強化学習を用いた二足歩行ロボットのための制御回路とソフトウェア環境の構築

工学院大学大学院
工学研究科機械工学専攻

電子素子物理工学研究室所属
ＡＭ-06073　　横溝　惇
指導教員　疋田　光孝 教授
　　金丸　隆志 講師

目次

1　まえがき　1

2　研究目的　2

3　理論　4

3-1　強化学習　4
3-1-1　強化学習とは　5
3-1-2　強化学習の特徴　5
3-1-3　強化学習の応用　5
3-1-4　強化学習アルゴリズム　5
3-2　モデリング　8
3-3　CPG　10
3-3-1　CPG とは　10
3-3-2　CPG から二足歩行モデルへ　10
3-4　強化学習による CPG 調整　15
3-5　制御則の簡略化　22
3-6　評価関数 V の簡略化　23

4　ソフトウェア　25

4-1　ソフトウェア概要　26
4-2　シミュレータ学習実験　33

5　ハードウェア　43

5-1　ロボット動作原理　43
5-2　サーボモータ動作原理　44
5-3　PIC による PWM 制御　46
5-3-1　PIC とは　46
5-3-2　PIC の特性　46
5-3-3　PWM 生成　47
5-3-4　PIC プログラム　48
5-4　PIC での CPG 生成　51
5-4-1　PIC での Van der Pol 方程式　51
5-4-2　サーボモータの複数制御　52
5-5　制御回路Ⅰ　55
5-6　ロボットへの適用　56
5-7　センサ　58
5-7-1　ポジションキャプチャー　58
5-7-2　ジャイロセンサ　59
5-8　データ通信　60
5-8-1　PIC から PC への通信　60
5-8-2　PC から PIC への通信　65
5-9　制御回路Ⅱ　66

6　実機体での学習環境　70

7　結果と考察　75

参考文献　77

1 まえがき

現在、制御が困難とされている多自由度ロボットが多く存在し、ヒト型ロボットがその例の一つとして挙げられる。ヒト型二足歩行ロボットは、機体が床に固定されていない為極めて不安定であり、正確な制御器の設計が必要とされている。そして、この制御問題をベンチマークとして多くの研究が現在なされている。代表的なアプローチとして ZMP（Zero Moment Point）に基づくアプローチがある。このアプローチでは、ZMP の位置を支持脚多角形から飛び出さないように各関節を制御することで転倒を回避した運動を実現する。この方法により、各関節の制御量を決定するためにはあらかじめロボットの機構を正確に把握した上で、逆運動学問題を解き理想の制御量を算出する必要がある。しかしその際、特異点を回避するため膝を曲げたままの歩行パターンになってしまうこと、エネルギー消費量が大きいことなどが指摘されており、その歩行はヒトが行う自然な動作とは異なっている。

一方、ヒトらしい歩行を目指したものとしては、ヒトの関節運動を位相振動子としてエンコードし、この振動子とロボットの身体ダイナミクスとの引き込み現象を利用して、ヒト歩行によく似た自然な歩行生成に成功している。しかし、この手法では各関節への制御パラメータを決定することが大変である。

そこで、本研究では強化学習を用いて二足歩行にアプローチし、ヒトが二足歩行を獲得したように、実際のロボットで試行錯誤を通じて歩行獲得するための制御回路とソフトウェア環境の構築を目指していく。その手法として、制御回路ではマイコン PIC を用いてヒトの自律神経部分に相当するものを作り出し、ソフトウェアでは学習計算をするヒトで言う頭脳に相当する部分を作り上げていく。さらにこの制御回路とソフトウェアの相互間のデータ通信を行い、実際のロボットで学習を行うことのできる環境を目指していく。

2 研究目的

本研究は、ニューラルネットワークの学習法の一つである強化学習を用いて歩行獲得するロボットの作成を目指し、そのための制御回路とソフトウェア環境を構築することが目的である。現在、強化学習を用いて二足歩行をさせるという研究は少なからず行われているが、未だに実際のロボットで二足歩行の獲得に成功した例は少ない。これは、歩行時の状態をフィードバックし学習に用いる際の計算の遅れや、実際の歩行獲得までに膨大な試行が必要とされている等、様々な原因があると考えられる。

そこで本研究では、強化学習を用いて歩行獲得するロボットを作成するためのソフトウェア環境として、様々な学習条件に対応できる学習シミュレータを作成して実際のロボットで学習できるようにするための枠組みを確立することを目的の 1 つとする。さらに、制御回路では学習した情報をもとに歩行するための制御回路、および学習に必要な情報を歩行と平行してソフトウェアにフィードバックする回路を設計することも目的とする。

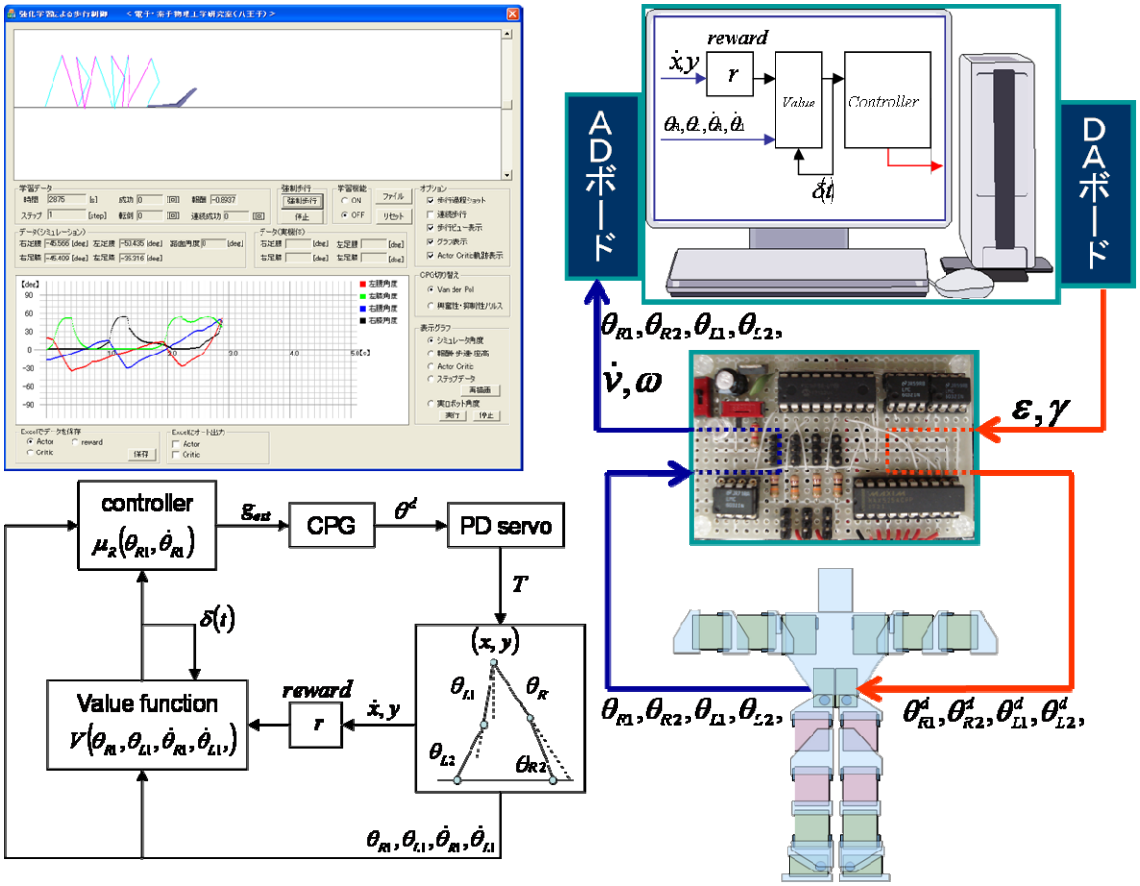


図 1　研究目的

ここで目的達成のための主な手法・研究の流れについて簡単に解説していく。

まず本研究は学習を用いて二足歩行を獲得するロボットのソフトウェア環境として、学

習シミュレータを作成した（図 1 左上）。この学習シミュレータ作成の構成要素として予め、学習実験をするために必要なモデル設計（詳細は **3-2**）、歩行を司る周期的信号CPGの生成（詳細は **1 章**）、モデルからの状態変数をフィードバックさせ学習していく学習アルゴリズムの確立（図 1 左下、詳細は **3-4**）が主な課題となり、付属構成要素としてモデルの視覚化（詳細は **4-1**）、実験データの取得・グラフ化（詳細は **4-1**）、制御回路との通信を行うプログラム作成等を行った。

次に実際のロボットで歩行を実現するための制御回路の作成をした。本研究では制御回路の核となるものとしてPICを用い（詳細は **5 章**）、このPICでロボットの動力であるサーボモータの制御（詳細は **5-3**）を行った。さらにシミュレータと同様にCPGをPICで生成し（詳細は **5-4**）、ロボットの状態を取得するためのセンサの導入（詳細は **5-7**）、PCとのデータ通信を行うための回路設計（詳細は **5-8**）を行った。

最後にソフトウェア・ハードウェアを統一し（図 1 右）、実際のロボットで歩行を獲得するための環境の構築を試みた。

以上が本研究の目的を達成のための手法であり、以後これらの手法について詳細に解説していく。

3 理論

3-1　　強化学習

3-1-1 強化学習とは

強化学習とは、試行錯誤を通じて環境に適応する学習制御の枠組みである。教師付き学習とは異なり、状態入力に対する正しい行動出力を明示的に示す教師が存在しない。その代わりに報酬というスカラーの情報を手がかりに学習する。

学習主体「エージェント」と制御対象「環境」は以下のやりとりを行う。

- エージェントは時刻 t において環境の状態観測 $S(t)$ に応じて意思決定を行い、行動 $a(t)$ を出力する
- エージェントの行動により、環境は $S(t+1)$ へ状態遷移し、その遷移に応じた報酬 $r(t)$ をエージェントへ与える
- 時刻 t を $t+1$ に進めてステップ 1 に戻る

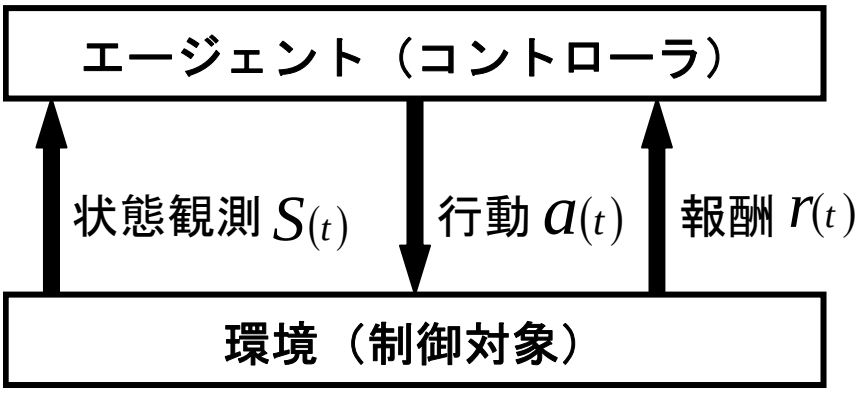


図 2 強化学習におけるエージェント・環境相互作用

エージェントは利得（*return*： 最も単純な場合、報酬の総計）の最大化を目的として、状態観測から行動出力へのマッピング（政策（ *policy*）と呼ばれる）を獲得する。環境とエージェントには一般に下記の性質が想定される。

- エージェントは予め環境に関する知識を持たない
- 環境の状態遷移は確率的
- 報酬の与えられた方は確率的
- 状態遷移を繰り返した後で報酬にたどり着くような、段取り的な行動を必要とする環境（報酬の遅れ）

3-1-2 強化学習の特徴

強化学習が注目を集める理由の一つは、不確実性のある環境を扱っている点にある。多くの実世界の制御問題では、不確実性のある扱いは厄介である。もう一つの理由は、報酬に遅れが存在し、離散的な状態遷移も含んだ段取り的な制御則の獲得を行う点にある。設計者がゴール状態で報酬を与えるという形で、所望のタスクをエージェントに指示しておけば、ゴールへの到達方法はエージェントの試行錯誤学習によって自動的に獲得される。つまり設計者が「何をすべきか」をエージェントに報酬という形で指示しておけば「どのように実現するか」をエージェントが学習によって自動的に獲得する枠組みになっている。

3-1-3 強化学習の応用

前記のような特徴から、強化学習は以下のようなことが応用上期待できる。

3-1-3-1　　制御プログラミングの自動化・省力化

環境に不確実性や計測不能な未知のパラメータが存在すると、タスクの達成方法やゴールへの到着方法は設計者にとって自明ではない。よってロボットへタスクを遂行するための制御規則をプログラムすることは設計者にとって重労働である。ところが、達成すべき目標を報酬によって指示することは前記に比べればはるかに簡単である。そのため、タスク遂行のためのプログラミングを強化学習で自動化することにより、設計者の負担軽減が期待できる。十分に優れた性能を持つ強化学習エージェントをコントローラとして 1 つだけ開発しておけば、あとはロボットの目的に応じて報酬の与え方だけを設計者が設定するだけで、あらゆる種類のロボット制御方法を同一のコントローラによって自動的に獲得できる。

3-1-3-2　　新たな解の発見

試行錯誤を通じて学習するため、人間のエキスパートが得た解よりも優れた解を発見する可能性がある。特に不確実性や計測が困難な未知パラメータが多い場合、人間の常識では対処しきれないことが予想され、強化学習の効果が期待できる。

3-1-3-3　　想定外の環境変化への対応

機械故障など急激な変化やプラントの経年変化のような緩慢な変化など、予め事態を想定してプログラミングしておくことが困難な環境の変化に対しても自動的に追従することが期待できる。特に宇宙や海底など、通信が物理的に困難な場合や、通信ネットワークの制御のように現象のダイナミクスが人間にとって速すぎる場合において、強化学習の自律的な適応能力が特に威力を発揮する。

3-1-4 強化学習アルゴリズム

強化学習の学習アルゴリズムとしていくつかの手法が提案されているが、本研究では連続的な行動空間を扱える代表的な方法である Actor-Critic 法を採用した。これは状態の価値

を評価する Critic と呼ばれる部分と、状態観測に応じて確率的に行動選択を行う Actor という 2 つの要素より構成される。以下に一般的な Actor-Critic のアルゴリズムの構成と、このアルゴリズムの流れを示す。

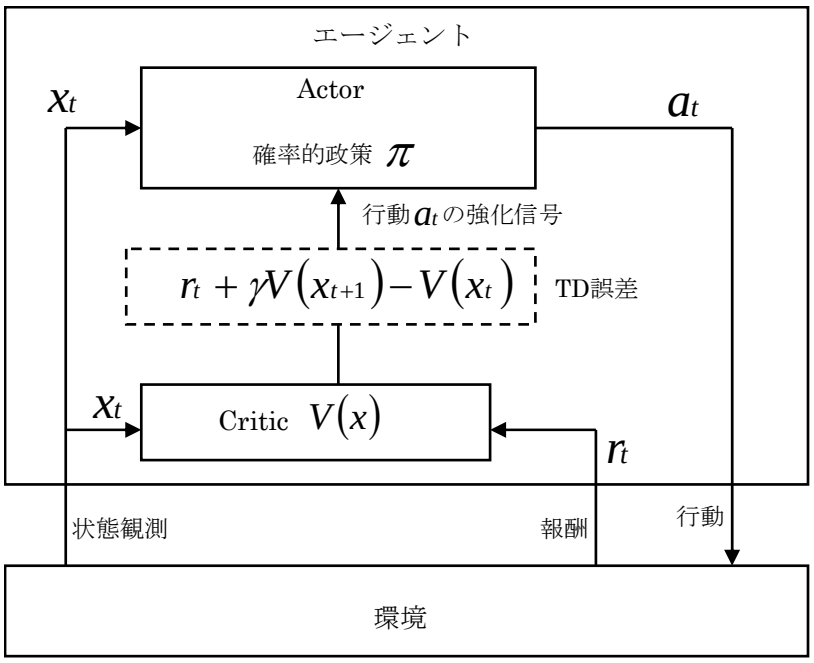


図 3　Actor-Critic アルゴリズムの構成

1.　エージェントは環境において状態 x_t を観測する。
Actor は、確率的政策 π に従って行動 a_t を実行する。

2.　Critic は報酬 r_t を受け取り、次の状態 x_{t+1} を観測し、Actor への強化信号として以下の TD 誤差を計算する。

$$TD\text{誤差} = \left[r_t + \gamma V(x_{t+1})\right] - V(x_t) \qquad \text{Eq. 1}$$

$\gamma(0 \le \gamma \le 1)$は割引率、$V(x)$は Critic が推定した割引報酬の期待値を表す。

3.　TD 誤差を用いて Actor の行動選択確率を更新する。
TD誤差 > 0 ならば、実行した行動 a は比較的好ましいものと考えられるので、この選択確率を増やす。
逆に、TD誤差 < 0 ならば、実行した行動 a は比較的好ましくないものと考えられるので、この選択確率を減らす。

4.　TD 学習を用いて Critic の value の推定値を更新する。

$$V(x) \leftarrow V(x) + \alpha(TD\text{誤差}) \qquad \text{Eq. 2}$$

α を学習率と言う。

5. ステップ 1. から繰り返す。

以上に示した通り、行動を選択した結果、よい状態へ遷移した場合に選択した行動を強化し、逆に間違った行動を選択した場合には行動選択確率を減らす。このことを繰り返し行うことにより、学習していくのが Actor-Critic の学習アルゴリズムである。

3-2　　モデリング

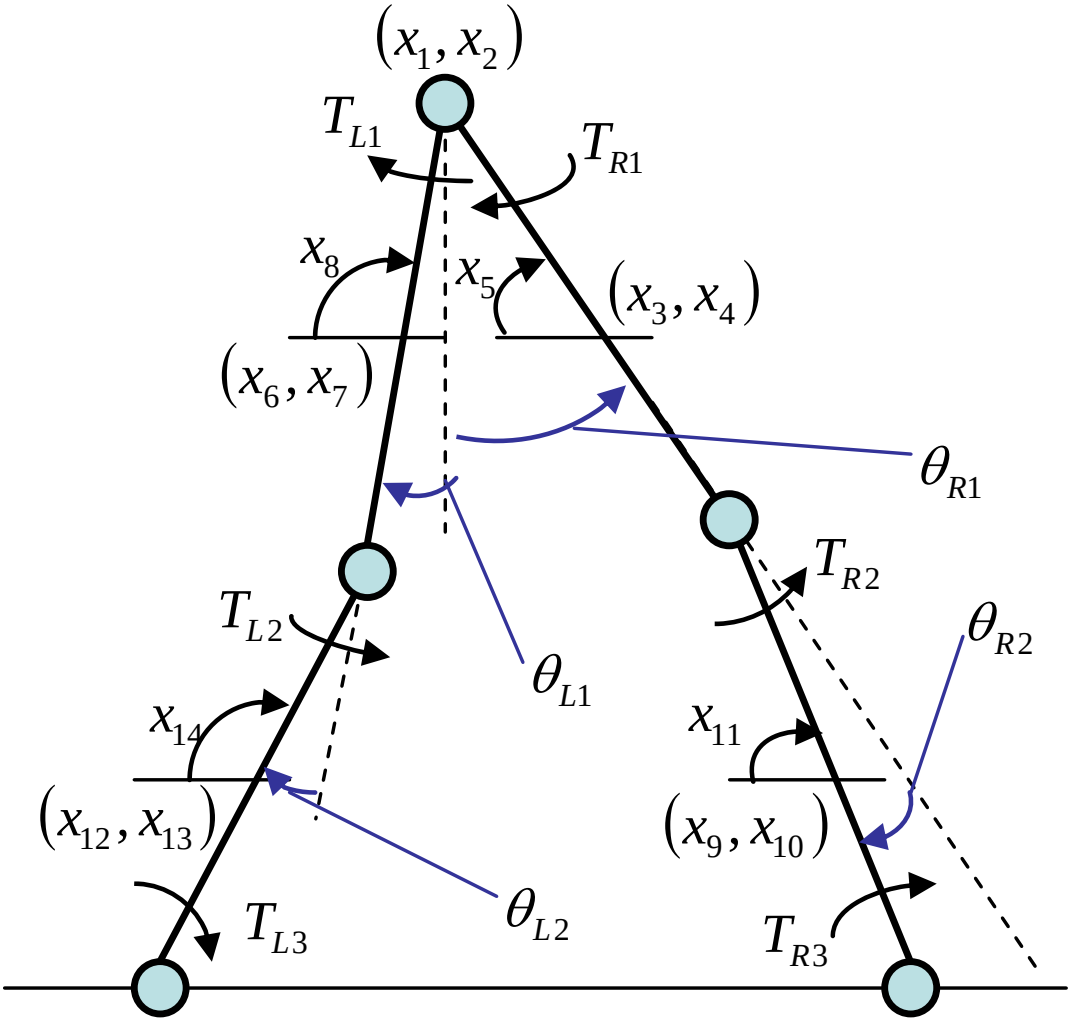


図 4　二足歩行モデル

本研究では二足歩行モデルとして文献[1]で用いられている 2 次元平面上の運動方程式モデルを用いる。

$$\ddot{x} = P(x)F + Q(x, \dot{x}, T, F_g(x, \dot{x})) \qquad \text{Eq. 3}$$

x は腰の位置(x_1, x_2)、両足の腿の位置と角度、両足のすねの位置と角度を並べた 14 次元ベクトルである（図 4）。$P(x)$は（14×8）の行列、F は腰、腿、すね間の関節に働く力を表す 8 次元ベクトル、Q は 14 次元ベクトル、T は腰、膝、くるぶしへの反力を表す 4 次元ベクトルである、今、行列$P(x)$とベクトルQは系の運動方程式を書き下すことで決まる量、トルクTは制御則によって決まる量であり、床からの反力F_gはばねーダンパ系で近似すると、Eq. 3 で未知の量は腰、腿、すね間に働く力Fである。この力Fは以下に従って消去さ

れる。まず、腿やすねの長さによって変数 $x_1 \cdots x_4$ に成り立つ関係式を 8 つ書き下すと、制約条件

$$C(x)\ddot{x} = D(x,\dot{x}) \qquad \text{Eq. 4}$$

が得られる、$C(x)$ は（8×14）の行列であり、$D(x,\dot{x})$ は 8 次元ベクトルである。Eq. 3 を Eq. 4 に代入して整理すると未知の力 F が

$$F = [C(x)P(x)]^{-1}[D(x,\dot{x}) - C(x)Q(x,\dot{x},T,F_g(x,\dot{x}))] \qquad \text{Eq. 5}$$

と決まるので、これを Eq. 3 に代入すると、

$$\ddot{x} = P(x)[C(x)P(x)]^{-1}[D(x,\dot{x}) - C(x)Q(x,\dot{x},T,F_g(x,\dot{x}))] + Q(x,\dot{x},T,F_g(x,\dot{x})) \qquad \text{Eq. 6}$$

が得られる。

これがこの二足歩行モデルをシミュレートするための運動方程式となる。なお、元の 14 変数の運動方程式 Eq. 3 に対し、8 個の制約条件 Eq. 4 を付加しているから、Eq. 6 は 14－8＝6 個の変数に対する運動方程式と等価である。この 6 変数は 図 4 のように、腰の位置 (x,y)、腰の角度 $(\theta_{R1},\theta_{L1})$、膝の角度 $(\theta_{R2},\theta_{L2})$ で考えると便利であり、以後これらの変数で運動を記述する。

さらに、各関節へのトルク（図 4）はPD制御により決定されるとする。ただし、今回は簡単のためにくるぶしのトルクはゼロとする。すなわち、右足に対しては目標角 $\theta^d{}_{R1}, \theta^d{}_{R2}$ を用いて

$$-T_{R1} = K_P(\theta^d{}_{R1} - \theta_{R1}) - K_d\dot{\theta}_{R1} \qquad \text{Eq. 7}$$

$$-T_{R2} = K_P(\theta^d{}_{R2} - \theta_{R2}) - K_d\dot{\theta}_{R2} \qquad \text{Eq. 8}$$

$$T_{R3} = 0 \qquad \text{Eq. 9}$$

であるとし、左足についても同様に定める。制御パラメータ K_p と K_d は今回 $K_p = 100$、$K_d = 2.0$ と定めた。

3-3　　CPG

ここまでのモデルでは単純に目標角に向かってトルクをかけモデルが立っているだけである。ここで問題となってくるのがどのようにトルクを変化させ、歩行をしていくかということである。仮にここまでのモデルでランダムにトルクを変化させ、学習によって最適な歩行を獲得できるようにした場合、ヒトのような歩行を獲得するまでにかかる歩行パターンがあまりにも多いため、計算にかなりの時間を要することが予想される。

そこで、本研究ではヒトの歩行に近づけるという観点、さらにはある程度歩行の方向性を決めるという目的から CPG という手法を取り入れ、この CPG を学習によって変化させ歩行を獲得するように目指すこととした。

3-3-1　　CPGとは

生物の基本的な運動である、歩行、遊泳、飛行等の周期的な運動は、生物の体内に存在する中枢パターン生成器(CPG：Central Pattern Generator)と呼ばれる神経回路によって制御されていると言われている。これは神経回路内で見られるニューロン間の興奮・抑制メカニズムによって作られる神経振動子と考えられる。

本研究ではこの CPG から生成された信号を腰のサーボモータへ入力し、予め歩行のような周期的な運動ができるようにした。そして、学習によって CPG を歩行に最適なものに調整し、歩行を実現することを目指した。

3-3-2　　CPGから二足歩行モデルへの入力

3-2 で定義した二足歩行モデルに対し、図 4 の角度$(\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2})$への目標角$(\theta^d{}_{R1},\theta^d{}_{L1},\theta^d{}_{R2},\theta^d{}_{L2})$を与えてこの二足歩行モデルの歩行制御を行うことを考える。

さらに、目標角の時系列としては CPG からの振動を用いることとする。本研究では微分方程式によって CPG が生成されるものと考える。

最終的に実際のロボットへの適用を考えており、そのため実際のロボットでの CPG 生成には PIC を用いるが（後述)、この PIC の特性として組み込みの数学関数が利用できないという問題がある。そこで乗算・除算のみの簡単な計算で実現され、初期値に無関係で一定の振動をするリミットサイクル振動子である van der Pol 方程式を利用することにする。

van der Pol（ファン・デル・ポール）方程式は次式で表される。

$$\ddot{x}-\varepsilon\left(1-x^2\right)\dot{x}+x=0 \qquad \text{Eq. 10}$$

ただしεは$\varepsilon>0$を満たす定数である。xが小さいとき Eq. 10 は負のダンピング$-\varepsilon\dot{x}$を持つバネ＝ダンパ系となるのでxが増大するが、xが大きくなると非線形項$\varepsilon x^2\dot{x}$が有効になるためxの増大は抑えられ、結果としてxの運動はある領域に閉じ込められることになる。このxの運動はリミットサイクルと呼ばれる周期軌道に収束することが知られている。

系の運動はεが小さいときと大きいときとで場合分けして解析されるが、ここではεが小さく$\varepsilon<<1$が満たされる場合について考える。

ここで、

$$y = \varepsilon\left(x - \frac{x^3}{3} \right) - \dot{x} \qquad \text{Eq. 11}$$

