2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Dynamic Programming for Global Control of The Acrobot and Its Chaotic Aspect

Ryuichi Ueda and Tamio Arai

Abstract— **In this paper, we investigate characteristics of a dynamic programming (DP) algorithm when it is applied to a control problem of a chaotic system. We choose a height task of the Acrobot, which is an underactuated robot. For various reasons, a straightforward method of DP is thought that it is not suitable for such a system. However, DP is attractive because of its conciseness, versatility, and flexibility toward boundary conditions. Our simulation results suggest that to apply DP methods to the task is possible under some conditions. Moreover, even when the conditions are not fulfilled, some interesting behavior of the Acrobot can be found.**

I. Introduction

Control of robots that have chaotic characters has been one of the most difficult challenges in robotics. One of the most important characteristics of chaotic systems is sensitivity to differences of state [1]. When no control is added to such a system, state transitions from two states in a vicinity suddenly diverge after a few moments. This effect is frequently referred to as the *butterfly effect*, or as *sensitive dependence on initial conditions*. To forecast the state of a system after a certain period of time, we must know an initial state and the dynamics with complete accuracy.

It does not mean that such a system is uncontrollable. In some cases, chaotic features can be eliminated by chaos control methods as typified by Ott-Grebogi-Yorke (OGY) control methods [2] and Pyragas's methods [3], which have been proposed in the 90's.

In this paper, we handle a control method of a type of the Acrobot defined in [4]–[9]. This robot is an underactuated robot, which has two links and one actuator. In almost all of the sets of parameters, its behavior without control has no certain orbit. Its behavior is thus chaotic.

In many studies, the Acrobot has been successfully controlled. Just as the above chaos control methods, kinds of feedback control are used in [4], [10]. In some studies, on the other hand, reinforcement learning methods are tested on the Acrobot [9], [11]. The former tries regulating the motion, like a vibratory control does. From that viewpoint, controllers obtained by these methods are artificial. Though the artificiality is not a problem, those methods sometimes cannot deal with problems about boundary conditions, or trivial constraints of hardware. The latter is rather natural. Learning algorithms understand dynamics of the Acrobot from some trials and make the Acrobot swing high or stand up. In this regard, however, learning methods can consider only a part of possible states of the Acrobot. The states exist infinitely in continuous space. Trials for learning are hardly adequate when we consider the sensitivity. The cases where the number of trials is adequate would be followings: available torque and frequency of control are enough to eliminate the sensitiveness, or almost all of the trials are finished before the butterfly effect occurs.

Numerical calculation is located midway in the above approaches. Some methods of dynamic programming (DP) [12] can handle boundary conditions without fatal problems. Moreover, some methods of DP must consider whole space of states fatefully and it is sometimes possible. When we try to apply DP with discretization to a chaotic system, however, the problem is not perfectly solved for many reasons. However, when we can use a powerful computer, the conciseness, versatility, and flexibility toward boundary conditions of DP are attractive.

The purpose of this paper is to find some characteristics of a straightforward implementation of DP with discretization when it is daringly applied to such a chaotic system. The value iteration algorithm is used as the implementation. As mentioned below, we find that DP can be successfully applied to a chaotic system under some conditions. Moreover, some interesting results are obtained when the conditions are not fulfilled.

We remove heuristics from our implementation as far as possible. In the case of the Acrobot, energy is used as such a heuristic [8]. Though some assumptions are inevitably required, they are usually in common with those of actual robots. As a result, this research requires huge computational resources. These results could never be obtained in previous decades. Thus no one has ever obtained the calculation results as far as we know. Though we have one result in [13], an easy setting of parameters was used in the study.

This paper is composed of five sections. In Sec.II, a problem of the Acrobot is defined. In Sec.III, our algorithm is implemented. We discuss interesting phenomena shown in the results of our algorithms in Sec.IV. This paper is concluded in Sec.V.

II. Robot, Task and Assumption

A. Its Dynamics

The Acrobot is a two-link robot shown in Fig.1. Link 1 can swing freely at this joint. Torque τ can be given at Joint 2. Its pose is represented by the pair of angles (θ_1, θ_2). As shown in the figure, the length of Link i is represented by ℓ_i. ℓ_{ci} denotes the distance from Joint i to mass center of

R. Ueda and T. Arai are with Department of Precision Engineering, School of Engineering, The University of Tokyo ueda@robot.t.u-tokyo.ac.jp

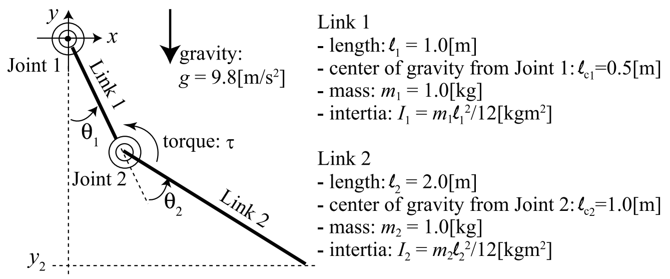


Fig. 1. The Acrobot and Its Parameters

Link i. The mass of Link i is m_i. The moment of inertia of Link i is then I_i.

