博士論文

強化学習への事前知識の組み込みと
システム制御への応用

東京大学大学院
工学系研究科 航空宇宙工学専攻
37-167060 青柳 祐基
指導教官 土屋 武司 教授

Table of Contents

List of Figures

List of Tables

List of Propositions, Lemmas and Theorems

List of Definitions

List of Algorithms

Chapter 1　序論

1.1　背景

近年の機械学習分野の研究開発の進歩と社会実装の進展は目覚ましく，現在は第 3 次 AI ブームの最中にあるといわれている．特に強化学習については，DQN [35,36] が Atari 2600 において人間のエキスパートを超える性能を示して以来，日進月歩で研究が進んでいる．

強化学習とは，環境 (状態遷移および報酬) に関する事前知識を用いずに，行動と観測のみから Bellman 方程式を on-line で解くアルゴリズムの総称であり，最適制御問題の数値解法の一種であるといえる．しかしながら，高い自律性を持ち，事前知識を必要としないが故の汎用性を長所とする反面，探索の初期に不安定な振る舞いをする，安定性の保証が難しいといった欠点により，制御問題への応用が進んでいるとは言い難い状況にある．

古典的な最適制御と強化学習は，Bellman 方程式を解くという点では目的を同一とするものの，その思想的・歴史的背景の違い故に，そもそも問題の前提が大きく異なっている．例として，それぞれの代表的な手法である LQR と DQN の比較を Table 1.1 に示す．当然ながら中には確率論的

Table 1.1: 最適制御と強化学習の前提の比較

	最適制御 (LQR)	強化学習 (DQN)
状態空間	連続	連続
行動空間	連続	離散
環境	決定論的かつ既知	確率論的かつ未知

な報酬を扱える最適制御の手法などの例外も存在するが，多くは Table 1.1 のような前提に立脚する．環境に対する前提の差異に着目すると，“決定論的かつ既知” は “確率論的かつ未知” の極めて特殊な場合であるから，強化学習の方が圧倒的に広い領域の問題を対象としていることが分かる．これは元来，制御工学の対象が物理的現象であり，報酬は設計者が定義するものである一方で，強化学習は行動心理学や神経科学等の生物の意思決定論に端を発することに因る．さらに，現在では連続な行動空間を扱える強化学習アルゴリズムも登場している [16,32,46] ことから，強化学習が対象とする問題領域は，古典的な最適制御が対象とする問題領域を完全に内包するといえる．

探索における No Free Lunch 定理 [57,58] に拠れば，あるゆる数理計画問題に対して性能の良い汎用アルゴリズムは理論上存在せず，対象とする問題領域の広さとアルゴリズムの性能は trade-off の関係にあるとされる．したがって，強化学習についてもその前提に何らかの制約を加え問題領域を限定することによって，探索の初期に不安定な振る舞いをするといった欠点を改善すると共に，

例えば耐故障制御のような，環境に関する事前知識が未知または不完全な状況における制御という新たな最適制御の枠組みとしても応用が期待される．

1.2　本論文の目的

本研究の第 1 の目的は，強化学習における事前知識とアルゴリズムの性能の関係について考察する，すなわち強化学習における No Free Lunch 定理について考察することである．第 2 の目的は，実際に強化学習に事前知識を組み込む (= 問題領域を限定する) ことにより，システム制御に特化した強化学習アルゴリズムを構築することである．そのためのアプローチとして，事前知識の多寡に応じて次の 2 通りについて検討する．

1. 環境に関する知識の組み込み
   制御問題においては遷移関数がある程度既知であり，報酬は設計者が定義する場合が多い．本論文では，これらを環境についての事前知識と呼ぶことにし，対象の近似モデルが既知かつ報酬が二次形式で表される場合，すなわち環境についての事前知識が (不完全であっても) 利用可能な場合に，Riccati 方程式の解を利用した学習効率の改善を試みる．

2. ドメイン知識の利用
   一般にドメインとは一定の広さを持つ問題領域を指すが，制御問題の多くにみられる，状態遷移が決定論的であるという前提そのものも，一種のドメイン知識であると考えられる．model-based 強化学習の代表的な枠組みである Dyna と，これらの仮定を組み合わせることで，決定論的環境に特化した手法を提案する．

1.3　本論文の構成

第 2 章では，最適制御問題の数値解法における強化学習の位置付けを定義した上で，強化学習に関する基礎的な事項である Markov 決定過程および Bellman 方程式とその諸性質，Q-learning 等の古典的な手法についてまとめる．また，深層強化学習や model-based 強化学習，安全な強化学習等の，強化学習の発展的な概念に関する近年の研究についても言及する．一部の内容は Sutton and Barto[50] や Puterman[41]，Bertsekas[6] 等の著名な教科書と重複するが，本論文では記号や名称を統一した一貫した内容とするとともに，Appendix にはより細かな証明を記した．

第 3 章では，Wolpert and Macready[57,58] による探索における No Free Lunch 定理とその解釈，さらには定理の拡張についての先行研究について述べる．これについても理解を深めるために，Appendix に詳細な証明を記した．

第 4 章では，探索における No Free Lunch 定理の一つの拡張として，強化学習における No Free Lunch 定理を提案し，強化学習の性能を向上するには何らかの事前知識を組み込むことにより，問題領域を限定する他ないと結論づけた．これは第 2 の目的である強化学習への事前知識の組み込みを支持する結果である．また，強化学習への事前知識の組み込み手法として，第 1 に “環境に関する知識の組み込み” として，対象の近似モデルが既知かつ報酬が二次形式で表されるという前提，すなわち環境についての不完全な事前知識が利用可能な場合に，Riccati 方程式の解を利用した状

態行動価値関数の初期化による学習効率の改善手法を提案する。第 2 に “ドメイン知識の利用” として，制御問題の多くにみられる，状態遷移が決定論的であるという前提そのものを一種のドメイン知識であると見做し，model-based 強化学習の代表的な枠組みである Dyna と，深層強化学習アルゴリズムである DQN および DDPG を組み合わせた手法を提案する．

第 5 章では，数値計算例として，“環境に関する事前知識の組み込みに” ついては Q-learning による double integrator の制御，ドメイン知識の利用については，深層強化学習アルゴリズムである DQN および DDPG による double integrator の制御ならびに，より応用的な例として，DDPG による宇宙機および航空機の姿勢制御に対する提案手法の有効性を検討する．

第 6 章では，結論として全体のまとめと今後の展望について述べる．

Chapter 2　強化学習

本章では，本論文の主要な興味の対象である強化学習の基礎理論について述べる．強化学習は，生物の行動選択の数理モデルとしての側面と，機械学習の 1 分野としての側面と，最適制御問題の数値解法としての側面を持ち，一言で定義するのは困難である．機械学習の 1 分野としての強化学習は，教師あり学習 (supervised learning) および教師なし学習 (unsupervised learning) と共に機械学習の 3 つの基本的な枠組みを構成し，この文脈においては “自ら教師データにアクセスし，教師ラベルを取得する入力の選択を最適化する” 手法であるといえる．ところで，強化という用語は本来は行動主義心理学に由来するもので，ある行動によって環境に好ましい変化がもたらされると，その行動の生起頻度が高まることを指す．したがって，心理学や生理学の文脈では強化学習は “生物の行動選択 (よりミクロに見れば意思決定における脳の作用) を説明するための数理モデル” であるといえる．

強化学習について上述の 2 つの観点を提示したが，我々が機械学習を通して解きたい問題も，生物の意思決定そのものも，結局は何らかの最適化問題に帰着し，かつ現実の問題の多くが変化を伴う動的な問題である．そのような文脈では，強化学習は最適制御問題の数値解法の 1 種であるともいえる．本章では，最適制御問題の数値解法としての強化学習を理解する上での基礎的な事項である Markov 決定過程および Bellman 方程式とその性質について定義した上で，深層強化学習や model-based 強化学習，安全な強化学習などの，より発展的な内容についても述べる．

2.1　Markov 決定過程

本節では，離散時間の確率的最適制御問題の枠組みである Markov 決定過程 (Markov Decision Process, MDP) および，最適性の条件である Bellman 方程式とその性質について述べる．MDP は，その名称にもあるように本来は意思決定問題[*1]の枠組みである．MDP では，意思決定の主体を agent と呼び，agent が行動を取ると，意思決定の客体である環境の状態が遷移し，agent は次の状態と，状態および行動の良さの指標としての報酬を観測する．

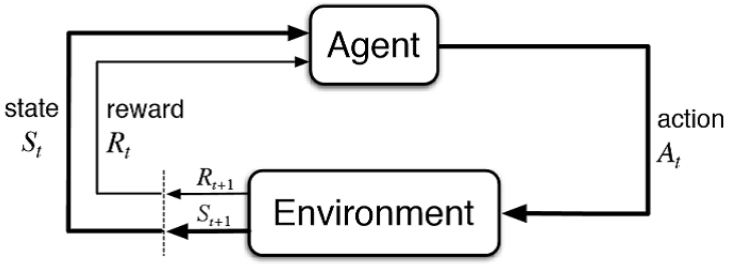

Figure 2.1: 強化学習による agent と環境間の相互作用 [50)]

ここで興味があるのは，累積報酬の期待値を最大化するような意思決定の規則 (方策と呼ぶ) を見つけることであるが，具体的な解法については 2.2 節で述べる．なお，制御工学の用語では，agent は制御器，行動は入力，環境は制御対象または plant，方策は制御則に相当する．

MDP の特徴の 1 つは，状態遷移，報酬および方策が確率論的な性質を有する点にある．例えば状態遷移は，状態 s_t および行動 a_t から次の状態 s_{t+1} へ遷移する条件付き確率として定式化される．いわば離散時間の非線形状態方程式の確率論的な拡張である．さらに，実装上の特徴として，普通は状態および行動が有限個であると仮定した有限 Markov 決定過程が用いられる．物理現象を対象とした最適制御問題の多くが連続時間かつ状態および行動が実数 (無限集合) として定式化されることを考えれば，有限 MDP は若干一般性を欠いているような印象を受けるが，実際にはこの仮定はさほど重大ではない．なぜなら連続時間かつ状態および行動が実数の最適制御問題であっても，数値的に解く際には離散時間かつ状態および行動が有限個[*2]の問題へと捨象されるからである．当然ながら離散化の性質や計算量は大きな問題となるが，数値的に解くことを前提とした最適制御問題の枠組みとしては，有限 MDP は極めて一般性の高いものであるといえる．

Definition 2.1 (離散時間 Markov 連鎖)
高々加算な集合[*3]$\mathcal{X}$ 上の確率過程 $\{X_n\}$ が任意の $n \geq 0$ と $x \in \mathcal{X}$ に対して

$$P(X_{n+1} = x_{n+1} \mid X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0) = P(X_{n+1} = x_{n+1} \mid X_n = x_n) \quad (2.1.1)$$

[*1] 意思決定問題と最適制御問題の包含関係は明確ではない (おそらく文脈によっても異なる) が，基本的には同一の概念と言ってよい．立場が意思決定者そのものか，意思決定規則の設計者かという差異で用語等が異なっているだけである．
[*2] 例えば倍精度浮動小数点 (IEEE 754) では 2^{64} 個の数が表現できる．全ての算術演算は 2^{64} 個の有限集合に対して閉じており，仮数部の bit 長 53 に対応して高々2^{-53} の丸め誤差を持つ．
[*3] 自然数全体の集合 $\mathbb{N}$ の濃度以下の濃度を持つ集合のこと．有限集合と可算無限集合．

を満たすとき，$\{X_n\}$ を離散時間 Markov 連鎖 (discrete-time Markov chain) と呼び，このように未来の状態が過去の状態に依らず，現在の状態のみによって決定されるような性質を Markov 性 (Markov property) と呼ぶ．

Remark. Markov 性は確率過程一般に対する概念であり，Markov 性を持つ確率過程である Markov 過程のうち，時間が離散的なものが離散時間 Markov 連鎖である．

Definition 2.2 (有限 Markov 決定過程)
有限 Markov 決定過程 (finite Markov Decision Process, finite MDP)[5, 21] は次の組 $(\mathcal{S}, \mathcal{A}, T, R)$ で定義される．

(i) 状態空間 $\mathcal{S}$: 状態の有限集合
(ii) 行動空間 $\mathcal{A}$: 行動の有限集合
(iii) 遷移関数 $T(s, a, s')$: 状態 s において行動 a を取ったときに状態 s' に遷移する確率

$$T(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a). \tag{2.1.2}$$

(iv) 報酬関数 $R(s, a, s')$: 状態 s において行動 a を取り，状態 s' に遷移した際の報酬 r_{t+1} の期待値

$$R(s, a, s') = E\left[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\right]. \tag{2.1.3}$$

Remark. 上記の定義は有限 MDP としては最も一般性の高いものであるが，よく用いられる仮定として，1. 報酬が現在の状態 s および行動 a のみに依存[*4] $R(s, a) = E\left[r_{t+1} \mid s_t = s, a_t = a\right]$，2. 遷移関数が決定論的 $s_{t+1} = T(s, a)$，3. 報酬関数が決定論的 $r_{t+1} = R(s, a, s')$，上記 3 つのいずれかの組み合わせがある．

Definition 2.3 (方策)
ある状態 s においてどのような行動 a を選択するかを表す関数を方策 (policy) と呼ぶ．次のように確率的方策と決定論的方策の 2 通りが考えられる．

(i) 確率論的方策 $\pi(a \mid s)$: 状態 s において行動 a が選択される確率

$$\pi(a \mid s) = P(a_t = a \mid s_t = s). \tag{2.1.4}$$

(ii) 決定論的方策 $\pi(s)$: 状態 s から行動 a への写像

$$a = \pi(s). \tag{2.1.5}$$

以上を用いれば，離散時間かつ状態および行動が有限個の確率的最適制御問題[*5]を次のように定義することができる．ここで，γ $(0 \leq \gamma < 1)$ は割引率 (discount factor) である．

[*4] 報酬が現在の状態 s および行動 a のみに依存する場合，報酬 r の index を状態 s および行動 a と揃えて $R(s, a) = E\left[r_t \mid s_t = s, a_t = a\right]$ (確率論的報酬)，$r = R(s, a)$ (決定論的報酬) と定義する場合もある．
[*5] 慣例として，動的計画法および強化学習は報酬の最大化問題，最適制御はコストの最小化問題として定式化されることが多いが，本論文においては最大化問題に統一して記述する．

Problem 2.1 (未知環境下における確率的最適制御問題)

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right]. \\ \text{subject to} \quad & s_{t+1}\sim P(s' \mid s,\, a), \\ & r_{t+1}\sim R(s,\, a,\, s'). \end{aligned} \tag{2.1.6}$$

式 (2.1.6) の定式化は離散時間の最適制御問題としては極めて一般的なものである．

Definition 2.4 (状態価値関数と行動価値関数)
方策 π の評価指標として，価値関数 (value function) を導入する．価値関数は，その引数によって状態価値関数 (state value function) と行動価値関数 (action-value function)[*6]の 2 通りが考えられる．状態 s において π に従って行動したときに得られる割引累積報酬の期待値を，状態価値関数 $V^\pi(s)$ として次のように定義する．

$$V^\pi(s) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right]. \tag{2.1.7}$$

同様に，s において行動 a を取った後，π に従って行動したときに得られる割引累積報酬の期待値を，行動価値関数 $Q^\pi(s,\, a)$ と定義する．

$$Q^\pi(s,\, a) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s,\, a_t = a\right]. \tag{2.1.8}$$

Remark. 定義より，状態価値関数 $V^\pi(s)$ と行動価値関数 $Q^\pi(s,\, a)$ の間には次の関係が成立する．

$$V^\pi(s) = \sum_a \pi(a \mid s) Q^\pi(s,\, a). \tag{2.1.9}$$

Proposition 2.1 (最適方策と最適状態価値関数)
状態価値関数 $V^\pi(s)$ を用いて方策 π の半順序[*7]関係を次のように定義する．

$$\pi_1 \preceq \pi_2 \iff V^{\pi_1}(s) \le V^{\pi_2}(s),\ \forall s \in \mathcal{S}. \tag{2.1.10}$$

このとき全ての方策よりも優れた，あるいは同等な方策が少なくとも 1 つ存在し，これを最適方策 (optimal policy) と呼ぶ．すなわち，最適方策 π^* は

$$\pi^* \in \arg\max_\pi V^\pi(s),\ \forall s \in \mathcal{S} \tag{2.1.11}$$

を満たす．最適方策は複数存在し得るが，全ての最適方策は唯一の最適状態価値関数 $V^*(s)$ および最適行動価値関数 $Q^*(s,\, a)$ を共有する．

$$V^*(s) = \max_\pi V^\pi(s),\ \forall s \in \mathcal{S}, \tag{2.1.12}$$

[*6]状態と行動の対の価値という点に着目して，状態行動価値関数とも呼ぶこともある．
[*7]例えば $\mathcal{S} = \{s_1,\, s_2\}$ に対して $V^{\pi_1}(s_1) < V^{\pi_2}(s_1)$ かつ $V^{\pi_1}(s_2) > V^{\pi_2}(s_2)$ のとき，π_1 と π_2 を比較することができない．このような比較不能な場合も許容する順序を半順序 (partial order) と呼ぶ．

$$Q^*(s,\, a) = \max_{\pi} Q^{\pi}(s,\, a),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A}. \tag{2.1.13}$$

Proof. 最適状態価値関数 $V^*(s)$ または最適行動価値関数 $Q^*(s,\, a)$ が存在すれば，π^* の 1 つは直ちに

$$\pi^*(s) = \arg\max_{a} \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^*(s')) = \arg\max_{a} Q^*(s,\, a),\ \forall s \in \mathcal{S} \tag{2.1.14}$$

として求めることができる．$V^*(s)$ および $Q^*(s,\, a)$ の存在と一意性は，最適 Bellman 作用素の唯一の不動点の存在によって示される．詳細は Proposition 2.1 を参照されたい．

Remark. 最適状態価値関数 $V^*(s)$ と最適行動価値関数 $Q^*(s,\, a)$ の間には次の関係が成立する (式 (2.1.9) の $\sum_a \pi(a \mid s)$ を形式的に $\max_a$ に置き換えればよい)．

$$V^*(s) = \max_{a} Q^*(s,\, a),\ \forall s \in \mathcal{S}. \tag{2.1.15}$$

Proposition 2.2 (Bellman 方程式)
状態 s の価値 $V^{\pi}(s)$ と，次の状態 s' の価値 $V^{\pi}(s')$ との間には，次の再帰的な関係

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^{\pi}(s')),\ \forall s \in \mathcal{S} \tag{2.1.16}$$

が成立し，これを状態価値関数 $V^{\pi}(s)$ についての Bellman 方程式 (Bellman equation または Bellman expectation equation) と呼ぶ．同様にして，行動価値関数 $Q^{\pi}(s,\, a)$ についての Bellman 方程式は

$$Q^{\pi}(s,\, a) = \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s',\, a') \right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A} \tag{2.1.17}$$

となる．

Proof. Appendix A.2 に示す．

Remark. 最適制御でしばしば用いられる，方策 π および報酬関数 R が決定論的かつ報酬が現在の状態 s および行動 a に依存するという仮定のもとでは，状態価値関数 $V^{\pi}(s)$ についての Bellman 方程式は

$$V^{\pi}(s) = R(s,\, \pi(s)) + \gamma \sum_{s'} P(s' \mid s,\, \pi(s)) V^{\pi}(s'),\ \forall s \in \mathcal{S} \tag{2.1.18}$$

とより単純な形[*8]で書くことができる．

Proposition 2.3 (最適 Bellman 方程式)
最適状態価値関数 $V^*(s)$ および最適行動価値関数 $Q^*(s,\, a)$ は次の最適 Bellman 方程式 (Bellman optimality equation) を満たす．

$$V^*(s) = \max_{a} \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^*(s')),\ \forall s \in \mathcal{S}, \tag{2.1.19}$$

$$Q^*(s,\, a) = \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \max_{a'} Q^*(s',\, a') \right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A}. \tag{2.1.20}$$

Proof. Appendix A.3 に示す．

[*8] 式 (2.1.16) においても，方策 π に関する周辺化をまとめて $R^{\pi}(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s,\, a) R(s,\, a,\, s')$, $P^{\pi}(s' \mid s) = \sum_a \pi(a \mid s) P(s' \mid s,\, a)$ と定義すると $V^{\pi}(s) = R^{\pi}(s) + \gamma \sum_{s'} P^{\pi}(s' \mid s) V^{\pi}(s')$ となり，式 (2.1.18) と同様の形に書き換えることができる．

Definition 2.5 (Bellman 作用素)
状態価値関数 $V^\pi(s)$ についての Bellman 方程式は，関数 V^π に対する線型変換と平行移動の組み合わせ，すなわち affine 変換と見做すことができる．そこで，任意の関数 $V : \mathcal{S} \to \mathbb{R}$ についての Bellman 方程式と同様の変換を

$$(\mathcal{T}_\pi V)(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V(s')),\ \forall s \in \mathcal{S} \tag{2.1.21}$$

と定義し，作用素 $\mathcal{T}_\pi$ を Bellman 作用素 (Bellman operator)[*9]と呼ぶ．同様にして，最適 Bellman 作用素 (optimal Bellman operator) $\mathcal{T}_*$ を次のように定義する．

$$(\mathcal{T}_* V)(s) = \max_a \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V(s')),\ \forall s \in \mathcal{S}. \tag{2.1.22}$$

Remark. Bellman 作用素 $\mathcal{T}_\pi$ を用いれば，Bellman 方程式は $(\mathcal{T}_\pi V)(s) = V(s),\ \forall s \in \mathcal{S}$ または単に $\mathcal{T}_\pi V = V$，最適 Bellman 方程式は $\mathcal{T}_* V = V$ と書ける．また，行動価値関数 $Q(s, a)$ についても同様にして，Bellman 作用素 $\mathcal{H}_\pi$ および最適 Bellman 作用素 $\mathcal{H}_*$ を次のように定義することができる．

$$(\mathcal{H}_\pi Q)(s, a) = \sum_{s'} P(s' \mid s, a) \left(R(s, a, s') + \gamma \sum_{a'} \pi(a' \mid s') Q(s', a') \right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A}, \tag{2.1.23}$$

$$(\mathcal{H}_* Q)(s, a) = \sum_{s'} P(s' \mid s, a) \left(R(s, a, s') + \gamma \max_{a'} Q(s', a') \right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A}. \tag{2.1.24}$$

Theorem 2.1 (Bellman 作用素の縮小性)
最適 Bellman 作用素 $\mathcal{T}_*$ は L^∞ ノルムに対して縮小作用素である．すなわち

$$\forall V_1\ \forall V_2,\ ||\mathcal{T}_* V_1 - \mathcal{T}_* V_2||_\infty \leq \gamma\, ||V_1 - V_2||_\infty \tag{2.1.25}$$

を満たす[*10]．したがって，Banach の不動点定理 [4] により $\mathcal{T}_*$ は唯一の不動点 $\mathcal{T}_* V^* = V^*$ を持つ．また，Bellman 作用素 $\mathcal{T}_\pi$ も同様に縮小作用素であり，唯一の不動点 $\mathcal{T}_\pi V^\pi = V^\pi$ を持つ．また，行動価値関数 $Q(s, a)$ についての Bellman 作用素 $\mathcal{H}_\pi$ および最適 Bellman 作用素 $\mathcal{H}_*$ も縮小作用素であり，唯一の不動点を持つことが知られている．
Proof. Appendix A.4 に示す．
Remark. 最適 Bellman 作用素の縮小性は，最適状態価値関数 $V^*(s)$ および最適方策 π^* の存在を保証するだけでなく，動的計画法と一部の強化学習アルゴリズムの収束性も保証する非常に重要な性質である．

[*9] Bellman backup operator とも呼ばれる．
[*10] 式 (2.1.25) の不等式を Lipschitz 条件，定数 γ を Lipschitz 定数と呼ぶ．

2.2　Bellman 方程式に基づく最適制御問題の数値解法

最適制御問題の数値解法は，しばしば間接法 (indirect method) と直接法 (direct method) の 2 つに分類される．間接法は，最適解が満たすべき最適性条件，すなわち古典変分法における Euler-Lagrange 方程式，Pontryagin の最大値原理 [40]，Bellman 方程式 (もしくは連続時間における Hamilton-Jacobi-Bellman 方程式 [29]) などを解く手法である．一方で，直接法は状態や行動を離散化して最適制御問題を数理計画問題に変換して解く手法である．間接法と直接法の代表的な枠組みについて分類すると，例えば Fig. 2.2 のようになる．

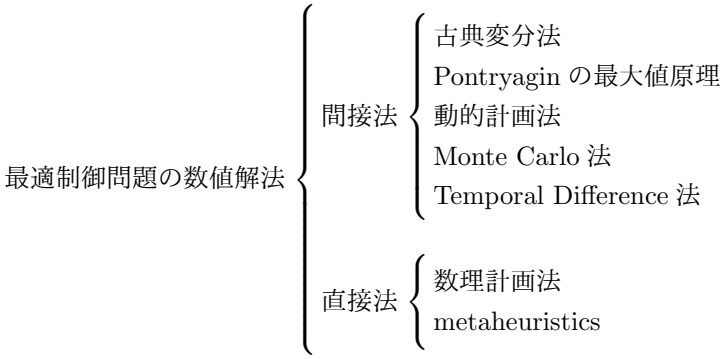

Figure 2.2: 最適制御問題の数値解法の分類

最適制御問題の数値解法を別の観点から分類するとすれば，環境についての事前知識を必要とするか，必要としないかの 2 通りに分けることができる．事前知識の必要性の観点から Fig. 2.2 をさらに分類すると，Table 2.1 のようになる．

Table 2.1: 最適制御問題の数値解法の分類

	事前知識を必要とする	事前知識を必要としない
間接法	古典変分法 Pontryagin の最大値原理 動的計画法	Monte Carlo 法 Temporal Difference 法
直接法	数理計画法[*11]	metaheuristics

本節では間接法のうち，Bellman 方程式に基づく手法である動的計画法と強化学習 (Monte Carlo 法および Temporal Difference 法) ならびに強化学習の発展的な概念について述べる．Bellman 方程式に基づく手法はさらに，最適性条件である Bellman 方程式を解く方策評価 (policy evaluation) または予測 (prediction)[*12] と呼ばれる問題と，価値関数に基づいて方策を改善する方策改善 (policy improvement) という問題を含んでいる．また，環境と相互作用しながら予測および方策改善を行うような手法を特に制御 (control)[*13] と呼ぶ．

[*11] 文脈によっては数理計画問題の数値解法そのものを数理計画法と呼ぶ場合もある．詳しくは 3 章で述べるが，ここでは数理モデルに基づいた手法，すなわち事前知識を必要とする手法のみを数理計画法と呼ぶこととする．
[*12] 事前知識を必要とする手法では方策評価，必要としない手法では予測と区別される場合が多い．
[*13] ここでの制御とは，行動が最適であるかどうかに関わらず，何らかの行動を出力するような手法を指す．

2.2.1　動的計画法

動的計画法 (Dynamic Programming, DP) とは，対象となる問題を部分問題に分割し，部分問題の再帰的な関係を利用して解く手法の総称[*14]である．中でも最適制御問題における動的計画法とは，Bellman 方程式で示される状態価値関数ないし行動価値関数の再帰的な関係を用いた解法を指し，このような再帰性を bootstrap 性と呼ぶ．2.1 節で示したように，Bellman 方程式は，それ自体に遷移関数および報酬関数を含んでいる．したがって，動的計画法が対象とするのは次のような環境についての知識が得られるという前提の問題である．

Problem 2.2 (既知環境下における確率的最適制御問題)

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right] & \\ \text{subject to} \quad & s_{t+1}\sim P(s'\mid s,\,a) & \text{(given)} \\ & r_{t+1}\sim R(s,\,a,\,s') & \text{(given)} \end{aligned} \tag{2.2.1}$$

2.2.1.1　方策反復法

方策反復法 (Policy Iteration, PI)[21] とは，以下に示す反復方策評価と方策改善を交互に繰り返す手法である．

Definition 2.6 (反復方策評価)
環境についての知識が与えられているとき，状態価値関数 $V^\pi(s)$ についての Bellman 方程式

$$V^\pi(s)=\sum_a \pi(a\mid s)\sum_{s'}P(s'\mid s,\,a)(R(s,\,a,\,s')+\gamma V^\pi(s')),\ \forall s\in\mathcal{S} \tag{2.2.2}$$

は，$|\mathcal{S}|$ 個の未知変数を持つ $|\mathcal{S}|$ 個の連立 1 次方程式となる．この連立方程式は厳密に解くこともできるが，$|\mathcal{S}|$ が大きい場合には計算量が膨大となるため，次の反復解法が用いられる．

$$V(s)\leftarrow\sum_a \pi(a\mid s)\sum_{s'}P(s'\mid s,\,a)(R(s,\,a,\,s')+\gamma V(s')) \tag{2.2.3}$$

式 (2.2.3) の解法を反復方策評価 (iterative policy evaluation) と呼ぶ．

Proposition 2.4 (反復方策評価の収束性)
反復方策評価 $V_{k+1}=\mathcal{T}_\pi V_k$ において，$V(s)$ は $k\to\infty$ の極限で $V^\pi(s)$ に収束する．
Proof. Proposition 2.1 と Banach の不動点定理 [4] により

$$\lim_{k\to\infty}\mathcal{T}_\pi^{(k)}V=V^\pi \tag{2.2.4}$$

となる．

次に，方策 π を別の方策 π' に変更することで，方策が改善するか，すなわち $V^\pi(s)\le V^{\pi'}(s)$ となるかを計算する．このような π' を見つけることができれば，逐次的に方策改善を行い，最終的に最適な方策 π^* を得ることが可能となる．次の方策改善定理 (policy improvement theorem) は，$V^\pi(s)\le V^{\pi'}(s)$ となるための十分条件を示し，π' を見つけるための指針を与えるものである．

[*14] 例えば最短経路問題の解法である Dijkstra 法も動的計画法に含まれる．

Proposition 2.5 (方策改善定理)
状態 s において方策 π に従って行動したときの状態価値関数 $V^{\pi}(s)$ と，s において別の方策 π' に基づいて行動 $\pi'(s)$ を取った後，π に従って行動したときの行動価値関数 $Q^{\pi}(s, \pi'(s))$ について，次の関係が成り立つ．

$$V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s)) \Longrightarrow V^{\pi}(s) \leq V^{\pi'}(s),\ \forall s \in \mathcal{S}. \tag{2.2.5}$$

Proof. Appendix A.4 に示す．

Definition 2.7 (Greedy 方策)
方策改善の条件 $V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$ を満たすような π' は複数考えられるが，最も単純なものとして，短期的に最良となるような行動を決定論的に選択する greedy 方策がある．

$$\begin{aligned}\pi'(s) &= \arg\max_{a} Q^{\pi}(s, a) \\ &= \arg\max_{a} \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V^{\pi}(s')),\ \forall s \in \mathcal{S}.\end{aligned} \tag{2.2.6}$$