を満たす変数 y を導入する。これを用いると、Eq. 10 は次の 1 階 2 変数の微分方程式に書き直すことができる。

$$\begin{aligned} \dot{x} &= \varepsilon\left(x - \frac{1}{3}x^3 \right) - y \\ \dot{y} &= x \end{aligned} \qquad \text{Eq. 12}$$

$\varepsilon = 0.1$、初期条件 $x = 0.5$、$y = 0$ に対して Eq. 12 を数値的に解いて表示したのが 図 5、図 6 である。図 5 は時刻 t を変化させたときの $(x(t), y(t))$ の軌跡を描いたものであり、ほぼ半径 2 のリミットサイクルに収束していることが分かる。また、t に対する $x(t)$ の変化を表したものが 図 6 であり、$x(t)$ が安定な振動状態へ収束することが分かる。

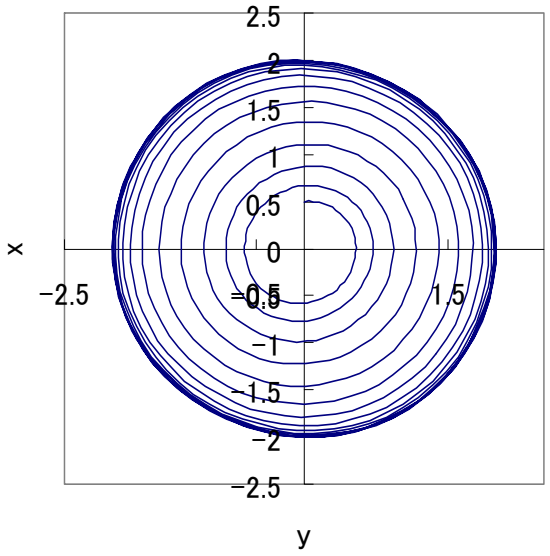


図 5　van der Pol 方程式の (x, y) の軌跡

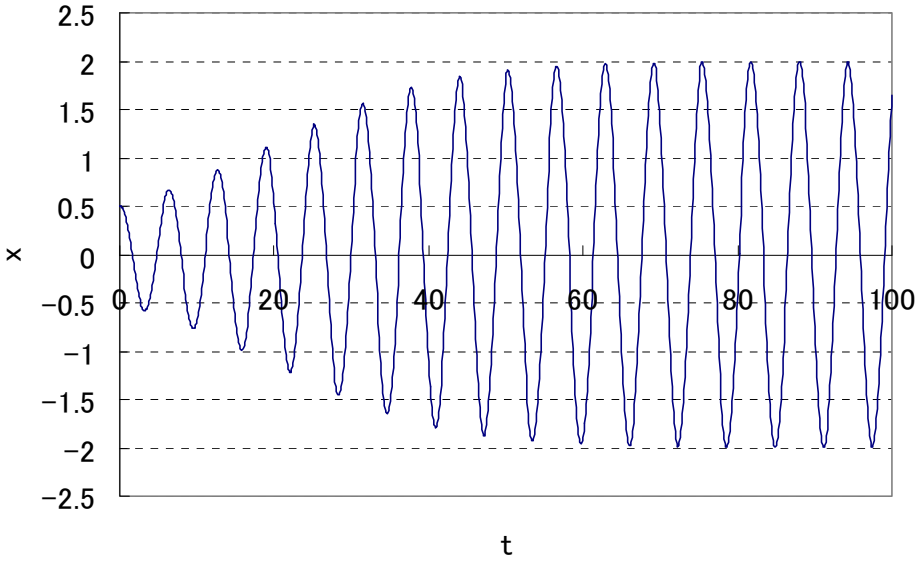


図 6　van der Pol 方程式の $x(t)$ の時間変化

今、van der Pol 方程式の出力を二足歩行モデルへの入力とする場合、両足用に二つの van der Pol 方程式が必要となる。さらに、両足の運動は位相が逆でなければならないから、van der Pol 方程式の出力も逆位相でなければならない。そのような出力を生む例として、以下の結合 van der Pol 方程式系を考える。

$$\dot{x}_1 = \varepsilon_1\left(x_1 - \frac{1}{3}x^3{}_1\right) - y_1 - g(x_2 - x_1) \qquad \text{Eq. 13}$$

$$\dot{y}_1 = x_1 \qquad \text{Eq. 14}$$

$$\dot{x}_2 = \varepsilon_2\left(x_2 - \frac{1}{3}x^3{}_2\right) - y_2 - g(x_1 - x_2) \qquad \text{Eq. 15}$$

$$\dot{y}_2 = x_2 \qquad \text{Eq. 16}$$

Eq. 13、Eq. 15 の末尾の項が逆位相の振動を生むための結合項である。図 7 が 2 つのvan der Pol方程式の結合系の時系列であり、逆位相の振動が得られていることが分かる。

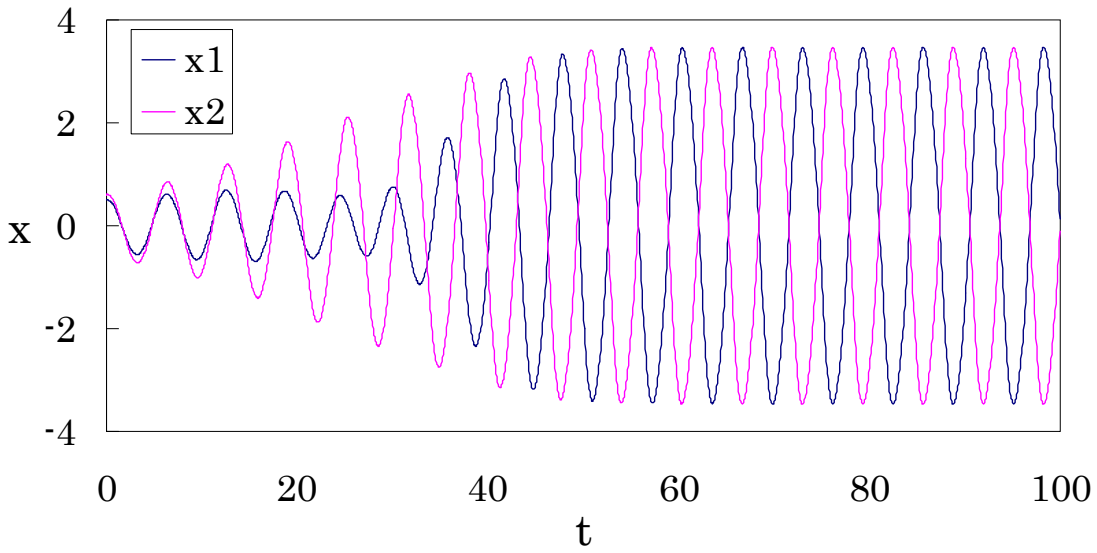


図 7　2 つの van der Pol 方程式の結合系の時系列（$\varepsilon_1 = \varepsilon_2 = 0.1$、$g = 0.1$、初期条件 $x_1 = 0.5$、$y_1 = 0$、$x_2 = 0.6$、$y_2 = 0$）のグラフ

ここまで取り扱ったvan der Pol方程式をロボットの歩行に適用するためには、その周期をコントロールする必要がある。図 5 や図 6 に見られる振動の周期はおおよそ 5 程度であるが、ロボットの歩行に適用するためには周期 1 程度の振動が必要である。

そこで、Eq. 13、Eq. 14、Eq. 15、Eq. 16 に新たな変数 γ を加えた以下のモデルをロボットの歩行に適用する。

$$\dot{x}_1 = \gamma\left(\varepsilon_1\left(x_1 - \frac{1}{3}x^3{}_1\right) - y_1 - g(x_2 - x_1)\right) \qquad \text{Eq. 17}$$

$$\dot{y}_1 = \gamma x_1 \qquad \text{Eq. 18}$$

$$\dot{x}_2 = \gamma\left(\varepsilon_2\left(x_2 - \frac{1}{3}x^3{}_2\right) - y_2 - g(x_1 - x_2)\right) \qquad \text{Eq. 19}$$

$$\dot{y}_2 = \gamma x_2 \qquad \text{Eq. 20}$$

γ は振動の速さを調整するためのパラメータで、大きい程速い振動が得られる。

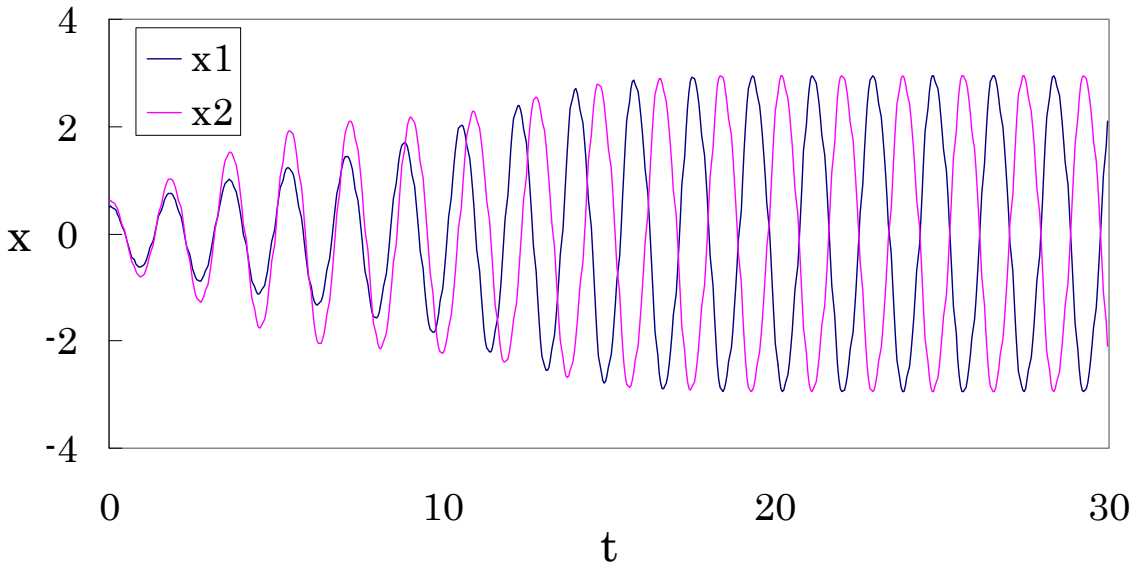


図 8　修正された結合 van der Pol 方程式の結合系の時系列（$\varepsilon_1 = \varepsilon_2 = 0.17$、$g = 0.1$、$\gamma = 3.5$、初期条件 $x_1 = 0.5$、$y_1 = 0$、$x_2 = 0.6$、$y_2 = 0$）のグラフ

以後ロボットの歩行のパラメータとして $\varepsilon_1 = \varepsilon_2 = 0.17$、$g = 0.1$、$\gamma = 3.5$ を用いる。$\gamma = 3.5$ であるので、3.5 倍の速さの振動になる。（図 8）

3-4　　強化学習によるCPG調整

図 4 のように目標角を定めることで二足歩行モデルの歩行を目指すわけではあるが、実際にモデルが歩行に成功するためにはCPGのパラメータを調整することで図 8の波形を最適なものに変化させる必要がある。しかし、試行錯誤によってパラメータを調整することは難しく、なんらかの指針が必要である。そこで、本研究では文献[3]で提案されているように強化学習により、CPGを調整し、最適な歩行の獲得を目指す。さらに、現実の二足歩行ロボットへの適用を目指してアルゴリズムの簡略化も行う。

まず、本研究における二足歩行モデルの制御の模式図を 図 9 に示す。このうち、CPGからPDサーボを経て二足歩行モデルへのトルクを得ることは **3-2** と **3-3-2** とで解説した。本章では残りの部分について説明する。

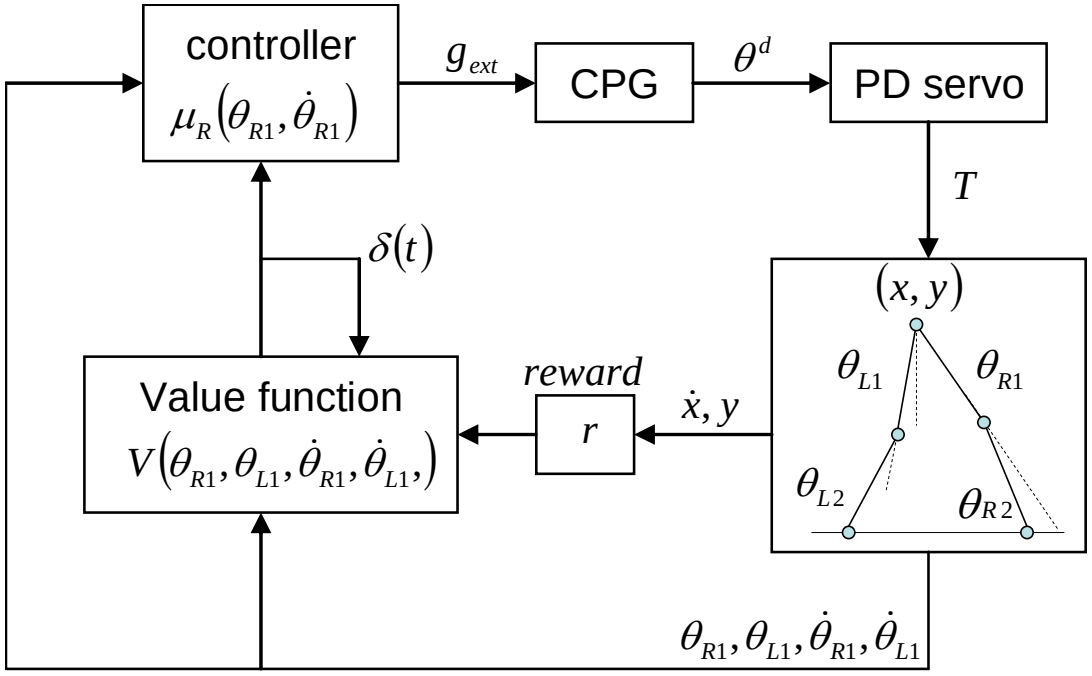


図 9　強化学習による制御の模式図

まず、モデルの 6 つの変数(x,y)、$(\theta_{R1},\theta_{L1})$、$(\theta_{R2},\theta_{L2})$のうち、腰関節の角度$(\theta_{R1},\theta_{L1})$、およびその時間微分$(\dot{\theta}_{R1},\dot{\theta}_{L1})$のみを観測するものとする、

次に、強化学習の特徴でもあり、最も重要な二足歩行モデルの現在の状態に対する報酬関数$r(x)$を定義する。本研究では、

「腰の高さを一定に保ちつつ、速さ v_d の水平移動」

と報酬を決定し、この条件を満たす場合に最も報酬が高いとして、以下のように報酬関数を定める。

$$r(x) = r_h(x) - r_v(x) \qquad \text{Eq. 21}$$

$$r_r(x) = \begin{cases} y + 0.1 & if \;\; y < 0.9 \\ 1 & otherwise \end{cases} \qquad \text{Eq. 22}$$

$$r_v(x) = \frac{1}{v_d^2}(\dot{x} - v_d)^2 \qquad \text{Eq. 23}$$

この報酬関数が最大となるように制御則を決めれば良い。本研究では$v_d = 0.8$とする。

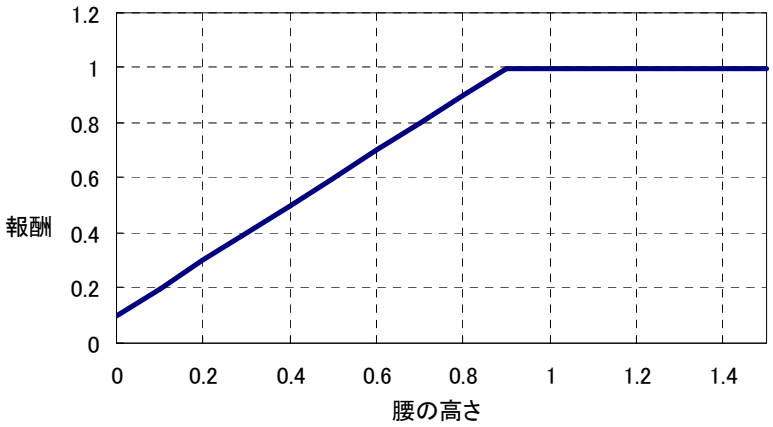


図 10　腰の高さの報酬

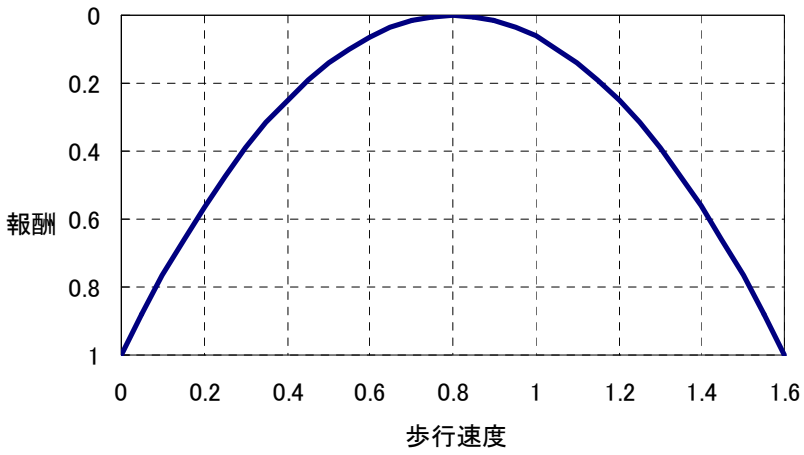


図 11　歩行速度の報酬

今、CPG への制御入力をベクトル関数$u(t)$表す、本研究では、CPG の両足に対応する 2 つのモジュールに対する制御入力

$$u(t)=\begin{pmatrix}u_R(t)\\u_L(t)\end{pmatrix} \quad \text{Eq. 24}$$

とする。今、Eq. 21 の報酬関数は時刻tでのモデルの状態$x(t)$だけではなく制御入力$u(t)$にも依存するから$r(x(t),u(t))$と書ける。制御入力を決める確率的制御則をπとし、この制御則の元の状態$x(t)$の価値関数をπに関する期待値として

$$V(x(t))=E\left\{\int_t^{\infty} e^{-\frac{s-t}{\gamma}} r(x(s),u(s))ds|\pi\right\} \quad \text{Eq. 25}$$

で表す。γは価値関数の時定数である。これはこの制御則に従い続けたときに獲得できる報酬の累積値を予測する関数になっており、この価値関数$V(x(t))$を最大化することで目的の運動が達成される。ただし、$V(x(t))$は未知の関数であり、$V(x(t))$は何らかの方法で推測しなければならない。

さらに確率的制御則の制御入力$u(t)$の成分を

$$u_R(t)=\mu_R(x(t))+\sigma_R\xi_R(t) \quad \text{Eq. 26}$$

$$u_L(t)=\mu_L(x(t))+\sigma_L\xi_L(t) \quad \text{Eq. 27}$$

と定める。ただし、$\xi_R(t),\xi_L(t)$は標準正規分布に従う乱数とする。さらに、これから CPG の結合強度を

$$g_{ext}^{R}=\frac{g_{\min}+g_{\max}}{2}+\frac{g_{\min}-g_{\max}}{2}g(u_R(t)) \quad \text{Eq. 28}$$

$$g_{ext}^{L}=\frac{g_{\min}+g_{\max}}{2}+\frac{g_{\min}-g_{\max}}{2}g(u_L(t)) \quad \text{Eq. 29}$$

$$g(x)=\frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right) \quad \text{Eq. 30}$$

と決める。$g(x)$は[-1, 1]の範囲の値を取るから、結果的にg^{R}_{ext}と$g^{L}_{ext}$$[g_{\min},g_{\max}]$の範囲の値を取ることになる。Eq. 26、Eq. 27 において、両足への入力の期待値となる$\mu_R(x(t))$と$\mu_L(x(t))$を決定して二足歩行モデルへの適切な制御入力とすることが目標となる。

以上をまとめると、系の状態 $x=\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$ を観測し、その状態に関する価値関数 $V\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$ および両足への制御則 $\mu_R\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$ $\mu_L\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$ を推測することが課題となる。これは 3 つの 4 変数関数を学習によって関数近似することを意味するが、その方法として文献[3]では方策こう配法を用いた学習法が提案されている。以下にその手法を示す。

価値関数 V を推定するにはＴＤ誤差

$$\delta(t)=r(t)-\frac{1}{\tau}V+\dot{V} \qquad \text{Eq. 31}$$

を 0 にする方向に V の学習を進める。これは、Eq. 25 を時間微分したもので、価値関数に対する拘束条件である。ここで、$V(x(t))=V(x(t);w^c)$ を価値関数の予測とする。ただし、w^c は価値関数の予測パラメータである。予測が正しければ、価値関数は

$$\frac{dV}{dt}=\frac{1}{\tau}V-r(t) \qquad \text{Eq. 32}$$

を満たす。逆に予測が正しくない場合は Eq. 31 に示したＴＤ誤差を減らすように価値関数の学習を行うものである。そのため価値関数の表現方法として正規化ガウス関数ネットワークを用いる。ここで、正規化ガウス関数ネットワークの定義をする。

今、$x=c_k$ を中心としたガウス関数は

$$\phi_k(x)=c^{-\frac{1}{2}\left|M_k(x-c_k)^2\right|} \qquad \text{Eq. 33}$$

と書くことができる。行列 M_k は本研究では $M_k=diag(m_i)$ とし、m_i は $\phi_k(x)$ の半径の逆数とする。この関数を点の総和が 1 となるように正規化したものを正規化ガウス関数の基底

$$b_k(x)=\frac{\phi_k(x)}{\sum_{l=1}^{N}\phi_k(x)} \qquad \text{Eq. 34}$$

として定義する。

このことから価値関数は、

$$V(x)=\sum_i w_i^c b_i^c(x) \qquad \text{Eq. 35}$$

と近似することができる。ただし、b_i^c は基底関数であり、w_i^c は価値関数のパラメータである。パラメータ w_i^c に対する eligibility trace e_i^c とＴＤ誤差を用いたパラメータ w_i^c の更新式

は以下のようになる。

$$\dot{e}_i^c = -\frac{1}{k_c} e_i^c + b_i^c(x) \quad \text{Eq. 36}$$

$$\dot{w}_i^c = \alpha \delta(t) e_i^c \quad \text{Eq. 37}$$

ただし、α は価値関数の学習率、k_c は eligibility trace の時定数である。

一方、制御則に関しては μ_j を正規化ガウス関数ネットワークによって、標準偏差 σ_j をシグモイド関数によって以下のように定義する。

$$\mu_j(x) = \sum_i w_{ij}^\mu b_i^\mu(x) \quad \text{Eq. 38}$$

$$\sigma_j = \frac{1}{1+\exp(-w_j^\sigma)} \quad \text{Eq. 39}$$

ただし、b_i^μ は基底関数であり、w_{ij}^μ と w_j^σ はフィードバック制御器のパラメータである。このパラメータは、

$$\dot{e}_{ij}^\mu = -\frac{1}{k_\mu} e_{ij}^\mu + (u_j - \mu_j) b_i^\mu(x) \quad \text{Eq. 40}$$

$$\dot{e}_j^\sigma = -\frac{1}{k_\sigma} e_j^\sigma + \left((u_j - \mu_j)^2 - \sigma_j^2\right)(1-\sigma_j) \quad \text{Eq. 41}$$

$$\dot{w}_{ij}^\mu = \beta^\mu \delta(t) e_{ij}^\mu \quad \text{Eq. 42}$$

$$\dot{w}_j^\sigma = \beta^\sigma \delta(t) e_j^\sigma \quad \text{Eq. 43}$$

$$u_j = \frac{2}{g_{\max} - g_{\min}} \left(g_{ext}^j - \frac{g_{\min} + g_{\max}}{2} \right) \quad \text{Eq. 44}$$

で定める。ただし、β^μ, β^σ は学習率であり、k_μ, k_σ は eligibility trace の時定数である。

以上において、パラメータは $\tau = 1$、$\alpha = 10$、$k_c = 0.1$、$\beta^\mu = 20$、$\beta^\sigma = 20$、$k_\mu = 1$、$k_\sigma = 0.1$ とした。

関数 V、μ_R、μ_L の推定には正規化ガウス関数ネットワークを用いているが、これは $\theta_{R1}, \theta_{L1}, \dot{\theta}_{R1}, \dot{\theta}_{L1}$ の各軸に基底関数を配置して関数近似を行う手法であり、例えば各軸の基底関数を N 個とすると、N^4 個のオーダーのパラメータの微分方程式を数値的に解く必要があ

り、非常に計算時間がかかる。そのため、この手法を現実のロボットの制御に適用するためには手法の簡略化が必要となる。

＜参考＞正規化ガウス関数ネットワークの定義

今、$x=c_k$ を中心としたガウス関数は

$$\phi_k(x)=e^{-\frac{1}{2}\|M_k(x-c_k)\|^2}$$　　c_k：ガウス関数の中心　　M_k：行列　　k：基底の番数

と書くことができる。本研究では行列 $M_k=diag(m_i)$ とし、m_i は $\phi_k(x)$ の半径の逆数とする。

この関数を点の総和が 1 となるように正規化したものを正規化ガウス関数の基底

$$b_k(x)=\frac{\phi_k(x)}{\sum_{l=1}^{N}\phi_l(x)}$$

と定義する。

Ex.　簡単のため 2 次元で考え、$\vec{x}=(x,y)$、$\vec{c}_k=(c_x,c_y)$ とする。

$$M_k=\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \quad \vec{x}-\vec{c}_k=\begin{pmatrix} x-c_x \\ y-c_y \end{pmatrix}$$

$$M_k(\vec{x}-\vec{c}_k)=\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}\begin{pmatrix} x-c_k \\ y-c_y \end{pmatrix}=\begin{pmatrix} m_1(x-c_x) \\ m_2(y-c_y) \end{pmatrix}$$

$$\|M_k(\vec{x}-\vec{c}_k)\|^2=m_1{}^2(x-c_x)^2+m_2{}^2(y-c_y)^2$$

m_i は $\phi_k(x)$ の半径の逆数としたので

$$m_1=\frac{1}{a} \qquad m_2=\frac{1}{b}$$

となり代入すると

$$\|M_k(\vec{x}-\vec{c}_k)\|^2=\frac{(x-c_x)^2}{a^2}+\frac{(y-c_y)^2}{b^2}$$

となる。よって 1 つのガウス関数は（図 12 の黒線）

$$\phi_k(x)=e^{-\frac{1}{2}\left(\frac{(x-c_x)^2}{a^2}+\frac{(y-c_y)^2}{b^2}\right)}$$

