equation 3.5 without explicit backward calculation. Figure 1 shows a schematic diagram of the proposed learning system. The RLSE consists of a state estimator and the reinforcement learning (RL) module to learn an estimation policy $\pi_{\mathbf{w}}$. The state estimator takes control input $\mathbf{u}$ and the output of RL module $\mathbf{a}$, that is, the output of the estimation policy, as input variables. Then the state estimator outputs estimated state $\hat{\mathbf{x}}$ as the RLSE output. Parameters of the plant model $\mathbf{p}$ used in the state estimator and used in a controller can also be estimated by the RLSE.

The problem of learning a state estimation policy is partially observable because we have hidden variables $\mathbf{x}$ to be estimated. We use a policy gradient method proposed by Kimura and Kobayashi (1998) in our RLSE framework. Although we can apply the policy gradient method directly to POMDPs without learning the state estimators for some target problems (Jaakkola et al., 1995; Baird & Moore, 1999; Kimura & Kobayashi, 1998), we will show in the pendulum swing-up problem in the following sections that many tasks require the state estimator. The RLSE can be used even when the order of a plant is known but its form is not and when there are hidden state variables that we cannot directly observe.

3.1 Gradient Calculation and Policy Parameter Update. In the policy gradient methods, we calculate the gradient direction of the expectation of the actual return with respect to parameters of a policy $\mathbf{w}$. Kimura and

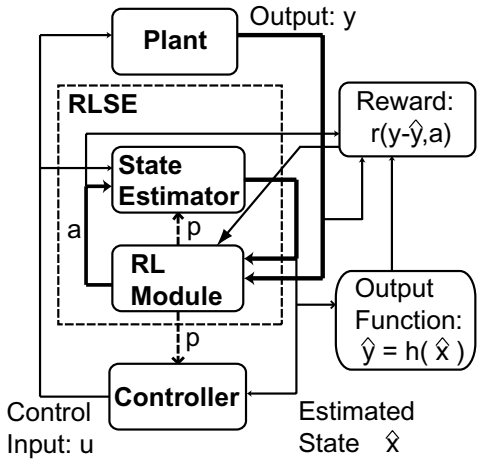

Figure 1: Reinforcement learning state estimator (RLSE). The RLSE consists of a state estimator and an RL module to learn an estimation policy. The state estimator takes control input $\mathbf{u}$ and the output of RL module $\mathbf{a}$, that is, the output of the estimation policy, as input variables. Then the state estimator outputs estimated state $\hat{\mathbf{x}}$ as the RLSE output. Parameters of the plant model $\mathbf{p}$ used in the state estimator and used in a controller can also be estimated by the RLSE.

Kobayashi (1998) suggested that we can estimate the expectation of the gradient direction as

$$\frac{\partial}{\partial \mathbf{w}} E\{V(0)|\pi_{\mathbf{w}}\} \approx E\left\{ \sum_{k=0}^{\infty} (V(k) - \hat{V}(\mathbf{x})) \frac{\partial \ln \pi_{\mathbf{w}}}{\partial \mathbf{w}} \middle| \pi_{\mathbf{w}} \right\}, \tag{3.6}$$

where $\hat{V}(\mathbf{x}(k))$ is an approximation of the value function: $V^{\pi_{\mathbf{w}}}(\mathbf{x}) = E\{V(k)|\mathbf{x}(k) = \mathbf{x},\ \pi_{\mathbf{w}}\}$.

3.1.1 Value Function Approximation. The value function is approximated using a normalized gaussian network (Doya, 2000; see appendix C),

$$\hat{V}(\hat{\mathbf{x}}) = \sum_{i=1}^{N} v_i b_i(\hat{\mathbf{x}}), \tag{3.7}$$

where v_i is a ith parameter, and N is the number of basis functions (see appendix C). An approximation error of the value function is represented

by TD error (Sutton & Barto, 1998):

$$\delta(k) = r(k) + \gamma \hat{V}(\hat{\mathbf{x}}(k+1)) - \hat{V}(\hat{\mathbf{x}}(k)). \tag{3.8}$$

We update the parameters of the value function approximator using the TD(0) method (Sutton & Barto, 1998),

$$v_i(k+1) = v_i(k) + \alpha\delta(k) b_i(\hat{\mathbf{x}}(k)), \tag{3.9}$$

where α is the learning rate.

3.1.2 Policy Parameter Update. We update the parameters of a policy $\mathbf{w}$ by using the estimated gradient direction in equation 3.6. Kimura and Kobayashi (1998) showed that we can estimate the gradient direction by using TD error,

$$E\left\{\sum_{k=0}^{\infty}(V(k) - \hat{V}(\mathbf{x}(k)))\frac{\partial \ln \pi_{\mathbf{w}}}{\partial \mathbf{w}}\middle|\pi_{\mathbf{w}}\right\} = E\left\{\sum_{k=0}^{\infty}\delta(k)\mathbf{z}(k)\middle|\pi_{\mathbf{w}}\right\}, \tag{3.10}$$

where $\mathbf{z}$ is the eligibility trace of the parameter $\mathbf{w}$. Then we can update the parameter $\mathbf{w}$ as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \beta\delta(k)\mathbf{z}(k), \tag{3.11}$$

where the eligibility trace is updated as

$$\mathbf{z}(k) = \eta\mathbf{z}(k-1) + \left.\frac{\partial \ln \pi_w}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}(k)}, \tag{3.12}$$

where η is the decay factor for the eligibility trace. Equation 3.10 can be derived if the condition $\eta = \gamma$ is satisfied.

3.2 State Estimation Policy. We construct the state estimation policy based on the normal distribution,

$$\pi_{\mathbf{w}}(\hat{\mathbf{x}}, \mathbf{y}, a_j) = \frac{1}{\sqrt{2\pi}\sigma_j(\hat{\mathbf{x}})}\exp\frac{-(a_j - \mu_j(\hat{\mathbf{x}}, \mathbf{y}))^2}{2\sigma_j^2(\hat{\mathbf{x}})}, \tag{3.13}$$

where a_j is the jth output of the policy $\pi_{\mathbf{w}}$. The mean output μ of the policy $\pi_{\mathbf{w}}$ is given by a feedback gain matrix L and the estimation error for observable variables $\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})$,

$$\mu_j(\hat{\mathbf{x}}, \mathbf{y}) = L_j(\hat{\mathbf{x}})(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})), \tag{3.14}$$

and the jth row vector L_j is modeled by the normalized gaussian network:

$$L_j(\hat{\mathbf{x}}) = \sum_{i=1}^{N} \mathbf{w}_i^{\mu_j} b_i(\hat{\mathbf{x}}). \tag{3.15}$$

Here, $\mathbf{w}_i^{\mu_j} \in \mathbf{R}^l$ denotes the ith parameter vector for jth output of the policy $\pi_{\mathbf{w}}$, and N is the number of basis functions. We represent the variance σ_j of the policy $\pi_{\mathbf{w}}$ using a sigmoid function (Kimura & Kobayashi, 1998),

$$\sigma_j(\hat{\mathbf{x}}) = \frac{\sigma_0}{1 + \exp(-\sigma_j^w(\hat{\mathbf{x}}))}, \tag{3.16}$$

where σ_0 denotes the scaling parameter, and the state-dependent parameter $\sigma_j^w(\hat{\mathbf{x}})$ is also modeled by the normalized gaussian network,

$$\sigma_j^w(\hat{\mathbf{x}}) = \sum_{i=1}^{N} w_i^{\sigma_j} b_i(\hat{\mathbf{x}}), \tag{3.17}$$

where $w_i^{\sigma_j}$ denotes the ith parameter for the variance of jth output.