Proposition 2.6 (Greedy 方策による方策改善)
greedy 方策 $\pi'(s) = \arg\max_{a} Q^{\pi}(s, a)$ は方策改善の条件 $V^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$ を満たす．
Proof. $f(\arg\max_{x} f(x)) = \max_{x} f(x)$ であることに注意すれば，

$$\begin{aligned}Q^{\pi}(s, \pi'(s)) &= Q^{\pi}(s, \arg\max_{a} Q^{\pi}(s, a)) = \max_{a} Q^{\pi}(s, a) \\ &\geq Q^{\pi}(s, \pi(s)) = V^{\pi}(s),\ \forall s \in \mathcal{S}\end{aligned} \tag{2.2.7}$$

となる．

方策反復法の擬似コードを Algorithm 1 に示す．第 3 行から第 7 行が反復方策評価，第 8 行から第 10 行が方策改善に相当する．

Algorithm 1 Policy Iteration

Require: transition function $P(s' \mid s, a)$, reward function $R(s, a, s')$, discount factor γ
1: Initialize policy $\pi(s)$ and value function $V(s)$ arbitrary
2: **repeat**
3:　**repeat**
4:　**for** all state s **do**
5:　$V(s) \leftarrow \sum_{s'} P(s' \mid s, \pi(s))(R(s, \pi(s), s') + \gamma V(s'))$
6:　**end for**
7:　**until** $V(s)$ converge
8:　**for** all state s **do**
9:　$\pi(s) \leftarrow \arg\max_{a} \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V(s'))$
10:　**end for**
11: **until** $\pi(s)$ converge

2.2.1.2　価値反復法

方策反復法の問題点は，方策の更新の度に Bellman 方程式を解いて方策評価を行う必要があるということである．反復方策評価は，全状態に対する多重の反復を行っているため，計算量の面で不利である．そこで，Bellman 方程式では無く，最適 Bellman 方程式を直接解くことで最適状態価値関数 $V^*(s)$ および最適方策 $\pi^*(s)$ を求める計算法が価値反復法 [5] である．

Algorithm Definition 2.1 (価値反復法)
環境についての知識が与えられているとき，最適 Bellman 方程式

$$V^*(s) = \max_a \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V^*(s')),\ \forall s \in \mathcal{S} \tag{2.2.8}$$

は，$|\mathcal{S}|$ 個の未知変数を持つ $|\mathcal{S}|$ 個の連立非線形方程式となる．したがって，反復方策評価と同様にして，次の反復解法を用いて解くことができる．

$$V(s) \leftarrow \max_a \sum_{s'} P(s' \mid s, a)\left(R(s, a, s') + \gamma V(s')\right). \tag{2.2.9}$$

さらに，Proposition 2.1 でも述べたように，最適状態価値関数 $V^*(s)$ に対する greedy 方策は当然最適方策 $\pi^*(s)$ である．

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V^*(s')),\ \forall s \in \mathcal{S}. \tag{2.2.10}$$

式 (2.2.9) および式 (2.2.10) の解法を価値反復法 (Value Iteration, VI)[5] と呼ぶ．

Proposition 2.7 (価値反復法の収束性)
価値反復法 $V_{k+1} = \mathcal{T}_* V_k$ において，$V(s)$ は $k \to \infty$ の極限で $V^*(s)$ に収束する．
Proof. Proposition 2.1 と Banach の不動点定理 [4] により

$$\lim_{k\to\infty} \mathcal{T}_*^{(k)} V = V^* \tag{2.2.11}$$

となる．

価値反復法の擬似コードを Algorithm 2 に示す．

Algorithm 2 Value Iteration

Require: transition function $P(s' \mid s, a)$, reward function $R(s, a, s')$, discount factor γ
1: Initialize policy $\pi(s)$ and value function $V(s)$ arbitrary
2: **repeat**
3:　　**for** all state s **do**
4:　　$V(s) \leftarrow \max_a \sum_{s'} P(s' \mid s, a)\left(R(s, a, s') + \gamma V(s')\right)$
5:　　**end for**
6: **until** $V(s)$ converge
7: $\pi(s) \leftarrow \arg\max_a \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V(s'))$

2.2.2 Monte Carlo 法

Monte Carlo (MC) 法とは，乱数を用いた数値計算手法の総称であるが，最適制御問題においては環境についての知識を用いない代わりに，ある方策 π に従って初期状態 s_0 から終端状態 s_T まで遷移したときの状態および報酬の一連の系列を用いて予測または制御を行う手法を指す．したがって，MC 法の対象は次のような終端状態 s_T または終端時間 T が固定された問題である．

Problem 2.3 (未知環境下における有限時間確率的最適制御問題)

$$\begin{aligned} &\underset{\pi}{\text{maximize}} && E_\pi\left[\sum_{t=0}^{T}\gamma^t r_{t+1}\right]. \\ &\text{subject to} && s_{t+1}\sim P(s'\mid s,\,a) && \text{(not given)}, \\ &&& r_{t+1}\sim R(s,\,a,\,s') && \text{(not given)}. \end{aligned} \tag{2.2.12}$$

このような問題を episode 型の問題と呼び，s_0 から s_T までの状態および報酬の一連の系列あるいは s_0 から s_T までの期間そのものを episode と呼ぶ．

Algorithm Definition 2.2 (Monte Carlo 予測)
ある方策 π に従ったときの，i 番目の episode $\{s_0,\,s_1,\,s_2,\ldots,\,s_T\}$, $\{r_1,\,r_2,\,r_3,\ldots,\,r_{T+1}\}$ における時刻 $t<T$ 以後の累積報酬

$$G_{t,i}(s)=\sum_{k=0}^{T}\gamma^k r_{t+k+1}. \tag{2.2.13}$$

を状態 s の収益 (return) と呼ぶ．各 episode が独立であるとき，次式のように状態価値関数 $V^\pi(s)$ を n 回目の episode までの収益の平均値として推定する手法を MC 予測 (MC prediction) と呼ぶ．

$$V_n(s)=\frac{1}{n}\sum_{i=1}^{n}G_{t,i}(s). \tag{2.2.14}$$

Remark. 各 episode において，ある状態 s を複数回訪問している場合は，どの時点での累積報酬を用いるかが問題となるが，代表的なものに，初めて s を訪れたときの累積報酬を用いる初回訪問 MC 予測 (first-visit MC prediction) と，それぞれの訪問の累積報酬の平均値を用いる逐一訪問 MC 予測 (every-visit MC prediction) がある．

Theorem 2.2 (Monte Carlo 予測の収束性)
MC 予測は，$n\to\infty$ の極限で，ほとんど確実[*15]に $V^\pi(s)$ に収束する．すなわち次式を満たす．

$$P\left(\lim_{n\to\infty}V_n(s)=V^\pi(s)\right)=1. \tag{2.2.15}$$

Proof. 定義より，$V^\pi(s)$ は $G_{t,i}(s)$ の期待値そのものである．したがって，各 episode が独立である限り，大数の強法則により $G_{t,i}(s)$ の平均値は，ほとんど確実に $V^\pi(s)$ に収束する．

ここでは一例として，初回訪問 MC 予測の擬似コードを Algorithm 3 に示す．

[*15]ほとんど確実に (almost surely) とは，式 (2.2.15) で示した通り，その事象が起こる確率が 1 であることを意味しており，通常の収束よりも弱い概念である．例えば硬貨を投げ続けていつかは表が出る確率は 1 であるが，裏が出続けるという事象が存在しないとまでは言えない．

Algorithm 3 First-Visit Monte Carlo Prediction

Require: policy $\pi(a \mid s)$, discount factor γ

Initialize value function $V(s)$ arbitrary
Initialize empty list $\mathcal{G}(s)$
repeat forever
    Generate an episode $\{s_0, s_1, s_2, \ldots, s_T\}$, $\{r_1, r_2, r_3, \ldots, r_{T+1}\}$ using π
    for each step of episode **do**
        if state s is **not** in $s_0, s_1, s_2, ..., s_{t-1}$ **then**
            Append return $G_t(s)$ to $\mathcal{G}(s)$
            $V(s) \leftarrow \text{average}(\mathcal{G}(s))$
        end if
    end for

MC 予測に方策改善を組み合わせることによって，MC 制御 (MC control) を行うことが可能である．既に述べたように，greedy 方策

$$\begin{aligned}\pi'(s) &= \arg\max_a Q^\pi(s, a) \\ &= \arg\max_a \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V^\pi(s')),\ \forall s \in \mathcal{S}\end{aligned} \tag{2.2.16}$$

は，方策改善の条件 $V^\pi(s) \leq Q^\pi(s, \pi'(s))$ を満たしているが，方策反復法と異なり環境についての知識を利用できないため，行動価値関数 $Q^\pi(s, a)$ を用いる必要がある．また，greedy 方策は決定論的方策であるから，一般に全ての状態行動対 (s, a) が訪問されるとは限らない．解決策の一つは，任意の状態行動対 (s, a) について $(s_0, a_0) = (s, a)$ となる確率が 0 ではない仮定することであり，このような仮定を開始点探査 (exploring starts, ES) と呼ぶ．開始点探査の仮定を用いた MC ES 制御 (Monte Carlo ES control) の擬似コードを Algorithm 4 に示す．

Algorithm 4 Monte Carlo ES Control

Require: discount factor γ

Initialize policy $\pi(s)$ and action-value function $Q(s, a)$ arbitrary
Initialize empty list $\mathcal{G}(s)$
repeat forever
    Choose (s_0, a_0) randomly s.t. all tuple have probability > 0
    Generate an episode $\{(s_0, a_0), (s_1, a_1), \ldots, (s_T, a_T)\}$, $\{r_1, r_2, \ldots, r_{T+1}\}$ using π
    for each step of episode **do**
        if tuple (s, a) is **not** in $(s_0, a_0), (s_1, a_1), \ldots, (s_{t-1}, a_{t-1})$ **then**
            Append $G_t(s)$ to $\mathcal{G}(s)$
            $Q(s, a) \leftarrow \text{average}(\mathcal{G}(s))$
            $\pi(s) \leftarrow \arg\max_a Q(s, a)$
        end if
    end for

初期状態 s_0 が given であるか not given であるかに関わらず，開始点探査の仮定は実際の最適制御問題においては非現実的なものである．代替策の一つは，確率論的かつ方策改善の条件を満たす方策[*16]を用いることであり，よく用いられるものとして次の ϵ-greedy 方策がある．

Definition 2.8 (ϵ-Greedy 方策)
ある定数 ϵ $(0 < \epsilon < 1)$ に対して確率 $1-\epsilon$ で greedy 方策と同じ行動を選択し，確率 ϵ で random な行動を選択する次のような方策を考える．

$$a = \begin{cases} \arg\max_a Q(s, a) & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon\,. \end{cases} \tag{2.2.17}$$

また，式 (2.2.17) を条件付き確率として書き換える[*17]と次のようになる．

$$\pi(a \mid s) = \begin{cases} 1-\epsilon+\epsilon/|\mathcal{A}| & \text{if } a = \arg\max_a Q(s, a) \\ \epsilon/|\mathcal{A}| & \text{otherwise}\,. \end{cases} \tag{2.2.18}$$

この方策を ϵ-greedy 方策と呼び，方策改善の条件を満たすことが知られている．

Proposition 2.8 (ϵ-Greedy 方策による方策改善)
ϵ-greedy 方策は方策改善の条件 $V^\pi(s) \leq Q^\pi(s, \pi'(s))$ を満たす．
Proof. Appendix A.6 に示す．

ϵ-greedy 方策は，それ自体に行動価値関数 $Q(s, a)$ を含んでいるため，ϵ-greedy 方策によって生成された episode を用いて MC 法で $Q(s, a)$ を推定する場合，方策と価値関数は相互に依存しているといえる．このように，実際の行動を生成する方策 (behavior policy) と，評価され改善される方策[*18] (target policy) とを同一とする手法を方策 on 型 (on-policy) と呼び，異なる手法を方策 off 型 (off-policy) と呼ぶ．ϵ-greedy 方策を用いて開始点探査の仮定を回避した方策 on 型 MC 制御の擬似コードを Algorithm 5 に示す．

開始点探査の仮定を回避するもう一つの方法は，前述の方策 off 型 MC 法を用いることである．擬似コードは割愛するが，例えば target policy を greedy 方策，behavior policy を ϵ-greedy 方策とすればよい．

[*16] このような方策を soft 方策と呼び，$\epsilon \to 1$ で greedy 方策に収束する方策を ϵ-soft 方策と呼ぶ．
[*17] 式 (2.2.18) は $1 \times (1-\epsilon+\epsilon/|\mathcal{A}|) + (|\mathcal{A}|-1) \times (\epsilon/|\mathcal{A}|) = 1$ となっており，式 (2.2.17) を満たすことが分かる．
[*18] estimation policy とも呼ばれる．

Algorithm 5 On-Policy First-Visit Monte Carlo Control

Require: discount factor γ

Initialize policy $\pi(a \mid s)$ and action-value function $Q(s, a)$ arbitrary
Initialize empty list $\mathcal{G}(s)$
repeat forever
    Generate an episode $\{(s_0, a_0), (s_1, a_1), \ldots, (s_T, a_T)\}$, $\{r_1, r_2, \ldots, r_{T+1}\}$ using π
    for each step of episode **do**
        if tuple (s, a) is **not** in $(s_0, a_0), (s_1, a_1), \ldots, (s_{t-1}, a_{t-1})$ **then**
            Append $G_t(s)$ to $\mathcal{G}(s)$
            $Q(s, a) \leftarrow \text{average}(\mathcal{G}(s))$
        end if
        $a^* \leftarrow \arg\max_a Q(s, a)$
        for all action a **do**
            $$\pi(a \mid s) = \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}| & \text{if } a = a^* \\ \epsilon/|\mathcal{A}| & \text{otherwise} \end{cases}$$
        end for
    end for

2.2.3　Temporal Difference 法

Temporal Difference (TD) 法 [49] は，動的計画法と MC 法の考え方を組み合わせたものである．MC 法の欠点は，価値関数の更新を行うためには episode の完了を待たなくてはならない点にある．これに対して，TD 法では，動的計画法と同様に価値関数の bootstrap 性を利用することにより，各時刻における価値関数の更新を可能にしたものである．

2.2.3.1　TD(0) 予測

Algorithm Definition 2.3 (Monte Carlo 予測の漸進的実装)
MC 予測では，収益 $G_{t,i}(s) = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$ の平均として状態価値関数 $V(s)$ を推定していた．ここで，n 番目と $n+1$ 番目の episode における推定値について

$$\begin{aligned} V_{n+1}(s) = \frac{1}{n+1}\sum_{i=1}^{n+1} G_{t,i}(s) &= \frac{1}{n+1}\left(G_{t,n+1}(s) + \sum_{i=1}^{n} G_{t,i}(s)\right) \\ &= \frac{1}{n+1}\left(G_{t,n+1}(s) + \frac{n+1}{n}\sum_{i=1}^{n} G_{t,i}(s) - \frac{1}{n}\sum_{i=1}^{n} G_{t,i}(s)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} G_{t,i}(s) + \frac{1}{n+1}\left(G_{t,n+1}(s) - \frac{1}{n}\sum_{i=1}^{n} G_{t,i}(s)\right) \\ &= V_n(s) + \frac{1}{n+1}\left(G_{t,n+1}(s) - V_n(s)\right) \end{aligned} \tag{2.2.19}$$

なる関係が成り立つ．したがって，MC 予測における $V(s)$ の更新式は次のように漸進的な形式に書き換えることができる．

$$V(s) \leftarrow V(s) + \frac{1}{n+1}\left(G_t - V(s)\right). \tag{2.2.20}$$

Algorithm Definition 2.4 (α 不変 Monte Carlo 予測)
式 (2.2.20) を簡略化して，$1/(n+1)$ を α $(0 < \alpha \leq 1)$ に置き換えれば

$$V(s) \leftarrow V(s) + \alpha\left(G_t - V(s)\right) \tag{2.2.21}$$

となる．ここで，α を step size parameter または学習率 (learning rate) と呼び，式 (2.2.21) において α を定数とする方法を α 不変 MC 予測 (contant-α MC prediction) と呼ぶ．
Remark. α 不変 MC 予測における $V(s)$ の更新式は，現時点での推定値 $V(s)$ を α 分だけ目標値 G_t に近づけていると解釈できる．しかしながら，前述のように MC 法では目標値 G_t を得るには episode の完了を待たなくてはならない．

Algorithm Definition 2.5 (TD(0) 予測)
episode の完了を待たずに予測を行うためには，次の Bellman 方程式の期待値表現 (Appendix A.5)

$$V^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] = E_{\pi}[r' + \gamma V^{\pi}(s') \mid s_t = s] \tag{2.2.22}$$

で示される $V^\pi(s)$ の bootstrap 性を利用する．すなわち，式 (2.2.21) の G_t を $r' + \gamma V(s')$ に置き換えると

$$V(s) \leftarrow V(s) + \alpha \left(r' + \gamma V(s') - V(s) \right) \tag{2.2.23}$$

となる．式 (2.2.23) 中の $r' + \gamma V(s') - V(s)$ を即時的な誤差 (Temporal Difference error, TD error) と呼び，更新式として式 (2.2.23) を用いる手法を TD(0) 予測と呼ぶ．

Theorem 2.3 (TD(0) 予測の収束性)
TD(0) 予測において以下の条件を満たすとき，$V(s)$ はほとんど確実に $V^\pi(s)$ に収束する．

(i) 状態空間 $\mathcal{S}$ が有限である
(ii) $0 \leq \alpha_t(s) \leq 1$，$\sum_t \alpha_t(s) = \infty$，$\sum_t \alpha_t^2(s) < \infty$
(iii) 報酬 r が有界である

Proof. Appendix A.7 に示す．

TD(0) 予測の擬似コードを Algorithm 6 に示す．

Algorithm 6 TD(0) Prediction

Require: policy $\pi(a \mid s)$, learning rate α, discount factor γ

Initialize value function $V(s)$ arbitrary
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        Choose action a from s using policy $\pi(a \mid s)$
        Execute action a, observe reward r' and next state s'
        $V(s) \leftarrow V(s) + \alpha[r' + \gamma V(s') - V(s)]$
        $s \leftarrow s'$
    end for
end for

2.2.3.2 SARSA

SARSA[44, 49] は最も基本的な TD 制御の手法であり，TD(0) 予測をそのまま制御に拡張したものである．MC 制御と同様に，soft 方策を用いて予測と方策改善を繰り返すことで，最終的に最適行動価値関数 $Q^*(s, a)$ および最適方策 $\pi^*(a \mid s)$ を得る．

Algorithm Definition 2.6 (SARSA)
TD(0) 予測の更新式

$$V(s) \leftarrow V(s) + \alpha \left(r' + \gamma V(s') - V(s)\right) \tag{2.2.24}$$

における状態価値関数 $V(s)$ を行動価値関数 $Q(s, a)$ に置き換えると

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r' + \gamma Q(s', a') - Q(s, a)) \tag{2.2.25}$$

となる．式 (2.2.25) の更新式と soft 方策とを用いる手法を SARSA と呼ぶ[*19]．

Theorem 2.4 (SARSA の収束性)
SARSA において以下の条件を満たすとき，$Q(s, a)$ はほとんど確実に $Q^*(s, a)$ に収束する．

(i) 状態空間 $\mathcal{S}$ および行動空間 $\mathcal{A}$ が有限である
(ii) $0 \leq \alpha_t(s, a) \leq 1$，$\sum_t \alpha_t(s, a) = \infty$，$\sum_t \alpha_t^2(s, a) < \infty$
(iii) 報酬 r が有界である
(iv) 方策が $t \to \infty$ の極限で greedy 方策に収束する[*20]

Proof. Appendix A.8 に示す．

なお，SARSA における更新の目標値 $r' + \gamma Q(s', a')$ は，現在の方策によって選択された行動 a' の価値 $Q(s', a')$ を含んでいる．したがって，SARSA は方策 on 型の手法であるといえる．SARSA の擬似コードを Algorithm 7 に示す．

Algorithm 7 SARSA

Require: learning rate α, discount factor γ
Initialize action-value function $Q(s, a)$ arbitrary
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        Execute action a, observe reward r' and next state s'
        Choose action a' from s' using policy derived from $Q(s, a)$
        $Q(s, a) \leftarrow Q(s, a) + \alpha(r' + \gamma Q(s', a') - Q(s, a))$
        $s \leftarrow s', a \leftarrow a'$
    end for
end for

[*19] SARSA という名称は，式 (2.2.25) 中に現れる変数 s，a，r'，s'，a' を繋げたものに由来する．
[*20] 時間とともに ϵ を減少させる場合，ϵ-soft 方策はこの条件を満たす．条件 (ii) と合わせて Greedy in the Limit with Infinite Exploration, GLIE とも呼ばれる．

2.2.3.3 Q-Learning

前述のように，SARSA における更新の目標値 $r'+\gamma Q(s', a')$ は，現在の方策によって選択された行動 a' の価値 $Q(s', a')$ を含んでいる．一方で，Q-learning[54] では更新の目標値を $r+\gamma \max_{a'} Q(s', a')$ とすることで，(greedy に選択された) 最も高い行動の価値を推定に用いる手法である．なお，探索を行う必要から，行動選択には SARSA と同様に soft 方策が用いられる．したがって，Q-learning は方策 off 型の手法であると言える．

Algorithm Definition 2.7 (Q-Learing)
次の更新式と soft 方策を用いる手法を Q-learning と呼ぶ．

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r' + \gamma \max_{a'} Q(s', a') - Q(s, a)). \tag{2.2.26}$$

Theorem 2.5 (Q-Learning の収束性)
Q-learning において以下の条件を満たすとき，$Q(s, a)$ はほとんど確実に $Q^*(s, a)$ に収束する [42]．

(i) 状態空間 $\mathcal{S}$ および行動空間 $\mathcal{A}$ が有限である
(ii) $0 \leq \alpha_t(s, a) \leq 1$，$\sum_t \alpha_t(s, a) = \infty$，$\sum_t \alpha_t^2(s, a) < \infty$
(iii) 報酬 r が有界である

Proof. Appendix A.9 に示す．
Remark. Q-learing は方策 off 型の手法であるため，SARSA の場合と異なり方策についての条件は存在しない．

Q-learing の擬似コードを Algorithm 8 に示す．

Algorithm 8 Q-Learning

Require: learning rate α, discount factor γ
Initialize action-value function $Q(s, a)$ arbitrary
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        Choose action a from s using policy derived from $Q(s, a)$
        Execute action a, observe reward r' and next state s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha(r' + \gamma \max_{a'} Q(s', a') - Q(s, a))$
        $s \leftarrow s'$
    end for
end for

2.3　強化学習の発展的な概念

2.2 節で述べた手法のバリエーションは数多く存在する．それらを全てを網羅することは到底出来ないが，本節では，より発展的な強化学習として，価値関数の関数近似に関しての発展型として深層強化学習と，事前知識の組み込みに関連するものとして，model-based 強化学習と安全な強化学習について述べる．

2.3.1　深層強化学習

深層強化学習 (deep reinforcement learning) とは，強化学習における価値関数および/または方策を Neural Network, NN で表現したものである．古典的な強化学習の欠点は，大規模な状態行動空間や連続な状態行動空間での学習が困難であるという点にある．例えば，SARSA や Q-learning では行動価値関数 $Q(s, a)$ は $|\mathcal{S}| \times |\mathcal{A}|$ の配列と見做すことができるため，状態空間 $\mathcal{S}$ や行動空間 $\mathcal{A}$ が大きくなるにしたがって，必要とする計算資源は爆発的に増大する．また，仮に状態の観測値が連続値であっても，学習の際には (適切な離散化幅で) 離散化する必要がある．

深層強化学習はこのような問題に対して，価値関数および/または方策を DNN で表現することで対処する．$Q(s, a)$ を NN で近似する試み自体は以前より存在していた [43] が，深層強化学習に分類される手法の多くは，DNN での表現に適応するために以下のような工夫を用いている．

Experience Replay[33]

これは回帰問題一般にも言えることであるが，標本間の強い相関はしばしば過剰適合を引き起こす．最適制御問題において，$Q(s, a)$ の引数である状態 s および行動 a の系列は，当然時間的に相関するはずであるから，それらを直接学習に用いるのは好ましくない．experience replay は，過去の経験 (状態 s，行動 a，次の状態 s'，報酬 r') を一旦 replay memory に保存し，学習には replay memory から無作為に抽出した経験を用いることで，過剰適合を回避する工夫である．

Fixed Target Network

例えば Q-learing の更新式は，現在の推定値 $Q(s, a)$ を目標値 $r' + \gamma \max_{a'} Q(s', a')$ に α の割合だけ近づけるというものであった．しかしながら，$Q(s, a)$ を NN で表現する場合，更新の対象そのものが目標値に含まれていると，学習が安定しにくいという問題がある．fixed target network は，目標値 $r + \gamma \max_{a'} Q(s', a')$ に相当する部分を一定の期間固定または平滑化することで，学習の安定化を図る工夫である．

Clipping Rewards

これは各時刻で得られる報酬を $[-1, 1]$ に正規化することで，$Q(s, a)$ の急激な変動を防ぐとともに，複数の問題を解きたい場合に，hyperparameter[*21] を再設計する手間を省くことが可能となる工夫である．一方で，clipping を行うと本来の報酬の多寡があまり区別されなくなってしまうため，上記 2 つの工夫に比べるとその重要性は低い．

[*21] 学習によって推定される parameter 以外の parameter のこと．例えば $Q(s, a)$ を NN で表現する場合には，NN そのものの parameter 以外の，NN の構造や学習条件等についての parameter を指す．

2.3.1.1 DQN

DQN (Deep Q-Network)[35,36] は，Q-learning における行動価値関数 $Q(s, a)$ を NN で近似した上で，前述のような学習を安定させる工夫を用いた手法である．

Algorithm Definition 2.8 (Deep Q Network)
DQN では，$Q(s, a)$ を $Q(s, a; \theta_Q)$ と近似する．ここで，θ_Q は NN の parameter である．Q-learning の更新式は，現在の推定値 $Q(s, a)$ を目標値 $r' + \gamma \max_{a'} Q(s', a')$ に近づけるというものであった．そこで，損失関数 L を

$$L = (r' + \gamma \max_{a'} Q(s', a'; \theta_Q) - Q(s, a; \theta_Q))^2 \tag{2.3.1}$$

として，各時刻で L を最小化するように θ_Q を更新する．このとき s, a, r', s' は replay memory から無作為に抽出された値を用いる．fixed target network を用いる場合には，目標値 $r' + \gamma \max_{a'} Q(s', a'; \theta_Q)$ の paramter を θ_Q と別個のものとして

$$L = (r' + \gamma \max_{a'} Q(s', a'; \theta_{Q'}) - Q(s, a; \theta_Q))^2 \tag{2.3.2}$$

とすればよい．

DQN の擬似コードを Algorithm 9 に示す．この例では mini-batch 学習を行い，target の更新法を smoothing としている．

Algorithm 9 DQN (Deep Q-Network)

Require: discount factor γ, epsilon ϵ, smoothing factor τ

Initialize critic $Q(s, a; \theta_Q)$ arbitrary
Initialize target critic parameters $\theta_{Q'} \leftarrow \theta_Q$
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        Choose action a from s using policy derived from $Q(s, a)$
        Execute action a, observe reward r' and next state s'
        Store experience (s, a, r, s') in $\mathcal{R}$
        Sample random minibatch of N experiences (s_i, a_i, r_i, s'_i) from $\mathcal{R}$
        Update critic by minimizing loss:
        $L = \frac{1}{N} \sum_{i=1}^{N} (r'_i + \gamma \max_{a'} Q(s'_i, a'; \theta_{Q'}) - Q(s_i, a_i; \theta_Q))^2$
        Update target critic:
        $\theta_{Q'} \leftarrow \tau \theta_Q + (1 - \tau) \theta_{Q'}$
        $s \leftarrow s'$
    end for
end for

2.3.1.2 DDPG

DDPG (Deep Deterministic Policy Gradient)[32, 46] は，actor-critic 型の手法である．actor-critic とは，将来に渡る報酬 (評価関数) 和の予測を行う critic とは独立に，状態から行動 (制御入力) を決定する actor を有するアルゴリズムの枠組みを指す．actor は制御工学でいうところの制御器と同等のものであり，これにより critic のみを有する古典的な強化学習とは異なり，連続な状態および行動を扱うことを可能としている．さらに，従来の actor-critic が確率的方策しか学習できなかったのに対し，DDPG は決定論的方策を学習することができ，actor および critic を NN で表現することで，より複雑なタスクに対応している．以上のことから，DDPG は制御工学の考え方に最も近い強化学習アルゴリズムであるといえる．

本論文で用いる DDPG の疑似コードを以下に示す．

Algorithm 10 DDPG

Require: discount factor γ, smoothing factor τ

Initialize critic $Q(s,\, a;\, \theta_Q)$ arbitrary
Initialize target critic parameters $\theta_{Q'} \leftarrow \theta_Q$
Initialize actor $\pi(s;\, \theta_\pi)$ arbitrary
Initialize target actor parameters $\theta_{\pi'} \leftarrow \theta_\pi$
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        $a \leftarrow \pi(s;\, \theta_\mu) + \mathcal{N}$
        execute action a, observe reward r' and next state s'
        store experience $(s,\, a,\, r',\, s')$ in $\mathcal{R}$
        sample random minibatch of N experiences $(s_i,\, a_i,\, r'_i,\, s'_i)$ from $\mathcal{R}$
        update critic by minimizing loss:
        $L = \dfrac{1}{N}\sum_{i=1}^{N}(r'_i + \gamma Q(s'_i,\, \pi(s';\, \theta_{\pi'});\, \theta_{Q'}) - Q(s_i,\, a_i;\, \theta_Q))^2$
        update actor using sampled policy gradient:
        $\nabla_{\theta_\pi} J \approx \dfrac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_\pi}\pi(s_i;\, \theta_\pi)\nabla_a Q(s_i,\, a;\, \theta_Q)|_{a=\pi(s_i;\, \theta_\pi)}$
        update target actor and critic:
        $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$
        $\theta_{\pi'} \leftarrow \tau\theta_\pi + (1-\tau)\theta_{\pi'}$
        $s \leftarrow s'$
    end for
end for

2.3.2　Model-Based 強化学習

model-based 強化学習とは，その名の通り agent が環境の model を持つ強化学習の手法である．2.2.3 節および 2.3.1 節で述べた手法は，環境についての事前知識を必要とせず，かつ環境のモデルを持たない model-free な手法である．なお，強化学習における model-based/model-free という概念は，制御工学におけるそれとは異なっている場合が多いので注意が必要である．一般に，model-based 制御とは，事前に与えられた model を用いた model-based 設計 (Model-Based Desgin, MBD) によって制御器を設計する手法を指す一方で，model-based 強化学習における model-based という言葉は，agent が環境のモデルを持っているという程度の意味しか持たず，そのモデルが事前に与えられたものか，agent と環境の相互作用の中で事後的に構築されたものかについては言及していない．言い換えれば，model-based 制御は (ほとんどの文脈において) 環境についての事前知識を必要とする手法であり，model-based 強化学習は環境についての事前知識を必要する場合と，必要としない場合の両方[*22]が考えられる．

model-based 強化学習の例としては，各状態行動対の訪問回数を記録して学習を効率化する R-MAX[8)]，モデル化にに Gaussian 過程回帰を用いる PILCO[10)]，その他に E^3 [47)]，MBIE[48)] があるが，以下では多くの強化学習の手法との組み合わせが可能な枠組みである Dyna[50)] について述べる．

