論　文

強化学習を用いた高次元連続状態空間における系列運動学習
——起き上がり運動の獲得——

森本　淳[†,††]　銅谷　賢治[†††]

Learning Dynamic Motor Sequence in High-Dimensional State Space by Reinforcement Learning——Learning to Stand Up——

Jun MORIMOTO[†,††] and Kenji DOYA[†††]

あらまし　人間のように多自由度の身体をもつロボットが，試行錯誤的に目的とする運動を獲得することは，ロボット学習の分野において大きな課題である．本論文では強化学習を用いて，ロボットが高次元状態空間中における運動学習を行うための手法，主に評価関数の近似方法についての考察を行う．制御タスクとしては，人間の身体の形を単純化した3リンク2関節ロボットの起き上がり運動を扱う．強化学習の一手法である，連続時間，連続状態の Temporal Difference 学習則を，正規化ガウス関数を必要に応じて逐次配置する関数近似手法と組み合わせ，高次元連続状態空間中での起き上がり運動学習を行った．その結果，シミュレーション上で約2400回の試行後に起き上がり運動が獲得された．

キーワード　強化学習，Actor-critic 法，動径基底関数（RBF），正規化ガウス関数ネットワーク，起き上がりロボット

1. まえがき

近年，人間が制御則を作り込んでロボットを動かすことに関しては，高度な動作を行うロボットが出現している [1], [2]．しかし，将来人間と共存し，人間の代わりとなって働くようなロボットを実現するためには，固定された制御則を用いるだけでなく，動的に変化する環境の中で，ロボット自身が学習によって制御則を獲得することが要求される．そのような要求にこたえるため，ロボットに未知環境において行動を獲得させる手法として，強化学習が注目を集めている．強化学習とは，模範となる出力系列が与えられなくても，最終的に課題がどれだけ達せられたかという評価信号から望ましい制御則を発見する学習の枠組みである．

しかし，人間のように多自由度の身体をもったロボットの運動獲得を，従来の強化学習法を用いて行うことは一般に困難である．なぜなら，その身体の姿勢などを表現するための状態空間が連続で，かつ状態の次元が多いため，探索に要する時間が非常に大きくなってしまうからである．そこで，本論文では，強化学習における高次元連続状態空間の扱いについて検討する．

従来強化学習は，2次元格子状の迷路探索に代表されるような離散状態空間の課題に対して盛んに適用されてきた [3]〜[5]．また，連続状態をもつ制御対象の運動学習を扱った例として，連続状態を BOXES [6] や CMAC [7] を用いて離散状態に量子化して，倒立振子の安定化 [6] や2重振子の振り上げ [8] を行ったものがある．更に，連続状態空間を動径基底関数（radial basis function: RBF）ネットワークを用いて連続的に表現し，台車–振子系における振子の振り上げ，安定化に成功した例 [9] などがある．これらの課題では，状態空間を表現するための基底関数の数，配置，大きさなどは，実験者の先験的知識によりあらかじめ決定されていた．また，状態空間は2〜4次元と低次元なため，格子状に配置しても基底関数の数は十分少なく抑えられていた．

† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan
†† （株）ATR 人間情報通信研究所，京都府
ATR Human Information Processing Research Labs., Kyoto-fu, 619–0288 Japan
††† 科学技術振興事業団 ERATO，京都府
ERATO, Japan Science and Technology Corp., Kyoto-fu, 619–0288 Japan

しかし，未知環境下での探索や，高次元状態空間中の探索を行う場合，状態空間のどの範囲までを考慮すべきかが未知であったり，状態空間の可能なすべての範囲に十分な精度の基底関数をあらかじめ用意すると，その数が膨大になってしまうという問題（次元の呪い）が生じる．そこで本論文では，学習中に必要に応じて状態空間の表現を決定していく手法を用いる．

状態空間と動作をまず量子化するという枠組みでは，ロボットの探索から得られたセンサデータをもとに，超楕円体によってオフラインで離散状態空間の表現を決定する方法 [10] や，この手法をオンラインで適用可能にした方法 [11]，また，センサデータと得られる報酬をもとに局所線形モデルを構築することで，状態表現をオンラインで決定する方法などが提案されている [12].

一方，人間が行っているような滑らかな運動制御を実現するためには，連続状態空間において，連続動作出力を用いる必要があると考えられる [9], [13]．そこで本論文では，状態の表現方法として連続状態空間を用い，その上での評価関数や制御則の実現方法として，ニューラルネットワークの枠組みを用いる．

シグモイド関数を中間層に用いたネットワークでは，各パラメータの変化が出力に大域的な影響を及ぼすため，環境を探索しながらオンラインで状態空間を構築し運動学習を行う場合には適していない [14]．また，強化学習に用いた場合，学習が破綻する可能性がある [15]．そこで，RBF のように局所的な基底関数を用いることが考えられるが，基底関数がカバーしている範囲外の状態空間の領域では汎化が行われないため，高次元状態空間での学習には非常に多くの素子と学習サンプルが必要とされる．そこで本論文では，RBF と同様に局所的な近似を行うと同時に，基底関数がカバーしている範囲外の状態空間の領域も緩やかな外挿によって汎化する正規化ガウス関数（**3. 3** 参照）を用いることにする．

高次元状態空間での制御課題として，3 リンク 2 関節のロボット（図 1，表 1 参照）を用いた起き上がり運動，つまり，ロボットが床に倒れている状態から立ち上がるような運動の獲得を扱う．人間型ロボットの起き上がり動作を扱った研究として [16] がある．しかし, [16] で扱われているのは静的な起き上がり運動であり，実用的には人間のように動的な起き上がりで素早く起き上がることが望まれる．そこで，本論文では動的な起き上がりを運動課題とする．また，動的な運動の中でも，歩行運動のような定常的な運動に比べて，起き上がり運動のような過渡的な運動は，理論的に制御則を導いたり，人間が発見的に制御則を求めることは困難であり，強化学習を用いて制御則を獲得する必然性が大きい課題である．

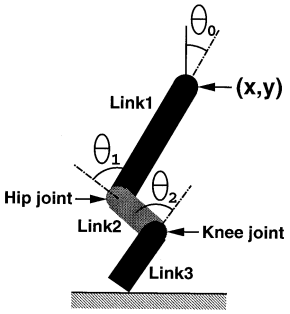


図 1　ロボットの形状．ロボットの関節可動範囲はそれぞれ次のようにした．$\theta_1 = 0$〜3.0 [rad]，$\theta_2 = -3.0$〜0 [rad]

Fig. 1　Robot configuration. The ranges of joint angles are $\theta_1 = 0$〜3.0 [rad], $\theta_2 = -3.0$〜0 [rad].

表 1　ロボットの物理パラメータ
Table 1　Physical parameters of the robot.

	Link1	Link2	Link3
長さ	1.25 m	0.5 m	0.5 m
幅	0.2 m	0.2 m	0.2 m
質量	20 kg	20 kg	20 kg

以下では，まず強化学習の基本的な説明を行い，その後に本論文における強化学習の具体的な実現方法，評価関数の近似方法を示す．最後に提案する手法を用いて，シミュレーション上でロボットが起き上がり運動の学習を行った結果を示す．

2. 強化学習

強化学習では，教師付き学習とは異なり，目標とする出力を学習者に教えることはしない．学習者の行動が良かったのか，悪かったのかを評価するだけで（行動に対して報酬や罰を与えるだけで）学習者自らが試行錯誤のうちにその目標出力を獲得するという学習を行う．具体的には，学習者は長期的に見て得られる報酬（累積報酬）が最大となるような行動をとるように学習する．

学習者を含めた系全体が確率過程でモデル化されるとき，累積報酬は期待値で表され，ある政策 μ のもとでの時刻 t における累積報酬の期待値 $V^{\mu}(t)$ は離散

時間系において式 (1) のように定義される．

$$V^{\mu}(t) = \mathrm{E}\left[\sum_{n=0}^{\infty} \gamma^n r(t+n)\right] \tag{1}$$

ただし，$r(t)$ は時刻 t における報酬（または罰）であり，γ は将来の報酬に対する割引率を示す．強化学習の基本的な戦略は，現在の状態 $\mathbf{x}(t)$ から今後得られる報酬の期待値 $V^{\mu}(t)$ を状態の評価関数 $V(\mathbf{x}(t))$ として同定し予測することである．これによって，後述のように各時点における行動の評価と計画が可能になる [17].