Considering equation 3.13, the eligibilities of the policy parameters are given as

$$\frac{\partial \ln \pi_{\mathbf{w}}}{\partial \mathbf{w}_i^{\mu_j}} = \frac{a_j - \mu_j}{\sigma_j^2}(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}))b_i(\hat{\mathbf{x}}), \tag{3.18}$$

$$\frac{\partial \ln \pi_{\mathbf{w}}}{\partial w_i^{\sigma_j}} = \frac{\left((a_j - \mu_j)^2 - \sigma_j^2\right)(1 - \sigma_j/\sigma_0)}{\sigma_j^2} b_i(\hat{\mathbf{x}}). \tag{3.19}$$

We update the parameters by applying the update rules in equations 3.11 and 3.12.

3.3 Model Estimation Policy. Since we cannot always have exact information on the target dynamics, it is also necessary to estimate parameters of the dynamics model as well. In this study, we use a reinforcement learning framework based on the policy gradient method (Kimura & Kobayashi, 1998) to estimate parameters of the dynamics model. This model estimation policy is constructed based on the normal distribution,

$$\pi_{\mathbf{w}^p}(\hat{\mathbf{x}}, p_j) = \frac{1}{\sqrt{2\pi}\sigma_j^p(\hat{\mathbf{x}})} \exp \frac{-\left(p_j - \mu_j^p(\hat{\mathbf{x}})\right)^2}{2\left(\sigma_j^p(\hat{\mathbf{x}})\right)^2}, \tag{3.20}$$

where p_j is the jth output of the policy $\pi_{\mathbf{w}^p}$. The mean output μ^p of the policy $\pi_{\mathbf{w}^p}$ is given by

$$\mu_j^p(\hat{\mathbf{x}}) = \sum_{i=1}^{N} w_i^{\mu_j^p} b_i(\hat{\mathbf{x}}). \tag{3.21}$$

Here, $w_i^{\mu_j}$ denotes the ith parameter for jth output of the policy $\pi_{\mathbf{w}^p}$, and N denotes the number of basis functions. We represent the variance σ_j^p of the policy $\pi_{\mathbf{w}^p}$ using a sigmoid function (Kimura & Kobayashi 1998),

$$\sigma_j^p(\hat{\mathbf{x}}) = \frac{\sigma_0^p}{1 + \exp(-\sigma_j^{w^p}(\hat{\mathbf{x}}))}, \tag{3.22}$$

where σ_0^p denotes the scaling parameter, and the state-dependent parameter $\sigma_j^{w^p}(\mathbf{x})$ is also modeled by the normalized gaussian network,

$$\sigma_j^{w^p}(\hat{\mathbf{x}}) = \sum_{i=1}^{N} w_i^{\sigma_j^p} b_i(\hat{\mathbf{x}}), \tag{3.23}$$

where $w_i^{\sigma_j^p}$ denotes the ith parameter for the variance of jth output. Considering equation 3.13, the eligibilities of the policy parameters are given as

$$\frac{\partial \ln \pi_{\mathbf{w}^p}}{\partial w_i^{\mu_j^p}} = \frac{a_j - \mu_j^p}{\left(\sigma_j^p\right)^2} b_i(\hat{\mathbf{x}}), \tag{3.24}$$

$$\frac{\partial \ln \pi_{\mathbf{w}^p}}{\partial w_i^{\sigma_j^p}} = \frac{\left(\left(a_j - \mu_j^p\right)^2 - \left(\sigma_j^p\right)^2\right)\left(1 - \sigma_j^p/\sigma_0^p\right)}{\left(\sigma_j^p\right)^2} b_i(\hat{\mathbf{x}}). \tag{3.25}$$

We update the parameters by applying the update rules in equations 3.11 and 3.12.

4 Simulation

In this section, we demonstrate the performance of the RLSE. First, in section 4.1, we apply the RLSE to a single-pendulum model to display its basic properties and performance. In section 4.2, we apply it to a two-linked biped model, which has more complicated dynamics than a single-pendulum model, to show that the RLSE can be applied to a more difficult problem.

In section 4.1, we first show that the RLSE can stabilize the error dynamics (see equation 2.8) of the linearized pendulum model (see section 4.1.1).

Second, we apply it to the pendulum model to estimate the state variables of free-swing movement (see section 4.1.2). Third, we show that it is possible to acquire a controller to swing the pendulum up by using state variables estimated by the RLSE. Here, the RLSE is also acquired simultaneously (see sections 4.1.3, 4.1.4, and 4.1.5).

In section 4.1.3, we compare swing-up performance among a memoryless policy gradient method, a memory-base policy gradient method, and a value-gradient-based method (Doya, 2000) with using the RLSE. We demonstrate that only the value-gradient-based method using the RLSE can acquire a swing-up controller when only the angular velocity can be observed.

In section 4.1.4, we show that the state estimator can be acquired even in the case that we do not know a precise parameter of the pendulum. In other words, we show that we can acquire control, state estimation, and model estimation policies simultaneously.

Section 4.1.5 shows that we can acquire an estimation policy for the pendulum swing-up task even when the dynamics of the pendulum is unknown.

Finally, in section 4.2, we show that the RLSE can be used to estimate state variables of the biped model. The task in this case for the biped robot is walking down a slope without falling over. We could simultaneously acquire both control and estimation policies for the biped walking task.

4.1 Application to a Single Pendulum. The proposed method was applied to pendulum dynamics,

$$ml^2\dot{\omega} = -\mu\omega - mgl\sin\theta + T, \tag{4.1}$$

where θ is the angle from the bottom position, ω is the angular velocity, T is the input torque, $\mu = 0.01$ is the coefficient of friction, $m = 1.0$ kg is the weight of the pendulum, $l = 1.0$ m is the length of the pendulum, and $g = 9.8\ \mathrm{m/s^2}$ is the acceleration due to gravity (see Figure 2). We define the state vector as $\mathbf{x} = (\theta, \omega)^T$ and the control output as $u = T$. Here we consider one-dimensional output y. Thus, the dynamics is given by the system

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{n}(k), \tag{4.2}$$

$$y(k) = C\mathbf{x}(k) + v(k), \tag{4.3}$$

where $\mathbf{n}(k)$ and $v(k)$ are noise inputs with parameters $\Gamma = \mathrm{diag}\{0.01, 0.01\}\Delta t$ and $R = 1.0\Delta t$, and the discrete-time dynamics of the pendulum is

$$\mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) = \mathbf{x}(k) + \int_{k\Delta t}^{(k+1)\Delta t} \mathbf{F}(\mathbf{x}(s), \mathbf{u}_c(s))ds, \tag{4.4}$$

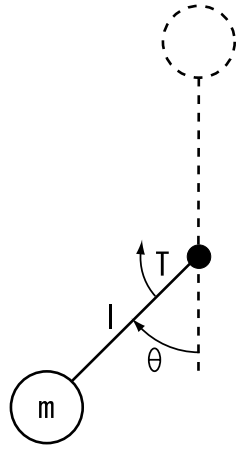

Figure 2: Single pendulum swing-up task. θ: joint angle, T: input torque, l: length of the pendulum, m: mass of the pendulum.

$$\mathbf{F}(\mathbf{x}(t), \mathbf{u}_c(t)) = \begin{pmatrix} \omega \\ -\frac{g}{l}\sin\theta - \frac{\mu}{ml^2}\omega \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u_c, \tag{4.5}$$

where Δt is a time step of the discrete-time system, and $\mathbf{u}_c$ is the control input in continuous time.