となり、すべてのガウス関数の総和は

$$\sum_{k=1}^{N}b_k(x)=\sum_{k=1}^{N}\phi_k(x)$$・・・Radial Basis Function（RBF）

となる。（図 13 の青線）

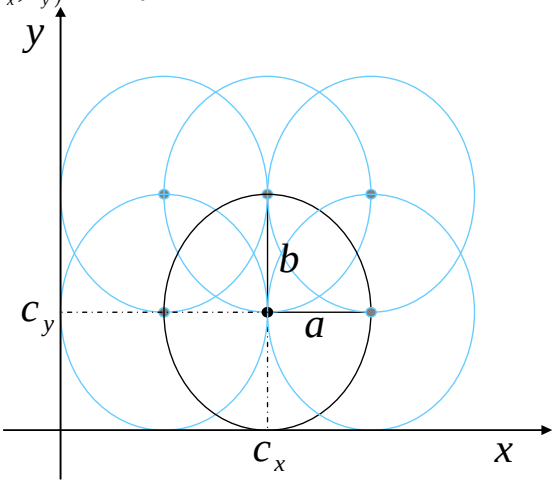


図 12　ガウス関数配置

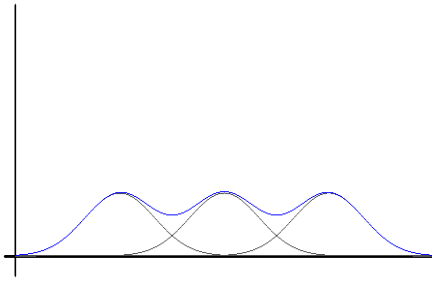


図 13　ガウス関数総和

しかしここで配置したガウス関数の両端は 0 に収束してしまい、両端で 0 でない値をとる関数を表現できないという問題がある。そこで

$$b_k(x)=\frac{\phi_k(x)}{\sum_{l=1}^{N}\phi_l(x)}$$ ・・・正規化ガウス関数ネットワーク

を定義する。するとこの総和は

$$\sum_{k=1}^{N}b_k(x)=\sum_{k=1}^{N}\left(\frac{\phi_k(x)}{\sum_{l=1}^{N}\phi_l(x)}\right)=1$$

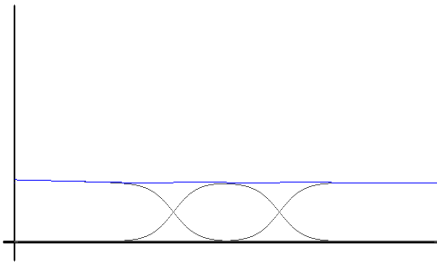


図 14　正規化したガウス関数総和

となることがわかる。これは$\sum_{l=1}^{N}\phi_l(x)$で割ることによって配置したガウス関数の総和がすべて 1 になっていることを示している。以下は基底 10 個（$k=10$）としたときの初期状態と、できた正規化ガウス関数ネットワークの各基底に適当な値を掛け合わせてできた関数である。

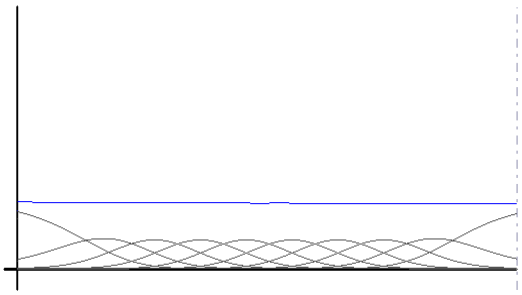


図 15　正規化ガウス関数ネットワーク（基底 10）

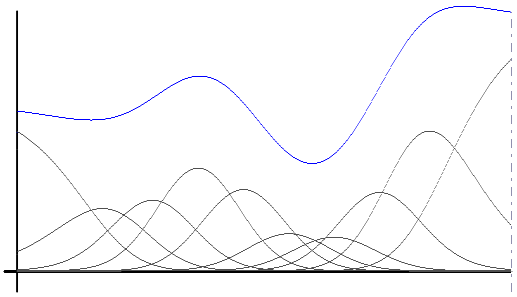


図 16　基底をランダムに増減させてできた関数

3-5　制御則の簡略化

ここで、まず制御則 μ_R と μ_L の簡略化を行う。理想的に学習が進むと、右足と左足の制御則は等しくなると考えられる。すなわち、

$$\mu_R\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)=\mu_L\left(\theta_{L1},\theta_{R1},\dot{\theta}_{L1},\dot{\theta}_{R1}\right) \qquad \text{Eq. 45}$$

なる対象性が満たされる。そこで、右足に対する制御則 μ_R のみを用いて両足の制御を行うことができることがわかる。

さらに、本研究では CPG からの逆位相の振動によって両足を制御しているため、右足の状態 θ_{R1}、$\dot{\theta}_{R1}$ を知ることで左足の状態 θ_{L1}、$\dot{\theta}_{L1}$ の状態も推定可能であると考えられる。そのため、制御則 μ_R の変数を θ_{R1}、$\dot{\theta}_{R1}$ のみとし、$\mu_R\left(\theta_{R1},\dot{\theta}_{R1}\right)$ と書く。左足の制御には同じ制御則 μ_R を用いて $\mu_R\left(\theta_{L1},\dot{\theta}_{L1}\right)$ を用いる。関数 μ_R の変数が 4 から 2 に減っているため、この関数の推定に用いる基底の数は N^4 から N^2 と減り、大幅な簡略化となっている。

以上をまとめると、二足歩行モデルの状態の評価には $V\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$ を用いるため両足からの情報が必要であるが、各足の制御には $\mu_R\left(\theta_{R1},\dot{\theta}_{R1}\right)$ または $\mu_R\left(\theta_{L1},\dot{\theta}_{L1}\right)$ を用いるための対象となる足の情報のみを用いるという描象が成り立つ。

3-6　　評価関数Vの簡略化

前章では制御則 μ_R と μ_L を 2 変数関数に簡略化した結果を示したが、評価関数は $V\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)$のように 4 変数のままであった。強化学習のアルゴリズムにおいて最も計算時間がかかるのは評価関数や制御則の計算であり、その次元（変数の数）が大きい程計算時間は指数的に増大する。そのため、計算の高速化には次元を減らすのが有効である。

そこで本章では評価関数V を 2 変数に簡略化する方法を示す。

まず、評価関数に対して変数変換を行い、

$$\begin{aligned} &V\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right) \\ &=V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},\frac{\theta_{R1}-\theta_{L1}}{2},\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},\frac{\dot{\theta}_{R1}-\dot{\theta}_{L1}}{2}\right) \end{aligned} \qquad \text{Eq. 46}$$

のように、平均角度$\left(\theta_{R1}+\theta_{L1}\right)/2$、角度差$\left(\theta_{R1}-\theta_{L1}\right)/2$、平均角速度$\left(\dot{\theta}_{R1}+\dot{\theta}_{L1}\right)/2$、角速度差$\left(\dot{\theta}_{R1}-\dot{\theta}_{L1}\right)/2$、の関数$V_2$として書き直す。

今、評価関数は左右の足について対象であるから、左右の足の角度を入れ替えしても同じ値を持つ、すなわち、

$$V\left(\theta_{R1},\theta_{L1},\dot{\theta}_{R1},\dot{\theta}_{L1}\right)=V\left(\theta_{L1},\theta_{R1},\dot{\theta}_{L1},\dot{\theta}_{R1}\right) \qquad \text{Eq. 47}$$

の関係が成り立つ。この関係はV_2では

$$\begin{aligned} &V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},-\frac{\theta_{R1}-\theta_{L1}}{2},\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},-\frac{\dot{\theta}_{R1}-\dot{\theta}_{L1}}{2}\right) \\ &=V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},\frac{\theta_{R1}-\theta_{L1}}{2},\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},\frac{\dot{\theta}_{R1}-\dot{\theta}_{L1}}{2}\right) \end{aligned} \qquad \text{Eq. 48}$$

と表される。これにより、V_2を角度差$\phi=\left(\theta_{R1}-\theta_{L1}\right)/2$と角速度差$\omega=\left(\dot{\theta}_{R1}-\dot{\theta}_{L1}\right)/2$についてマクローリン展開すると

$$\begin{aligned} &V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},\frac{\theta_{R1}-\theta_{L1}}{2},\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},\frac{\dot{\theta}_{R1}-\dot{\theta}_{L1}}{2}\right) \\ &=V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},0,\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},0\right)+\sum_{m=1}^{\infty}\sum_{k=0}^{2m}\frac{1}{k!(2m-k)!}\frac{\partial^{2m}V_2}{\partial\theta^k\partial^{2m-k}}\theta^k\omega^{2m-k} \end{aligned} \qquad \text{Eq. 49}$$

と偶数次の項のみとなる。今回は簡単のため Eq. 49 右辺第 2 項以降を無視し、

$$V_{red}\left(\frac{\theta_{R1}+\theta_{L1}}{2},\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2}\right)\equiv V_2\left(\frac{\theta_{R1}+\theta_{L1}}{2},0,\frac{\dot{\theta}_{R1}+\dot{\theta}_{L1}}{2},0\right) \qquad \text{Eq. 50}$$

を評価関数として用いる。

4 ソフトウェア（学習シミュレータ）

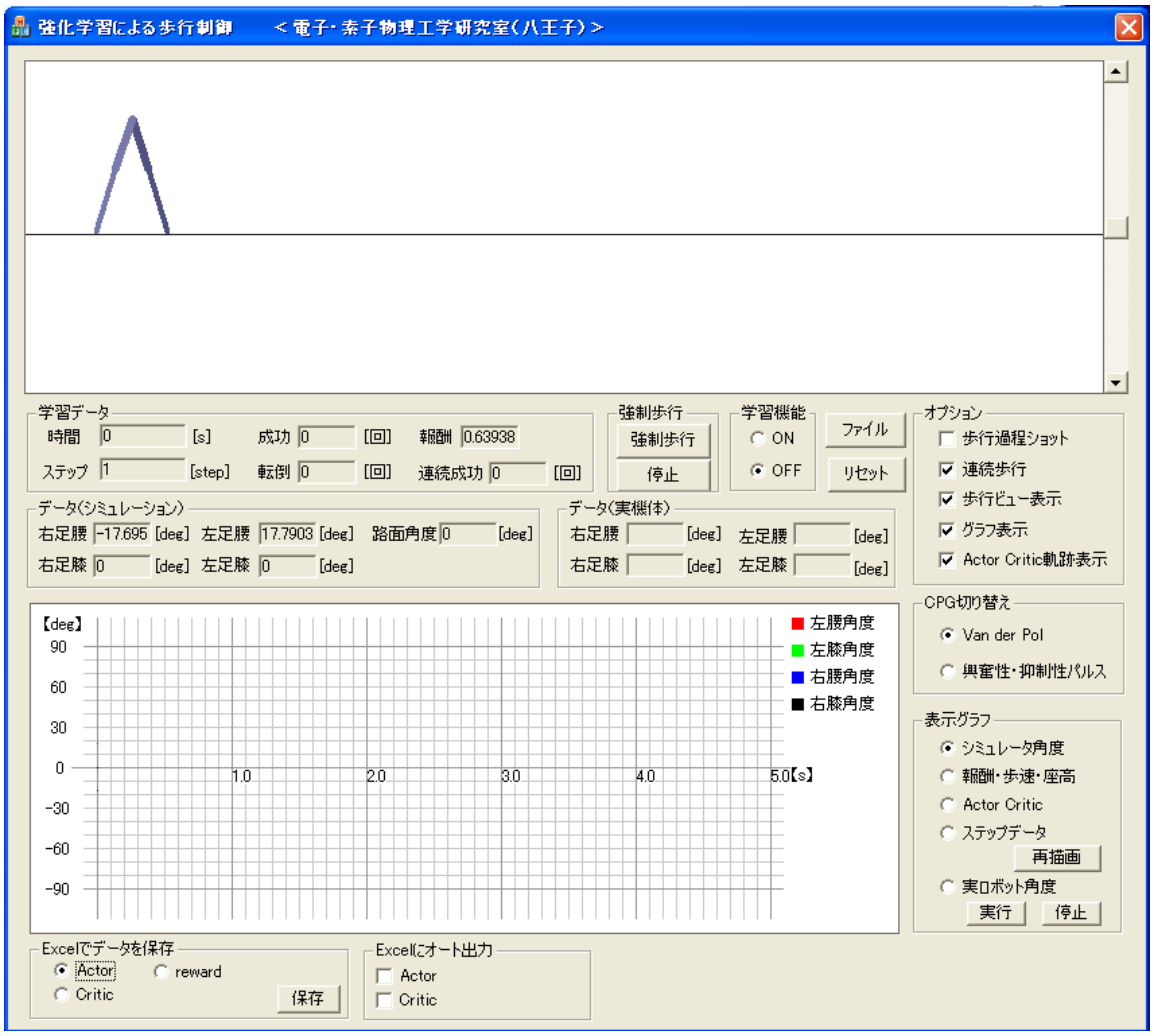


図 17　作成した学習シミュレータ

本研究では、強化学習を用いて歩行獲得するロボットを作成するためのソフトウェア環境として、学習シミュレータをVisual Studio C++で作成した。これはどのような学習条件で歩行を獲得できるのかを予め把握する必要があること、さらには学習後の状態を視覚化できる理由から作成した。図 17 は実際に作成した学習シミュレータのインターフェイスである。

この章では 3 章の理論を元に作成した学習シミュレータの概要を説明していき、モデルでの学習実験を行った結果を示していく。

4-1　　ソフトウェア概要

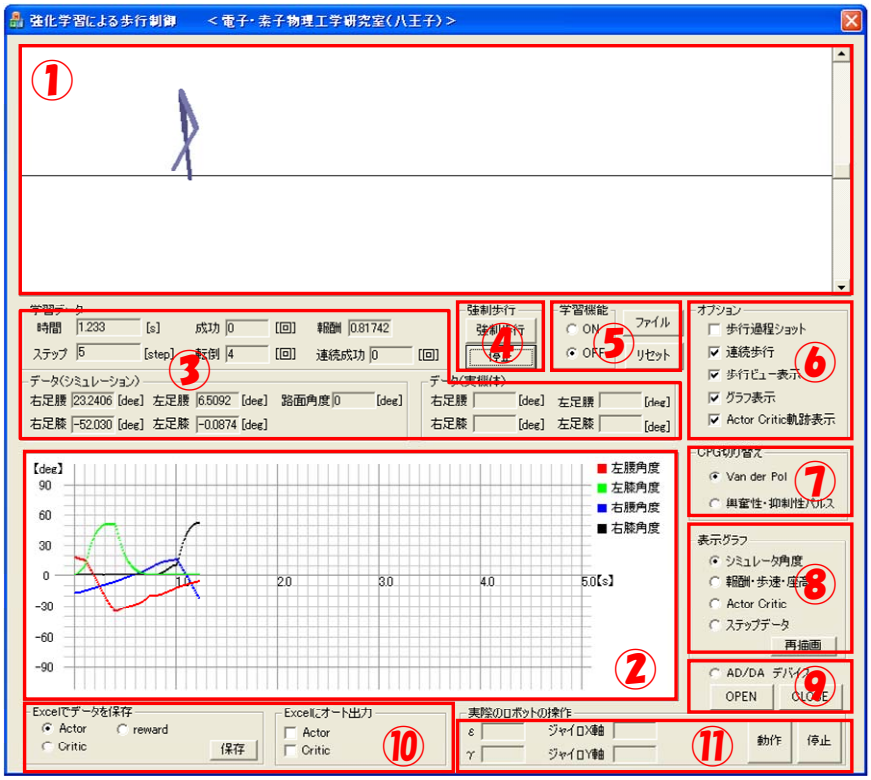


図 18　学習シミュレータ機能番号

本研究で作成したソフトウェアの機能は以下のようなものである。

I.　　モデルの計算（図 18 中の①）
II.　　モデルの視覚化（図 18 中の①⑥）
III.　坂道機能（図 18 中の①の右端スクロールバーで任意角度に指定）
IV.　データ表示機能（図 18 中の③）
V.　　グラフ表示機能（図 18 中の②）
VI.　データ保存機能（図 18 中の⑩）
VII.　学習機能（図 18 中の⑤）
VIII.　学習データ読み込み機能（図 18 中の⑤）
IX.　CPG切り替え機能（図 18 中の⑦）
X.　　制御回路への信号出力機能（図 18 中の⑪、**5-8-2** 参照）
XI.　　制御回路からのデータ受信機能（図 18 中の⑪、**5-8-1** 参照）

ここでこれらの機能について簡単に説明していく。

まずⅠ., Ⅱ.のモデルの計算・視覚化である。ここでは、モデルは **3-2** で説明した二足歩行モデルを用いて 4 次のルンゲクッタ法で数値的に解き、その結果の座標をソフトウェア

上のピクチャーボックスに表示させるという手法をとっている。図 18 中の①がモデルの状態を表示している領域である。モデルの動作は、アニメーション機能を用いてモデルの計算結果をすぐに表示させている。この際に用いたVisual Studio上の関数がOnTimer関数で、任意の時間周期で他の関数を呼び出すときに用いる関数である。作成したプログラムでは実時間で 25msec進むたびに表示を切り替えるようにしてある。ここで問題となるのがプログラムの高速化である。アニメーション機能において一番の問題となるのがこのプログラムの高速化であり、モデルを動かして表示するためには、次のモデルの状態を表示するたびに前のモデルの表示を消す必要がある。実際にこのプログラムでは 1 秒間に 40 コマ表示しているため、再描画回数削減の工夫が必要となった。そこでリージョン機能を各所に入れプログラムの高速化を図った。このリージョン機能というのは指定した場所のみの表示を切り替えるものであり、作成したプログラムでは表示切り替え範囲をリージョン機能を使わないものと比べて描画量が 10 分の 1 にまで削減することができ、大幅なプログラムの高速化に成功した。

本研究で作成した作成したグラフィック系のプログラムではモデルを視覚するためのオプション機能として 図 18 の⑥で切り替えることのできる機能がいくつかある。オプションボタンを上から解説していくとまず歩行ショット機能である。この機能はモデルがどのような歩行過程を経て歩行を行っているのかを確認するための機能であり、この機能を「ON」にすることで 0.5[s]ごとの歩行を残して描画していく。以下の 図 19 が歩行ショット機能を用いてモデルに歩行をさせている様子である。

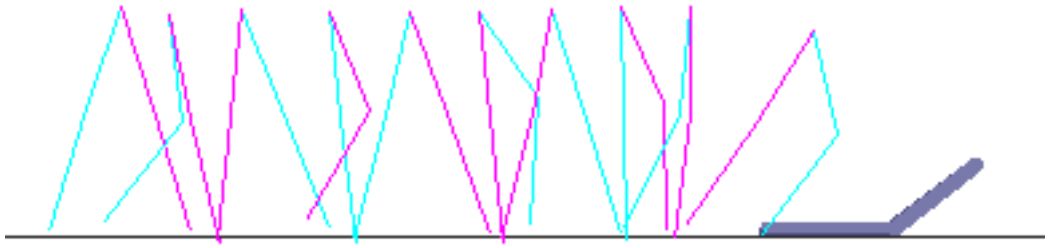


図 19　歩行ショット機能を用いてモデルに歩行をさせている様子

次のオプション機能としては、連続歩行機能でチェックボックスにチェックを入れておくことで、歩行実験行う際など連続して試行を行うことができる。逆にチェックをはずしておくと 5[s]経過後若しくは 図 19 のようにモデルが転倒した時点でプログラムが終了するようになっている。

最後に残りの 3 つのオプション機能をまとめて解説する。残りの 3 つは歩行ビュー・グラフ・グラフの軌跡の表示可否をするものである。上記したがソフトウェアの描画表示は PC に相当の負担をかけているため学習計算など遅延に影響する恐れがある。そこで 3 つのオプションとして学習実験を行っている際の歩行描画表示・グラフ表示・グラフ軌跡表示をそれぞれ選択することにより表示させないようにした。

次にⅢ.の坂道機能である。本研究では将来的な問題として考えられる不整地歩行（坂道）

に対応すべく、図 18 中の①右端にあるスクロールバーを上下させることにより、モデルの歩行スタートラインから 1mのところを起点とした坂道を作り出すことができる。本研究では坂道に対する学習実験を現在のところ行っていないがモデル上の坂道歩行の様子は以下の 図 20 のようになる。

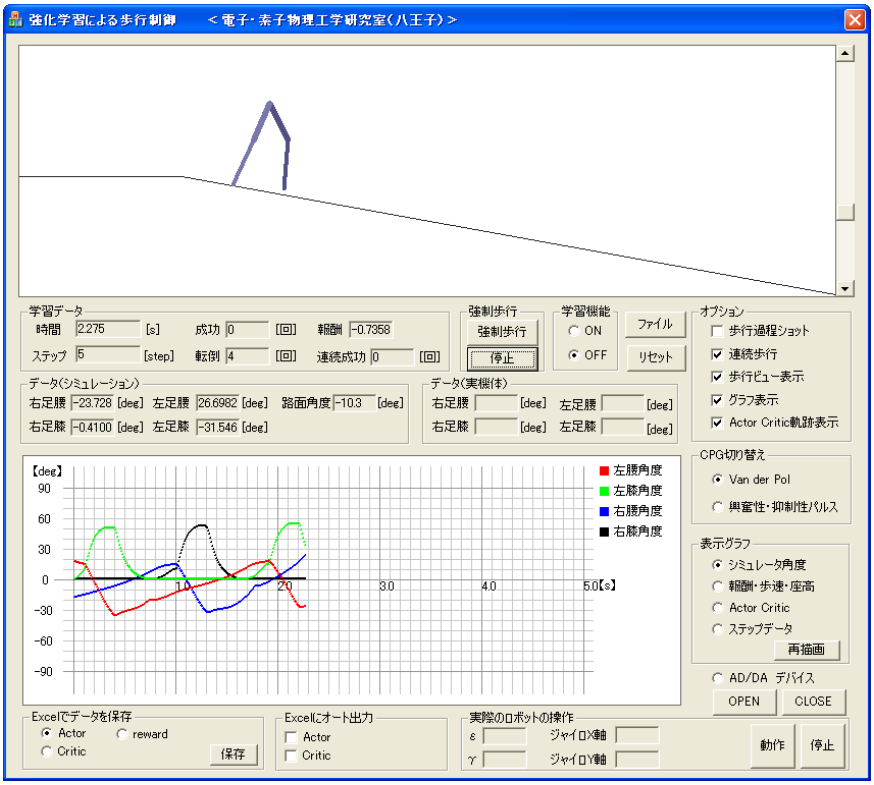


図 20　坂道を歩行しているモデル。本研究は坂道での学習実験は行っていない。

次にⅣ., Ⅴ.のデータ・グラフ表示機能である。これは学習した結果の様子を随時把握し、より良い学習条件への変更に利用できることや、現在の状態を数値化・グラフ化して表示させることにより学習にどれだけの試行回数が必要なのか等を確認するためのものである。図 18 の②がグラフが表示される場所で、⑧で表示グラフの選択ができるようになっている。

この学習シミュレータではグラフを 4 種類用意し、常に切り替え可能な状態にプログラムした。表示させることのできるグラフは次の 4 つである。

【モデル角度表示】

これは各関節の角度の時間変化を表示するもので、実際のソフトウェア上でのグラフは図 21 のようになっている。

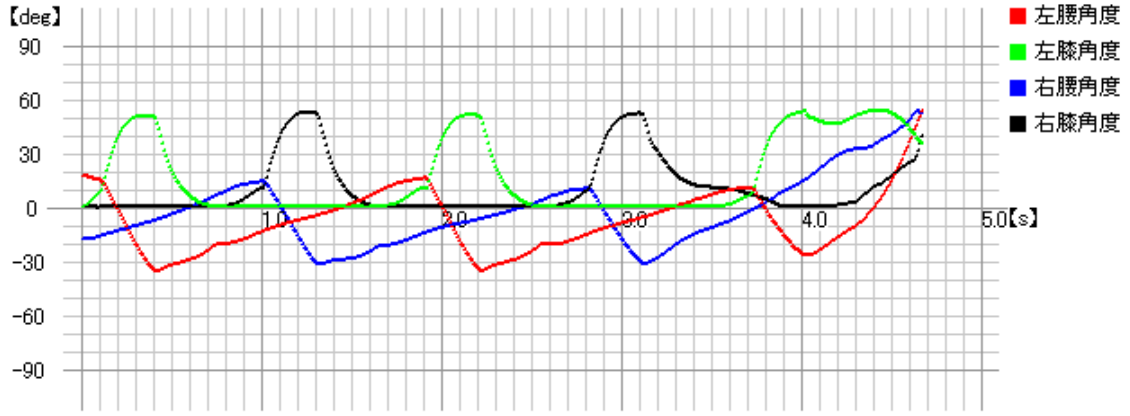


図 21　モデル歩行時の各関節角度グラフ（シミュレーション上）

【報酬・歩行速度・腰の高さを表すグラフ】

3-4 で解説したが、本研究では報酬を『歩行速度・腰の高さを一定に保つ』という基準にしているため、学習実験をする際に単位時間当たりの歩行速度・腰の高さを確認する目的でこれらのグラフ表示を行う。実際のプログラムでのグラフは以下の 図 22 のようになっている。

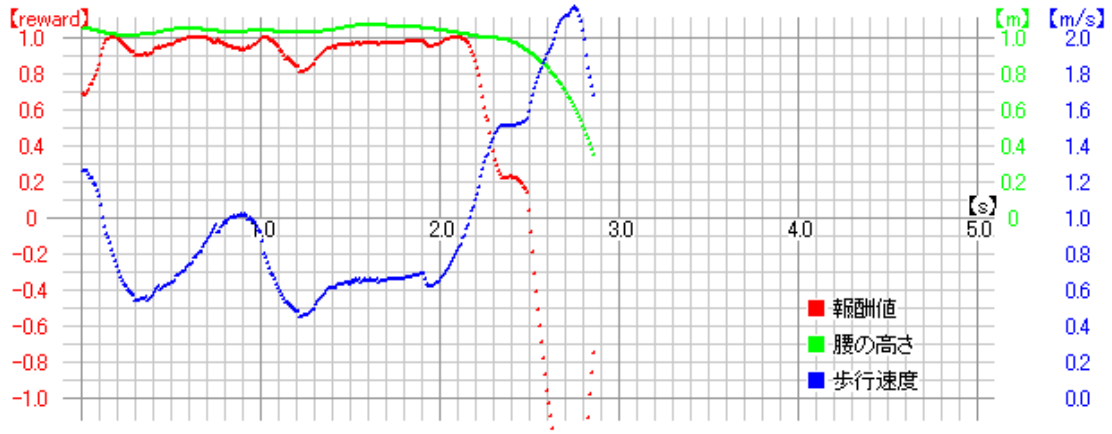


図 22　モデルの腰の高さ・歩行速度・報酬値のグラフ（シミュレーション上）

【Actor-Critic 表示機能】

次も学習に関わるグラフでActor-Criticのカラー表示である。学習実験を行う際、現在どのような学習をしているかを常に把握するため作成した。実際のプログラムでのActor-Criticを表示しているのが以下の 図 23 である。

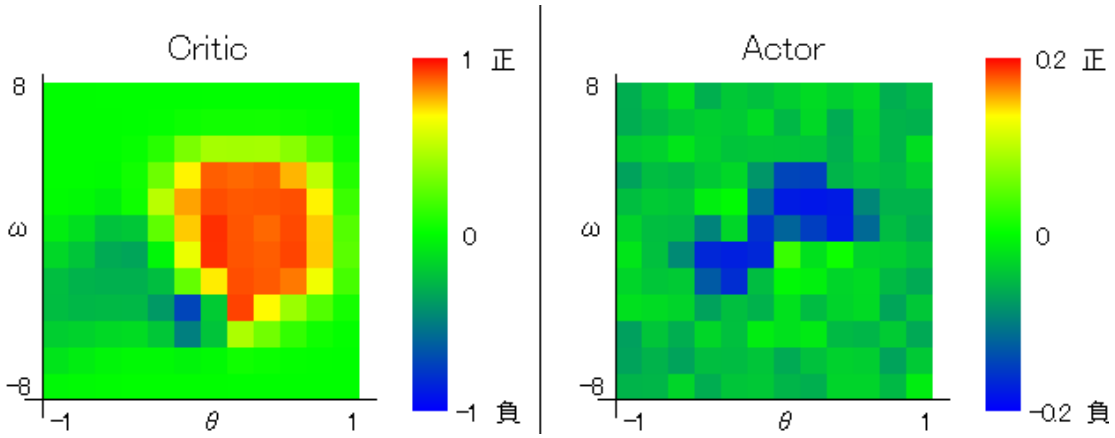


図 23　Actor-Critic カラー表示（シミュレーション上）

実際にはActor-Critic共に３次元グラフを描くことが理想的だが、プログラムの負担が大きいため本研究では色分けによって表示している。さらに現在の状態とActor-Criticを組み合わせて表示しているものが 図 24 である。