2.3.2.1　Dyna

Dyna とは，過去の経験 (状態 s，行動 a，次の状態 s'，報酬 r') に基づいて環境をモデル化し，環境との相互作用によって得られた経験に加えてモデルから生成した経験を学習に用いる枠組みを指す．Dyna の概念図を Fig. 2.3 に示す．環境との相互作用によって得られた経験による学習は，2.2 節で述べた通常の強化学習 (direct RL) そのものであり，Dyna の特徴であるモデルから生成した経験による学習は planning と呼ばれる．Fig. 2.3 からも分かるように，Dyna は通常の強化学習に modeling と planning を組み合わせることで容易に実現することができる．具体的例として，Q-learning に Dyna を組み合わせた Dyna-Q の擬似コードを Algorithm 11 に示す．Q-learning との差異は，9 行の modeling と，11 行から 15 行の planning のみである．ここで問題となるのは modeling の手法であるが，最も単純なものは table 型のモデルを用いる方法である．例えば，状態 s において行動 a を選択した回数を $N(s, a)$ とすれば，状態遷移のモデル $\hat{T}(s, a, s')$ を次のように構成することができる．

$$\hat{T}(s, a, s') = \frac{1}{N(s, a)} \sum_{t=1}^{T} 1(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{2.3.3}$$

[*22]実際にはほとんど model-based 強化学習は，事前知識無しで事後的に数理モデルを構築するものである．

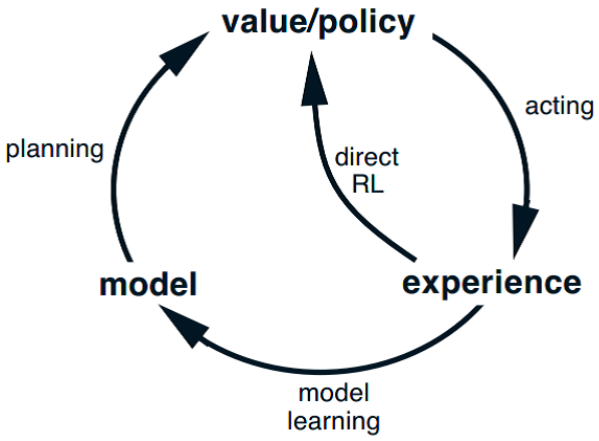

Figure 2.3: 学習，planning，行動間の関係 [50)]

Algorithm 11 Dyna-Q

Require: learning rate α, discount factor γ

Initialize action-value function $Q(s, a)$ arbitrary
Initialize model $M(s, a)$ arbitrary
for each episode **do**
    Initialize state s arbitrary
    for each step **do**
        Choose action a from s using policy derived from $Q(s, a)$
        Execute action a, observe reward r' and next state s'
        $Q(s, a) \leftarrow Q(s, a) + \alpha(r' + \gamma \max_{a'} Q(s', a') - Q(s, a))$
        $M(s, a) \leftarrow r', s'$
        $s \leftarrow s'$
        loop n times
            $\hat{s} \leftarrow$ random previousy observed state
            $\hat{a} \leftarrow$ random action previously taken in $\hat{s}$
            $\hat{r}', \hat{s}' \leftarrow M(\hat{s}, \hat{a})$
            $Q(\hat{s}, \hat{a}) \leftarrow Q(\hat{s}, \hat{a}) + \alpha(\hat{r}' + \gamma \max_{\hat{a}'} Q(\hat{s}', \hat{a}') - Q(\hat{s}, \hat{a}))$
        end loop
    end for
end for

2.3.3　安全な強化学習

強化学習の応用上の問題点は，探索の過程において catastrophic な状況に陥る可能性が多分にある点にある．例えば Q-learning では，行動価値関数 $Q(s, a)$ を乱数で初期化した上で探索および学習を行っているが，これは学習の初期にはほぼランダムな行動が選択されることを意味する．多くの強化学習の手法は累積報酬の期待値の最大化を目的としているため，探索の過程におけるリスクは軽視されているがが，実問題においては，起こる確率は小さいが大きな損失が発生するような状況を回避することが望ましい場合が多い．安全な強化学習 (Safe RL) とは，これらの強化学習の欠点を改善しようとする試みである．

García and Fernández[13] によれば，安全な強化学習は大きく分けて 2 つに分類される．1 つ目は目的関数を変更する，すなわち価値関数の定義や更新式そのものに手を加える手法であり，2 つ目は探索または価値関数に事前知識を知識を組み込む手法である．

1 つ目の代表例としては，例えば Q̂-learning[18] や β-pessimistic Q-learning[14] のような minmax 戦略がある．

Algorithm Definition 2.9 (Q̂-learning)
次の更新式を用いて，累積報酬の期待値の最小値を最大化するような手法を Q̂-learning と呼ぶ．

$$\hat{Q}(s, a) \leftarrow \min(\hat{Q}(s, a), r' + \gamma \max_{a'} \hat{Q}(s', a')) \tag{2.3.4}$$

Remark. Q̂-learning は極端に悲観的な，すなわち最悪の状態遷移が起こることを想定して状態価値を見積もる手法である．目的関数は累積報酬の期待値の最小値となっているため，保守的な行動選択が行われる．

Algorithm Definition 2.10 (β-pessimistic Q-learning)
次式のように parameter β を導入することにより，極端に楽観的な Q-learning と極端に悲観的な Q̂-learning の中間的な振る舞いを実現した手法を β-pessimistic Q-learning と呼ぶ．

$$Q_\beta(s, a) \leftarrow Q_\beta(s, a) + \alpha(r' + \gamma((1-\beta)\max_{a'} Q_\beta(s', a') + \beta \min_{a'} Q_\beta(s', a')) - Q_\beta(s, a)) \tag{2.3.5}$$

Remark. $\beta = 0$ のとき通常の Q-learning と一致する．

その他の手法としては，遷移関数の不確かさを考慮した robust 戦略 [15,51] や，非線形な目的関数 (指数効用関数) を用いる手法 [22] がある．

2 つ目の事前知識の組み込みとしては，価値関数の初期化において事前知識を組み込む手法，方策に対する制約を組み込む手法，人間による demonstration を利用する手法，強化学習に教師付き学習を組み合わせた手法 (supervised RL)[23,37] などがある．この内，価値関数の初期化において事前知識を組み込む手法について第 4 章で詳しく述べる．

Chapter 3　No Free Lunch定理

第 2 章では，最適制御問題の数値解法に関して 1. 間接法と直接法 2. 事前知識を用いるか否かという 2 つの観点を提示し，間接法のうち Bellman 方程式に基づく手法について事前知識を用いるものと，事前知識を用いないもの双方の代表例を示した．

ところで，直接法とは最適制御問題を数理計画問題に帰着させて解く手法であったが，本章で議論の対象とする No Free Lunch (NFL) 定理は，事前知識を用いない数理計画問題の数値解法である metaheuristics に関するものである．meta (高次の) heuristics (発見的手法) という名称からも分かるように，事前知識を用いないということは，特定の問題に依存しない，すなわち汎用的な手法であるということを意味する．metaheuristics の具体的な例としては，遺伝的アルゴリズム (Genetic Algorithm, GA)[19]，焼きなまし法 (Simulated Annealing, SA)[30,39] 等[*1]がある．metaheuristics の性能を例えば反復回数や計算時間とすれば，ある特定の問題を異なる 2 つの手法で解いたとき，性能の優劣が生じることは自明であろう．では，考えうる全ての問題に対する性能について，ある手法が他のある手法を全て上回るということは起こり得るだろうか．探索における NFL 定理は，そのようなことは起こり得ない[*2]と主張する．Wolpert and Macready[57,58] の言葉を引用すれば，“ 評価関数の極値を (事前知識を用いずに) 探索する全てのアルゴリズムは，考えうる全ての評価関数について平均化した場合，厳密に等しい性能を示す ” ことを示している．

[*1] その他の例としては，蟻コロニー最適化 (Ant Colony Optimization, ACO)[11]，人工蜂コロニーアルゴリズム (Artificial Bee Colony algorithm, ABC algorithm)[25]，人工免疫システム (Artificial Immune System, AIS)[27] 等がある．自然現象や生物に着想を得た手法が多いのも 1 つの特徴といえる．

[*2] 仮に起こり得るとすれば，性能の改善を繰り返すことで，いずれは汎用的かつ高性能な万能の手法が得られるはずである．

3.1　数理計画問題の数値解法

数理計画問題とは，与えられた制約条件もとで，解の候補の中から “好ましい” ものを見つけ出すような問題である．“好ましさ” の尺度を目的関数あるいは評価関数と呼び，解候補 $x \in \mathcal{X}$ から評価値 $y \in \mathcal{Y}$ への写像 $f : \mathcal{X} \to \mathcal{Y}$ で表される．数理計画問題中でも最も初歩的なものは，制約および評価関数が線形である線形計画問題である．線形計画問題の解法を線形計画法と呼び，代表的な手法に simplex 法 [9] と内点法 [26] がある．より一般的性の高い問題の定式化としては，次の非線形計画問題がある．

Problem 3.1 (非線形計画問題)

$$\begin{aligned} &\underset{x}{\text{maximize}} && f(x) && \\ &\text{subject to} && g_i(x) \leq 0 && \text{(given)} \\ &&& h_j(x) = 0 && \text{(given)} \end{aligned} \tag{3.1.1}$$

非線形計画問題の解法を非線形計画法と呼び，主要な解法の枠組みとして勾配法がある．これらの手法はいずれも目的関数および制約条件が既知であることを前提とした (場合によってはさらに目的関数および制約条件の勾配を必要とする)，事前知識を必要とする手法である．一方で，metaheuristics は目的関数や制約条件そのものの関数形は必要ではなく，black box としての目的関数が存在しさえすればよい．

3.2　No Free Lunch 定理

3.2.1　探索における No Free Lunch 定理

探索における NFL 定理 [31,57,58] は，考えうる全ての問題に対する metaheuristics の平均性能は等しい，すなわち万能な metaheuristics は存在しないことを主張する．

Theorem 3.1 (探索における No Free Lunch 定理)
解候補 x および評価値 y の集合をそれぞれ $\mathcal{X}$，$\mathcal{Y}$，目的関数の集合を $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$ とし，k $(k < |\mathcal{X}|)$ 回評価が行われたときの探索履歴 $d_k \in \mathcal{D}_k$ を $d_k = \{d_k^x, d_k^y\} = \{\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_k, y_k\}\}$ と定義する．このとき metaheuristics $\mathscr{A}$ は，一般に探索履歴から解候補への写像 $\mathscr{A} : \mathcal{D} \to \mathcal{X}$ と見做すことができるが，ここでは既に訪れた解候補との重複を避けるような metaheuristics $\mathscr{A} : \mathcal{D} \to \{x \mid x \notin d^x\}$ を仮定する．このとき評価値の探索履歴 d_k^y について以下が成り立つ．

$$\forall \mathscr{A}_1 \, \forall \mathscr{A}_2 \, \forall k \sum_f P(d_k^y \mid f, k, \mathscr{A}_1) = \sum_f P(d_k^y \mid f, k, \mathscr{A}_2).$$

Proof. Appendix A.10 に示す．
Remark. NFL 定理の主張を正確に述べると，任意の評価値の探索履歴 d_k^y が得られる確率は，metaheuristics $\mathscr{A}$ に依存しないというものである．数理計画問題の数値解法の性能，例えば収束の速さ，探索の安定性などは，当然 d_k^y に依存するはずであるから，NFL 定理を根拠として全ての metaheuristics の平均性能は等しいことが主張できる．

3.2.2　定理の解釈

Fig. 3.1 に示すように，探索における NFL 定理の主張は，あらゆる問題に対して高い性能を示す万能な metaheuristics の存在を否定するものであるが，同時に，可能な限り問題領域を限定して特定の問題に特化した手法を利用すべきであるということも示唆している．そして，問題領域を限定するものは，まさに問題についての事前知識そのものである．

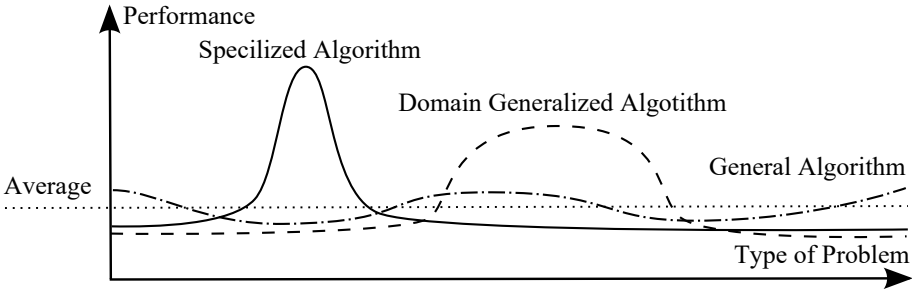

Figure 3.1: Concept of No Free Lunch Theorem for serch

3.2.3　証明の補足

更に理解を深めるために，例として $|\mathcal{X}| = 3$，$|\mathcal{Y}| = 2$ の場合を例に証明の補足を行う．解候補および評価値の集合をそれぞれ $\mathcal{X} = \{x^0, x^1, x^2\}$，$\mathcal{Y} = \{y^0, y^1\}$ とすれば，可能な評価関数 $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$ は $|\mathcal{Y}|^{|\mathcal{X}|} = 2^3 = 8$ 個ある．この場合の f を列挙したものを Table 3.1 に示す．表の第 1 列は写像の始域を，第 2 列以降は第 1 行の f による像を表している．なお，写像の全射性や単射性については何ら仮定を置いておらず，全ての可能な写像について考えている (実際に f^0 と f^7 は全射でも単射でも無い)．つまり $\mathcal{Y}$ の要素全てが f の像として現れなくてもよいし，異なる解候補 x に対して，評価値 y が重複していてもよい．

Table 3.1: All possible f when $|\mathcal{X}| = 3$, $|\mathcal{Y}| = 2$

	f^0	f^1	f^2	f^3	f^4	f^5	f^6	f^7
x^0	y^0	y^1	y^0	y^1	y^0	y^1	y^0	y^1
x^1	y^0	y^0	y^1	y^1	y^0	y^0	y^1	y^1
x^2	y^0	y^0	y^0	y^0	y^1	y^1	y^1	y^1

式 (A.10.1) で示すように，y_1 を 1 つ決めたときの $\sum_f P(d_k^y \mid f, k, \mathscr{A})$ の値は $\sum_f P(y_1 \mid f(x_1)) = |\mathcal{Y}|^{|\mathcal{X}|-1}$ であった．例えば $d_1 = \{x^0, y^0\}$ を満たすような f は，残りの $\mathcal{X} \setminus x^0$ と $\mathcal{Y}$ への対応関係のみを考えればよく，Table 3.2 に示すように確かに $|\mathcal{Y}|^{|\mathcal{X}|-1} = 2^{3-1} = 4$ 個存在することが分かる．当然ながら任意の d_1 についても同様のことがいえて，例えば $d_1 = \{x^0, y^1\}$ ならば f^1, f^3, f^5, f^7，$d_1 = \{x^2, y^1\}$ ならば f^4, f^5, f^6, f^7 というように，d_1 を満たす f が絞られる．

Table 3.2: Possible f when $d_1 = \{x^0, y^0\}$

	f^0	f^2	f^4	f^6
x^0	y^0	y^0	y^0	y^0
x^1	y^0	y^1	y^0	y^1
x^2	y^0	y^0	y^1	y^1

さらに k を増やすと，$d_2 = \{\{x^0, y^0\}, \{x^2, y^1\}\}$ ならば f^4 と f^6，$d_3 = \{\{x^0, y^0\}, \{x^2, y^1\}, \{x^1, y^0\}\}$ ならば f^4 というように，d_k を満たす f の数は $1/2$ ずつ減っていることが分かる．探索における No Free Lunch 定理はこれらを一般化したものであり，d_k の要素が重複しない限り，d_k を満たすような f の減り方は x の選び方に依らず一定だということを示している．ここで，d_k の要素が重複しないという条件は，d_k^x が重複しない，すなわち $x_{k+1} \notin d_k^x$ という条件と等しい．なぜなら具体例を見れば分かるように，異なる x に対して y が重複する可能性は多分にあるが，x が重複しない限り，d_k の要素 $\{x_1, f(x_1)\}, \{x_2, f(x_2)\}, \ldots, \{x_k, f(x_k)\}$ の独立も必ず担保されるからである．

3.2.4　No Free Lunch 定理の拡張

前述の探索における NFL 定理は，解候補 x および評価値 y がそれぞれ有限集合に属する，したがって目的関数 f も有限個であるという条件を前提としている．探索における NFL 定理の拡張は，目的関数 f の集合 $\mathcal{F}$ を縮小または拡大するという背反する 2 つの考え方に基づいている．

探索における NFL 定理が成り立つ，すなわち metaheuristics の平均性能が等しいと主張できるのは，あくまで考えうる全ての f について平均したときだけである．現実世界の問題は $\mathcal{F}$ の部分集合に属すると考えられるが，それについては NFL 定理は何も主張できない．$\mathcal{F}$ の部分集合に関する NFL 定理の拡張の最も重要なものとして，次の Sharpened No Free Lunch 定理 [45] がある．

Theorem 3.2 (Sharpened No Free Lunch 定理)
F が置換について閉じている (closed under permutations, c.u.p.) ときに限り，評価値の探索履歴 d_k^y について以下が成り立つ．

$$\forall \mathscr{A}_1 \, \forall \mathscr{A}_2 \, \forall k \sum_{f \in F} P(d_k^y \mid f, k, \mathscr{A}_1) = \sum_{f \in F} P(d_k^y \mid f, k, \mathscr{A}_2).$$

Remark. Table 3.3 に示すように，例えば 3.2.2 項の例の f_1, f_2, f_4 は置換について閉じた $\mathcal{F}$ の部分集合である．同様に，f_3, f_5, f_6 も置換について閉じている．

Table 3.3: c.u.p f when $|\mathcal{X}| = 3$, $|\mathcal{Y}| = 2$

	f^1	f^2	f^4
x^0	y^1	y^0	y^0
x^1	y^0	y^1	y^0
x^2	y^0	y^0	y^1

探索における NFL 定理のもう 1 つの拡張は，解候補 x および評価値 y が連続な空間においても NFL 定理の結果が保たれるか否かという問題であるが，これについては依然 open problem となっている．連続な空間における NFL 定理の存在を否定する研究として Auger and Teytaud[1, 2]，肯定する研究として Vose[53]，Lockett[34] がある．

強化学習における No Free Lunch 定理については先行研究が極めて少ないが，Everitt[12] は歪んだ報酬のもとでの MDP (Corrupt Reward MDP) における regret の下界を明らかにしている．

Chapter 4　強化学習への事前知識の組み込み

第 3 章では数理計画問題の数値解法のうち，事前知識を必要としない metaheuristics の，全ての問題に対する平均性能は等しいことを主張する No Free Lunch 定理について述べた．第 2 章で述べたように，強化学習は基本的には事前知識を必要としない最適制御問題の解法であるが，強化学習のような動的な問題に対しても No Free Lunch 定理が成立するかどうかは自明ではない．

本章ではまず，強化学習における事前知識と制御性能の関係について考察する．第 2.3.2 項で，安全な強化学習には目的関数を変更する手法と，方策または価値関数に事前知識を組み込む手法の 2 通りがあると述べた．前者の目的関数そのものを変更する手法は，本来の最適制御の目的である累積報酬の最大化とは目的が全く異なる (むしろ robust 制御に近い) ため，安全な強化学習の考え方に従えば，最適制御問題の数値解法としての強化学習の学習過程を安定させ，性能を向上させるには，何らかの方法で事前知識を組み込む他ないということになる．仮に強化学習においても No Free Lunch 定理が成り立つのならば，強化学習の性能を向上させるには事前知識を組み込む他ないという主張が支持されることになる．

次に，実際に強化学習に事前知識を組み込む (= 問題領域を限定する) ことにより，システム制御に特化した強化学習アルゴリズムの構築を目指す．そのためのアプローチとして，事前知識の多寡に応じて次の 2 通りについて検討する．

1. 環境に関する知識の組み込み
   制御問題においては遷移関数がある程度既知であり，報酬は設計者が定義する場合が多い．本論文では，これらを環境についての事前知識と呼ぶことにし，対象の近似モデルが既知かつ報酬が二次形式で表される場合，すなわち環境についての事前知識が (不完全であっても) 利用可能な場合に，Riccati 方程式の解を利用した学習効率の改善を試みる．

2. ドメイン知識の利用
   一般にドメインとは一定の広さを持つ問題領域を指すが，制御問題の多くにみられる，状態遷移が決定論的であるという前提そのものも，一種のドメイン知識であると考えられる．model-based 強化学習の代表的な枠組みである Dyna と，これらの仮定を組み合わせることで，決定論的環境に特化した手法を提案する．

4.1　強化学習における No Free Lunch 定理

探索における No Free Lunch 定理は，目的関数 f を解候補 x から評価値 y への写像 $y = f(x)$，metaheuristics $\mathscr{A}$ を探索履歴 d_k から解候補 x_{k+1} への写像 $x_{k+1} = \mathscr{A}(d_k)$ としたとき，任意の評価値の履歴 d_k^y が得られる確率は metaheuristics $\mathscr{A}$ というものであった．このとき各 step における解候補 x，評価値 y，探索履歴 d_k 相互の関係を書き下すと次のようになる．

$$\begin{aligned} y_1 &= f(\mathscr{A}()) = f(x_1) \\ y_2 &= f(\mathscr{A}(x_1, y_1)) = f(x_2) \\ &\vdots \\ y_{k+1} &= f(\mathscr{A}(d_k)) = f(x_{k+1}) \end{aligned} \tag{4.1.1}$$

一方で，最適制御問題における強化学習 agent は，例えば Q-learing であれば内部状態として行動価値関数 $Q(s, a)$ や学習率 α 等を保持しているが，入出力だけに着目すれば，過去の経験 (状態 s，行動 a，次の状態 s'，報酬 r') を入力として行動を出力する写像であるといえる．

$$\begin{aligned} s_1 &= T(s_0, \mathscr{A}()) = T(s_0, a_0) \\ s_2 &= T(s_1, \mathscr{A}(s_0, a_0, s_1, r_1) = T(s_1, a_1)) \\ &\vdots \\ s_{t+1} &= T(s_t, \mathscr{A}(\{s_0, a_0, s_1, r_1\}, \dots \{s_{t-t}, a_{t-1}, s_t, r_t\})) = T(s_t, a_t) \end{aligned} \tag{4.1.2}$$

ここで，決定論的な報酬 $r_{k+1} = R(s_k, a_k, s_{k+1})$ が既知であると仮定すれば，agent の作用は結局のところ

$$\begin{aligned} s_{t+1} &= T(s_t, \mathscr{A}(\{s_0, a_0, s_1\}, \dots \{s_{t-t}, a_{t-1}, s_t\})) \\ &= T(s_t, \mathscr{A}(d_t^s, d_{t-1}^a)) = T(s_t, a_t) \end{aligned} \tag{4.1.3}$$

となる．ここで，状態行動対 (s_t, a_t) を解候補 x，遷移先の状態 s_{t+1} を評価値 y と見做せば，強化学習の問題設定は，数理計画問題のそれと類似しているように見える．ただし，任意の解候補を選べない，つまり，数理計画問題では目的関数 f の引数は解候補 x のみであるのに対して，強化学習では遷移関数 T の引数が s_t と a_t の 2 つであり，a_t は agent によって任意に選ぶことができても，引数全体 (状態行動対) は任意に選ぶことができず，常に s_t に制限されるという点で異なっている．

しかしながら，状態行動対 (s_t, a_t) を，数理計画問題における解候補 x と見做したとして，仮に解候補 x が制限されたとしても，NFL 定理は保持される．No Free Lunch 定理は，解候補 x が選択される確率については言及しておらず，既に訪れた解候補との重複を避ける限り成立する (実際にはこの制約も Wolpert and Macready は取り除けると述べている．敢えて revisit するようなアルゴリズムは考慮する必要が無く，重複を取り除いた探索履歴のみがアルゴリズムの性能に関係するからである)．そもそも TD 法に属する強化学習は，その収束条件において，任意の状態行動対 (s_t, a_t) を無限回訪問することを要請している (Theorem 2.3, 2.4, 2.5 参照)．したがって，厳密な証明は省くが，TD 法に属する強化学習に対して，探索における No Free Lunch 定理は容易に拡張することが可能である．

4.2　環境に関する知識の組み込み

制御や強化学習における事前知識とは，制御対象の数理モデル，すなわち運動方程式や (確率的な) 遷移関数，有限オートマトンなどの状態遷移モデルと，そのパラメータである．多くの問題では完全な事前知識は得られない (得られるならば動的計画法で Bellman 方程式を直接解けば良い) 上に，実世界においてはそもそも完全な事前知識など存在せず，必ず実対象とモデルの差異，すなわちモデル化誤差が存在する．

モデル化誤差には大きく分けてパラメータの不確かさと，モデルの構造そのもの (線形あるいは非線形であるかなど) の不確かさがある．前者に対してはロバスト制御や適応制御，モデル予測制御などが有効であるが，制御器の構造がモデルに大きく依存するが故に，モデルの構造の不確かさには脆弱である．したがって，モデルを必要としない強化学習に，モデルの構造に依存しない形で事前知識を組み込むことが出来れば，事前知識を活用しつつ，かつモデルの構造の不確かさに対してもロバストあるいは適応的に振る舞うことが可能な，強力なアルゴリズムと成りうると考えられる．

本項の目的は，本来は事前知識を必要としない強化学習に対して，事前知識を組み込むことによる学習の効率化や安定性の保証を図ることである．本稿では事前知識として制御対象の大まかなモデル (離散時間線形モデル) が与えられ，かつ評価関数が二次形式で記述される場合に，事前知識に基づき最適な行動価値関数 $Q(s, a)$ を予め計算した上で，それを初期値として Q-learning を行った場合の挙動を調べることを目的とする．

4.2.1　LQR 問題

離散時間システムおける LQR 問題とは，離散時間線形システムに対する全状態フィードバック

$$\boldsymbol{s}_{t+1} = \boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{a}_t \tag{4.2.1}$$

$$\boldsymbol{a}_t = -\boldsymbol{K}\boldsymbol{s}_t \tag{4.2.2}$$

に対して，二次形式評価関数

$$J = -\sum \boldsymbol{s}_t^{\mathrm{T}}\boldsymbol{S}\boldsymbol{s}_t + \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{R}\boldsymbol{a}_k \tag{4.2.3}$$

を最大化するようなゲイン $\boldsymbol{K}$ を求める問題である．線形な状態方程式と二次形式評価関数を仮定することにより，最適 Bellman 方程式は Riccati 代数方程式

$$\boldsymbol{P} = \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P})\boldsymbol{A} + \boldsymbol{S} \tag{4.2.4}$$

に帰着され，正定解 $\boldsymbol{P}$ を用いて

$$\boldsymbol{K} = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{B} + \boldsymbol{R})^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{A} \tag{4.2.5}$$

が得られる．このとき，フィードバック則 $\boldsymbol{a}_t = -\boldsymbol{K}\boldsymbol{s}_t$ のもとでの最適状態価値関数 $V^*(\boldsymbol{s}_t)$ は，

$$V^*(\boldsymbol{s}_t) = \boldsymbol{s}_t^{\mathrm{T}}\boldsymbol{P}\boldsymbol{s}_t \tag{4.2.6}$$

であるので，$Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t)$ は，

$$\begin{aligned} Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) &= r_t + \gamma V^*(\boldsymbol{s}_{t+1}) \\ &= -(\boldsymbol{s}_t^{\mathrm{T}} \boldsymbol{S} \boldsymbol{s}_t + \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}_k) \\ &\quad + \gamma(\boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{a}_t)^{\mathrm{T}} \boldsymbol{P}(\boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{a}_t) \\ &= \begin{bmatrix} \boldsymbol{s}_t \\ \boldsymbol{a}_t \end{bmatrix}^{\mathrm{T}} \boldsymbol{H} \begin{bmatrix} \boldsymbol{s}_t \\ \boldsymbol{a}_t \end{bmatrix} \end{aligned} \tag{4.2.7}$$

$$\boldsymbol{H} = \begin{bmatrix} -\boldsymbol{S} + \gamma \boldsymbol{A}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{A} & \gamma \boldsymbol{A}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{B} \\ \gamma \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{A} & -\boldsymbol{R} + \gamma \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{B} \end{bmatrix} \tag{4.2.8}$$

となり，二次形式になることが確認できる．

4.2.2　事前知識の組み込み

環境に関する事前知識の組み込みとして，二次形式評価関数と線形近似モデルに基づいて計算した $Q(s, a)$ を初期値として学習を行う．LQR と Q-learning の組み合わせについては，LQR 問題においては $Q(s, a)$ も二次形式になるという性質を利用して，推定を簡略化する方法が提案されている [3,7] が，本稿ではモデルの構造に依存しない形で事前知識を組み込むことを目的とするため，真の $Q(s, a)$ の構造を予め規定することはしない．

対象の近似モデルおよび二次形式評価関数

$$\boldsymbol{s}_{t+1} = \boldsymbol{A}\boldsymbol{s}_t + \boldsymbol{B}\boldsymbol{a}_t \tag{4.2.9}$$

$$r_t = -(\boldsymbol{s}_t^{\mathrm{T}} \boldsymbol{S} \boldsymbol{s}_t + \boldsymbol{a}_t^{\mathrm{T}} \boldsymbol{R} \boldsymbol{a}_t) \tag{4.2.10}$$

が既知であるとすると，事前知識に基づく Bellman 方程式の近似解は，代数的 Riccati 方程式の唯一の正定解 $\boldsymbol{P}$ を利用して次のように書ける．

$$Q^*(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{bmatrix} \boldsymbol{s}_t \\ \boldsymbol{a}_t \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\boldsymbol{S} + \boldsymbol{A}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{A} & \boldsymbol{A}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{B} \\ \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{A} & -\boldsymbol{R} + \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_t \\ \boldsymbol{a}_t \end{bmatrix} \tag{4.2.11}$$

強化学習において式 (4.2.11) を初期解として用いることにより，事前知識に基づく安定性および最低限の制御性能の保証をした上で，経験的な知識を基に徐々に真の対象に対する最適解を学習するような振る舞いが期待される．

4.3　ドメイン知識の利用

第 2 章で述べたように，Dyna とは，過去の経験を基に環境のモデルを構成し，実際の経験に加えてモデルから取得した経験も学習に用いる枠組みを指す．Dyna に関する研究の多くは，確率論的環境を前提として modeling と planning を行っているが，仮に環境が決定論的であるという事前知識のみを組み込んだ手法として，過去の経験そのものを table 型の入出力モデルと見做し，planning に利用する tabular Dyna と深層強化学習を組み合わせた手法を提案する．これは experience replay に類似する概念であるが，1 制御周期あたりに複数回 planning を行う点で experience replay とは異なっている．