The dynamics of this robot can be represented by

$$\ddot{\theta}_2 = \frac{(\tau + \phi_1 d_2/d_1 - m_2\ell_1\ell_{c2}\dot{\theta}_1^2 \sin\theta_2 - \phi_2)}{(m_2\ell_{c2}^2 + I_2 - d_2^2/d_1)}, \text{ and} \quad (1)$$

$$\ddot{\theta}_1 = -(d_2\ddot{\theta}_2 + \phi_1)/d_1, \text{ where} \quad (2)$$

$$d_1 = m_1\ell_{c1}^2 + m_2(\ell_1^2 + \ell_{c2}^2 + 2\ell_1\ell_{c2}\cos\theta_2) + I_1 + I_2,$$

$$d_2 = m_2(\ell_{c2}^2 + \ell_1\ell_{c2}\cos\theta_2) + I_2,$$

$$\phi_1 = -m_2\ell_1\ell_{c2}\dot{\theta}_2^2 \sin\theta_2 - 2m_2\ell_1\ell_{c2}\dot{\theta}_2\dot{\theta}_1 \sin\theta_2 + (m_1\ell_{c1} + m_2\ell_1)g\cos(\theta_1 - \pi/2) + \phi_2, \text{ and}$$

$$\phi_2 = m_2\ell_{c2}g\cos(\theta_1 + \theta_2 - \pi/2).$$

The energy of this Acrobot can be calculated as

$$E(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = m_1\ell_{c1}(1 - \cos\theta_1)g + m_2\ell_1(1 - \cos\theta_1)g + m_2\ell_{c2}(1 - \cos(\theta_1 + \theta_2))g + \tfrac{1}{2}I_1\dot{\theta}_1^2 + \tfrac{1}{2}I_2(\dot{\theta}_1 + \dot{\theta}_2)^2 + \tfrac{1}{2}m_1(\ell_{c1}\dot{\theta}_1)^2 + \tfrac{1}{2}m_2(\ell_1\dot{\theta}_1)^2 + \tfrac{1}{2}m_2(\dot{\theta}_1 + \dot{\theta}_2)^2\ell_{c2}^2 + m_2\dot{\theta}_1(\dot{\theta}_1 + \dot{\theta}_2)\ell_{c2}\ell_1\cos\theta_2 \quad (3)$$

when we define as $E(0, 0, 0, 0) = 0$.

This paper handles a simulated robot that has identical parameters with the Acrobot referred to in [7], [14]. The parameters are shown in the figure. They are more severe for the height task than the parameters used in [5], [8], and in our work [13]. The most important difference between them is the setting of inertia. The former regards the links as thin rods. The latter sets a simple value to every inertia and it is much larger than that of a thin rod. Since the values in [7], [14], and this paper are small, it is difficult to swing up by a large momentum when a speed limit is set as mentioned below.

B. Boundary Conditions, and Other Assumptions

The following assumptions are added to the Acrobot due to limitations of numerical calculations. These limitations sometimes reduce performance of a robot. However, actual robots sometimes require identical restrictions. On the other hand, methods of mathematical analysis frequently cannot deal with some kinds of limitation on actual robots (e.g. boundary conditions).

- The maximum speeds are limited to $\pm 4\pi$[rad/s] at Joint 1 and $\pm 9\pi$[rad/s] at Joint 2. The set of these values is shown in [5] and it is very interesting for a number of reasons. The limitation is not only for reducing the amount of calculations. It prevents hardware from burning up if we use an actual robot for experimentation. Moreover, this limitation banishes *brute force* solutions. The more the Acrobot charges its mechanical energy, the riskier the trial is. For example, $E(\pi, 0, 0, 0) = 5g \approx 49$[N·m] and $E(0, \pi, 4\pi, -\pi) = 14\pi^2/3 \approx 46$[N·m] when Eq.(3) is calculated with the parameters in Fig.1. These values imply that the Acrobot that stands up has enough energy to violate the limitation.
- Chances to change the torque τ visit every 0.2[s]. The value of the torque at each time instant is chosen from a finite set: $\mathcal{T}$.
- The $\theta_1\theta_2\dot{\theta}_1\dot{\theta}_2$-space, which is written as $\mathcal{X}$, is then divided into 10[deg]×10[deg]×10[deg/s]×10[deg/s]. Though these values can be easily changed in the value iteration mentioned below, we fix them beforehand. That is because those values also become a limitation on hardware (e.g. sensors, delay of information). By the discretization, state space is divided into $60, 466, 176$ discrete states when the speed limitation is considered. A discrete state is symbolized as s. The set of these discrete states are written as $\mathcal{S}$. Each state in $\mathcal{S}$ has a symmetry state, which is identical to when we observe the Acrobot in Fig.1 from the back side of the paper.

C. Height Task under Boundary Conditions

The height task is given to the Acrobot in this paper. The word *height task* can be shown in [8]. In this task, the Acrobot is controlled so that the free end of Link 2 gets over a target height. When the height of the free end of Link 2 is represented by y_2 in the xy coordinate system defined in Fig.1, the relation between y_2 and the pair of (θ_1, θ_2) is

$$y_2 = -\ell_1\cos\theta_1 - \ell_2\cos(\theta_1 + \theta_2). \quad (4)$$

When the target height is denoted by h on the y-axis, $y_2 \geq h$ is the purpose of control.