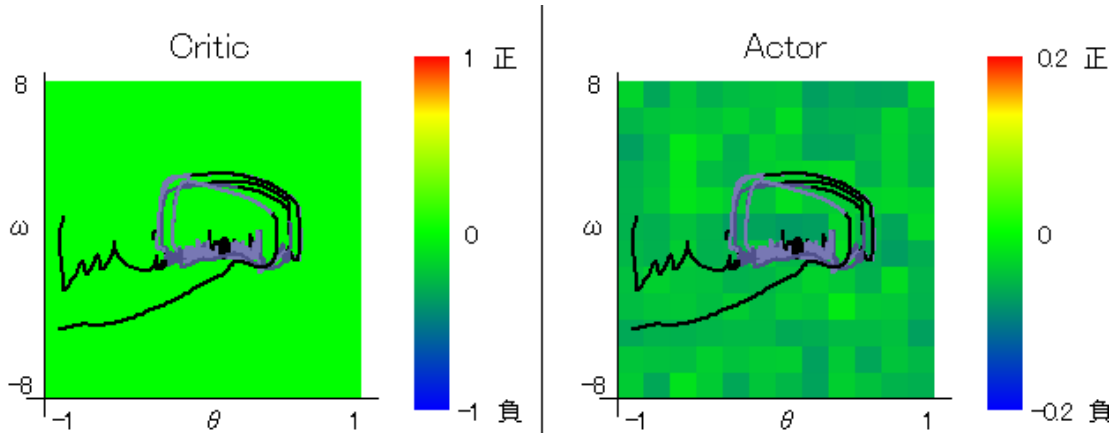


図 24　　Actor-Critic のカラー表示に現在のモデルの軌跡を付けたもの

この軌跡を表示することで、モデルがどのように歩行をしていくのかや、どのように CPG を強めていくのかが視覚的に把握できる。

次章 **4-2** の学習実験の章において改めて詳しく解説する。

【ステップデータ】

これは 1 試行当りの報酬の総和をグラフ化したものと、現在までの歩行確率をグラフ化したもので、おおよその学習までに必要な試行回数が確認できる。実際にプログラムでのグラフは以下の 図 25 のようになっている。

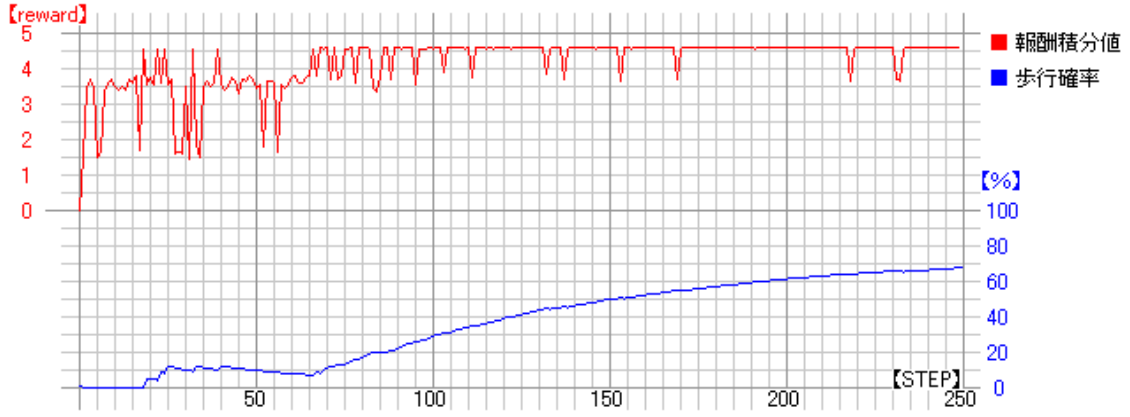


図 25　毎試行ごとの報酬積分値と歩行確率（シミュレーション上）

このグラフに関しても Actor-Critic のグラフと同様にシミュレーション学習実験の際に解説する。

ここまでが作成した学習シミュレータ上で表示可能なグラフである。

次の機能としてⅤ.のデータ保存機能である。これまでグラフ機能について説明してきたが、このグラフ機能にはシミュレーション上での表示機能しかなく、単位時間当たりの詳細なデータはプログラムを終了した時点ですべて失われる。そこで 2 種類のデータ保存機能を作成した。図 18 の⑩である。

1 つ目は現在の状態を必要なときに応じて保存する機能であり、Microsoft Excel の CSV 形式で保存される。

もう 1 つがオート出力機能である。これは試行開始から終了までの Actor-Critic のデータを毎試行保存してくことのできる機能で、学習実験をする場合にチェックボックスにチェックを入れておくだけで全て実行ファイルと同じ階層のフォルダに 1 つ目と同じく Microsoft Excel の CSV 形式で保存されていく。

次に Ⅵ.、Ⅶ.の学習・学習データ読み取り機能である。これはこのシミュレータの核となる機能で、ラジオボタンを「ON」をクリックすることでプログラムの学習機能をONの状態にすることができる。この状態にすることで上記のActor-Criticのグラフも更新していき、歩行も獲得していくものとなっている。さらに学習データを読み取る機能を作成し、学習実験で得た過去のデータを読み込ませることができる。インターフェイス上での場所は図 18 の⑤である。

次にⅧ.のCPG切り替え機能である（図 18 の⑦）。この機能は今後新たなCPGに変更するときに用いるもので、現在は **3-3-2** で解説したvan der Pol方程式によるCPGモデルになっている。

なお、このプログラムでは van der Pol 方程式による CPG モデル以外に文献[6]で提案されているパルスニューロンの興奮性・抑制性パルスニューラルネットワークによる CPG モデルにも切り替えることが可能となっている。この CPG モデルはシミュレーション上では問題なく歩行をすることが可能だが、実際のロボットに適応する場合に PIC での計算が不可能になるため本研究では利用してない。

最後にⅨ.、Ⅹ.の制御回路との入出力機能である（図 18 の⑪）。本研究は実際のロボットでの学習を行うことのできる環境の構築が課題となるため、モデル以外にロボットの制御回路との情報の入出力機能が必要となる。この機能の詳細は制御回路を解説した後に 6 章で解説していく。

以上が本研究で作成した学習シミュレータの機能である。以後この学習シミュレータを用いてモデルの学習実験や、制御回路とのデータ通信を行っていく。

4-2　　シミュレータ学習実験

ここで二足歩行モデルの学習実験を行う。二足歩行モデルの初期状態は以下のようにランダムに定める。

- 前足は確率1/2で右足と左足のいずれかを選ぶ
- 前足の初期角度は［0.095π，0.1π］の範囲の一様分布に従う
- 後足の初期角度は［-0.1π，-0.095π］の範囲に一様分布に従う
- モデルの初期水平速度は［1.25，1.3］の範囲の一様分布に従う

一回の試行は最大 5［s］の間行うとする、5［s］経過する前にモデルが転倒した場合は$r(t)=-1$の罰則を 0.1［s］の間与えてその施行は終了する。その試行が成功したかどうかの指標として、報酬の積分値

$$r_{all}=\int_0^T r(t)dt \qquad \text{Eq. 51}$$

を観測する。ただしTは試行が終了した時刻とし、モデルが途中で転ばなかった場合は$T=5$［s］となる。

評価関数$V_{red}\left((\theta_{R1}+\theta_{L1})/2,(\dot{\theta}_{R1}+\dot{\theta}_{L1})/2\right)$の関数近似は、平均角度$(\theta_{R1}+\theta_{L1})/2$に関しては[-8, 8]の範囲に 17 の基底を配置し、平均角速度$(\dot{\theta}_{R1}+\dot{\theta}_{L1})/2$に対しては[-6，6]の範囲に 17 の基底を配置した。すなわち、$17^2=289$個の基底である。制御則$\mu_R(\theta_{R1},\dot{\theta}_{R1})$の関数近似は、角度$\theta_{R1}$に対しては[-1，1]の範囲に 17 個の基底を配置し、角速度$\dot{\theta}_{R1}$に対しては[-8，8]の範囲に 17 個の基底を配置した。すなわち、$17^2=289$個の基底である。

以上の準備のもと二足歩行シミュレータ学習実験を行った結果が以下の通りである。

まず始めに初期状態の歩行の様子を示したものが 図 26 である。この状態ではただ単純にCPGを入れているだけなので数歩で転倒してしまっている。そこで学習を進めていく。図 27 が試行を 250 回まで進めたもので、図中のグラフでは毎ステップあたりの報酬総和と歩行確率を示している。図 28 は図 27 の報酬総和グラフを取り出したもので、このグラフから 80 試行程度で歩行が獲得できていることが分かる。

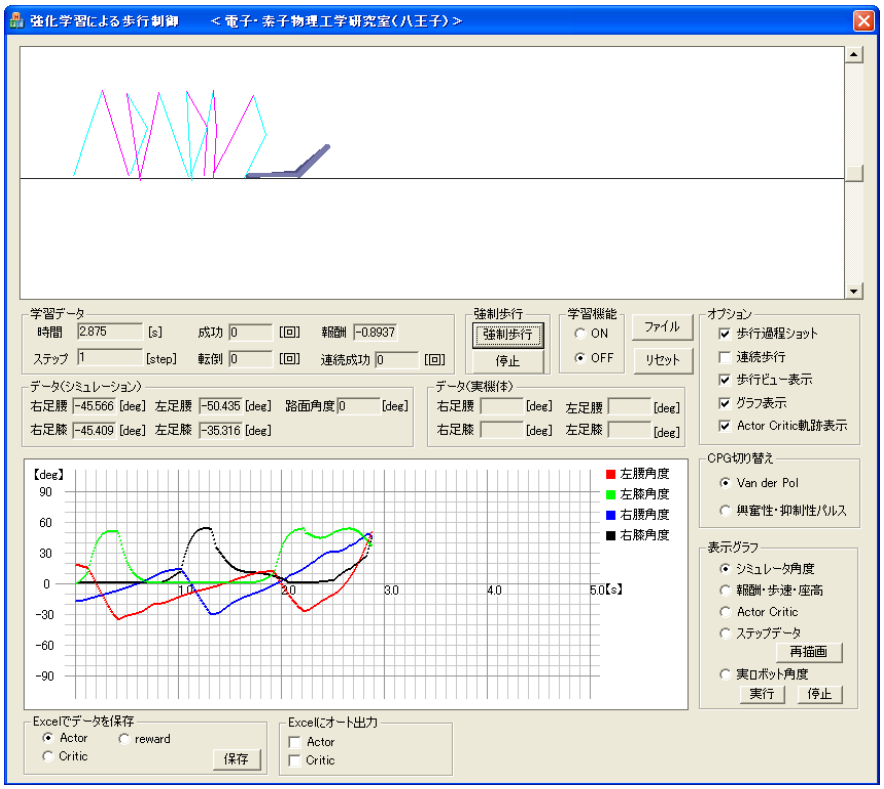


図 26　初期状態での歩行の様子

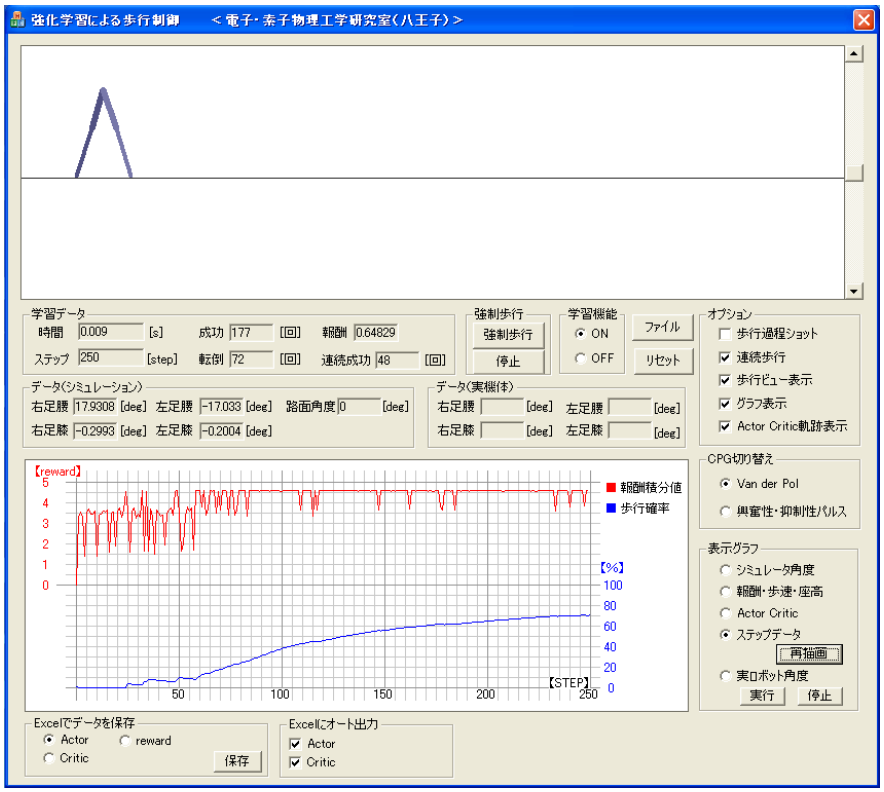


図 27　250 試行後の状態

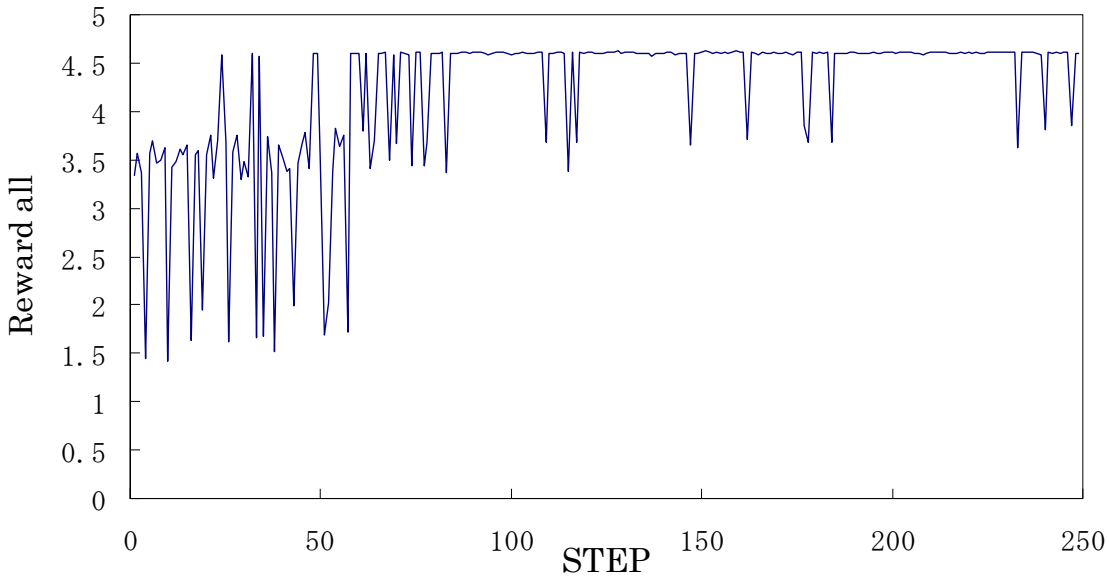


図 28　毎ステップあたりの報酬値の積分

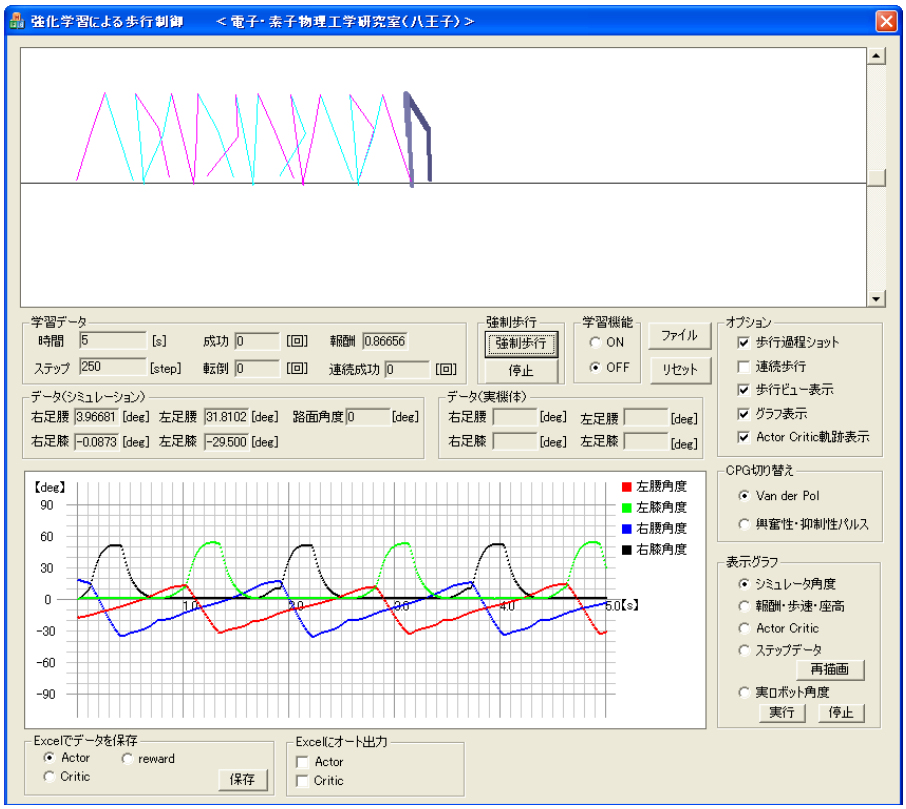


図 29　250 試行後の歩行の様子

図 29 は 250 試行後の歩行の様子を表しているものである。このように歩行が獲得できていることが分かる。また、図中のグラフは歩行時の右腰、左腰、右膝、左膝の角度を示しているもので、歩行初期の状態（図 26）に比べ、最後まで安定していることが分かる。

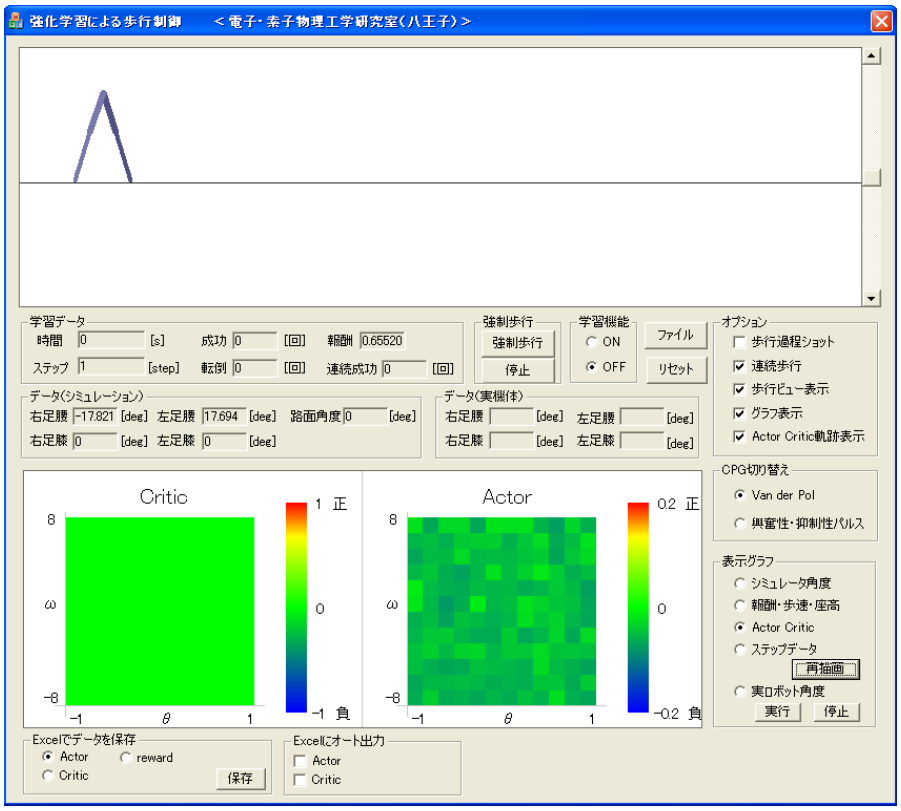


図 30　初期の価値関数と制御則

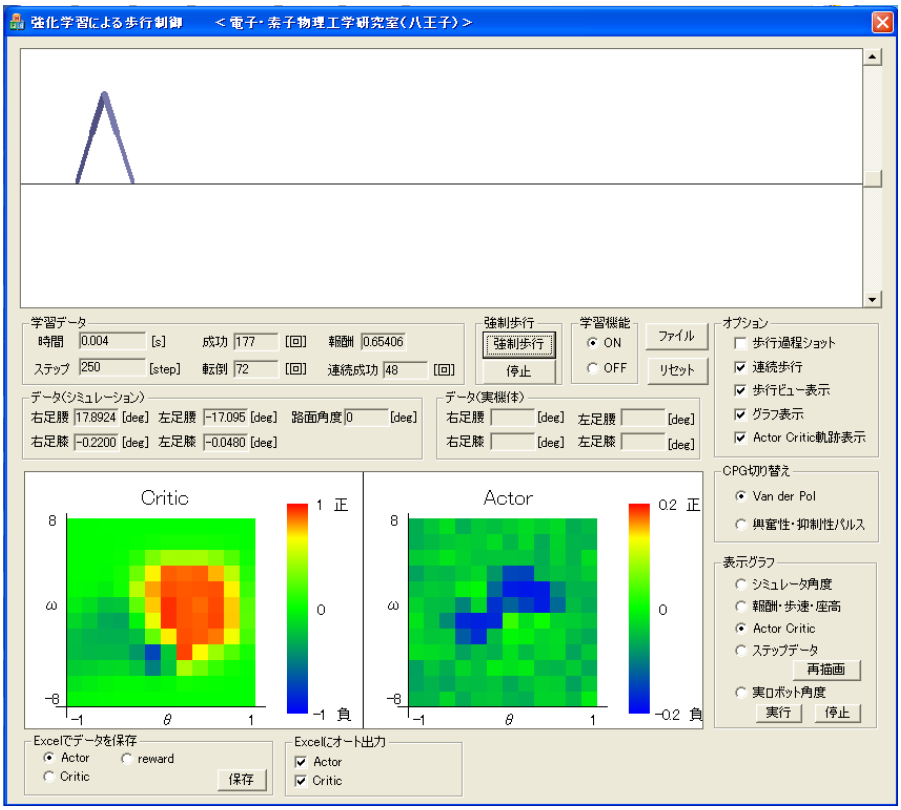
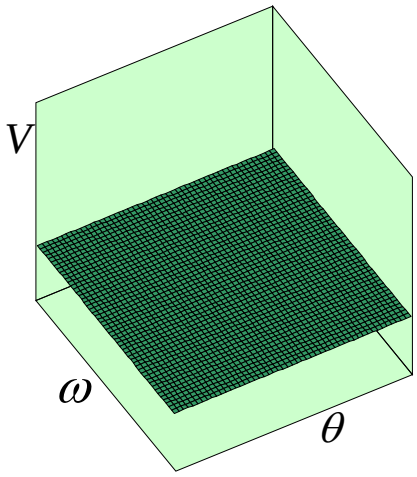


図 31　250 試行後の価値関数と制御則

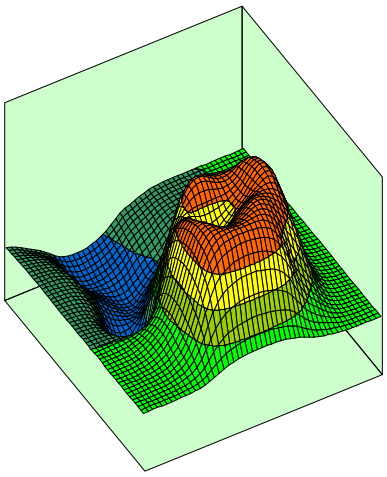
図 30 は初期状態の価値関数v（Critic）と制御則$\mu(t)$（Actor）の状態である。**3-6** までに説明してきた正規化ガウス関数ネットワークで近似している価値関数Vと制御則$\mu(t)$であり、初期状態では価値関数Vは 0 の状態であり、制御則$\mu(t)$はランダムで決定されている。次の 図 31 は 250 試行後の価値関数Vと制御則$\mu(t)$を示しているものであり、学習によって 図 30 から変化したものである。

この学習後の価値関数Vの結果では、歩行に最適な角度と角速度の関係は 図 31 の図中のCriticの赤くなった部分をまわるようにして歩行をしたほうが転倒しにくいということを表しており、そのための行動としてActorでは青くなった部分のときに腰のトルクを強めることを示している。

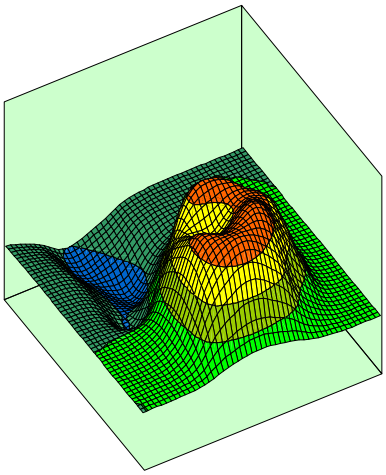
以下に価値関数Vと制御則$\mu(t)$の学習の様子を 10 試行ごとに 3D グラフにしたものを示す。特に変化の大きい 100 試行までの価値関数Vと制御則$\mu(t)$のグラフを示す。



