遺伝的アルゴリズムを用いた干潟調査用の移動体制御システムの研究

修士学位論文

遺伝的アルゴリズムを用いた干潟調査用の移動体制御システムの研究

平成 30 年度

(2019 年 3 月)

東京海洋大学大学院

海洋科学技術研究科

海洋システム工学専攻

中村　圭

まえがき

本研究では，干潟[1]の環境に対応した海洋調査ロボットの開発を目的としている．干潟とは，陸域と水域を併せ持つ複合環境である．また，様々な生物が生息し多様な生態系を構築している．そのため，環境調査や研究が盛んに行われている．現在，環境調査機器としてはドローンや ROV(Remotely Operated Vehicle)など様々なものがあるが，干潟の様な特殊な環境に対応した調査機器はほとんどない．したがって，調査・研究は定点カメラなどを用いた静的観測法が主な手法となっている．仮に干潟に対応した調査機器があれば，調査対象の追跡や任意の地点での待ち伏せなどの動的観測が行えるようになり，調査・研究をより効率的に進められると言える．

そこで，本研究では干潟に対応した調査機器として，μ-ASV(Micro-Autonomous Surface Vehicle)を搭載した多足歩行型ロボットを提案する．この調査機器については上部がμ-ASV，下部が多足歩行型ロボットで構成されているものを想定している．多足歩行型ロボットは 6 脚 6 節を有する機体で，陸上では脚部を連続回転させることで歩行し，水域では脚部をフィン運動させることによって推進する．このように，多足歩行型ロボットであれば 1 つの移動機構で周囲の環境に対応した移動が行えるため，干潟の様な特殊な環境であっても自由な移動が可能だと考えられる．一方でμ-ASV については,水域において多足歩行型ロボットと分離し,水上と水中に分かれて同時並行で探査を行っていく．この時，μ-ASV は多足歩行型ロボットと陸上との洋上中継器としての役割も果たす．また，水域において多足歩行型ロボットを目標地点まで運ぶキャリヤーとしても使用する．本研究では，この調査機器をμ-ASV と多足歩行型ロボットに分けて考え，まずは多足歩行型ロボットについて開発を進めてきた．

初めに，多足歩行ロボットが陸上で安定した歩行が行える制御システムについて検討を行った．ここで，検討にはロボット開発支援ソフトウェアの ROS (Robot Operating System) [2],[3]および，3 次元動力学シミュレータの Gazebo[2]を使用する．ROS では，3D ロボットモデルやモデルの制御器の作成をおこない，Gazebo はシミュレーション環境の構築と物理運動計算に使用した．これらのシミュレーションに使用したソフトウェアの詳細については 2 章で示す．

多足歩行型ロボットモデルは，遊脚時と接地脚時で各関節の各加速度を制御することで安定な歩行を行う，また，この制御に必要なパラメータは機械学習の 1 手法である遺伝的アルゴリズム（GA：Genetic Algorithm）[4]によって適切なものを設計した．これまでの研究成果[5]より，適切な遺伝子の評価式を用いることで，平面上で安定した姿勢を保ちながら直進的に歩行することが確認されている．しかし，ここでは各関節に与える制御パラメータを全ての脚で共通のものとしていた．陸上以外の水域や沼地環境での動作を考慮すると，脚の動作にはより高い自由度が必要と言える．そこで，共通のものとしていた各関節に与える制御パラメータの一部を個別化し，動作の自由度を高くすることを検討した．また，ロボットが陸上でより自由な移動を行うために，旋回運動を行う方法についても同様に GA を用いて検討した．これらの検討の詳細については 3 章にて示す．

また，歩行ロボットモデルの運動制御手法として，非線形同期を使用した場合についても検討を行った．GA によって歩行に適した遺伝子（制御パラメータ）を作成するには多くの試行回数を要し，実時間で約 4 時間必要となる．さらに，水域や沼地などでの動作を考えた場合，流体運動が絡むため，物理計算により多くの時間を要することとなる．また，GA ではオープンループで歩行パターンを作成しているに過ぎないため，環境が変化すると再度設計し直す必要がある．そこで，自然界の多

脚生物の歩行に注目した．多脚生物は，非線形同期現象によって歩行を行っていることが知られており[7]，各脚を振動子として結合・同期させることで歩行していると考えられる．よって，脚全体の運動を非線形同期現象で表すことが出来る．これを利用し，非線形同期によって安定した歩行パターン作成する手法について検討を行った．この検討結果については 4 章に示す．

また，多足歩行型ロボットの実機作成も同時に進めている．これまでに，6 脚 6 節を有する歩行型ロボットの機構部分についてはほぼ作成が終わっている．現在は各脚に取り付けるモータの制御回路作成に取り組んでおり，制御回路側でのモータ制御プログラムの作成を主に行っている．この詳細については 5 章に示す．

次に，μ-ASV についても 3D モデルを作成し，同様に制御手法の検討を行った．現状の Gazebo では，水域のシミュレーション環境が十分に整っていない．そのため，μ-ASV の船体運動を陸上で疑似的に再現する方法として，全方位への移動が可能であるオムニホイールをシミュレーションモデルに使用することとした．ここでは，このモデルが任意の目標座標まで移動し，目標座標付近で定点保持を行う簡易的な DPS（Dynamic Positioning System）制御システムの作成を主な目的とし，制御手法としては，スライディングモード制御[6]を使用した．詳細については 6 章に示す．

目次

1. 緒言

干潟[1]は河川によって運搬された砂泥や有機物が堆積することによって形成され，無数のバクテリア，藻類，貝類やエビ類などの底生動物や魚類（特に稚魚）などが豊富に生息している．さらに，これらを餌としてガンやハクチョウなどの渡り鳥が飛来する．そのため，干潟には多種多様な生物たちによって複雑な生態系が構築されている．また，干潟に生息するバクテリアや底生動物が堆積した有機物を餌として消費することによって，河川等から流入する生活排水や産業排水の浄化も行われている．このような干潟の持つ特徴や役割などが注目され，環境調査および研究などが盛んに行われてきた．

調査・研究方法としては，干潟に対応した調査機器がほとんどないこと，またラムサール条約などによって環境保護が行われている場合があるなどの理由から，定点カメラなどによる静的観測法が主な手法となっている．一方で，干潟では‘飛来したハクチョウがアライグマによって襲われている’といった問題が現在発生しているが，この様な場合について静的観測法は適しているとは言えない．そのため，本研究では干潟環境に対応した調査機器として，μ -ASV を搭載した多足歩行型ロボット（図 1-1）を提案する．この調査ロボットを用いることで，観測対象の追跡や超至近距離での観測，任意の観測地点での待ち伏せなどの動的観測が行えるようになり，従来の観測法よりも効率的に調査および研究が進められるようになると考えている．

ここで，本研究で提案する調査ロボットを用いた干潟調査の概要図を図 1-2 に示す．このロボットは上部がμ-ASV，下部が多足歩行型ロボットで構成されている．干潟環境に適した調査機器としては，水域や陸域を自由にかつ安定な姿勢で移動でき，環境破壊を行わないような移動機構が必要である．干潟の陸域では，草地や岩地などの不整地が多いため，車輪型の移動機構では安定した移動が行えない．また，クローラ型であれば不整地に対する安定性は向上するが，地面を掘り返してしまうため環境を破壊してしまう恐れがある．そこで，本研究では移動機構として多足歩行型に注目した．多足歩行型であれば，陸上ではアリやカブトムシなどの多足生物が行う静的歩行方法(トライポッド歩行)を用いて安定した移動が行える．また，水域では脚部を振動翼としてフィン運動によって推進し，水域と陸域の中間地点となる沼地ではウミガメのように腹ばいで移動することができる．このように，多足歩行型ロボットであれば 1 つの移動機構で周囲の環境に対応した移動ができる．環境破壊の可能性についても，接地する時間が少なくなるため最小限にとどめられると考えている．そのため，主な探査は多足歩行型ロボットを用いて行っていく．μ-ASV については，水域において多足歩行型ロボットの補助的な使用を考えている．具体的には，水域において多足歩行型ロボットを任意の地点まで迅速に移動させるキャリヤーとして利用する．また，このロボットは水域で分離し，μ-ASV は水面上の，多足歩行型ロボットは水中の探査をそれぞれ同時並行に行う．この時，μ-ASV は地上と多足歩行型ロボットをつなぐ洋上中継器としての役割も兼ねている．

本研究では，このロボットをμ-ASV と多足歩行型ロボットの 2 つに分けて考え，まずは多足歩行型ロボットの運動制御システム開発から着手した．また，開発にはロボット開発支援 OS である ROS[2],[3]と 3 次元動力学シミュレータである Gazebo[2]を使用して 3D シミュレーションモデルを作成し，これを用いて運動制御システムの検討を行った．また，並行して多足歩行型ロボットの実機の作成も現在行っている．μ-ASV についても同様で，シミュレーションモデルが任意の目標地点まで移動するような制御手法の検討について主に行った．したがって，本論文ではμ-ASV および多足歩行型ロボットの

