Acceleration of Reinforcement Learning by Policy Evaluation Using Nonstationary Iterative Method

Kei Senda, *Member, IEEE,* Suguru Hattori, Toru Hishinuma, and Takehisa Kohda *Member, IEEE,*

***Abstract*—Typical methods for solving reinforcement learning problems iterate two steps, policy evaluation and policy improvement. This study proposes algorithms for the policy evaluation to improve learning efficiency. The proposed algorithms are based on the Krylov Subspace Method (KSM), which is a nonstationary iterative method. The algorithms based on KSM are tens to hundreds times more efficient than existing algorithms based on the Stationary Iterative Methods (SIM). Algorithms based on KSM are far more efficient than they have been generally expected. This study clarifies what makes algorithms based on KSM makes more efficient with numerical examples and theoretical discussions.**

***Index Terms*—reinforcement learning, policy iteration, policy evaluation, nonstationary iterative method.**

I. Introduction

In a reinforcement learning (RL) problem, state and action are evaluated as a Q-factor, where an appropriate action might be selected by comparing Q-factors. The objective of RL is to obtain the Q-factors that yield the optimal policy. A typical problem solving method using RL is composed of plant estimation steps, policy evaluation steps, and policy improvement steps [1], [2]. Many solution methods are derived from different ways of combining these steps. Once the plant is estimated, an iteration of the policy evaluation steps and the policy improvement steps, e.g. policy iteration and value iteration, will achieve the optimal policy, improving Q-factor efficiently [2], [3]. Hence, this study considers model-based reinforcement learning with an estimated plant. However, the enormous computation cost of the solution methods often becomes an issue. Therefore, this study discusses the efficient algorithms derived by improving the policy evaluation step efficiency. Existing methods are based on the Stationary Iterative Method (SIM) [2], [3]. This study proposes efficient policy evaluation methods based on the Krylov Subspace Method (KSM), which is a nonstationary iterative method.

The policy evaluation algorithms are classified into direct methods and iterative methods. The total number of calculation operations in a direct method is finite. But, the direct method is not used generally since it cannot obtain the Q-factors that are utilized for the policy evaluation until the calculation is completed. On the other hand, an iterative method can terminate its iteration in mid-course since it exponentially decreases the error, and the iterative method algorithm is generally applied to policy evaluation. SIM and KSM are both classified as iterative method algorithms, but KSM has the properties of a direct method, i.e. a finite number of iteration operations.

It is a simple idea to use KSM instead of SIM. But, compared with a general algebraic problem, which will be shown later in a numerical example, there is little difference between the computation efficiency of KSM and SIM. Therefore, this idea has not attracted notice because KSM does not seem to be more efficient than SIM for RL problems.

However, this study will show that KSM can evaluate policy tens to hundreds of times more efficiently than SIM in an RL problem. The entire RL algorithm using KSM also becomes efficient. The achieved results have been far more efficient than expected with KSM. The advantages of KSM have not been systematically examined yet, but this research will reveal them. Simultaneously, this research will clarify the reason why efficient policy evaluation methods can be so efficient. These results are suggestive for future research in RL.

The rest of this paper is organized as follows. The RL problem is formulated in Section II. Some assumptions and structures of the problem are mentioned in Section III. Existing and proposed algorithms are specified in Section IV. A theoretical review evaluating the efficiency of each algorithm is given in Section V. The efficiency of proposed methods is examined with numerical examples in Section VI. Finally some concluding remarks are given in Section VII.

II. Reinforcement Learning Problem

Following general dynamic programming (DP) formulations, this paper treats a discrete-time dynamic system [2]. A state s_i and an action u_k are the discrete variables and the elements of finite sets $\mathcal{S}$ and $\mathcal{U}$, respectively. The state set $\mathcal{S}$ is composed of N states denoted by $s_1, s_2, \ldots, s_N$, and an additional termination state s_0. The action set $\mathcal{U}$ is composed of K actions denoted by $u_1, u_2, \ldots, u_K$. If an agent is in state s_i and chooses action u_k, it will move to state s_j and incur a one-step cost $g(s_i, u_k, s_j)$ within state transition probability $p_{ij}(u_k)$.

This study deals with a discrete-time finite Markov Decision Process (MDP): probability $p_{ij}(u_k)$ is dependent on only current state s_i and action u_k. The system does not explicitly depend on time. Stationary policy μ is a function mapping states into actions with $\mu(s_i) = u_k \in \mathcal{U}$, and μ is given by the corresponding time-independent action selection probability $\pi(s_i, u_k)$. Symbols used in this paper are shown in the Appendix and details follow in the reference [3].

Kei Senda is with the Department of Aeronautics and Astronautics, Kyoto University, Kyoto 615-8540, JAPAN e-mail: senda@kuaero.kyoto-u.ac.jp.

S. Hattori, T. Hishinuma, and T. Kohda are with Kyoto University.

Manuscript received April 10, 2013; revised September 18, 2013 and December 28, 2013.

The DP problems are distinguished from finite horizon problems where the cost accumulates over a finite number of stages and infinite horizon problems where the cost accumulates indefinitely. This study treats only infinite horizon problems, but no generality is lost since finite horizon problems can be converted into infinite horizon problems by regarding time as an extra component of the state. The optimal policy of an infinite horizon problem is generally deterministic and stationary [2].

The expected total cost, Q-factor, starting from an initial state $s^{(0)} = s_i$ and an initial action $\mu\left(s^{(0)}\right) = u_k$, and using a stationary policy μ is

$$Q^{\mu}(s_i, u_k) = E\left[\sum_{m=0}^{\infty} \alpha^m g\left(s^{(m)}, \mu\left(s^{(m)}\right), s^{(m+1)}\right)\right]$$

where $E[\cdot]$ denotes an expected value and α is a scalar called the discount factor ($0 < \alpha \leq 1$). The optimal Q-factor is defined as $Q^*(s_i, u_k) = \min_{\mu} Q^{\mu}(s_i, u_k)$, and μ is optimal if $Q^{\mu}(s_i, u_k) = Q^*(s_i, u_k),\ ^{\forall}(s_i, u_k)$. This study considers stochastic shortest path problems that are a class of infinite horizon problems. It is assumed that $\alpha = 1$ but there is a cost-free termination state s_0 where $p_{00}(u_k) = 1,\ g(s_0, u_k, s_0) = 0,\ Q(s_0, u_k) = 0,\ ^{\forall}u_k$. Under those conditions, the goal is to find the optimal policy minimizing $Q^{\mu}(s_i, u_k)$, i.e. to reach the state s_0 with minimum expected total cost. A method for solving this problem can be applied to many RL problems [2].

III. Solution Methods for RL

A. Fundamental Assumptions for Solution Methods

The optimal policy is obtained by having a proper policy and improving its Q-factor as well as many other RL methods. A stationary policy is said to be proper if it leads to the termination state s_0 from any initial state s_i within M stages ($0 < M < \infty$) with positive probability. For example, an ϵ-greedy policy and a softmax action selection law used as a behavior policy are proper [1]. The Q-factor is calculated by state transition probabilities which are estimated and retained by sampling. This study calculates the Q-factors using the state transition probability model, which is supposed to be estimated so accurately that the estimation error does not affect the discussions in this study.

B. Fundamental Constructions for Solution Methods

Under these assumption, there are many solution methods to obtain the optimal Q-factor $\boldsymbol{Q}^*$ and the corresponding optimal policy. This study considers policy iteration, starting from a proper policy μ_0 and generating a sequence of proper policies $\mu_1, \mu_2, \ldots$. A policy evaluation step solves the following linear algebraic equations for $\boldsymbol{Q}^{\mu_l}$ concerning policy μ_l given by probability $\pi_l(s_i, u_k)$:

$$Q^{\mu_l}(s_i, u_k) = \sum_{j=0}^{N} p_{ij}(u_k)\left\{g(s_i, u_k, s_j) + \sum_{\ell=1}^{K} \pi_l(s_j, u_\ell) Q^{\mu_l}(s_j, u_\ell)\right\},\ ^{\forall}(s_i, u_k), \quad (1)$$

where $p_{ij}(u_k)$ is given. Eq. (1) is called Bellman's equation when μ_l is optimal. Then, the policy improvement step obtains a new policy μ_{l+1}, a greedy policy, which is deterministic as:

$$\mu_{l+1}(s_i) = \arg\min_{u_k} Q^{\mu_l}(s_i, u_k),\ ^{\forall}s_i. \quad (2)$$

These two steps are repeated until the obtained policy μ_l satisfies $\boldsymbol{Q}^{\mu_{l+1}} = \boldsymbol{Q}^{\mu_l}$. Policy iteration terminates after finding the optimal policy μ^* with a finite number of policy improvement steps.