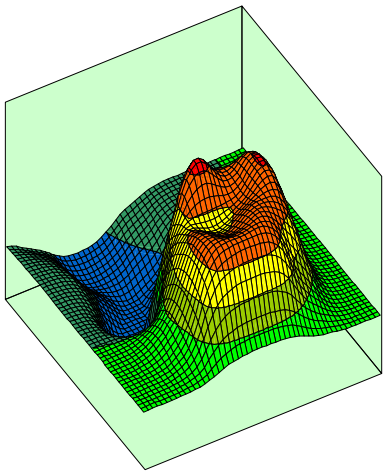
価値観数 V　0 試行時の状態



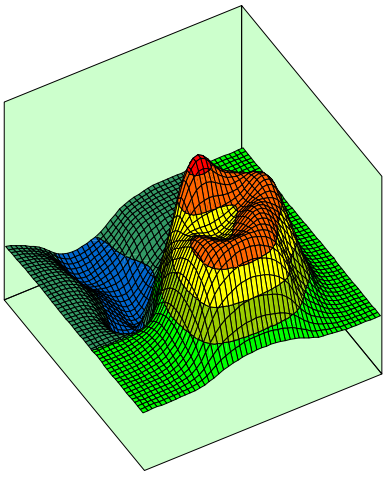
価値観数 V　30 試行時の状態



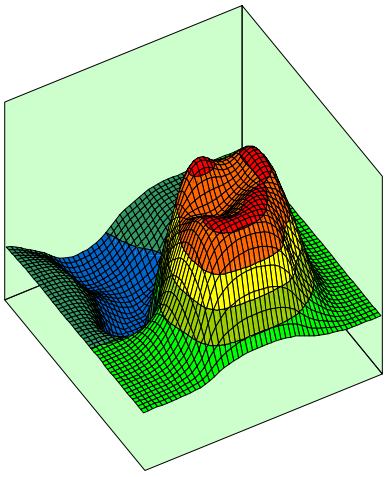
価値観数 V　10 試行時の状態



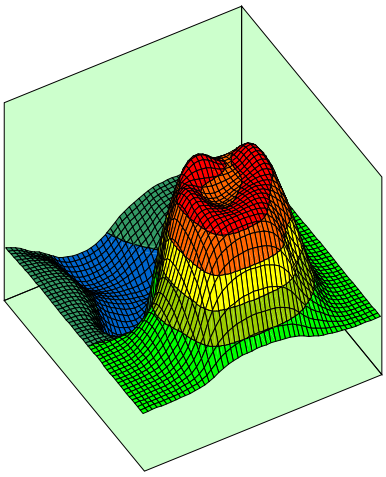
価値観数 V　40 試行時の状態



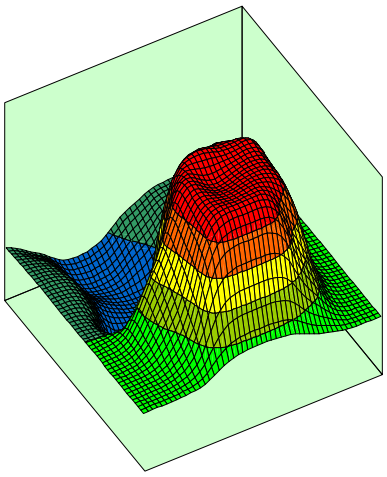
価値観数 V　20 試行時の状態



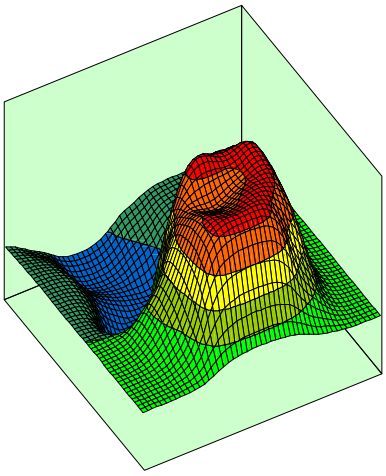
価値観数 V　50 試行時の状態



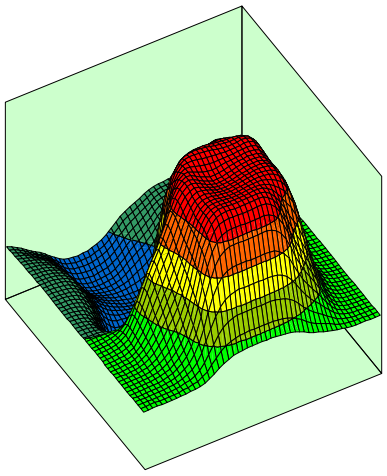
価値観数 V　60 試行時の状態



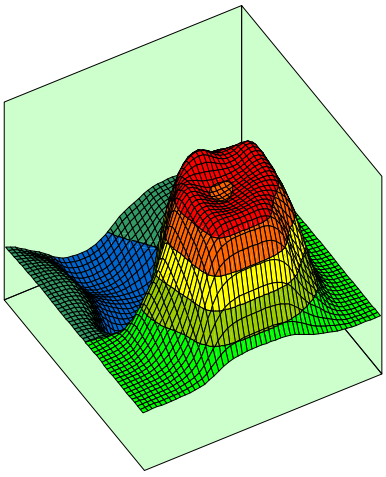
価値観数 V　90 試行時の状態



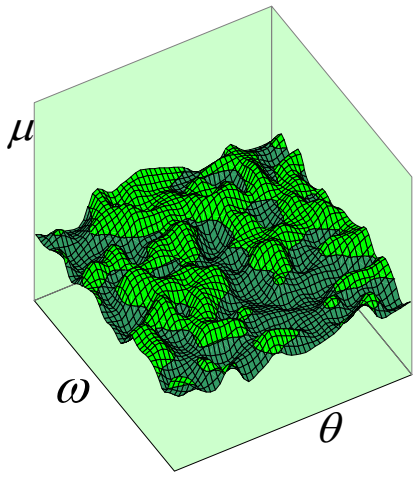
価値観数 V　70 試行時の状態



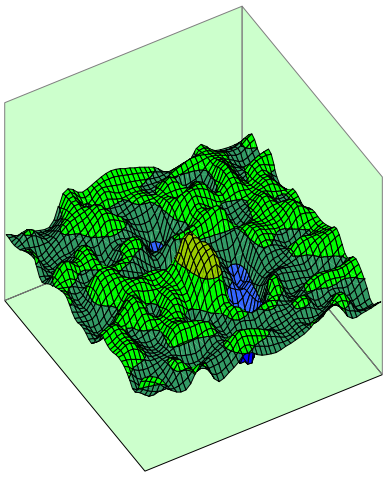
価値観数 V　100 試行時の状態



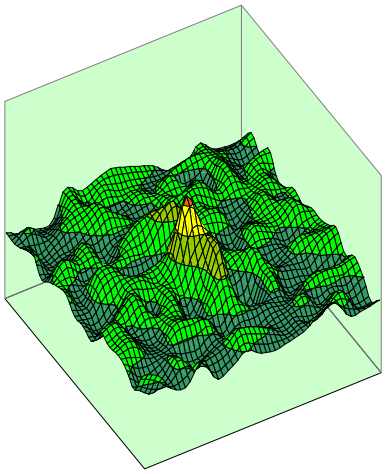
価値観数 V　80 試行時の状態



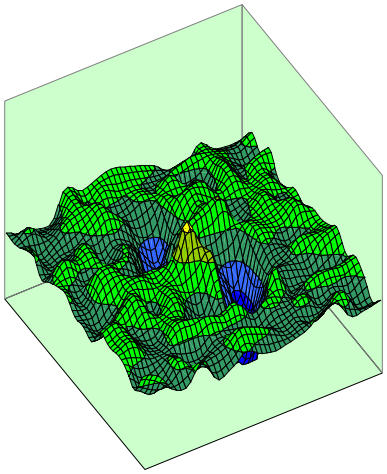
制御則 $u(t)$　0 試行時



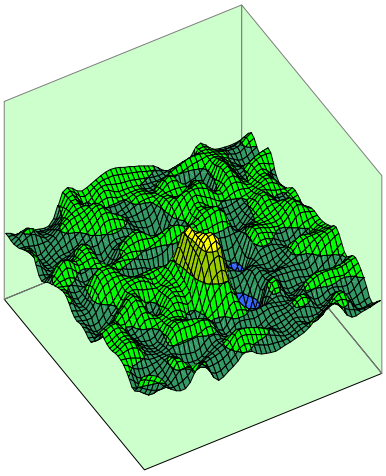
制御則 $u(t)$　30 試行時



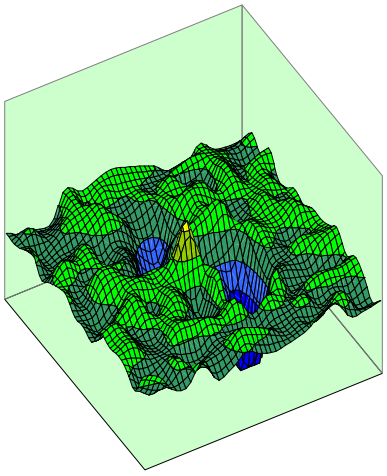
制御則 $u(t)$　10 試行時



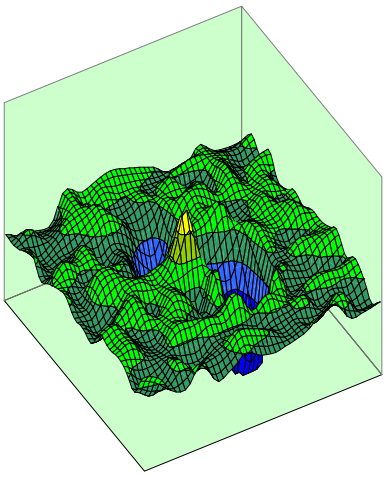
制御則 $u(t)$　40 試行時



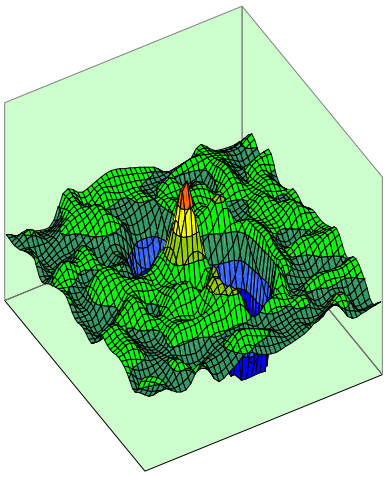
制御則 $u(t)$　20 試行時



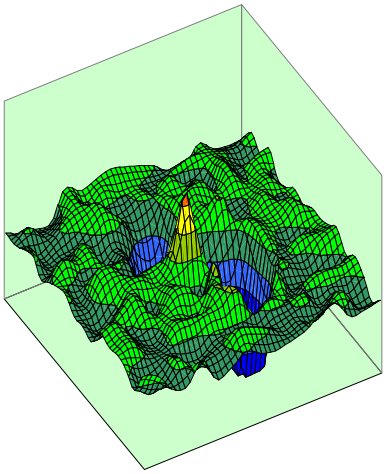
制御則 $u(t)$　50 試行時



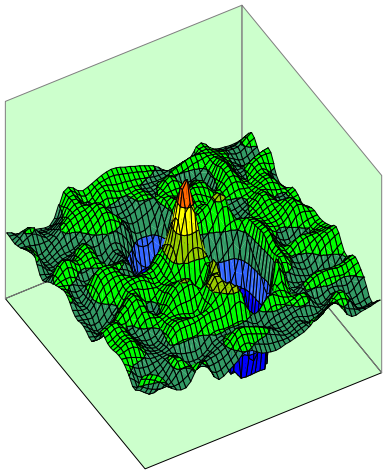
制御則 $u(t)$　60 試行時



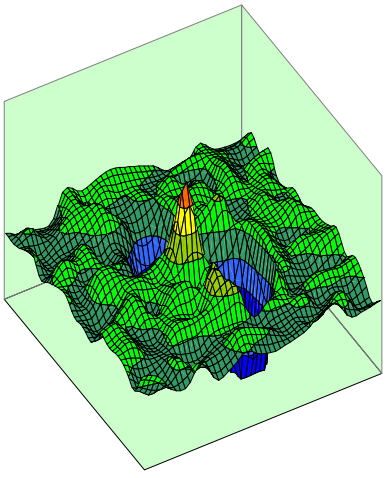
制御則 $u(t)$　90 試行時



制御則 $u(t)$　70 試行時



制御則 $u(t)$　100 試行時



制御則 $u(t)$　80 試行時

以上の結果から、学習シミュレータ上での歩行獲得に成功した。今回作成した学習シミュレータでは、歩行を獲得するための学習条件として一番効率が良いと考えた『腰の高さ、歩行速度を一定に保つ』という報酬関数を立て、CPG を van der Pol 方程式で生成しているが、この学習シミュレータは様々な学習条件や CPG で学習実験をできるようにプログラム設計しているため、プログラムを数行書き換えるだけで変更可能な汎用性の高いものを作成することができた。

5 ハードウェア

これまで説明してきたものはあくまでモデル上でのことであり、実際のロボットに適用するためには PC 上での学習を反映させ、実際に動かすための制御回路が必要となってくる。

そこで本研究では、実際のロボットでの動作に必要なサーボモータへの指令や、学習に必要な情報の PC への通信等を統括するものとしてマイコンの一つである PIC を用い、この PIC を中心として制御回路を作成した。以下にその手法を示していく。

5-1　　ロボット動作原理

本研究では歩行実験のためのヒューマノイドロボットとして、近藤科学から発売されている「KHR1」「KHR1HV」の枠組みを用いる。このロボットの各関節にはサーボモータが配置されており、上半身 6 自由度、下半身 10 自由度の計 16 自由度の動作が可能であるが、本研究では学習の効率化から上半身の 6 自由度、下半身の 6 自由度を固定し、下半身の 4 つのサーボモータを制御させ歩行を獲得できるように目指す。

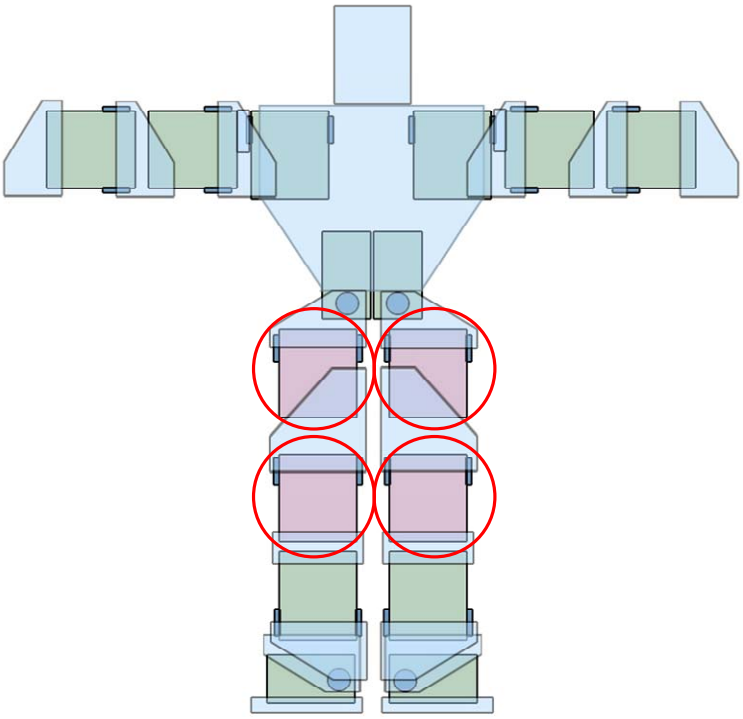


図 32　KHR-1 で制御するサーボモータ

図 32 の○がついているところが本研究で利用するサーボモータである。また、利用する「KHR-1」には既存の制御基盤が付属されているが、本研究では利用せず独自の制御回路を作成する。

5-2　　サーボモータ動作原理

図 33　KHR-1 のサーボモータ　KRS シリーズ

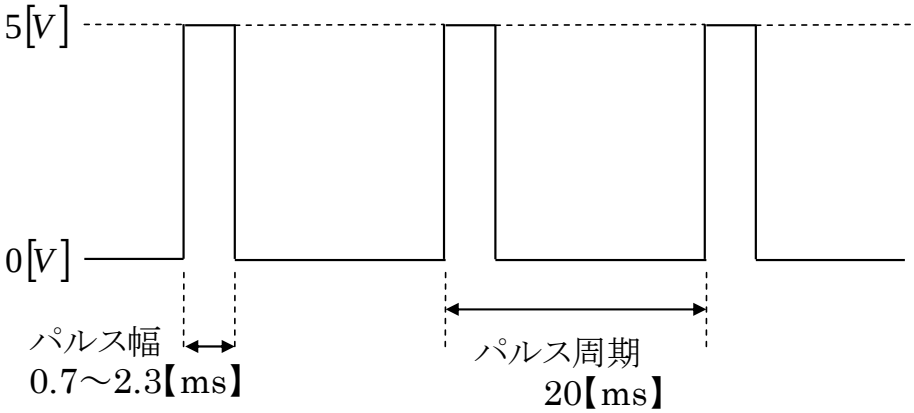


図 34　一般的なサーボモータへの信号

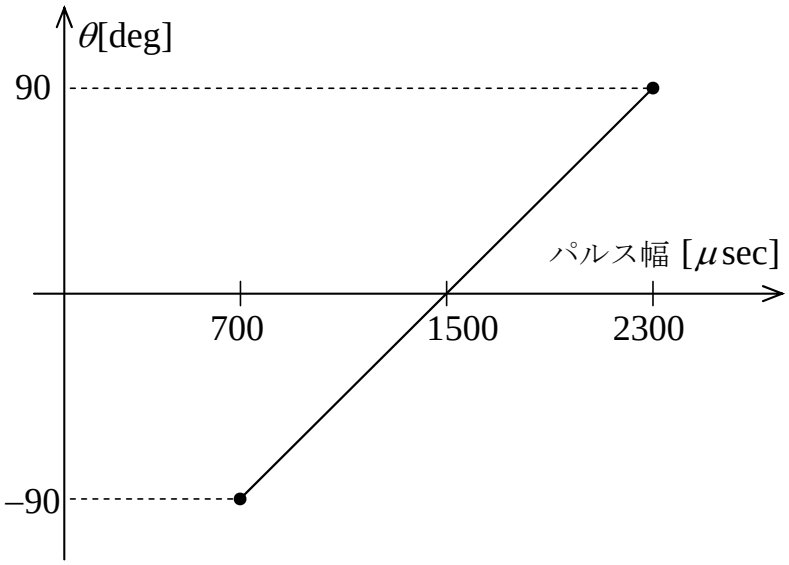


図 35　サーボモータ入力されたパルスと角度の対応

サーボモータには電源線 2 本と制御信号線 1 本とがあるが、この制御信号線には一般的に 図 34 に示したようなパルス信号を与える必要がある。サーボモータはこのパルスの幅に応じて駆動軸が対応した角度になるまで回転する仕組みになっている（図 35)。その原理は以下のようになっている。まずサーボモータの内部では駆動軸にポテンショメータ（可変抵抗）が取り付けられており、ポテンショメータの角度に対応した幅のパルスが発生するようになっている。モーターへ入力されたパルス幅がサーボモータ内部のパルス幅と異なっている場合、二つのパルス幅が同じになるまで駆動軸が回転する。このような方式によってサーボモータの駆動軸角度が外力によって変化した場合でも、サーボモータの駆動軸の範囲内で元の角度に戻ろうとする力が発生され、駆動角度が目標角度へ追従する。

5-3　　PICによるPWM制御

前章で説明したように、ロボットの動作制御にはパルスの幅を目標角度に従って変調する必要がある（以後 PWM 制御 : Pulse Width Modulation)。パルス信号の生成には FPGA の利用など、いくつかの方法があるが、本研究では PIC を利用し制御を行う。

5-3-1　　PICとは

図 36　本研究で利用した PIC16F88

PIC（Peripheral Interface Controller）とは、コンピュータに接続される周辺機器を制御するために開発された「マイクロコントローラ（マイコン)」と呼ばれる領域の IC である。つまり、使用目的がある範囲に限られているのでそれほど高い機能、高速性は持っていないが、周辺機器を制御するのに便利な機能は内蔵しているというものである。

5-3-2　　PICの特徴

PIC の特徴として以下のようなことが挙げられる。

- I/O ポート、メモリといったマイコンを構成するパーツがひとつのチップに納められているので回路の簡素化ができ、電子工作初心者でも扱いが容易
- 数百円～２千円程度と安価で購入可能
- アーキテクチャが共通であるために、上位互換性を持っている
- 最大 20MHz の発振子を使うことで 1 命令単位 0.2μsec と高速処理が可能
- 命令数が 35 と非常に簡素

PIC にはピン数やメモリ容量、D/A 変換機能の有無などによって様々な種類があり、主に、ベースラインシリーズ（命令長　12bit)、ミッドレンジシリーズ（命令長　14bit)、ハイエンドシリーズ（命令長　16bit）の 3 つに分けられる。本研究ではベースラインシリーズの 12F675、ミッドレンジシリーズの 16F88 を利用した。開発環境はマイクロチップ社製の「MPLAB IDE」を利用し、C 言語にて行い、コンパイラには「CCS コンパイラ」を用いた。

5-3-3　　PWM生成

本研究では始めに電源電圧で変化させた電圧をもとに、パルス幅を変化させる回路を作成した。実際に作成した回路の回路図が 図 37 であり、図 38 が作成した回路とサーボモータを接続したパルス生成実験を示しているものである。

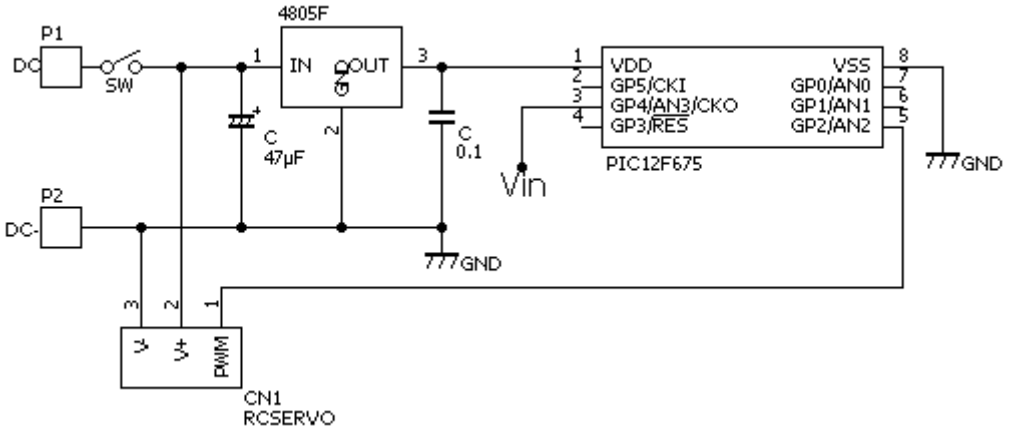


図 37　　PWM 制御を行う回路図

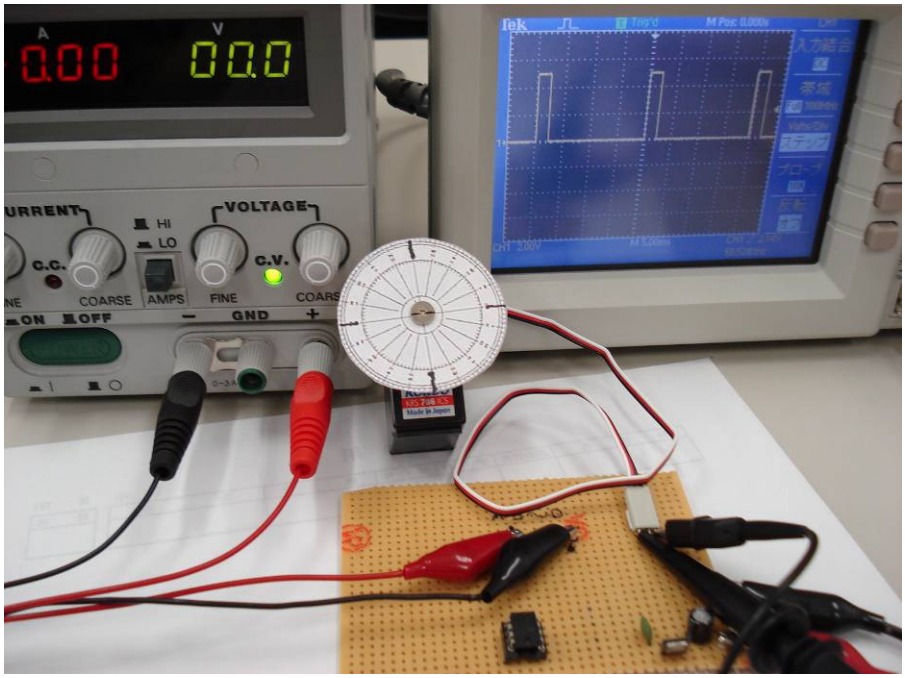


図 38　電源電圧最大値のときに生成されたパルス

図 38 の実験ではPICに電圧 0[v]を与え、そのときに出力されたパルスが 700μsecであることを示している。さらに電圧を 0[v]～5[v]へ上げていくとパルス幅も 700μsec～2300μ

secと比例して変化していくことが確認できた。

5-3-4　　PICプログラム

サーボモータの PWM 制御を PIC で行う方法を本章で示す。まず PIC 内部のタイマを利用し 20msec という一定周期で 0.7msec～2.3msec の幅のパルスを出力するという処理を行った。

処理の工程を示したフロー図が以下の 図 39 である。この図を元に以下で詳細に解説する。

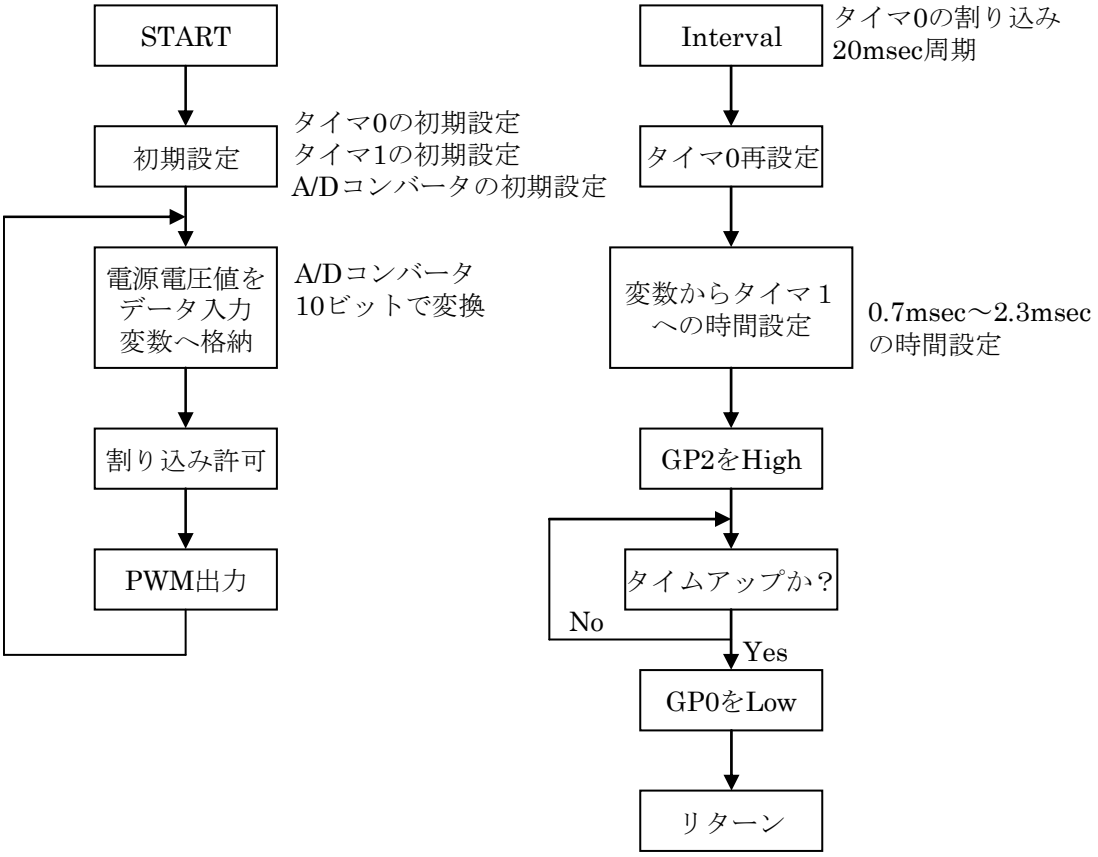


図 39　電源電圧値を利用した PWM 生成の処理工程

5-3-4-1　　PIC周期（タイマ 0）

まず 20msec 周期という一定時間はタイマ 0 の割り込みを使用して生成する。割り込みを使うことにより、メインプログラムがどこを実行している最中でも割り込みさえ許可されていれば、タイマ 0 がタイムアップしたときには、割り込み処理にハードウェアで強制的にジャンプさせられる。そして割り込み処理を完了すると、自動的に強制的にジャンプさせられたところに戻って、以前の処理を継続して実行する。こうすると、メインプログラムでは 20msec という周期を気にしなくても、時間がくれば自動的に割り込みが行われるの