運動制御システムについて，それぞれのシミュレーション結果と考察を示す．また，多足歩行型ロボットの実機の作成経過についても示す．

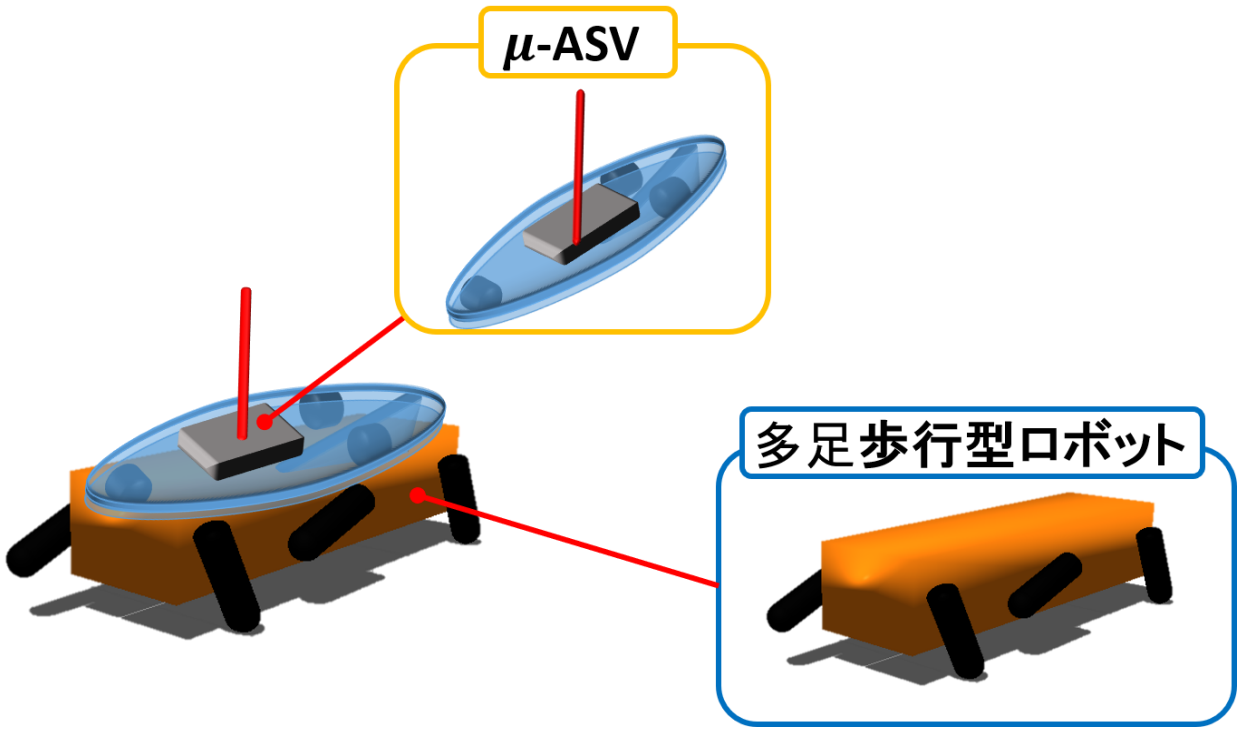

図 1-1 干潟調査用ロボットのモデルイメージ図

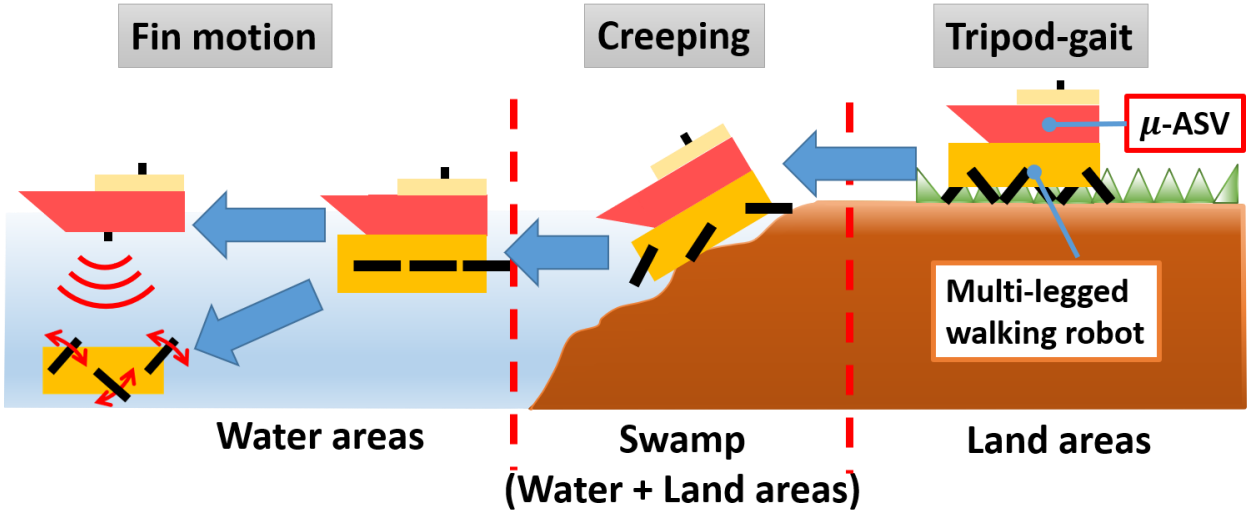

図 1-2 干潟調査概要

2. シミュレーションソフトウェア

本章では，運動制御システムの開発に用いたROSおよびGazeboの概要について，2.1節に示す．また，ROSにおける制御システムの通信方式については2.2節に，シミュレーションモデルに実装した制御器の概要については2.3節にそれぞれ示す．

2.1. シミュレーションソフトウェアの概要

ROSおよびGazeboは共にLinux上で使用されるオープン規格のソフトウェアである．本研究では，Linux OSとしてUbuntu16.04LTSを使用し，現在ユーザー数が最も多いROS Kinetic KameとGazebo 7.0の組み合わせを使用した．ROSの概要については2.1.1節に，Gazeboの概要については2.1.2節にそれぞれ示す．

2.1.1. ROS

ROS[2],[3]とはRobot Operating Systemの略称である．名称にOSと付いているものの，ソフトウェアの役割としてはミドルウェアに近いと言える．開発状況はGitHubで公開されており，動力学シミュレータやセンサ情報の可視化ツール，自動地図生成モジュールなど数千ものパッケージが利用可能となっている．ROSの機能の構成要素は以下の4つに大別できる[3]．

1) 通信ライブラリ

HTTPを元にしたXML RPGで通信ライブラリを提供している．用途に応じて一対多や多対多などの通信形態を選択できる．通信の仕組みはpublish/subscribe型と呼ばれるもので，これを用いた通信の詳細は2.2節にて示す．

2) 開発・操作ツール

代表的なものとしては,可視化ツール(ビューア)のRVizや3次元動力学シミュレータのGazeboなどが挙げられる．この他にも，パッケージ間の依存関係を自動的に解決できるrosdepや複数のパッケージを一括してコンパイル可能なcatkinビルドシステムなどがある．

3) 高機能ライブラリ

多関節ロボット動作計画ツールなどが挙げられる．これによって，逆運動学の数式や確率的探査アルゴリズムの記述を行わなくとも，数行のコードを記述するだけで，多関節ロボットの作業をプログラムすることができる．

4) エコシステム（開発コミュニティ）

専用の質問用webサイト「ROS wiki」が存在し，全世界のユーザーと情報を共有できる．

また，ROSは分散型処理システムであり，各種ロボット向けの基本機能のうち，汎用性のある部分をノード（Linuxにおけるプロセスに相当する）と呼ばれる機能モジュールで表現しており，これらのノードを組み合わせることで，より複雑な制御機能を分散的に実現している．例えば移動ロボットのシステムを構築する場合，センサドライバ，センサデータ処理，障害物の検出，モータ駆動，エンコーダ入力，ナビゲーションなど，動作に必要な処理を細分化し，それぞれの処理を実装したノードを組み合わせることでシステムを構築する．本研究では，主にシミュレーションモデルの作成とモデルの制御シミュレーションにROSの機能を用いた．

2.1.2. Gazebo

Gazebo[2]とは，OSRF（Open Source Robotics Foundation）によって開発された，物理エンジンを搭載したロボットアプリケーション開発を目的とする 3 次元動力学シミュレータである．Gazebo は ROS の推奨する基本シミュレータの 1 つでもあるため，ROS との連携が充実しており，ROS への導入が容易となっている．そのため，ROS を基準とし Arduino などのマイコンを介することで，物理シミュレータである Gazebo と実機を自在に切り替えることも可能である．例えば，実機ではセンサなど一部のみを使い，そのセンサからの入力データで Gazebo 内のシミュレーションモデルを動作させる．あるいは Gazebo 内のセンサデータを入力とし実機のロボットを駆動させるといった実機とシミュレータとの混在シミュレーションが可能となる．Gazebo の特徴としては以下のもの[2]があげられる．

1）様々な物理エンジンによる動力学シミュレーション

ODE，Bullet，Simbody，DART などの多くの物理エンジンが選択できるようになっている． 本研究では物理エンジンとして ODE を使用した．

2）高度な 3 次元グラフィック機能

3 次元グラフィックスエンジンとして OGRE（Open-source Graphics Rendering Engines）を採用し，高度なレンダリング機能を提供している．

3）多様なセンサ

カメラ，接触センサ，力-トルクセンサなど，多くの内界・外界センサを再現できる． さらに，検出されたセンサデータへのノイズの付加も可能である．

4）プラグインによる機能拡張

プラグイン開発のための API が提供されているため，ユーザーはロボットやセンサ，制御プログラムなどの独自のプラグインを開発および導入することが容易にできる．

5）多様なロボットモデル

PR2，Pioneer2 DX，iRobot Create，TurtleBot など市販の多くのロボットが，Gazebo のモデルファイルである SDF（Simulated Description Format）形式で提供されている． また，独自のロボットを作成し Gazebo に追加することも可能．

6）リモートサーバー機能の提供

ソケットベースのメッセージパッシングであるグーグルプロトバッファ（Protbufs）を利用することで，リモートサーバーでもシミュレーションを実行することが可能となっている．

7）コマンドラインツール

豊富なコマンドラインツールを利用して，シミュレーションの状態を確認および制御することができる． 例えば，後述するノード間における通信状態などをグラフ化するツールなどがある．

本研究では，Gazebo によってシミュレーション環境の作成，およびシミュレーションモデルの物理運動計算，状態量の取得を行う．また，Gazebo の座標系の定義は以下の図 2-1 のように右手系となっている．そのため，本研究でも Gazebo の座標系に準じてモデルの作成や制御システムの設計を行った．図 2-1 中のモデルについては 3 章で用いる歩行ロボットモデルであり，詳細については 3 章で示す．また，赤い三角マーカについてはモデルの前方方向を示している．このように，本研究ではモデルの前方を x 軸＋方向としてモデルの作成を行った．