Value iteration is a method in which the policy evaluation step is terminated after only one iteration, and it moves into a policy improvement step. TD-learning, e.g. TD(λ), is a method between policy iteration and value iteration, and Q-learning is an approximate method of value iteration. This being the case, most methods are based on policy evaluation and policy improvement. From now on, this study focuses on making policy evaluation efficient.

In addition to the method based on the Q-factor, there is a method based on state value function, i.e. J-factor. The J-factor based method calculates all Q-factors of action and state pairs using J-factors in policy improvement steps. In this study, Q-factors are directly calculated without using J-factors because this method makes it easy to evaluate the computation costs and so on. On the other hand, in every following algorithm, the computation cost of the Q-factor method is about K times as many as that of the J-factor method. Therefore, the amounts of computation cost of J-factor methods are almost proportional to Q-factor methods shown in this study.

IV. Policy Iteration Algorithms

A. SIM Type Algorithms

Existing policy evaluation algorithms are based on SIM. This paper shows their outline according to references [2], [3]. Consider a situation in which we seek a solution $\boldsymbol{Q}^{\mu}$ satisfying Eq. (1) for proper policy μ with action selection probability $\pi(s_i, u_k)$. The iteration will be terminated if the obtained solution satisfies required accuracy, because it is guaranteed to converge on the true solution as $m \to \infty$.

1) Policy Evaluation Iteration (PEI) Algorithm: The following mapping H_{μ} is applied:

$$\begin{aligned} Q^{(m+1)}(s_i, u_k) &= H_{\mu}\left(\boldsymbol{Q}^{(m)}\right) \\ &\equiv \sum_{j=0}^{N} p_{ij}(u_k)\left\{g(s_i, u_k, s_j) + \sum_{\ell=1}^{K} \pi(s_j, u_\ell) Q^{(m)}(s_j, u_\ell)\right\}. \end{aligned} \quad (3)$$

2) Jacobi Algorithm: The following mapping F_{μ} is applied:

$$\begin{aligned} Q^{(m+1)}(s_i, u_k) &= F_{\mu}\left(\boldsymbol{Q}^{(m)}\right) \\ &\equiv \frac{1}{1 - p_{ii}(u_k)\pi(s_i, u_k)} \sum_{j=0}^{N} p_{ij}(u_k)\left\{g(s_i, u_k, s_j) + \sum_{\ell=1}^{K} \pi(s_j, u_\ell) Q^{(m)}(s_j, u_\ell)(1 - \delta_{ij}\delta_{k\ell})\right\}, \end{aligned} \quad (4)$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. The method using Eq. (3) or Eq. (4) is a synchronous iteration since iteration numbers updating all Q-factor elements are the same as m.

3) Gauss-Seidel (GS) Algorithm: The following asynchronous iteration is applied:

$$Q^{(m+1)}(s_i, u_k) = F_\mu\left(\hat{\boldsymbol{Q}}\right), \tag{5}$$

where $\hat{\boldsymbol{Q}}$ denotes a vector containing the newest Q-factor elements.

4) SOR Algorithm: The following asynchronous iteration is applied:

$$Q^{(m+1)}(s_i, u_k) = (1-\omega)Q^{(m)}(s_i, u_k) + \omega F_\mu\left(\hat{\boldsymbol{Q}}\right), \tag{6}$$

where ω denotes a relaxation factor and $0 < \omega < 2$ is a necessary condition for convergence.

B. KSM Type Algorithms

This paper proposes policy evaluation algorithms based on KSM [4], [5], [6], which are different from existing ones. We then developed the Conjugate Gradient (CG) algorithm [7] and its extensions, i.e. the Bi-Conjugate Gradient (BCG) [8] and the Block Product-type Krylov Subspace Method (BPKSM) [9] algorithms. The $\boldsymbol{A} \in \boldsymbol{R}^{NK\times NK}$ is a coefficient matrix where $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P\Pi}$ is defined by using symbols in the Appendix. The $(\boldsymbol{a}, \boldsymbol{b})$ represents the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

1) CG Algorithm:

1) Let $m = 0$ and prepare the initial vector $\boldsymbol{Q}^{(0)}$,

$$\boldsymbol{r}^{(0)} = \bar{\boldsymbol{g}} - \boldsymbol{AQ}^{(0)},\ \boldsymbol{c}^{(0)} = \boldsymbol{r}^{(0)}.$$

2) Sequentially calculate from top to bottom:

$$\begin{aligned}
\alpha^{(m)} &= \frac{\left(\boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)}\right)}{\left(\boldsymbol{c}^{(m)}, \boldsymbol{Ac}^{(m)}\right)}\\
\boldsymbol{Q}^{(m+1)} &= \boldsymbol{Q}^{(m)} + \alpha^{(m)}\boldsymbol{c}^{(m)}\\
\boldsymbol{r}^{(m+1)} &= \boldsymbol{r}^{(m)} - \alpha^{(m)}\boldsymbol{Ac}^{(m)}\\
\beta^{(m)} &= \frac{\left(\boldsymbol{r}^{(m+1)}, \boldsymbol{r}^{(m+1)}\right)}{\left(\boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)}\right)}\\
\boldsymbol{c}^{(m+1)} &= \boldsymbol{r}^{(m+1)} + \beta^{(m)}\boldsymbol{c}^{(m)}
\end{aligned}$$

3) End the iteration if terminal conditions are satisfied. Otherwise, substitute $m+1$ for m and return to step 2).

2) BCG Algorithm:

1) Let $m = 0$ and prepare the initial vector $\boldsymbol{Q}^{(0)}$,

$$\boldsymbol{r}^{(0)} = \bar{\boldsymbol{g}} - \boldsymbol{AQ}^{(0)},\ \boldsymbol{c}^{(0)} = \boldsymbol{r}^{(0)},\ \tilde{\boldsymbol{r}}^{(0)} = \tilde{\boldsymbol{c}}^{(0)} = \boldsymbol{r}^{(0)}.$$

2) Sequentially calculate from top to bottom:

$$\begin{aligned}
\alpha^{(m)} &= \frac{\left(\tilde{\boldsymbol{r}}^{(m)}, \boldsymbol{r}^{(m)}\right)}{\left(\tilde{\boldsymbol{c}}^{(m)}, \boldsymbol{Ac}^{(m)}\right)}\\
\boldsymbol{Q}^{(m+1)} &= \boldsymbol{Q}^{(m)} + \alpha^{(m)}\boldsymbol{c}^{(m)}\\
\boldsymbol{r}^{(m+1)} &= \boldsymbol{r}^{(m)} - \alpha^{(m)}\boldsymbol{Ac}^{(m)}\\
\tilde{\boldsymbol{r}}^{(m+1)} &= \tilde{\boldsymbol{r}}^{(m)} - \alpha^{(m)}\boldsymbol{A}^T\tilde{\boldsymbol{c}}^{(m)}\\
\beta^{(m)} &= \frac{\left(\tilde{\boldsymbol{r}}^{(m+1)}, \boldsymbol{r}^{(m+1)}\right)}{\left(\tilde{\boldsymbol{r}}^{(m)}, \boldsymbol{r}^{(m)}\right)}\\
\boldsymbol{c}^{(m+1)} &= \boldsymbol{r}^{(m+1)} + \beta^{(m)}\boldsymbol{c}^{(m)}\\
\tilde{\boldsymbol{c}}^{(m+1)} &= \tilde{\boldsymbol{r}}^{(m+1)} + \beta^{(m)}\tilde{\boldsymbol{c}}^{(m)}
\end{aligned}$$

3) End the iteration if terminal conditions are satisfied. Otherwise, substitute $m+1$ for m and return to step 2).