We define the state estimator dynamics as

$$\hat{\mathbf{x}}(k+1) = \mathbf{f}(\hat{\mathbf{x}}(k), \mathbf{u}(k)) + \mathbf{a}(k), \tag{4.6}$$

where $\hat{\mathbf{x}}(k) = (\hat{\theta}(k), \hat{\omega}(k))^T$ is the estimated state and $\mathbf{a}(k)$ is the output of the estimation policy. We define the reward function for the state estimation task as

$$r = \left\{ \exp\left(-\frac{(y - C\hat{\mathbf{x}})^2}{\sigma_r^2} \right) - \sum_{j=1}^{2} c(a_j) \right\} \Delta t, \tag{4.7}$$

where $\sigma_r^2 = 0.25$ and $c(a_j)$ is the output cost term. We saturate the output of the policy using a sigmoid function in equation A.3 (Doya, 2000; see appendix A) with the maximum output $\mathbf{a}^{\max} = (a_1^{\max}, a_2^{\max}) = (5.0, 5.0)^T$. Then we use the corresponding cost function,

$$c(a_j) = d \int_0^{a_j} \Omega^{-1}\left(\frac{u}{a_j^{\max}} \right) du, \tag{4.8}$$

where $d = 0.1$ and Ω are monotonically increasing functions (see appendix A).

The learning parameters were set as follows. The learning rate for the value function was $\alpha = 0.5$, the learning rate for the policy was $\beta = 0.5$, the discount factor was $\gamma = 0.99$, and the decay factor for the eligibility trace was $\eta = 0.95$. A 5×5 basis normalized gaussian network for two-dimensional space $\hat{\mathbf{x}}$ was used for the state estimation policy in equations 3.15 and 3.17 and the value function in equation 3.7. We located basis functions on a grid with an even interval in each dimension of the input space ($-\pi \leq \theta \leq \pi$, $-3\pi \leq \omega \leq 3\pi$).

In the current simulations, the centers are fixed in a grid, which is analogous to the "boxes" approach (Barto, Sutton, & Anderson, 1983) often used in discrete RL. Grid allocation of the basis functions enables efficient calculation of their activation as the outer product of the activation vectors for individual input variables. We used the fourth-order Runge-Kutta method with a time step of $\Delta t = 0.01$ sec to derive the discrete pendulum dynamics in equations 4.4 and 4.6 and put a zero-order hold on the input as $u_c(s) = u(k)$ for $k\Delta t \leq s < (k+1)\Delta t$.

4.1.1 Estimation of Free-Swing Trajectories: Linear Pendulum. We first consider a linear problem in order to check whether the pole of the error dynamics learned by the RLSE is inside a unit circle. Thus, we have considered the pendulum dynamics only near the stable equilibrium point $\mathbf{x} = (0, 0)^T$,

$$\boldsymbol{F}(\mathbf{x}(t), \mathbf{u}(t)) = \begin{pmatrix} 0 & 1 \\ -\frac{g}{l} & -\frac{\mu}{ml^2} \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ \frac{1}{ml^2} \end{pmatrix} u_c(t), \tag{4.9}$$

where the discrete pendulum dynamics was derived with equation 4.4. Here, since the RLSE attempts to learn a policy to estimate free-swing trajectories, we always set the control input to zero $u(k) = 0$. Let us consider two types of observation noise that have different amplitudes: $R = 1.0\Delta t$ and $R = 10.0\Delta t$. Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), 0.0)^T$ and an initial estimate $\hat{\mathbf{x}}(0) = (\hat{\theta}(0), 0.0)^T$, where $\theta(0)$ was selected from a uniform distribution that ranged over $-\frac{1}{4}\pi < \theta(0) < \frac{1}{4}\pi$. Then $\hat{\theta}(0)$ was selected from a uniform distribution around $\theta(0)$, which ranged over $-\frac{1}{6}\pi + \theta(0) < \hat{\theta} < \frac{1}{6}\pi + \theta(0)$. Each trial was terminated after 20 sec.

We applied an acquired estimation policy after 100 learning trials to estimate linear pendulum states and used a deterministic policy, represented by the mean of the acquired stochastic policy, to illustrate the estimation performance. The initial state was set as $\mathbf{x}(0) = (-\frac{1}{6}\pi, 0.0)^T$ and $\hat{\mathbf{x}}(0) = (-\frac{1}{3}\pi, \frac{1}{2}\pi)^T$. Figures 3A and 3B show the successful estimation of the linearized pendulum state, and Figure 3C indicates that the pole of the error dynamics stayed within the unit circle, that is, the absolute eigenvalues of error dynamics $|\lambda|$ are always less than one, $|\lambda| < 1$, during state estimation. Furthermore,

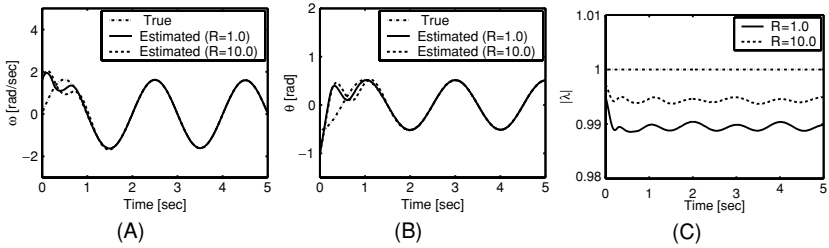

Figure 3: Estimated pendulum states and absolute eigenvalues of error dynamics. (A) Estimated angular velocity ω. (B) Estimated joint angle θ. (C) Time course of absolute eigenvalues $|\lambda(A - LC)|$.

the RLSE acquired different estimation policies with different observation noise. The error dynamics had slower convergence; absolute eigenvalues $|\lambda|$ had larger values with greater observation noise ($R = 10.0\Delta t$), all results consistent with the properties of the Kalman filter.

4.1.2 Estimation of Free-Swing Trajectories: Nonlinear Pendulum. Now we try to estimate free-swing trajectories of nonlinear pendulum dynamics, equation 4.4, by using the RLSE. Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), \omega(0))^T$, where $\theta(0)$ was selected from a uniform distribution that ranged over $-\frac{\pi}{2} < \theta(0) < \frac{\pi}{2}$, and $\omega(0)$ was selected from a uniform distribution ranging over $-\frac{\pi}{2} < \omega(0) < \frac{\pi}{2}$. In addition, each trial was started from an initial estimate $\hat{\mathbf{x}}(0) = (\hat{\theta}(0), \hat{\omega}(0))^T$, where $\hat{\theta}(0)$ was selected from a uniform distribution around $\theta(0)$ that ranged over $-\frac{\pi}{4} + \theta(0) < \hat{\theta}(0) < \frac{\pi}{4} + \theta(0)$, and $\hat{\omega}(0)$ was selected from a uniform distribution around $\omega(0)$ that ranged over $-\frac{\pi}{4} + \omega(0) < \hat{\omega}(0) < \frac{\pi}{4} + \omega(0)$. Each trial was terminated after 20 sec.

The following three conditions are considered:

1. Joint angular velocity ω is observed—$C = [0\ \ 1]$ in equation 4.3.
2. Joint angle θ is observed—$C = [1\ \ 0]$ in equation 4.3.
3. Combined state of joint angle θ and angular velocity ω is observed—$C = [1\ \ 1]$ in equation 4.3.

Figure 4 illustrates the learning performance of the RLSE under the three different conditions.

We applied the acquired policy after 100 learning trials to estimate free-swing trajectories under each condition using deterministic policy, which is represented by the mean of the acquired stochastic policy, to demonstrate the estimation performance. Initial states were set as $\mathbf{x}(0) = (-\frac{\pi}{2}, 0.0)^T$ and $\hat{\mathbf{x}}(0) = (-\pi, \pi)^T$. Results reveal that acquired policies successfully estimated the true state trajectories (see Figure 5).