3. 連続時間・状態 TD 学習

本論文で扱うタスクのように，連続時間・連続状態の制御課題に対しては，連続系で定義された学習法を用いることにより，滑らかで効率良い制御が実現されることが期待される．そこで本論文では，連続時間・状態 TD 学習法 [9], [13] を用いて起き上がり運動の獲得を行う．

本研究の主題は，連続空間での強化学習であり，実際のシミュレーションやロボットのディジタル制御では，以下のアルゴリズムは有限時間刻み Δt で離散時間化され（付録 **3.** 参照），その場合には，離散時間の TD 学習 [18] と一致することが示されている [9], [13]. しかし連続時間での定式化をしておくことで，時間刻み Δt を制御の必要に応じて学習中に変更しても，評価関数自体は不変であるという利点をもつ．

3.1 連続時間・状態系での TD 学習

3.1.1 評価関数

連続時間・状態系のダイナミックスを

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

で表す．ただし，$\mathbf{x} \in X \subset \mathbf{R}^n$ は状態，$\mathbf{u} \in U \subset \mathbf{R}^m$ は制御入力を表す．

報酬は状態と制御入力の関数として

$$r(t) = r(\mathbf{x}(t), \mathbf{u}(t)) \tag{3}$$

と与えられるとする．ある制御則

$$\mathbf{u}(t) = \mu(\mathbf{x}(t)) \tag{4}$$

のもとで，状態 $\mathbf{x}(t)$ の評価関数を

$$V^{\mu}(\mathbf{x}(t)) = \int_t^{\infty} e^{-\frac{s-t}{\tau}} r(\mathbf{x}(s), \mathbf{u}(s)) ds \tag{5}$$

のように定義する．ここで τ は評価の時定数である．離散時間の場合の割引率 γ とは，$\gamma = 1 - \frac{\Delta t}{\tau}$ という関係にある（付録 **2.** 参照）．

3.1.2 TD 誤差

式 (5) によって定義した評価関数を，ニューラルネットワークのような関数近似法によって近似することを考える．

$$V^{\mu}(\mathbf{x}(t)) \simeq V(\mathbf{x}(t); \mathbf{w}) \tag{6}$$

ただし，$\mathbf{w}$ は関数近似のパラメータベクトルである．

TD 学習においては，評価関数が満たすべき局所的な拘束条件を用いることにより，評価関数の予測をオンラインで更新する．式 (5) の両辺を時間微分すると，

$$\frac{dV^{\mu}(\mathbf{x}(t))}{dt} = \frac{1}{\tau} V^{\mu}(\mathbf{x}(t)) - r(t) \tag{7}$$

という拘束条件が与えられる．$V(\mathbf{x}(t)) = V(\mathbf{x}(t); \mathbf{w})$ を評価関数 $V^{\mu}(\mathbf{x}(t))$ の予測値とする．予測が正しければ，$\dot{V}(\mathbf{x}(t)) = \frac{1}{\tau} V(\mathbf{x}(t)) - r(t)$ を満たす．予測が正しくない場合，下式に示した予測誤差を減らすように学習を行う．

$$\delta(t) \equiv r(t) - \frac{1}{\tau} V(\mathbf{x}(t)) + \frac{dV(\mathbf{x}(t))}{dt} \tag{8}$$

上式は連続時間系での TD 誤差である [9], [13]. この TD 誤差は，評価関数と **3.2** に示す Actor-critic 法の制御則の学習のために用いられる．

3.2 Actor-critic 法

本論文では，起き上がり運動の学習方法として，TD 学習の実現方法の一つである Actor-critic 法 [6] を用いる．

Actor-critic 法では，図 2 に示すような Actor と呼ばれる制御ネットワークと，Critic と呼ばれる評価ネットワークを用いる．それぞれのネットワークが以下に示すような学習を行うことで，Critic が評価値

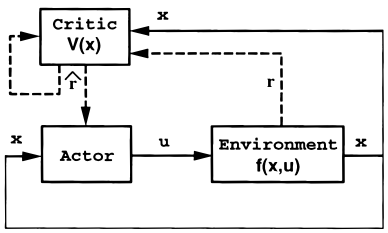


図 2 Actor-critic 構造
Fig. 2 Actor-critic architecture.

$V(\mathbf{x}(t))$ を正しく予測し，Actor が $V(\mathbf{x}(t))$ を最大にする制御出力を獲得する．

3.2.1 Critic における評価値の出力と学習

Critic は，式 (8) で示される予測誤差（TD 誤差）$\delta(t)$ を 0 にするように学習を行う．Critic のネットワークへの入力は現在の状態であり，出力は評価値 $V(\mathbf{x}(t))$ である．Critic に RBF のような基底関数の結合によるネットワークを用いるとすると，評価値の出力方法と学習手順は以下のようになる．

i） 現在の状態より評価値 $V(\mathbf{x}(t))$ を出力する．

$$V(\mathbf{x}(t)) = \sum_i v_i b_i(\mathbf{x}(t)) \tag{9}$$

ただし，$b_i()$ は基底関数であり，v_i は荷重を表す．

ii） 予測誤差（TD 誤差）$\delta(t)$ を計算する．

$$\delta(t) = r(t) - \frac{1}{\tau}V(\mathbf{x}(t)) + \frac{dV(\mathbf{x}(t))}{dt} \tag{10}$$

iii） 荷重を更新する．

$$\dot{v}_i = \alpha\delta(t)e_i(t) \tag{11}$$

ただし，α は学習係数，e_i は eligibility trace を表す（付録 **3.** 参照）．

iv） eligibility trace e_i を更新する．

$$\dot{e}_i(t) = -\frac{1}{\kappa}e_i(t) + b_i(\mathbf{x}(t)) \tag{12}$$

3.2.2 Actor における制御出力と学習

Actor は，より評価値 $V(\mathbf{x}(t))$ の大きい状態にたどりつくような学習を行い，それによって累積報酬の期待値を最大にする．Actor に RBF のような基底関数の結合によるネットワークを用いるとすると，制御出力の生成方法と学習手順は以下のようになる．

i） j 番目の制御出力を次のように与える [19].

$$u_j(t) = u_j^{\max} g\left(\sum_i w_{ij} b_i(\mathbf{x}(t)) + \sigma n_j(t)\right) + u_{bj} \tag{13}$$

ただし，$b_i()$ は基底関数であり，w_{ij} は荷重を表す．$u_j^{\max}$ は j 番目の制御出力の最大値であり，u_{bj} はバイアス出力を表す．また関数 $g()$ は制御出力を最大出力で飽和させるためのものであり，シグモイド関数を使う．$\sigma n_j(t)$ は，制御出力の探索のためのノイズであり，探索を十分行うために低周波成分を多く含むノイズを用いる [9], [13].

ii） 荷重を更新する．

$$\dot{w}_{ij} = \beta\delta(t)\sigma n_j(t) b_i(\mathbf{x}(t)) \tag{14}$$

ただし，β は学習係数．

また，上述の Actor-critic 法は連続時間系で記述されているが，実際に計算機上で実現するための方法は付録 **3.** に示した．

3.3 正規化ガウス関数ネットワークによる評価関数の近似

前述の Critic と Actor の評価関数及び制御関数を表現するために正規化ガウス関数ネットワーク（normalized Gaussian network: NGnet）を用いる．NGnet は図 3 に示したような 3 層のネットワークで，中間素子は正規化ガウス関数である [20].