3) BPKSM Algorithm:

1) Let $m = 0$ and prepare the initial vector $\boldsymbol{Q}^{(0)}$,

$$\begin{aligned}
&\boldsymbol{r}^{(0)} = \bar{\boldsymbol{g}} - \boldsymbol{AQ}^{(0)},\ \boldsymbol{c}^{(0)} = \boldsymbol{r}^{(0)},\ \tilde{\boldsymbol{r}}^{(0)} = \tilde{\boldsymbol{c}}^{(0)} = \boldsymbol{r}^{(0)},\\
&\boldsymbol{t}^{(-1)} = \boldsymbol{u}^{(-1)} = \boldsymbol{w}^{(-1)} = \boldsymbol{z}^{(-1)} = \boldsymbol{0},\ \beta^{(-1)} = 0.
\end{aligned}$$

2) Sequentially calculate from top to bottom:

$$\begin{aligned}
\alpha^{(m)} &= \frac{\left(\tilde{\boldsymbol{r}}^{(0)}, \boldsymbol{r}^{(m)}\right)}{\left(\tilde{\boldsymbol{r}}^{(0)}, \boldsymbol{Ac}^{(m)}\right)}\\
\boldsymbol{y}^{(m)} &= \boldsymbol{t}^{(m-1)} - \boldsymbol{r}^{(m)} - \alpha^{(m)}\left(\boldsymbol{w}^{(m-1)} - \boldsymbol{Ac}^{(m)}\right)\\
\boldsymbol{t}^{(m)} &= \boldsymbol{r}^{(m)} - \alpha^{(m)}\boldsymbol{Ac}^{(m)}\\
&\text{Parameters } \eta^{(m)} \text{ and } \zeta^{(m)} \text{ are calculated.}\\
\boldsymbol{u}^{(m)} &= \zeta^{(m)}\boldsymbol{Ac}^{(m)}\\
&\quad + \eta^{(m)}\left(\boldsymbol{t}^{(m-1)} - \boldsymbol{r}^{(m)} + \beta^{(m-1)}\boldsymbol{u}^{(m-1)}\right)\\
\boldsymbol{z}^{(m)} &= \zeta^{(m)}\boldsymbol{r}^{(m)} + \eta^{(m)}\boldsymbol{z}^{(m-1)} - \alpha^{(m)}\boldsymbol{u}^{(m)}\\
\boldsymbol{Q}^{(m+1)} &= \boldsymbol{Q}^{(m)} + \alpha^{(m)}\boldsymbol{c}^{(m)} + \boldsymbol{z}^{(m)}\\
\boldsymbol{r}^{(m+1)} &= \boldsymbol{t}^{(m)} - \eta^{(m)}\boldsymbol{y}^{(m)} - \zeta^{(m)}\boldsymbol{At}^{(m)}\\
\beta^{(m)} &= \frac{\alpha^{(m)}\left(\tilde{\boldsymbol{r}}^{(0)}, \boldsymbol{r}^{(m+1)}\right)}{\zeta^{(m)}\left(\tilde{\boldsymbol{r}}^{(0)}, \boldsymbol{r}^{(m)}\right)}\\
\boldsymbol{w}^{(m)} &= \boldsymbol{At}^{(m)} + \beta^{(m)}\boldsymbol{Ac}^{(m)}\\
\boldsymbol{c}^{(m+1)} &= \boldsymbol{r}^{(m+1)} + \beta^{(m)}\left(\boldsymbol{c}^{(m)} - \boldsymbol{u}^{(m)}\right)
\end{aligned}$$

3) End the iteration if terminal conditions are satisfied. Otherwise, substitute $m+1$ for m and return to step 2).

There are various methods to calculate $\eta^{(m)}$ and $\zeta^{(m)}$. We chose two ways, CGS and BiCGSTAB. The CGS algorithm uses the following parameters

$$\eta^{(m)} = \frac{\alpha^{(m)}\beta^{(m-1)}}{\alpha^{(m-1)}},\ \zeta^{(m)} = \alpha^{(m)}.$$

The BiCGSTAB algorithm uses the following parameters

$$\eta^{(m)} = 0,\ \zeta^{(m)} = \frac{\left(\boldsymbol{t}^{(m)}, \boldsymbol{At}^{(m)}\right)}{\left(\boldsymbol{At}^{(m)}, \boldsymbol{At}^{(m)}\right)}.$$

V. Theoretical Efficiency of Policy Evaluation

This section provides a theoretical discussion to explain the reasons for policy evaluation efficiency.

A. Policy Evaluation and Algebraic Equation

Defining $\boldsymbol{P}$ by $\boldsymbol{p}$, $\boldsymbol{\Pi}$ by μ, and $\bar{\boldsymbol{g}}$ by $\boldsymbol{p}$ and $\boldsymbol{g}$ as in the Appendix, Eq. (1) are formulated as

$$\boldsymbol{Q}^{\mu} = \bar{\boldsymbol{g}} + \boldsymbol{P}\boldsymbol{\Pi}\boldsymbol{Q}^{\mu}. \tag{7}$$

Eq. (7) is regarded as a large-scale algebraic equation for unknown $\boldsymbol{Q}^{\mu}$, which is solved as

$$\boldsymbol{Q}^{\mu} = (\boldsymbol{I} - \boldsymbol{P}\boldsymbol{\Pi})^{-1}\bar{\boldsymbol{g}} = \boldsymbol{A}^{-1}\bar{\boldsymbol{g}}. \tag{8}$$

If policy evaluation is regarded as an algebraic problem in Eq. (8), algorithms for the problem can be classified into direct methods and iterative methods. A direct method's total number of calculation operations is finite, but it cannot obtain the Q-factor for policy evaluation until its calculations completed. On the other hand, an iterative method can terminate its iteration since it exponentially decreases the error. Typical policy evaluation is not completed accurately or fully. An iterative method algorithm is applied to policy evaluation because it can generally terminate with less iteration. SIM and KSM are both iterative method algorithms, but KSM has a property common to direct methods. The number of calculation operations is finite. SIM requires infinite iterations in order to solve the problem accurately without taking computational error into consideration. But, KSM's iteration number is finite. The size of the problem determines the iteration number, which is estimated to be as many as the direct method, e.g. LU factorization.

B. Features of SIM and Efficiency Evaluation Index

A general iterative form of Eq. (7) denoting the policy evaluation is

$$\boldsymbol{Q}^{(m+1)} = \boldsymbol{b} + \boldsymbol{C}\boldsymbol{Q}^{(m)}, \tag{9}$$

where $\boldsymbol{C}$ is a constant iteration matrix. Now $\boldsymbol{A}$ can be uniquely factored to

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}, \tag{10}$$

where $\boldsymbol{D}$ is a diagonal matrix with non-zero diagonal elements, $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular matrices with zero diagonal elements. The iteration matrices of SIM algorithms in Section IV-A are [3]

$$\begin{aligned}
&\boldsymbol{C}_{PEI} = \boldsymbol{P}\boldsymbol{\Pi},\ \boldsymbol{C}_{Jacobi} = -\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U}),\\
&\boldsymbol{C}_{Gauss-Seidel} = -(\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{U},\\
&\boldsymbol{C}_{SOR} = -(\boldsymbol{D}+\omega\boldsymbol{L})^{-1}\left\{(1-\omega)\boldsymbol{D} - \omega\boldsymbol{U}\right\}.
\end{aligned}$$

The error $\Delta\boldsymbol{Q}^{(m)} \equiv \boldsymbol{Q}^{(m)} - \boldsymbol{Q}^{\mu}$ rearranges Eq. (9) as

$$\Delta\boldsymbol{Q}^{(m)} = \boldsymbol{C}^{m}\Delta\boldsymbol{Q}^{(0)}. \tag{11}$$

Therefore, $\Delta\boldsymbol{Q}^{(m)}$ and $\boldsymbol{Q}^{(m)}$ converge if $\rho(\boldsymbol{C}) < 1$, where $\rho(\boldsymbol{C})$ is the spectral radius (the maximum absolute eigenvalue) of $\boldsymbol{C}$. They converge exponentially because $\|\Delta\boldsymbol{Q}^{(m)}\|_2 \propto \rho(\boldsymbol{C})^m$. Hence, $\rho(\boldsymbol{C})$ can be an index of the convergence rate. Here $\|\cdot\|_p$ is the p-norm, and the 2-norm is a Euclidean norm.

C. Features of KSM and Efficiency Evaluation Index

The KSM algorithms in Section IV-B have properties of both iterative methods and direct methods that can obtain an accurate solution of the policy evaluation step within $n = NK$ iterations unless a breakdown occurs, i.e. a denominator becomes zero. CG guarantees convergence on the correct solution if $\boldsymbol{A}$ is a positive-definite symmetric matrix. BCG and BPKSM guarantee convergence, even if $\boldsymbol{A}$ is not symmetric. When $\boldsymbol{A}$ is not symmetric in a general RL problem, BCG applies CG iteration to the following coefficient matrix

$$\tilde{\boldsymbol{F}}\tilde{\boldsymbol{A}} \equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}. \tag{12}$$

BPKSM is regarded as a modification of BCG. Hence, their behavior is similar to CG [10].