A continuous state that fulfills $y_2 \geq h$ is called a successful final state in this paper. We define $\mathcal{X}_{\text{success}}$ as the set of all successful state. On the other hand, there are states whose angular velocities violate the boundary conditions: $|\dot{\theta}_1| > 4\pi$[rad/s] or $|\dot{\theta}_2| > 9\pi$[rad/s]. Such a state is called a failure final state. Their set is written as $\mathcal{X}_{\text{failure}}$ hereafter.

In this paper, the task is regarded as a problem of minimum time control under boundary conditions. In this case, the evaluation functional is defined as

$$J[\tau] = V(\boldsymbol{x}(T)) + \int_0^T -1dt = V(\boldsymbol{x}(T)) - T, \quad (5)$$

where T is the first time when $\boldsymbol{x} \in \mathcal{X}_{\text{success}}$ or $\boldsymbol{x} \in \mathcal{X}_{\text{failure}}$. τ is regarded as a function that gives the torque at time t $(0 \leq t < T)$ in the above equation. $V(\boldsymbol{x}(T))$ is defined as

$$V(\boldsymbol{x}(T)) = \begin{cases} 0 & \text{if } \boldsymbol{x}(T) \in \mathcal{X}_{\text{success}} \\ v_{\text{failure}} & \text{if } \boldsymbol{x}(T) \in \mathcal{X}_{\text{failure}}, \end{cases} \quad (6)$$

where v_{failure} is a large negative value, which means a penalty.

III. Value Iteration for Height Task

Dynamic programming (DP), which has been established by Bellman [12] is a group of methods for solving optimal control problems and Markov decision processes (MDPs) with numerical calculation. DP can solve optimal control problems in discrete time and space. The value iteration algorithm is an implementation of DP.

A. Approximation to Finite Markov Decision Process

A state of the Acrobot can be represented by $\boldsymbol{x} = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathcal{X}$. This definition fulfills the Markov property. In continuous space, the state transition is deterministic and follows Eq.(1)-(2). When we want to obtain a state $\boldsymbol{x}'$ after torque τ is applied, the posterior state $\boldsymbol{x}'$ after 0.2[s] can be calculated by a Runge-Kutta method, for example. Though this is an approximation, this transition is still deterministic.

However, when the continuous space $\mathcal{X}$ is discretized into the discrete set $\mathcal{S}$ as mentioned in Sec.II-B, we must handle state transitions as stochastic. A state transition can be represented by $\mathcal{P}^{\tau}_{ss'}$. This symbol denotes a state transition probability that the posterior state $\boldsymbol{x}'$ is contained in $s' \in \mathcal{S}$ by a torque τ. It is approximated as

$$\mathcal{P}^{\tau}_{ss'} = \int_{\boldsymbol{x}\in s} P(\boldsymbol{x}' \in s') p(\boldsymbol{x}|\boldsymbol{x} \in s) d\boldsymbol{x}. \qquad (7)$$

Since the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{x} \in s)$ is unknown, it is usually approximated as a uniform distribution. Therefore, calculations of $\mathcal{P}^{\tau}_{ss'}$ are approximation.

The set of $\mathcal{X}_{\text{success}}$ has to be discretized too. The discrete set is referred to as $\mathcal{S}_{\text{success}}$. We decide that all of continuous states in $s \in \mathcal{S}_{\text{success}}$ must be in $\mathcal{X}_{\text{success}}$. $\mathcal{X}_{\text{failure}}$ is not discretized since this set is outside of $\mathcal{S}$.

In DP, the value of each state $V^*(s) \in \Re$ is defined. This value denotes the expected value of the functional Eq.(5) when the optimal torques are always chosen from a continuous state in s. V^* is called the optimal state value function. Values of V^* are fixed in the final states. That is based on Eq.(6) and the definitions of $\mathcal{S}_{\text{success}}$ and $\mathcal{S}_{\text{failure}}$. If chaotic properties of the Acrobot are neglected, V^* fulfills the following equation:

$$V^*(s) = \max_{\tau\in\mathcal{T}} \sum_{s'} \mathcal{P}^{\tau}_{ss'}[-0.2 + V^*(s')] \qquad (8)$$
$$(s \in \mathcal{S} - \mathcal{S}_{\text{success}};\ s' \in \mathcal{S} \text{ or } s' \subset \mathcal{X}_{\text{failure}}).$$

This equation is a Bellman equation. The word *dynamic programming* (DP) [12] means the methodology or methods to solve Bellman equations.

The value iteration algorithm [11], [15], [16] is one of the most popular methods in DP. In this algorithm, the following procedure:

$$V(s) \longleftarrow \max_{\tau\in\mathcal{T}} \sum_{s'} \mathcal{P}^{\tau}_{ss'}[-0.2 + V(s')] \qquad (9)$$

is repeatedly applied to all states except for final states until V converges to V^*. When the above procedure is applied to all $s \in \mathcal{S} - \mathcal{S}_{\text{success}}$, the set of procedures is called a *sweep*.

The optimal policy, which give the best torque τ at s, is then obtained by:

$$\tau^*(s) = \operatorname*{argmax}_{\tau\in\mathcal{T}} \sum_{s'} \mathcal{P}^{\tau}_{ss'}[-0.2 + V^*(s')]. \qquad (10)$$

In this case, τ^* is regarded as a mapping from $\mathcal{S} - \mathcal{S}_{\text{success}}$ to $\mathcal{T}$. Note that τ^* can answer the torque at *every continuous state*: $\forall \boldsymbol{x} \in s \in \mathcal{S} - \mathcal{S}_{\text{success}}$.