深層強化学習と Dyna の組み合わせについては，DDQ (Deep Dyna-Q)[38]，Dyna-DDPG[59] などの先行研究があるが，いずれも確率論的環境を前提としたものであり，決定論的環境の対象に直接用いるには適さない．本論文では，DQN および DDPG と tabular Dyna を組み合わせたアルゴリズムを提案し，環境が決定論的であることが事前知識として与えられている場合における性能を評価する．

4.3.1　Tabular Dyna DQN/DDPG

本論文の提案手法である tabular DDQ と tabular Dyna DDPG の擬似コードを Algorithm 12 および Algorithm 13 に示す．tabular DDQ では第 15 行から第 19 行が，tabular Dyna DDPG が第 20 行から第 26 行が planning に相当する．これらの手法はモデルの fitting を行わないという点でオリジナルの Dyna とは異なっており，1 制御周期に複数回 batch 学習を行うという点で experience replay とも異なっている．

また，Dyna と深層強化学習の組み合わせにあたり，target network の更新を 1 時間 step に 1 回のみと工夫している．2.3.1 項で述べたように，深層強化学習では fixed target network なる工夫を用いて target network を一定の期間固定または smoothing 行うことで学習の不安定化を防いでいるが，仮に Dyna との組み合わせにおいて planning 部でも target の更新を行うと，1 時間 step あたりの target network および behavior network の変化が過大となり，学習に悪影響を及ぼす恐れがある．この現象は，actor (制御則) を NN で表現する DDPG で特に顕著であるから，planning において敢えて target の更新を行わないことで，1 時間 step あたりの actor の大幅な変動を防ぎ，学習を安定化させている．

4.3.2　確率論的環境における Dyna との比較

確率論的環境における tabuler Dyna の最も単純なモデル化手法は，Monte Carlo 法 (MDP の数値解法としての MC 法ではなく，広義の MC 法) を用いて状態 s において行動 a を選択した回数を $N(s, a)$ として，状態遷移のモデル $\hat{T}(s, a, s')$ を次のように表すことである．

$$\hat{T}(s, a, s') = \frac{1}{N(s, a)} \sum_{t=1}^{T} 1(s_{t+1} = s' \mid s_t = s, a_t = a) \tag{4.3.1}$$

仮に決定論的環境でこの手法を用いれば，ある状態行動対 (s, a) の遷移先は常に s' となるから，遷移確率 $\hat{T}(s, a, s')$ は 1/1, 2/2, 3/3, ... として決定論的なモデルが得られるため，モデルの挙動そのものは提案手法と等価である．しかしながら，状態行動対 (s, a) の訪問回数 $N(s, a)$ を保持するために，最大で $|\mathcal{S}| \times |\mathcal{A}|$，モデル $\hat{T}(s, a, s')$ を保持するために最大で $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$ のメモリが必要となり，状態行動空間が大きい場合には計算資源の観点で現実的ではない．また，先行研究である DDQ のように NN を用いてモデルを近似的に表現すればメモリは少なくて済むが，今度はモデル化を行うための計算量と，モデル化誤差が学習に悪影響を及ぼす可能性が問題となってしまう．

Algorithm 12 tabular DDQ

Require: discount factor γ, epsilon ϵ, smoothing factor τ

Initialize critic $Q(s, a; \theta_Q)$ arbitrary
Initialize target critic parameters $\theta_{Q'} \leftarrow \theta_Q$
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        Choose action a from s using policy derived from $Q(s, a)$
        Execute action a, observe reward r' and next state s'
        Store experience (s, a, r, s') in $\mathcal{R}$
        Sample random minibatch of N experiences (s_i, a_i, r_i, s'_i) from $\mathcal{R}$
        Update critic by minimizing loss:
        $L = \frac{1}{N} \sum_{i=1}^{N} (r'_i + \gamma \max_{a'} Q(s'_i, a'; \theta_{Q'}) - Q(s_i, a_i; \theta_Q))^2$
        Update target critic:
        $\theta_{Q'} \leftarrow \tau\theta_Q + (1 - \tau)\theta_{Q'}$
        $s \leftarrow s'$
        loop n times
            Sample random minibatch of N experiences (s_i, a_i, r_i, s'_i) from $\mathcal{R}$
            Update critic by minimizing loss:
            $L = \frac{1}{N} \sum_{i=1}^{N} (r'_i + \gamma \max_{a'} Q(s'_i, a'; \theta_{Q'}) - Q(s_i, a_i; \theta_Q))^2$
        end loop
    end for
end for

Algorithm 13 tabular Dyna DDPG

Require: discount factor γ, smoothing factor τ

Initialize critic $Q(s,\, a;\, \theta_Q)$ arbitrary
Initialize target critic parameters $\theta_{Q'} \leftarrow \theta_Q$
Initialize actor $\pi(s;\, \theta_\pi)$ arbitrary
Initialize target actor parameters $\theta_{\pi'} \leftarrow \theta_\pi$
for each episode **do**
    Initialize state s arbitrary
    for each step of episode **do**
        $a \leftarrow \pi(s;\, \theta_\mu) + \mathcal{N}$
        execute action a, observe reward r' and next state s'
        store experience $(s,\, a,\, r',\, s')$ in $\mathcal{R}$
        sample random minibatch of N experiences $(s_i,\, a_i,\, r'_i,\, s'_i)$ from $\mathcal{R}$
        update critic by minimizing loss:
        $$L = \frac{1}{N}\sum_{i=1}^{N}(r'_i + \gamma Q(s'_i,\, \pi(s';\, \theta_{\pi'});\, \theta_{Q'}) - Q(s_i,\, a_i;\, \theta_Q))^2$$
        update actor using sampled policy gradient:
        $$\nabla_{\theta_\pi} J \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_\pi}\pi(s_i;\, \theta_\pi)\nabla_a Q(s_i,\, a;\, \theta_Q)|_{a=\pi(s_i;\, \theta_\pi)}$$
        update target actor and critic:
        $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$
        $\theta_{\pi'} \leftarrow \tau\theta_\pi + (1-\tau)\theta_{\pi'}$
        $s \leftarrow s'$
        loop n times
            sample random minibatch of N experiences $(s_i,\, a_i,\, r'_i,\, s'_i)$ from $\mathcal{R}$
            update critic by minimizing loss:
            $$L = \frac{1}{N}\sum_{i=1}^{N}(r'_i + \gamma Q(s'_i,\, \pi(s';\, \theta_{\pi'});\, \theta_{Q'}) - Q(s_i,\, a_i;\, \theta_Q))^2$$
            update actor using sampled policy gradient:
            $$\nabla_{\theta_\pi} J \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_\pi}\pi(s_i;\, \theta_\pi)\nabla_a Q(s_i,\, a;\, \theta_Q)|_{a=\pi(s_i;\, \theta_\pi)}$$
        end loop
    end for
end for

Chapter 5　数値計算例

本章では，第 4 章で提案した手法の実施例として，環境に関する事前知識の組み込みについては 2 次遅れ系の制御，ドメイン知識の利用についてはより応用的な例として宇宙機の姿勢制御と航空機の姿勢制御に提案手法を適用したときの数値計算例を示す．

5.1　環境に関する事前知識の組み込み

ここでは Q-learning に対して提案手法である行動価値関数 $Q(s, a)$ の初期化を適用した上で，double integrator の制御によりその性能を検証する．

5.1.1　2 次遅れ系の制御

5.1.1.1　運動方程式

本実施例では，実際の環境が次の連続な決定論的遷移

$$\ddot{s} + ks = a \tag{5.1.1}$$

および決定論的報酬

$$r = -(s^2 + \dot{s}^2 + 0.1a^2) \tag{5.1.2}$$

に従うと仮定する．遷移関数を状態空間モデルで書くと

$$\bar{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 \\ -k & -1 \end{bmatrix} \tag{5.1.3}$$

$$\bar{\boldsymbol{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.1.4}$$

となる．これらを制御周期 Δt で 0 次ホールドで離散化すれば，離散時間線形モデル

$$\boldsymbol{A} = \mathrm{e}^{\bar{\boldsymbol{A}}\Delta t}, \quad \boldsymbol{B} = \boldsymbol{B}\int_0^{\Delta t} \mathrm{e}^{\bar{\boldsymbol{A}}\lambda}\mathrm{d}\lambda \tag{5.1.5}$$

が得られる．本数値計算例ではこのモデルが何らかの方法によって事前に得られたと仮定して，真の環境に対する方策を学習する．

5.1.1.2　学習条件

学習においては，制御周期 $\Delta t = 0.1$ [s] で 0 次ホールドされた状態遷移モデルが事前に与えられるとし，真の環境の状態の遷移は $\Delta t/2$ の周期で Runge-Kutta 法を用いて計算した．また，本数値計算例で事前知識の組み込みの対象とする Q-learning は，離散状態かつ離散状態にしか対応していない手法であるから，学習に際しては観測値および行動は $\hat{s}$, $\hat{\dot{s}}$, $a = [-1, -0.8, -0.6, \ldots, 1]$ のように離散化される．学習は 1 つの episode を 100 step (10 sec.) とし，各 episode における状態変数の初期条件は $[s\ \dot{s}] = [1\ 0]$ とした．行動選択は ϵ-greedy 法を用い，以下の Table に示す hyperparameter で学習を行った．

Table 5.1: Second-Order System, Hyperparameters

prameter	
learning rate	0.5
discount factor	0.99
epsilon	$\epsilon = 0.5 \times 1/\text{Episodes}$

5.1.1.3　結果

今回は，1. 通常の Q-learning (事前知識無し)，2. 正しい事前知識を与えた場合，3. モデル化誤差を含む事前知識 ($k = 0.5$) を与えた場合の全 3case を実施した．それぞれの case の学習前および学習後の制御結果，各 episode における報酬の履歴を Fig. 5.1 から Fig. 5.9 に示す．

case 1 は事前知識を利用していないため，初期の方策はランダムなものとなり，最適な方策を獲得するのに約 300 episode を要している．case 2 は正しい事前知識を与えたものであるが，状態の離散化の影響がある (与えた事前知識は連続な状態行動空間に対する価値関数であるが，実際の agent は離散的な状態行動空間を扱う) ため，初期状態は準最適なものとなっている．実際，100 episode 経過後には静定時間が僅かに減少しているが，初期状態においても概ね良好な制御性能を示しているといえる．case 3 は事前知識と実際の環境が異なっている，すなわちモデル化誤差が存在する場合 (適応制御やロバスト制御と同様の問題設定) を想定したものである．初期の想定よりも実際の環境の剛性が小さい ($k = 1$) ため，過大な制御力が発生すると考えられたが，実際には離散化の影響等もあり，静定時間が増加し，case 2 では見られなかったチャタリングも発生している．しかしながら，概ね 20 episode 程度で episode 全体の報酬が上限値に達しており，case 1 に比べれば学習に要する時間は大幅に短縮している．

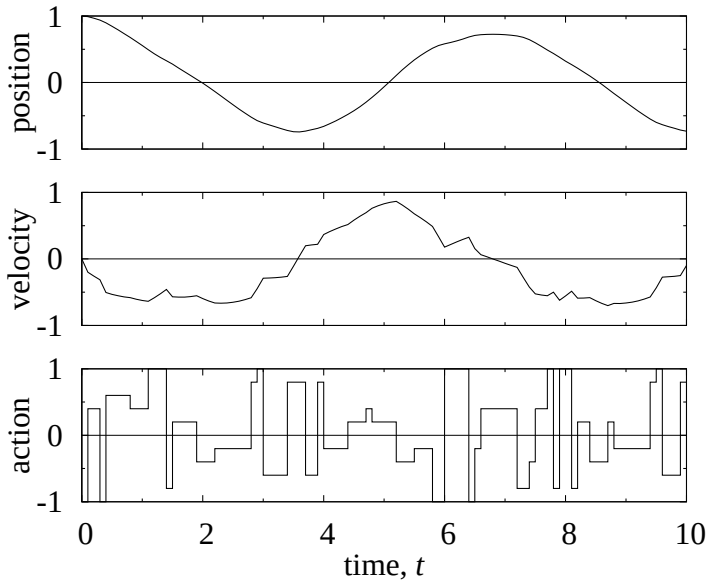

Figure 5.1: SO, Control History before Learning (QL without Prior Knowledge)

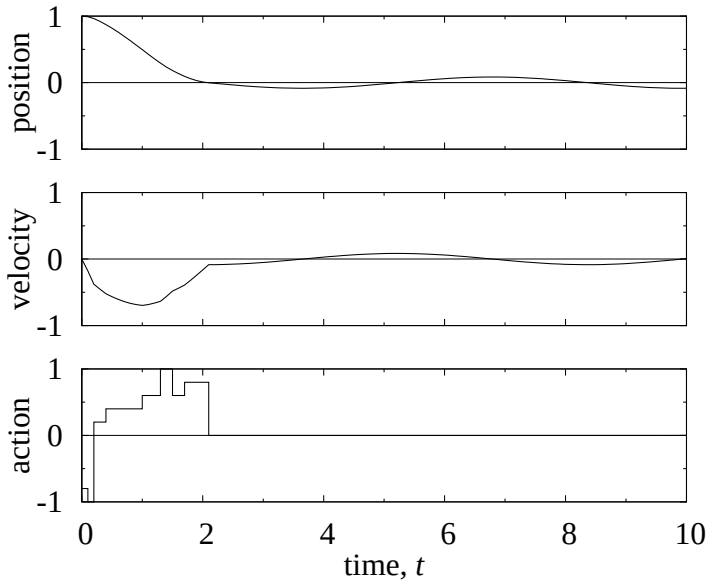

Figure 5.2: SO, Control History after Learning (QL without Prior Knowledge)

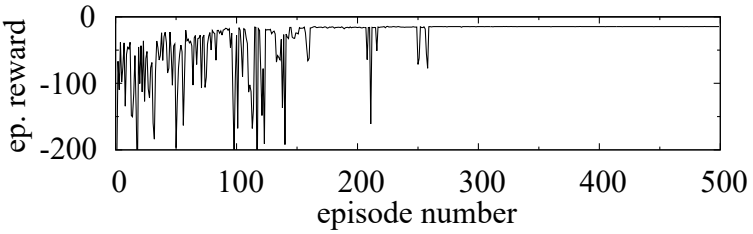

Figure 5.3: SO, Episode Reward (QL without Prior Knowledge)

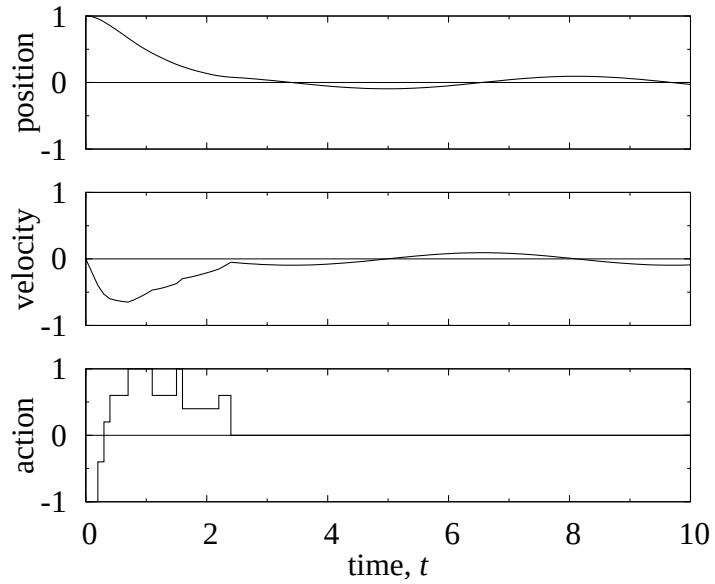

Figure 5.4: SO, Control History before Learning (QL with Correct Prior Knowledge)

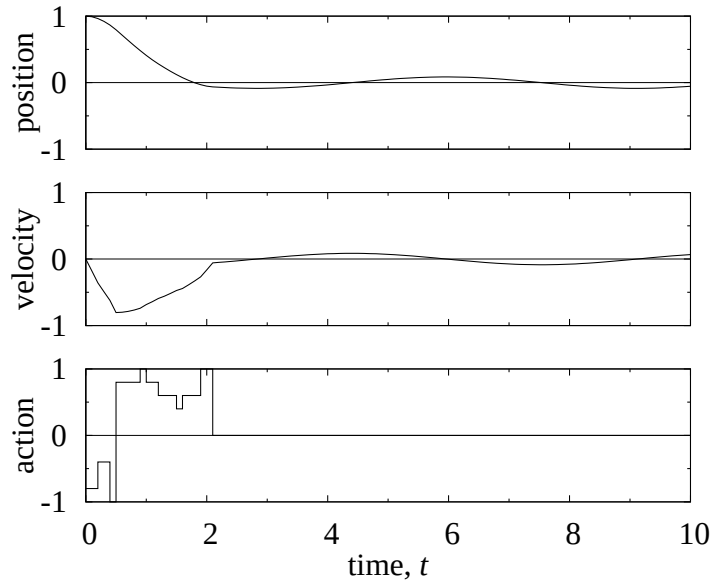

Figure 5.5: SO, Control History after Learning (QL with Correct Prior Knowledge)

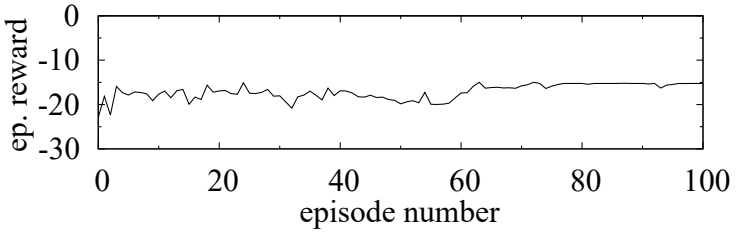

Figure 5.6: SO, Episode Reward (QL with Correct Prior Knowledge)

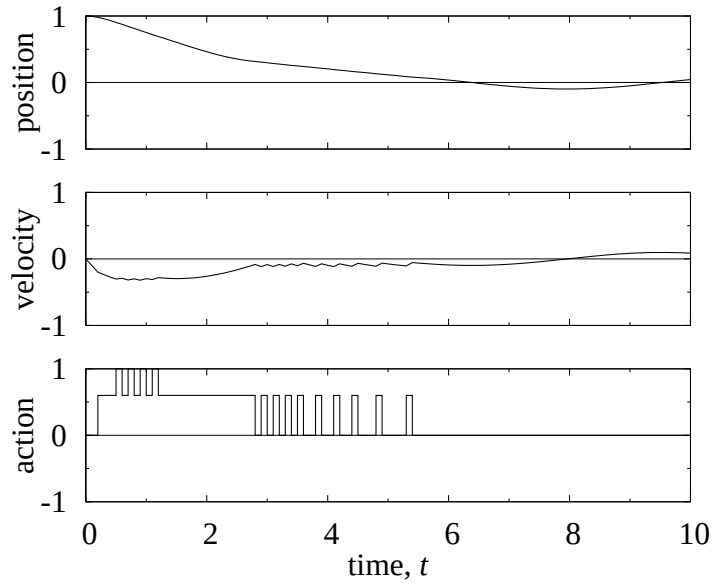

Figure 5.7: SO, Control History before Learning (QL with Incorrect Prior Knowledge)

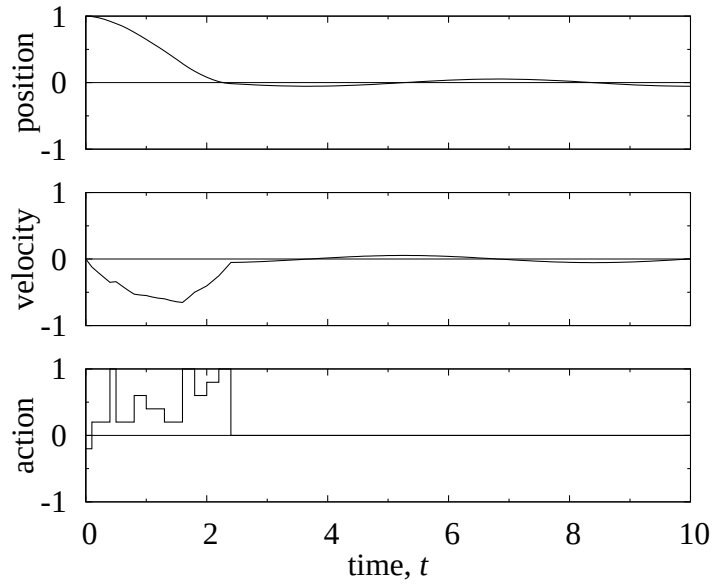

Figure 5.8: SO, Control History after Learning (QL with Incorrect Prior Knowledge)

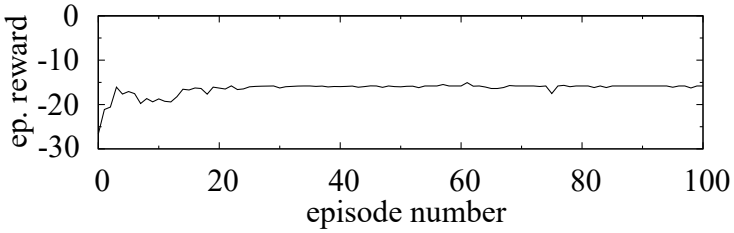

Figure 5.9: SO, Episode Reward (QL with Incorrect Prior Knowledge)

5.2　ドメイン知識の利用

4.2 節の問題設定は，対象の線形近似モデルが既知であるという，比較的緩い条件のもとでの事前知識の組み込みであるといえる．これをより広いドメインの問題に適用する (事前知識のレベルを下げる) とすれば，制御問題の多くにみられる，状態遷移が決定論的かつ報酬が既知であるという前提そのものを一種のドメイン知識であるとみなして，そのドメインに問題領域を限定した (事前知識を利用する) 手法が考えられる．

本節では，model-based 強化学習の代表的な枠組みである Dyna と，これらの前提となる事前知識を組み合わせることで，決定論的環境に特化した提案手法について，初歩的なベンチマークとして 5.1 節と同様の double integrator に加えて，耐故障制御を想定したより応用的な問題として，宇宙機の姿勢制御および航空機の姿勢制御問題を例にその性能を評価する．

5.2.1　Double Integrator の制御

5.2.1.1　運動方程式

本実施例では，実際の環境が次の連続な決定論的状態遷移

$$\ddot{s} = a \tag{5.2.1}$$

および決定論的報酬

$$r = -(10s^2 + \dot{s}^2 + 0.01a^2) \tag{5.2.2}$$

に従うと仮定する．ここでは，連続状態-離散行動の手法である DQN と，連続状態-連続行動に対応した DDPG に提案手法である tabular Dyna を適用する．DQN は離散行動なので，行動を $a \in [-2,\, 0,\, 2]$ の 3 値に限定する．

5.2.1.2　学習条件

学習においては，制御周期を $\Delta t = 0.1$ [s] とし，episode を 200 step (20 sec.) とし，各 episode における状態変数の初期条件は $[s\ \dot{s}] = [\pm 4\ 0]$ とした．既存手法および提案手法それぞれに適用した hyperparameter を Table 5.2 に，network 構造を Table 5.3 に示す．

Table 5.2: Double Integrator, Hyperparameters (ドメイン知識の利用)

	DQN	tabular DDQ	DDPG	tabular Dyna DDPG
learning rate	5e-3	5e-3	5e-3	5e-3
discount factor	0.99	0.99	0.99	0.99
minibatch size	256	32	32	32
plannning steps	-	8	-	16

Table 5.3: Double Integrator, Network Architecture

DQN / tabular DDQ	DDPG / tabular Dyna DDPG	
critic	actor	critic
input layer (state + action)	input layer (state)	input layer (state + action)
quadratic layer	fully connected layer (1)	quadratic layer
fully connected layer (1)		fully connected layer (1)

5.2.1.3　結果

それぞれの手法で double integrator の制御を行った場合の学習履歴を Fig. 5.10 から Fig. 5.13 に示す．また，既存手法と提案手法の計算時間も含めた性能をまとめたものを Table 5.4 および Table 5.5 に示す．

表中の episode to terminate は報酬が規定の値に達するまでに経過した episode 数を指すが，提案手法は既存手法に比べ極めて早い episode 数で学習が収束していることが分かる．なお，本提案手法は planning として 1 ステップあたりに価値関数および/または方策を複数回更新しているため，1 ステップあたりの経過時間は既存手法に比べ大きくなっているが，episode 数と比較しても現実的な時間で計算を行えているといえる．

Table 5.4: Double Integrator, DQN vs. tabular DDQ (ドメイン知識の利用)

	PG	DQN	tabular DDQ
episodes to terminate	799	54	11
total elapsed time	1123	206	52
time per step	0.0070	0.019	0.023

Table 5.5: Double Integrator, DDPG vs. tabular Dyna DDPG (ドメイン知識の利用)

	DDPG	tabular Dyna DDPG
episodes to terminate	442	17
total elapsed time	363	239
time per step	0.0041	0.070

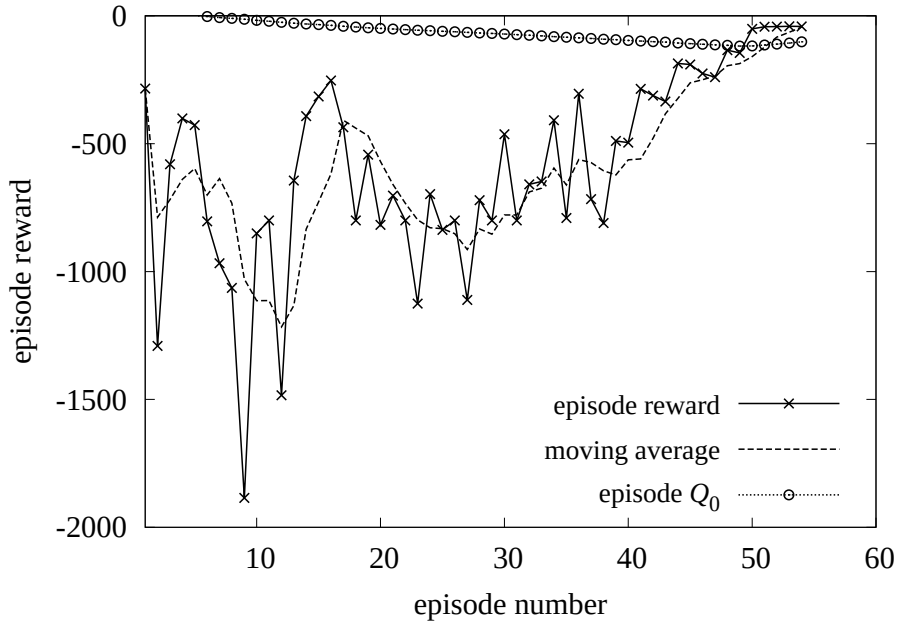

Figure 5.10: DI, Episode Reward (DQN)

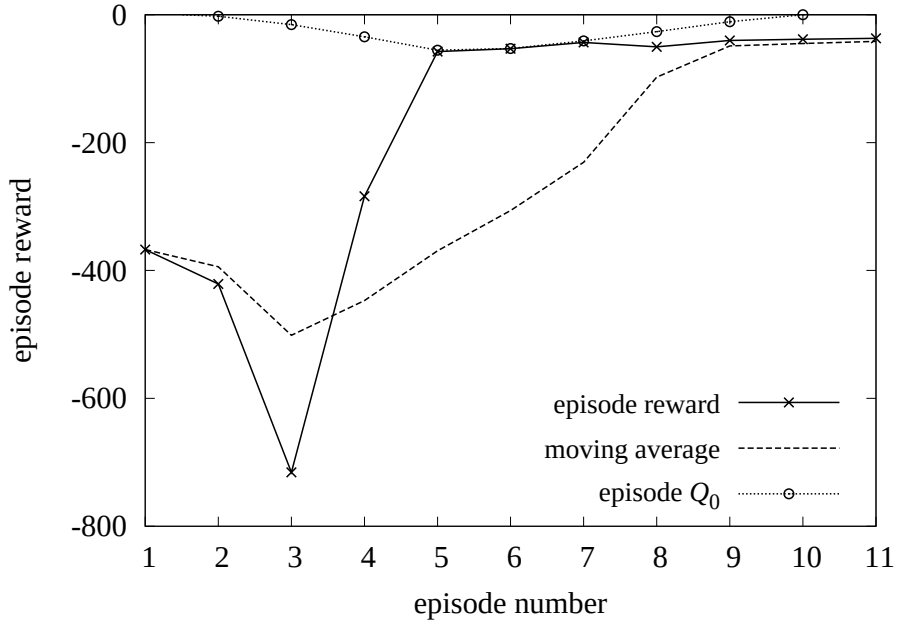

Figure 5.11: DI, Episode Reward (tabular DDQ)

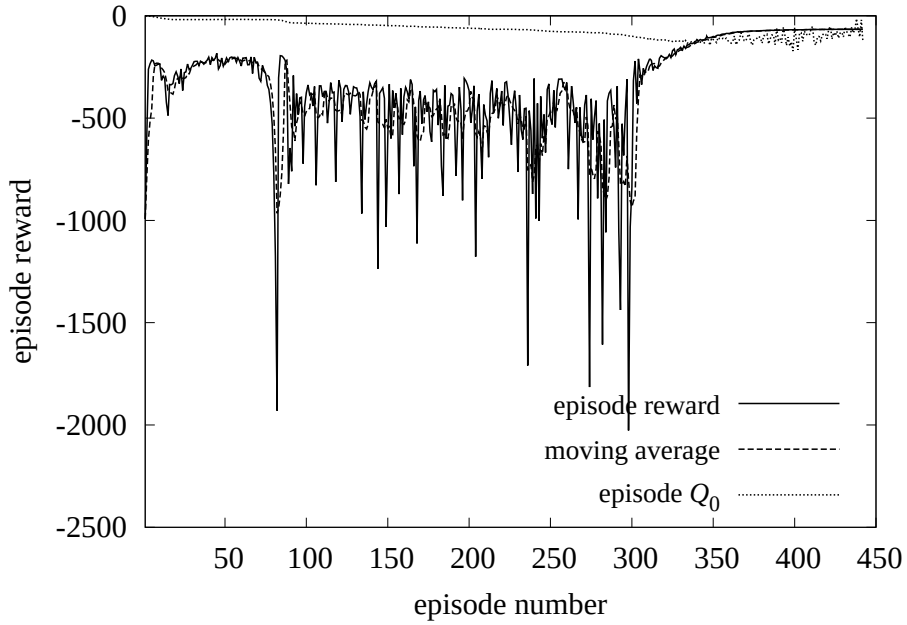

Figure 5.12: DI, Episode Reward (DDPG)