The CG iteration monotonically decreases the evaluation value $J(\boldsymbol{Q}^{(m)}) \equiv \frac{1}{2}(\boldsymbol{Q}^{(m)}, \boldsymbol{A}\boldsymbol{Q}^{(m)}) - (\bar{\boldsymbol{g}}, \boldsymbol{Q}^{(m)})$ for a positive-definite symmetric $\boldsymbol{A}$ with size NK. Therefore, the error norm weighted by $\boldsymbol{A}$, i.e. $\|\Delta\boldsymbol{Q}^{(m)}\|_{\boldsymbol{A}} \equiv \sqrt{(\Delta\boldsymbol{Q}^{(m)}, \boldsymbol{A}\Delta\boldsymbol{Q}^{(m)})} = \sqrt{2J(\boldsymbol{Q}^{(m)})}$, also decreases monotonically. Hence, $\|\Delta\boldsymbol{Q}^{(m)}\|_{\boldsymbol{A}}$ is used to evaluate the efficiency of KSM algorithm. Moreover, λ_i and $\boldsymbol{v}_i$ denote the eigenvalue and corresponding eigenvector of $\boldsymbol{A}$, where $0 < \lambda_1 \leq \ldots \leq \lambda_n$. This defines transformation matrix $\boldsymbol{U}$ as

$$\begin{aligned}
&\boldsymbol{U} \equiv \sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^T,\\
&\sqrt{\boldsymbol{\Lambda}} \equiv \mathrm{diag}\left[\ \sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\ \right],\ \boldsymbol{V} \equiv [\ \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\ ].
\end{aligned}$$

By introducing the transformation $\hat{\boldsymbol{Q}}^{(m)} = \boldsymbol{U}\boldsymbol{Q}^{(m)}$ and $\hat{\boldsymbol{Q}}^{\mu} = \boldsymbol{U}\boldsymbol{Q}^{\mu}$, we have the formula $\|\Delta\boldsymbol{Q}^{(m)}\|_{\boldsymbol{A}} = \|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\|_2$. Hence, the weighted norm becomes the normal Euclidean norm of transformed variables $\hat{\boldsymbol{Q}}^{(m)}$ and $\hat{\boldsymbol{Q}}^{\mu}$.

This paper applies this transformation and considers a convergence rate corresponding to the spectral radius of SIM along with the convergence theorem of the CG algorithm [11].

$$\begin{aligned}
&\left\|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\right\| \leq 2\nu^m \left\|\hat{\boldsymbol{Q}}^{(0)} - \hat{\boldsymbol{Q}}^{\mu}\right\| \\
&\nu \equiv \left(\sqrt{\lambda_n} - \sqrt{\lambda_1}\right) \Big/ \left(\sqrt{\lambda_n} + \sqrt{\lambda_1}\right)
\end{aligned} \tag{13}$$

The ν in Eq. (13) gives the worst convergence rate, i.e. the upper limit, where m is the iteration number.

The residual vectors of BCG denoted by $\boldsymbol{r}^{(m)}_{BCG}$ satisfies

$$\boldsymbol{r}^{(m)}_{BCG} = P^{(m)}(\boldsymbol{A})\boldsymbol{r}^{(0)}_{BCG}, \tag{14}$$

where $P^{(m)}(\lambda)$ is the m-th order residual polynomial [10]. In the meantime, the residual vectors of CGS denoted by $\boldsymbol{r}^{(m)}_{CGS}$ satisfies

$$\boldsymbol{r}^{(m)}_{CGS} = P^{(m)}(\boldsymbol{A})P^{(m)}(\boldsymbol{A})\boldsymbol{r}^{(0)}_{CGS}, \tag{15}$$

where $P^{(m)}(\lambda)$ is the same polynomial. Therefore, the residual vectors of BCG and CGS are

$$\boldsymbol{r}^{(m)}_{BCG} = P^{(m)}(\boldsymbol{A})\boldsymbol{r}^{(0)}, \tag{16}$$

$$\boldsymbol{r}^{(m)}_{CGS} = P^{(m)}(\boldsymbol{A})P^{(m)}(\boldsymbol{A})\boldsymbol{r}^{(0)}, \tag{17}$$

for the same $\boldsymbol{Q}^{(0)}$, i.e. $\boldsymbol{r}^{(0)} = \boldsymbol{r}^{(0)}_{BCG} = \boldsymbol{r}^{(0)}_{CGS}$. Hence, the CGS algorithm is expected to converge twice as fast as the BCG

TABLE I
CULCULATION COST FOR UPDATING VECTOR $\boldsymbol{Q}^{(m)}$

Algorithm	Nunber of multiplication / division operations
PEI	N^2K^2
Jacobi	$N^2K^2 + NK$
Gauss-Seidel	$N^2K^2 + NK$
SOR	$N^2K^2 + 3NK$
BCG	$2N^2K^2 + 7NK + 2$
CGS	$2N^2K^2 + 12NK + 5$
BiCGSTAB	$2N^2K^2 + 14NK + 4$

algorithm, which will be confirmed in a numerical example. We also find that BCG and CGS intrinsically the same.

D. Computation Cost Estimation of KSM from Learning Problem Characteristics

According to the regularity of $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P\Pi}$ and Gerschgorin's theorem [12], all eigenvalues of $\boldsymbol{A}$ are crowded together in the following interval.

$$\lambda_i \in \left(0,\, 2(1 - \min_{(j,k)} \pi(s_j, u_k) p_{jj}(u_k))\right],\ ^{\forall} i \tag{18}$$

As shown in a numerical example, $\boldsymbol{A}$ has many degenerated eigenvalues of 1. Its degeneracy is calculated as follows. We find that $\mu_i = 1 - \lambda_i,\ ^{\forall} i$ for any eigenvalue μ_i of $\boldsymbol{P\Pi} \in \boldsymbol{R}^{NK \times NK}$. Hence, the degeneracy degree of zero-eigenvalues of $\boldsymbol{P\Pi}$, denoted by m_0, are equal to that of eigenvalues 1 of $\boldsymbol{A}$. On the other hand, the rank of $\boldsymbol{P\Pi}$ is no more than N, since $\boldsymbol{P}$ and $\boldsymbol{\Pi}$ are $(NK) \times N$ and $N \times (NK)$ matrices. The rank cancellation $NK - N$ is degeneracy of zero-eigenvalues of $\boldsymbol{P\Pi}$, and $m_0 \geq NK - N$.

As mentioned before, a rigorous solution is obtained at most NK iterations by KSM algorithm. But, KSM algorithm can stop a policy evaluation step with $n - m_0 + 1 \leq N + 1$ iterations since the iteration number is decreased as many times as the degeneracy degree of the eigenvalues of $\boldsymbol{A}$ [6]. This is also decreased more if there are degenerated eigenvalues other than 1. Hence KSM algorithm guarantees to get the optimal policy with finite iterations far fewer than $\boldsymbol{A}$.

Table I lists the number of multiplications and division operations required in each algorithm to update $\boldsymbol{Q}^{(m)}$ once for $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P\Pi}$. The computation cost of the CG algorithm is the same as SIM, and BCG, and BPKSM requires twice as many computations as SIM. However, the total cost of the proposed methods is smaller than SIMs, as shown in numerical examples. When KSM algorithms converge at most $N + 1$ iterations, as mentioned before, total computation cost required in a rigid evaluation is about N^3K^2 or $2N^3K^2$.

VI. NUMERICAL EXAMPLES

A. Convergence Properties in A General Numerical Problem

This section examines the convergence performance of SIM and KSM algorithms in a general numerical problem. The problem is to solve the simultaneous equations $\boldsymbol{AQ}^{\mu} = \bar{\boldsymbol{g}}$ for $\boldsymbol{Q}^{\mu}$ with a given positive definite symmetric matrix $\boldsymbol{A}$ and a vector $\bar{\boldsymbol{g}}$, where their size is $n = 200$. We compare the Jacobi, the Gauss-Seidel, SOR (where the relaxation factor is $\omega = 1.5$) as SIM algorithms, and BCG, CGS, BiCGSTAB as KSM algorithms.

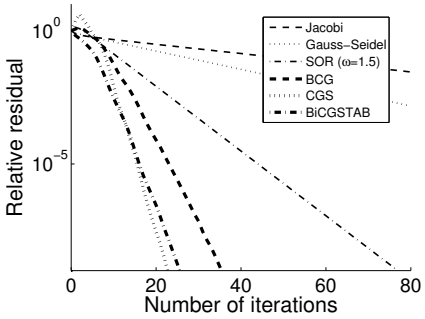


Fig. 1. Convergence of solution $\boldsymbol{AQ}^{\mu} = \bar{\boldsymbol{g}}$, where $\boldsymbol{A}$ is a general matrix.