B. Calculation of State Transition Probabilities

$\mathcal{P}^{\tau}_{ss'}$ should be calculated in feasible time. To solve $\mathcal{P}^{\tau}_{ss'}$ with the procedure in Eq.(7), we need to investigate all state transitions from all $\boldsymbol{x} \in s$. However, this calculation is intractable.

In our past work [13], only four continuous states in s were used for the calculation. Though we change the number of continuous states that are sampled in a discrete state, this method is also used in this paper.

When μ continuous states $\boldsymbol{x}^{[i]}$ $(i = 1, 2, \ldots, \mu)$ are chosen from s, $\mathcal{P}^{\tau}_{ss'}$ is regarded as

$$\mathcal{P}^{\tau}_{ss'} = \frac{1}{\mu}\sum_{i=1}^{\mu} I(\boldsymbol{x}'^{[i]} \in s'), \qquad (11)$$

where $I = 1$ if the argument is true. If it is false, $I = 0$. $\boldsymbol{x}'^{[i]}$ is the posterior state from $\boldsymbol{x}^{[i]}$. In our implementation, this equation is used in the procedure Eq.(9) as

$$V(s) \longleftarrow -0.2 + \max_{\tau\in\mathcal{T}} \frac{1}{\mu}\sum_{i=1}^{\mu} V(s' \ni \boldsymbol{x}'^{[i]}). \qquad (12)$$

A 4th order Runge-Kutta method is used for the calculation of each posterior state in our implementation. To solve one posterior state, this method uses hundreds of arithmetic operations. However, there is no problem when the latest computer technology is used.

The value of $V^*(s)$ should be

$$V^*(s) = -E[T]P_{\text{success}} - (E[T'] - v_{\text{failure}})P_{\text{failure}} \qquad (13)$$

in an ideal calculation. $E[T]$ and $E[T']$ are the expected values of time to each type of final state with an optimal policy, and P_{success} and P_{failure} are the probabilities of success and failure respective. Since the value function calculated by value iteration without heuristics is directly related to a meaningful quantity, its results have large reliability.

C. Expected Problems on Discretization

In a precise sense, the value iteration algorithm explained above cannot solve problems on chaotic systems, which include tasks of the Acrobot. We can understand the fact from the following brief analysis.

1) Averaging of Values: Each value $V^*(s)$ is averaged in discrete state s. That is a problem when it is calculated toward a task of the Acrobot. Since its motion becomes different with small difference of states, each continuous state $\boldsymbol{x} \in s$ may have a much different value from the others. Even if the system is continuous, a chaotic system fulfills

$$\lim_{\boldsymbol{\varepsilon}\to\boldsymbol{0}}\{V^*(\boldsymbol{x}) - V^*(\boldsymbol{x} - \boldsymbol{\varepsilon})\} \neq 0 \ (\forall \boldsymbol{x} \in s) \qquad (14)$$

from any direction except for forward and inverse directions of the state transition. It implies that a control that is obtained with discretization cannot be confirmed not only as optimal but also as good approximation. Even if the discretization is extremely fine, it is impossible.

2) Accumulation of Errors: In value iteration, moreover, the averaged value is used for calculating values of other states. Therefore, errors diffuse and accumulate. We should expect that values of discrete states that are far from all final states cannot be calculated exactly.

D. Calculation

Though the problems are clear, we create global policies with various combinations of available sets of τ and μ. The target height is set to $h = 2$[m]. Though the computation time sometimes reaches to tens of days, we do not think that it is a problem. That is because once a global policy, which is a look-up table, can be obtained, it can answer a torque toward every state instantly even on a very cheap computer.

We prepare six sets of torques $\mathcal{T}_i$ $(i = 1, 2, 3, 4, 5, \text{all})$. In other words, six kinds of actuator are prepared. i is directly related to the values of torque τ[N]. When $i = 1, 2, \dots, 5$, $\mathcal{T}_i = \{\pm i, 0\}$. In the case of $\mathcal{T}_{\text{all}}$, the Acrobot can choose all of the torques. Hence, $\mathcal{T}_{\text{all}} = \{\pm i | i = 0, 1, 2, \dots, 5\}$.

$\mu = 1, 4, 16, 81$ are chosen as the numbers of sample points. In every case, the problem is which continuous states should be chosen as samples from a discrete state. When $\mu = 1$, the center point of each discrete state should be sampled. When $\mu = 4$, the center point is chosen on $\theta_1\theta_2$-plane. In $\dot{\theta}_1\dot{\theta}_2$-plane, four corners of a discrete state are chosen. This choice is identical with that of [13]. When $\mu = 16$, all corners of a discrete state is chosen. When $\mu = 81$, three points are sampled on each axis. By their combinations from every axis, 81 continuous states can be sampled. The three points are a midpoint and both ends of a discrete states.In value iteration, $v_{\text{failure}} = -1000$[s] is given as a large penalty. Though this parameter may change the results of value iteration, we fix it in this paper.

The computer that is used for value iteration has the following resources: two Xeon-D5160 (3.9GHz), 8GB DRAM (PC2-5300 CL5 DDR2), and two 250GB HDD. Our implementation uses only one process, while four processes are available on the computer. In the value iteration, the difference of value by the procedure in Eq.(12) is checked. If the maximum change is less than 10^{-4}[s] in a sweep, we regard it as convergence and the process is stopped.