与えられた n 次元入力ベクトル $\mathbf{x} = (x_1, \ldots, x_n)^T$ に対して，k 番目のユニットの活性化関数は，次のように計算される．

$$a_k(\mathbf{x}) = e^{-\frac{1}{2}||M_k(\mathbf{x}-\mathbf{c}_k)||^2} \tag{15}$$

ただし，$\mathbf{c}_k$ は活性化関数の中心であり，M_k は活性化関数の形状を決定する行列である．ここで，活性化関数 $a_k(\mathbf{x})$ を各点で総和が 1 になるように正規化したものを，基底関数 $b_k(\mathbf{x})$ とする．

$$b_k(\mathbf{x}) = \frac{a_k(\mathbf{x})}{\sum_{l=1}^{K} a_l(\mathbf{x})} \tag{16}$$

ただし，K は基底関数の数である．このように正規化を行うことで，図 4 に示すように，中心点 $\mathbf{c}_k$ が密に配置されている部分では $b_k(\mathbf{x})$ は局所的な基底関数となり，$\mathbf{c}_k$ の分布の端の部分では $b_k(\mathbf{x})$ はシグモイド関数のように大域的な基底関数になる．

ネットワークの出力は，基底関数と重みの内積により，

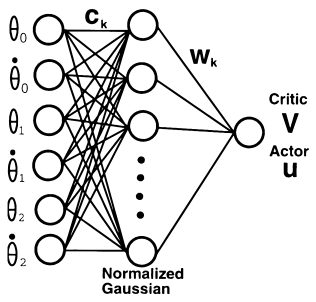
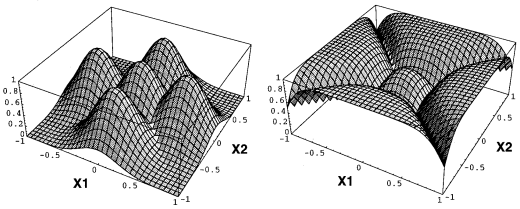


図 3 正規化ガウス関数ネットワーク
Fig. 3 Normalized Gaussian network.



(a) 活性化関数 $a_k(\mathbf{x})$　　(b) 基底関数 $b_k(\mathbf{x})$

図 4　ガウス活性化関数 $a_k(\mathbf{x})$ と対応する正規化ガウス関数 $b_k(\mathbf{x})$ の例．$\mathbf{x} \in \mathbf{R^2}$

Fig. 4　Examples of Gaussian activation functions and corresponding Normalized Gaussian basis functions. $\mathbf{x} \in \mathbf{R^2}$.

$$y(\mathbf{x}) = \sum_{k=1}^{K} w_k b_k(\mathbf{x}) \tag{17}$$

と与えられる．上述のシグモイド状の基底関数により，ネットワークは滑らかな外挿の機能をもつことになる．

学習は，与えられた目標出力 $\hat{y}(\mathbf{x})$ に対して，2 乗誤差を最小にするように重みを更新することにより行われる．

$$\Delta w_k = \eta(\hat{y}(\mathbf{x}) - y(\mathbf{x}))b_k(\mathbf{x}) \quad (k = 1, \ldots, K). \tag{18}$$

ただし，η は学習係数である．

3. 3. 1　Incremental NGnet

上述した正規化ガウス関数を学習中に逐次配置するネットワークをここでは，Incremental NGnet（INGnet）と呼ぶことにする．

INGnet では，誤差がある基準 e_{max} より大きく，すべての存在するユニットの活性度があるしきい値 a_{min} より小さければ，つまり，

$$|\hat{y}(\mathbf{x}) - y(\mathbf{x})| > e_{max} \quad \text{and} \quad \max_k a_k(\mathbf{x}) < a_{min} \tag{19}$$

のときに新しいユニットを配置する．新しいユニットは，$\mathbf{c}_k = \mathbf{x}$，$M_k = \mathrm{diag}(\mu_i)$ と $w_k = \hat{y}(\mathbf{x})$ で初期化する．ここで μ_i は活性化関数の半径の逆数である．また，$\mathbf{c}_k$，M_k を出力誤差のこう配法により更新することも可能であるが，オンライン学習に適用する場合，学習が不安定になる例が見られたため，本論文では $\mathbf{c}_k$，M_k は固定とした．

上述の手法と同様，正規化ガウス関数を基底関数にして，基底関数の数を学習中に増加させる手法を用いた研究として [21], [22] などがある．[21] では，新たな基底関数を追加する代わりに，既存の基底関数を分裂させるという形式で基底関数を増加させる．この手法では，粗い近似から必要に応じて細かい近似が行われるため，最終的に必要となる基底関数の数は少なくてすむ可能性がある．しかし，分割する方向を求めるため，基底関数の分散共分散行列の固有値，固有ベクトルを求める計算が必要になる．[22] では，基底関数群を確率モデルとして定式化し，オンラインの EM アルゴリズムを適用することで学習を高速に行う．これらに対して，本論文で用いる手法は，比較的実現が容易であり，また，基底関数を格子状に配置する場合や，RBF を基底関数として用いる場合との比較を明確に行うことが可能である．よって，NGnet の有効性を示すには適当であるといえる．

3. 3. 2　Actor-critic への適用

以下のように，上述の INGnet の学習を Actor-critic に適用する．

- Critic の学習

式 (18) において，$\hat{y}(\mathbf{x}(t)) = V^{\mu}(\mathbf{x}(t))$ となる．しかし，TD 誤差 $\delta(t)$ が生じた時刻に活性となっている基底関数に対して信用割当てを行うのではなく，過去に活性となった基底関数に対して信用割当てをを行うので，eligibility trace を用いて式 (11)，(12) のように Critic の INGnet の学習を行う．よってこの場合，式 (18) と式 (11) については，$\hat{y}(t) - y(t)$ と $\delta(t)$，b_i と e_i が対応しているといえる．

- Actor の学習

式 (14) と式 (18) の対応より，$\hat{y}(\mathbf{x}(t)) = y(\mathbf{x}(t)) + \delta(t)\sigma n_j(t)$ として Actor の INGnet の学習を行う．

4.　起き上がり運動学習

起き上がり運動学習のために，連続系の Actor-critic 法 [9], [13] を用いた．ロボット（図 1，表 1 参照）のシミュレーションプログラム（Boston Dynamics Inc. 製）は，リンクのダイナミックス計算，床との摩擦，衝突計算と，それぞれの関節の単純なサーボ系の計算を行う（摩擦，サーボ系の計算モデルを付録 **4.** に示した）．ロボットに送る制御出力は，腰と膝の目標関節角度を示す 2 次元ベクトル $(\theta_{1d}, \theta_{2d})$ とした．INGnet を，評価関数を近似する Critic と，二つの関節の非線形フィードバック制御を行う Actor の両方に用いた．ロボットは地面に固定されていないため，ロボットの状態を表すためには，頭の水平垂直位置座標，二つの

関節角，頭のピッチ角（傾き角）とそれらの時間微分で，延べ 10 次元の情報が必要になる．しかし，起き上がり運動においては，ロボットの身体の一部分が床に接地していると仮定すると，垂直位置の情報がなくとも姿勢が一意に決まる．また，水平方向の絶対位置情報は起き上がり運動には必要ではない．そこで，ネットワークへの入力として $(\theta_0, \dot{\theta}_0, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$ の 6 次元の状態を選んだ（図 3 参照）．ただし，θ_0 はロボットの頭のピッチ角，θ_1 は腰関節，θ_2 は膝関節を表す（図 1 参照）．

それぞれの試行は，ロボットが横に倒れている状態から始まり，試行がシミュレーション時間で 10 秒間続くか，または腰が一度地面を離れてから再び腰や膝を地面についたら（転倒したら）終了とした．更に，起立状態での安定化を目指すために，過度の跳躍（足の裏の高さ $> 0.2\,[m]$[(注 1)]）を禁止し，この場合は試行を終了した．報酬関数は，以下のようにロボットの頭の高さ y に応じて与えた．

$$r(y) = \begin{cases} \left(\dfrac{y}{l}\right) - 1 & \text{(試行中)}(l\text{: ロボットの全長}) \\ -1 & \text{(転倒時, 跳躍時)} \end{cases} \tag{20}$$