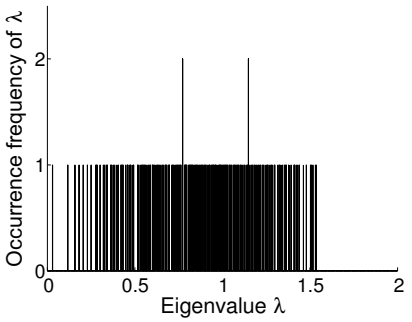


Fig. 2. Eigenvalue distribution of matrix $\boldsymbol{A}$.

Figure 1 shows their relative residuals $\|\bar{\boldsymbol{g}} - \boldsymbol{AQ}^{(m)}\|_2 / \|\bar{\boldsymbol{g}}\|_2$ starting from $\boldsymbol{Q}^{(0)} = \boldsymbol{0}$. It is found that all KSM algorithms converge two to three times faster than SOR. However, considering Table I, there is almost no difference in performance among the algorithms. This result has been reconfirmed by their computation periods using a computer. Here $\boldsymbol{A}$ is chosen so its eigenvalues exist in the same interval as Eq. (18) in order to compare the following learning problem. The $\boldsymbol{A}$ is very sparse and has little degeneracy, as shown in Figure 2. Even if other matrices similar to $\boldsymbol{A}$ are used, their results are similar.

B. Rapid Convergence Rate of KSM in A RL Problem

1) Setting and Features of a RL Problem: This section considers a mass control problem shown in reference [3]. The objective as the optimal regulator problem is to establish control input u_t that minimizes cost function

$$J = \sum_{t=0}^{\infty} \frac{1}{2} \left(x_t^2 + \dot{x}_t^2 + u_t^2\right).$$

The state of the mass is updated as follows. At time t, the position of mass x_t is updated by Eq. (19a) with a probability of 0.9 or Eq. (19b) with a probability of 0.1:

$$\begin{cases} x_{t+1} = x_t + \dot{x}_t & \text{(19a)} \\ x_{t+1} = x_t + \dot{x}_t - \text{sign}\left(\dot{x}_t\right), & \text{(19b)} \end{cases}$$

TABLE II
SPECTRAL RADIUS $\rho(\boldsymbol{C})$

Algorithm	Spectral radius
PEI	0.997439
Jacobi	0.997380
Gauss-Seidel	0.995058
SOR ($\omega = 1.5$)	0.986642

where sign( ) denotes sign function. The velocity of mass $\dot{x}_t$ is updated by Eq. (20a) with a probability of 0.9 or Eq. (20b) with a probability of 0.1.

$$\begin{cases} \dot{x}_{t+1} = \dot{x}_t + u_t & \text{(20a)} \\ \dot{x}_{t+1} = \dot{x}_t + u_t - \text{sign}\,(u_t) & \text{(20b)} \end{cases}$$

The $\boldsymbol{P}$ is defined by the above probabilities.

We quantize the interval $[-2, 12]$ of x_t into 15, the interval $[-5, 5]$ of $\dot{x}_t$ into 11, and the interval $[-5, 4]$ of u_t into 10. Hence, the state number, except the termination state, is $N = 164$, and the action number is $K = 10$, which results in the size of $\boldsymbol{A}$, $NK = 1640$.

The eigenvalue distribution of $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P}\boldsymbol{\Pi}$ is shown in Figure 3, where $\boldsymbol{\Pi}$ is given as a random policy, i.e. $\pi(s_i, u_k) = 1/K \ ^\forall(s_i, u_k)$. All eigenvalues of $\boldsymbol{A}$ exist in the interval $(0, 2]$ because of Eq. (18). Simultaneously, it is found that the degeneracy degree of eigenvalue 1 is more than $N(K-1) = 1476$.

2) Policy Evaluation: This section compares the policy evaluation steps of the algorithms by convergence of the Q-factor, where the random policy is evaluated. We evaluate each algorithm with a relative residual versus the iteration number. The initial Q-factor for the iteration is $\boldsymbol{Q}^{(0)} = \boldsymbol{0}$.

Figure 4 shows the relative residuals of SIM and KSM algorithms, where $\omega = 1.5$ for SOR. Figure 5 shows those of KSM only. KSM algorithms converge much more rapidly than SIM algorithms, which is different from Figure 1. Among SIM algorithms, the asynchronous iteration methods, Gauss-Seidel and especially SOR are efficient. However, thousands of iterations are required to satisfy the convergence condition. The relative residual becomes smaller than 10^{-6}. Each spectral radius is listed in Table II. KSM algorithms converge within a few tens of iteration numbers if the same convergence condition as above is used, and the iteration numbers of SIM algorithms are tens to hundreds of times more than KSM algorithms. It is concluded that KSM is much more efficient than SIM, considering the computation cost per iteration of KSM is at most twice as many as SIM. In addition, we find that KSM algorithms tend to accelerate the convergence ratios, whereas SIM algorithms converge at constant rates.

3) Policy Iteration: This section evaluates the total computation cost of the policy iteration for the same mass control problem. Starting from a random policy μ_0 and generating policy sequences μ_l ($l = 0, 1, \ldots$), we show the total computation cost of all iterations until optimal policy μ^* is obtained. Here we compare two initializing methods. The first method initializes the Q-factor $\boldsymbol{Q}^{(0)}$ as $\boldsymbol{0}$ at each policy evaluation step. The second method initializes the Q-factor as $\boldsymbol{Q}^{\mu_{l-1}}$ of the policy μ_{l-1} of the previous policy evaluation step. Each policy iteration step ends its iteration if both conditions are satisfied: the relative residual of the Q-factor is smaller than 10^{-3}, and the average update amount of all Q-factor elements $\|\boldsymbol{Q}^{(m)} - \boldsymbol{Q}^{(m-1)}\|_1/NK$ is smaller than 10^{-4}.

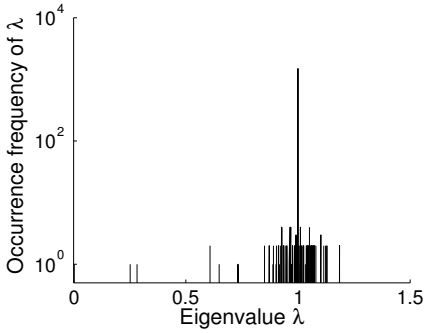


Fig. 3. Eigenvalue distribution of $\boldsymbol{I} - \boldsymbol{P}\boldsymbol{\Pi}$.

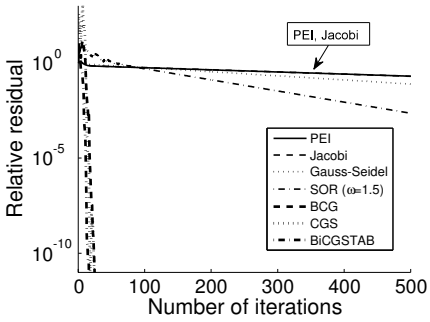


Fig. 4. Convergence of SIM and KSM.

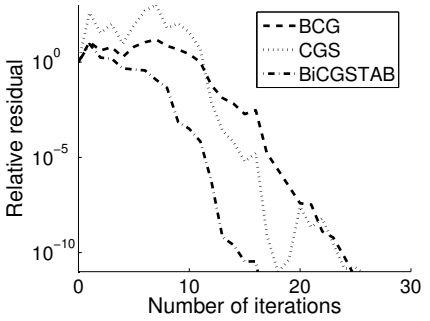


Fig. 5. Convergence of KSM.

Table III and Table IV list each iteration number and computation period. The number within parentheses denotes the number of states where their actions agree with those of the optimal policy, and the total number of system states is 164. All algorithms acquire the optimal policy via the same policy sequence with 4 policy improvement steps. We find that policy evaluation based on KSM is tens to hundreds of times more rapid than SIM. Hence, the KSM algorithm's efficiency is confirmed by the entire RL procedure to obtain the optimal policy.