Table I shows the computation time to the convergence. The values in this table are rather rough because more than two processes run on the computer at the same time. The most important thing in this table is that all of the processes converge. If the number of μ and available torques are fixed, the difference of time occurs due to the difference of the number of sweeps. From that perspective, small torque induces many sweeps when $\mu = 16, 81$.

The time consumption can be drastically reduced if all state transitions are calculated and recorded in advance. If all results of the Runge-Kutta method are cached, the algorithm does not have to execute hundreds of operations to obtain one state transition result in each sweep. In that case, however, a huge amount of memory is required.

TABLE I
TIME TO CONVERGENCE

	time ($\mu = 1$: [min], the others: [day])					
μ	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_{\text{all}}$
1	5	5	7	7	7	29
4	0.2	0.1	0.2	0.1	0.2	0.2
16	3	1	0.8	0.6	0.7	4
81	11	6	4	3	3	11

E. Obtained Control Policies

We show parts of policies in Fig.2 and in Fig.3. A pose of the Acrobot is illustrated in Fig.2(a). The angles are $(\theta_1, \theta_2) = (5, 5)$[deg]. Fig.2(b)-(d) show parts of policies and value functions that relate to the pose. From (b) to (d), $\mathcal{T}_1$, $\mathcal{T}_3$, are $\mathcal{T}_5$ used respectively. Each figure from (b) to (d) contains two kinds of gray scale image. The upper one gives the torque for each pair of velocities: $(\dot{\theta}_1, \dot{\theta}_2)$. Black and white represent positive and negative torques respectively. The gray part corresponds to $\tau = 0$. The lower image represents the value of each state. White represents $V^*(s) = 0$ and black represents $V^*(s) \leq -60$[s]. When $-60 < V^*(s) \leq 0$, the strength of gray is in proportional to the value.

When the pose of the Acrobot is down as shown in (a), we can intuitively imagine three phases: acceleration, braking, and final swinging up. When both of the angular velocities are small, the robot must enhance its energy by swing-like motion. In each policy, black and white parts are changed around the origin of $\dot{\theta}_1\dot{\theta}_2$-plain. This contrast represents this motion. When the velocities are relatively high, there is a danger of violation of speed after some seconds. Therefore, the direction of torque at the high velocity region is different from the region around the origin as shown in the images. If the set of $(\dot{\theta}_1, \dot{\theta}_2)$ is suitable for successfully finishing the task, the torque should be decided so as to bring the toe of the robot safely and quickly to the target height. From the images, however, we cannot understand where that region exists on each images.

In Fig.3(b)-(d), the set of torque is fixed to $\mathcal{T}_5$, while μ is changed. Policies are sliced at $\theta_1 = 85$[deg] and $\theta_2 = 175$[deg]. The pose of this moment is shown in Fig.3(a). The slice with $\mu = 1$ is not shown here since no meaningful pattern is obtained.

In every slice of policy, an area that is assigned the negative torque exists in the $\dot{\theta}_1 > 0$ and $\dot{\theta}_2 < 0$ part. The explanation of this white area is rather easy. From almost all of this area, the Acrobot can finish the task successfully without any control. By the negative torque, the Acrobot can reduce the time consumption.

There is a vertical stripe pattern in the $\dot{\theta}_2 > 0$ area of each slice. The larger μ is, the sharper this pattern is. Since this stripe is not strongly related to $\dot{\theta}_1$, we cannot grasp why such a control rule is obtained.

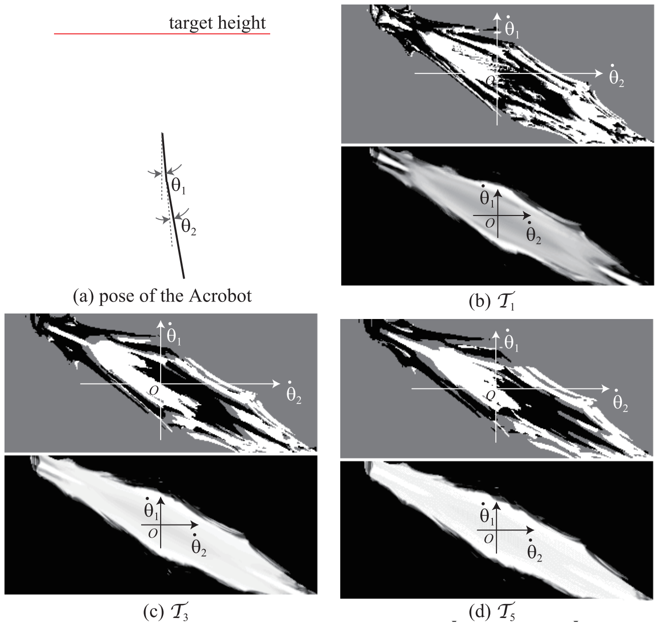


Fig. 2. Parts of Value Iteration Results when $\mu = 81$ (The length of each axis is changed for visibility.)