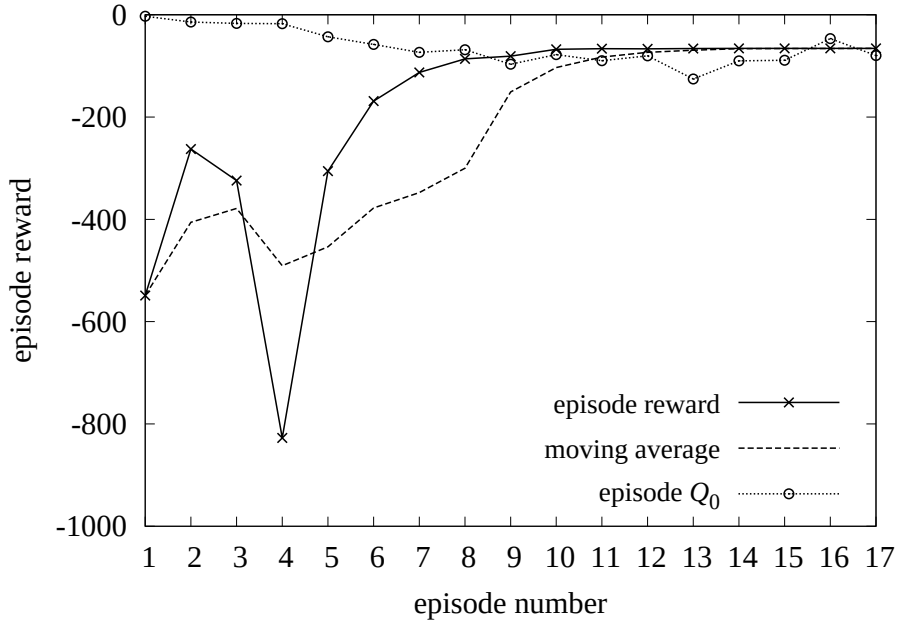

Figure 5.13: DI, Episode Reward (tabular Dyna DDPG)

5.2.2　宇宙機の姿勢制御

宇宙機の姿勢制御問題は，ダイナミクスの性質 (ex. 剛体 or 柔軟構造)，アクチュエータの性質 (ex. 連続 or 離散，全駆動 or 劣駆動) 等の組み合わせにより多岐に渡るが，特に剛体宇宙機の劣駆動姿勢制御は，応用の観点からも制御理論の観点からも非常に重要な課題であり，以前から多くの研究例 [20,28,52] が存在する．姿勢制御則の設計はしばしば MBD (Model-Based Design)，すなわち環境が既知であるものとして，非線形制御やロバスト制御等のアプローチが取られることが多い．しかしながら，運用年数の長さ，要求される信頼性・耐故障性の高さといった宇宙機特有の性質を鑑みれば，事前知識を用いずに自律的に制御則を獲得する適応制御や強化学習等のアプローチにも充分に大きな意義があると考えられる．

本項では，通常の強化学習アルゴリズムによる宇宙機の姿勢制御則の獲得に加え，状態遷移が決定論的であるという事前知識を組み込んだ本論文の提案手法の性能についても検討する．

5.2.2.1　運動方程式と姿勢表現

ここでは姿勢表現法として quaternion を用いる．Euler 軸 $\boldsymbol{\alpha}$ および Euler 回転角 ϕ に対して，回転を表す quaternion $\boldsymbol{q}$ は次のように定義される．

$$\boldsymbol{q} = \begin{bmatrix} \bar{\boldsymbol{q}} \\ q_4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \sin(\phi/2) \\ \cos(\phi/2) \end{bmatrix} \tag{5.2.3}$$

ここで，$\bar{\boldsymbol{q}}$ および q_4 をそれぞれ quaternion のベクトル部およびスカラー部と呼ぶ．

宇宙機を剛体と仮定すると，その回転運動と姿勢は次の Euler の運動方程式および運動学方程式に従う．

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\times}\boldsymbol{J}\boldsymbol{\omega} = \boldsymbol{a} \tag{5.2.4}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} q_4\boldsymbol{I}_3 + \bar{\boldsymbol{q}}^{\times} \\ -\bar{\boldsymbol{q}}^{\mathrm{T}} \end{bmatrix}\boldsymbol{\omega} \tag{5.2.5}$$

ここで，$\boldsymbol{J}$ は剛体の慣性テンソル，$\boldsymbol{\omega}$ は角速度ベクトル，$\boldsymbol{u}$ は制御トルクであり，上付き添字 $^{\times}$ は列ベクトルから同一要素を有する歪対称行列への変換

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{\times} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{5.2.6}$$

を表す．

現在姿勢と目標姿勢を表す quaternion をそれぞれ $\boldsymbol{q}$ および $\boldsymbol{q}_{\mathrm{d}}$ とすれば，quaternion 偏差 $\boldsymbol{q}_{\mathrm{e}}$ は，

$$\boldsymbol{q}_{\mathrm{e}} = \begin{bmatrix} q_{4\mathrm{d}}\boldsymbol{I}_3 - \bar{\boldsymbol{q}}_{\mathrm{d}}^{\times} & -\bar{\boldsymbol{q}}_{\mathrm{d}} \\ \bar{\boldsymbol{q}}_{\mathrm{d}}^{\mathrm{T}} & q_{4\mathrm{d}} \end{bmatrix}\boldsymbol{q} \tag{5.2.7}$$

となる．式 (5.2.7) から分かるように，$\boldsymbol{q}_{\mathrm{d}} = [0,\ 0,\ 0,\ 1]^{\mathrm{T}}$ のとき $\boldsymbol{q}_{\mathrm{e}} = \boldsymbol{q}$ となる．すなわち，$[0,\ 0,\ 0,\ 1]^{\mathrm{T}}$ は quaternion 積の単位元である．以下では，簡単のために特に断りの無い限り，目標姿勢を $\boldsymbol{q}_{\mathrm{d}} = [0,\ 0,\ 0,\ 1]^{\mathrm{T}}$，quaternion 偏差を $\boldsymbol{q}_{\mathrm{e}} = \boldsymbol{q}$ として記述する．

5.2.2.2　姿勢制御則

quaternion に基づく宇宙機の姿勢制御については，以前から数多くの研究がなされている．本研究では，three control torques と two control torques の両方の場合について，以下に示す先行研究を benchmark として用いることとする．

(a) Three Control Torques

3 つの制御トルクを持つ，すなわち $\boldsymbol{a} = [a_1,\, a_2,\, a_3]^{\mathrm{T}}$ のときは，Wie[55,56] による quaternion feedback

$$\boldsymbol{a} = -k_1\boldsymbol{\omega} - k_2\bar{\boldsymbol{q}} \tag{5.2.8}$$

を施すことにより，大域的漸近安定化が可能である．

(b) Two Control Torques

制御トルクが 2 つ，すなわち $\boldsymbol{a} = [a_1,\, a_2,\, 0]^{\mathrm{T}}$ の場合にはシステムは非ホロノミックとなり，静的連続状態フィードバックでは漸近安定化が不可能である．制御則の一例としては，quaternion feedback に加え，非制御軸を安定化するための非線形 feedback を有する singular controller approach[28,52] がよく知られている．

$$a_i = -K_1(\omega_i - \omega_{i\mathrm{d}}) + K_2\dot{\omega}_{i\mathrm{d}} \tag{5.2.9}$$

$$\omega_{1\mathrm{d}} = -k_1 q_1 + k_2\frac{q_2 q_3}{q_1^2 + q_2^2} \tag{5.2.10}$$

$$\omega_{2\mathrm{d}} = -k_1 q_2 - k_2\frac{q_1 q_3}{q_1^2 + q_2^2} \tag{5.2.11}$$

式 (8) および式 (9) から分かるように，非線形 feedback 項は平衡点 $\boldsymbol{q} = [0,\, 0,\, 0,\, 1]^{\mathrm{T}}$ に特異点を有するため，外乱等によって不安定化する可能性がある．そこで，より一般的かつ実用的な拡張として，非線形 feedback に飽和要素を付加した制御則が Horri and Palmer[20] によって提案されている．

$$\omega_{1\mathrm{d}} = -k_1 q_1 + k_2\,\mathrm{sat}\left(\frac{q_2 q_3}{q_1^2 + q_2^2},\, a_1\right) \tag{5.2.12}$$

$$\omega_{2\mathrm{d}} = -k_1 q_2 - k_2\,\mathrm{sat}\left(\frac{q_1 q_3}{q_1^2 + q_2^2},\, a_2\right) \tag{5.2.13}$$

Zou, et al.[60] は前述の制御則をさらに発展させ，$|\bar{\boldsymbol{q}}| = 0$ での切替機構を備えた制御則を提案している．

$$\omega_{1\mathrm{d}} = \begin{cases} -k_1 q_1 + k_2\dfrac{q_2 q_3}{|\bar{\boldsymbol{q}}|^2} & (|\bar{\boldsymbol{q}}| \neq 0) \\ 0 & (|\bar{\boldsymbol{q}}| = 0) \end{cases} \tag{5.2.14}$$

$$\omega_{2\mathrm{d}} = \begin{cases} -k_1 q_2 - k_2\dfrac{q_1 q_3}{|\bar{\boldsymbol{q}}|^2} & (|\bar{\boldsymbol{q}}| \neq 0) \\ 0 & (|\bar{\boldsymbol{q}}| = 0) \end{cases} \tag{5.2.15}$$

5.2.2.3　学習条件

制御対象である宇宙機の慣性テンソルを $\boldsymbol{J} = \mathrm{diag}(0.5,\, 0.7,\, 1)$，制御トルクの大きさを $u_i = [-1,\, 1]$ とし，計算はステップ幅 $\Delta t = 0.1$ の Runge-Kutta-Fehlberg 法で行った．観測は角速度 $\boldsymbol{\omega}$ および quaternion のベクトル部 $\bar{\boldsymbol{q}}$ とし，次の報酬

$$r = -(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} + \bar{\boldsymbol{q}}^{\mathrm{T}}\bar{\boldsymbol{q}} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}) \tag{5.2.16}$$

の総和を最大化することを目標とした．また，episode 開始時に $\boldsymbol{\omega}_0$ および $\boldsymbol{q}_0$ を乱数で初期化することで，任意の初期値に対する制御則を学習した．three control torques の場合には，線形状態 feedback (quaternion feedback) で安定化可能であることが既知である．そこで，Table 5.6 に示す hyper parameter と Table 5.7 に示す network 構造で学習を行った．また，two control torques の場合には，制御則に微分要素を含むことが推定されるため，network の入力を現在の観測および 1 step 前の観測とし，actor および ciritc の全結合層の要素数を 128 に増加させた．

5.2.2.4　結果

(a) Three Control Torques

初期条件 $\boldsymbol{\omega}_0 = [0.205,\, 0.404,\, 0.487]^{\mathrm{T}}$, $\boldsymbol{q}_0 = [0.0338,\, -0.391,\, 0.283,\, -0.875]^{\mathrm{T}}$ のとき DDPG および tabular Dyna DDPG で獲得した制御則と，Levenberg-Marquardt 法でパラメータを最適化した Wie による制御則 ($k_1 = 1.73$, $k_2 = 1.43$) の計算例をそれぞれ Fig. 5.14 および Fig. 5.15 に示す．両者を比較するとほぼ同様な制御履歴を示しており，最適に近い制御則の獲得に成功している．

DDPG と tabular Dyna DDPG の比較のために，学習に要する episode 数，学習に要する時間，1step あたりの所要時間を計測した結果を Table 5.8 に示し，それぞれの学習履歴を Fig. 5.16 から Fig. 5.23 に示す．ここでは tabular Dyna DDPG の planning step 数は固定とし，minibatch サイズ，すなわち 1 回の学習に用いる experience の数を変更した．Table 5.8 によれば，総じて DDPG よりも tabular Dyna DDPG の方が早く収束している，すなわちサンプル効率が向上しているといえる．DDPG と tabular Dyna DDPG に共通して minibatch サイズが小さくなるほど 1step あたりの所要時間が短くなるが，小さくし過ぎるとかえって学習が不安定になる傾向がある．minibatch サイズについての特筆すべき点として，DDPG では minibatch サイズ 16 で学習に失敗したのに対して，tabular Dyna DDPG では 8 および 16 でも成功しており，1step に複数回学習を行うことにより，少ない minibatch サイズでも比較的安定して学習が行えることが確認できる．学習の効率の面では，minibatch サイズが 8, 16, 32 と大きくする毎に少ない episode 数で収束に至っているが，64 ではかえって効率が悪化しており，学習の安定化のためには minibatch サイズを最適化する必要があることが分かる．

また，ここでは minibatch サイズ 32 を用いて，非 episode 型で tabular Dyna DDPG による姿勢制御則の獲得を試みた．Fig. 5.22 に示す状態変数および行動，報酬の履歴より，学習を開始してから 50 秒 (500 step) ほどで角速度の安定化に成功し，約 300 秒ほどで目標姿勢に到達している．

(b) Two Control Torques

three control torques の場合と同様の初期条件のとき DDPG で獲得した制御則と，Zou, et al. による制御則の計算例を Fig. 5.24 および Fig. 5.25 に示す．DDPG による学習の結果，ω_1 および ω_2 のみを制御し，非制御軸である ω_3 の制御を積極的に行わない局所解に収束してしまい，3 軸姿勢制御を実現する制御則の獲得には至らなかった．また tabular Dyna DDPG でも同様であった．

Table 5.6: Spacecraft, Hyperparameters

control freq.	0.1	max episode steps	200
actor learn rate	1e-2	critic learn rate	1e-2
actor grad. thre.	1	critic grad. thre.	1
smooth factor	1e-3	discout factor	1
mini-batch size	16/32/64/128	exp. buffer length	1e6
noise variance	0	variance decay rate	-

Table 5.7: Spacecraft, Network Architecture

DDPG / tabular Dyna DDPG	
actor	critic
input layer (state)	input layer (state + action)
fully connected layer (1)	quadratic layer
	fully connected layer (1)

Table 5.8: Spacecraft, DDPG vs. tabular Dyna DDPG

	DDPG				tabular Dyna DDPG			
mini-batch size	16	32	64	128	8	16	32	64
planning step	-	-	-	-	8	8	8	8
episodes to terminate	∞	372	374	346	257	206	217	395
total elapsed time	∞	1154	1232	1362	5227	4367	4722	8934
time per step	∞	0.015	0.016	0.019	0.102	0.106	0.109	0.113

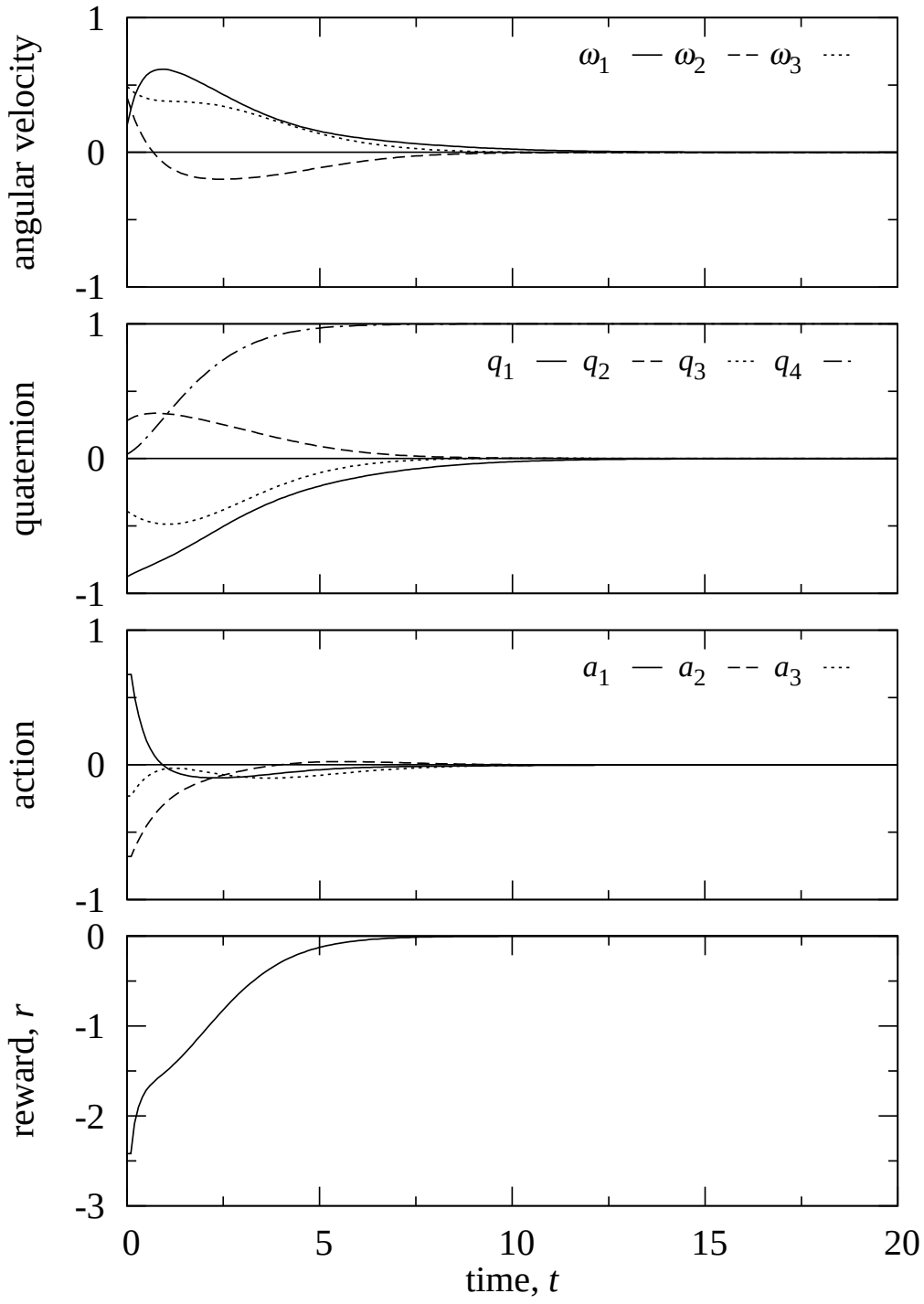

Figure 5.14: Spacecraft, Control Histroy (DDPG)

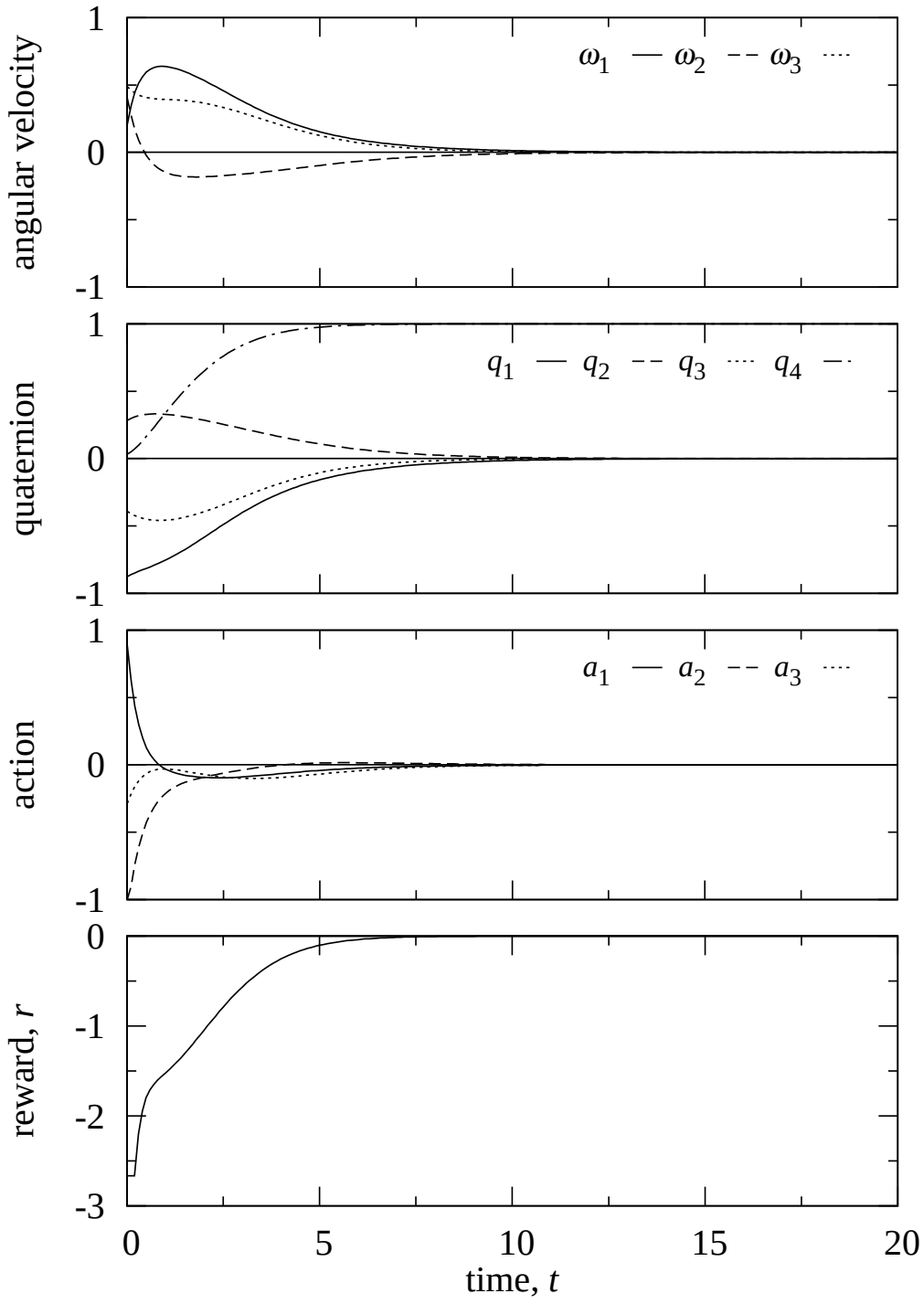

Figure 5.15: Spacecraft, Control Histroy (Wie)

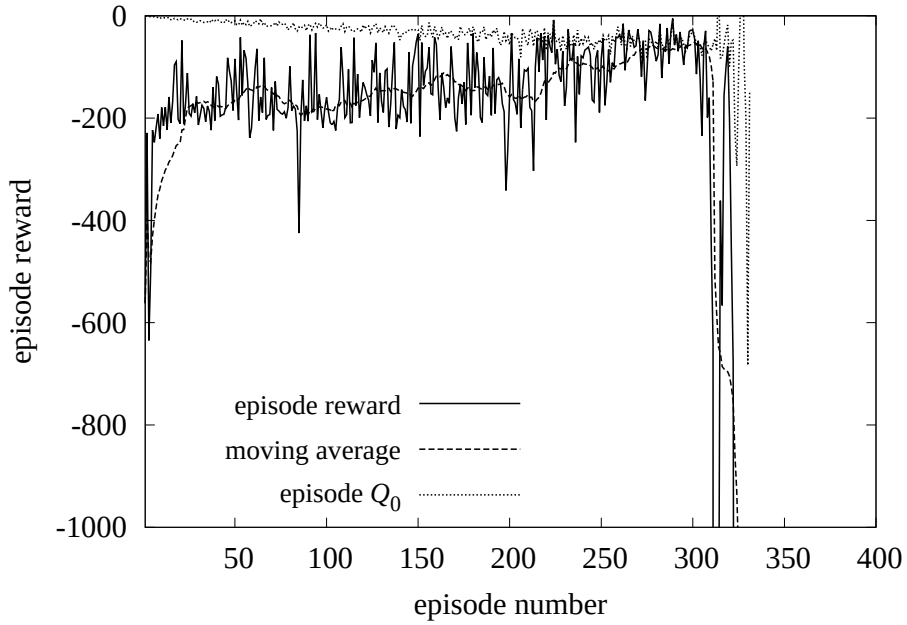

Figure 5.16: Spacecraft, Episode Reward (DDPG, Mini-Batch Size = 16)

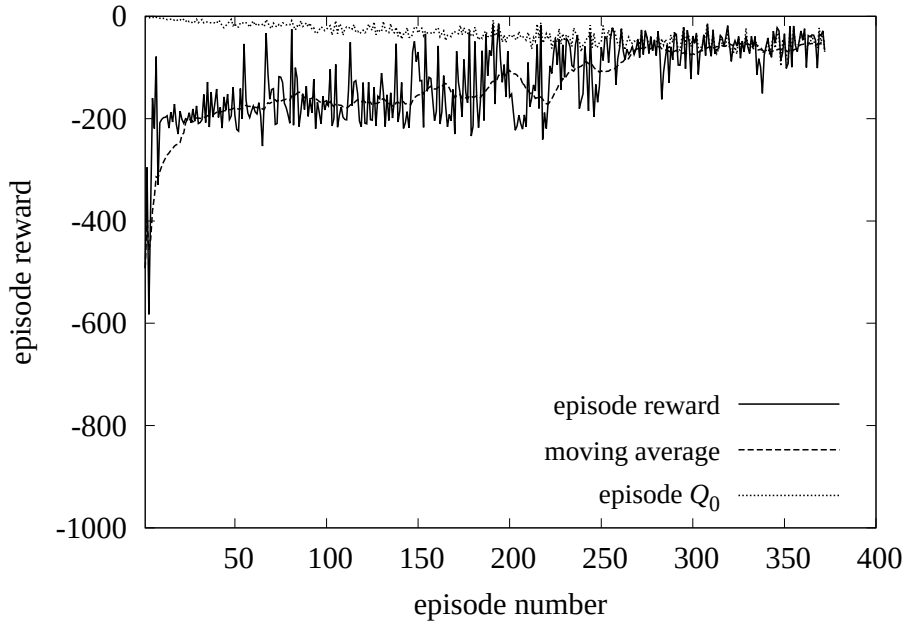

Figure 5.17: Spacecraft, Episode Reward (DDPG, Mini-Batch Size = 32)

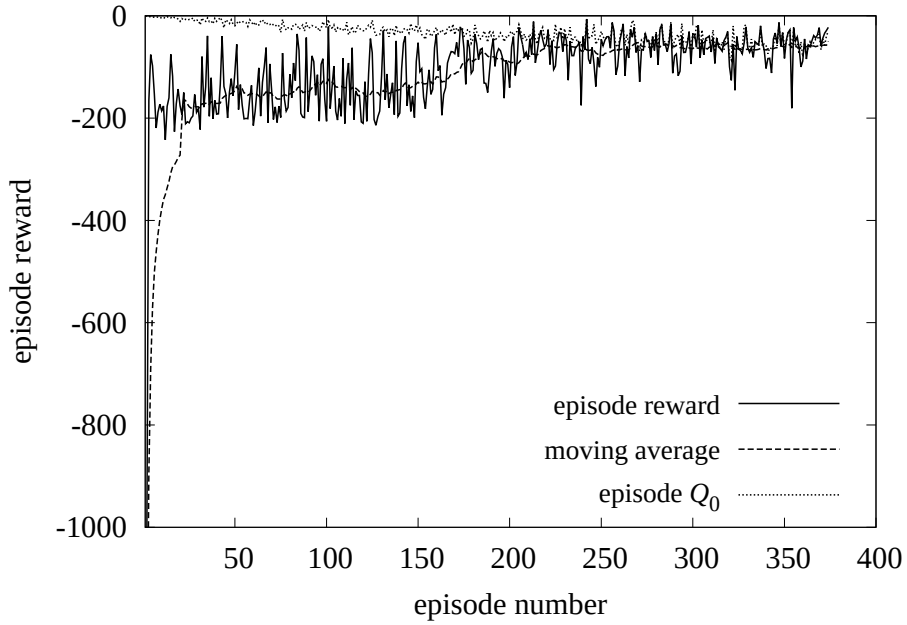

Figure 5.18: Spacecraft, Episode Reward (DDPG, Mini-Batch Size = 64)

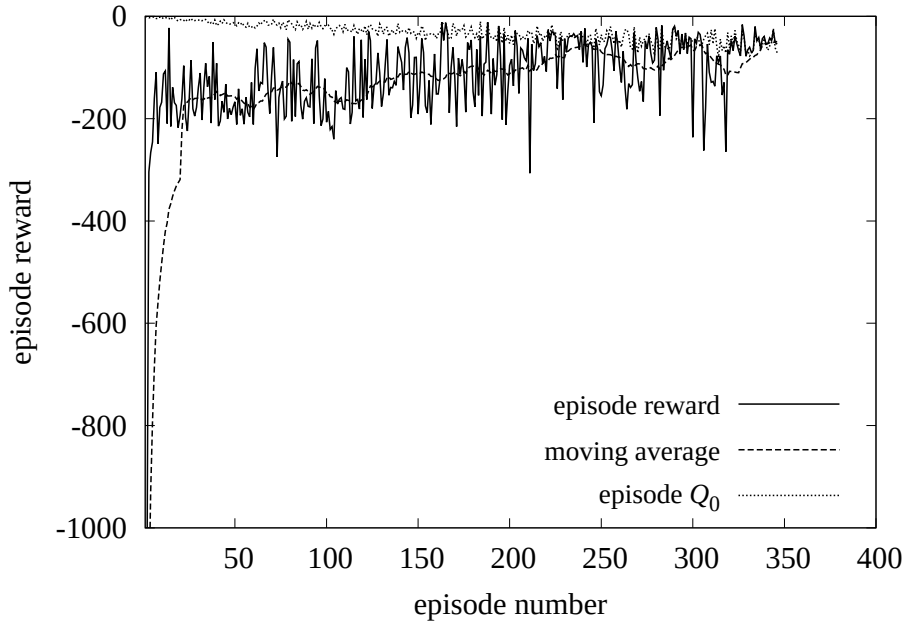

Figure 5.19: Spacecraft, Episode Reward (DDPG, Mini-Batch Size = 128)

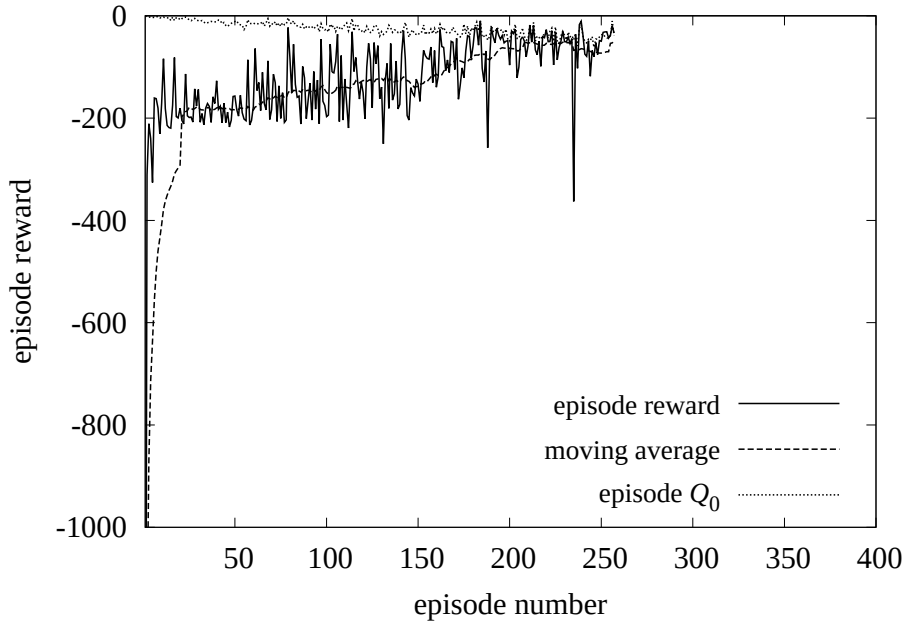

Figure 5.20: Spacecraft, Episode Reward (tabular Dyna DDPG, Mini-Batch Size = 8)