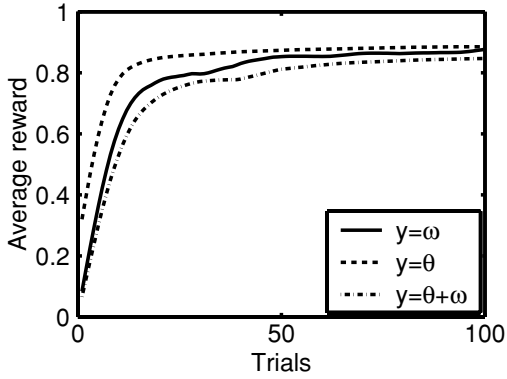

Figure 4: Learning performance of free-swing trajectory estimation. (Average over 10 simulation runs. We filtered the data with a moving average of 10 trials.) If the initial estimated state $\hat{\mathbf{x}}(0)$ is equal to the actual state $\mathbf{x}(0)$ and the RLSE perfectly follows an actual state trajectory, the average reward becomes 1 without considering cost term $c(\mathbf{a})$.

4.1.3 Pendulum Swing-Up Task in POMDPs: With Exact Model. Here we try to acquire the swing-up policies while estimating state variables of the pendulum simultaneously. First, we assume that we have an exact model of the pendulum.

We apply the value-gradient-based reinforcement learning (Doya, 2000) to acquire pendulum swing-up policies in POMDPs. The task of swinging up a pendulum (Doya, 2000) is a simple but strongly nonlinear control task. Here we assume that the swing-up policies cannot observe the position of the pendulum θ; only the angular velocity ω can be observed. The reward function for the swing-up control policies is given as

$$q(\mathbf{x}, u) = \{-\cos\theta - v(u)\}\Delta t, \tag{4.10}$$

where $v(u)$ is a cost function for control variable u (see appendix A). We use the same reward function in equation 4.7 for the estimation policies.

Each trial was started from an initial state $\mathbf{x}(0) = (\theta(0), \omega(0))^T$, where $\theta(0)$ was selected from a uniform distribution that ranged over $-\frac{\pi}{4} < \theta(0) < \frac{\pi}{4}$, and $\omega(0)$ was selected from a uniform distribution ranging over $-\frac{\pi}{4} < \omega(0) < \frac{\pi}{4}$. In addition, each trial was started from an initial estimate $\hat{\mathbf{x}}(0) = (\hat{\theta}(0), \hat{\omega}(0))^T$, where $\hat{\theta}(0)$ were selected from a uniform distribution around $\theta(0)$ that ranged over $-\frac{\pi}{4} + \theta(0) < \hat{\theta}(0) < \theta(0) + \frac{\pi}{4}$, and $\hat{\omega}(0)$ were selected from a uniform distribution around $\omega(0)$, which ranged over $-\frac{\pi}{4} + \omega(0) < \hat{\omega}(0) < \omega(0) + \frac{\pi}{4}$. Each trial was terminated after 20 sec. As a measure of the swing-up performance, we defined the time for which the pendulum

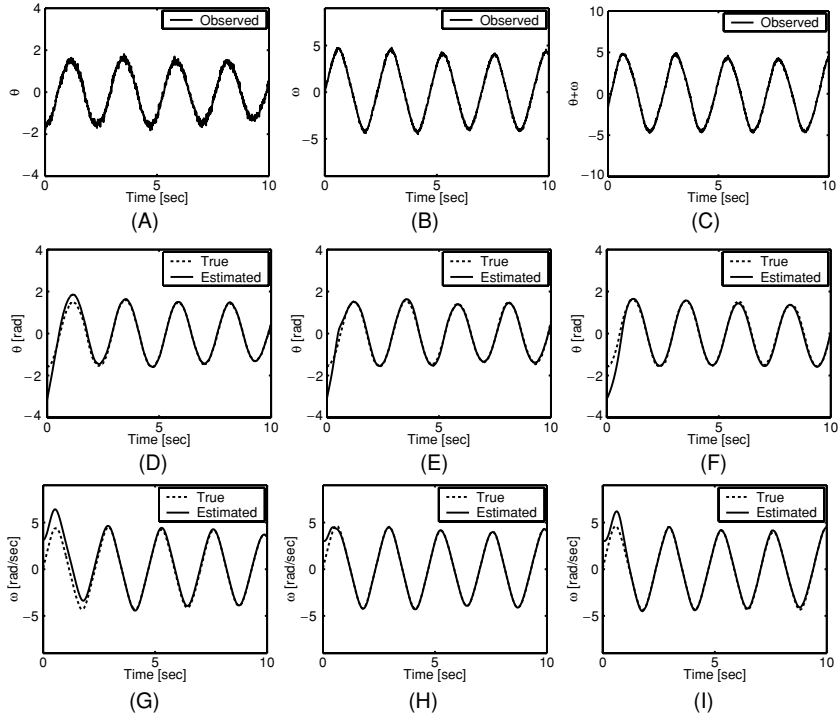

Figure 5: Estimation of free-swing trajectories: (A–C) Observed state. (D–F) Estimated joint angle θ. (G–I) Estimated angular velocity ω. (A, D, G) Joint angle θ is observed. (B, E, H) Joint angular velocity ω is observed. (C, F, I) Combined state of joint angle and angular velocity $\theta + \omega$ is observed.

stayed up ($\pi - \frac{\pi}{4} < |\theta| < \pi + \frac{\pi}{4}$) as t_{up}. A trial was regarded as successful when $t_{up} > 10$ sec. We used the number of trials made before achieving 10 successful trials as the measure of the learning speed and limited the output torque of the controller as $u_{\max} = 3.0\ N \cdot m$. Appendix A explains the implementation of the value-gradient-based policy.

Figure 6A shows the learning performance of the pendulum swing-up task. After 85 trials (averaged over 10 simulation runs), the swing-up policies and the state estimation policies were acquired. This result reveals that RLSE can obtain estimation policies that are capable of acquiring pendulum swing-up policies.

Several studies (Jaakkola et al., 1995; Baird & Moore, 1999; Kimura & Kobayashi 1998; Meuleau et al., 1999) suggested that we can find a good memoryless or memory-based policy for POMDPs using the policy gradient method. Here we directly applied the policy gradient method (Kimura & Kobayashi 1998) to learn the swing-up controller (see appendix B). If full

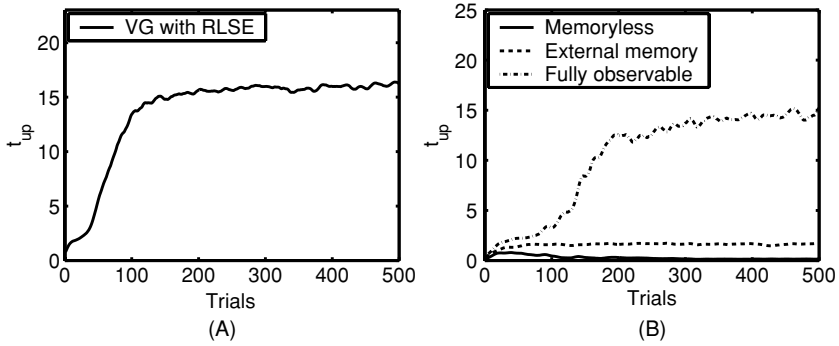

Figure 6: Learning performance for the pendulum swing-up task (average over 10 simulation runs). (A) Value-gradient polices with RLSE (we filtered the data with a moving average of 10 trials). (B) Direct use of policy gradient (we filtered the data with a moving average of 10 trials).

state information (θ, ω) is available to the controller, this policy gradient controller can acquire the swing-up policy with 171 trials.