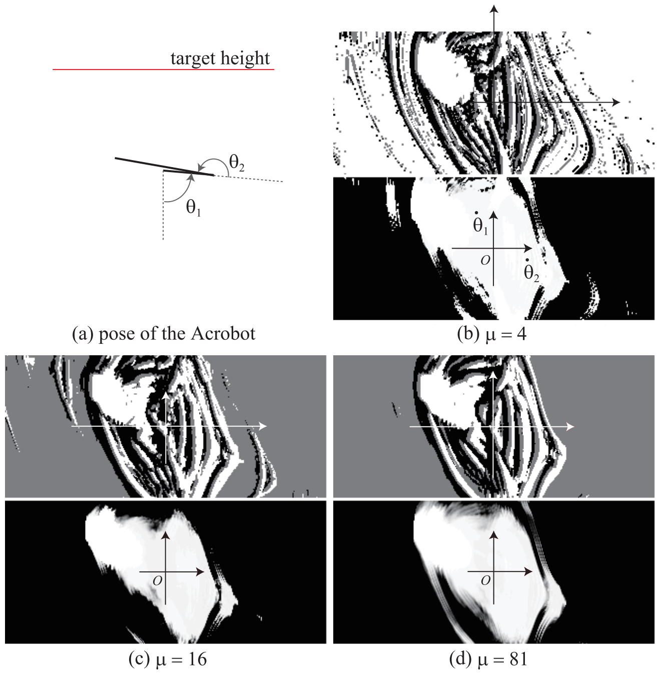


Fig. 3. Parts of Results for $\mathcal{T}_5$

IV. Simulation with Different Parameters

A. Simulated Acrobot

The dynamics of the Acrobot is simulated by the 4th order Runge-Kutta method that is also used in the value iteration. The time interval for computation is set to 0.01[s], which is $1/20$ of the time interval for decision of torque. The accuracy of this calculation can be evaluated from Fig.4. Since there is only one trial, it is just for reference. This graph shows the energy of the Acrobot that is not given torque. Unfortunately, the energy is not constant but constantly reducing. If the interval is shorter than 0.01[s], the simulation will be more accurate. However, there may be no end to the accuracy. Since the energy does not increase unnaturally, it is enough to evaluate control policies.

Figure 5 shows each pose of the Acrobot on $\theta_1\theta_2$-plane (two-dimensional torus, to be exact) at every time instant when we measure the energy loss with 0.01[s] interval. The Acrobot has no certain orbits and covers the torus except for a high energy region.

B. The Purposes

There are three purposes of simulation:

- to evaluate the feasibility of the value iteration algorithm toward the task,
- to investigate the performance with different numbers of sampling points and different sets of available torques, and
- to find some unique results.

In a trial of the simulation, the time from an initial state to a final state in $\mathcal{S}_{\text{success}}$ or $\mathcal{S}_{\text{failure}}$ is recoded. The trials are held with various initial states though we fix the velocities as zero. Initial states are set to $(\theta_1, \theta_2) = (3i, 3j)$[deg] $(i, j =$

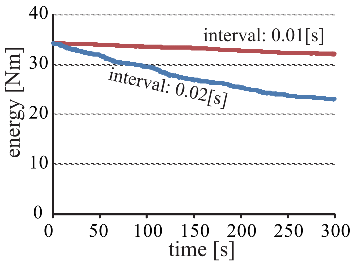


Fig. 4. Reduction of Energy by Approximation Error

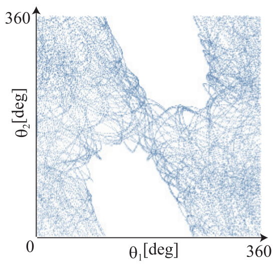


Fig. 5. Poses at Every 0.01[s] Step

$0, 1, 2, \ldots, 119)$ and $(\dot{\theta}_1, \dot{\theta}_2) = (0, 0)$. All final states and symmetric states are removed from the set and 6,399 trials are held for each policy. We have to stop a trial if the Acrobot does not finish the task for a long time. We set $|v_{\text{failure}}| =$ 1000[s] as the limit.

As a reference, we have prepared a brute force control policy for the height task, which can be also shown in [13]. When one of the sets $\mathcal{T}_i$ $(i = 1, 2, \ldots, 5)$ is used, the policy is based on

$$\tau = \begin{cases} -i & \text{if } \dot{\theta}_1 > 0 \\ i & \text{otherwise} \end{cases}. \tag{15}$$

This is a certain rule for increasing the energy. If policies that are obtained by value iteration are poorer than the above control rule, the policies are meaningless.

We think that more efficient policies will be obtained if analytical methods in previous works are used for planning. However, policies that are obtained by the straightforward value iteration are meaningful in the point of versatility of the method.

C. Evaluation

Since a functional such as Eq.(5) tells only one evaluation criterion, the values of the functional should be used for evaluation. However, Eq.(5) does not consider a trial that does not finish within the time limit. Moreover, since we handle a harder problem than that of [13], failure occurs more frequently.

In this paper, a control policy is evaluated by its success rate and its average time needed at successful trials. We think that the success rate is more important than the average time. That is because a large penalty is given to the overspeed limitation. Moreover, the average time can be exactly discussed when all of trials can be successfully finished.