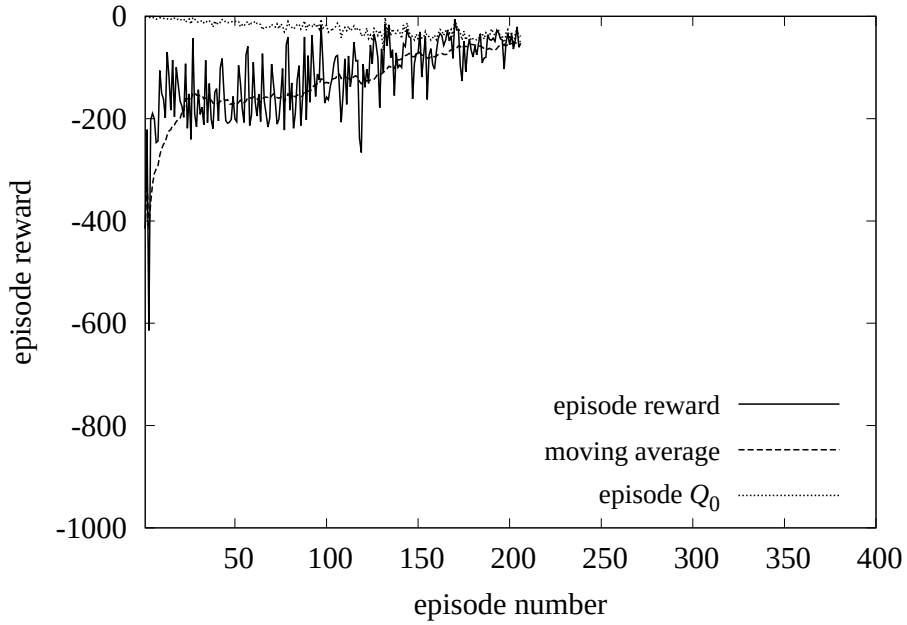

Figure 5.21: Spacecraft, Episode Reward (tabular Dyna DDPG, Mini-Batch Size = 16)

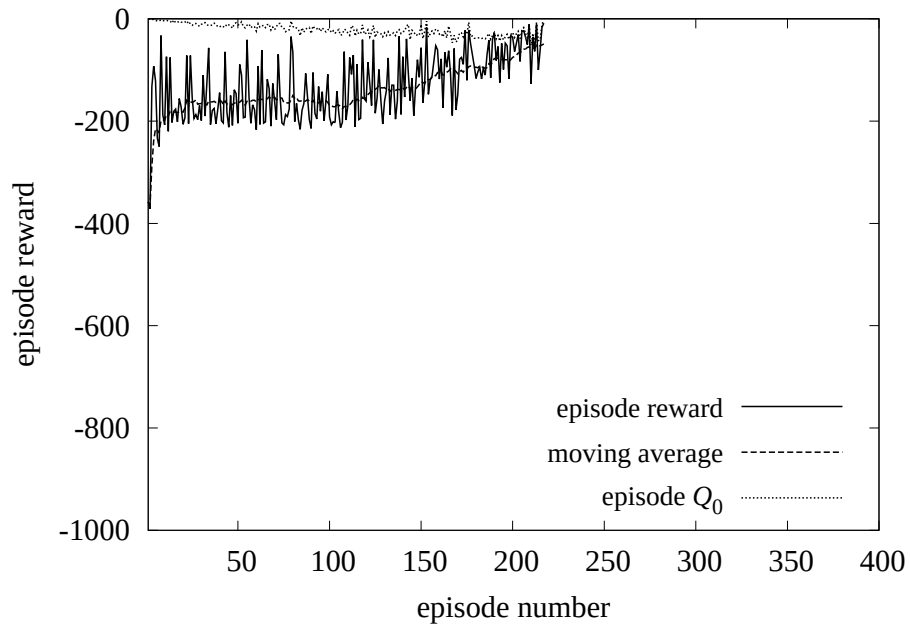

Figure 5.22: Spacecraft, Episode Reward (tabular Dyna DDPG, Mini-Batch Size = 32)

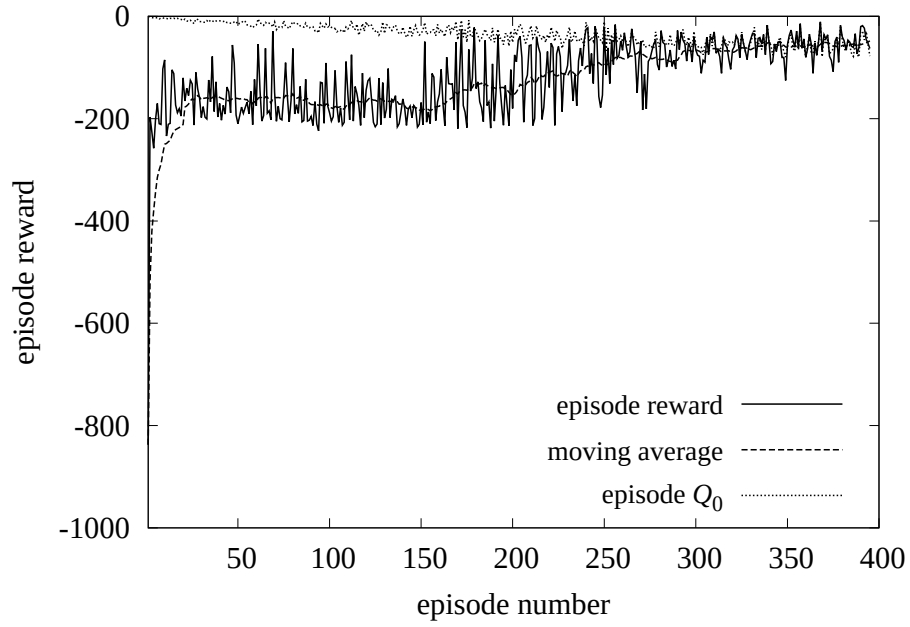

Figure 5.23: Spacecraft, Episode Reward (tabular Dyna DDPG, Mini-Batch Size = 64)

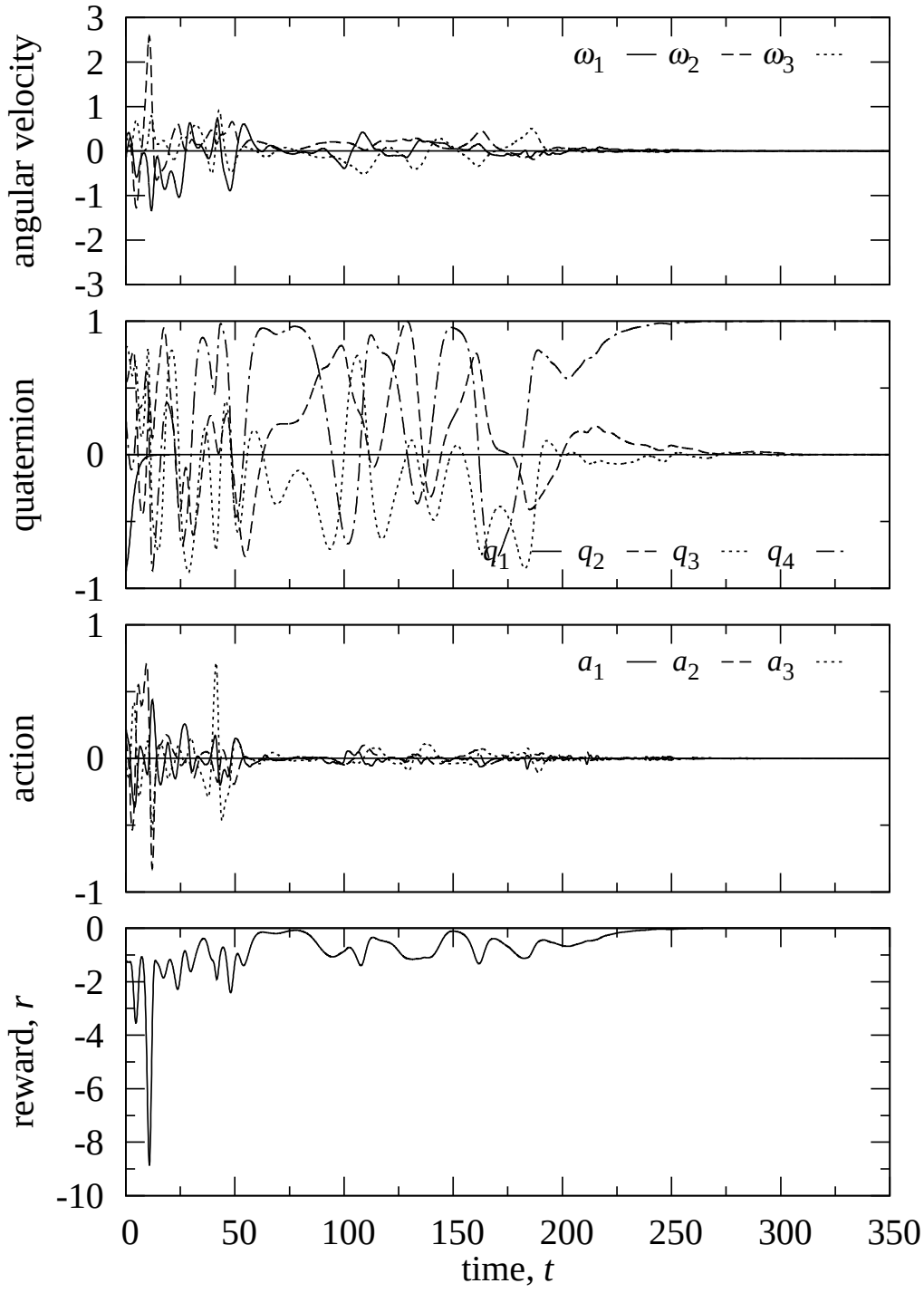

Figure 5.24: Spacefraft, Control Histroy (tabular Dyna DDPG)

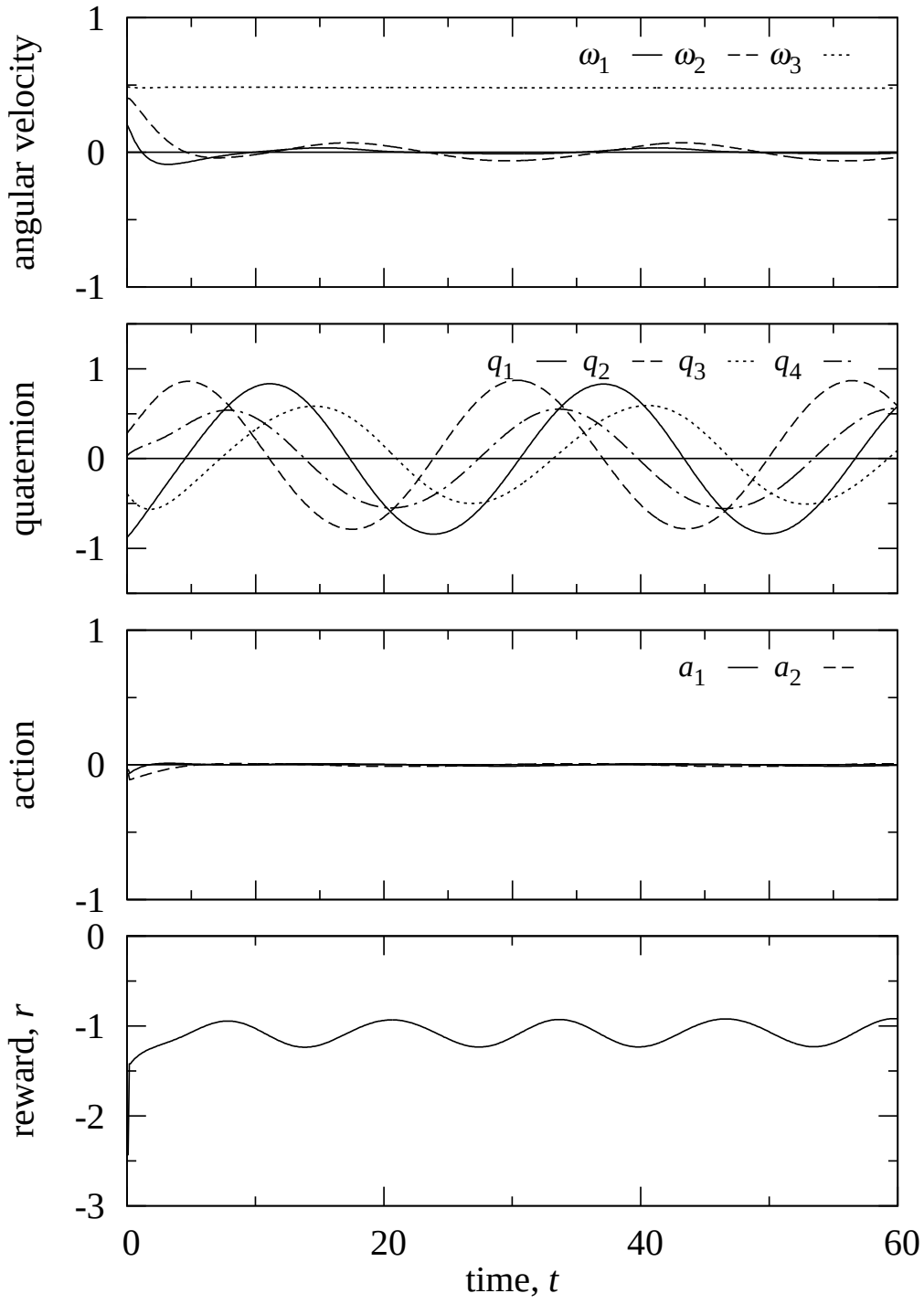

Figure 5.25: Two Control Torques Spacecraft, Control Histroy (DDPG)

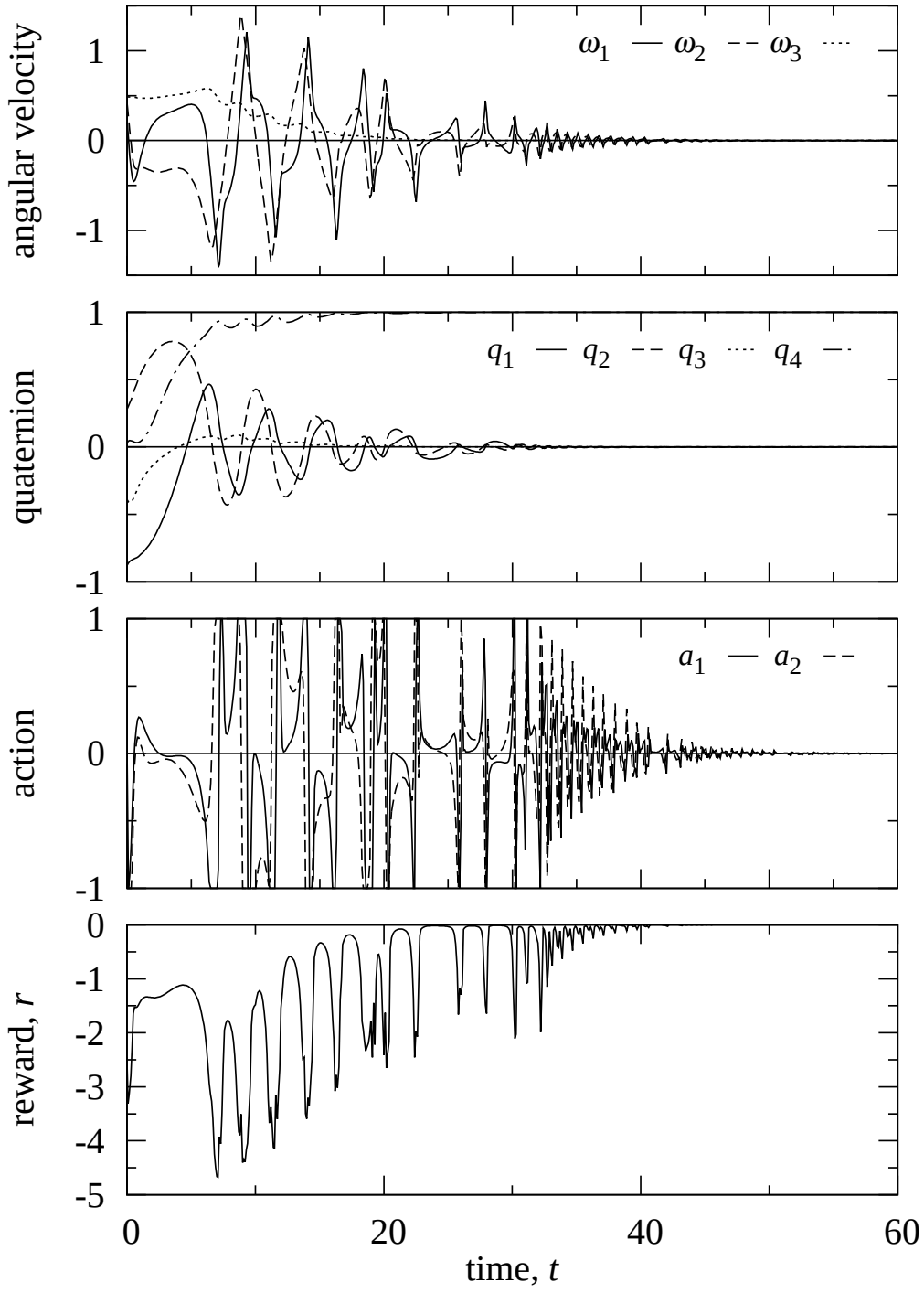

Figure 5.26: Two Control Torques, Spacecraft, Control Histroy (Zou, et al.)

5.2.3　航空機の姿勢制御

続いての応用例として，DDPG および tabular Dyna DDPG を用いて，故障を想定した航空機の縦系の姿勢制御の獲得を試みた．

5.2.3.1　運動方程式

航空機の縦系の微小擾乱運動方程式は次式で与えられる．

$$\begin{aligned}\left[\frac{\mathrm{d}}{\mathrm{d}t}-X_u\right]u-X_w w+g_0\cos\theta_0\cdot\theta &= X_{\delta_\mathrm{e}}\delta_\mathrm{e}+X_{\delta_\mathrm{T}}\delta_\mathrm{T}\\ -Z_u u+\left[(1-Z_{\dot{w}})\frac{\mathrm{d}}{\mathrm{d}t}-Z_w\right]w-[u_0+Z_q]q+g_0\sin\theta_0\cdot\theta &= Z_{\delta_\mathrm{e}}\delta_\mathrm{e}+Z_{\delta_\mathrm{T}}\delta_\mathrm{T}\\ -M_u u+\left[M_{\dot{w}}\frac{\mathrm{d}}{\mathrm{d}t}+M_w\right]w+\left[\frac{\mathrm{d}}{\mathrm{d}t}-M_q\right]q &= M_{\delta_\mathrm{e}}\delta_\mathrm{e}+M_{\delta_\mathrm{T}}\delta_\mathrm{T}\end{aligned}\tag{5.2.17}$$

ここで，X_i，Z_i，M_i は有次元安定微係数であり，δ_e はエレベータの舵角，δ_T は推力に相当する係数である．

航空機の縦系の運動は，それぞれ周期が短く減衰が大きい短周期モードと，周期が長く減衰が小さい phugoid モードと呼ばれる 2 つのモードからなる．いずれも安定なモードであり制御は比較的容易であるが，ここでは舵面の故障を想定し，推力のみによる制御を試みる．

5.2.3.2　学習条件

安定微係数は B747 のもの [17] を利用し，それぞれ $X_u=-0.0212, X_w=0.0466, Z_u=-0.2306, Z_w=-0.6038, Z_{\dot{w}}=-0.0341, Z_q=-7.674, M_u=0, M_w=-0.0019, M_{\dot{w}}=-0.0002, M_q=-0.4381, g_0=9.8, \theta_0=0, U_0=306.25085$ と設定する．また，前述のように舵面の故障を想定し，制御入力に係る微係数は $X_{\delta_\mathrm{e}}=Z_{\delta_\mathrm{e}}=M_{\delta_\mathrm{e}}=Z_{\delta_\mathrm{T}}=M_{\delta_\mathrm{T}}=0$，$X_{\delta_\mathrm{T}}=10$ として，推力のみによる制御を行う．なお，制御周期はこれまでの例と同様に，0.1 sec. とする．

観測は速度 $[u,\, w]$ および姿勢角 θ とその変化率 q とし，次の報酬 (評価関数)

$$r=-(u^2+w^2+q^2+\theta^2+\delta_\mathrm{T}^2)\tag{5.2.18}$$

を最大化することを目標とした．入力である推力に関する係数 δ_T の大きさは [-1, 1] に制限し，実際の推力の値は入力に対して 1 次遅れで追従するとした．敢えて非常に悪い条件を想定し 10 sec. と，通常の値として 1 sec. の場合について検証した．

学習における hyperparameter は Table 5.9 の通りである．また，今回はより実際の耐故障制御に近い状況を想定し，episode 型の学習ではなく online での学習，すなわち仮に状態が発散してもやり直しができない状況とした．

5.2.3.3　結果

まず，全く制御を行わないときの航空機の縦の運動を Fig. 5.27 に示す．X 軸方向の速度 v に何らかの外乱が加わると，航空機は phugoid 運動を始め，概ね 300 秒程度で減衰することが確認できる．次に，DDPG で制御を試みた場合を Fig. 5.28 に示す．制御を行わない場合に比べて短い 200 秒程度で phugoid 運動が収束しているが，探索の過程において，速度が 100 程度，ピッチ角は約 40 度まで達しており，実質的には制御が破綻しているといえる．次に，planning step が 8 の tabular Dyna DDPG による制御の結果を Fig. 5.29 に示す．収束に要する時間は，制御なしおよび通常の DDPG に比べると大きく減少しており，速度の変動も初期の擾乱 (10) 以下に抑えられていることから，学習は極めて高速かつ安定的に行われていることが分かる．時定数 1 sec. の場合についても，探索の過程において僅かに速度の変動がみられるものの，高速かつ安定的な学習が行われている．ただし，時定数 10 sec. の場合に比べてピッチ角の偏差は微増しており，学習が高速に行われる分，精緻な制御則を学習する前に探索を終えてしまうという探索と知識利用の trade-off がみられた．

Table 5.9: Aircraft, Hyperparameters

actor learn rate	1e-2	critic learn rate	1e-2
actor grad. thre.	1	critic grad. thre.	1
smooth factor	1e-4	discout factor	1
mini-batch size	32	exp. buffer length	1e6
noise variance	0.1	variance decay rate	1e-3

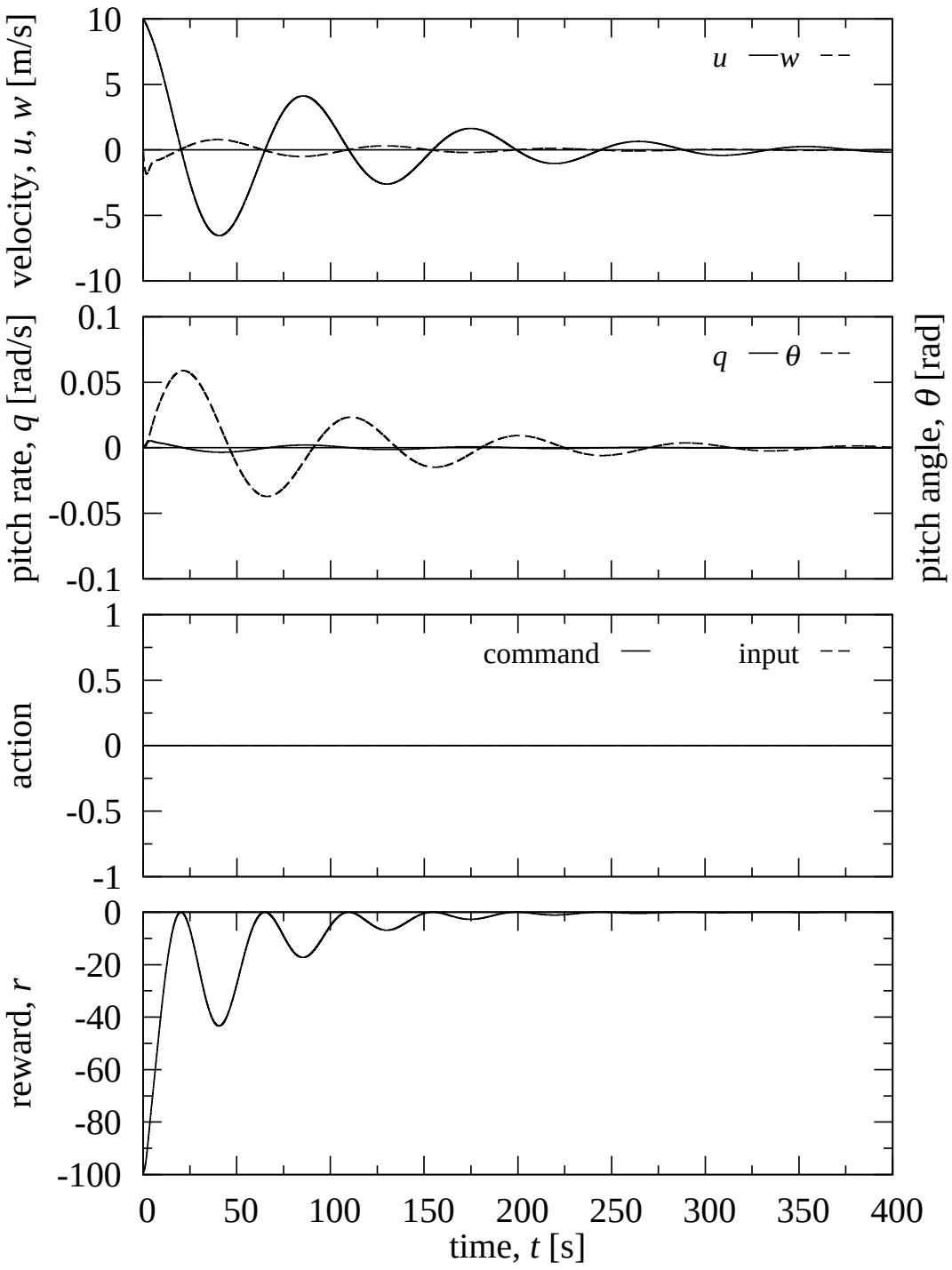

Figure 5.27: Aircraft, Control History (without Control)

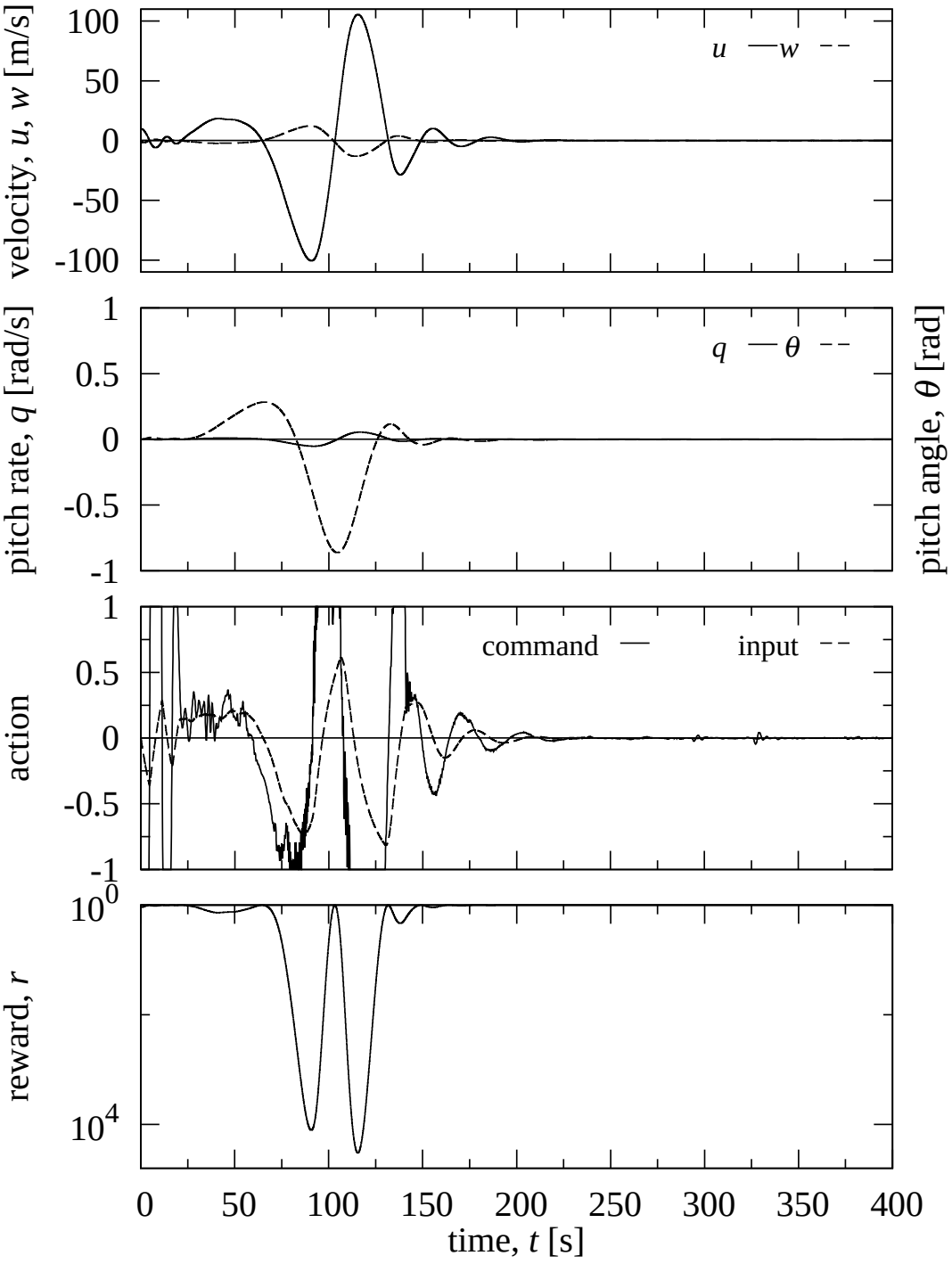

Figure 5.28: Aircraft, Control History (DDPG, Time Constant = 10)