4.1 実験の終了条件

1 回の実験は以下の条件に基づいて終了とした．

- ロボットが 10 回の起き上がりに成功したとき．
- 4000 回試行を行っても，上記の条件を満たさないとき．

4.2 Actor-critic のパラメータ設定

式 (5) における報酬評価の時定数を $\tau = 0.5\,[\mathrm{s}]$，式 (A·7) における eligibility trace の時定数を $\kappa = 0.1\,[\mathrm{s}]$ とした．また，Critic と Actor の学習係数は以下のように試行回数 T に応じて増加させた．学習初期においては，Critic が過大な評価値を学習し，Actor がその評価値を信じて学習することがあるため，局所最大値に陥る可能性が高い．そこで，学習初期に学習係数を小さくしておくことで，局所最大値に陥る可能性を減らしている．

Critic の学習係数：$\alpha = \min[\alpha_c, \frac{\alpha_c}{2}(1 + \frac{T}{T_1})]$

Actor の学習係数：$\beta = \min[\beta_c, \frac{\beta_c}{2}(1 + \frac{T}{T_1})]$

ただし，$\alpha_c = 0.02$，$\beta_c = 0.1$，$T_1 = 1000$［回］とした．Critic，Actor それぞれの学習をともに $\Delta t = 0.01\,[\mathrm{s}]$ の周期で行った．Actor の出力関数として $g(x) = \frac{2}{\pi}\arctan(\frac{\pi}{2}x)$ を用い，出力の振幅を $u_j^{\max} = \pi\,[\mathrm{rad}]$，バイアスを（腰関節）$u_{b1} = \frac{2}{3}\pi\,[\mathrm{rad}]$,（膝関節）$u_{b2} = -\frac{2}{3}\pi\,[\mathrm{rad}]$ とした．また，式 (13) の探索のためのノイズ $n(t)$ にはガウスノイズを用い，ノイズの大きさ σ は下式のように評価関数 $V(t)$ に応じて決定した（評価値が高いときは探索を抑える）．

$$\sigma = \frac{\sigma_0}{\sqrt{\Delta t_{act}}} \min\left[1, \max\left[0, \frac{V_1 - V(t)}{V_1 - V_0}\right]\right] \tag{21}$$

ただし，$\sigma_0 = 0.6$ はノイズの大きさを示すパラメータであり，V_0 と V_1 はそれぞれ評価関数の予測される下限と上限に設定される．ここでは，式 (20) に示した報酬関数を考慮して，$V_0 = -1.0$，$V_1 = 0.0$ とした．また本実験では，十分に探索を行うために Actor が出力する目標関節角度 $(\theta_{1d}, \theta_{2d})$ の更新を系の時間刻み $\Delta t = 0.01\,[\mathrm{s}]$ より大きな時間刻み $\Delta t_{act} = 0.2$ で行った．よって，ある時刻 t_0 における Actor の出力 $u(t_0)$ は $t_0 \leqq t \leqq t_0 + \Delta t_{act}$ の間ホールドされる．そのため，ノイズ $n(t)$ は Δt_{act} ごとに呼ばれる離散時間関数となる．よってこの場合，Actor の出力は連続値であるが離散時間となる．

4.3 INGnet のパラメータ設定

式 (15) に示した活性化関数の半径の逆数を決める行列 M_k の各成分は次のように設定した．

$$M_k = \mathrm{diag}(2.0, 0.2, 2.0, 0.2, 2.0, 0.2) \quad [1/\mathrm{rad}],\ [\mathrm{s}/\mathrm{rad}]$$

また，式 (19) に示した活性度のしきい値は $a_{min} = 0.4$ とした．一方，式 (19) に示した誤差のしきい値は，学習初期においては荷重の学習が十分でないため，関数の近似誤差が多いことを考慮して試行回数に応じて $e_{max} = 0.5\exp(-\frac{T}{T_1})$ により減少させた．ただし，T は試行回数，$T_1 = 1000$［回］とした．更に，学習初期において TD 誤差が正確に得られないことを考慮して，新たに配置する基底関数の荷重は $w = 0.0$ で初期化した．また，学習前に配置する基底関数は初期状態に 1 個とした．

4.4 実験結果

上述の実験を 20 回行った結果，そのうちの 15 回で起き上がり運動の獲得に成功した，また，起き上がりに成功したときの平均試行回数は 2328 回（シミュレー

(注 1)：ロボットの状態は接地を仮定しているため，どの程度ロボットが地面から離れているかは Actor-critic の入力状態としては入ってこない．しかし，このような報酬を用いることで，結果としてロボットが地面から大きく離れるような制御出力系列を抑制することが期待できる．

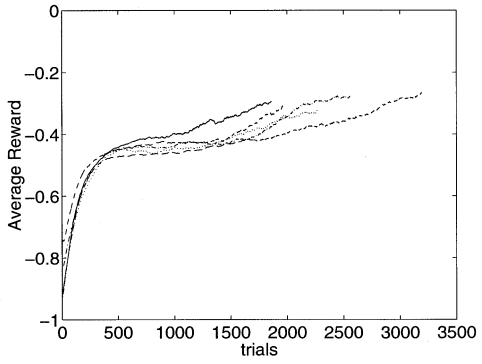


図 5　各試行における平均獲得報酬．15 回の成功実験のうち 5 回分を示している．起き上がり成功までの試行回数が少ない順に，1，4，8，11，15 番目の例を示した．また，学習の進捗の平均的な傾向がわかるように，それぞれのグラフには時定数 100 試行の低域フィルタを適用している．

Fig. 5　The average of reward acquired during trials in 5 out of 15 successful experiments: 1st, 4th, 8th, 11th, and 15th in the increasing order of trials required for successful stand-up. A low-pass filter with 100 trial time constant was used for smoothing.

ション時間：134 分実計算時間：96 分（mips R10000 195 MHz））であった．5 回の失敗の原因は，学習が局所最適解に陥ってしまったためと考えられる（ロボットは座り込んだまま立ち上がろうとしなくなる）．図 5 において，各試行における平均獲得報酬の学習に伴う変化の一例を示した．

学習に伴う運動パターンの変化の一例を図 6 (a)〜(d) に示した．(a) 第 1 試行目では，ランダムな動作で探索を行っている．(b) 第 100 試行目では，上半身の勢いを利用することを学習できているが，まだ足の裏に重心を乗せることはできていない．(c) 第 1000 試行目では，足の裏に重心を乗せ，伸び上がることを学習できているが，安定して立つことはできていない．(d) 第 2000 試行目では，起き上がり動作の後，安定して立つことを学習した．

学習により獲得された起き上がり運動の関節角とピッチ角の時間変化の一例を図 7 に示した．各関節角は初期位置と立ち上がり状態時の位置が同じであるが（$\theta_1, \theta_2 = 0.0\,[\mathrm{rad}]$），ピッチ角が目的とする位置にくるよう（$\theta_0 = \pi\,[\mathrm{rad}] \rightarrow 0.0\,[\mathrm{rad}]$）適切な中間軌道が生成されていることがわかる．また，報酬と評価値の時間変化の一例を図 8 に示した．運動開始から 0.7 秒付近で報酬がいったん下がるにもかかわらず，評価値は増加している．これは，将来的に得られる報酬を予測できているためといえる．理想的には評価値は単調増加すべきであるが，Critic の関数近似誤差のため，多少の増減が生じていると考えられる．また，学習の結果，得られる報酬の時系列がわかるので，式 (5) より評価値の理論値 V^{μ} を計算することができる．そこでこの V^{μ} を同様に図 8 に示した．V が V^{μ} を比較的よく近似していることがわかる．

図 9 (a) には，関節角 (θ_1, θ_2) 空間における獲得された評価関数の一例を示した．他の次元 $(\theta_0, \dot{\theta}_0, \dot{\theta}_1, \dot{\theta}_2)$ については，図 9 (b) に示したロボットの状態における値を用いた．太い実線は実際のロボットの運動方向を表し，評価値の上昇する方向に運動が行われていることがわかる．