TABLE III
MASS CONTROL PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY POLICY EVALUATION ALGORITHMS. Q-FACTOR IS INITIALIZED AS **0**. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
μ_0 (0)	4900	4800	2670	1060	20	18	14
μ_1 (83)	20	20	20	540	12	9	9
μ_2 (133)	30	20	20	260	11	9	9
μ_3 (163)	30	20	20	270	11	9	9
μ_4 (164)	30	20	20	270	11	9	9
Total	5010	4880	2750	2400	65	54	50
Time (s)	74.2	68.8	39.0	34.1	3.1	2.3	2.2

TABLE IV
MASS CONTROL PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY POLICY EVALUATION ALGORITHMS. Q-FACTOR IS INITIALIZED AS $\boldsymbol{Q}^{\mu_{l-1}}$. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
μ_0 (0)	4900	4800	2670	1060	20	18	14
μ_1 (83)	40	30	20	870	14	11	11
μ_2 (133)	20	20	10	220	10	7	6
μ_3 (163)	10	10	10	30	6	4	4
μ_4 (164)	10	10	10	20	3	2	2
Total	4980	4870	2720	2200	53	42	37
Time (s)	72.6	68.5	38.7	31.5	2.6	1.9	1.7

C. Acceleration of Convergence Rate of KSM

This section solves the simultaneous equations $\boldsymbol{A}\boldsymbol{Q}^{\mu} = \bar{\boldsymbol{g}}$ in Section VI-A for $\boldsymbol{Q}^{\mu}$ by CG in order to observe the convergence of KSM. The error norm $\|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\|_2$ plotted in Figure 6 decreases monotonically. For $i = 1, \ldots, 10$, the absolute value of the i-th element of $\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}$ denoted by $\hat{d}_i^{\,(m)}$ is plotted in Figure 7. The solid line in Figure 6 represents above error norm and the broken line represents its upper bound based on ν of Eq. (13). It is found that the error norm is truly decreasing and its convergence rate becomes better.

Here $\hat{d}_i^{\,(m)}$ and $\|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\|$ satisfy below [11]:

$$\left\|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\right\|^2 = \min_{R \in \mathcal{P}_m,\, R(0)=1} \sum_{i=1}^{n} \left|\hat{d}_i^{\,(0)} R(\lambda_i)\right|^2 \quad (21)$$

$$R^{(m)}(\lambda) \equiv \arg \min_{R \in \mathcal{P}_m,\, R(0)=1} \sum_{i=1}^{n} \left|\hat{d}_i^{\,(0)} R(\lambda_i)\right|^2, \quad (22)$$

$$\hat{d}_i^{\,(m)} = \hat{d}_i^{\,(0)} R^{(m)}(\lambda_i), \ {}^{\forall} i, \quad (23)$$

where $\mathcal{P}_m$ denotes a set of real polynomials where their orders are no more than m and $R(\lambda) \in \mathcal{P}_m$ satisfies $R(0) = 1$. From Eq. (23), the decreasing rate to the initial error element, $\hat{d}_i^{\,(m)}/\hat{d}_i^{\,(0)}$, is equal to $R^{(m)}(\lambda_i)$. And $R^{(m)}(\lambda)$ is determined uniquely so that it minimizes the error norm, as shown in Eq. (22). As the iteration number m increases, the order of $R^{(m)}(\lambda)$ increases and the error norm decreases. Due to the constraint $R^{(m)}(0) = 1$, the decreasing rate $\hat{d}_i^{\,(m)}/\hat{d}_i^{\,(0)} = R^{(m)}(\lambda_i)$ of eigenvalue λ_i rapidly approaches zero as m increases, which is observed in Figure 7.

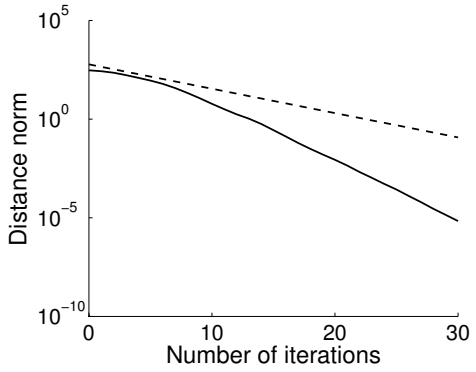


Fig. 6. Distance norm $\|\hat{\boldsymbol{Q}}^{(m)} - \hat{\boldsymbol{Q}}^{\mu}\|_2$.

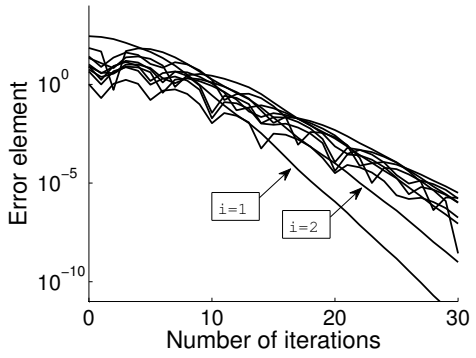


Fig. 7. Error element $|\hat{d}_i^{\,(m)}|$.

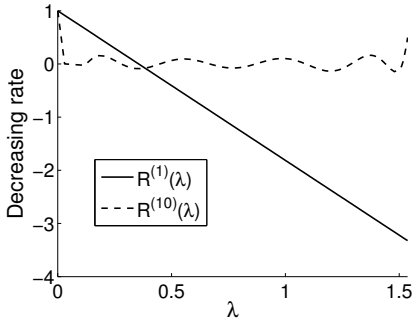


Fig. 8. $R^{(m)}(\lambda)$ for $m = 1, 10$.

Figure 8 shows the m-th order polynomials $R^{(m)}(\lambda)$ for $m = 1, 10$. We find that almost all error elements $\hat{d}_i^{\,(m)}$ decrease since $R^{(m)}(\lambda_i)$ is almost zero for all λ_i as m becomes 10.

It is said that KSM is suited for sparse matrices [11] and is considered to be very efficient because coefficient matrix $\boldsymbol{A}$ of the RL problem is also sparse. However, sparsity is not the only reason for this result, as shown in Section VI-A. There are two properties which do not necessarily result from sparseness: (i) the RL's property of eigenvalue degeneracy of the coefficient matrix and (ii) the KSM algorithm's property of improving convergence of the modes whose convergence rates are small and make learning slow. It is also shown that property (i) does not require any pre-conditioning to decrease

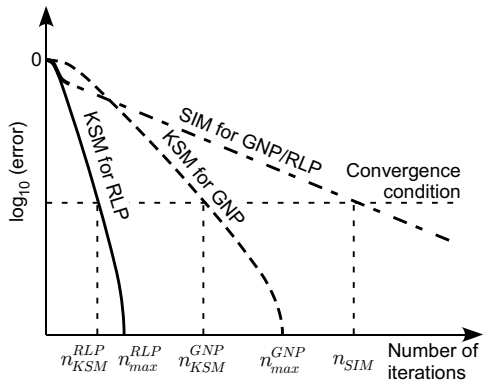


Fig. 9. Convergence characteristics of SIM and KSM for GNP and RLP (GNP: general numerical problem, RLP: reinforcement learning problem).

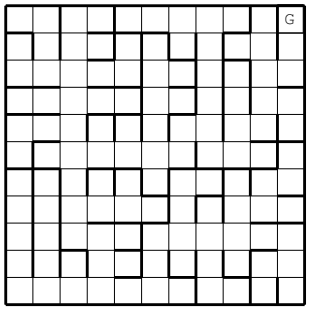


Fig. 10. Maze problem.

the round-off error and improve convergence. As a result, it has been clarified that the algorithms based on KSM are efficient for policy evaluation.

D. Discussion on Rapid Convergence of KSM in RL Problems

The reason why the KSM type algorithms converge rapidly in RL problems is based on the theoretical discussion in section V and the above numerical discussion. Fig. 9 is a semi-logarithmic graph. The horizontal axis is the number of iterations, and the vertical axis is the error. The error is considered the relative residual, $\|\bar{\boldsymbol{g}} - \boldsymbol{A}\boldsymbol{Q}^{(m)}\|_2/\|\bar{\boldsymbol{g}}\|_2$, or the normalized error norm weighted by $\boldsymbol{A}$, $\|\Delta\boldsymbol{Q}^{(m)}\|_{\boldsymbol{A}}/\|\Delta\boldsymbol{Q}^{(0)}\|_{\boldsymbol{A}}$. Without taking computational error into consideration, schematic graphs are plotted when a general numerical problem and a RL problem are solved by SIM and KSM type algorithms.

Consider the SIM type algorithm first. The final convergence rate of SIM is determined by the spectral radius $\rho(\boldsymbol{C})$, as shown in sections V-B and VI-A. The error converges on zero along a straight line, a constant gradient. Figs. 1 and 4 show the situation. Hence, SIM requires infinite iterations in order to solve the problem. For simplicity, we assume SIM convergence in a general numerical problem is the same as that in a RL problem.