Now we apply this controller to the pendulum swing-up task under the same POMDP assumption: that the controller cannot observe the position of the pendulum θ. The solid line in Figure 6B shows the learning performance for 500 trials. This result indicates that the pendulum swing-up policy cannot be acquired by observing only the angular velocity of the pendulum ω using the policy gradient method. We also applied the policy gradient method with external memory. Because the pendulum dynamics is represented by a second-order differential equation, we defined the state space of the controller as $(\omega(t), \omega(t - \Delta T))^T$, where $\Delta T = 0.5$ sec, which is a quarter of the actual period of the linearized pendulum. The dashed line in Figure 6B shows learning performance for 500 trials. This result also indicates that the pendulum swing-up policy cannot be acquired through observing only the angular velocity of the pendulum ω even when using the external memory.

Figure 7 shows swing-up trajectories using RLSE from an initial state $\mathbf{x}(0) = (\theta(0), \omega(0)) = (0.0, 0.0)^T$ and an initial estimate $\hat{\mathbf{x}}(0) = (\hat{\theta}(0), \hat{\omega}(0)) = (-\frac{1}{2}\pi, \pi)^T$.

4.1.4 Pendulum Swing-Up Task in POMDPs: With Modeling Errors. In the previous section, it was assumed that we have an exact model of the pendulum dynamics. However, especially in the real environment, it is difficult to have a perfect model of the target dynamics. Here, we try to acquire an estimation policy while at the same time also learning a model parameter of the pendulum. In other words, we attempt to acquire the swing-up control policy, the state estimation policy, and the model estimation policy

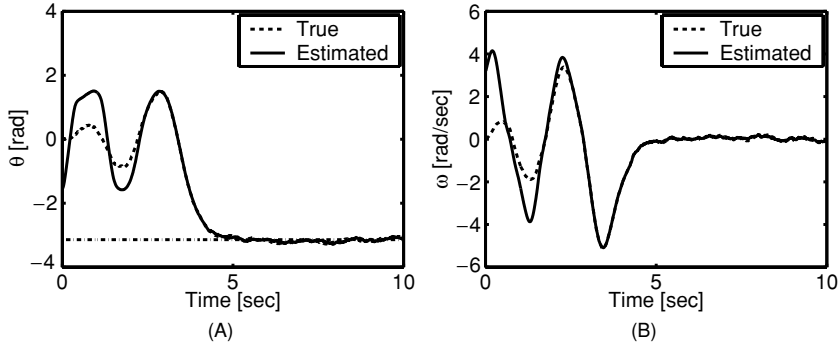

Figure 7: Swing-up trajectories and estimation performance. The dash-dotted line represents the upright position. The initial state $\mathbf{x}(0) = (\theta(0), \omega(0)) = (0.0, 0.0)^T$; the initial estimate $\hat{\mathbf{x}}(0) = (\hat{\theta}(0), \hat{\omega}(0)) = (-\frac{1}{2}\pi, \pi)^T$. (A) Estimated joint angle $\hat{\theta}$. (B) Estimated angular velocity $\hat{\omega}$.

simultaneously. A 1×1, that is, only one, basis normalized gaussian network was used as the model estimation policy in equations 3.21 and 3.23. We set the decay factor for the eligibility trace of the model estimation policy as $\eta = 0.90$.

After 227 trials and 128 trials (average over 10 simulation runs), the swing-up policies and the state estimation policies were acquired with initially incorrect models, $l = 0.5$ m and $l = 2.0$ m, respectively (see Figure 8A).

Figure 8B shows the learning performance of the model estimation policy with two different modeling errors. Results reveal that the RLSE could acquire the correct model parameter during learning of the swing-up task and the state estimation task. The estimated mean model parameter of 10 simulation runs after 500 trials was $l = 1.03$ m, and the variance was 1.05×10^{-2} with the initially incorrect model $l = 0.5$ m. Furthermore, with the initially incorrect model $l = 2.0$ m, the estimated mean model parameter for 10 simulation runs after 500 trials was $l = 1.08$ m, and the variance was 0.27×10^{-2}.

4.1.5 Pendulum Swing-Up Task in POMDPs: With Unknown Model. Here we show that the RLSE can be used even when a form of the pendulum dynamics is unknown. We assume to know that the pendulum model is an input-affine system and know the input matrix of equation 4.5:

$$\mathbf{F}(\hat{\mathbf{x}}(t), \mathbf{u}(t)) = \begin{pmatrix} \omega \\ f_\omega(\mathbf{x}) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_c. \tag{4.11}$$

The model estimation policy introduced in section 3.3 can be directly used to estimate the dynamics model of the pendulum. We used the output

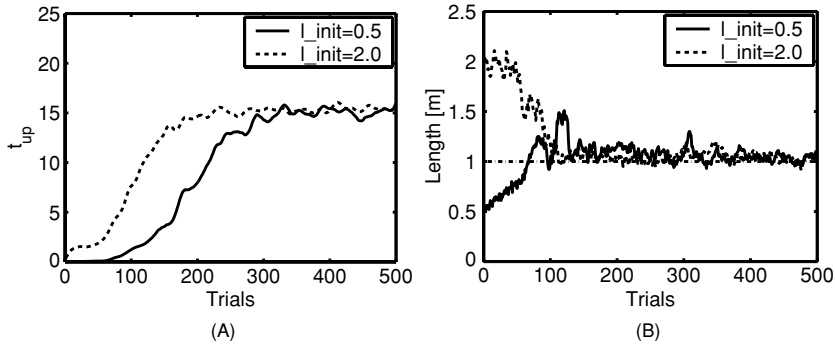

Figure 8: Learning the swing-up controller and the state estimation policy with the model estimation policy. l_{init} represents the initial model parameter of the pendulum length. (A) Learning performance of the swing-up policy when using RLSE (average over 10 simulation runs; we filtered the data with a moving average of 10 trials). (B) Estimated length of the pendulum at each trial.

of the model estimation policy as $f_\omega(\mathbf{x}) \approx 10.0 \times \mathbf{p}$. A 9×4 basis normalized gaussian network was used for the model estimation policy in equations 3.21 and 3.23. We used the learning parameter set $\sigma_r = 0.2$, $\alpha = 1.0$, $\beta = 1.0$, $\gamma = 0.98$, $\eta = 0.90$ for the state estimation policy and $\eta = 0.96$ for the model estimation policy.

Figure 9A shows an acquired pendulum dynamics. A sinusoidal shape that represents the pendulum dynamics (see equation 4.5) was successfully acquired. Figure 9B illustrates the learning performance of swing-up policy with estimating pendulum dynamics and state variables. After 692 trials (average over 10 simulation runs), the swing-up policies and the state estimation policies were acquired with an initially unknown model.

4.2 Application to a Biped Model. Finally, the RLSE is applied to a biped model (see Figure 10). The task in this case for the biped robot is walking down a slope without falling over. A biped dynamics model introduced in Goswami, Thuilot, and Espiau (1996) was employed here. Although it is well known that this two-linked biped model can walk down the slope without any actuation, it is necessary to set proper initial angular velocities at each degree of freedom to generate this passive walking pattern. In this study, we try to acquire a controller that can make the biped robot walk even when the initial angular velocities are not suitable for passive walking.

We define the state vector as $\mathbf{x} = (\theta_{sw}, \theta_{st}, \omega_{sw}, \omega_{st})^T$, where θ_{sw} and θ_{st} are leg angles and ω_{sw} and ω_{st} are leg angular velocities, and we define applied torque at the hip joint $u = T$ as the action. We assume that the covariance of the system noise $\mathbf{n}(k)$ and the observation noise $\mathbf{v}(k)$ are $\Gamma = \mathrm{diag}(0.01, 0.01, 0.01, 0.01)\Delta t$ and $R = \mathrm{diag}(0.01, 0.01)\Delta t$, respectively.