5. 考　　察

5.1 格子状に基底関数を配置した場合（Grid NGnet）との比較

ここではあらかじめ基底関数を配置した場合と，INGnet により基底関数を必要に応じて逐次配置した場合との比較を行う．あらかじめ基底関数を状態空間上に配置する場合，ロボットが実際に行動するセンサ空間上に格子状に配置するのが適当であると考えられる（ここでは Grid NGnet（GNGnet）と呼ぶ）．そこで，以下のような条件で基底関数を配置した．

- **4.** の実験において配置された基底関数の中心位置の範囲を測定し，その範囲内に格子状に基底関数を配置する．
- INGnet の場合と同じ大きさの（同じ精度を実現する）活性化関数を用いる．

その結果，基底関数の個数はそれぞれの次元で $(\theta_0, \dot{\theta}_0, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$: $(9, 4, 7, 7, 6, 7)$，計 74088 個となった．一方，INGnet の場合，平均で（Critic: 240 Actor: 145）個(注 2)であった．GNGnet では，実際には探索されない状態空間の領域にも基底関数を配置することになり，その分 INGnet を用いる場合と比べて基底関数の数が大幅に増加している．そして，その基底関数の数の多さのために学習は困難となる．実際，74088 個の基底関数を配置した GNGnet を用いると，シミュレーション時間の 10 秒を計算するのに実計算時間で 68 秒（5 回の測定平均）かかった．一方，(Critic: 249 Actor: 165）個の基底関数を配置した INGnet を

(注 2)：15 回の起き上がり成功実験の平均．

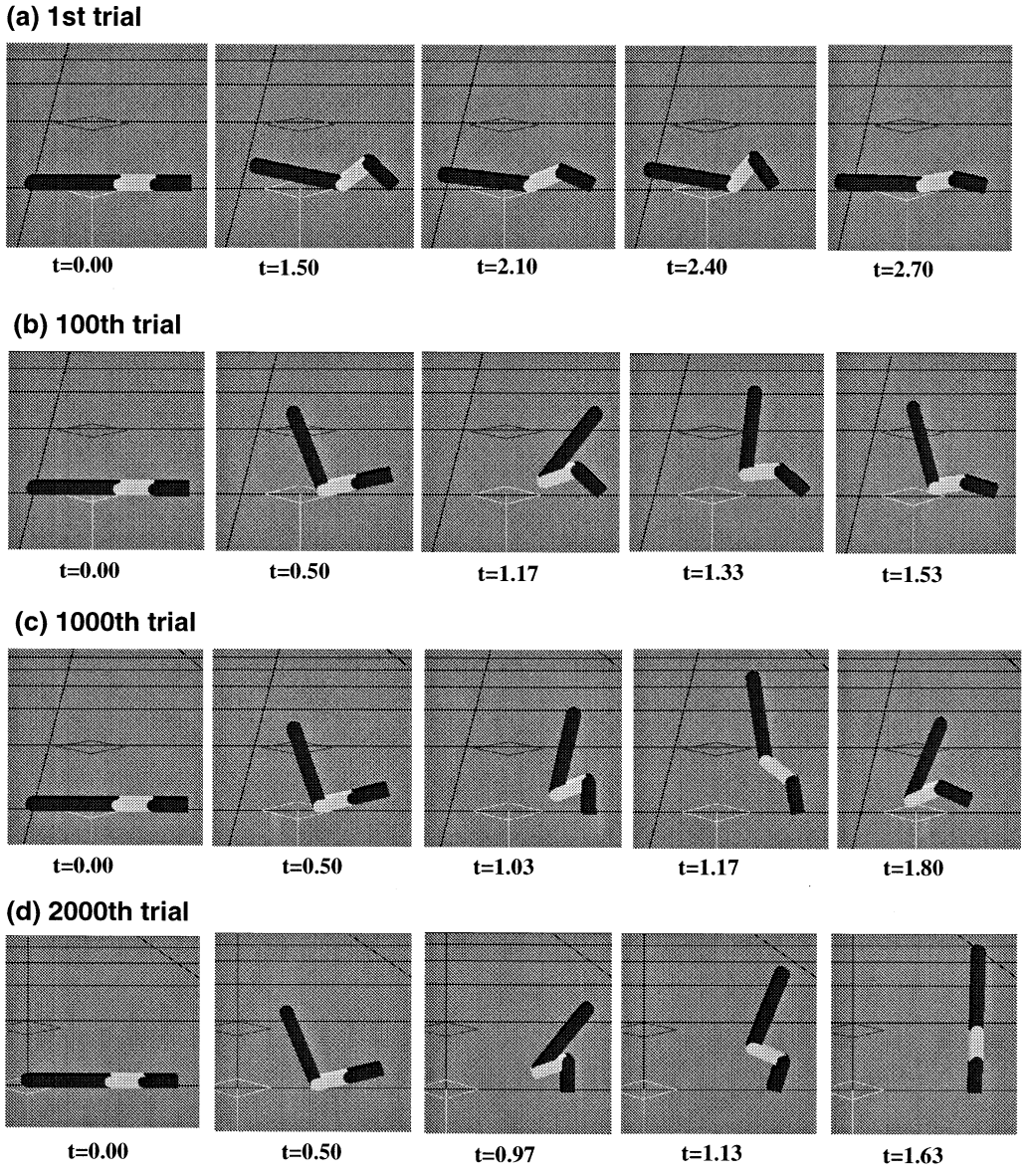


図 6　起き上がり運動系列：(a) 1 試行目 (b) 100 試行目 (c) 1000 試行目 (d) 2000 試行目（t：シミュレーション時間 [s]）

Fig. 6　Stand-up motion sequence: (a) 1st trial (b) 100th trial (c) 1000th trial (d) 2000th trial (t: simulation time [s])

用いた場合，同様の計算をするのに必要な時間は 18 秒であった（基底関数の数が平均値に近い実験を選んだ）．

更に，74088 個の基底関数を配置した GNGnet を使って起き上がり運動学習を行った．その結果，10000 試行後にも起き上がり運動を獲得することはできなかった．その理由としては，個々の基底関数のカバーする領域が狭いために，汎化特性が悪いということが考えられる．また，はじめから配置した基底関数の位置が，評価関数を近似するのに適当な位置に置かれて

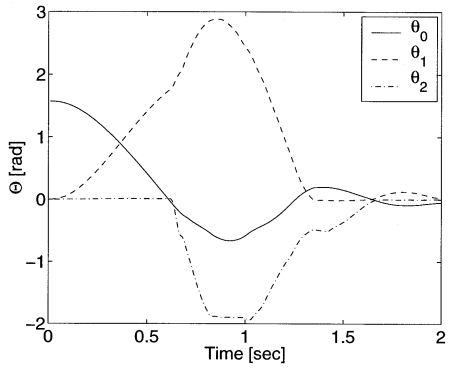


図 7　関節角，ピッチ角軌道
Fig. 7　Trajectories of joint and pitch angles.

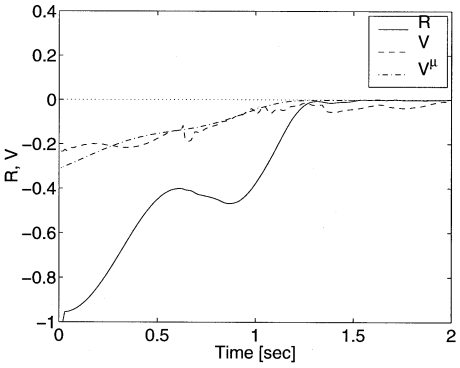


図 8　報酬（R），評価値（V），評価値の理論値（V^{μ}）
Fig. 8　Reward (R), acquired value (V), theoretical value (V^{μ}).