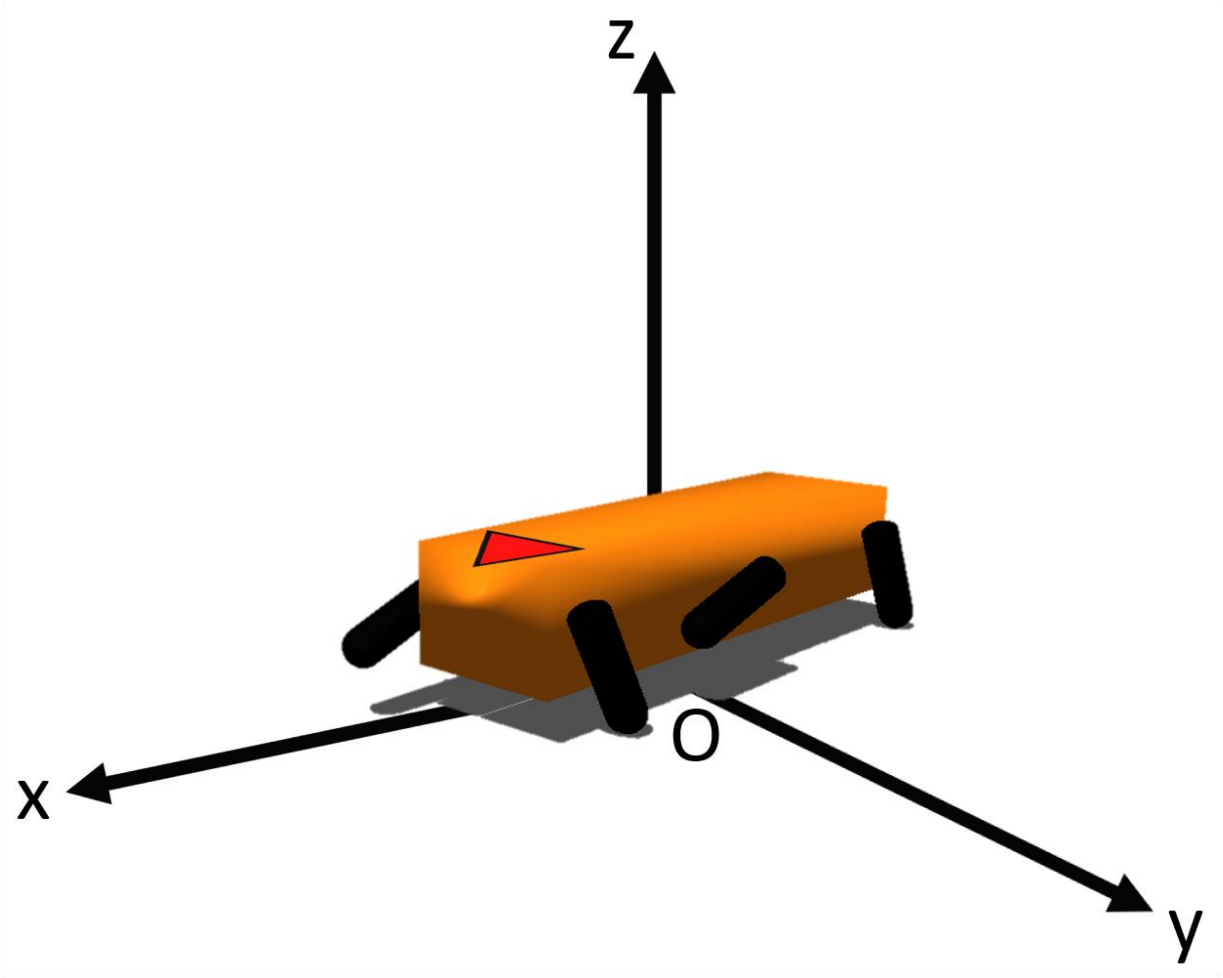

図 2-1 Gazebo における座標系の定義（右手系）

2.2. ノード間の通信方式

ノード間で通信を行うためには，ノード間をつなぐパスの役割を果たすトピックと，マスターと呼ばれるサーバ（ネームサーバ）が必要となる[2]．マスターとは，ROS のノードやトピックの管理を行うものであり，これを利用することで異なるコンピュータ同士の通信も実現することが出来る． マスターの仕組みは ROSCORE というプロセスにより提供され，ROSCORE ではノードやトピックの名前の登録，ノードの URI およびトピックの検索，トピックに対する購読者（Subscriber，データの受信を行うノードを示す）の参加通知などを提供する． そのため，ROS システムの中で，ROSCORE は必ず 1 つは起動させておく必要がある．ROS における通信手段は複数存在するが，本研究では最も基礎的なトピックを用いた通信を行う．

ノード間の通信では，データを送信することを配信（publish）と呼び，配信を行うノードを配信者（Publisher）と呼ぶ．同様に，データを受信することを購読（subscribe），購読を行うノードを購読者（Subscriber）と呼ぶ． トピックを用いたノード間の通信[5]は以下の手順で行われ，ノード，トピック，マスターの関係は図 2-2 のようになる．

1) データを送信するノード（配信者）は，マスターに自分のトピック情報（トピック名（name），ノードのピアアドレス（afo:1234））を登録する．
2) データの受信を行うノード（購読者）は，マスターによって目的のトピック(name)を検索する．
3) 目的のトピック(name)がすでに登録されている場合，配信者ノードのピアアドレス(afo:1234)を取得する
4) 購読者は，取得したピアアドレスを利用して配信者ノードと接続し，データの受信を行う

また，実際のトピック通信の例として本研究で作成した多足歩行型ロボットモデルのノードグラフを図 2-2 に示す．ここで，ノードグラフとはノード間の通信のやり取りを可視化したものであり，ノードを四角で，トピックを楕円形で示す．図 2-3 では，各ノードは制御プログラム，各関節の制御器，シミュレーションモデルの 3 つのグループに大きく分かれている．

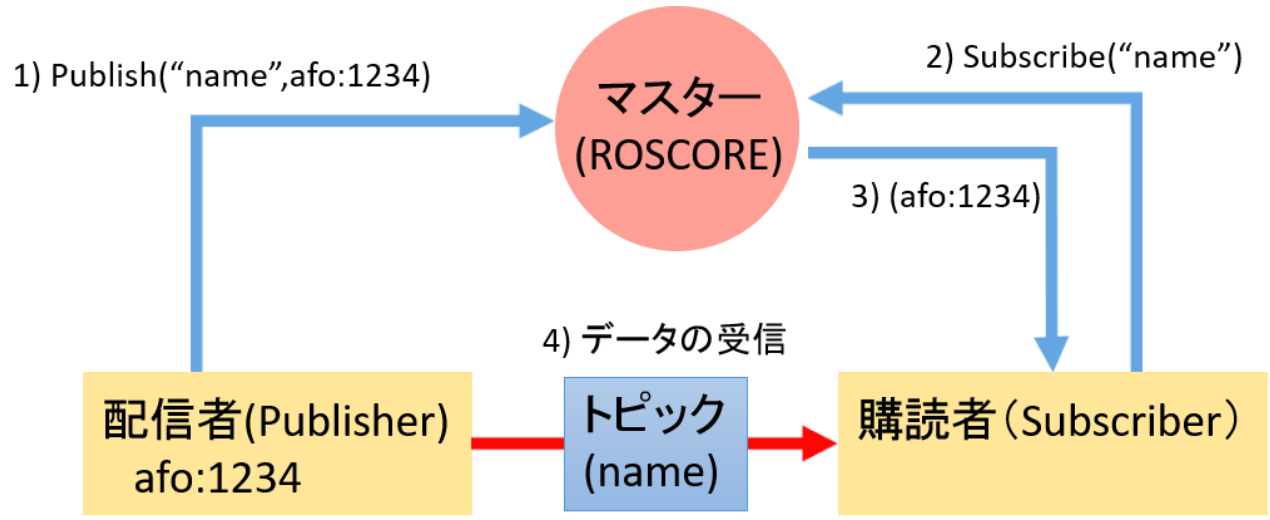

図 2-2 トピック通信の概略図

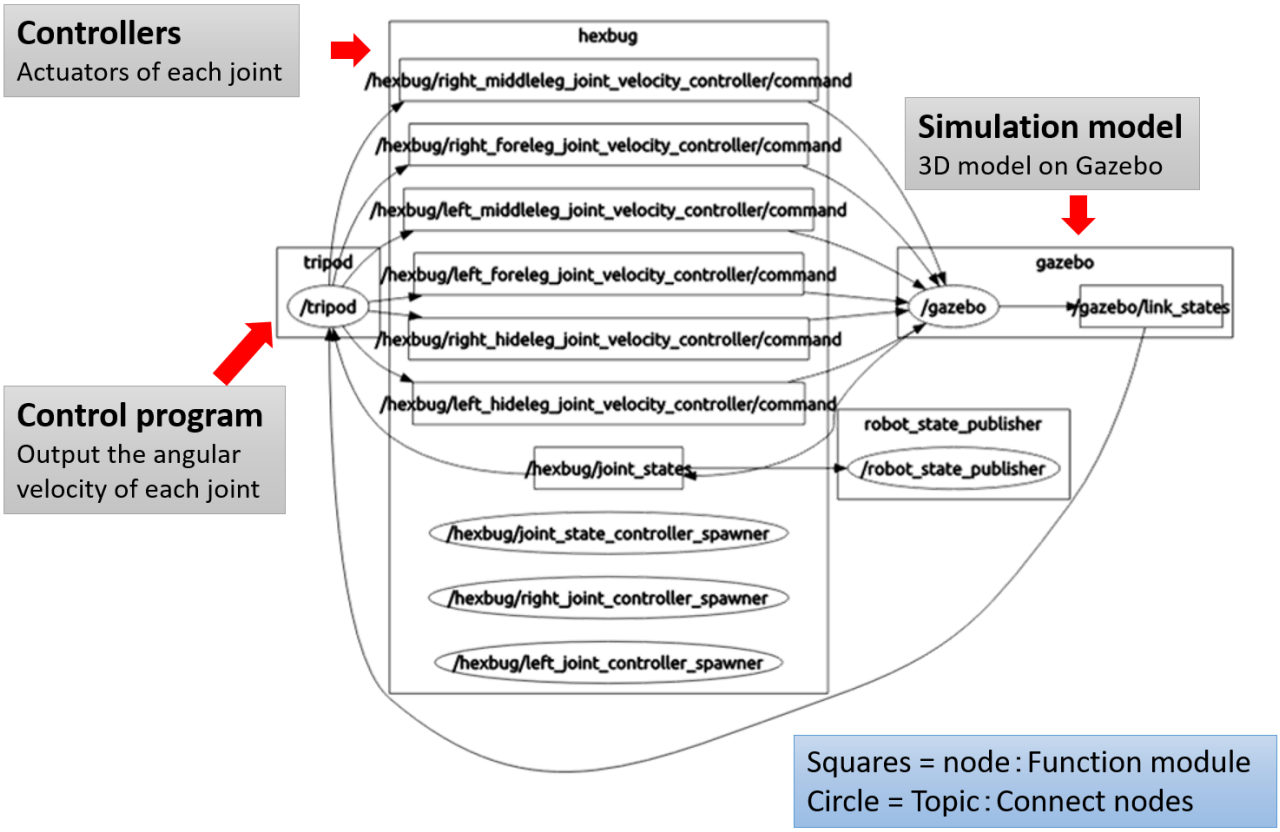

図 2-3 多足歩行型ロボットモデルのノードグラフ

2.3. 制御器（ROS Controller）

ROS Controller[8]とは，モデルの関節（joint）を個別に指定して実装することが出来る制御器である．このROS Controllerは標準のライブラリとして，角度制御を行うjoint position controllerや角速度制御を行うjoint velocity controllerなどが数種類あるほか，オリジナルのコントローラを製作することも可能となっている．また，ROS Controllerは図2-4のようなツリー構造になっている．複数のcontrollerをまとめたものをcontrollersと呼び，さらにcontrollersをまとめたものをController Managerと呼ぶ．

ROS ControllerとGazeboの通信系統を図2-5に示す．図2-5より，実装されたROS Controllerは対応するhardware_interfaceを介して，Gazebo上のモデルの関節に指令値を送る．また，joint_state_controllerを用いることでhardware_interfaceを介してGazebo上のモデルから関節の情報を受け取ることも出来る．このように，ROS Controllerとjoint_state_controllerを実装することで，シミュレーションモデルの関節の制御が行える．

本研究では，図2-4に示したeffort_controllersおよびjoint_state_controllerを多足歩行ロボットとμ-ASVのシミュレーションモデルに実装する．