TABLE II

SIMULATION RESULTS (**from 6,399 different poses**)

(a) Percentage of Success [%]

μ	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_{\text{all}}$
brute force	65.4	53.1	48.1	40.9	36.4	—
1	*8.0*	*18.7*	74.8	68.0	64.0	63.8
4	97.2	91.4	84.4	80.3	78.1	75.9
16	*61.1*	**99.4**	**99.5**	98.3	97.9	96.7
81	**99.5**	98.7	**99.2**	98.2	96.7	96.0

(b) Average Time on Successful Trials [s]

μ	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_{\text{all}}$
brute force	8.2	4.5	3.2	2.5	2.1	—
1	208.6	138.6	158.2	17.8	9.6	9.6
4	43.4	13.6	7.2	5.0	3.7	4.2
16	405.6	**74.0**	**13.2**	7.4	5.3	14.3
81	**147.9**	47.4	**10.8**	6.7	5.1	11.1

The results are shown in Table II. Table (a) shows the success probabilities. In Table (b), values of average time are illustrated. These averages are calculated without failure (overspeed or more than 1000[s]) trials.

In the table, we can grasp some understandable tendencies though we have expected poorer results as explained in Sec.III-C. The more points are sampled, the higher the success probabilities become. In the cases of $\mathcal{T}_i$ $(i = 2, 3, 4, 5)$, large numbers of sampling bring good results. The higher torque we can use, then, the shorter the average time becomes. In the cases of $\mu = 1$, low torques $(|\tau| \leq 2[\text{N}])$ cannot cause many state transitions due to rough discretization. Therefore, the success probabilities of $(\mu, \mathcal{T}) = (1, \mathcal{T}_1), (1, \mathcal{T}_2)$ are much smaller than those of the brute force method.

On the other hand, there are some unaccountable or unique results in the figures. First of all, the averages of time are too long when $\mathcal{T}_1$ and some cases of $\mathcal{T}_2$. If we use a feedback method, such a long time solution may never obtained. Table III implies the reason. These tables shows the required time to finish from slightly different initial states with two policies that are obtained with $\mu = 81$. $\theta_2, \dot{\theta}_1$ and $\dot{\theta}_2$ are fixed to zero, while θ_1 is slightly changed with 0.001[deg] step (not [rad]). This table shows the results with $\mathcal{T}_i$ $(i = 1, 3, 5)$. As shown in the cases of $\mathcal{T}_1$, the chaotic property cannot be eliminated when the torque is small although all trials are successfully finished over a long time.

Even when $\mathcal{T}_3$, time and orbits are changed by the slight differences. Such a change of orbits is sometimes a fatal problem for a control policy. That is because a policy must prepare all of possible orbits. Since the policy with $\mu = 81$ for $\mathcal{T}_3$ can deal with the changes and finish the task in short time, it has good property as a policy that is obtained by value iteration. In other words, it can give a proper torque for every state without feedback.

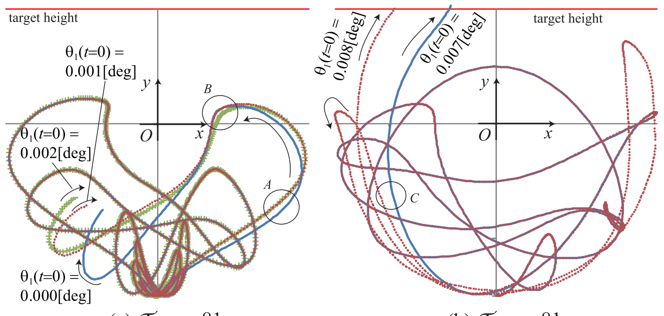
In the case of $\mathcal{T}_5$, the task is finished before butterfly effects appear until $\theta_1 = 0.008$[deg]. Since the Acrobot is more powerful than the case of $\mathcal{T}_3$, this result is not a surprise. However, as shown in Table II, the risk of overspeed is larger than the policy for $\mathcal{T}_3$. Moreover, each policy for $\mathcal{T}_{\text{all}}$ is not the best one in the policies that have identical μ. When $\mathcal{T}$ is a subset of $\mathcal{T}'$, the optimal policy obtained for $\mathcal{T}'$ is equal to or more efficient than the optimal policy for $\mathcal{T}$ as a principle of DP. It implies that some problems as mentioned in Sec.III-C gives bad effect to the policies.

Finally, we show the observed sensitivity in some trials on Table III. They are orbits of the toe of Link 2 on xy-plain. Figure 6(a) shows three orbits from $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (i \cdot 0.001[\text{deg}], 0, 0, 0)$ $(i = 0, 1, 2)$. In this figure, the orbits after about 14.5[s] are omitted. The deviance of orbits starts suddenly. While the Acrobot is swinging like a pendulum, these three orbits are virtually identical. After the pendulum motion, the toe of the Acrobot draws complicated orbits. When the toe starts swinging to the right side widely, the orbit from $\theta_1 = 0.000$[deg] breaks away from the others.