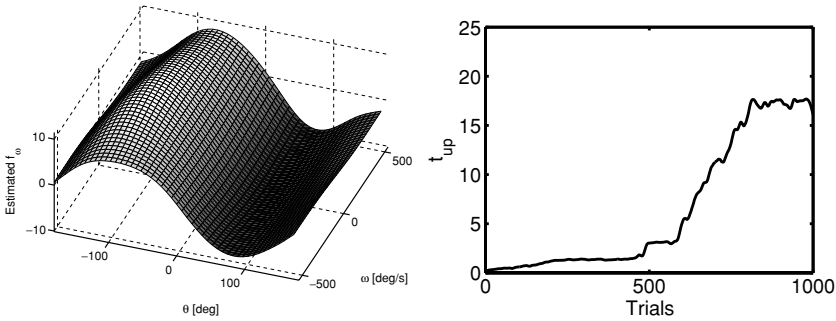

Figure 9: Learning the state estimation policy with the dynamics of the pendulum in the pendulum swing-up task. (A) Acquired pendulum model. A sinusoidal shape that represents the pendulum dynamics was successfully acquired. (B) Learning performance of the swing-up policy with estimating pendulum dynamics and state variables (average over 10 simulation runs; we filtered the data with a moving average of 10 trials).

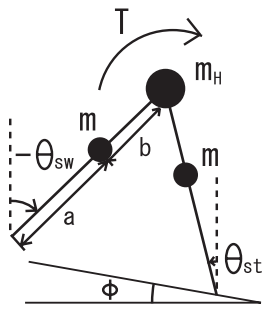

Figure 10: Biped model: $m_H = 10$ kg, $m = 5$ kg, $a = b = 0.5$ m, $\phi = 0.03$ rad.

This biped model includes a discontinuity when the swing foot touches the ground. Furthermore, we assume that only the leg angles θ_{sw} and θ_{st} can be observed; in other words, we consider linear observation function $\mathbf{h}(\mathbf{x}(k)) = C\mathbf{x}(k)$ with observation matrix $C = [1\ 1\ 0\ 0]$ in equation 2.2. We also assume that we know the exact biped model.

The learning parameters were set as follows. The learning rate for the value function was $\alpha = 0.5$, the learning rate for the policy was $\beta = 1.0$, the discount factor was $\gamma = 0.98$, and the decay factor for the eligibility trace was $\eta = 0.97$. A $5 \times 5 \times 10 \times 10$ basis normalized gaussian network for four-dimensional space $\hat{\mathbf{x}}$ was used for the state estimation policy in equations 3.15 and 3.17 and the value function in equation 3.7. We

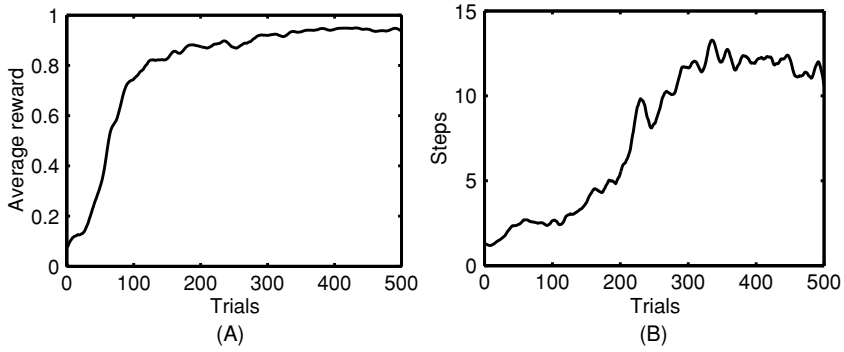

Figure 11: (A) Learning performance of estimation policy for the biped walking task (average over 10 simulation runs; we filtered the data with a moving average of 10 trials). (B) Learning performance for the biped walking task. The horizontal axis represents the number of walking steps (average over 10 simulation runs; we filtered the data with a moving average of 10 trials).

located basis functions on a grid with an even interval in each dimension of the input space ($-\frac{1}{4}\pi \leq \theta_{sw} \leq \frac{1}{4}\pi$, $-\frac{1}{4}\pi \leq \theta_{st} \leq \frac{1}{4}\pi$, $-3\pi \leq \omega_{sw} \leq 3\pi$, $-3\pi \leq \omega_{st} \leq 3\pi$). We used the fourth-order Runge-Kutta method with a time step of $\Delta t = 0.01$ sec to derive the discrete biped dynamics and used the reward function

$$r = \left\{ \exp\left(-(\mathbf{y} - C\hat{\mathbf{x}})^T S(\mathbf{y} - C\hat{\mathbf{x}})\right) - \sum_{j=1}^{4} c(a_j) \right\} \Delta t, \tag{4.12}$$

where $S = diag(1/\sigma_r^2, 1/\sigma_r^2)$ with $\sigma_r^2 = 0.001$. We saturate the output of the policy using a sigmoid function in equation A.3 with the maximum output $\mathbf{a}^{\max} = (a_1^{\max}, a_2^{\max}, a_3^{\max}, a_4^{\max}) = (150.0, 150.0, 150.0, 150.0)$. Again, we used value-gradient-based policy (see appendix A). The penalty -0.5 is given to the biped controller when the robot falls over.

Each trial was started from an initial state $\mathbf{x}(0) = (0.0, 0.0, \omega_{sw}(0), \omega_{st}(0))^T$, where $\omega_{sw}(0)$ and $\omega_{st}(0)$ were selected from uniform distributions that ranged over $2.0 < \omega_{sw}(0) < 2.2$ rad/sec and $-0.5 < \omega_{st}(0) < -0.7$ rad/sec. In addition, each trial was started from initial estimate $\hat{\mathbf{x}}(0) = (0.0, 0.0, \hat{\omega}_{sw}(0), \hat{\omega}_{st}(0))^T$, where $\hat{\omega}_{sw}(0)$ and $\hat{\omega}_{st}(0)$ were selected from uniform distributions ranging over $-\frac{\pi}{6} + \omega_{sw} < \hat{\omega}_{sw}(0) < \frac{\pi}{6} + \omega_{sw}(0)$ and $-\frac{\pi}{6} + \omega_{st}(0) < \hat{\omega}_{st}(0) < \frac{\pi}{6} + \omega_{st}(0)$. Each trial was terminated after 10 sec. A trial was regarded as successful when the number of steps is more than 10.