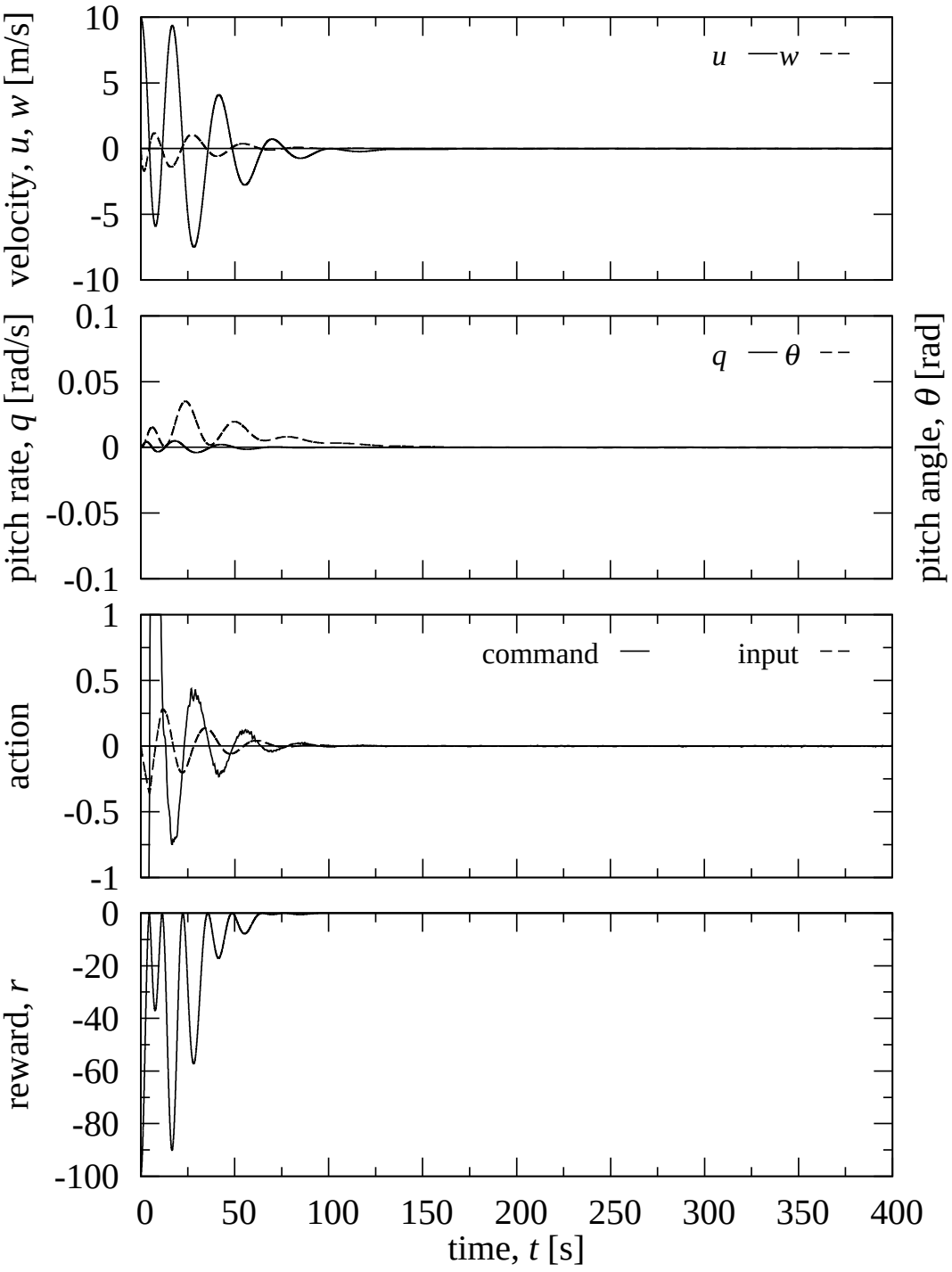

Figure 5.29: Aircraft, Control History (tabular Dyna DDPG, Time Constant = 10)

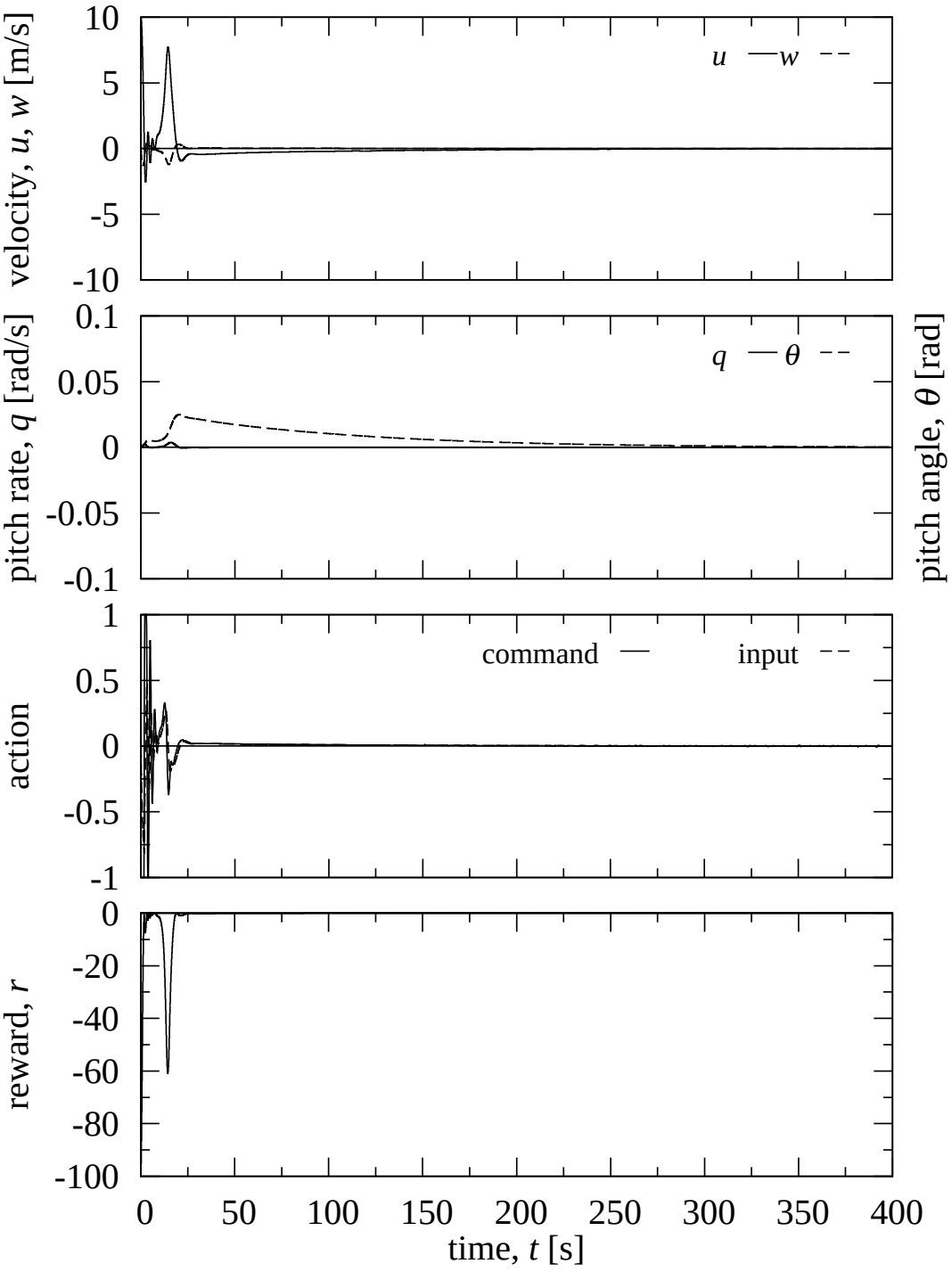

Figure 5.30: Aircraft, Control History (tabular Dyna DDPG, Time Constant = 1)

5.3　考察

5.3.1　環境に関する事前知識の組み込み

本論文における数値例は，Double Integrator の制御問題に対して，事前知識として 2 次形式評価関数と平衡点での線形近似モデルが既知である状況を想定した．結果により，一定のモデル化誤差が存在しても，最低限の制御性能を保ちつつ，適応的に最適解を探索する振る舞いが見られた．

この問題設定は比較的初等的なものであるが，今後の研究に十分に示唆を与えるものである．第 1 に，安定な平衡点ではなく不安定な平衡点へ収束させるような制御に対して，提案手法が有効であるかどうかは自明ではない．仮に事前知識に誤りがあったとしても，状態が安定な平衡点の近傍に留まってさえいれば，平衡点の近傍の価値の学習は効率的に進むはずであるが，不安定な平衡点への収束を目的とするときは，事前知識がかえって探索を阻害するような可能性も考えられる．

第 2 に，平衡点以外の点での線形近似が事前知識として与えられていた場合は，モデル化誤差に対する感度は大幅に変化することが予想される．今回の例のように，平衡点での線形近似が与えられていた場合，仮にモデル化誤差が存在したとしても，状態が平衡点に近づくほど状態価値の誤差の度合いは小さくなるはずであり，平衡点に関してはほぼ正しい知識が与えられていると言えるからである．

5.3.2　ドメイン知識の利用

前述の環境に関する事前知識の組み込みに共通するが，この場合も安定な平衡点への収束を目的とする Double Integrator，航空機の計算例に対しては高い効果がみられ，中立安定な平衡点への収束を目的とする宇宙機の計算例に対しても，提案手法は一定の効果が見られた．特に，宇宙機の 3 軸姿勢制御については，minibatch サイズを変更して学習効率を比較した結果，提案手法である tabular Dyna DDPG は，既存手法に比べて少ない minibatch サイズでも安定して学習を行うことができ，学習効率が最大となる minibatch サイズも，既存手法よりも小さいことが明らかになった．一般に minibatch サイズが小さくなるほど 1step あたりの所要時間が短くなるが，小さくし過ぎるとかえって学習が不安定になる傾向があるが，tabular Dyna により小さい minibatch サイズで 1 step に複数回価値関数および方策の更新を行うことによって，安定かつ高サンプル効率で学習が進行することが示された．これにより，1 step に要する計算時間は当然増大するが，宇宙機および航空機の両方のベンチマークにおいて，現実的な計算時間で online での学習に成功しており，実問題への応用可能性を示す結果となった．

Chapter 6　結論

6.1　結論と本論文の貢献

本研究の第 1 の目的である強化学習における事前知識とアルゴリズムの性能の関係については，探索における No Free Lunch 定理の一つの拡張として，強化学習における No Free Lunch 定理を提案し，強化学習の性能を向上するには何らかの事前知識を組み込むことにより，問題領域を限定する他ないと結論づけた．これは第 2 の目的である強化学習への事前知識を組み込みを支持する結果である．

次に，強化学習に事前知識を組み込む (= 問題領域を限定する) ことにより，システム制御に特化した強化学習アルゴリズムを構築することを目的として，事前知識の多寡に応じて 1. 環境に関する知識の組み込み と 2. ドメイン知識の利用 について，新たな手法を提案し，その性能について評価した．その結果，1. 環境に関する知識の組み込み として提案した Q-learning における $Q(s, a)$ の初期化では，初等的な例ながらも安全な強化学習の 1 つの実装例として効果を実証することができた．2. ドメイン知識の利用 については，環境が決定論的であるという問題領域のもとで，experience replay と Dyna の中間的な手法と深層強化学習アルゴリズムである DQN と DDPG を組み合わせることにより，現実的な計算時間で学習を大幅に短縮することができた．

さらに，提案手法の応用例として，強化学習による宇宙機の 3 軸姿勢制御則の獲得を試みた結果，既存手法よりも安定かつ高サンプル効率で学習を行えることが示され，さらには非 episode 型，すなわち online での 3 軸姿勢制御則の獲得にも成功した．

また，航空機の耐故障制御を想定した例では，推力のみによる phugoid モードの制御において，本論文で提案する手法が，既存の手法に比べて静定時間およびピッチ角変動が大幅に減少するという高い性能を示すことが確認された．

現在提案されている強化学習の多くが確率論的環境を想定し，1 ステップに 1 回しか価値関数および/または方策の更新を行わないことを鑑みれば，決定論的環境を前提とすることで，従来の model-based 強化学習よりも高速に，かつ制御周期の中で時間が許す限り planning を行う本論文の提案手法は，宇宙機や航空機の耐故障制御のみならず，強化学習一般の実問題への適用に大きく貢献するものである．

6.2　今後の課題と展望

本論文ではまず，強化学習における事前知識とアルゴリズムの性能について，強化学習における探索は，全ての状態行動対を無限回訪問すること前提とするが故に，静的な数理計画問題に帰着するため，強化学習に対しても No Free Lunch 定理は成立することを述べた．探索における No Free Lunch 定理に関しても盛んに議論されていることであるが，No Free Lunch 定理の帰結は評価関数 (最適制御問題においては状態遷移関数) の分布が一様であるという前提に基づくものであるにも関わらず，実世界の問題は明らかにそれを満たさない．したがって，3.2.4 項でも述べたように，可能な全ての問題の部分集合に対する No Free Lunch 定理を見出すことが最も重要な課題である．

さらに，環境に関する知識の組み込みとして，Q-learning における $Q(s, a)$ の初期化について検討したが，依然として初等的な例題に対する実証に留まっている．考察でも述べたが，安定な平衡点ではなく不安定な平衡点へ収束させるような制御および平衡点以外の点での線形近似が事前知識として与えられていた場合の有効性については今後の課題である．

ドメイン知識の利用については，環境が決定論的であるという問題領域のもとで，experience replay と Dyna の中間的な手法と深層強化学習アルゴリズムである DQN と DDPG を組み合わせることにより，計算時間が長くなる代わりに既存手法よりも安定かつ高サンプル効率で学習を行えることを示した．これにより，既存手法では学習が不可能あるいは非常に不安定であった 3 トルクによる宇宙機の姿勢制御および推力のみによる航空機の縦系の姿勢制御の online での学習を成功させている．一方で，劣駆動 (2 トルク) 宇宙機の場合には既存手法と提案手法の双方で学習に失敗しており，提案手法の耐故障制御としての適用可能性は十分ではない．劣駆動宇宙機の姿勢制御則の獲得に失敗した原因として考えられるのは，順方向 NN の表現力の限界である．model-based 設計による制御則の例 (式 (5.2.9)〜式 (5.2.15)) をみると，いずれも 2 重の loop を持つカスケード系かつ分母に quaternion を含む特異な制御則である．NN の層を増やし表現力を増大させたとしても，このような複雑な写像をそもそも NN で表現できるかどうかは自明ではなく，更なる検討が必要である．また，model-based 設計による制御則の制御履歴 (Fig. 5.26) によれば，非制御軸の角速度の減衰は，corning 運動の 1 周期毎に段階的に行われる．ベンチマークの制御則がカスケード系であったことからも分かるように，劣駆動宇宙機の姿勢制御は，いわば短期的な戦略と中期的な戦略の組み合わせによって行われる．このような戦略の価値および方策は，本論文で検討した限りでは順方向 NN では表現することができないという結論に至った．考えられる対策としては，1 周期の判定を行い NN の入力として含めるか，RNN (Recurrent Neural Network) や LSTM (Long Short-Term Memory) のような時系列を学習することが可能な NN を用いることである．特に後者の強化学習と RNN/LSTM の組み合わせについては近年その高い効果が示されており，劣駆動宇宙機の姿勢制御への応用も大いに期待される．

No Free Lunch 定理の文脈では，本論文で検討した環境の決定論的性質以外にも，例え物理法則，ひいては空間の連続性や時間の一方向性なども一種の事前知識であると考えられるが，これら汎用事前知識の理解とアルゴリズムへの適用は，単に強化学習の社会実装を推し進めるだけでなく，汎用人工知能 (AGI, Artificial General Intelligence) の実現に不可欠なものである．本論文の成果がそのような道程の一端となることを期待する．

Bibliography

[1] Auger, A. and Teytaud, O.: Continuous lunches are free!, *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, London, UK, 2007.

[2] Auger, A. and Teytaud, O.: Continuous lunches are free plus the design of optimal optimization algorithms, *Algorithmica*, **57**, 1 (2010), pp.121–146.

[3] Baird, L. C.: Reinforcement learning in continuous time: Advantage updating, *Proceeding of IEEE International Conference on Neural Networks*, **4** (1994), pp.2448–2453.

[4] Banach, S.: Sur les opérations dans les ensembles abstraits et leur application aux équations intégrale, *Fundamenta Mathematicae*, **3**, 1 (1922), pp.133–181.

[5] Bellman, R. E.: A Markovian decision process, *Journal of Mathematics and Mechanics*, **6** (1957), pp.679–684.

[6] Bertsekas, D. P. and Tsitsikli, J. N.: Neuro-Dynamic Programming, Athena Scientific, Nashua, NH, USA, 1996.

[7] Bradtke, S. J.: Reinforcement learning applied to linear quadratic regulation, *Advances in Neural Information Processing Systems*, **5** (1992), pp.295–302.

[8] Brafman, R. I. and Tennenholtz, M.: R-MAX A general polynomial time algorithm for near-optimal reinforcement learning, *Journal of Machine Learning Research*, **3** (2002), pp.213-231.

[9] Dantzig, G. B.: Linear Programming and Extensions, Princeton University Press, Princeton, NJ, USA, 1963.

[10] Deisenroth, M. P. and Rasmussen, C. E.: PILCO: A model-based and data-efficient approach to policy search, *Proceedings of the 28th International Conference on Machine Learning*, Bellevue. WS, USA, 2011.

[11] Dorigo, M. and Stutzle, T.: Ant Colony Optimization, MIT Press, Cambridge, MA, USA, 2004.

[12] Everitt, T. et al.: Reinforcement learning with a corrupted reward channel, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017.

[13] García, J. and Fernández, F.: A comprehensive survey on safe reinforcement learning, *Journal of Machine Learning Research*, **15** (2015), pp.1437–1480.

[14] Gaskett, C.: Reinforcement learning under circumstances beyond its control, *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation*, Vienna, Austria, 2003.

[15] Ghaoui, L. E. and Nilim, A.: Robust control of Markov decision processes with uncertain transition matrices, *Operational Research*, **53**, 5 (2005), pp.780-798.

[16] Gu, S. et al.: Continuous deep Q-learning with model-based acceleration, *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, USA, 2016.

[17] Heffley, R. K. and Jewell, W. F.: Aircraft Handling Qualities Data, NASA CR-2144, 1972.

[18] Heger, M.: Consideration of risk in reinforcement learning, *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, USA, 1994.

[19] Holland, J. H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1975.

[20] Horri, N. and Palmer, P.: Practical implementation of attitude-control algorithms for an underactuated satelite, *Journal of Guidance, Control and Dynamics*, **35**, 1 (2012), pp. 40–50.

[21] Howard, R. A.: Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, USA, 1960.

[22] Howard, R. A. and Matheson, J. E.: Risk-sensitive Markov decision processes, *Management Science*, **18**, 7 (1972), pp.356–369.

[23] Iglesias, R. et al.: Supervised reinforcement learning: Application to a wall following behaviour in a mobile robot, *11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Benicassim, Spain, 1998.

[24] Jaakkola, T., Jordan, M. I. and Singh, S. P.: On the convergence of stochastic iterative dynamic programming algorithms, *Neural Computation*, **6**, 6 (1994), pp.1185–1201.

[25] Karaboga, D. and Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, **39** (2007), pp.459–471.

[26] Karmarkar, N.: A new polynomial-time algorithm for linear programming, *Combinatorica*, **4** (1984), pp373–395.

[27] Kephart, J. O.: A biologically inspired immune system for computers, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

[28] Kim, S. and Kim, Y.: Sliding mode stabilizing control law of underactuated spacecraft, *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, USA, 2000.

[29] Kirk, D. E.: Optimal Control Theory: An Introduction, Dover Publications, Mineola, NY, USA, 1970.

[30] Kirkpatrick, S., Gelatt Jr, C. D. and Vecchi, M. P.: Optimization by simulated annealing, *Science*, **220** (1983), pp.671–680.

[31] Köppen, M., Wolpert, D. H., and Macready W. G.: Remarks on a recent paper on the "No Free Lunch" theorems, *IEEE Transactions on Evolutionary Computation*, **5**, 3 (1995), pp.295–296.

[32] Lillicrap, T. P. et al.: Continuous control with deep reinforcement learning, *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[33] Lin, L. J.: Self-improving reactive agents based on reinforcement learning, planning and teaching, *Machine Learning*, **8** (1992), pp.293–321.

[34] Lockett, A. J.: A probabilistic reformulation of no free lunch: Continuous lunches are not free, *Evolutionary Computation*, **25**, 3 (2017), pp.503–528.

[35] Mnih, V. et al.: Playing Atari with deep reinforcement learning, *NIPS Deep Learning Workshop*, 2013.

[36] Mnih, V. et al.: Human-level control through deep reinforcement learning, *Nature*, **518** (2015), pp.529–533.

[37] Moreno, D. L. et al.: Using prior knowledge to improve reinforcement learning in mobile robotics, *Proceedings of Towards Autonomous Robotics Systems*, Essex, UK, 2004.

[38] Peng, B. et al.: Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, **1** (2018), pp.2182–2192.

[39] Pincus, M. :A Monte Carlo method for the approximate solution of certain types of constrained optimization problems, *Operations Research*, **18**, 6 (1970), pp.967–1235.

[40] Pontryagin, L. S et al.: The Mathematical Theory of Optimal Processes, John Wiley & Sons, Hoboken, NJ, USA, 1962.

[41] Puterman, M. L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, Hoboken, NJ, USA, 1994.

[42] Ribeiro, C. and Szepesvári, C.: Q-learning combined with spreading: Convergence and results, *Proceedings of the ISRF-IEE International Conference on Intelligent and Cognitive Systems*, Tehran, Iran, 1996.

[43] Riedmiller, M.: Neural fitted Q iteration – First experiences with a data efficient neural reinforcement learning method, *Proceedings of the 16th European conference on Machine Learning*, Porto, Portugal, 2005.

[44] Rummery, G. A. and Niranjan, M.: Online Q-learning using connectionist systems, TR 166, Cambridge University Engineering Department, 1994.

[45] Schumacher, C., Vose, M. D., and Whitley, L. D.: The no free lunch and description length, *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, USA, 2001.

[46] Silver, D. et al.: Deterministic policy gradient algorithms, *Proceedings of the 31st International Conference on International Conference on Machine Learning*, Beijing, China, 2014.

[47] Singh, S. and Kearns, M.: Near-optimal reinforcement learning in polynomial time, *Machine Learning*, **49** (2002), pp.209–232.

[48] Strehl, A. L. and Littman, M. L.: A theoretical analysis of model-based interval estimation, *Proceedings of the 22nd International Conference on Machine Learning*, Edinburgh, Scotland, 2012.

[49] Sutton, R. S.: Learning to predict by the methods of temporal differences, *Machine Learning*, **3**, 1 (1988), pp.9-44.

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

[51] Tamar, A., Xu, H. and Mannor, S.: Scaling up robust MDPs by reinforcement learning, *Proceedings of the 31 st International Conference on Machine Learning*, Beijing, China, 2014.

[52] Tsiotras, P. and Doumtchenko, V.: Control of spacecraft subject to actuator failures: State of the art and open problems, *Journal of the Astronautical Sciences*, **48**, 2–3 (2000), pp. 337–358.

[53] Vose, M. D.: Continuous lunches are not free, arXiv:1507.00581, 2015.

[54] Watkins, C. J. C. H.: Learning from delayed rewards, Ph.D. thesis, Cambridge University, 1989.

[55] Wie, B. and Barba, P. M.: Quaternion feedback for spacecraft large angle maneuvers, *The Journal of Guidance, Control, and Dynamics*, **8**, 3 (1985), pp.360–365.

[56] Wie, B. and Arapostathis, A.: Quaternion feedback regulator for spacecraft eigenaxis rotations, *The Journal of Guidance, Control, and Dynamics*, **12**, 3 (1989), pp.375–380.

[57] Wolpert, D. H. and Macready, W. G.: No free lunch theorems for search, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[58] Wolpert, D. H. and Macready, W. G.: No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, **1**, 1 (1997), pp.67–82.

[59] Zhang, S., Boehmer, W. and Whiteson, S.: Deep residual reinforcement learning, *International Conference on Autonomous Agents and Multiagent Systems*, 2020.

[60] Zou, A.M., Kumar, K. D. and de Ruiter, A.H.J.: Spacecraft attitude control using two control torques, *Information Sciences*, **408** (2017), pp.23–40.

Appendix A

A.1 証明で用いる公式

Definition A.1 (条件付き確率)
B が起こったときの A の条件付き確率を $P(A \mid B) = P(A,\, B)/P(B)$ と定義する.

Definition A.2 (独立)
A, B が独立かつ $P(B) \neq 0$ ならば $P(A \mid B) = P(A)$.

Formula A.1 (条件付き同時確率)
$P(A,\, B \mid C) = P(A \mid B,\, C)P(B \mid C)$.

Formula A.2 (全確率の公式)
$B_i \cap B_j = \phi\ (i \neq j)$ かつ $B_1 \cup \cdots \cup B_n = \Omega$ ならば

$$P(A) = \sum_{i=1}^{n} P(A,\, B_i) = \sum_{i=1}^{n} P(A \mid B_i)P(B_i),$$
$$P(A \mid C) = \sum_{i=1}^{n} P(A,\, B_i \mid C) = \sum_{i=1}^{n} P(A \mid B_i,\, C)P(B_i \mid C).$$

A.2 Proposition 2.2 (Bellman 方程式) の証明

累積報酬の再帰的な関係を用いて式 (2.1.7) を変形すると

$$\begin{aligned}
V^{\pi}(s) &= E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+1} \mid s_t = s\right] \\
&= E_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_t = s\right] \\
&= \sum_{a}\pi(a \mid s)\sum_{s'}P(s' \mid s,\, a)E_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] \\
&= \sum_{a}\pi(a \mid s)\sum_{s'}P(s' \mid s,\, a)\Bigg(E_{\pi}\left[r_{t+1} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] \\
&\qquad\qquad + \gamma E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right]\Bigg) \qquad \text{(A.2.1)}
\end{aligned}$$

となる．式 (A.2.1) の第 1 の期待値は式 (2.1.3) より

$$\begin{aligned}
E_{\pi}\left[r_{t+1} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] &= E\left[r_{t+1} \mid s_t = s,\, a_t = a,\, s_{t+1} = s',\, \pi\right] \\
&= E\left[r_{t+1} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] \\
&= R(s,\, a,\, s') \qquad \text{(A.2.2)}
\end{aligned}$$

であり，Markov 性に注意すれば第 2 の期待値は式 (2.1.7) より

$$\begin{aligned}
E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] &= E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_{t+1} = s'\right] \\
&= V^{\pi}(s') \qquad \text{(A.2.3)}
\end{aligned}$$

である．式 (A.2.1)–式 (A.2.3) は当然任意の $s \in \mathcal{S}$ について成立するので，式 (A.2.2) および式 (A.2.3) を式 (A.2.1) に代入すると，状態価値関数 $V^{\pi}(s)$ についての Bellman 方程式

$$V^{\pi}(s) = \sum_{a}\pi(a \mid s)\sum_{s'}P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^{\pi}(s')),\ \forall s \in \mathcal{S} \qquad \text{(A.2.4)}$$

が得られる．さらに，式 (A.2.4) に式 (2.1.9) の関係式を適用すれば

$$\begin{aligned}
Q^{\pi}(s,\, a) &= \sum_{s'}P(s' \mid s,\, a)\left(R(s,\, a,\, s') + \gamma V^{\pi}(s')\right) \qquad \text{(A.2.5)} \\
&= \sum_{s'}P(s' \mid s,\, a)\left(R(s,\, a,\, s') + \gamma\sum_{a'}\pi(a' \mid s')Q^{\pi}(s',\, a')\right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A} \qquad \text{(A.2.6)}
\end{aligned}$$

となり，行動価値関数 $Q^{\pi}(s,\, a)$ についての Bellman 方程式が得られる．

A.3 Proposition 2.3 (最適 Bellman 方程式) の証明

Appendix A.2 と同様にして式 (2.1.15) を変形すると

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,\, a) \\
&= \max_a E_{\pi^*}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s,\, a_t = a\right] \\
&= \max_a E_{\pi^*}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s,\, a_t = a\right] \\
&= \max_a \sum_{s'} P(s' \mid s,\, a) E_{\pi^*}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] \\
&= \max_a \sum_{s'} P(s' \mid s,\, a)\Bigg(E_{\pi^*}\left[r_{t+1} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right] \\
&\qquad\qquad + \gamma E_{\pi^*}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s,\, a_t = a,\, s_{t+1} = s'\right]\Bigg) \\
&= \max_a \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^*(s')),\ \forall s \in \mathcal{S}
\end{aligned}
\tag{A.3.1}
$$

として，最適状態価値関数 $V^*(s)$ についての最適 Bellman 方程式が得られる．さらに，式 (A.2.5)，式 (A.2.6) と同様にして式 (A.3.1) の変形過程に式 (2.1.15) の関係式を適用すれば

$$
\begin{aligned}
Q^*(s,\, a) &= \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V^*(s')) \\
&= \sum_{s'} P(s' \mid s,\, a)\left(R(s,\, a,\, s') + \gamma \max_{a'} Q^*(s',\, a')\right),\ \forall s \in \mathcal{S}\ \forall a \in \mathcal{A}
\end{aligned}
\tag{A.3.2}
$$

となり，最適行動価値関数 $Q^*(s)$ についての最適 Bellman 方程式が得られる．

A.4 Theorem 2.1 (Bellman 作用素の縮小性) の証明

任意の関数 V_1 および V_2 に最適 Bellman 作用素 $\mathcal{T}_*$ を作用させたときの出力の距離について

$$
\begin{aligned}
&|(\mathcal{T}_*V_1)(s) - (\mathcal{T}_*V_2)(s)| \\
&= \left| \max_a \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V_1(s')) - \max_a \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V_2(s')) \right| \\
&\leq \max_a \left| \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V_1(s')) - \sum_{s'} P(s' \mid s,\, a)(R(s,\, a,\, s') + \gamma V_2(s')) \right| \\
&= \gamma \max_a \sum_{s'} P(s' \mid s,\, a)\, |V_1(s') - V_2(s')| \qquad \text{(A.4.1)}
\end{aligned}
$$

が成り立つ．なお，不等号については $\forall f\ \forall g,\ |\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$ なる関係を用いた．関数 f に対する L^∞ ノルムの定義は

$$
||f||_\infty = \max_x |f(x)| \qquad \text{(A.4.2)}
$$

であり，$f(x) - g(x) \leq \max_{x'} |f(x') - g(x')|$ であることから

$$
\begin{aligned}
||\mathcal{T}_*V_1 - \mathcal{T}_*V_2||_\infty &= \max_s |(\mathcal{T}_*V_1)(s) - (\mathcal{T}_*V_2)(s)| \\
&\leq \gamma \max_{s,\, a} \sum_{s'} P(s' \mid s,\, a)\, |V_1(s') - V_2(s')| \\
&\leq \gamma \max_{s'} |V_1(s') - V_2(s')| \\
&= \gamma\, ||V_1 - V_2||_\infty \qquad \text{(A.4.3)}
\end{aligned}
$$

となり，式 (2.1.25) は示された．なお，Bellman 作用素 $\mathcal{T}_\pi$ は線形作用素であるから，より単純に

$$
\begin{aligned}
||\mathcal{T}_\pi V_1 - \mathcal{T}_\pi V_2||_\infty &= \max_s |(\mathcal{T}_\pi V_1)(s) - (\mathcal{T}_\pi V_2)(s)| \\
&= \gamma \max_s \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s,\, a)\, |V_1(s') - V_2(s')| \\
&\leq \gamma \max_{s'} |V_1(s') - V_2(s')| \\
&= \gamma\, ||V_1 - V_2||_\infty \qquad \text{(A.4.4)}
\end{aligned}
$$