Consider the KSM type algorithm. The number of iterations to solve the problem by KSM is finite, as mentioned in sections V-B and V-D. The iteration number is at most $n_{max} = n - \sum_i (n_i - 1)$ as explained in section V-D, where $n = NK$ is the size of $\boldsymbol{A}$, and n_i is the degeneracy degree of eigenvalue λ_i of $\boldsymbol{A}$. The maximum iteration number becomes $n_{max}^{GNP} \simeq NK$ if almost no eigenvalue degenerates when KSM solves the general numerical problem. On the other hand, the maximum iteration number becomes $n_{max}^{RLP} \leq N + 1$ because the degeneracy degree of eigenvalue 1 is at least $N(K-1)$ when KSM solves the RL problem. At the same time, the error norm weighted by $\boldsymbol{A}$ decreases monotonously as mentioned in section V-C. In addition, the convergence rate of KSM becomes smaller and smaller as explained in sections VI-B and VI-C. The error is zero at iteration number n_{max}. Therefore, the convergence rate finally becomes zero.

As a result, graphs become like Fig. 9 when the general numerical problem and the RL problem are solved by SIM and KSM.

For any numerical problem, we regard the solution as converged when the error becomes a specific value. As shown in the figure, n_{SIM} is the convergence iteration number of SIM, n_{KSM}^{GNP} is that of KSM for the general numerical problem, and n_{KSM}^{RLP} is that of KSM for the RL problem. In section VI-A, $n_{KSM}^{GNP} \simeq n_{SIM}/2$ held when KSM and SIM solved the general numerical problem. The calculation cost for an iteration of KSM was twice as much as that of SIM. Hence, the calculation cost for the convergence by KSM was almost the same as SIM in the case of the general numerical problem. On the other hand, $n_{KSM}^{RLP} \ll n_{SIM}/2$ held when KSM and SIM solved the RL problem in section VI-B. Therefore, the calculation cost for the convergence by KSM was less than SIM in the case of the RL problem.

KSM is more efficient than SIM for RL problems, whereas the degree of efficiency depends on the problem. This will be shown by additional numerical examples in the next section.

E. Additional Reinforcement Learning Problems

To show the comprehensive effectiveness of the proposed method, two popular examples, a maze problem and an inverted pendulum problem, are solved using SIM and KSM algorithms. Their learning efficiencies are compared.

1) Maze Problem: Figure 10 shows an 11×11 maze. A tile placed within the maze is an agent. The agent's location is a state. The top right tile G is the termination state. The agent chooses an action among 4 actions: up, down, right, left. By choosing any action, the agent stays at the present state with a probability 0.1. By choosing an action in the direction of a wall, the agent stays at the present state with probability 1. The agent incurs a one-step cost 1 for each action, except from the termination state. Hence, the state number, except the termination state, is $N = 120$, and the action number is $K = 4$, which results in the size of $\boldsymbol{A}$, $NK = 480$.

Starting from a random policy, the optimal policy is obtained by the policy iteration. The relaxation factor for SOR is $\omega = 1.5$. Each policy iteration step ends its iteration if the same convergence conditions as the mass control problem are satisfied.

According to this condition, the optimal policy is obtained by policy improvement first, just after the policy evaluation of random policy. Hence, Table V lists each iteration number and computation period for evaluation of the random policy.

TABLE V
MAZE PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY POLICY EVALUATION ALGORITHMS. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
Total	67400	59100	32400	12500	102	189	87
Time (s)	288.9	251.7	138.4	53.3	1.2	2.0	1.0

Policy evaluation based on KSM is tens to hundreds of times more rapid than SIM.

2) Inverted Pendulum Problem: This section considers an inverted pendulum problem, which is made referring problems in reference [13]. The object of this exercise is to move a simple pendulum to the top and stop at minimum cost. Torque is applied to the pendulum axis, clockwise or counterclockwise rotation. The state $(x,\, v)$ is composed of the rotational position x and rotational velocity v of the pendulum. The action u is the applied torque. The top position is $x = \pi$. The inverted pendulum is modeled using the following state equations:

$$\dot{x} = v, \tag{24}$$

$$\dot{v} = \frac{1}{m\ell^2}\left(u - cv - mg\ell \sin x\right), \tag{25}$$

where $m = 2.0$ is the mass at the tip of pendulum, $g = 9.8$ is the gravitational acceleration, $\ell = 0.5$ is the pendulum length, and $c = 0.01$ is the viscous coefficient of rotation. In accordance with Eqs. (24) and (25), the pendulum in state $s_t = (x_t,\, v_t)$ chooses action u_t at time t and transits to state $s_{t+1} = (x_{t+1},\, v_{t+1})$ after Δt.

This study considers a learning problem that discretizes the time and the state-action space. The time is divided by $\Delta t = 0.1$. We discretize the interval $[0,\ 2\pi]$ of x into 20, the interval $[-10.25,\ 10.25]$ of v into 41, and the interval $[-5,\ 5]$ of u into 5. The state transition probability $\boldsymbol{P}$ is obtained by calculating all transitions among discretized states. The quantized state including $(x,\ v) = (\pi,\ 0)$ is the termination state. It incurs a one-step cost 1 for each action except from the termination state. Hence, the state number, except the termination state, is $N = 819$, and the action number is $K = 5$, which results in the size of $\boldsymbol{A}$, $NK = 4095$.

Starting from a random policy μ_0 and generating policy sequences μ_l ($l = 0, 1, \ldots$), we show the total computation cost of all iterations until optimal policy μ^* is obtained. The relaxation factor for SOR is $\omega = 1.1$. We compare the two initializing methods introduced in the mass control problem. The first method initializes the Q-factor $\boldsymbol{Q}^{(0)}$ as $\mathbf{0}$ at each policy evaluation step. The second method initializes the Q-factor as $\boldsymbol{Q}^{\mu_{l-1}}$ of policy μ_{l-1} of the previous policy evaluation step. Each policy iteration step ends its iteration if the same convergence conditions as the mass control problem are satisfied.

Tables VI and VII list each iteration number and computation period. In the tables, $\bar{J}$ denotes the mean state value function of all states. Each $\bar{J}$ of the policy converges to within 6 digits. All algorithms achieve the optimal policy via the same policy sequence with 4 policy improvement steps. We find that policy evaluation based on KSM is several tens of times more rapid than SIM. Hence, KSM algorithm efficiency is confirmed.

TABLE VI
INVERTED PENDULUM PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY POLICY EVALUATION ALGORITHMS. Q-FACTOR IS INITIALIZED AS $\mathbf{0}$. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	$\bar{J}$	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
μ_0	116.702	8113	8088	4460	3750	48	49	42
μ_1	1.92651	92	84	52	45	28	22	20
μ_2	1.65661	73	57	37	32	30	21	21
μ_3	1.59852	71	62	41	36	31	20	20
μ_4	1.59850	69	64	41	36	26	19	19
Total		8418	8355	4631	3899	163	131	122
Time (s)		2139.9	2075.4	1143.2	936.1	63.0	36.7	34.8

TABLE VII
INVERTED PENDULUM PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY POLICY EVALUATION ALGORITHMS. Q-FACTOR IS INITIALIZED AS $\boldsymbol{Q}^{\mu_{l-1}}$. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	$\bar{J}$	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
μ_0	116.702	8113	8088	4460	3750	48	49	42
μ_1	1.92651	144	130	79	68	36	29	23
μ_2	1.65661	54	44	29	26	29	19	21
μ_3	1.59852	45	40	26	23	22	19	17
μ_4	1.59850	20	19	14	12	1	1	1
Total		8376	8321	4608	3879	136	117	104
Time (s)		2109.4	2055.7	1137.0	961.1	63.0	36.7	34.8

To investigate whether a similar result is obtained by a learning method other than the policy iteration, TD(λ), $\lambda = 0.7$ is applied. The number of policy improvement steps increase because the policy evaluation is more inaccurate as λ becomes smaller. Only the total iteration number and the total computation period are listed in Table VIII because there are more than 200 policy improvement steps. The λ minimizing the iteration number is dependent on the problem and the policy evaluation algorithm. In general, an efficient policy evaluation algorithm uses the minimum iteration number at $\lambda \simeq 1$ and an inefficient algorithm uses the minimum iteration number at a smaller λ. We find that learning by KSM is almost ten times faster than SIM, even though the efficient algorithms use more iteration numbers at $\lambda = 0.7$. Hence, the KSM algorithm efficiency is confirmed.