TABLE III

TIME FROM SLIGHTLY DIFFERENT INITIAL STATES

initial	$\mathcal{T}_1 = \{\pm1, 0\}$[N]		$\mathcal{T}_3 = \{\pm3, 0\}$[N]		$\mathcal{T}_5 = \{\pm5, 0\}$[N]	
θ_1[deg]	time[s]	result	time[s]	result	time[s]	result
0.000	54	success	9.79	success	7.75	success
0.001	103	success	8.16	success	7.75	success
0.002	420	success	8.16	success	7.75	success
0.003	198	success	8.17	success	7.75	success
0.004	69	success	10.00	success	7.75	success
0.005	827	success	10.16	success	7.75	success
0.006	588	success	10.16	success	7.75	success
0.007	36	success	10.17	success	7.75	success
0.008	102	success	10.07	success	11.15	success
0.009	305	success	10.45	success	7.90	success



(a) $\mathcal{T}_1$, μ = 81 (b) $\mathcal{T}_5$, μ = 81

Fig. 6. Different Orbits from Proximity States (Sensitive Dependence on Initial Conditions)

We can see the derailment at "A" in the figure. After that, the orbits from $\theta_1 = 0.001$[deg] and $\theta_1 = 0.002$[deg] are apart from each other at "B". When these orbits come to the left side, their difference become significant.

Even in the case of $\mathcal{T}_5$, the sensitivity can be observed. Fig.6(b) compares the orbits from $\theta_1 = 0.007, 0.008$[deg]. The orbits are almost the same until the Acrobot from $\theta_1 = 0.007$[deg] is going to finish the task. In the case from $\theta_1 = 0.008$[deg], unfortunately, the toe of the Acrobot is drifted to left of the orbit from $\theta_1 = 0.007$[deg]. We can see it at "C". After that, the trial from $\theta_1 = 0.008$[deg] can be successfully finished with another swing.

V. Conclusion

This paper deals with the height task of the Acrobot, which has the parameters shown in [7], [14] and the delicate boundary conditions shown in [5]. From the straightforward usage of value iteration, we can obtain the following knowledge.

- We have assumed a crude Acrobot in the simulation. It must decide torques with the resolution of 10[deg], 10[deg/s], and 0.2[s]. Available torque is also limited. However, some policies can bring the Acrobot to the target height with more than 99[%].
- Even though the chaotic behavior is significant with small torque, the Acrobot can reach the target height with high probabilities for each conditions of torque. Even when $\mathcal{T}_1 = \{\pm 1, 0\}$[N], the success probability is 99.5[%] with $\mu = 81$. The policy seems to concentrate the Acrobot on avoidance of the overspeed.
- With $\mathcal{T}_5 = \{\pm 5, 0\}$[N], 0.001[deg] difference does not always cause the butterfly effect. That is because the trials can be finished before the effect occurs.
- With $\mathcal{T}_3 = \{\pm 3, 0\}$[N], the success rate is 99.5[%] with $\mu = 16$. In this case, the orbits are drifted by the 0.001[deg] difference of initial states. However, the time to finish is not too long (13.2[s] with $\mu = 16$ and 10.8[s] with $\mu = 81$). This result suggests the task can be finished without elimination of chaos.
- The problems mentioned in Sec.III-C cause bad effect for value iteration. The evaluation results of policies for $\mathcal{T}_{\text{all}}$ imply this fact.
- Even though accurate values cannot be calculated, the number of sampling points should be large as far as we can see from the results in Table II.
- It takes 11[days] to obtain a policy with one process of a Xeon-D5160 processor (3.9GHz) when $\mathcal{T}_1 = \{\pm 1, 0\}$[N] and 81 sampling points.

References

[1] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer-Verlag, 1990.

[2] E.Ott, C.Grebogi, and J.A.Yorke, "Controlling Chaos," *Physical Review Letters*, vol. 64, pp. 1196–1199, 1992.

[3] K. Pyragas, "Continuous Control of Chaos by Self-Controlling Feedback," *Physics Letters A*, vol. 170, pp. 421–428, 1992.

[4] M. W. Spong, "Swing Up Control of the Acrobot," in *Proc. of IEEE ICRA*, 1994, pp. 2356–2361.

[5] R. S. Sutton, "Generalization in Reinforcement Learning: Successful Examples Using Space Coarse Coding," in *Neural Information Processing Systems*, 1996, pp. 1038–1044.

[6] R. N. Banavar and A. D. mahindrakar, "Energy based swing-up of the Acrobot and Time-optimal Motion," in *Proc. of IEEE International Conference on Control Applications*, 2003, pp. 706–711.

[7] X. Xin and M. Kaneda, "The Swing up Control for the Acrobot based on Energy Control Approach," in *Proc. of IEEE Conf. on Decision and Control*, 2002, pp. 3261–3266.

[8] G. Boone, "Minimum-time Control of the Acrobot," in *Proc. of IEEE ICRA*, 1997, pp. 3281–3287.

[9] J. Yoshimoto *et al.*, "Application of Reinforcement Learning to Balancing of Acrobot," in *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, 1999, pp. 516–521.

[10] X. Xin and M. Kaneda, "A Robust Control Approach to the Swing up Control Problem for the Acrobot," in *Proc. of IEEE/RSJ IROS*, 2001, pp. 1650–1655.

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

[12] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[13] R. Ueda and T. Arai, "Creation and Compression of Global Control Policy for Swinging up Control of the Acrobot," in *Proc. of IROS*, 2006, pp. 2232–2237.

[14] M. W. Spong, "The swing up control problem for the Acrobot ," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 49–55, 1995.

[15] J. N. Tsitsiklis and B. V. Roy, "Feature-Based Methods for Large Scale Dynamic Programming," *Machine Learning*, vol. 22, pp. 59–94, 1996.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic ROBOTICS*. MIT Press, 2005.