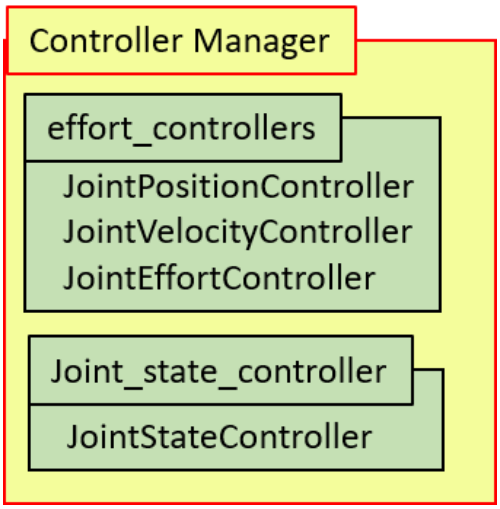

図 2-4 ROS Controllerのツリー構造

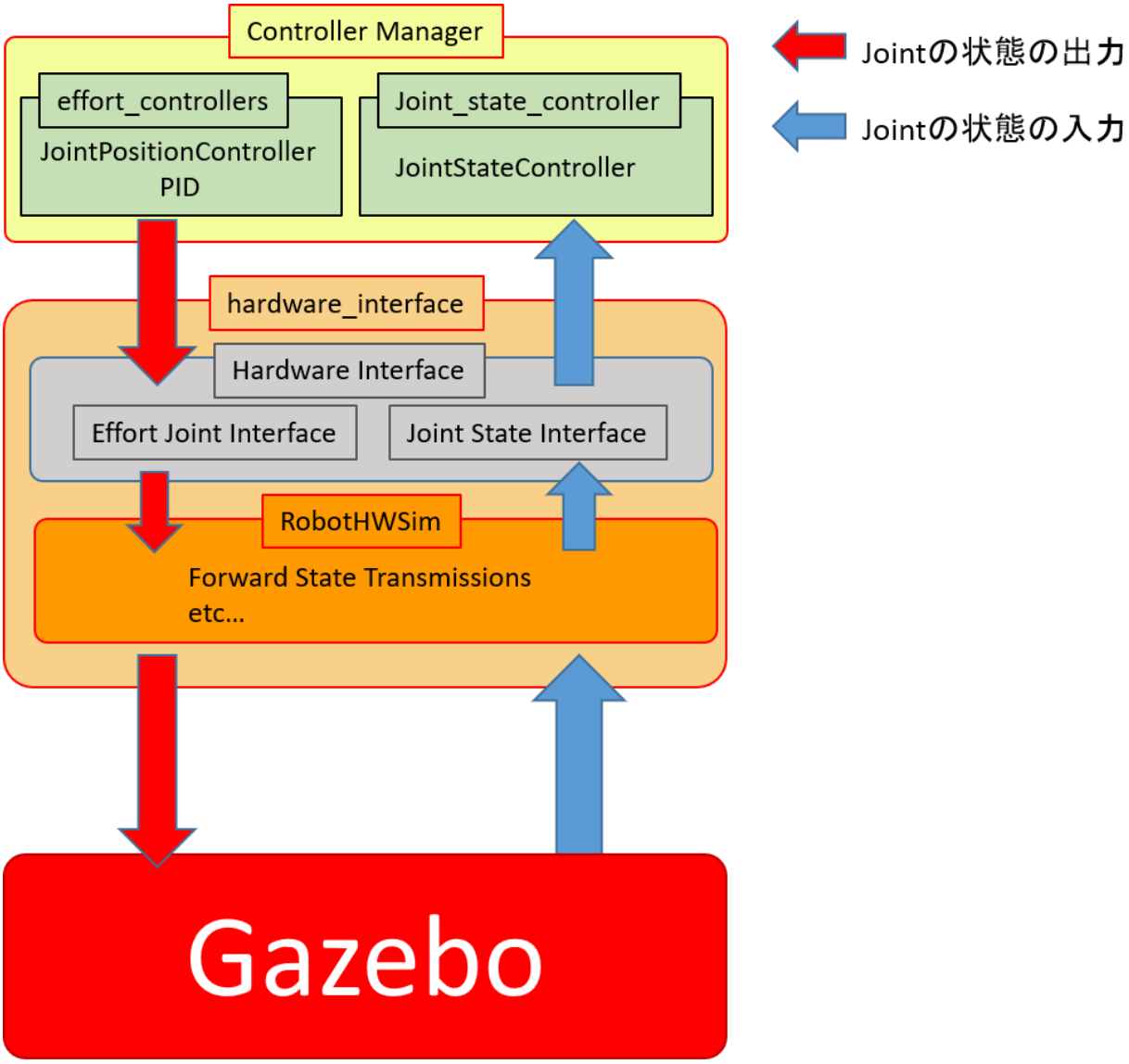

図 2-5 ROS Controller と Gazebo の通信系統図

3. 多足歩行ロボット

本章では，主に多足歩行型ロボットのモデル同定から始め，安定した歩行パターンの作成方法の検討の詳細について示す．ここで，シミュレーションモデルの構成については 3.1 節に示す．また，歩行パターンの作成手法については遺伝的アルゴリズム（GA：Genetic Algorithm）[4]を使用する手法と非線形同期現象を利用する手法の 2 つについて検討した．本章では，GA を用いた歩行パターンの作成方法について示す．また，Gazebo 上での水域環境の再現方法について検討した結果を 3.4 節に示す．非線形同期による手法については 4 章に示す．

3.1. モデル構成

多足歩行型ロボットが干潟の様な不整地を移動する場合，脚の数が多いほど安定性および冗長性は増加していくが，同時に制御するアクチュエータの数も増加していく．これを踏まえて，安定した歩行を行うために最低限必要となる足の数を検討し，自然界で静歩行を行っているアリやカブトムシなどの昆虫と同じ 6 脚型とした．また，アリやカブトムシなどの 6 足生物は通常，1 脚につき 3 節，計 18 節の関節を有している．しかし，本研究で提案するロボットは沼地や水中などの故障しやすい動作環境を考慮して，1 脚につき 1 節，計 6 節の関節を有するものとした．このように，駆動させるアクチュエータの数を最小限にすることで構造を単純化することができ，故障時の修理を容易にし，さらに製作コストを抑えることができる．

また，6 脚 6 節の多足歩行型ロボットについては，McGill University の Chris Prahacs らによって，本研究で提案する多足歩行型ロボットと同型である RHex（図 3-1）の先行研究[9]が行われている．この RHex は本研究で提案するものとは異なり，水中で動作する場合には脚部をフィン形状のものに変える必要がある，しかし，陸上においては，脚部を連続回転させることによって安定した歩行が行えることが確認されている．よって，シミュレーションモデルの基本構造は RHex を元に設計した．

作成したシミュレーションモデルを図 3-2 に，モデル構成図を図 3-3 にそれぞれ示す．今回作成するシミュレーションモデルは干潟での移動のみを目的としている．そのため，直方体の胴体部と足先が半球状となっている円柱形の脚部で構成される単純な物とした．また，モデルの腹部四隅には，腹部を地面に付けた際のセンサとして球オブジェクトを搭載している．モデルのサイズは、全長 1.5[m] および全高 0.45[m]（脚を伸ばした状態）、総重量は 10.2[kg]として設計した．モデルの各関節には角速度制御を行うために，ROS Controller の内 joint velocity controller を実装している．GA を用いたシミュレーションでは，このモデルの胴体中心部（図 3-3 の MC；Model Center）の移動量，および腹部の球オブジェクトの地面への接触回数を用いて遺伝子の評価を行う．また，図 3-2 における三角マーカについてはモデルの前方方向を表すために追加で記述しているだけであり，実際のシミュレーションモデルには存在しない．以降はこのモデルの前方を三角マーカで示すこととする．このモデルの構成プログラムについては付録 1 プログラムリストの List.01 に，各関節の制御コントローラについては List.02 にそれぞれ示す．