として縮小作用素であることが示される．

行動価値関数 $Q(s,\, a)$ についての Bellman 作用素 $\mathcal{H}_\pi$ および最適 Bellman 作用素 $\mathcal{H}_*$ も，同様の手法で縮小性を示すことが可能である．任意の関数 Q_1 および Q_2 に $\mathcal{H}_*$ を作用させたときの出力の距離について

$$
\begin{aligned}
|(\mathcal{H}_* Q_1)(s,\, a) - (\mathcal{H}_* Q_2)(s,\, a)| &= \left| \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \max_{a'} Q_1(s',\, a') \right) \right. \\
&\qquad \left. - \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \max_{a'} Q_2(s',\, a') \right) \right| \\
&= \gamma \left| \sum_{s'} P(s' \mid s,\, a) \left(\max_{a'} Q_1(s',\, a') - \max_{a'} Q_2(s',\, a') \right) \right| \\
&\leq \gamma \sum_{s'} P(s' \mid s,\, a) \left| \max_{a'} Q_1(s',\, a') - \max_{a'} Q_2(s',\, a') \right| \\
&\leq \gamma \sum_{s'} P(s' \mid s,\, a) \max_{s',\, a'} |Q_1(s',\, a') - Q_2(s',\, a')| \qquad \text{(A.4.5)}
\end{aligned}
$$

が成り立つ．したがって，

$$
\begin{aligned}
||\mathcal{H}_* Q_1 - \mathcal{H}_* Q_2||_\infty &= \max_{s,\, a} |(\mathcal{H}_* Q_1)(s,\, a) - (\mathcal{H}_* Q_2)(s,\, a)| \\
&\leq \gamma \max_{s,\, a} \sum_{s'} P(s' \mid s,\, a) \max_{s',\, a'} |Q_1(s',\, a') - Q_2(s',\, a')| \\
&\leq \gamma \max_{s',\, a'} |Q_1(s',\, a') - Q_2(s',\, a')| \\
&= \gamma\, ||Q_1 - Q_2||_\infty \qquad \text{(A.4.6)}
\end{aligned}
$$

となり，$\mathcal{H}_*$ は縮小作用素であることが示された．さらに，$\mathcal{H}_\pi$ についても

$$
\begin{aligned}
||\mathcal{H}_\pi Q_1 - \mathcal{H}_\pi Q_2||_\infty &= \max_{s,\, a} |(\mathcal{H}_\pi Q_1)(s,\, a) - (\mathcal{H}_\pi Q_2)(s,\, a)| \\
&= \gamma \max_{s,\, a} \sum_{s'} P(s' \mid s,\, a) \sum_{a'} \pi(a' \mid s')\, |Q_1(s',\, a') - Q_2(s',\, a')| \\
&\leq \gamma \max_{s',\, a'} |Q_1(s',\, a') - Q_2(s',\, a')| \\
&= \gamma\, ||Q_1 - Q_2||_\infty \qquad \text{(A.4.7)}
\end{aligned}
$$

となり，縮小作用素であることが示される．

A.5 Theorem 2.5 (方策改善定理) の証明

準備として，Bellman 方程式の期待値表現を導出する．式 (2.1.7) と全期待値の公式より

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+1} \mid s_t = s\right] \\ &= E_{\pi}\left[r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_t = s\right] \\ &= E_{\pi}\left[r_{t+1} + \gamma E_{\pi}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+2} \mid s_{t+1} = s'\right] \mid s_t = s\right] \\ &= E_{\pi}[r_{t+1} + \gamma V^{\pi}(s') \mid s_t = s] \end{aligned} \tag{A.5.1}$$

となり，式 (2.1.16) と等価な式が得られる．また，行動価値関数 $Q^{\pi}(s,\, a)$ についても同様にして

$$Q^{\pi}(s,\, a) = E_{\pi}[r_{t+1} + \gamma V^{\pi}(s') \mid s_t = s,\, a_t = a] \tag{A.5.2}$$

が得られる．これは式 (A.2.5) と等価である．式 (A.5.2) を用いて Proposition 2.5 の条件を変形すると

$$\begin{aligned} V^{\pi}(s) &\leq Q^{\pi}(s,\, \pi'(s)) \\ &= E_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s,\, a_t = \pi'(s)] \\ &= E_{\pi'}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s] \end{aligned} \tag{A.5.3}$$

となる．この不等式を $V^{\pi}(s)$ に再帰的に適用すると

$$\begin{aligned} V^{\pi}(s) &\leq E_{\pi'}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s] \\ &\leq E_{\pi'}[r_{t+1} + \gamma E_{\pi'}[r_{t+2} + \gamma V^{\pi}(s_{t+2})] \mid s_t = s] \\ &\leq E_{\pi'}[r_{t+1} + \gamma E_{\pi'}[r_{t+2} + \gamma E_{\pi'}[r_{t+3} + \gamma V^{\pi}(s_{t+3})]] \mid s_t = s] \\ &\qquad\qquad \vdots \\ &\leq E_{\pi'}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \cdots \mid s_t = s] \\ &= E_{\pi'}\left[\sum_{k=0}^{\infty}\gamma^{k}r_{t+k+1} \mid s_t = s\right] \\ &= V^{\pi'}(s) \end{aligned} \tag{A.5.4}$$

となり，Proposition 2.5 は証明された．

A.6 Proposition 2.8 (ϵ-greedy 方策による方策改善) の証明

式 (2.2.17) および式 (2.2.18) より，ϵ-greedy 方策では確率 $1-\epsilon$ で greedy な行動 $\arg\max_a Q^\pi(s,\, a)$ が選ばれ，残りの行動についての重みは $\epsilon/|\mathcal{A}|$ であった．したがって，$Q^\pi(s,\, \pi'(s))$ を展開すると

$$\begin{aligned} Q^\pi(s,\, \pi'(s)) &= \sum_a \pi'(a \mid s) Q^\pi(s,\, a) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^\pi(s,\, a) + (1-\epsilon) Q^\pi(s,\, \arg\max_a Q^\pi(s,\, a)) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^\pi(s,\, a) + (1-\epsilon) \max_a Q^\pi(s,\, a),\ \forall s \in \mathcal{S} \end{aligned} \tag{A.6.1}$$

となる．ここで

$$\sum_a \frac{\pi(a \mid s) - \dfrac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} = \frac{1-\epsilon}{1-\epsilon} = 1 \tag{A.6.2}$$

であるから，$Q^\pi(s,\, a)$ の重み付き加重平均について次の不等式が成り立つ．

$$\sum_a \frac{\pi(a \mid s) - \dfrac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} Q^\pi(s,\, a) \leq \max_a Q^\pi(s,\, a) \tag{A.6.3}$$

式 (A.6.3) を式 (A.6.1) に適用すると

$$\begin{aligned} Q^\pi(s,\, \pi'(s)) &= \sum_a \pi'(a \mid s) Q^\pi(s,\, a) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^\pi(s,\, a) + (1-\epsilon) \max_a Q^\pi(s,\, a) \\ &= \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^\pi(s,\, a) + (1-\epsilon) \max_a Q^\pi(s,\, a) \sum_a \frac{\pi(a \mid s) - \dfrac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} \\ &\geq \frac{\epsilon}{|\mathcal{A}|} \sum_a Q^\pi(s,\, a) + (1-\epsilon) \sum_a \frac{\pi(a \mid s) - \dfrac{\epsilon}{|\mathcal{A}|}}{1-\epsilon} Q^\pi(s,\, a) \\ &= \sum_a \pi(a \mid s) Q^\pi(s,\, a) \\ &= V^\pi(s),\ \forall s \in \mathcal{S} \end{aligned} \tag{A.6.4}$$

となり，ϵ-greedy 方策は方策改善の条件を満たすことが示された．

A.7 Theorem 2.3 (TD(0) 予測の収束性) の証明

TD(0) 予測の収束性の証明では，確率過程についての次の Lemma を用いる．

Lemma A.1 (逐次的確率過程の収束性)
次の逐次的確率過程

$$\Delta_{t+1}(x) = \Delta_t(x) + \alpha_t(x)(F_t(x) - \Delta_t(x)) \tag{A.7.1}$$

は，以下の条件を満たすとき，ほとんど確実に 0 に収束する．

(1) 状態 x が有限
(2) $0 \leq \alpha_t(x) \leq 1$，$\sum_t \alpha_t(x) = \infty$，$\sum_t \alpha_t^2(x) < \infty$
(3) 任意の γ $(0 \leq \gamma < 1)$ に対して $||E[F_t(x) \mid P_t]||_w \leq \gamma ||\Delta_t||_w + c_t$ かつ $P\left(\lim_{t\to\infty} c_t = 0\right) = 1$
(4) 任意の定数 $k > 0$ に対して $V[F_t(x) \mid P_t] \leq k(1 + ||\Delta_t||_w)^2$

ここで，P_t は Δ_t，F_t，α_t の履歴 $P_t = \{\Delta_t, \Delta_{t-1}, \ldots, F_{t-1}, \ldots, \alpha_{t-1}, \ldots\}$ であり，$V[\cdot]$ は分散[*1]，$||\cdot||_w$ は任意の重み付き最大値ノルムを表す．
Proof. Jaakkola[24] を参照
Remark. 式 (A.7.1) からも分かるように，この定理は確率的勾配降下法の一種である Robbins-Monro 法 $\Delta_{t+1}(x) = \Delta_t(x) - \alpha_t \Delta_t(x)$ の収束条件[*2]の拡張である．条件 (2) における学習率 $\alpha_t(x)$ は，k 回目の iteration における状態 x に対する学習率を意味する．したがって，条件 (2) は，全ての状態 $x \in \mathcal{X}$ を無限回訪問することを要請する．なお，条件 (3) は $F_t(x)$ の縮小性に関する条件であり，条件 (4) は $F_t(x)$ の有界性に関する条件である．

TD(0) 予測の更新式は次式で与えられる．

$$V_{t+1}(s) = V_t(s) + \alpha_t(s)\left(r' + \gamma V_t(s') - V_t(s)\right) \tag{A.7.2}$$

ここで，$\Delta_t(s)$ を予測誤差として

$$\Delta_t(s) = V_t(s) - V^\pi(s) \tag{A.7.3}$$
$$F_t(s) = r' + \gamma V_t(s') - V^\pi(s) \tag{A.7.4}$$

とおくと，式 (A.7.2) は

$$\Delta_{t+1}(s) = \Delta_t(s) + \alpha_t(s)(F_t(s) - \Delta_t(s)) \tag{A.7.5}$$

となる．状態価値関数 $V(s)$ についての Bellman 作用素の定義 (式 (2.1.21)) より，条件 (3) は

$$\begin{aligned} E[F_t(s) \mid P_t] &= \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a)\left(R(s, a, s') + \gamma V_t(s') - V^\pi(s)\right) \\ &= (\mathcal{T}_\pi V_t)(s) - V^\pi(s) = (\mathcal{T}_\pi V_t)(s) - (\mathcal{T}_\pi V^\pi)(s) \end{aligned} \tag{A.7.6}$$

となる．

[*1] 状態価値関数と記号が重複するが，本論文中では鉤括弧を用いた表記 $V[\cdot]$ は全て分散を表し，それ以外 ($V(\cdot)$ または単に V) は状態価値関数を表す．
[*2] このことから，条件 (2) は Robbins-Monro 条件とも呼ばれる．

$\mathcal{T}_\pi$ の縮小性は Appendix A.4 で既に示した通りであるから，

$$||E[F_t(s) \mid P_t]||_\infty \leq \gamma||V_t - V^\pi||_\infty = \gamma||\Delta_t||_\infty \tag{A.7.7}$$

となり，条件 (3) を満たすことが示された．条件 (4) については，分散の定義 $V[X] = E[(X - E[X])^2]$ より

$$\begin{aligned} V[F_t(s) \mid P_t] &= E[(r' + \gamma V_t(s') - V^\pi(s) - ((\mathcal{T}_\pi V_t)(s) - V^\pi(s)))^2] \\ &= E[(r' + \gamma V_t(s') - (\mathcal{T}_\pi V_t)(s))^2] \\ &= V[r' + \gamma V_t(s')] \end{aligned} \tag{A.7.8}$$

となり，仮定 (iii) より報酬 r は有界であるから，直ちに条件 (4) を満たすことが示される．

条件 (1) および (2) はそれぞれ仮定 (i) および (ii) に対応するため，Lemma A.1 より，TD(0) 予測において $V(s)$ はほとんど確実に $V^\pi(s)$ に収束する．

A.8 Theorem 2.4 (SARSA の収束性) の証明

TD(0) 予測の場合と同様に，証明には Lemma A.1 を用いる．
SARSA の更新式は次式で与えられる．

$$Q_{t+1}(s,\, a) \leftarrow Q_t(s,\, a) + \alpha_t(s,\, a)(r' + \gamma Q_t(s',\, a') - Q_t(s,\, a)) \tag{A.8.1}$$

ここで，$\Delta_t(s,\, a)$ を予測誤差として

$$\Delta_t(s,\, a) = Q_t(s,\, a) - Q^*(s,\, a) \tag{A.8.2}$$

$$F_t(s,\, a) = r' + \gamma Q_t(s',\, a') - Q^*(s,\, a) \tag{A.8.3}$$

とおくと，式 (A.8.1) は

$$\Delta_{t+1}(s,\, a) = \Delta_t(s,\, a) + \alpha_t(s,\, a)(F_t(s,\, a) - \Delta_t(s,\, a)) \tag{A.8.4}$$

となる．さらに，$F_t(s,\, a)$ について

$$\begin{aligned} F_t(s,\, a) &= r' + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) + \gamma(Q_t(s',\, a') - \max_{a'} Q_t(s',\, a')) \\ &= r' + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) + C_t \end{aligned} \tag{A.8.5}$$

と再定義する．状態価値関数 $Q(s,\, a)$ についての最適 Bellman 作用素の定義 (式 (2.1.24)) より，条件 (3) は

$$\begin{aligned} E[F_t(s,\, a) \mid P_t] &= \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) \right) + E[C_t \mid P_t] \\ &= (\mathcal{H}_* Q_t)(s,\, a) - Q^*(s,\, a) + E[C_t \mid P_t] \\ &= (\mathcal{H}_* Q_t)(s,\, a) - (\mathcal{H}_* Q^*)(s,\, a) + E[C_t \mid P_t] \end{aligned} \tag{A.8.6}$$

となる．$\mathcal{H}_*$ の縮小性は Appendix A.4 で既に示した通りであるから，

$$||E[F_t(s) \mid P_t]||_\infty \leq \gamma ||Q_t - Q^*||_\infty + ||E[C_t \mid P_t]||_\infty \leq \gamma ||\Delta_t||_\infty + ||E[C_t \mid P_t]||_\infty \tag{A.8.7}$$

となる．仮定 (iv) より，$C_t = \gamma(Q_t(s',\, a^{\mathcal{O}}$もとでの主張である $\prime) - \max_{a'} Q_t(s',\, a'))$ はほとんど確実に 0 に収束するので，条件 (3) を満たすことが示された．条件 (4) については，分散の定義 $V[X] = E[(X - E[X])^2]$ より

$$\begin{aligned} V[F_t(s,\, a) \mid P_t] &= E[(r' + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) + C_t \\ &\qquad - ((\mathcal{H}_* Q_t)(s,\, a) - Q^*(s,\, a) + E[C_t \mid P_t]))^2] \\ &= E[(r' + \gamma \max_{a'} Q_t(s',\, a') + C_t - ((\mathcal{H}_* Q_t)(s,\, a) + E[C_t \mid P_t]))^2] \\ &= V[r' + \gamma \max_{a'} Q_t(s',\, a') + C_t] \end{aligned} \tag{A.8.8}$$

となり，仮定 (iii) より r は有界であるから，直ちに条件 (4) を満たすことが示される．

条件 (1) および (2) はそれぞれ仮定 (i) および (ii) に対応するため，Lemma A.1 より，SARSA において $Q(s,\, a)$ ほとんど確実に $Q^*(s,\, a)$ に収束する．

A.9 Theorem 2.5 (Q-learningの収束性) の証明

TD(0) 予測および SARSA の場合と同様に，証明には Lemma A.1 を用いる．
Q-learing の更新式は次式で与えられる．

$$Q_{t+1}(s,\, a) \leftarrow Q_t(s,\, a) = \alpha_t(s,\, a)(r' + \gamma \max_{a'} Q_t(s',\, a') - Q_t(s,\, a)) \tag{A.9.1}$$

ここで，$\Delta_t(s,\, a)$ を予測誤差として

$$\Delta_t(s,\, a) = Q_t(s,\, a) - Q^*(s,\, a) \tag{A.9.2}$$

$$F_t(s,\, a) = r' + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) \tag{A.9.3}$$

とおくと，式 (A.8.1) は

$$\Delta_{t+1}(s,\, a) = \Delta_t(s,\, a) + \alpha_t(s,\, a)(F_t(s,\, a) - \Delta_t(s,\, a)) \tag{A.9.4}$$

となる．状態価値関数 $Q(s,\, a)$ についての最適 Bellman 作用素の定義 (式 (2.1.24)) より，条件 (3) は

$$\begin{aligned} E[F_t(s,\, a) \mid P_t] &= \sum_{s'} P(s' \mid s,\, a) \left(R(s,\, a,\, s') + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) \right) \\ &= (\mathcal{H}_* Q_t)(s,\, a) - Q^*(s,\, a) \\ &= (\mathcal{H}_* Q_t)(s,\, a) - (\mathcal{H}_* Q^*)(s,\, a) \end{aligned} \tag{A.9.5}$$

となる．$\mathcal{H}_*$ の縮小性は Appendix A.4 で既に示した通りであるから，

$$||E[F_t(s) \mid P_t]||_\infty \leq \gamma ||Q_t - Q^*||_\infty \leq \gamma ||\Delta_t||_\infty \tag{A.9.6}$$

となり，条件 (3) を満たすことが示された．条件 (4) については，分散の定義 $V[X] = E[(X - E[X])^2]$ より

$$\begin{aligned} V[F_t(s,\, a) \mid P_t] &= E[(r' + \gamma \max_{a'} Q_t(s',\, a') - Q^*(s,\, a) - ((\mathcal{H}_* Q_t)(s,\, a) - Q^*(s,\, a)))^2] \\ &= E[(r' + \gamma \max_{a'} Q_t(s',\, a') - (\mathcal{H}_* Q_t)(s,\, a))^2] \\ &= V[r' + \gamma \max_{a'} Q_t(s',\, a')] \end{aligned} \tag{A.9.7}$$

となり，仮定 (iii) より r は有界であるから，直ちに条件 (4) を満たすことが示される．

条件 (1) および (2) はそれぞれ仮定 (i) および (ii) に対応するため，Lemma A.1 より，Q-learning において $Q(s,\, a)$ ほとんど確実に $Q^*(s,\, a)$ に収束する．

A.10 Theorem 3.1 (探索における No Free Lunch 定理) の証明

$\sum_f P(d_k^y \mid f, k, \mathscr{A})$ が $\mathscr{A}$ に依らないことを数学的帰納法を用いて示す.
$k = 1$ のとき，d_1^x は $\mathscr{A}$ によって与えられる．したがって，

$$P(d_1^y \mid f, k=1, \mathscr{A}) = P(d_1^y \mid f, k=1, \mathscr{A}, d_1^x) = P(d_1^y \mid f(d_1^x)) = \begin{cases} 0 & (d_1^y \neq f(d_1^x)) \\ 1 & (d_1^y = f(d_1^x)) \end{cases} \tag{A.10.1}$$

である．$d_1^y = f(d_1^x)$ を満たすような $f|_{\mathcal{X}\setminus d_1^x} : \mathcal{X} \setminus d_1^x \to \mathcal{Y}$ は $\mathcal{F}$ の中に $|\mathcal{Y}|^{|\mathcal{X}|-1}$ 個存在するため，

$$\sum_f P(d_1^y \mid f, k=1, \mathscr{A}) = \sum_f P(d_1^y \mid f(d_1^x)) = |\mathcal{Y}|^{|\mathcal{X}|-1} \tag{A.10.2}$$

となり，明らかに $\mathscr{A}$ に依らない.

次に，$\sum_f P(d_k^y \mid f, k, \mathscr{A})$ が $\mathscr{A}$ に依らないと仮定したとき，$\sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A})$ も $\mathscr{A}$ に依らないことを示す．公式 A.1 より

$$\begin{aligned} \sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) &= \sum_f P(d_k^y, d_{k+1}^y(k+1) \mid f, k+1, \mathscr{A}) \\ &= \sum_f P(d_{k+1}^y(k+1) \mid d_k^y, f, k+1, \mathscr{A}) P(d_k^y \mid f, k+1, \mathscr{A}) \end{aligned} \tag{A.10.3}$$

である．公式 A.2 を用いて $P(d_{k+1}^y(k+1) \mid d_k^y, f, k+1, \mathscr{A})$ を x_{k+1} に関して周辺化すると

$$\begin{aligned} P(d_{k+1}^y(k+1) \mid d_k^y, f, k+1, \mathscr{A}) &= \sum_{x_{k+1}} P(d_{k+1}^y(k+1) \mid d_k^y, f, k+1, \mathscr{A}, x_{k+1}) \\ &\qquad \times P(x_{k+1} \mid d_k^y, f, k+1, \mathscr{A}) \\ &= \sum_{x_{k+1}} P(d_{k+1}^y(k+1) \mid f, x_{k+1}) P(x_{k+1} \mid d_k^y, f, k+1, \mathscr{A}) \\ &= \sum_{x_{k+1}} P(d_{k+1}^y(k+1) \mid f(x_{k+1})) P(x_{k+1} \mid d_k^y, f, k+1, \mathscr{A}) \end{aligned} \tag{A.10.4}$$

となる．同様にして，$P(x_{k+1} \mid d_k^y, f, k+1, \mathscr{A})$ を d_k^x に関して周辺化すると

$$\begin{aligned} P(x_{k+1} \mid d_k^y, f, k+1, \mathscr{A}) &= \sum_{d_k^x} P(x_{k+1} \mid d_k^x, d_k^y, f, k+1, \mathscr{A}) P(d_k^x \mid d_k^y, f, k+1, \mathscr{A}) \\ &= \sum_{d_k^x} P(x_{k+1} \mid d_k, \mathscr{A}) P(d_k^x \mid d_k^y, f, k+1, \mathscr{A}) \\ &= \sum_{d_k^x} P(x_{k+1} \mid \mathscr{A}(d_k)) P(d_k^x \mid d_k^y, f, k+1, \mathscr{A}) \end{aligned} \tag{A.10.5}$$

となる．式 (A.10.4) および式 (A.10.5) を式 (A.10.3) に代入すると，

$$\begin{aligned} \sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) = \sum_f \sum_{x_{k+1}} \sum_{d_k^x} & P(d_{k+1}^y(k+1) \mid f(x_{k+1})) P(x_{k+1} \mid \mathscr{A}(d_k)) \\ & \times P(d_k^x \mid d_k^y, f, k+1, \mathscr{A}) P(d_k^y \mid f, k+1, \mathscr{A}) \end{aligned} \tag{A.10.6}$$

が得られる．ここで，公式 A.1 および公式 A.2 より

$$P(d_k^x \mid d_k^y, f, k+1, \mathscr{A})P(d_k^y \mid f, k+1, \mathscr{A}) = P(d_k^x, d_k^y \mid f, k+1, \mathscr{A}) \\ = P(d_k \mid f, k+1, \mathscr{A}), \tag{A.10.7}$$

$$\sum_{x_{k+1}} P(d_{k+1}^y(k+1) \mid f(x_{k+1}))P(x_{k+1} \mid \mathscr{A}(d_k)) = P(d_{k+1}^y(k+1) \mid f(\mathscr{A}(d_k))) \tag{A.10.8}$$

であり，さらに k 回評価後の探索履歴 d_k が得られる確率は，当然，k 回までの試行に依存するので，

$$P(d_k \mid f, k+1, \mathscr{A}) = P(d_k \mid f, k, \mathscr{A}) \tag{A.10.9}$$

である．したがって，

$$\sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) = \sum_f \sum_{d_k^x} P(d_{k+1}^y(k+1) \mid f(\mathscr{A}(d_k)))P(d_k \mid f, k, \mathscr{A}) \tag{A.10.10}$$

となる[*3]．

ここで，$f : \mathcal{X} \to \mathcal{Y}$ を d_k^x への制限 $f|_{d_k^x} : d_k^x \to \mathcal{Y}$ と，$\mathcal{X} \setminus d_k^x$ への制限 $f|_{\mathcal{X}\setminus d_k^x} : \mathcal{X} \setminus d_k^x \to \mathcal{Y}$ に分割する．

$$\sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) = \sum_{d_k^x} \sum_{f|_{d_k^x}} \sum_{f|_{\mathcal{X}\setminus d_k^x}} P(d_{k+1}^y(k+1) \mid f(\mathscr{A}(d_k)))P(d_k \mid f, k, \mathscr{A}). \tag{A.10.11}$$

なお，ここでの分割とは，集合 $\mathcal{F}$ の加法的な分割ではなく，f の始域の分割のことを指す．実際に数え上げてみると，$|\mathcal{Y}|^k$ 個の $f|_{d_k^x}$ それぞれに対して，$|\mathcal{Y}|^{|\mathcal{X}|-k}$ 個の $f|_{\mathcal{X}\setminus d_k^x}$ が対応することにより，$|\mathcal{Y}|^{|\mathcal{X}|}$ 個の f 全体を被覆していることが分かる．

$P(d_{k+1}^y(k+1) \mid f(\mathscr{A}(d_k)))$ および $P(d_k \mid f, k, \mathscr{A})$ の $f|_{d_k^x}$ および $f|_{\mathcal{X}\setminus d_k^x}$ に対する独立性と従属性に着目すると，$P(d_k \mid f, k, \mathscr{A})$ は明らかに $f|_{d_k^x}$ のみに従属である．また，仮定より $\mathscr{A}(d_k) \notin d_k^x$，すなわち $\mathscr{A}(d_k) \in \mathcal{X} \setminus d_k^x$ であるから，$P(d_{k+1}^y(k+1) \mid f(\mathscr{A}(d_k)))$ は $f|_{\mathcal{X}\setminus d_k^x}$ のみに従属である．したがって，

$$\sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) = \sum_{d_k^x} \sum_{f|_{d_k^x}} P(d_k \mid f|_{d_k^x}, k, \mathscr{A}) \sum_{f|_{\mathcal{X}\setminus d_k^x}} P(d_{k+1}^y(k+1) \mid f|_{\mathcal{X}\setminus d_k^x}(\mathscr{A}(d_k))) \tag{A.10.12}$$

と変形することができる．$d_{k+1}^y(k+1) = f|_{\mathcal{X}\setminus d_k^x}(\mathscr{A}(d_k))$ を満たすような $f|_{\mathcal{X}\setminus d_k^x}$ すなわち $f|_{\mathcal{X}\setminus\{d_k^x, \mathscr{A}(d_k)\}}$ は数え上げることができ，$|\mathcal{Y}|^{|\mathcal{X}|-k-1}$ 個ある[*4] ので，

$$\sum_f P(d_{k+1}^y \mid f, k+1, \mathscr{A}) = |\mathcal{Y}|^{|\mathcal{X}|-k-1} \sum_{d_k^x} \sum_{f|_{d_k^x}} P(d_k \mid f|_{d_k^x}, k, \mathscr{A}) \tag{A.10.13}$$

[*3] ここまでは各変数の依存関係を用いて単純な式変形を繰り返したに過ぎず，式 (A.10.10) の意味するところも直観的に理解しやすいものとなっている．すなわち $k+1$ 回評価後の y の履歴 d_{k+1}^y が得られる確率は，d_k が与えられたときのアルゴリズムの出力の評価値 $f(\mathscr{A}(d_k))$ と，d_k が得られる確率とを乗したものを，すべての d_k について総和をとることで表現される．

[*4] $f|_{\mathcal{X}\setminus d_k^x}$ は $|\mathcal{X}|-k$ 個の $\mathcal{X} \setminus d_k^x$ から $|\mathcal{Y}|$ 個の y への写像なので，$|\mathcal{Y}|^{|\mathcal{X}|-k}$ 個あり，さらに始域を 1 つ決める $(\mathscr{A}(d_k))$ とその数は $|\mathcal{Y}|^{|\mathcal{X}|-k-1}$ 個となる．

となる．$f|_{d_k^x}$ についての総和を f についての総和に戻す．公式 A.2 より

$$\begin{aligned}\sum_{f|_{d_k^x}} P(d_k \mid f|_{d_k^x},\, k,\, \mathscr{A}) &= \sum_{f|_{d_k^x}} \sum_{f|_{\mathcal{X}\setminus d_k^x}} P(d_k \mid f|_{d_k^x},\, f|_{\mathcal{X}\setminus d_k^x},\, k,\, \mathscr{A}) P(f|_{\mathcal{X}\setminus d_k^x} \mid f|_{d_k^x},\, k,\, \mathscr{A}) \\ &= P(f|_{\mathcal{X}\setminus d_k^x} \mid f|_{d_k^x}) \sum_f P(d_k \mid f,\, k,\, \mathscr{A}) \\ &= \frac{1}{|\mathcal{Y}|^{|\mathcal{X}|-k}} \sum_f P(d_k \mid f,\, k,\, \mathscr{A}) \qquad \text{(A.10.14)}\end{aligned}$$

であるから，

$$\sum_f P(d_{k+1}^y \mid f,\, k+1,\, \mathscr{A}) = \frac{1}{|\mathcal{Y}|} \sum_{d_k^x} \sum_f P(d_k \mid f,\, k,\, \mathscr{A}) \qquad \text{(A.10.15)}$$

となる．さらに d_k についての総和を d_k^y についての総和に直すと，

$$\begin{aligned}\sum_f P(d_{k+1}^y \mid f,\, k+1,\, \mathscr{A}) &= \frac{1}{|\mathcal{Y}|} \sum_f \sum_{d_k^x} P(d_k^x,\, d_k^y \mid f,\, k,\, \mathscr{A}) \\ &= \frac{1}{|\mathcal{Y}|} \sum_f P(d_k^y \mid f,\, k,\, \mathscr{A}) \qquad \text{(A.10.16)}\end{aligned}$$

となるが，仮定により $\sum_f P(d_k^y \mid f,\, k,\, \mathscr{A})$ は $\mathscr{A}$ に依らないため，$\sum_f P(d_{k+1}^y \mid f,\, k+1,\, \mathscr{A})$ も $\mathscr{A}$ に依らない．以上により，探索における No Free Lunch 定理

$$\forall \mathscr{A}_1\, \forall \mathscr{A}_2\, \forall k \sum_f P(d_k^y \mid f,\, k,\, \mathscr{A}_1) = \sum_f P(d_k^y \mid f,\, k,\, \mathscr{A}_2)$$

は証明された．

Index

Acknowledgements

本論文を結ぶにあたり，本研究を遂行する上でご指導，ご鞭撻とご援助をいただいた方々に感謝の意を表します．東京大学大学院工学系研究科 航空宇宙工学専攻 土屋武司教授には指導教官として本研究の実施の機会を与えて戴き，その遂行にあたって終始，ご指導を戴きました．ここに深謝の意を表します．同専攻堀浩一教授，中須賀真一教授，矢入健久教授，東京都立大学 システムデザイン研究科 システムデザイン専攻 航空宇宙システム工学域 小島広久教授には副査としてご助言を戴くとともに，本論文の細部にわたりご指導を戴きました．ここに深謝の意を表します．本研究を進めるにあたり，土屋・伊藤研究室の皆様には種々のご指摘やご協力を頂きました．これら多くの方々に感謝の意を表します．最後に，本研究および本論文の作成を通じて，精神的にも経済的にも支えてくれた妻 歩に心から感謝します．