で一定時間の処理が可能となる。

タイマ 0 本体は、8 ビットのカウンタなので 0 から 255 までカウントできる。そしてカウントが 255 の時にパルスが入力されると値が 0 に戻り、プログラムでチェック可能なオーバーフラグというビットが 1 になる。

実際に 20msec という待ち時間を作成するには、内部サイクルパルスを指定して、必要な時間になるようにタイマ 0 に値を設定して使用する。待ち時間は次のように決めることができる。

$$\begin{aligned}(\text{待ち時間}) &= (256 - \text{タイマー0設定値}) \\ &\quad \times (\text{プリスケーラ値}) \times \text{サイクル時間}\end{aligned} \qquad \text{Eq. 52}$$

ここでサイクル時間は 4MHz の内部サイクルパルスを指定しており、内部パルスの特性でクロック周波数の 1/4 の周波数のパルスが入力されるので 1MHz のパルスがプリスケーラの入力パルスということになり、1μsec ごとに 1 パルスが入ることになる。このため 20msec 数えるためには 20000 個のパルスを数えなければいけないが、タイマ 0 が 8 ビットで 255 までしか数えられないことからプリスケーラを利用し、パルスが 256 個入って初めて 1 個タイマ 0 に出力されるように初期設定する。

$$\begin{aligned}(\text{待ち時間}) &= (256 - 178) \times 256 \times 1\mu\text{sec} \\ &= 19968\mu\text{sec} \\ &\cong 20m\text{sec}\end{aligned} \qquad \text{Eq. 53}$$

この Eq. 53 によりタイマ 0 を設定し、サーボモータへ出力するパルスの周期とした。

5-3-4-2　　パルス生成（タイマ 1）

パルス幅はタイマ 1 を利用して生成させる。ここでは電源電圧から入ってくるデータ（A/D コンバータで 10 ビットに変換されたもの）をもとに対応するパルスへ変換し、そのパルス幅データをタイマ 1 の時間として設定してからパルスを High にし出力を開始する。その後タイマ 1 のタイムアウトを割り込みフラグをチェックすることで待ち、タイムアウトになったらパルス出力を Low にしてオフとする。これでタイマ 1 のカウント単位は 1μsec なので、この精度でパルス幅が制御できることになる。

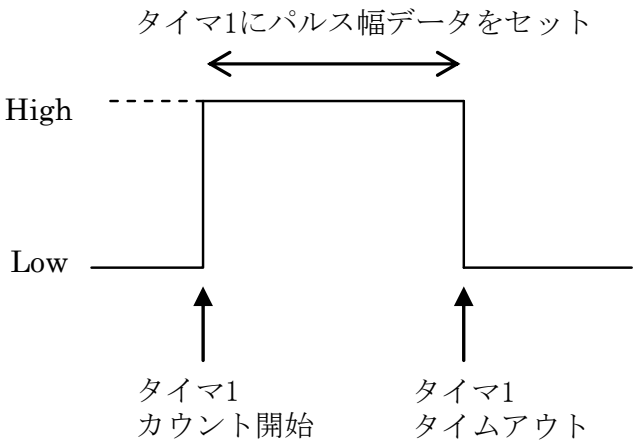


図 40　タイマ 1 を利用したパルス生成

以後の制御回路でもこの方法で 20msec 周期、可変幅のパルスを発生させる。

5-4　　PICでのCPG生成

3-3-2 でも少し述べが、PICの特性上、三角関数などの組み込み数学関数が使えないため、van der Pol方程式でCPGを生成する。

5-4-1　　PICでのvan der Pol方程式

学習シミュレータでも利用している結合van der Pol方程式（Eq. 17、Eq. 18、Eq. 19、Eq. 20）をPICで解くためにはパルス幅に対応した変換が必要である。PICでは浮動小数点演算も行えるものの、プログラムサイズや計算時間などの制限から符号なし整数で計算を行えると都合が良い。まず、変数 x_1、y_1、x_2、y_2 を新たな変数 u_1、w_1、u_2、w_2 に変換する。

変数 x_1、y_1、x_2、y_2 は 0 を中心に振動するが、これを 1500（パルス幅の中心）を中心に振動する変数とするため、以下の変換を施す。

$$x_1 = \frac{1}{100}(u_1 - 1500) \qquad \text{Eq. 54}$$

$$y_1 = \frac{1}{100}(w_1 - 1500) \qquad \text{Eq. 55}$$

$$x_2 = \frac{1}{100}(u_2 - 1500) \qquad \text{Eq. 56}$$

$$y_2 = \frac{1}{100}(w_2 - 1500) \qquad \text{Eq. 57}$$

ここで分数の分母の 100 は振動の振幅を調整する量である。この数を大きくする程 u_1、w_1、u_2、w_2 は大きな振動に変換される。この変換の両辺を微分すると

$$\dot{x}_1 = \frac{\dot{u}_1}{100} \qquad \text{Eq. 58}$$

$$\dot{y}_1 = \frac{\dot{w}_1}{100} \qquad \text{Eq. 59}$$

$$\dot{x}_2 = \frac{\dot{u}_2}{100} \qquad \text{Eq. 60}$$

$$\dot{y}_2 = \frac{\dot{w}_2}{100} \qquad \text{Eq. 61}$$

となるので、これら全て Eq. 17、Eq. 18、Eq. 19、Eq. 20 に代入して整理すると

$$\dot{u}_1 = \gamma\varepsilon_1\left(u_1 - 1500 - \frac{1}{3\times 100^2}(u_1 - 1500)^3\right) - \gamma(w_1 - 1500) - \gamma g(u_2 - u_1) \qquad \text{Eq. 62}$$

$$\dot{w}_1 = \gamma(u_1 - 1500) \qquad \text{Eq. 63}$$

$$\dot{u}_2 = \gamma\varepsilon_2\left(u_2 - 1500 - \frac{1}{3\times 100^2}(u_2 - 1500)^3\right) - \gamma(w_2 - 1500) - \gamma g(u_1 - u_2) \qquad \text{Eq. 64}$$

$$\dot{w}_2 = \gamma(u_2 - 1500) \qquad \text{Eq. 65}$$

となる。

次にこれを数値的に解くために、Euler 法で離散化する。そこで $\dot{u}_1 = (u_1(t+\Delta t) - u_1(t))/\Delta t$、$\dot{w}_1 = (w_1(t+\Delta t) - w_1(t))/\Delta t$、$\dot{u}_2 = (u_2(t+\Delta t) - u_2(t))/\Delta t$、$\dot{w}_2 = (w_2(t+\Delta t) - w_2(t))/\Delta t$ を代入して整理すると

$$\begin{aligned} &\dot{u}_1(t+\Delta t) = u_1(t) \\ &+ \Delta t\gamma\varepsilon_1\left(u_1(t) - 1500 - \frac{1}{3\times 100^2}(u_1(t) - 1500)^3\right) \\ &- \Delta t\gamma(w_1(t) - 1500) - \Delta t\gamma g(u_2(t) - u_1(t)) \end{aligned} \qquad \text{Eq. 66}$$

$$\dot{w}_1(t+\Delta t) = w_1(t) + \Delta t\gamma(u_1(t) - 1500) \qquad \text{Eq. 67}$$

$$\begin{aligned} &\dot{u}_2(t+\Delta t) = u_2(t) \\ &+ \Delta t\gamma\varepsilon_2\left(u_2(t) - 1500 - \frac{1}{3\times 100^2}(u_2(t) - 1500)^3\right) \\ &- \Delta t\gamma(w_2(t) - 1500) - \Delta t\gamma g(u_1(t) - u_2(t)) \end{aligned} \qquad \text{Eq. 68}$$

$$\dot{w}_2(t+\Delta t) = w_2(t) + \Delta t\gamma(u_2(t) - 1500) \qquad \text{Eq. 69}$$

これで $u_1(t)$、$w_1(t)$、$u_2(t)$、$w_2(t)$ は整数演算で計算でき、1500 を中心とした量となる。

5-4-2　　サーボモータの複数制御

これまで 1 つのサーボモータに対して PWM 制御を行っていたが、ここからは複数のサーボモータに対して CPG を用いた PWM 制御を行うことを考える。

当初､複数のサーボモータに対して同時に 図 41 のように出力処理を行うことを考えた。

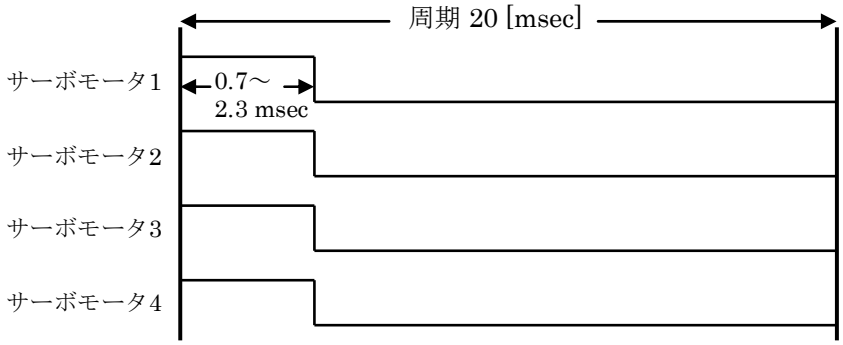


図 41　PIC からの同時 PWM 制御

しかしこの方法で処理を行おうとすると周期を測るタイマとパルス幅を計るタイマの計 5 つのタイマが必要多なるが、PIC 内部のタイマはタイマ 0・タイマ 1・タイマ 2 の 3 つしかないため、実現することができなかった。

そこで、一周期をいくつかに分割し、1 つずつパルスをずらしながら出力処理を行うようにすることで複数サーボモータの制御を実現した（図 42)。

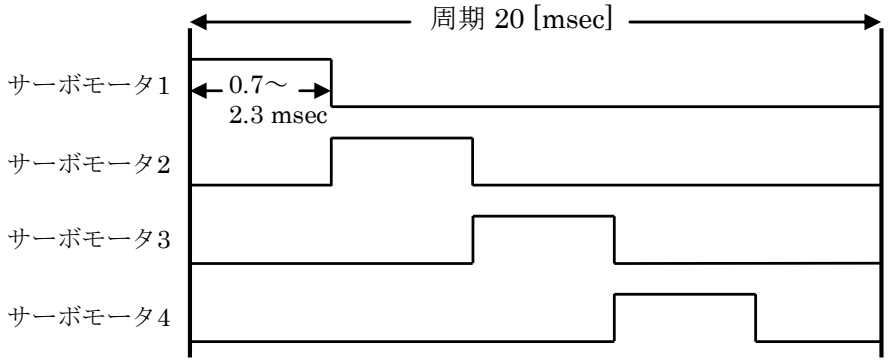


図 42　PIC からの遅れ PWM 制御

このように、20msec の周期の中でパルスをずらしながら生成することで、サーボモータのスムーズな動作を実現することができた。

以下の 図 43 はこのプログラムの処理工程を表しているものである。

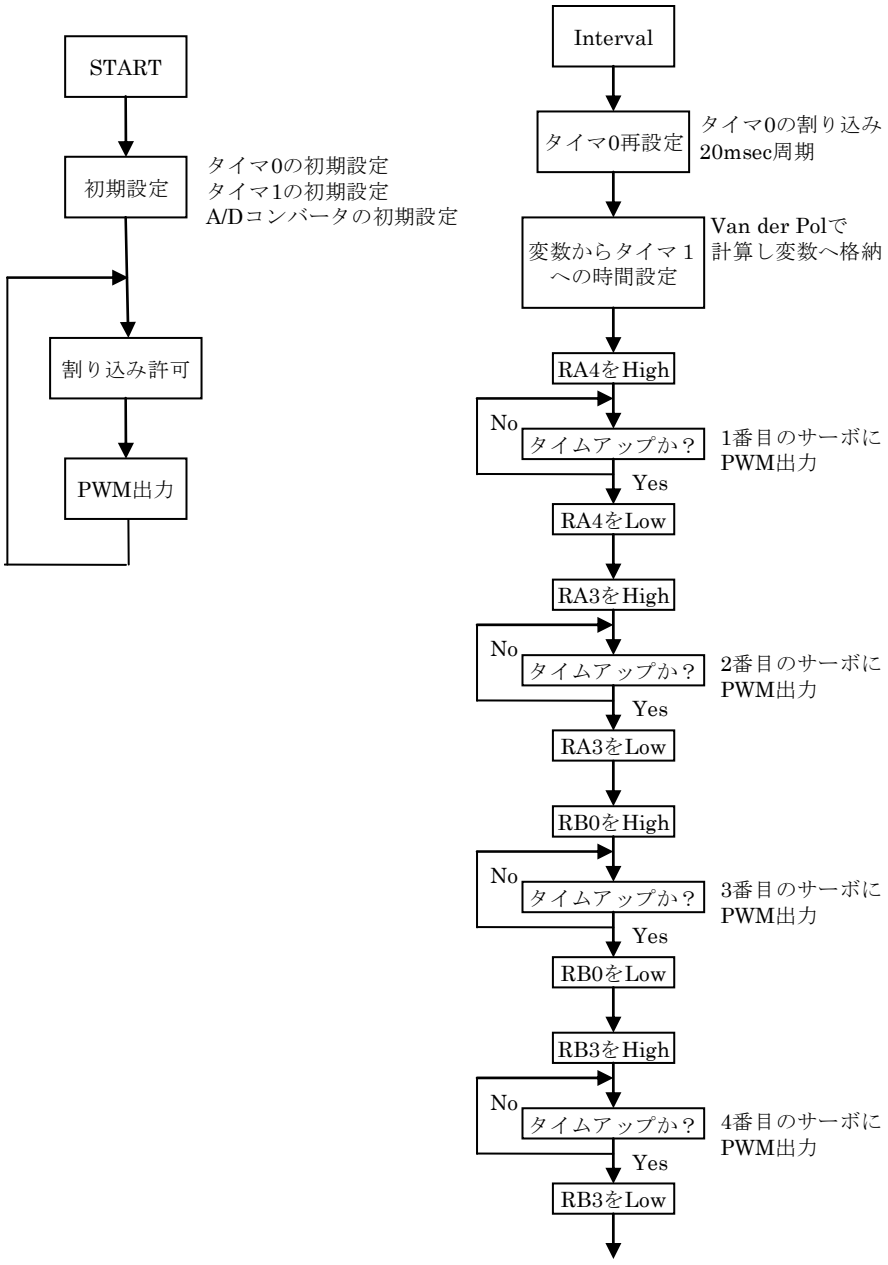


図 43　CPG を用いた PWM 制御の処理工程

5-5　　制御回路 I

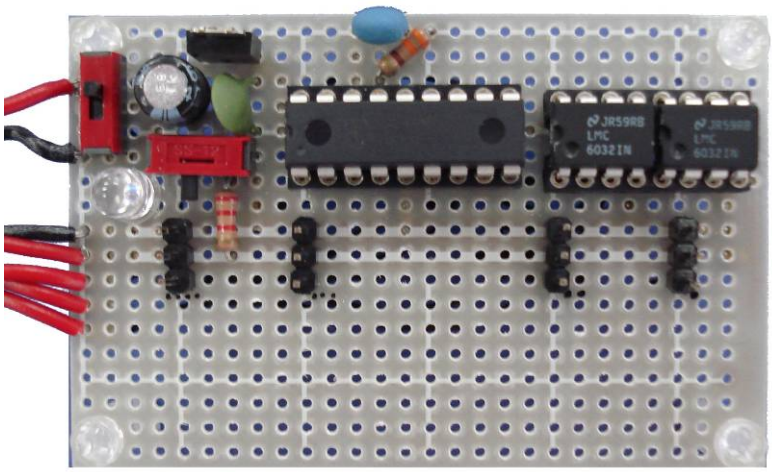


図 44　CPG を用いた PWM 制御を行う制御回路

前章までの処理をPICにプログラムし、4 つのサーボモータへPWM制御を行う回路を作成した（図 44）。5-3-3 で作成した電源電圧を用いたPWM制御の回路との相違点は、PIC を 12F675 から 16F88 に変更したことである。これは、CPG生成のための微分方程式の計算にある程度のメモリ確保が必要なこと、さらには制御するサーボモータが複数になったことによりPICのPINの本数が不足したためであり、ハイエンドな 16F88 を採用することでこれらが解決できた。

5-6　　ロボットへの適用

これまで作成した回路を実際のロボットへ搭載し、動作実験を行った。

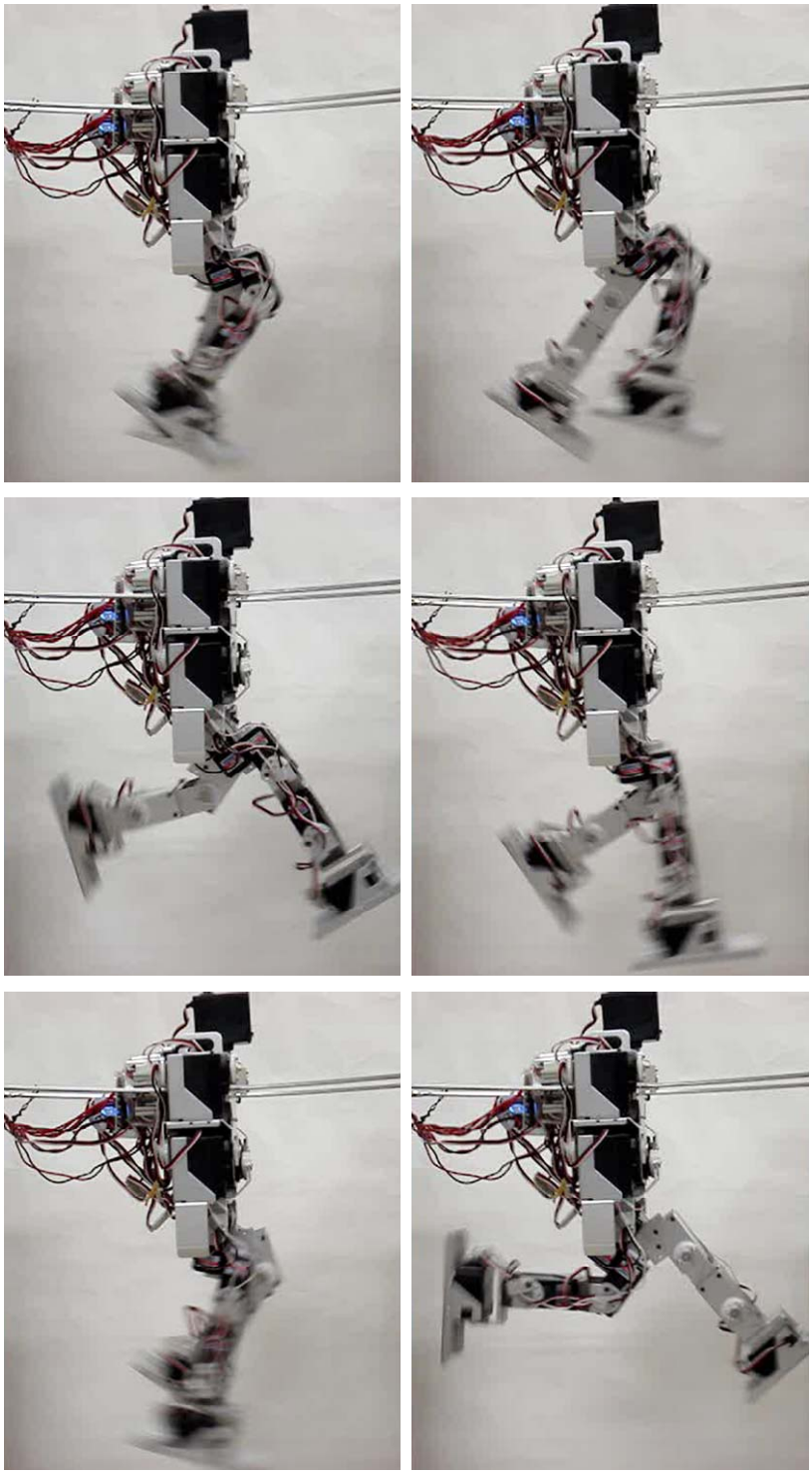


図 45　実際のロボットへの CPG 入力実験

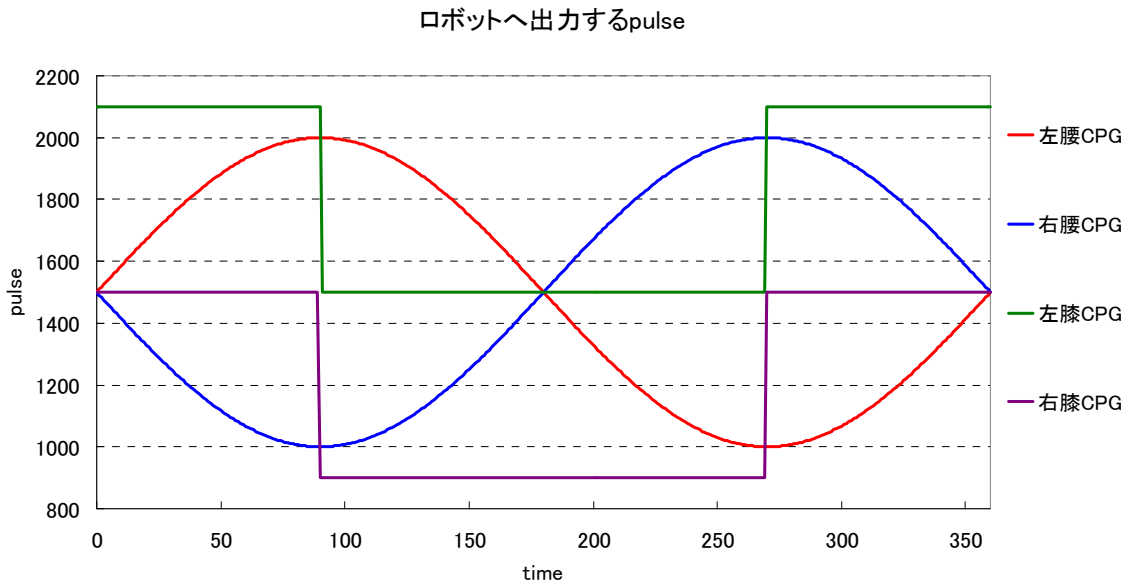


図 46　ロボットへ入力した CPG

ここで、図 46 のように腰の目標角には 5-4-1 で解説した結合van der Pol方程式の出力を用い、膝の目標角には腰のCPGが $\dot{x} > 0$ となったときに曲げるといった条件式を用いてPIC上で計算し出力した。

このCPGを用いて実際に動作実験を行っている様子が 図 45 である。この実験では、ロボットは空中に浮いているものの、歩行のような動作を行うことが確認された。

5-7　　センサ

本研究では学習する制御対象であるロボットの情報として腰、膝の現在のサーボモータの角度を用いることとし、それを取得するためにサーボモータにポジションキャプチャー機能を用いる。さらに、ロボットの歩行速度を計算するための加速度センサ、ロボットの転倒判定のためのジャイロセンサの 2 つのセンサを用いる。

5-7-1　　ポジションキャプチャー

今回利用したサーボモータ（近藤科学製 KRS シリーズ RED Version）は通常のラジコン用サーボモータとは異なり、ロボットならではの新機能が搭載されている。信号制御により、予め設定した動作のパラメータの切り替えが可能なキャラクタリスティックチェンジ機能、制御の開放が動作中に可能なパワーリダクション機能、サーボモータからの位置情報のフィードバックが可能なポジションキャプチャー機能などである。本研究では 3 番目のポジションキャプチャー機能を利用して、サーボモータの角度を取得し、それを学習に活用することとした。

ポジションキャプチャーでサーボモータの現在の角度を獲得するには、50μsec のパルスをサーボモータに出力する必要がある。すると、サーボモータ側は 100μsec 以内に、信号線を入力から出力に切り替えて、そのときのサーボモータの角度に対するパルス幅を出力する。このパルス幅を PIC で測定することにより、角度の取得が可能となる。なお角度のパルス幅を出力後、100μsec 以内にサーボの信号線は入力に戻る。

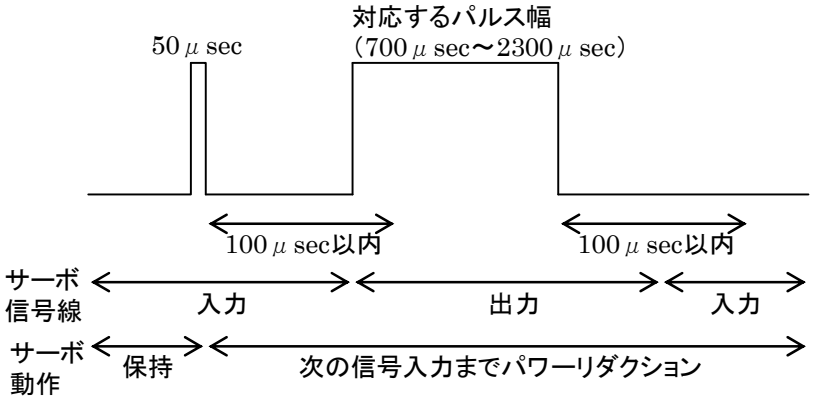


図 47　ポジションキャプチャーの流れ

図 47 はポジションキャプチャーの流れを示しているが、この図からも分かるように、ポジションキャプチャー機能で現在のポジションを獲得しようとすると、50[μsec]の信号を出力してから次の動作信号を出力するまで、パワーリダクション（脱力）を起こしてしまい、4 つ全部のサーボモータに毎回適用した場合、周期の半分の時間でどこかのサーボモータが脱力していることになる。このことから、ロボットの体勢維持が困難になってしまうと考えられる。

そこで、本研究では 4 つのサーボモータに順番にポジションキャプチャーを適用していき、脱力時間の減少を試みた。この方法でプログラムし、PICからの入出力をオシロスコープで測定し示しているのが 図 48 である。