Figure 11A shows results of the learning performance of estimation policies. The RLSE successfully acquired estimation policies in the biped walking task. Figure 11B illustrates the learning performance for a biped

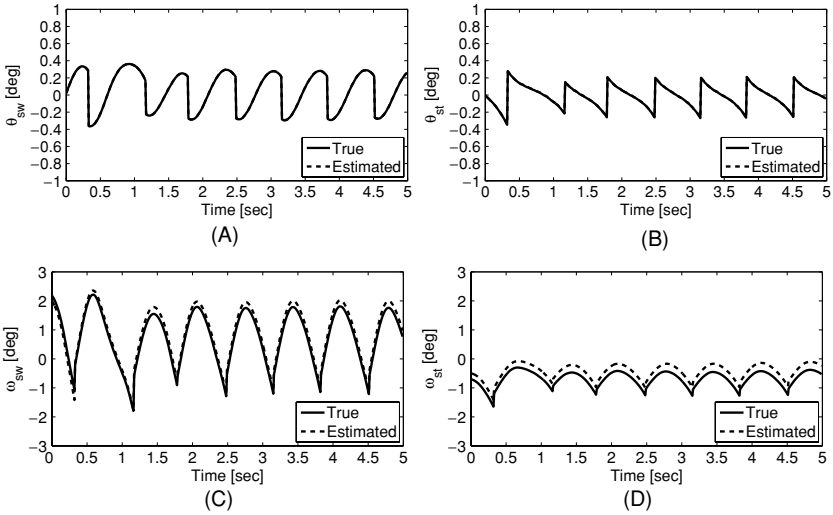

Figure 12: Biped walking trajectory and estimation performance. Initial state $\mathbf{x} = (\theta_{sw}, \theta_{st}, \omega_{sw}, \omega_{st}) = (0.0, 0.0, 2.2, -0.7)$; initial estimate $\hat{\mathbf{x}} = (\hat{\theta}_{sw}, \hat{\theta}_{st}, \hat{\omega}_{sw}, \hat{\omega}_{st}) = (0.0, 0.0, 2.0, -0.5)$. (A) Swing leg angle θ_{sw}. (B) Stance leg angle θ_{st}. (C) Swing leg angular velocity ω_{sw}. (D) Stance leg angular velocity ω_{st}.

walking task. The biped controllers were successfully acquired after 273 trials (average over 10 simulation runs).

Figure 12 shows the estimation performance for a biped walking trajectory. The RLSE could estimate state variables even for a biped model. Although slight estimation errors remained in angular velocities, biped walking policies could be successfully acquired by using RL.

Figure 13A displays the initial performance of the control policy. The biped fell over right after its first step. Figure 13B gives the acquired performance of the control policy after 500 trials. The RLSE could be successfully used to acquire a biped walking controller.

5 Discussion

A conventional way to solve POMDPs is to use belief state representation. By doing so, the problem can be considered as MDPs in continuous state space (Littman, Cassandra & Kaelbling, 1995). However, the conventional methods can be applied to only small problems in discrete state space, whereas the RLSE can be applied to problems in continuous state space.

Backprop-through-time algorithms (Erdogmus, Genc, & Principe, 2002; Xu, Erdogmus, & Principe, 2005) were proposed to learn a state estimator. Although this approach can be applied to the state estimation problems

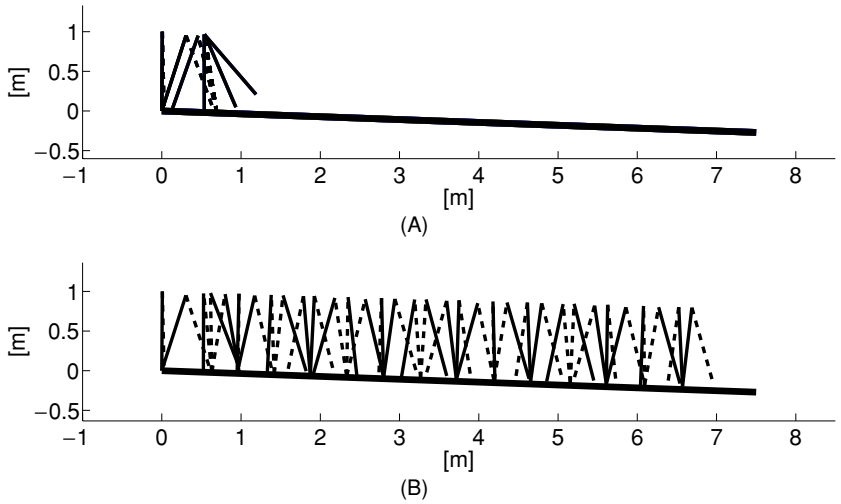

Figure 13: Simultaneous learning of the biped walking controller and the state estimator. The solid line represents the right leg, and the dotted line represents the left leg. Initial state $\mathbf{x} = (\theta_{sw}, \theta_{st}, \omega_{sw}, \omega_{st})^T = (0.0, 0.0, 2.2, -0.7)^T$; initial estimate $\hat{\mathbf{x}} = (\hat{\theta}_{sw}, \hat{\theta}_{st}, \hat{\omega}_{sw}, \hat{\omega}_{st})^T = (0.0, 0.0, 2.0, -0.5)^T$. (A) Initial performance. (B) Acquired performance.

in continuous state space, backprop-through-time algorithms can be used only for off-line estimation. On the other hand, RL can improve parameters of a state estimator through online update since RL can reduce an accumulated estimation error over time instead of reducing only an instantaneous estimation error.

Application of genetic algorithms to the learning state estimator problem was proposed by Porter and Passino (1995). In the genetic algorithm approach, the gain parameter of a state estimator L in equation 3.15 was constant, whereas the RLSE acquired state-dependent gain $L(\hat{\mathbf{x}})$. This constant gain L can limit the performance of the state estimator if we apply the state estimator to a nonlinear dynamics.

Thau (1973) and Raghavan and Hedrick (1994) proposed a nonlinear observer to design an observer that can guarantee the convergence of the error dynamics even for a nonlinear environment. However, this approach considered any nonlinear term of a target dynamics as a modeling error from a linear dynamics. Consequently, the applications of this method are limited.

The duality between estimation and control is discussed in Crassidis and Junkins (2004). They showed that the RTS smoother can be derived from optimal control theory. They used the negative log-likelihood function for the

reward and the cost functions, that is, $e(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})) = -\frac{1}{2}(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}))^T R^{-1}(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}))$ and $c(\mathbf{a}) = \frac{1}{2}\mathbf{a}^T \Gamma^{-1}\mathbf{a}$ in equation 3.4. To acquire the RTS smoother, we need off-line calculation to derive the costate vector. In our method, we can acquire estimation policies without explicit backward calculation by using the reinforcement learning framework.

6 Conclusion

We proposed a novel use of reinforcement learning, the reinforcement learning state estimator (RLSE), to estimate hidden variables and parameters of nonlinear dynamical systems. RLSE was applied to the pendulum swing-up task, and simultaneous learning of the RLSE and a swing-up controller, which used the estimated state of the RLSE, was accomplished, whereas the direct use of the policy gradient method could not accomplish this task. We also showed that a swing-up policy could be acquired even in the case that the RLSE initially held an incorrect model of the pendulum or did not know the form of the model. Application of the RLSE to the two-linked biped model was also presented.

In this study, we used a gaussian distribution to represent an estimation policy. In the future, it may be possible to use a mixture of gaussian representations for the estimation policy to cope with nongaussian state transition probability.

Because the state estimation policy and the model estimation policy are mutually dependent, future work will involve analysis of our method to show a relationship between the accuracy of state estimation and model approximation performance.

Appendix A: Value-Gradient-Based Policy for Pendulum Swing-Up Task

The approximated value function is represented as

$$\hat{V}^c(\mathbf{x}) = \sum_{i=1}^{N_c} v_i^c b_i(\mathbf{x}), \tag{A.1}$$

where v_i^c is the ith parameter of the function approximator and $b_i(\mathbf{x})$ is the normalized gaussian basis function (see appendix C). A 21×21 basis normalized gaussian network for two-dimensional space $\hat{\mathbf{x}}$ was used for the value function approximation in the swing-up task. A $5 \times 5 \times 17 \times 17$ basis normalized gaussian network for four-dimensional space $\hat{\mathbf{x}}$ was used for the value function approximation in the biped walking task.