VII. CONCLUSION

This study has proposed algorithms based on KSM for effective policy evaluation and accelerated learning. Policy evaluation using KSM is a far more efficient calculation method than existing SIM algorithms. The reason is explained by RL problem features and their solution methods. Despite the fact that coefficient matrix $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{P\Pi}$ has the size $n = NK$, the maximum iteration number of KSM algorithm has decreased from NK to $N + 1$ due to the features of RL problems. The KSM algorithm's convergence rate improves as the iteration number is increased, whereas the SIM algorithm's convergence rate remains constant. For these reasons, the KSM algorithm has a smaller iteration number and is more efficient for RL problems than SIM. The numerical examples have

TABLE VIII
INVERTED PENDULUM PROBLEM. ITERATION NUMBERS FOR TOTAL LEARNING BY TD(λ) WITH $\lambda = 0.7$. (GS: GAUSS-SEIDEL, STAB: BICGSTAB)

Policy	PEI	Jacobi	GS	SOR	BCG	CGS	STAB
Total	7603	7358	6098	5585	1097	926	684
Time (s)	1770.7	1603.0	1347.0	1229.3	433.7	299.1	227.42

also shown the above results. Therefore, the methods based on KSM are recommended for RL problems.

Recently, some methods of Temporal Difference (TD) learning with function approximation have been developed [2], [14], [15], [16], [17], [18], [19]. The proposed KSM can be combined with popular learning methods: Least Squares TD (LSTD), Recursive LSTD, Kernel LSTD, etc. The algorithm developments have been left for future subjects, where combined methods are expected to be more effective than the existing methods.

APPENDIX
DEFINITION OF SYMBOLS

Symbols of state transition probability, action selecting probability, costs and values are shown as follows: where $\boldsymbol{I}$ is an identity matrix. The $\boldsymbol{e}_i$ denotes a vector, the i-th element is 1 and all others are 0.

$$
\begin{aligned}
[ik] &\equiv (i-1)\times K+k \\
\boldsymbol{\pi}(s_i) &\equiv [\ \pi(s_i,u_1),\pi(s_i,u_2),\ldots,\pi(s_i,u_K)\ ]^T \\
\boldsymbol{\Pi}(s_i) &\equiv \boldsymbol{e}_i\boldsymbol{\pi}^T(s_i) \\
\boldsymbol{\Pi} &\equiv [\ \boldsymbol{\Pi}(s_1),\boldsymbol{\Pi}(s_2),\ldots,\boldsymbol{\Pi}(s_N)\ ] \\
\boldsymbol{p}_{i|j\neq 0}(u_k) &\equiv [\ p_{i1}(u_k),p_{i2}(u_k),\ldots,p_{iN}(u_k)\ ]^T \\
\boldsymbol{P}_i &\equiv [\ \boldsymbol{p}_{i|j\neq 0}(u_1),\boldsymbol{p}_{i|j\neq 0}(u_2),\ldots,\boldsymbol{p}_{i|j\neq 0}(u_K)\ ]^T \\
\boldsymbol{P} &\equiv [\ \boldsymbol{P}_1^T,\boldsymbol{P}_2^T,\ldots,\boldsymbol{P}_N^T\ ]^T \\
\boldsymbol{g}(s_i,u_k) &\equiv [\ g(s_i,u_k,s_0),\ldots,g(s_i,u_k,s_N)\ ]^T \\
\bar{\boldsymbol{G}}(s_i,u_k) &\equiv \boldsymbol{e}_{[ik]}\boldsymbol{g}^T(s_i,u_k) \\
\bar{\boldsymbol{G}}(s_i) &\equiv [\ \bar{\boldsymbol{G}}(s_i,u_1),\bar{\boldsymbol{G}}(s_i,u_2),\ldots,\bar{\boldsymbol{G}}(s_i,u_K)\ ] \\
\bar{\boldsymbol{G}} &\equiv [\ \bar{\boldsymbol{G}}(s_1),\bar{\boldsymbol{G}}(s_2),\ldots,\bar{\boldsymbol{G}}(s_N)\ ] \\
\bar{\boldsymbol{g}} &\equiv \bar{\boldsymbol{G}}\boldsymbol{p} \\
\boldsymbol{Q}(s_i) &\equiv [\ Q(s_i,u_1),Q(s_i,u_2),\ldots,Q(s_i,u_K)\ ]^T \\
\boldsymbol{Q} &\equiv [\ \boldsymbol{Q}^T(s_1),\boldsymbol{Q}^T(s_2),\ldots,\boldsymbol{Q}^T(s_N)\ ]^T
\end{aligned}
$$

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning, An Introduction*. Cambridge, MA: MIT Press, 1998.

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

[3] S. Fujii, K. Senda, and S. Mano, "Acceleration of reinforcement learning by estimating state transition probability model," *Transactions of Society of Instrument and Control Engineers*, vol. 42, no. 1, pp. 47–53, Jan. 2006.

[4] H. Hironaka, Ed., *Gendai Suuri Kagaku Jiten*. Tokyo, Japan: Maruzen Publishing, 2009.

[5] M. Mori, M. Sugihara, and K. Murota, *Linear Caululation*. Tokyo, Japan: Iwanami, 1994.

[6] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1995.

[7] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, Dec. 1952.

[8] R. Fletcher, "Conjugate gradient methods for indefinite systems," *Lecture notes in Mathematics*, vol. 506, pp. 73–89, 1976.

[9] H. A. van der Vorst, "Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, March 1992.

[10] M. Natori, "The BCG method and the CGS method," *RIMS Kokyuroku, Kyoto University*, vol. 613, pp. 135–143, 1987.

[11] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.

[12] T. Yamamoto, *Fundamentals of Matrix Calculus*. Tokyo, Japan: Science-Sha, 2010.

[13] A. Dutech, T. Edmunds, J. Kok, M. Lagoudakis, M. Littman, M. Riedmiller, B. Russell, B. Scherrer, R. Sutton, S. Timmer, N. Vlassis, A. White, and S. Whiteson, "Reinforcement learning benchmarks and bake-offs ii," in *Workshop in Annual Conference on Neural Information Processing Systems*, Whistler, Canada, Dec. 2005.

[14] C. Szepesvári, *Algorithms for Reinforcement Learning*. San Rafael, CA: Morgan and Claypool, 2010.

[15] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Hoboken, NJ: Wiley, 2007.

[16] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

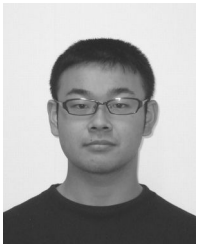
[17] X. Xu, D. Hu, and X. Lu, "Kernel based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[18] J. Boyan, "Technical update: Least-squares temporal difference learning," *Machine Learning*, vol. 49, no. 2–3, pp. 233–246, 2002.

[19] S. Mahadevan and M. Maggioni, "Proto-value functions: A laplacian framework for learning representation and control in markov decision processes," *Journal of Machine Learning Research*, vol. 8, pp. 2169–2231, 2007.

Kei Senda Kei Senda received the M.S. degree in aeronautical engineering in 1988 and the Ph. D. in engineering in 1993, both from Osaka Prefecture University. From 1988 to 2008, he worked for Osaka Prefecture University and Kanazawa University. From 2008, has been a professor of Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University. His research activities have included more than 100 papers on the dynamics and control of aerospace systems, the intelligence and autonomy for mechanical systems, and the motion intelligence of animals. He received the Best Presented Paper Award of the AIAA Guidance, Navigation, and Control Conference in 1992, the Best Paper Award of the Institute of Systems, Control and Information Engineers in 1994, the Best Paper of the Multi-Conference on Systemics, Cybernetics and Informatics in 2003, Finalist of the Best Conference Paper of IEEE International Conference on Systems, Man, and Cybernetics in 2011, the Best Paper Award of IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics in 2012, etc. He is a member of IEEE.

Suguru Hattori Suguru Hattori received the M.S. degree in aeronautics and astronautics in 2013 from Kyoto University. His interests include the intelligence of robots and reinforcement learning.



Toru Hishinuma Toru Hishinuma is a graduate student of Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University. His is in charge of researches on the intelligence of robots and reinforcement learning.

Takehisa Kohda Takehisa Kohda received the M.S. degree in aeronautics and astronautics in 1980 and the Ph. D. in engineering in 1983, both from Kyoto University. From 1983 to 1988, he worked for Mechanical Engineering Laboratory, Agency of Industrial Science and Technology. He was a research associate from 1988 to 1993 and he has been an associate professor from 1993, both of Department of Aeronautics and Astronautics, Graduate School of Engineering, Kyoto University. His research activities have included more than 100 papers on the reliability and safety of systems. He received the Best Paper Awards from the Society of Instrument and Control Engineers in 1983 and 1992, the Best Paper Award from the Japan Society for Safety Engineering in 1986. He is a member of IEEE.