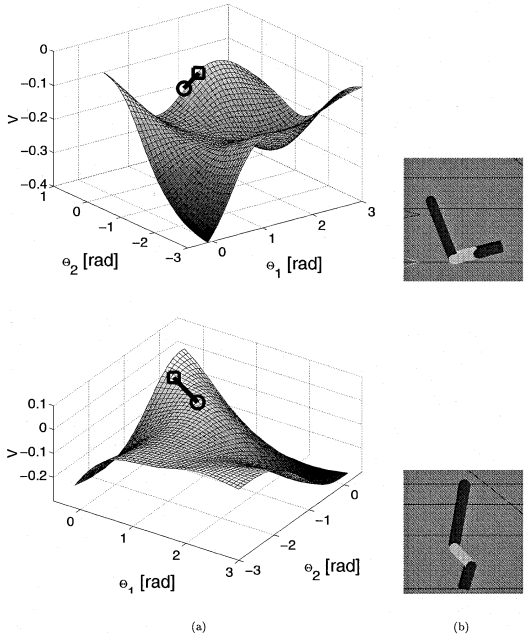


図 9　(a) 獲得された評価関数．図中 "○" で示した位置が，(b) に図示したロボットの関節角空間における位置である．更に，図中 "□" で示した位置がそこから 0.1 秒後の状態である．(上段)：上半身を起こす方向に評価関数が高くなっている．(下段)：膝，腰関節を伸ばす方向に評価関数が高くなっている．
Fig. 9　(a) Learned value functions. The position shown by "○" denotes the posture of the robot shown in (b). After 0.1 second, the robot moved from the position "○" to the position "□". (Top figure): The value function increases as the robot raises its upper body. (Bottom figure): The value function increases as the robot stretches its knee and hip.

いないことも考えられる．また，10000 試行を行うのに実計算時間で約 4 日間必要であった（10000 試行中の各試行における平均獲得報酬を図 10 に示した）．

5.2　RBF との比較

INGnet で起き上がり運動学習を行ったときと，学習率（$\alpha_c = 0.002$，$\beta_c = 0.01$）を除いて同じ学習パラメータを使って，正規化を行わない RBF を基底関数として逐次的に配置する方法により起き上がり運動学習を行った．しかし，20 回の実験で 1 度も起き上がり運動を獲得することはできなかった（実験の打切り条件は **4.1** と同様にした．つまり，4000 回の試行の間に起き上がり運動を獲得することができなかった）．また，基底関数の数は平均（Actor: 92, Critic: 147），偏差（Actor: 52, Critic 86）となった．正規化ガウス関数を用いる場合と比べて配置した基底関数の数の平均値は少なくなっているが，これはロボットが十分に探索を行わないまま（起き上がりに必要なだけの基底関数を配置しないまま）4000 回の試行が終了しているためである．更に配置した基底関数の数の偏差を比較すると，RBF を用いた方が正規化ガウス関数を用いた場合の偏差（Actor: 20, Critic: 19）に比べてかなり大きくなっている．つまり，RBF を用いた場合，制御関数と評価関数の近似が安定して行えていないということがわかる．これは，基底関数を逐次配置した場合に，正規化ガウス関数のように，外挿，汎化が行われないことにより，関数近似が適切に行われていないためと考えられる．図 11 に各試行における平均獲得報酬を示した．実験によってかなり平均獲得報酬のばらつきが見られることがわかる．

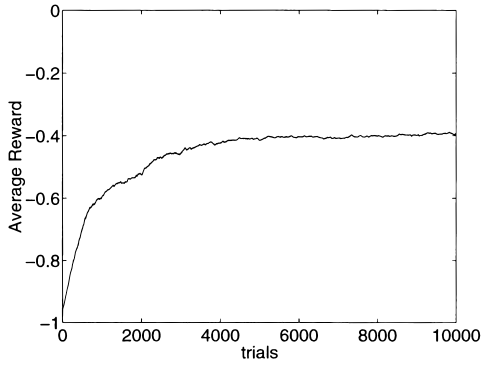


図 10　GNGnet を用いた場合の各試行における平均獲得報酬（時定数 100 試行の低域フィルタを適用している）．約 5000 試行の後平均獲得報酬の上昇は見られない．

Fig. 10　The average of reward acquired during trials using GNGnet (A low-pass filter with 100 trial time constant was used for smoothing). The average of reward acquired during trials did not increase after 5000 trials.

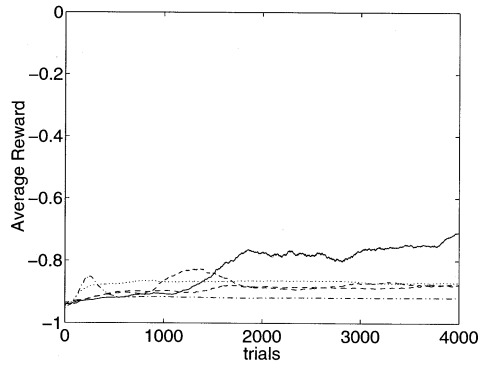


図 11　RBF を基底関数として用いた場合の各試行における平均獲得報酬（時定数 100 試行の低域フィルタを適用している）．配置した基底関数の数が少ない順に 1,5,10,15,20 番目の例を示した．

Fig. 11　The average of reward acquired during trials using RBF as a basis function (A low-pass filter with 100 trial time constant was used for smoothing): 1st, 5th, 10th, 15th, and 20th in the increasing order of basis functions generated in 4000 trials.

5.3　INGnet による基底関数の配置

INGnet により逐次的に基底関数を配置した場合，どの程度起き上がり軌道に沿って基底関数を配置しているかを確かめるために，基底関数の位置と起き上がり軌道との距離（1 基底関数当りの平均）を計算し，GNGnet を用いた場合との比較を行った．その結果，

表 2　各主成分の寄与率と第 1,2 主成分の係数ベクトル
Table 2　Contribution ratio of each principle component, and coefficient vectors of 1st and 2nd principle components.

	寄与率
第 1 主成分	31.85
第 2 主成分	22.13
第 3 主成分	19.38
第 4 主成分	15.86
第 5 主成分	8.90
第 6 主成分	1.88

	θ_0	$\dot{\theta}_0$	θ_1	$\dot{\theta}_1$	θ_2	$\dot{\theta}_2$
第 1 主成分	0.06	−0.58	−0.03	0.68	0.32	−0.31
第 2 主成分	−0.18	−0.07	0.70	0.15	−0.61	−0.26

GNGnet を用いた場合は 3.75，INGnet を用いた場合は 2.55 となった(注 3)．よって，INGnet を用いた場合は，格子状に配置した場合に比べて，起き上がり軌道に近いところに基底関数を配置していることがわかる．つまり，タスク達成のために用いる基底関数の無駄が少なくなっている．

また，上述の結果より，実際に基底関数が配置されている空間の次元は，6 次元の入力空間より縮退している可能性が考えられる．そこで，起き上がり成功実験の一例によって用いられた基底関数の中心位置に対して主成分分析を行い，それぞれの主成分の寄与率と第 1，2 主成分の係数ベクトルを表 2 に示した．

表 2 より，第 1 主成分はピッチ角速度（$\dot{\theta}_0$）と腰関節角速度（$\dot{\theta}_1$），第 2 主成分は腰関節（θ_1）と膝関節（θ_2）の変化を主に表していることがわかる．第 5，6 主成分の寄与率が少ないことより，入力次元に比べて基底関数の中心位置が張る空間の次元は縮退しているといえる．

6. むすび

本論文では，強化学習を用いて，高次元状態空間中でロボットが運動獲得を行うための手法を検討した．運動の例としては，特に非定常な運動に着目し，非線形性の強い力学系をもつ 3 リンク 2 関節ロボットの起き上がり運動を取り上げた．

連続系 Actor-Critic における制御関数と評価関数のための関数近似ネットワークの基底関数に，局所的に関数近似を行うと同時に緩やかな外挿も行うことが

(注 3)：15 回の起き上がり成功実験の平均．それぞれの距離の値は基底関数間の最小距離（$d_{min} = 0.48$ [rad], 4.8 [rad/s]）で正規化しているため無次元量となっている．また，d_{min} は活性度のしきい値 a_{min} より得られる．

できる正規化ガウス関数ネットワーク（NGnet）を用い，しかもその基底関数を必要に応じて配置する方式，INGnet を用いることにより，起き上がり運動を獲得することができた．

近年，強化学習を行う際に効率的に評価関数の近似を行う手法が提案されている [21], [22]．今後，本論文で扱ったような，高次元状態空間中における運動制御学習に対して，これらの手法が適用可能かどうかを検討する．

また，今回は冗長性をもつ状態空間表現から学習に本質的に重要な入力次元の選択を人間が行ったが，自動的に次元選択を行う手法についても今後検討する予定である．

文　献

[1] 広瀬真人，竹中　透，五味　洋，小澤信明，“人間型ロボット，” 日本ロボット学会誌，vol.15, no.7, pp.23–25, 1997.