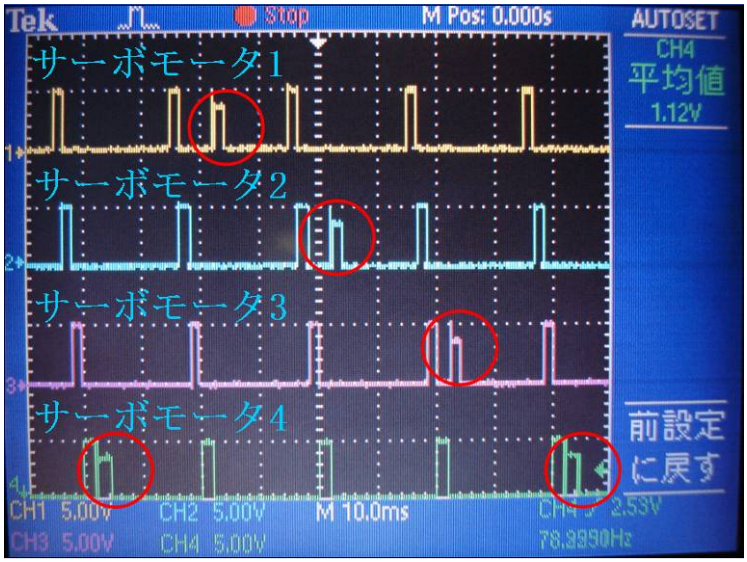


図 48　ポジションキャプチャー実行時の入出力パルス。赤丸で示されたパルスが角度に対応するパルスである。

図 48 の〇で囲まれているところがポジションキャプチャー機能によって出力された対応するパルスで、1 つのサーボモータににつき 80[msec]の周期でパルスが出力されている。それ以外のパルスは 図 38 などと同様、目標角に対応するパルスである。

本研究ではこのポジションキャプチャーを用いて各サーボモータの角度検出を行うだけでなく、学習計算に必要な各サーボモータの角速度、さらには報酬関数で利用する腰の高さを検出するのに用いる。

5-7-2　　ジャイロセンサ

本研究で使用するジャイロセンサは、近藤科学社製の「KRG-2」であり、歩行の状態姿勢を観測するために使用する。以下が KRG-2 のスペックである。

- 検出範囲　　±300[deg/sec]
- 静止時出力　　1.35[Vdc]
- 感度　　0.67[mV/deg/sec]
- リニアリティ　　±550[%FS]
- 応答性　　0.4[Hz]

このジャイロセンサをロボットの体の中心に設置し、体の中心から進行方向への角速度を検出することにより歩行時の転倒判定を行うようにした。

5-8　　データ通信

本章では歩行獲得時に必要な PC―PIC 間の通信について解説する。本研究では実際のロボットでの学習をできる制御器の設計を課題としており、学習を PC で計算させ、行動を PIC に担当させることを考えてきた。これはヒトに例えると、頭脳に相当する部分が PC で、自律神経に相当する部分を PIC に担当させようということであり、その相互間の情報のやり取りで学習を進めていくようにする必要がある。そこで本研究では PC 上の AD/DA ボード、PIC 上の AD コンバータ、制御回路上の DA コンバータを活用し、PC－PIC 間の通信を行う。

5-8-1　　PICからPCへの通信

これまでPIC内部で計算していた結果は実際にサーボモータを動かしてみないとわからず、さらにポジションキャプチャー機能により得たデータはオシロスコープ上のパルス幅としてしか観測できなかった。そこでこれらのデータをPC上で利用するためデータ通信を行った。PICから通信を行う先にはDA変換チップであるMAX525 があり、このDA変換チップからPCにアナログデータを送信するという手順である（図 49)。

PIC に内蔵されている MSSP（Master Synchronous Serial Port）モジュールは、EEPROM などの周辺 IC や、ほかのマイクロコンピュータと専用のシリアルインターフェースで接続することができ、高速の同期式通信を可能とする。本研究ではシリアル通信で接続される周辺 IC として A/D コンバータ MAX525 を用いた。

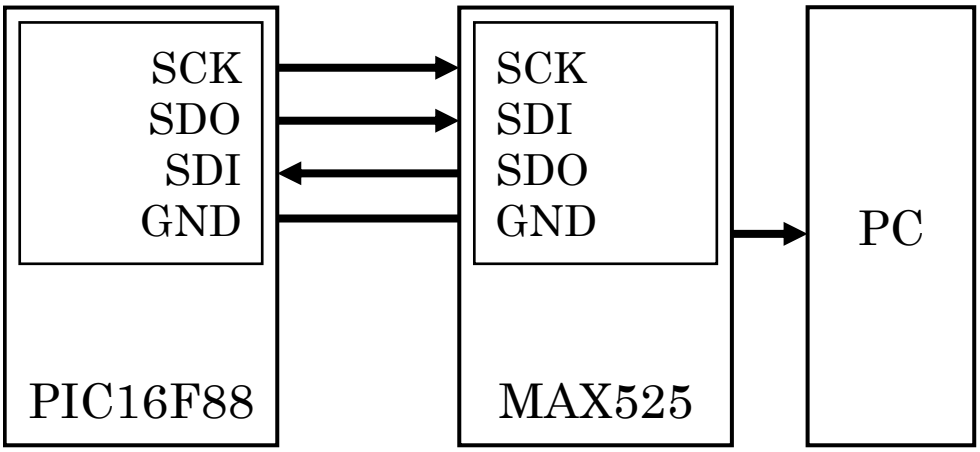


図 49　データ通信

MSSP の使い方には次の 2 種類がある。

- I2C（Inter-Integrated Circuit）モード

フィリップス社が提唱した方式で、2 本の接続線で 1 個のマスタに対し複数のスレーブとの間でパーティラインを構成し、最大 400kbps の通信が可能。

- SPI（Serial Peripheral Interface）モード

モトローラ社が提唱した方式で、3 本の接続線で構成し、数 Mbps の通信が可能。
本研究では SPI 通信を利用し、PIC－D/A コンバータの通信を行った。

MSSPをSPIモードで使うときの方式を表したものが 図 49 になっている。つまり、2 つのSPIのモジュールが互いにグラウンドを含め 4 本の線で接続され､PICがマスタ､MAX525 がスレーブとなっている。

通信は、マスタが出力するクロック信号（SCK）を基準にして、互いに向かい合わせて接続した SDI と SDO で、同時に 8 ビットごとのデータの送受信を行う。

本研究での利用法は､まずポジションキャプチャー機能で現在の各サーボモータの角度を検出し PIC 上で検出したパルスの幅を変数に格納する。そして、MAX525 が 12 ビットの DA コンバータであるため、格納した 16 ビットのパルス幅を 12 ビットに丸め、8 ビットごと 2 回に分けて DA コンバータに送信する｡その後 DA コンバータから PC 上の AD ボードに出力し読み取るという流れになっている。

ここで問題となるのが、ポジションキャプチャーによって検出した角度データのサンプリング時間が 80[msec]であるということでる（図 50)。

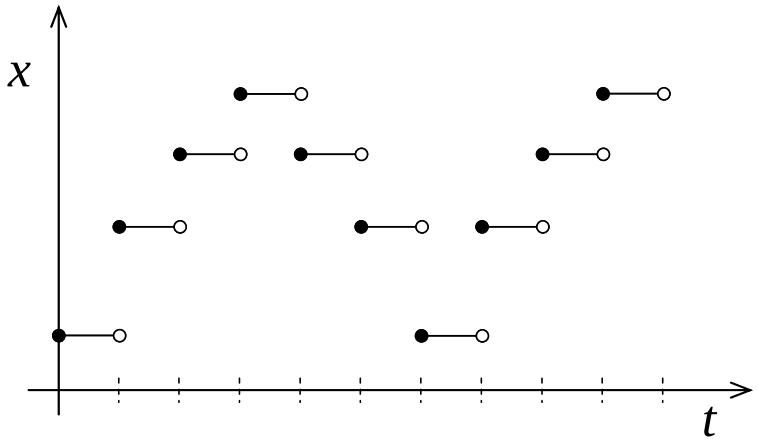


図 50　ポジションキャプチャーによる出力

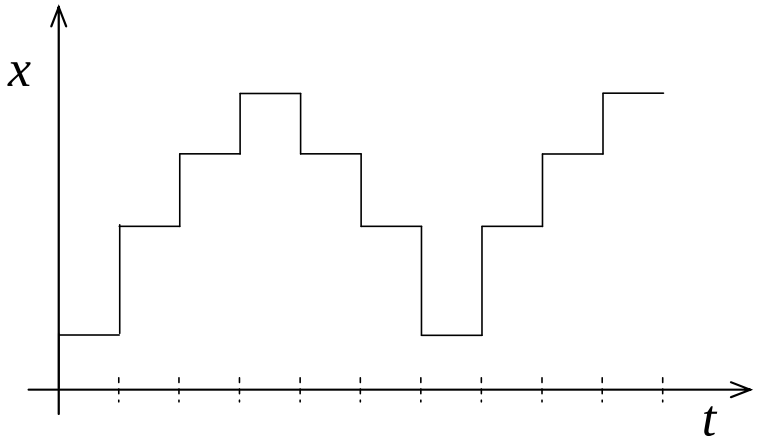


図 51　ポジションキャプチャーから出力されたデータを PC 上でグラフ化

この状態のデータをPC上で読み取ると 図 51 のように断続したデータとなってしまう。この結果、後に行う角速度等の計算ができないため、PC上のプログラムでローパスフィルタを作成し、図 52 のように滑らかなデータへと書き換える必要がある。

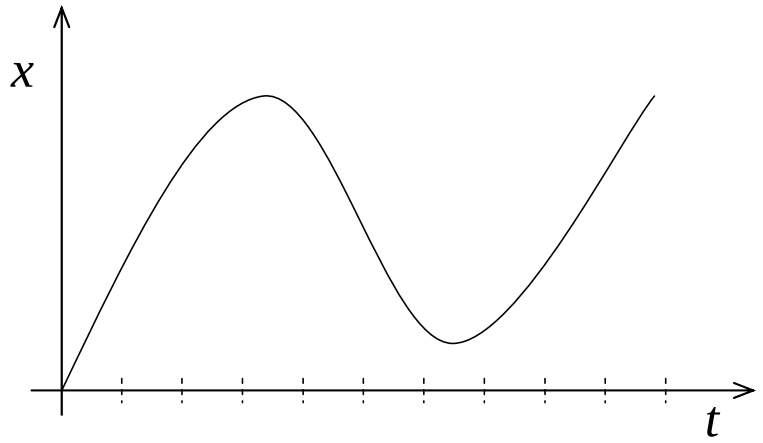


図 52　ローパスフィルタ後のポジションキャプチャー出力データ

ローパスフィルタは入力を $x_{(t)}$ 出力を $y_{(t)}$ とした場合

$$\dot{y}_{(t)} = -\frac{1}{\tau} y_{(t)} + \frac{1}{\tau} x_{(t)} \qquad \text{Eq. 70}$$

で計算できる。ただし τ は時定数とする。この式を Euler 法で解くと

$$\begin{aligned} y_{(t+\Delta t)} &= y_{(t)} + \Delta t\left(-\frac{1}{\tau} y_{(t)} + \frac{1}{\tau} x_{(t)}\right) \\ &= \left(1-\frac{\Delta t}{\tau}\right) y_{(t)} + \frac{\Delta t}{\tau} x_{(t)} \end{aligned} \qquad \text{Eq. 71}$$

と書くことができる。

そこで

$$\alpha = \frac{\Delta t}{\tau} << 1 \qquad \text{Eq. 72}$$

と定め Eq. 71 に代入すると

$$y_{(t+\Delta t)} = (1-\alpha)y_{(t)} + \alpha x_{(t)} \qquad \text{Eq. 73}$$

という式を求めることができる。この Eq. 73 が本研究で用いたローパスフィルタの計算式で、α は現在の入力を出力へ加える際の割合を表している。本研究では $\alpha = 0.1$ と定めた。

ここでポジションキャプチャーを用いてサーボモータの特性実験を行う。実験方法は 4 章で作成した学習シミュレータに通信のためのプログラムを書き、まずサーボモータに一定周期のパルス出力を行った。この結果が以下の 図 53 である。

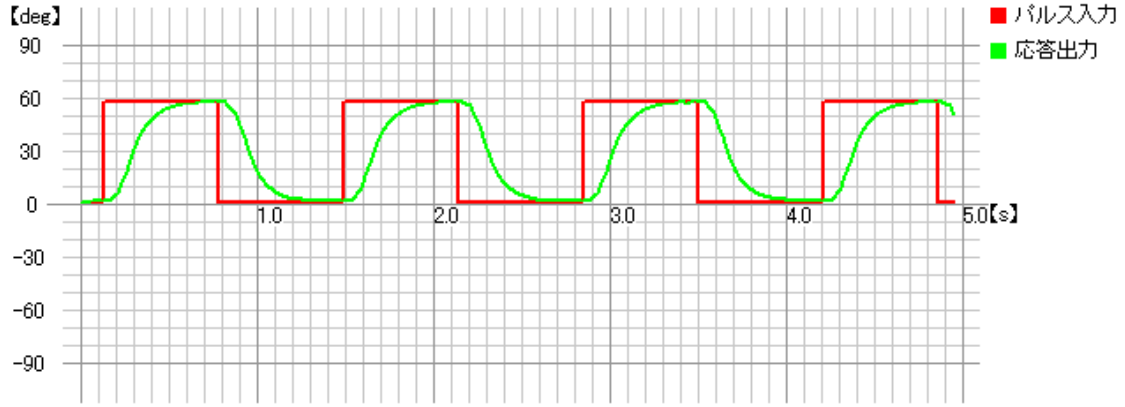


図 53　本研究で用いたサーボモータの特性

図 53 はシミュレータ上のグラフで、赤い線がサーボモータの目標角であり、緑の線がそのときの実際のサーボモータの角度変化である。これは本研究で用いたサーボモータ「KRS シリーズRED Version」の特性を表すグラフで、入力から出力への応答遅れを示している。

なお、本研究では 図 53 で示したとおり $\alpha = 0.1$ と定めて進めたが、$\alpha = 1$ の状態（ローパスフィルタなし）で測定した場合、$\alpha = 0.5$ の状態で測定した場合の実験も行ったので以下に示しておく。図 54 では 80msec でサンプリングしてきている様子を確認することができる。

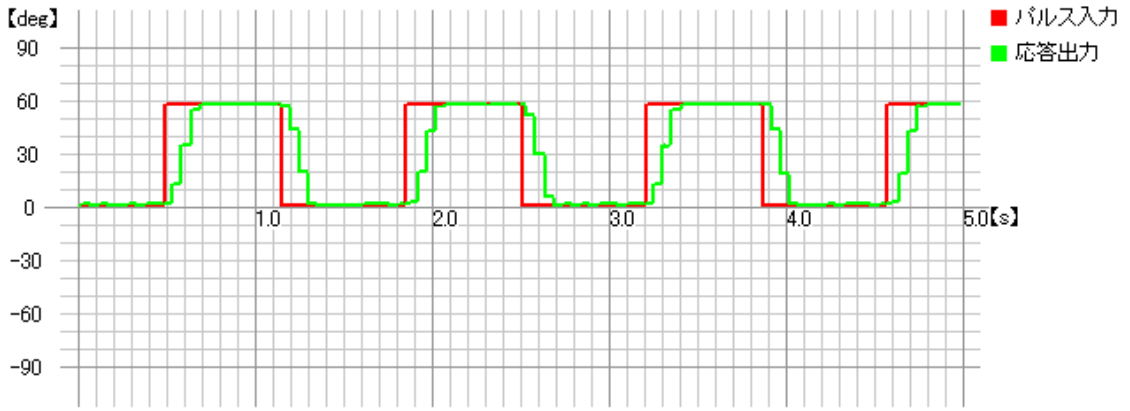


図 54　$\alpha = 1$

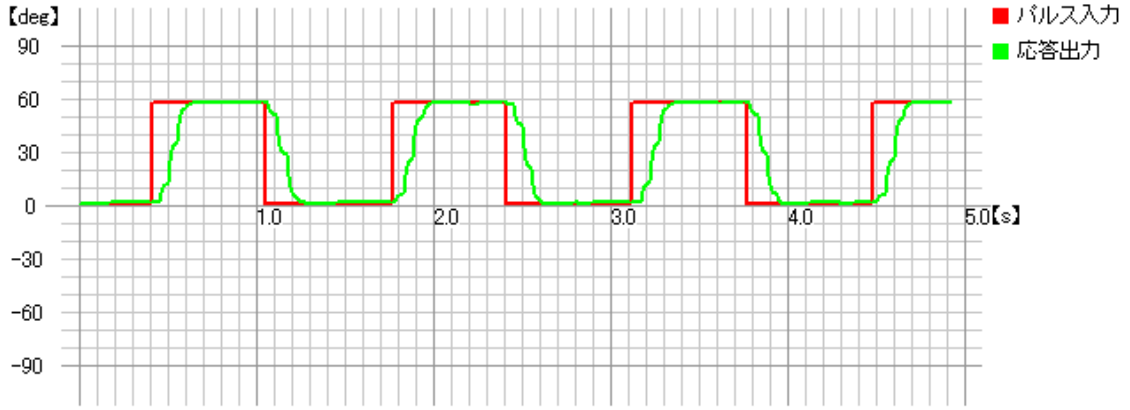


図 55　$\alpha = 0.5$

次に **5-6** で行った実験でも同じように応答遅れの測定実験を行った。これはPIC上で計算されたvan der Pol方程式の値と実際に動作した状態を 図 53 と同じく学習シミュレータ上で測定しグラフ化したものである。結果は以下の 図 56 である。このように目標角と実際の動作の応答遅れがあることが分かる。

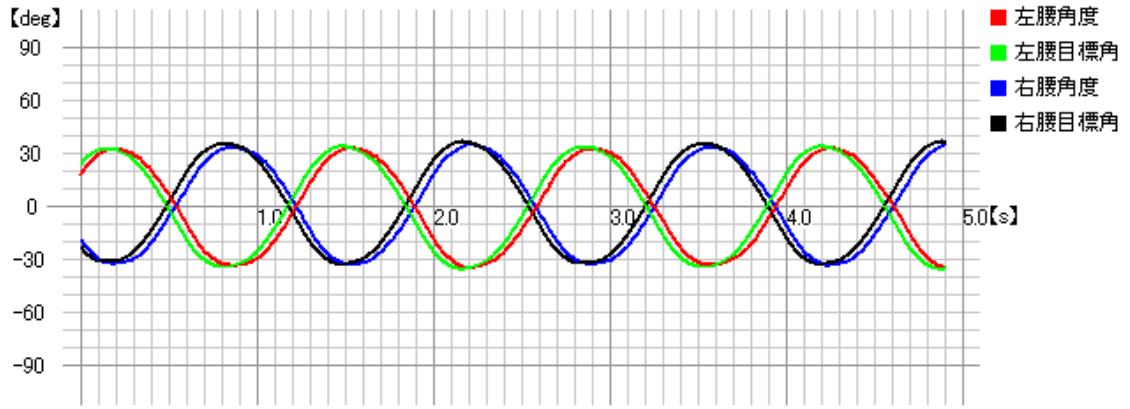


図 56　目標角と実際の動作の応答遅れ

5-8-2　　PCからPICへの通信

前章までは PIC から PC へ現在の状態を出力する通信を説明してきた。次はこの逆の通信である PC から PIC 間の通信を説明していく。

本研究では PC から PIC への通信として PC 上にある DA ボードと PIC 内部にある AD コンバータを用いて実現した。

PIC内蔵のAD変換器は、逐次変換型の変換機で高速に変換が行われる。さらに、変化する信号を安定して変換できるようにするサンプルホールド回路も内蔵している。以下の 図 57 はAD変換器の内部構造である。

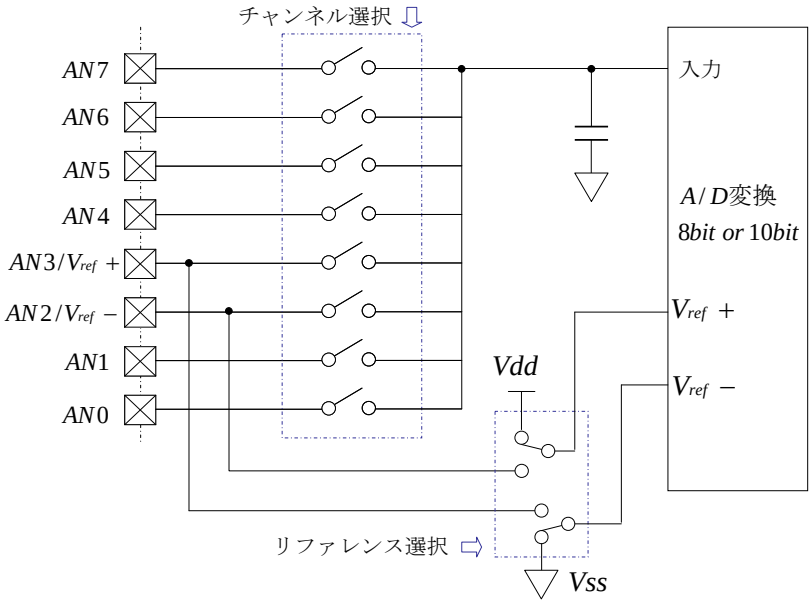


図 57　AD 変換器の内部構造

5-9　　制御回路Ⅱ

これまでのジャイロセンサ、ポジションキャプチャー機能、PIC－DA コンバータ間のデータ通信をすべて１つの回路で実現するため、制御回路Ｉを改良した。

実際に作成した制御回路Ⅱは以下のようになっている。

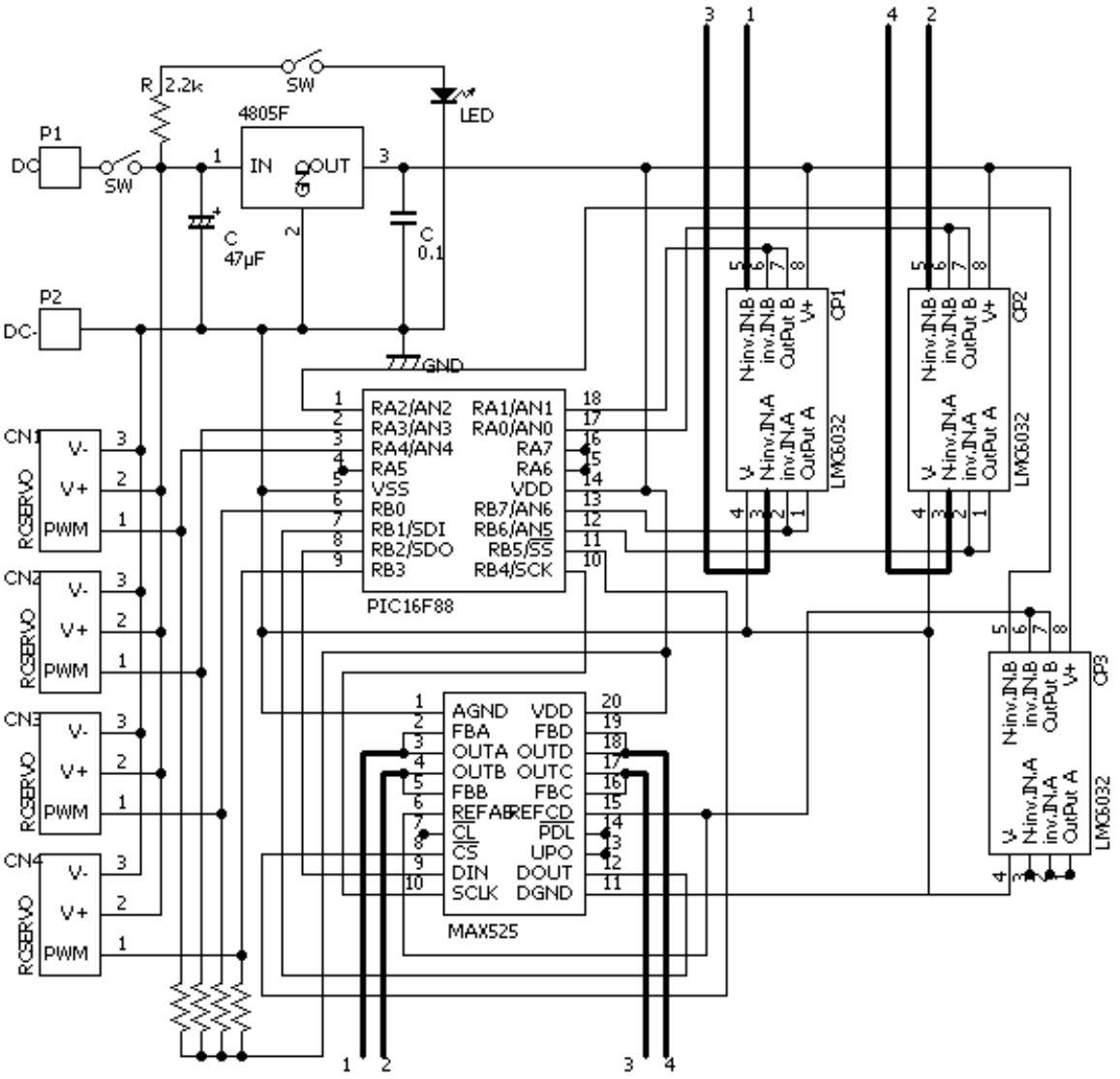


図 58　回路図

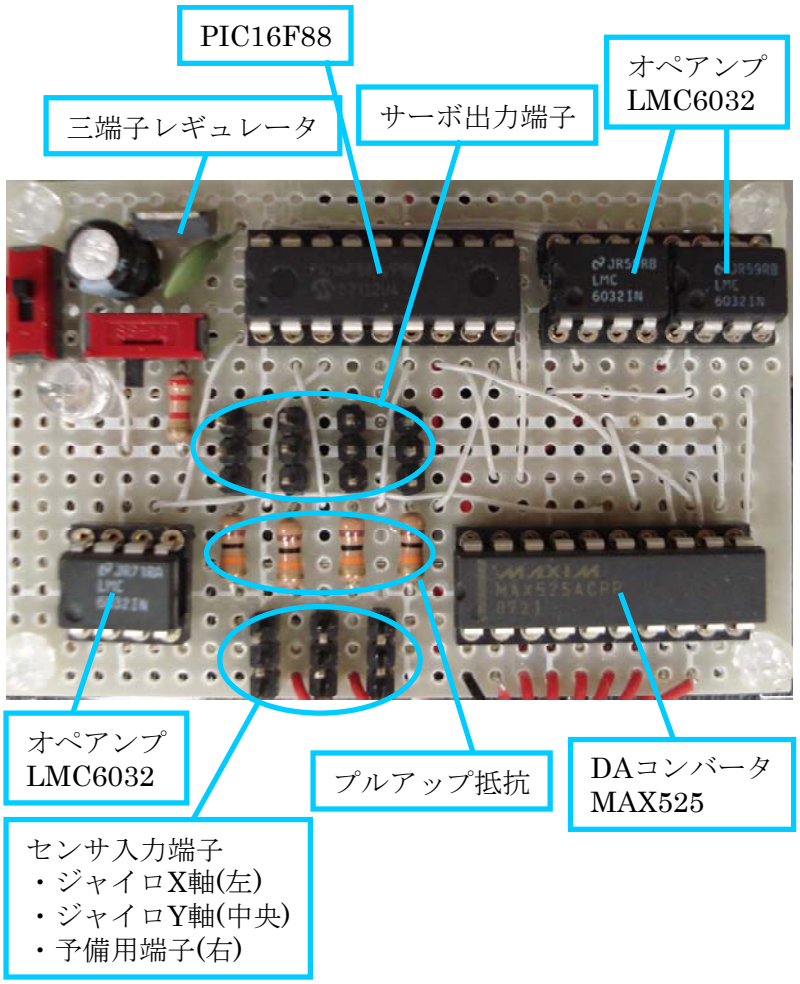


図 59　最終的なロボットに搭載する制御回路

図 59 の制御回路は最終的にロボットに搭載する制御回路であり、このときのPICの各PINの主な用途は以下のようになっている。

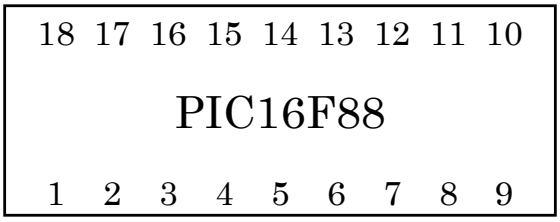


図 60　PIC16F88 の PIN 配列

PIN	PIN 種類	用途
1	RA2/AN2	DAC リファレンス電源
2	RA3/AN3	PWM 出力（サーボ 2）

3	RA4/AN4	PWM 出力（サーボ 1）
4	RA5/AN5	未使用
5	VSS	Ground
6	RB0	PWM 出力（サーボ 3）
7	RB1/SDI	SPI 通信用
8	RB2/SDO	SPI 通信用
9	RB3	PWM 出力（サーボ 4）
10	RB4/SCK	SPI 通信用
11	RB5/ $\overline{SS}$	SPI 通信用
12	RB6/AN5	PC からのアナログ入力用 1
13	RB7/AN6	PC からのアナログ入力用 2
14	VDD	電源 5V
15	RA6	未使用
16	RA7	未使用
17	RA0/AN0	PC からのアナログ入力用 3
18	RA1/AN1	PC からのアナログ入力用 4