図 3-1 Rugged-RHex platform

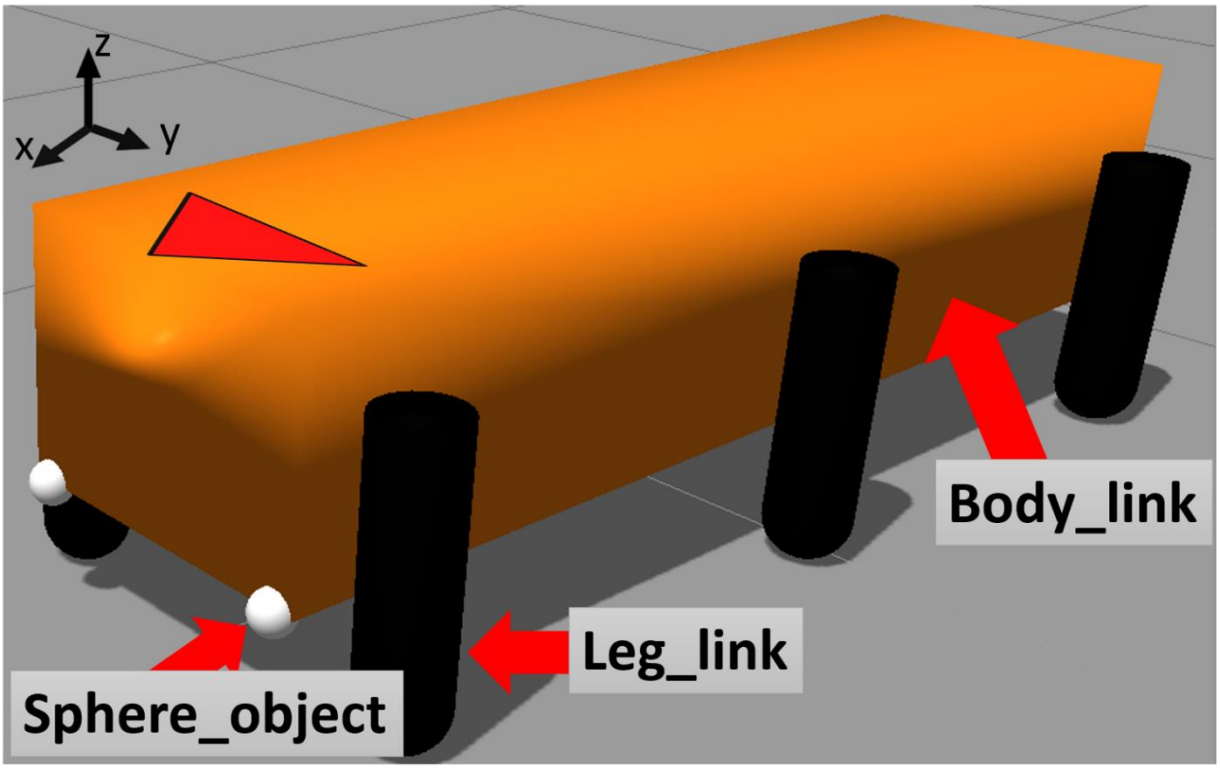

図 3-2 多足歩行型ロボットの 3D シミュレーションモデル

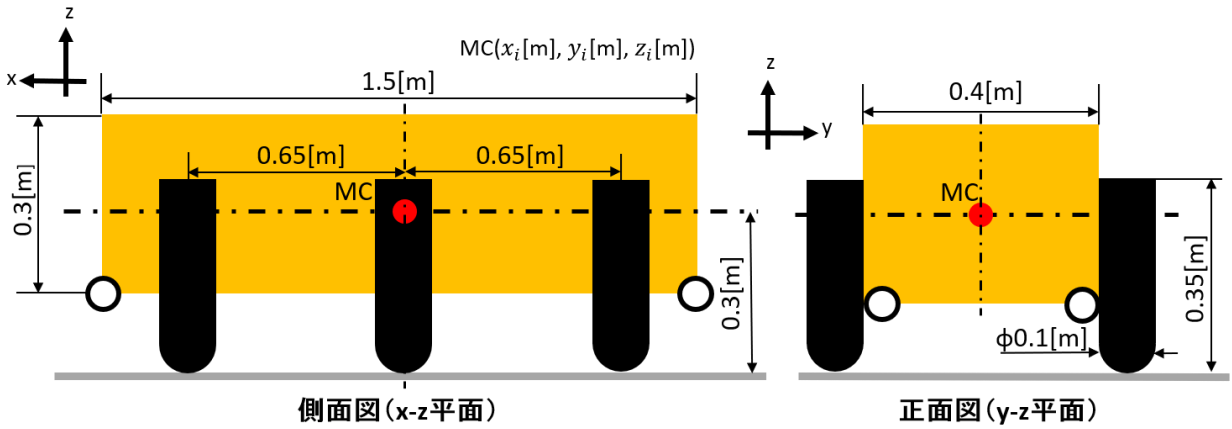

図 3-3 歩行ロボットモデルの構成図

3.2. GAによる静歩行パターンの作成

3.1節，図3-2の多足歩行型ロボットモデルが，安定した姿勢で移動するための制御パラメータをGA によって作成する．ここでは，モデルの各関節を一方向に連続回転させ，角速度制御を行うことにより安定した姿勢での歩行を行う．また，本研究ではモデルの安定な姿勢での歩行として，腹部を地面に付けずに歩行することを目指した．3.2.1節ではモデルが前進する場合について検討し，3.2.2節では旋回運動する場合について検討した．また，これらの検討結果のまとめは3.2.3節に示す．

3.2.1. 前進する静歩行パターンについて

本節では，モデルが前進する静歩行パターンについて検討する．初めに，3.1節の図3-2のモデルが確実に安定した歩行ができることを確かめるため，トライポッド歩行のみを行う制御パラメータの作成を行った．次に，将来的に沼地や水域で動作することを考え，各関節の動作自由度を増加させた場合について検討した．

3.2.1.1. トライポッド歩行のみの遺伝子へ進化

まずは，図 3-2 のシミュレーションモデルが平地でトライポッド歩行を行えることを確認するため，GA によって必要な制御パラメータの設計を行った[5]．図 3-4 にトライポッド歩行の概要図を示す．トライポッド歩行では，基準脚(A グループ)と基準脚に対して位相差：dt[s]を持つ位相差脚(B グループ)があり，これを交互に動かすことによって重心が接地脚面積の中に常に入るように歩行する手法である．ここで，本研究ではBグループに与える位相差：dtは，脚の動き出しを遅らせる待ち時間[s]とした．本制御では図3-4のように制御に必要なパラメータは全ての脚で共通のものとし，図3-5に示す概要図のように定義した．図3-5において，θは接地脚領域を決定する角度[deg]，ω_1は接地脚時の角速度[rad/s]，ω_2は遊脚時の角速度[rad/s]である．加えて，基準脚に対して位相差を持つ脚に与えるdtの計4つのパラメータを遺伝子に見立ててGAにより設計する．遺伝子の評価式には以下の(1)式を用い，適応度：fが最大化される遺伝子（制御パラメータ）の作成を行う．

$$\begin{cases} f = a(x_{end} - x_{start})^2 + b\dfrac{1}{\sum {y_i}^2} + c\sum {z_i}^2 + d\dfrac{1}{penalty} \\ penalty = \sum {p_i}^2 + \sum {p1_i}^2 + \sum {p2_i}^2 + \sum {p3_i}^2 + \sum {p4_i}^2 + 1 + p_x \\ a = 10,\ b = 0.01,\ c = 0.1,\ d = 1 \times 10^3 \end{cases} \tag{1}$$

ここで，(1)式の右辺第一項から第三項まではモデル胴体中心部の x，y，z 方向の移動量についての評価である．右辺第一項のx_{end}はモデルの最終移動地点の x 座標であり，x_{start}はモデルの移動開始地点の x 座標である．右辺第四項は，モデルの腹部および胴体四隅の球オブジェクトが地面に接触した時($p_i, p1_i \sim p4_i$)，またはモデルが一定以上前方へ進まなかった際に発生するペナルティ(p_x)として設定している．ここで，(1)式では安定な姿勢で歩行する遺伝子へ進化するように，右辺各項の重みは右辺第四項：dを一番重く設定した．また，この時のGAシミュレーション用プログラムは付録1プログラムリストのList.03に示す．

世代数50，個体数100としてGAシミュレーションを行った．作成した制御パラメータを表3-1に示す．また，この表3-1のパラメータを用いて10秒間の歩行試験を行った．この歩行試験時のモデル移動軌跡を図3-6に，各脚の動きをsin波形で示した図を図3-7に，胴体中心部の傾きの時間変化

を図 3-8 にそれぞれ示す．ここで，図 3-6 において，Gazebo の座標系は 2.1.2 節の図 2-1 のようにモデルの前後方向が x 軸で，左右方向が y 軸となる．そのため，モデルが前進する方向は x 軸+方向となる．図 3-7 より，モデルは基準脚（A グループ）と位相差を持つ脚（B グループ）の 2 組を交互に動作させる事によって全ての脚が遊脚となる瞬間が無く歩行できていることが確認できる．また，モデル胴体の傾きが無く，安定して前進する歩行が行えていることが確認できた．