[2] J. Yamaguchi, S. Inoue, D. Nishino, and A. Takanishi, “Development of a bipedal humanoid robot having antagonistic driven joints and three DOF trunk,” Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol.1, pp.96–101, 1998.

[3] L.-J. Lin, “Self-improving reactive agents based on reinforcement learning, planning and teaching,” Machine Learning, vol.8, pp.293–321, 1992.

[4] R.S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” Proceedings of the Seventh International Conference on Machine Learning, pp.216–224, Morgan Kaufmann, 1990.

[5] M. Wiering and J. Schmidhuber, “HQ-learning,” Adaptive Behavior, vol.6, no.2, pp.219–246, 1997.

[6] A.G. Barto, R.S. Sutton, and C.W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” IEEE Trans. Systems, Man, & Cybernetics, vol.13, pp.834–846, 1983.

[7] J.S. Albus, Brain, Behavior, and Robotics, Byte Books, 1981.

[8] R.S. Sutton, “Generalization in reinforcement learning: Successful examples using sparse coarse coding,” in Advances in Neural Information Processing Systems 8, eds. D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, pp.1038–1044, MIT Press, Cambridge, MA, USA, 1996.

[9] K. Doya, Reinforcement learning in continuous time and space, Neural Computation, vol.12, pp.243–269, 1999.

[10] 浅田　稔，野田彰一，細田　耕，“ロボットの行動獲得のための状態空間の自律的構成，” 日本ロボット学会誌，vol.15, no.6, pp.76–82, 1997.

[11] A. Ueno, K. Hori, and S. Nakasuka, “Simultaneous learning of situation classification based on rewards and behavior selection based on the situation,” Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol.3, pp.1510–1517, 1996.

[12] 高橋泰岳，浅田　稔，“実ロボットによる行動学習のための状態空間の漸次的構成，” 日本ロボット学会誌，vol.17, no.1, pp.118–124, 1999.

[13] K. Doya, “Temporal difference learning in continuous time and space,” in Advances in Neural Information Processing Systems 8, eds. D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, pp.1073–1079, MIT Press, Cambridge, MA, 1996.

[14] S. Schaal and C.G. Atkeson, “From isolation to cooperation: An alternative view of a system of experts,” in Advances in Neural Information Processing Systems 8, eds. D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, pp.605–611, MIT Press, Cambridge, MA, USA, 1996.

[15] J.A. Boyan and A.W. Moore, “Generalization in reinforcement learning: Safely approximating the value function,” in Advances in Neural Information Processing Systems 7, eds. G. Tesauro, D.S. Touretzky, and T.K. Leen, pp.369–376, MIT Press, Cambridge, MA, USA, 1995.

[16] 金広文男，稲葉雅幸，井上博允，“倒れても起き上がることのできる人間型ロボット，” 第 13 回日本ロボット学会学術講演会予稿集，pp.195–196, 1995.

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

[18] R.S. Sutton, “Learning to predict by the methods of temporal difference,” Machine Learning, vol.3, pp.9–44, 1988.

[19] V. Gullapalli, “A stochastic reinforcement learning algorithm for learning real-valued functions,” Neural Networks, vol.3, pp.671–192, 1990.

[20] J. Moody and C.J. Darken, “Fast learning in networks of locally-tuned processing units,” Neural Computation, vol.1, pp.281–294, 1989.

[21] 鮫島和行，大森隆司，“強化学習における適応的状態空間構成法，” 日本神経回路学会誌，vol.6, no.3, pp.144–154, 1999.

[22] 石井　信，佐藤雅昭，“正規化ガウス関数ネットワーク，Mixture of experts と EM アルゴリズム，” 日本神経回路学会誌，vol.6, no.1, pp.30–40, 1999.

付　録

1. 連続時間 TD（λ）学習

ここでは，本論文で用いた連続時間で定義された TD（λ）学習 [9], [13] について説明する．

ここでは，ある試行における軌道 $\mathbf{x}(t)$，$\mathbf{u}(t)$ を固定し，評価関数の推定値 $V(\mathbf{x}(t))$，その真の値 $V^{\mu}(\mathbf{x}(t))$，報酬 $r(\mathbf{x}(t), \mathbf{u}(t))$ を時間の関数 $V(t)$，$V^{\mu}(t)$，$r(t)$ と

表記する．評価関数 V^μ の積分形式の定義式 (5) から微分の条件式 (7) を導いた逆の変換を，TD 誤差の定義式 (8) から得られる式

$$\frac{dV(t)}{dt} = \frac{1}{\tau}V(t) - (r(t) - \delta(t)) \tag{A·1}$$

に対して施すことにより，

$$V(t) = \int_t^\infty e^{-\frac{s-t}{\tau}}(r(s) - \delta(s))ds$$

$$V^\mu(t) - V(t) = \int_t^\infty e^{-\frac{s-t}{\tau}}\delta(s)ds \tag{A·2}$$

が得られる．評価関数の推定値 $V(t)$ をその真の値 V^μ に近づけるための学習は $V(t)$ のパラメータを w_i とし，$\Delta w_i = \int_0^\infty \dot{w}_i\, dt$ とすると，

$$\begin{aligned}\Delta w_i &\propto -\int_0^\infty \frac{\partial \frac{1}{2}(V^\mu(t) - V(t))^2}{\partial w_i}dt \\ &= -\int_0^\infty -(V^\mu(t) - V(t))\frac{\partial V(t)}{\partial w_i}dt \\ &= \int_0^\infty \int_t^\infty e^{-\frac{s-t}{\tau}}\delta(s)ds\frac{\partial V(t)}{\partial w_i}dt \end{aligned} \tag{A·3}$$

で与えられる．ここで，積分順序を入れ替えることにより，

$$\Delta w_i \propto \int_0^\infty \int_0^s e^{-\frac{s-t}{\tau}}\frac{\partial V(t)}{\partial w_i}dt\ \delta(s)ds \tag{A·4}$$

が得られる．ここで，

$$e_i(s) = \int_0^s e^{-\frac{s-t}{\tau}}\frac{\partial V(t)}{\partial w_i}dt \tag{A·5}$$

とおくことにより，オンラインの学習則

$$\dot{w}_i = \eta\delta(t)e_i(t) \tag{A·6}$$

が得られる．一般的な更新則は，新たな時定数 $\kappa \leqq \tau$ を用いることで求められる．ここで，時定数 κ を用いて式 (A·5) を次式のように表す．

$$e_i(t) = \int_0^t e^{-\frac{t-s}{\kappa}}\frac{\partial V(s)}{\partial w_i}ds \tag{A·7}$$

上式の両辺を微分することで，$e_i(t)$ の更新則が得られる．

$$\dot{e}_i(t) = -\frac{1}{\kappa}e_i(t) + \frac{\partial V(t)}{\partial w_i} \tag{A·8}$$

$e_i(t)$ は連続時間 TD 学習における eligibility trace [6] といえる．また，$\lambda = \frac{1-\Delta t/k}{1-\Delta t/\tau}$ とし Δt で離散化したとき，離散時間 TD（λ）[18] 学習と一致する（付録 **2.** 参照）．