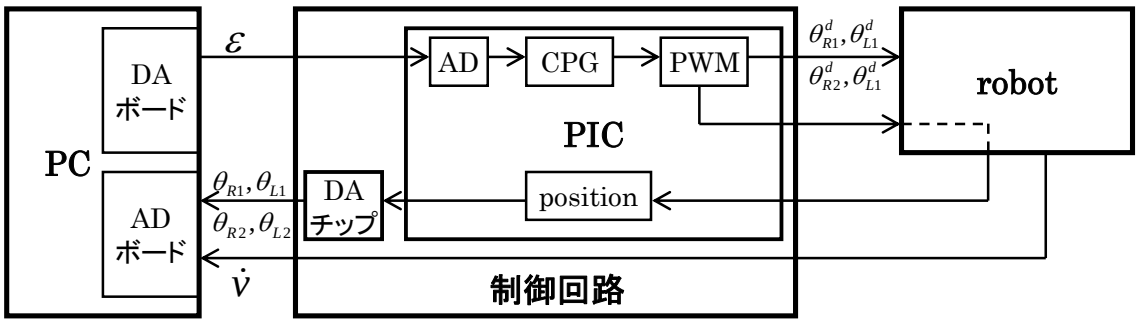


図 61　制御回路構成

ここで、今まで説明してこなかったオペアンプの利用法について説明していく。本研究ではオペアンプをバッファとして利用した。

【ボルテージフォロワ】

本研究で作成した制御回路はPC上で計算した結果をDAボードによって変換して0から5[V]の電圧によってPICを制御する。ここで問題となるのがPC－PIC相互間の影響である。PC－PIC間をなにもなく接続した場合、予期せぬ電流がPCやPICに流れる可能性があり、このことにより故障することも考えられる。そこで指定した電圧だけを後ろに伝える回路をPC－PIC間に入れる必要がある。この回路がボルテージフォロアで以下にその概要を説明していく。

増幅回路の主な役目の一つは電圧を大きくするという意味の増幅である。もう一つの意味が、「電圧は変えないけど、大きな電流出力に耐えられるようにする」増幅である。多く

のセンサや受動部品（抵抗、コンデンサ）を組み合わせた回路は、それに続く回路で電流を吸ってしまう回路が多いため、その間だけ電流を増やす必要がある。この目的で使われるのがボルテージフォロワで以下に示す図 62 がその回路図である。

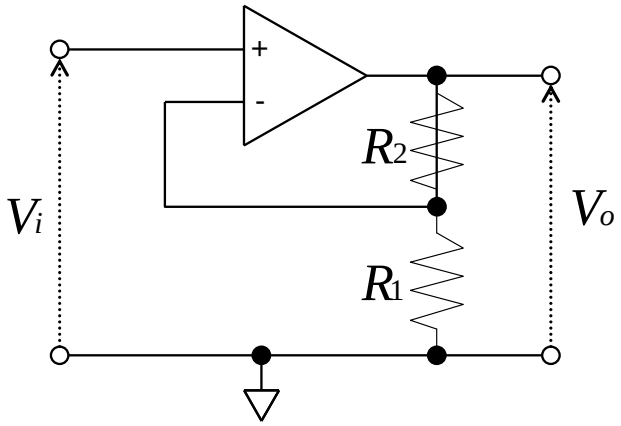


図 62　ボルテージフォロワ回路図

これは、見方によっては、非反転増幅回路の一種である。非反転増幅では、出力電圧は

$$V_o = \left(1 - \frac{R_2}{R_1}\right) V_i$$

で得られる。ここでボルテージフォロワの接続では、$R_2 = 0$（直結）、$R_1 = \infty$（無接続）に相当する。すると

$$V_o = V_i$$

が成り立つ。

この回路を利用し、PC―PIC 間の直接的な電流の行き来を減少し、データのやり取りに必要な電圧だけを互いに伝えることができる。

以上がロボットを制御するために製作した制御回路の説明である。次にこの制御回路をどのように利用して学習に結び付けていくかを解説する。

6　実機体での学習

これまで学習計算するための学習シミュレータ、そして実際のロボットでの学習を実現するための制御回路を解説してきた。ここからはソフトウェアとハードウェアを一体化し、実際のロボットでの学習をするための手法について解説していく。

ここで学習に必要な状態変数をもう一度整理する。まず本研究では強化学習によって歩行を獲得するため、価値関数Vと制御則uをTD学習によって予測している。このとき必要な状態変数がいくつかある。1つには現在のロボットの状態であり、内訳は腰（左右）の角度θ_{R1},θ_{L1}、さらにこのときの腰の角速度$\dot{\theta}_{R1},\dot{\theta}_{L1}$である。この角度$\theta_{R1},\theta_{L1}$の取得には 5-7-1 で説明したポジションキャプチャーを利用し、角速度$\dot{\theta}_{R1},\dot{\theta}_{L1}$は取得した角度から計算で求めることができた。2つ目は報酬関数で利用する歩行の速度である。本研究では歩行速度vを検出するために加速度センサを設置し積分することによって速度を出している。3つ目も報酬関数で利用する歩行時の腰の高さである。腰の高さhを検出するにはポジションキャプチャーによって取得した角度$\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2}$から計算して求めることができた。以上が学習するときに必要な状態変数である。さらに学習によって更新された制御則を制御回路上で反映させるために必要なパラメータがPIC上のvan der Pol方程式で利用するεとγである。

以上をまとめると、角度$\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2}$、角速度$\dot{\theta}_{R1},\dot{\theta}_{L1}$、歩行速度$v$、腰の高さ$h$をもとに学習させ最適な CPG に調整するために PIC 上のεとγを書き換える必要がある。以下が実際のロボットで学習していくための詳細な処理工程である。

1　PC から初期の CPG パラメータε、γを出力する。

2　PC からの出力データε、γをもとに PIC で van der Pol 方程式を解き CPG を生成する。

3　生成された CPG をパルスに変換し、各サーボモータへ PWM 制御を行う。

4　入力されたパルスをもとにサーボモータが動作しロボットが歩行する。

5　動作しているロボットの現在の状態をポジションキャプチャーによって$\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2}$を検出、歩速速度$v$を加速度センサによって検出、ロボットの転倒判定をジャイロセンサによって検出する。

6　加速度センサ、ジャイロセンサの値は直接 PC 上の AD ボードに出力し、ポジションキャプチャー機能で PIC に出力された現在の角度データ$\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2}$は一度 DA コンバータに SPI 通信し、その後 PC 上の AD ボードに出力する。

7　工程 6 で得たデータをもとに歩行速度v、腰の高さh、腰のサーボモータの角速度$\dot{\theta}_{R1},\dot{\theta}_{L1}$を PC 上で計算する。

8　歩行速度v、腰の高さh、腰と膝の左右のサーボモータ角度$\theta_{R1},\theta_{L1},\theta_{R2},\theta_{L2}$、腰のサーボモータの角速度$\dot{\theta}_{R1},\dot{\theta}_{L1}$をもとに価値関数$V$、制御則$u$を学習によってPC上で計算し新たなCPGのパラメータε、γを更新する。

9　更新されたε、γのデータをアナログ値に変換してPC上のDAボードからPICに出力する。

10　PICに入力したデータε、γをAD変換し、新たなCPGパラメータとして更新する。

11　工程2～工程10を繰りかえし行う。

以上が実際のロボットで学習していくための処理工程である。図 63は実際のロボットへの適用をするときのPC、PIC、ロボットの接続を図にしたものである。

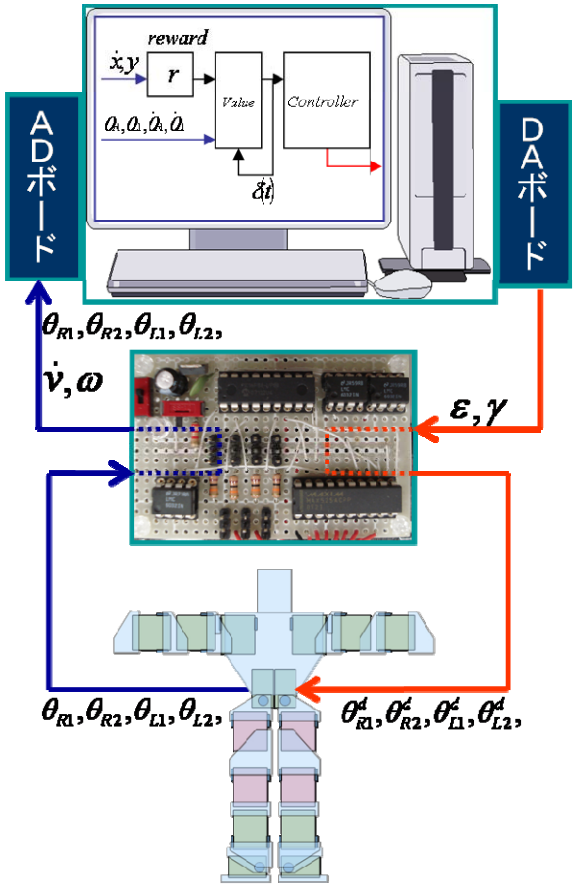


図 63　実際のロボットへの適用

ここで作成した環境を用いて歩行実験を行う。これはPCから出力されたε（van der Pol方程式のパラメータ）を元にPIC内部で動作出力に変換し、さらにロボットの状態をPCにフィードバックさせる実験である。図 64図 65 に実験の様子を示す。

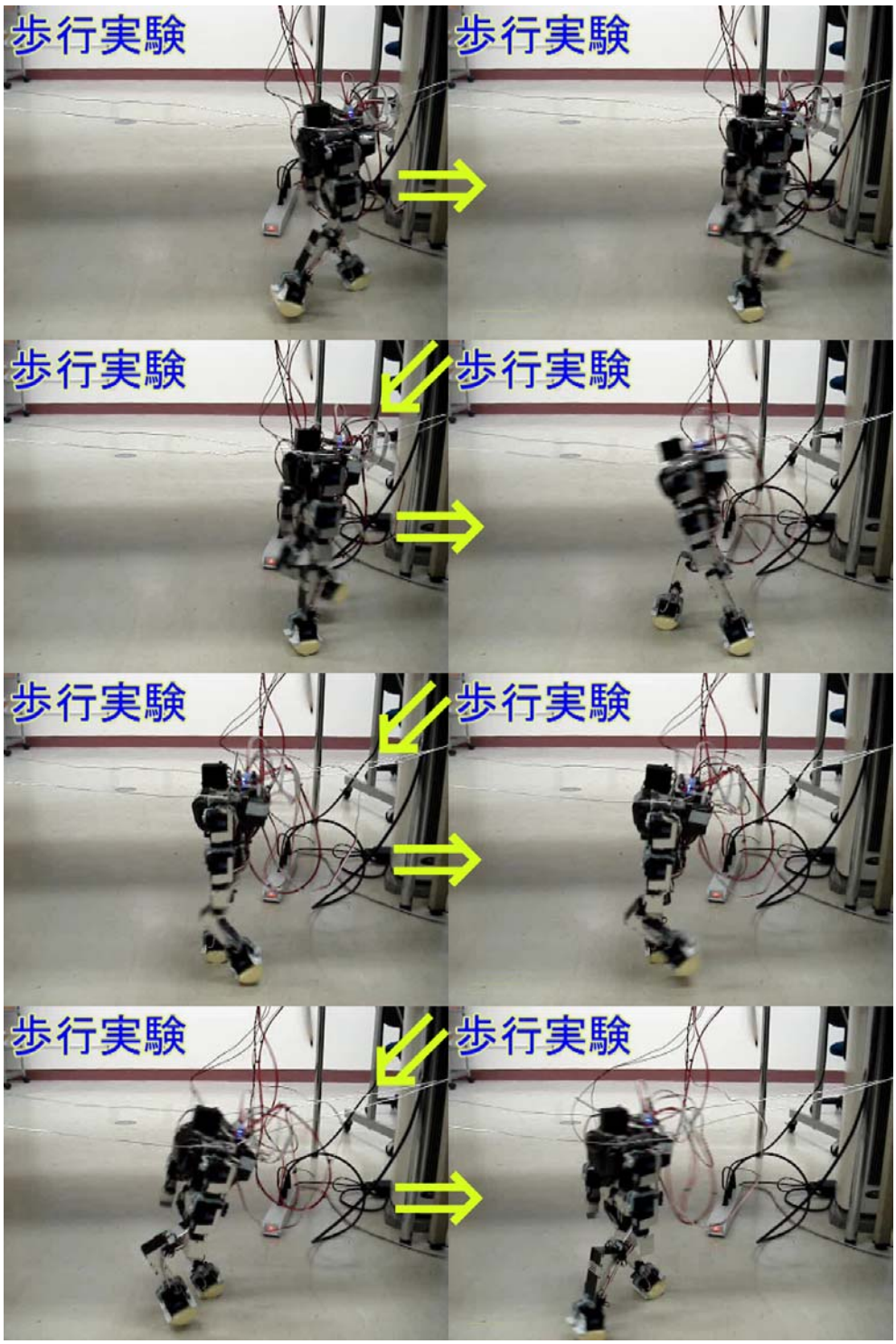


図 64　歩行実験 1

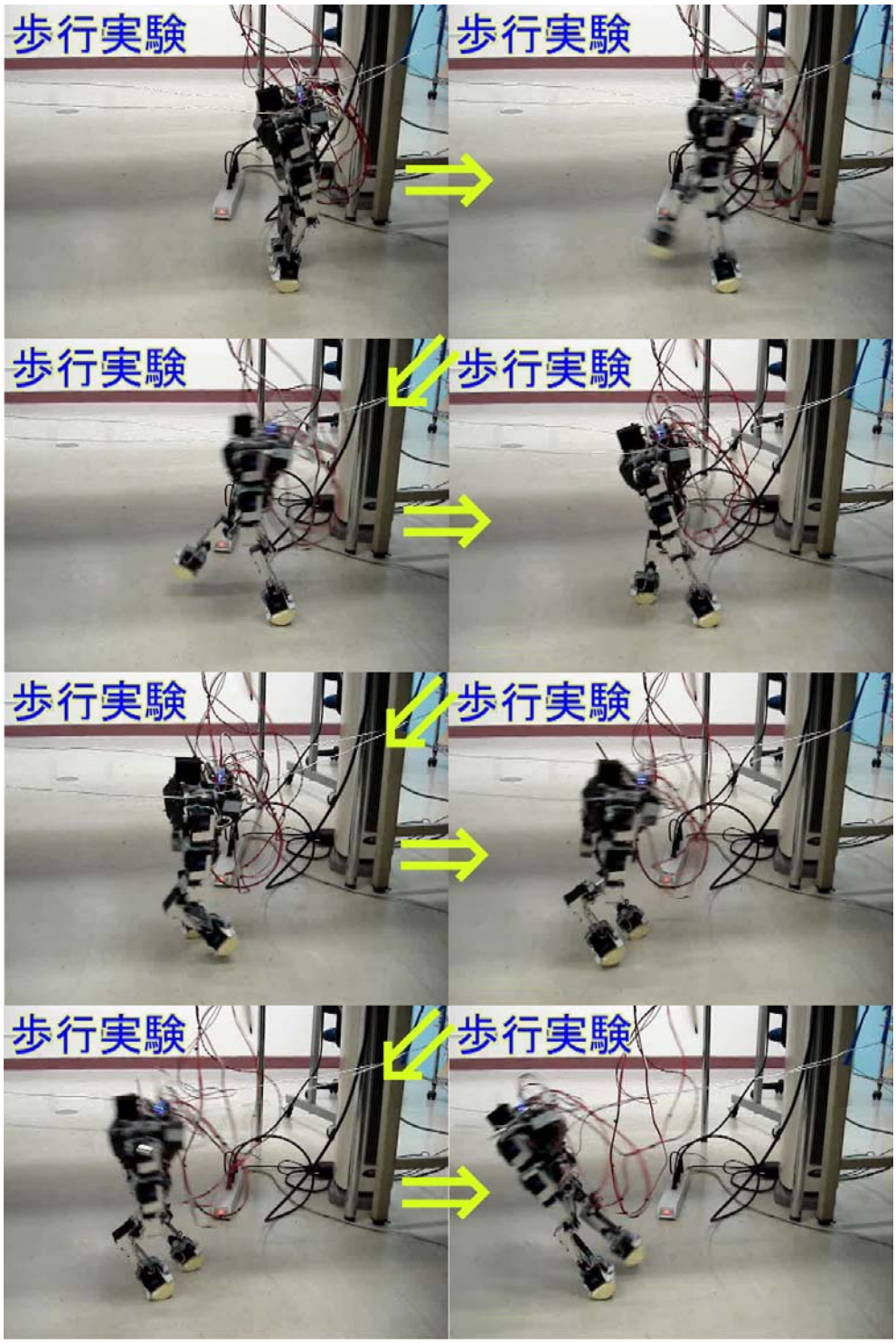


図 65　歩行実験 2

この実験で用いたロボットは動作実験でも用いた「KHR1-HV」で、地面に接地する面を円弧にしてモデルと同じ点接地に近づけている。さらにシミュレーションでは二次元平面で行われているため、ワイヤーを直進方向へ配置し直進歩行になるようロボットを拘束した。

実験の結果、数歩の歩行を行うことができたが KHR1-HV での学習実験を行うまでには至らなかった。原因はシミュレーションでは上半身を考慮しないモデルとなっていることが考えられる。このため実際ロボットで歩行獲得を目指すには、下半身だけのロボット作成、若しくは上半身を考慮したモデルの作成が課題となるが、モデルを 1 次元増やすだけで学習にかかる時間はかなりの増大が予想されることから前者の下半身だけのロボット作成が現実的であると考える。

7　結果と考察

本研究の構成はこれまでに解説したように、ソフトウェア、ハードウェア、そしてこれらの統一という三部から成り立っている。この章ではそれぞれの結果についてまとめていく。

まずソフトウェアでは二次元平面上の二足歩行モデルを用い、歩行のためのリズム生成器である CPG は van der Pol 方程式を用いた。そしてこれらを用いて Visual Studio 上の C++でプログラムし、視覚的に表示した。その際、画像を表示するプログラムが遅いという理由からリージョン機能を用いて再度プログラムし、ある程度快適なアニメーションを作り出すという工夫を行った。次にモデル上での歩行獲得をすべく、学習アルゴリズムの構築を試みた。歩行獲得へのアプローチとしては強化学習を用い、報酬には『腰の高さ・歩行速度を一定に保つ』というものに定めた。さらに、左右の腰の制御則を統一するという簡略化を試み、両足別の制御則と比べて半分の計算で処理が行えるようになった。同様に学習のための価値関数も、状態変数を通常の左右腰角度、左右腰角速度の 4 変数から、左右の平均角度、左右の平均角速度を用いることより 2 変数へと簡略化し、計算時間は簡略化前のものと比べて半分以下の計算で行えるようになった。これらを用いて二足歩行獲得実験を行った結果、約 80 試行前後という短い試行回数での歩行獲得に成功した。

ハードウェアを制御するためには PIC を用い、サーボモータへの指令である PWM 制御では PIC のタイマ機能を用いることによって最適なパルスを生成することに成功した。また CPG は PIC プログラムの制限から計算をすべて整数で行うという手法で生成した。

最後にソフトウェアとハードウェアの統一では、相互のデータ通信を行うことを試みた。ソフトウェアでは PC 上の AD/DA ボードを用いるためのプログラムを書き、PIC への入出力を行った。ハードウェアでは PC からの入力のために PIC 内部の AD 変換機能を用いた。また、PC への出力のために DA 変換チップ MAX525 を制御回路に取り付けた。これらの結果モデルで歩行実験を行ったときのように、実際のロボットでも歩行実験を行うことができる環境を実現した。

以上の結果、本研究の目的『強化学習を用いた二足歩行ロボットのための制御回路とソフトウェア環境の構築』は実現されたと言える。

本研究では学習アルゴリズムとして腰の高さ・歩行速度を報酬を用いて評価し、最適な CPG の振幅変数を見つけ出している。しかし、このときのパラメータ（制御則の学習率や価値関数の学習率等）は実験を行いながら手動で決定しているため、本研究で作り上げた環境を実際のロボットへ適応していく場合にはこのパラメータの再設定が必要となっている。この問題を解決するには、各パラメータを学習によって見つけ出すことが考えられるが、1 つパラメータを増やすごとに計算時間は指数的に増大していく。将来的な学習制御としては、この問題を解決することが課題の 1 つではないかと考える。

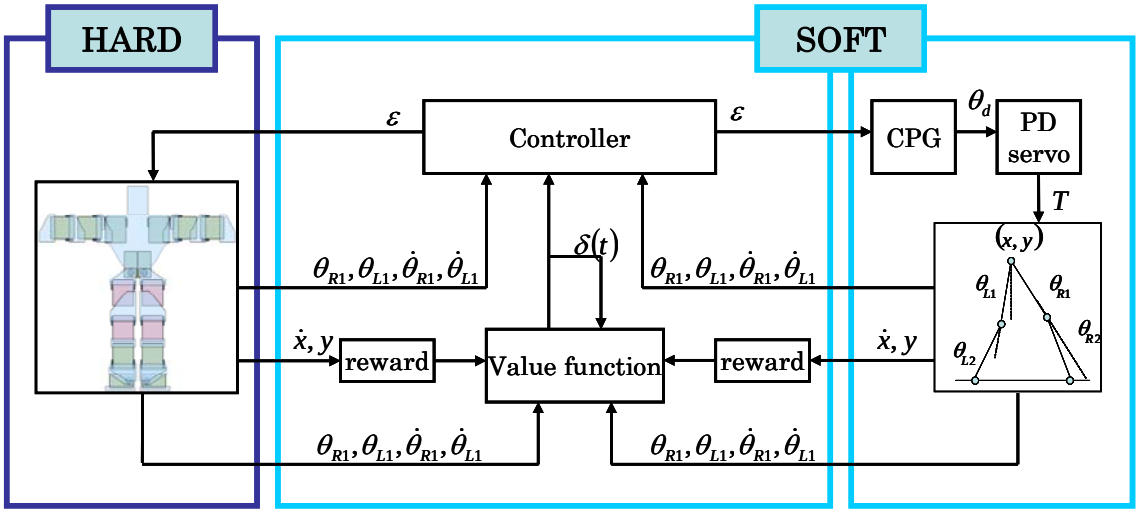


図 66　全体システム構成

本研究は、学習して制御則を自ら見つけ出すロボットに興味を持ち、その学習させる対象として制御困難な多自由度の二足歩行ロボットを選択して研究を進めてきた。そこで強化学習という学習制御方法の 1 つからアプローチを試みたが、学習制御という分野は未だ発展途上の部分が多く、現実に存在するものとしてはエンターテイメントロボット等の少ない分野でしか利用されていないのが現状である。この原因はいくつか考えられるが、主な原因として現状の学習制御は、試行錯誤して最適なものを見つけ出すというものであるため、失敗することも前提としているということが問題である。このため現実のものに学習制御を取り入れていくには想定される制御則に補助的なものとして学習制御を取り入れ、少しずつ補助の割合を増やしていくというのが学習制御発達の 1 番の近道ではないかと考える。

参考文献

[1] Taga, G., Yamaguchi, Y., and shimizu, H. (1991) "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," Biological Cybernetics, vol. 65, pp.147-159.

[2] Doya, K. (2000) "Reinforcement learning in continuous time and space," Neural Computation, vol. 12, pp.219-245

[3] 松原崇充, 森本淳, 中西淳, 佐藤雅昭, 銅谷賢治(2005) "方策こう配法を用いた動的行動則の獲得 : 2 足歩行運動への適用," 電子データ通信学会誌 D-Ⅱ, vol.J88-D-Ⅱ, pp.53-65

[4] Richard S. Sutton and Andrew G. Barto 著 三上貞芳, 皆川雅章 共訳, "強化学習" 森北出版株式会社 (2000)

[5] 後閑哲也著, "電子工作のためのPIC16F活用ガイドブック" 技術評論社 (2004)

[6] Kanamaru, T. (2006) "Analysis of synchronization between two modules of pulse neural networks with excitatory and inhibitory connections," Neural Computation, vol.18, pp.1111-1131.

[7] Tucker!著 "憂鬱なプログラムのためのオブジェクト指向開発講座" 株式会社翔泳社 (1998)

[8] 林晴比古著 "新 Visual C++ .NET 入門ビギナー編" ソフトバンクパブリッシング株式会社 (2005)

[9] 林晴比古著 "新 Visual C++ .NET 入門シニア編" ソフトバンクパブリッシング株式会社 (2005)

[10] 後閑哲也著 "PIC マイコンではじめる作って遊べるロボット工作" 株式会社技術評論社 (2003)

[11] 後閑哲也著 "C 言語により PIC プログラム入門" 株式会社技術評論社 (2002)

[12] 後閑哲也著 "8 ピン PIC マイコンではじめる作る、できる電子工作入門" 株式会社技術評論社 (2005)

[13] 松野研司, 中村秦, 柴田智広, 石井信 (2006) "CPG-Actor-Critic 法によるミミズ型ロボットの推進運動の獲得" 電子データ通信学会誌 信学技報 IEICE Technical Report NC2006-152(2007-3), pp.203-208

[14] 中村秦, 佐藤雅昭, 石井信(2004) "神経振動子ネットワークを用いたリズム運動に対する強化学習法," 信学論 (D-Ⅱ), vol.J87-D-Ⅱ No.3, pp.893-902

[15] 水戸部和久, 森直樹, 相田恒一, 那須康雄(1996) "非線形フィードバックによる二足歩行ロボットの制御" 日本

ロボット学会誌　Vol.14　No.8，pp.1194-1199
[16] 浅田稔，野田彰一，俵積田健，細田耕(1995) “視覚に基づく強化学習によるロボットの行動獲得” 日本ロボット学会誌　Vol.13　No.1，pp.68-74