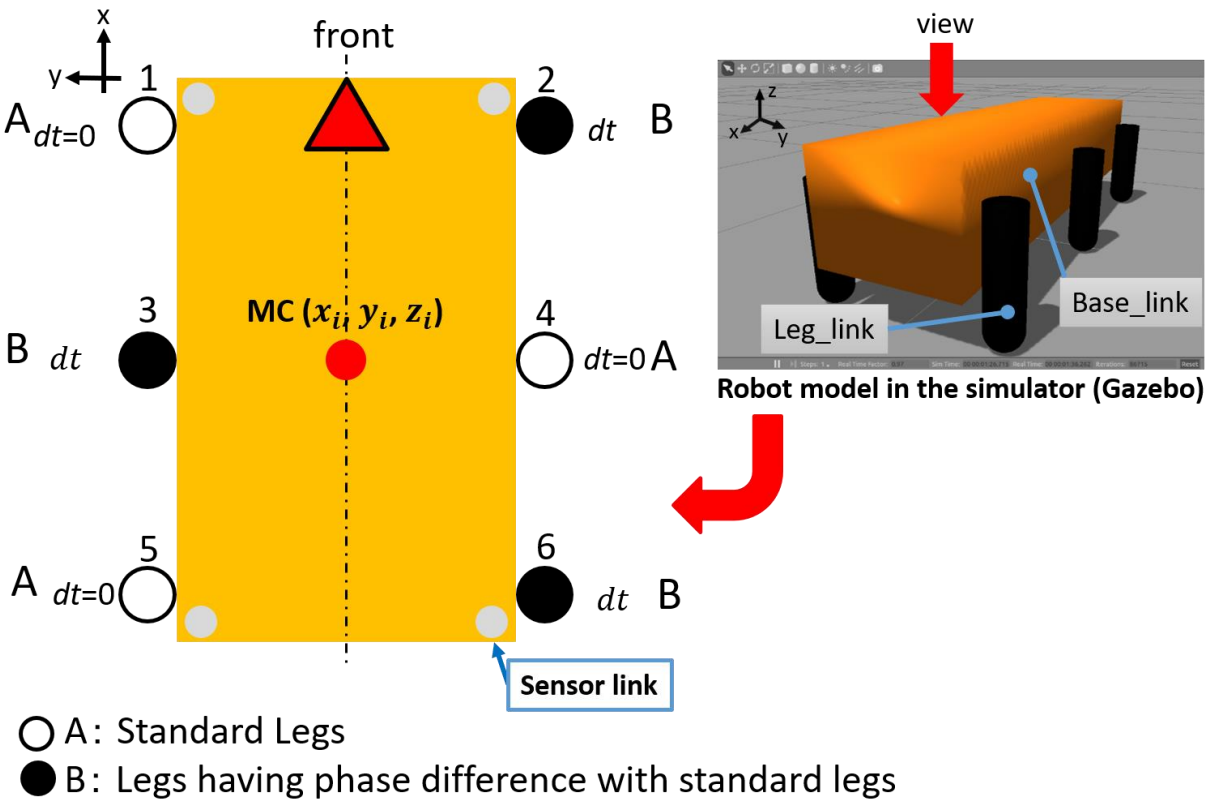

図 3-4 トライポッド歩行の概要

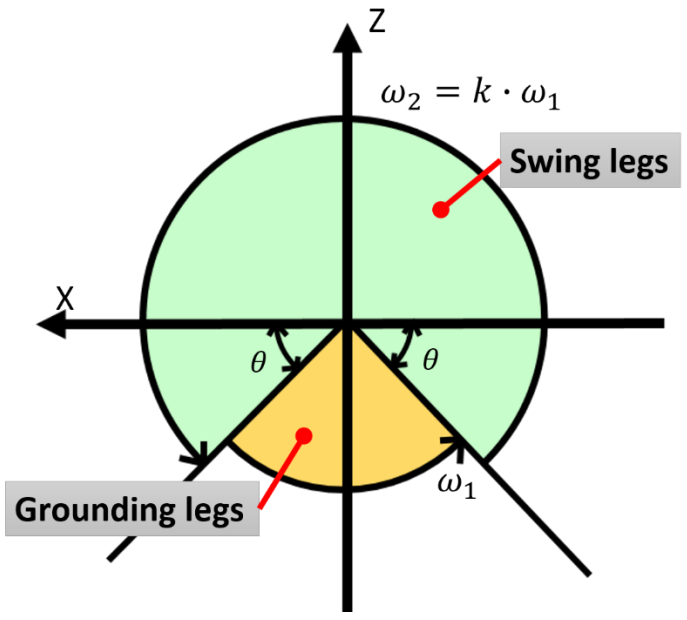

図 3-5 角速度制御パラメータの定義

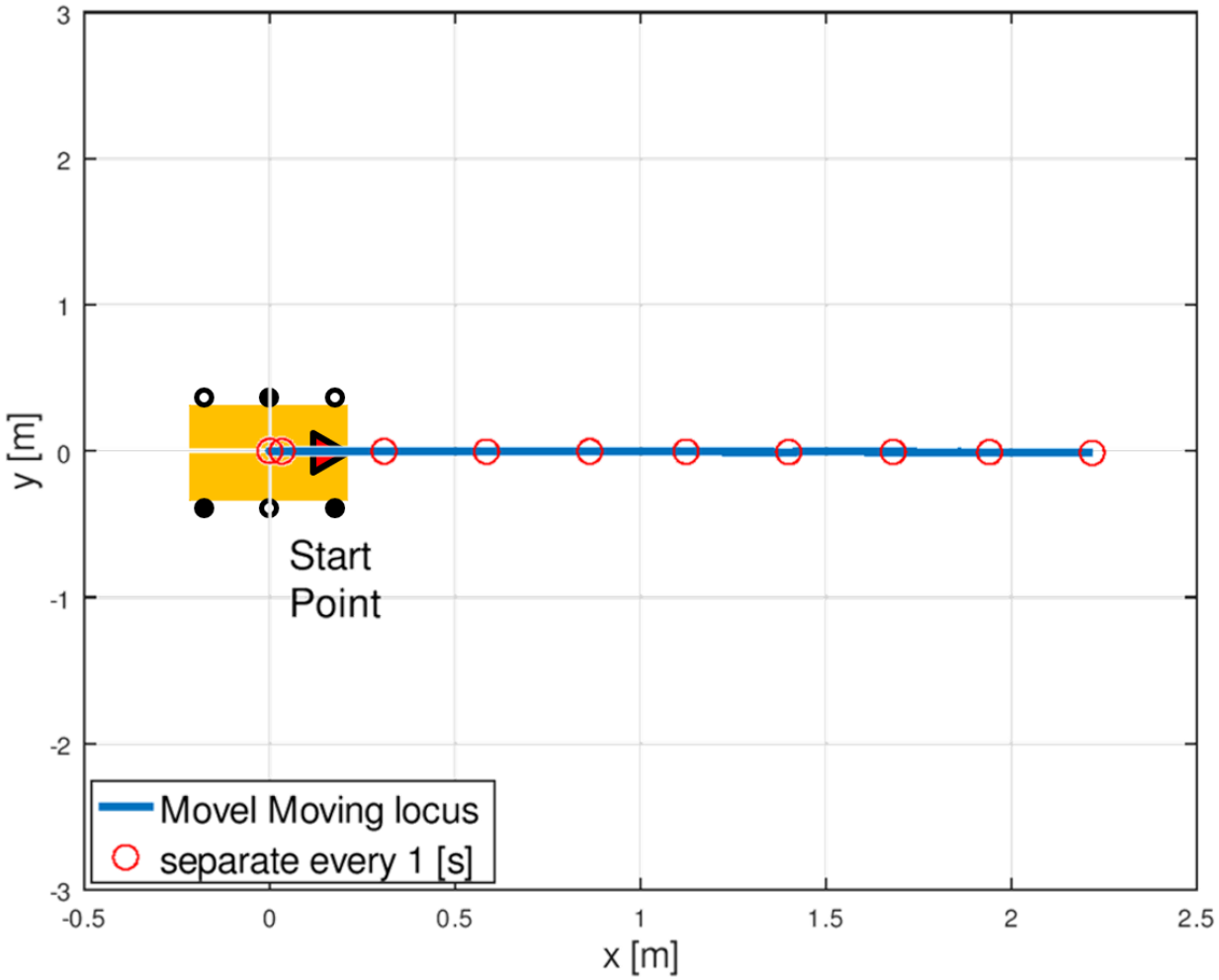

図 3-6 モデルの移動軌跡

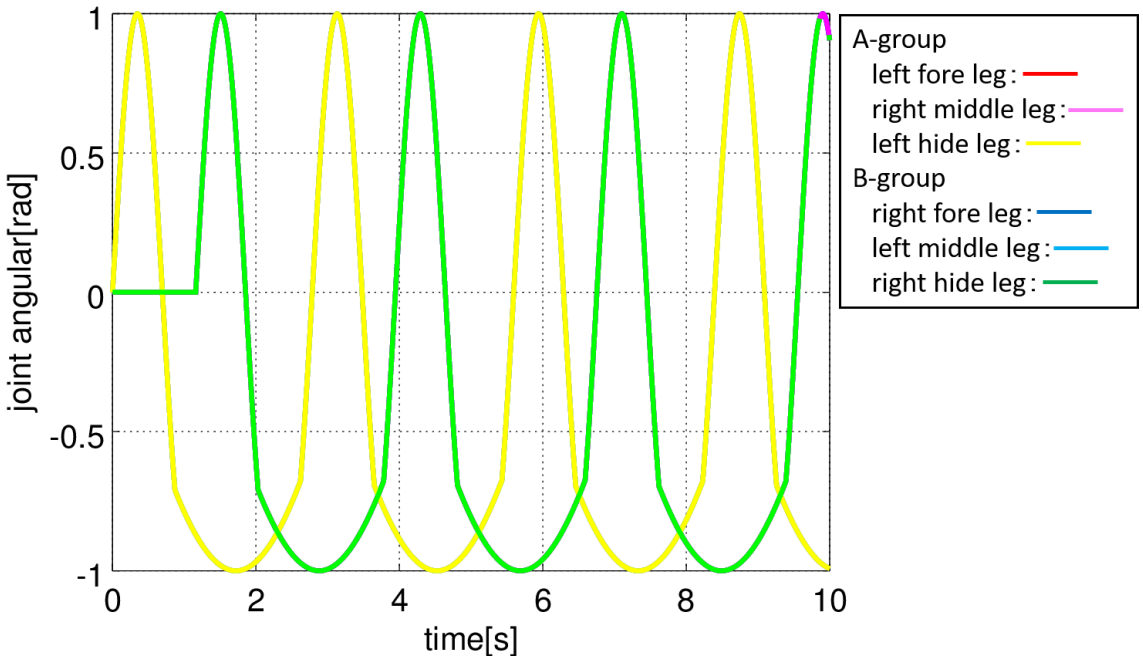

図 3-7 トライポッド歩行時の各脚の動作を示す sin 波形

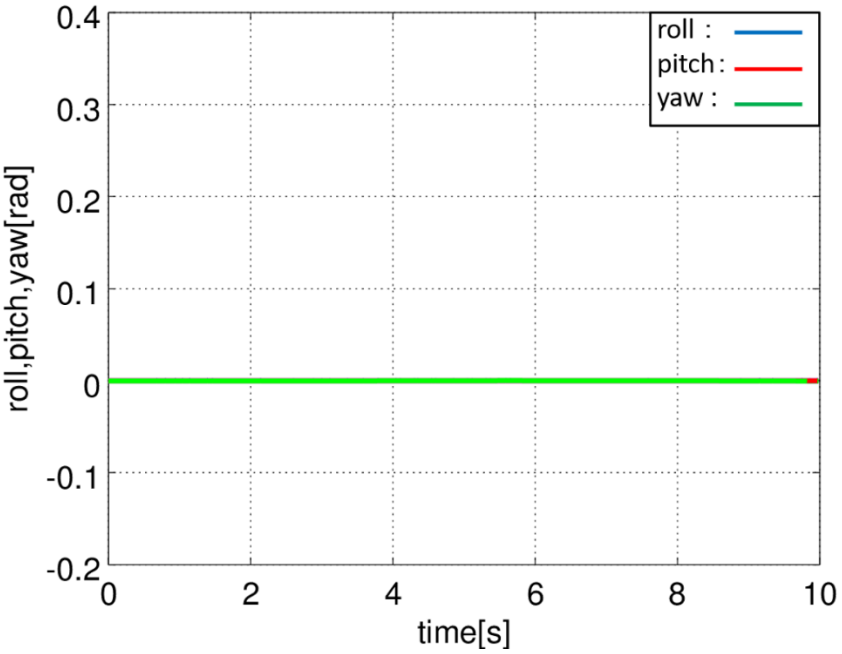

図 3-8 トライポッド歩行時の胴体の傾きに関する時間変化量

表 3-1 GA によって作成したトライポッド歩行の制御パラメータ

θ：接地脚域を定める角度[deg]	42.89
ω_1：接地脚時の角速度[rad/s]	0.92
k：遊脚時の角速度の係数[-]	4.92
dt：位相差[s]	1.66

3.2.1.2. 動作自由度の増加

3.2.1.1 節では，遺伝子がトライポッド歩行を行うものに進化するように，脚部に与える位相差dtは全ての脚で共通としていた．しかし，モデルが沼地や水域で動作することを考えた場合，動作の自由度は高くする必要がある．そこで，各脚の動作の自由度を高くするため，各脚に与える位相差（待ち時間）dt[s]を個別に設定し，各脚の動き出すタイミングをバラバラにした場合について検討した．

遺伝子の評価式は 3.2.1.1 節の(1)式を使用し，より静歩行のパラメータへと進化するように右辺第四項の重み:dのみ$d = 2 \times 10^6$に変更した．各脚に与える位相差は，図 3-9 に示すように左前脚を基準脚($dt_1 = 0$)として，各脚に位相差dt_1からdt_6を設定し，GA によって静歩行を行う制御パラメータの作成を行った．(1)式を用いて世代数を 30，個体数を 100 に設定してシミュレーションを行った．シミュレーションに使用したプログラムを付録 1 プログラムリストの List.04 に，GA で作成した制御パラメータを表 3-2 に示す．また，表 3-2 のパラメータを用いて 10 秒間の歩行試験を行った．この歩行時のモデル移動軌跡を図 3-10 に，モデル胴体の傾きを表す roll，pitch，yaw 角の時間変化量を図 3-11 にそれぞれ示す．

表 3-2 のパラメータを用いて歩行させた結果，モデルは前方に直進するものの胴体を大きくねじるような動歩行に近い進化を行っていた．また，腹部を地面に接触させながらの歩行となっていた．図 3-11 より roll，pitch，yaw 角は最大で約 0.3[rad]となっている．静歩行時の胴体の傾きは図 3-8 のように 0[rad]となるため，本シミュレーション結果は胴体を大きく傾けながら歩行を行っていると言える．これは，各脚に位相遅れを設定したため遺伝子の進化における自由度が増したためだと考えられる．そこで，この進化を抑制し，遺伝子が静歩行を行うもののみに進化するためには，モデル胴体の傾きについても評価する必要がある．