2. 離散時間 TD 学習と連続時間 TD 学習の対応

2.1 TD 誤差の定義

連続時間 TD 誤差は式 (8) で表される．ここで，$\dot{V}(t) = (V(t) - V(t-\Delta t))/\Delta t$ と近似し，式 (8) に代入すると次式のようになる．

$$\delta(t) = r(t) + \frac{1}{\Delta t}\left[\left(1 - \frac{\Delta t}{\tau}\right)V(t) - V(t-\Delta t)\right]. \tag{A·9}$$

一方，離散時間の TD 誤差は下式のように表される [18].

$$\delta(t) = r(t) + \gamma V(t) - V(t-\Delta t) \tag{A·10}$$

式 (A·9) を式 (A·10) と比較すれば，割引率 $\gamma = 1 - \frac{\Delta t}{\tau}$ とするとスケーリング定数 $\frac{1}{\Delta t}$ を除いて一致することがわかる．

2.2 評価値の更新

連続時間 TD 学習における評価値の更新則は式 (A·6) と式 (A·8) で表される．そこで，$\dot{e}_i(t) = \frac{e_i(t+\Delta t) - e_i(t)}{\Delta t}$，$\dot{w}_i = \frac{\Delta w_i}{\Delta t}$ で近似すると次式のようになる．

$$e_i(t+\Delta t) = \left(1 - \frac{\Delta t}{\kappa}\right)e_i(t) + \frac{\partial V(t)}{\partial w_i}\Delta t \tag{A·11}$$

$$\Delta w_i = \eta\delta(t)e_i(t)\Delta t \tag{A·12}$$

一方，離散時間の評価値の更新則は次式のようになる [17].

$$e_i(t+\Delta t) = \gamma\lambda e_i(t) + \frac{\partial V(t)}{\partial w_i} \tag{A·13}$$

$$\Delta w_i = \alpha\delta(t)e_i(t) \tag{A·14}$$

よって，式 (A·11) と式 (A·13) を比較すれば，$\lambda = \frac{1-\Delta t/k}{1-\Delta t/\tau}$ とするとスケーリング定数 Δt を除いて一致する．また，式 (A·12) と式 (A·14) を比較すれば，$\alpha = \eta$ とするとスケーリング定数 Δt を除いて一致する．

3. 連続時間 TD 学習の計算機上での実現

計算機上で連続時間 TD 学習を実現するときに問題となるのは，連続時間で定義された TD 誤差を求める際，評価値の時間微分が必要となることである．ここで式 (A·8) について考えると，この式では，e_i は V を荷重 w_i で微分したものに時定数 κ で時間方向に荷重平均したものとなっている．言い換えれば，e_i は V を時定数 κ で時間方向に荷重平均し（ここで

は $\bar{V}$ で表すことにする）荷重 w_i で微分したものと考えることができる（つまり，$e_i = \frac{\partial \bar{V}}{\partial w_i}$ となる）．ここで，$\dot{V} = \frac{V-\bar{V}}{\kappa}$ とし，式 (8) に代入すると $\delta(t) = r(t) - \frac{1}{\tau}V(t) + \frac{V(t)-\bar{V}(t)}{\kappa}$ となる．更に，TD 誤差が減少するようにネットワークの荷重を更新することを考えると次式のようになる．

$$\begin{aligned}\dot{w}_i &= -\nu\frac{1}{2}\frac{\partial\delta^2(t)}{\partial w_i} \\ &= -\nu\delta(t)\frac{1}{\kappa}\left(\left(1-\frac{\kappa}{\tau}\right)\frac{\partial V(t)}{\partial w_i} - \frac{\partial\bar{V}(t)}{\partial w_i}\right)\end{aligned} \tag{A·15}$$

ここでは，過去の評価値の情報を含んだ滑らかな評価関数である $\bar{V}(t)$ に関してのみ TD 誤差を求めることが適当であるといえるので，ネットワークの荷重の更新は次式のようになり，式 (A·6) と一致する．

$$\dot{w}_i = \eta\delta(t)\frac{\partial\bar{V}(t)}{\partial w_i} \tag{A·16}$$

ただし，$\eta = \nu\frac{1}{\kappa}$ とした．つまり，$\dot{V} = \frac{V-\bar{V}}{\kappa}$ として $\dot{V}$ を求めることが適当であるといえる．以上を含めて計算機上での学習アルゴリズムを以下に示す．

（1） $\bar{V}(0) = V(0)$ で初期化．

（2）（a）〜（g）を繰り返す．

（a） 式 (13) より状態 $\mathbf{x}(t)$ における制御出力 $u(t)$ を決定．また，状態 $\mathbf{x}(t)$ における Critic の基底関数の出力を $b^C(t)$，Actor の基底関数の出力を $b^A(t)$ とする．

（b） 式 (A·11) のように，基底関数の eligibility trace を更新．

$$e_i(t+\Delta t) = \left(1-\frac{\Delta t}{\kappa}\right)e_i(t) + b^C{}_i(t)\Delta t \tag{A·17}$$

（c） $u(t)$ を用いてロボットのダイナミックスを計算する．時刻は t から $t+\Delta t$ となり，状態は $\mathbf{x}(t+\Delta t)$ となる．また，環境より $r(t+\Delta t)$ が得られる．

（d） 次式のように TD 誤差 $\delta(t+\Delta t)$ を計算する．

$$\begin{aligned}\delta(t+\Delta t) = r(t+\Delta t) &- \frac{1}{\tau}V(\mathbf{x}(t+\Delta t)) \\ &+ \frac{V(\mathbf{x}(t+\Delta t)) - \bar{V}(t)}{\kappa}\end{aligned} \tag{A·18}$$

（e） 式 (11) のように，Critic を更新

$$v_i(t+\Delta t) = v_i(t) + \alpha\delta(t+\Delta t)e_i(t+\Delta t) \tag{A·19}$$

表 A·1 床摩擦，サーボゲイン
Table A·1 Ground friction and servo gain.

床摩擦	$k_x[\frac{N}{m}]$	$b_x[\frac{N\cdot s}{m}]$
Link3（足の裏）	10000	2000
その他	500	200

サーボゲイン	$k_\tau[\frac{N\cdot m}{rad}]$	$b_\tau[\frac{N\cdot m\cdot s}{rad}]$
腰関節	120	12
膝関節	300	30

（f） 式 (14) のように，Actor を更新

$$w_i(t+\Delta t) = w_i(t) + \beta\delta(t+\Delta t)\sigma n_j(t)b^A{}_i(t) \tag{A·20}$$

（g） $\bar{V}$ の更新

$$\bar{V}(t+\Delta t) = \left(1-\frac{\Delta t}{\kappa}\right)\bar{V}(t) + V(\mathbf{x}(t+\Delta t))\Delta t \tag{A·21}$$

4. ロボットのシミュレーションプログラム（Boston Dynamics Inc. 製）の計算モデル

［摩擦計算］

リンクの初期接地点（リンクが床と接していない状態から初めて接地した点）x_{init}，現在のリンク接地点 x_{gc}，現在のリンク接地点の速度 $\dot{x}_{gc}$，位置ゲイン k_x，速度ゲイン b_x とすると，生じる摩擦力 f_x は，

$$f_x = k_x(x_{init} - x_{gc}) - b_x\dot{x}_{gc} \tag{A·22}$$

で与えられる．

［サーボ系］

現在の角度 θ，現在の角速度 $\dot{\theta}$，目標角度 θ_d，位置ゲイン k_τ，速度ゲイン b_τ とすると，出力トルク τ は，

$$\tau = k_\tau(\theta_d - \theta) - b_\tau\dot{\theta} \tag{A·23}$$

で与えられる．また本実験においては，シミュレーション時間で 1 kHz のサーボを行っている．

論文中の実験では，それぞれ表 A·1 に示した値を用いた．

（平成 11 年 3 月 4 日受付，6 月 10 日再受付）