We use the value-gradient-based policy represented as

$$u = u^{\max}\Omega\left(\frac{\partial V}{\partial \mathbf{x}}\frac{\partial \mathbf{f}(\mathbf{x}, u)}{\partial u}\right), \tag{A.2}$$

where $u_{\max} = 3.0N \cdot m$ for the pendulum swing-up task and $u_{\max} = 20N \cdot m$ for the biped walking task, and we limit the amplitude of the action using the sigmoid function:

$$\Omega(x) = \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right). \tag{A.3}$$

This corresponds to defining the cost function in equation 4.10 as

$$v(u) = d^c \int_0^u \Omega^{-1}\left(\frac{u'}{u^{\max}}\right) du', \tag{A.4}$$

where we set the parameter for the cost function $d^c = 0.1$ for both tasks. The learning rate for the value function was $\alpha = 1.0$, and the discount factor was $\gamma = 0.99$.

Appendix B: Policy Gradient Method for the Pendulum Swing-Up Task

The value function is represented as in equation A.1. Then we construct the model for the swing-up policy based on the normal distribution $\pi_{\mathbf{w}^c}(\mathbf{x}, u) = P(u|\mathbf{x}; \mathbf{w}^c)$,

$$\pi_{\mathbf{w}^c}(\mathbf{x}, u) = \frac{1}{\sqrt{2\pi}\sigma^c(\mathbf{x})}\exp\frac{-(u - \mu^c(\mathbf{x}))^2}{2(\sigma^c(\mathbf{x}))^2}, \tag{B.1}$$

where u is the output of the policy $\pi_{\mathbf{w}^c}$. We limit the amplitude of the action using the sigmoid function in equation A.3. The mean output μ^c of the policy $\pi_{\mathbf{w}^c}$ is modeled by the normalized gaussian network,

$$\mu^c(\mathbf{x}) = \sum_{i=1}^{N^c} w_i^{\mu^c} b_i^c(\mathbf{x}), \tag{B.2}$$

where $w_i^{\mu^c}$ denotes the parameter for the output of the policy $\pi_{\mathbf{w}^c}$, N^c denotes the number of basis functions, and $b^c(\mathbf{x})$ is the normalized gaussian basis

function (see appendix C). We represent the variance σ^c of the policy $\pi_{\mathbf{w}^c}$ using a sigmoid function,

$$\sigma^c(\hat{\mathbf{x}}) = \frac{\sigma_0^c}{1 + \exp(-\sigma^{w^c}(\hat{\mathbf{x}}))}, \tag{B.3}$$

where σ_0^c denotes the scaling parameter, and the state-dependent parameter $\sigma^{w^c}(\hat{\mathbf{x}})$ is also modeled by the normalized gaussian network,

$$\sigma^{w^c}(\hat{\mathbf{x}}) = \sum_{i=1}^{N_c} w_i^{\sigma^c} b_i(\hat{\mathbf{x}}), \tag{B.4}$$

where $w_i^{\sigma^c}$ denotes the ith parameter for the variance of output. Considering equation B.1, the eligibility of the policy parameters are given as

$$\frac{\partial \ln \pi_{\mathbf{w}^c}}{\partial w_i^{\mu^c}} = \frac{u - \mu^c}{(\sigma^c)^2} b_i(\hat{\mathbf{x}}), \tag{B.5}$$

$$\frac{\partial \ln \pi_{\mathbf{w}^c}}{\partial w_i^{\sigma^c}} = \frac{((u - \mu^c)^2 - (\sigma^c)^2)(1 - \sigma^c/\sigma_0^c)}{(\sigma^c)^2} b_i(\hat{\mathbf{x}}). \tag{B.6}$$

We update the parameters by using the update rules in equations 3.11 and 3.12. A 21×21 basis normalized gaussian network for two-dimensional space $\hat{\mathbf{x}}$ was used for the policy represented in equations B.2 and B.4. The learning rate for the control policy was $\beta = 1.0$, and the decay factor for the eligibility trace was $\eta = 0.90$.

Appendix C: Normalized Gaussian Network

Basis functions for the normalized gaussian network are represented as

$$b_i(\mathbf{x}) = \frac{\xi_i(\mathbf{x})}{\sum_{j=1}^{N} \xi_j(\mathbf{x})}, \tag{C.1}$$

where N is the number of basis functions and

$$\xi_j(\mathbf{x}) = e^{-||\mathbf{s}_j^T(\mathbf{x} - \mathbf{c}_j)||^2}. \tag{C.2}$$

The vectors $\mathbf{c}_j$ and $\mathbf{s}_j$ define the center and the size of the jth basis function, respectively.

Acknowledgments

We thank Hirokazu Tanaka and reviewers for their helpful information and comments.

References

Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing, 50*(2), 174–188.

Baird, L. C., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. In M. S. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems, 11*. Cambridge, MA: MIT Press.

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

Crassidis, J. L., & Junkins, J. L. (2004). *Optimal estimation of dynamic systems*. London: Chapman and Hall.

Doucet, A., Godsill, S. J., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing, 10*, 197–208.

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation, 12*(1), 219–245.

Erdogmus, D., Genc, A. U., & Principe, J. C. (2002). A neural network perspective to extended Luenberger observers. *Institute of Measurement and Control, 35*(2), 10–16.

Goswami, A., Thuilot, B., & Espiau, B. (1996). *Compass-like biped robot part I: Stability and bifurcation of passive gaits* (Tech. Rep. No. RR-2996). Rocguencourt, France: INRI.

Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 345–352). Cambridge, MA: MIT Press.

Kalman, R. E., & Bucy, R. S. (1961). New results in linear filtering and prediction theory. *Trans., ASME, Series D, J. Basic Engineering, 83*(1), 95–108.

Kimura, H., & Kobayashi, S. (1998). An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions. In *Proceedings of the 15th Int. Conf. on Machine Learning* (pp. 284–292). San Francisco: Morgan Kaufmann.

Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 362–370). San Francisco: Morgan Kaufmann.

Luenberger, D. G. (1971). An introduction to observers. *IEEE Trans., AC, 16*, 596–602.

McCallum, R. A. (1995). *Reinforcement learning with selective perception and hidden state*. Unpublished doctoral dissertation, University of Rochester.

Meuleau, N., Kim, K. E., & Kaelbling, L. P. (2001). *Exploration in gradient-based reinforcement learning* (Tech. Rep., AI Memo 2001-003). Cambridge, MA: MIT.

Meuleau, N., Peshkin, L., Kim, K., & Kaelbling, L. P. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence* (pp. 427–436). San Francisco: Morgan Kaufmann.

Morimoto, J., & Doya, K. (2002). Development of an observer by using reinforcement learning. In *Proc. of the 12th Annual Conference of the Japanese Neural Network Society* (pp. 275–278). Tottorl, Japan. (in Japanese)

Porter, L. L., & Passino, K. M. (1995). Genetic adaptive observers. *Engineering Applications of Artificial Intelligence, 8*(3), 261–269.

Raghavan, I. R., & Hedrick, J. K. (1994). Observer design for a class of nonlinear systems. *International Journal of Control, 59*(2), 515–528.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Thau, F. E. (1973). Observing the state of nonlinear dynamic systems. *International Journal of Control, 17*(3), 471–479.

Thrun, S. (2000). Monte Carlo POMDPs. In S. A. Solla, T. K. Leen, & K. R. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 1064–1070). Cambridge, MA: MIT Press.

Wan, E., & van der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proc. of IEEE Symposium 2000 (AS-SPCC)*. Piscataway, NJ: IEEE.

Wan, E. A., & Nelson, A. T. (1997). Dual Kalman filtering methods for nonlinear prediction, smoothing, and estimation. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9* (pp. 793–799). Cambridge, MA: MIT Press.

Xu, J. W., Erdogmus, D., & Principe, J. C. (2005). Minimum error entropy Luenberger observer. In *Proc. of American Control Conference*. Piscataway, NJ: IEEE.

Received May 10, 2005; accepted July 17, 2006.