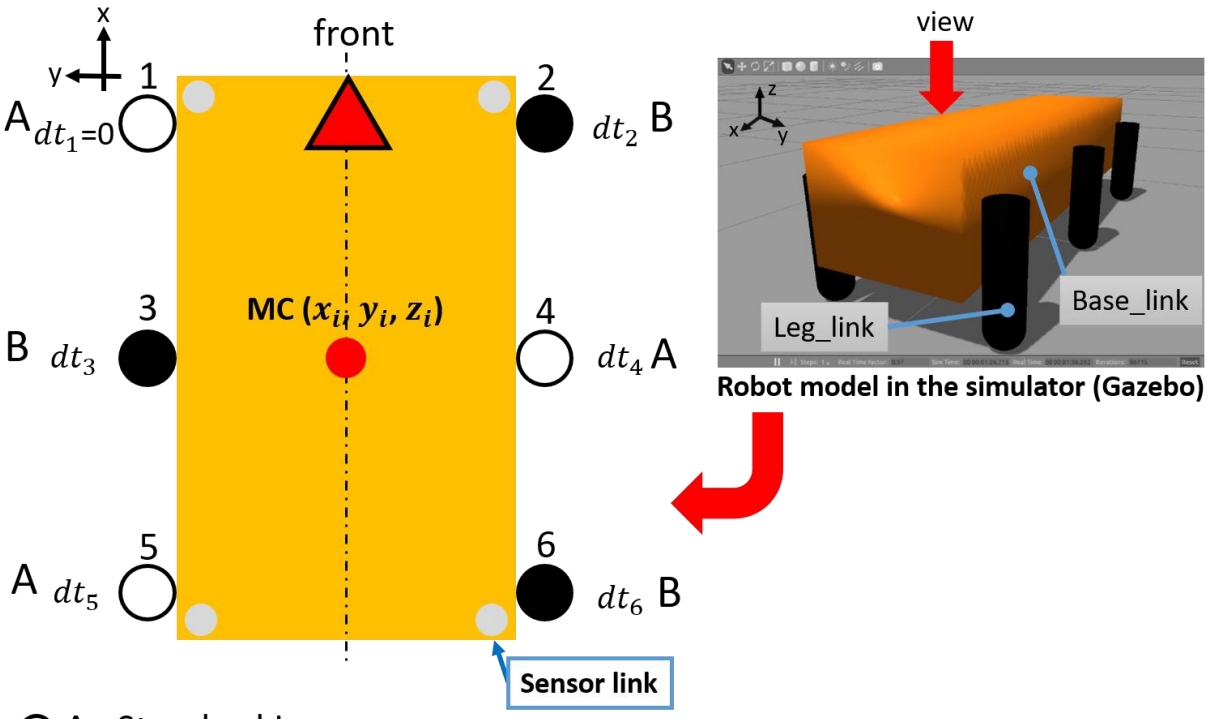

図 3-9 各脚に設定する位相差（待ち時間）の概要

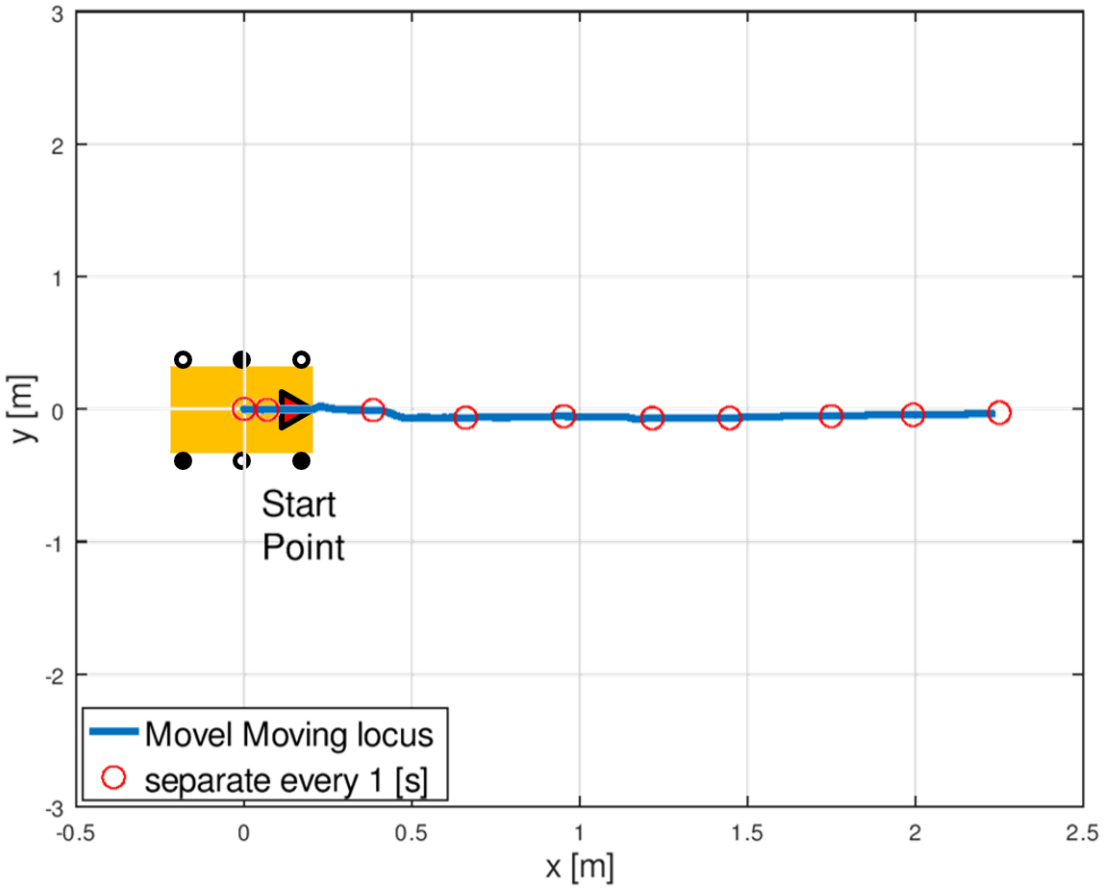

図 3-10 動歩行時のモデル移動軌跡

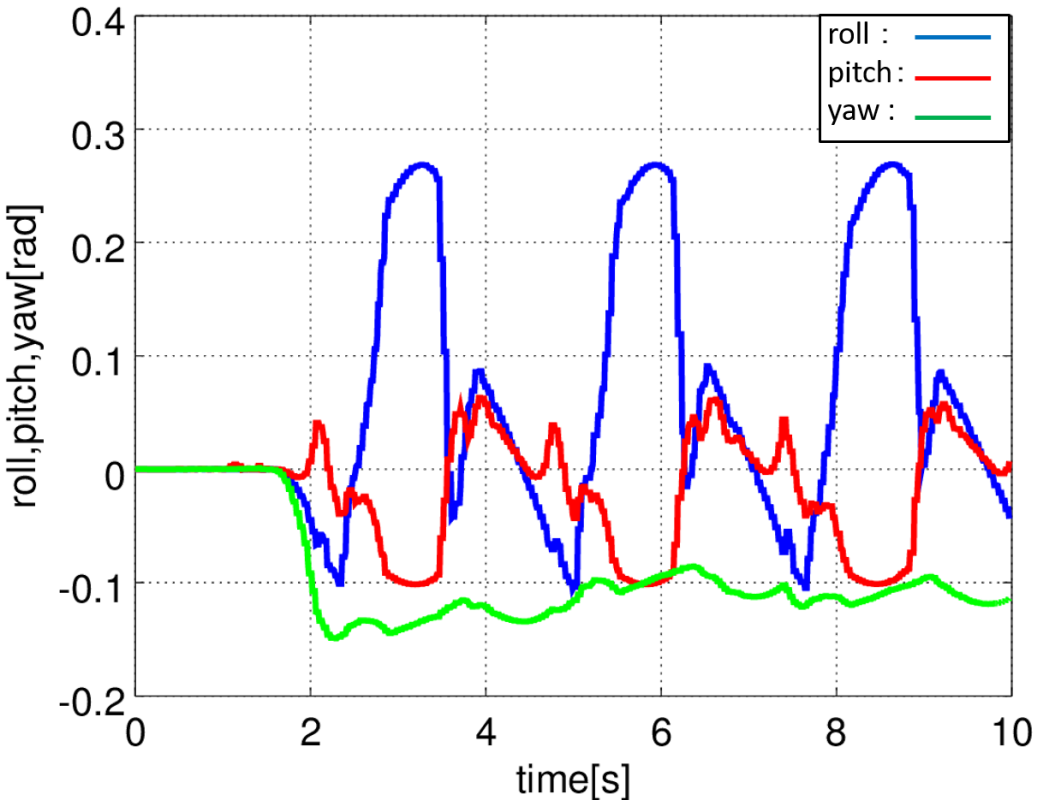

図 3-11 動歩行時の胴体の傾きに関する時間変化量

表 3-2 GA で設計した各脚に位相差を設定した時の制御パラメータ（動歩行時）

θ：接地脚域を定める角度[deg]	61.52
ω_1：接地脚時の角速度[rad/s]	0.81
k：遊脚時の角速度の係数[-]	4.84
dt_2：位相差(右前脚)[s]	1.24
dt_3：位相差(左中脚) [s]	1.40
dt_4：位相差(右中脚) [s]	1.66
dt_5：位相差(左後脚) [s]	1.80
dt_6：位相差(右後脚) [s]	0.59

3.2.1.3. 評価式の再検討

3.2.1.2 節の結果より，評価式を(1)式から以下の(2)式に変更した．

$$\begin{cases} f = a(x_{end} - x_{start})^2 + b\dfrac{1}{\sum {y_i}^2} + c\sum {z_i}^2 + d\dfrac{1}{penalty} + e\dfrac{1}{\sum {roll_i}^2 + \sum {pitch_i}^2 + \sum {yaw_i}^2} \\ penalty = \sum {p_i}^2 + \sum {p1_i}^2 + \sum {p2_i}^2 + \sum {p3_i}^2 + \sum {p4_i}^2 + 1 + p_x \\ a = 10,\ b = 0.01,\ c = 0.1,\ d = 10^8, e = 10 \end{cases} \tag{2}$$

(2)式では，モデル胴体の x 軸に対する傾き$roll_i$，y 軸に対する傾き$pitch_i$，　z 軸に対する傾きyaw_iの時間変化量を用いて胴体の傾きに対する評価を行うように右辺第 5 項を設定している．また，(2)式の右辺各項の重みは静歩行を行うもののみに遺伝子を進化させるため，特にモデル胴体の傾きに対する重みが一番大きくなるように設定した．この(2)式の適応度:fが最大となる制御パラメータの作成を GA により行う．また，(2)式を用いた GA シミュレーションプログラムを付録 1 プログラムリストの List.05 に示す．

(2)式を用いて世代数を 30，個体数を 100 に設定してシミュレーションを行った．その結果，遺伝子は静歩行を行うものに進化していることが確認できた．ここで，この検討中に興味深い歩行パターンが得られた．この時の各脚の sin 波形を図 3-12 に，モデルの移動軌跡を図 3-13 に，モデル胴体の傾きの時間変化を図 3-14 に，GA で設計したパラメータを表 3-3 にそれぞれ示す．図 3-12 と 3.2.1.1 節の図 3-7 を比較すると，基準脚（A グループ）の一脚がずれているが，A もしくは B のどちらかのグループが常に接地脚となることが確認できる．図 3-13 では，モデルは前方に直進していることがわかり，腹部を地面に付けず安定した歩行を行っていることを確認した．また，図 3-14 からモデルの傾きは 0.015[rad]とかなり低く，この結果からもモデルは安定した姿勢で移動できていると言える．そのため，今回の結果では，実質 5 脚のみを使用して安定した歩行が行えたと言える．このように GA では適切な評価式を用いることで周囲の環境に合わせた歩行パターンを作成することができる．また，GA を用いることで人間が想定していないユニークな歩行パターンも作成可能であると言える．

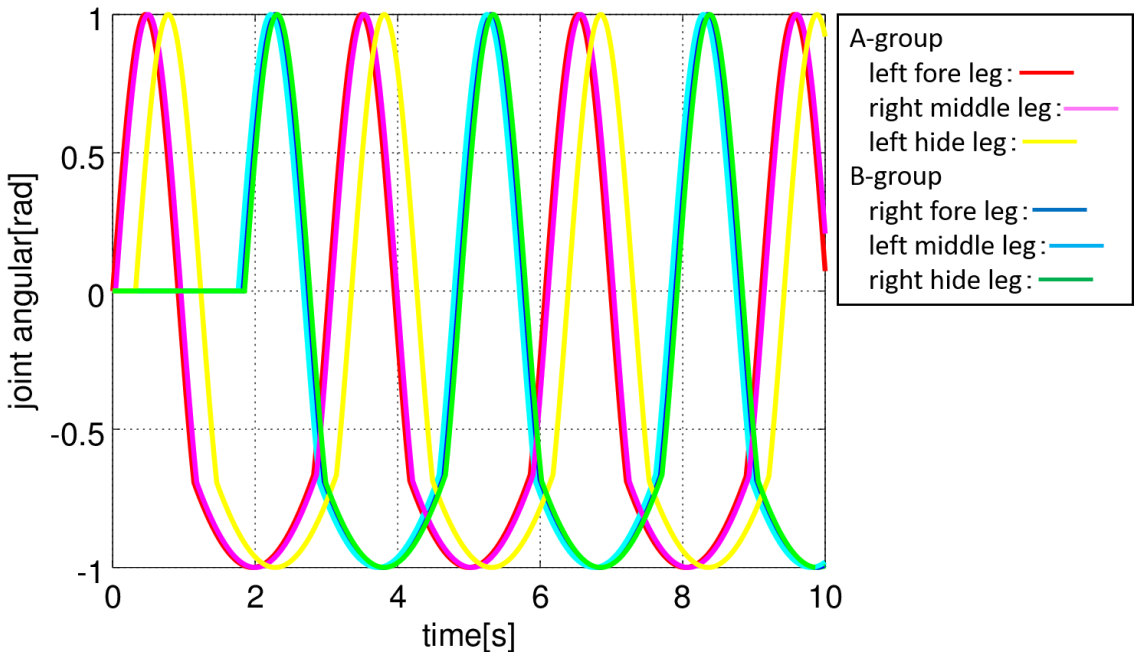

図 3-12 各脚に位相差を個別に設定した時の各脚の動作を示す sin 波形（5 脚歩行時）

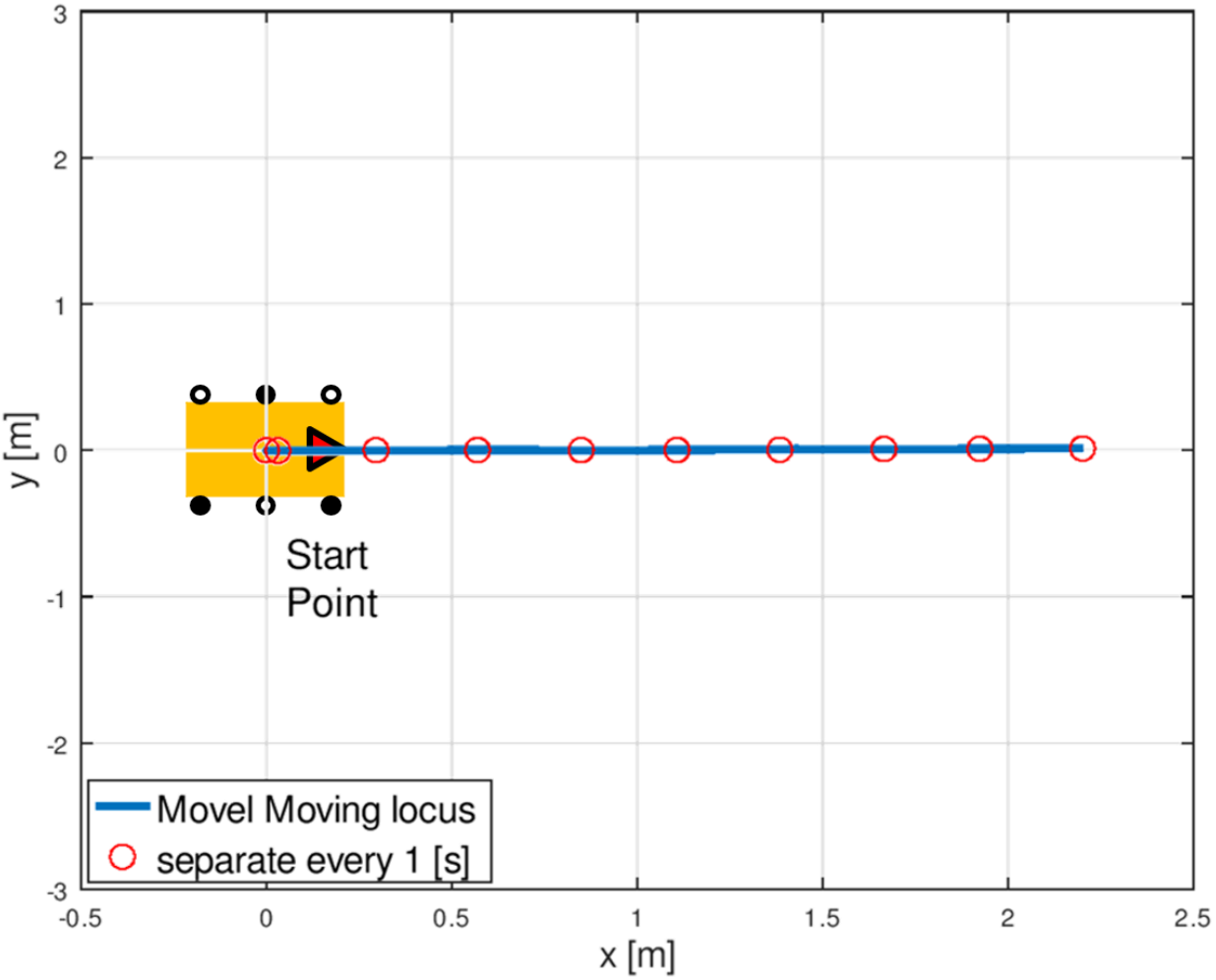

図 3-13 5 脚歩行時のモデル移動軌跡

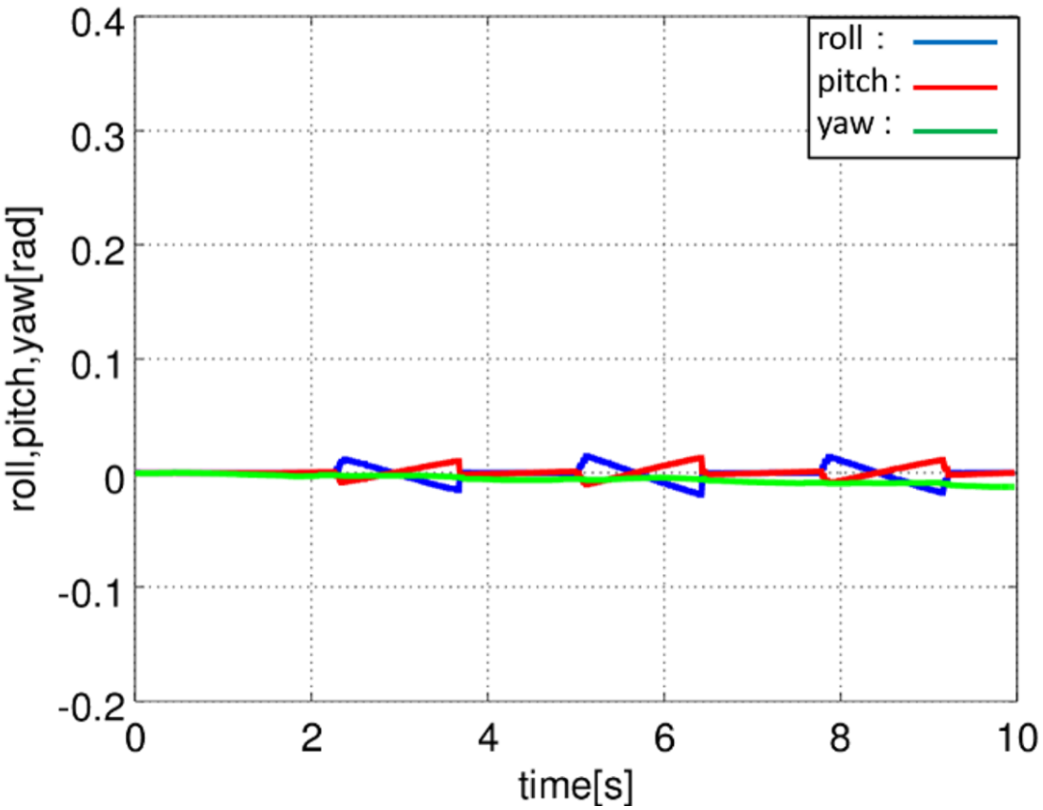

図 3-14 5 脚歩行時の胴体の傾きに関する時間変化量

表 3-3 GA で設計した各脚に位相差を設定した時の制御パラメータ (5 脚歩行時)

θ：接地脚域を定める角度[deg]	42.18
ω_1：接地脚時の角速度[rad/s]	0.98
k：遊脚時の角速度の係数[-]	3.50
dt_2：位相差(右前脚)[s]	1.82
dt_3：位相差(左中脚) [s]	1.77
dt_4：位相差(右中脚) [s]	0.05
dt_5：位相差(左後脚) [s]	0.32
dt_6：位相差(右後脚) [s]	1.86

3.2.2. 旋回する静歩行パターンの検討

GA によって，6 脚 6 節の歩行ロボットモデルが平地で安定な歩行が行えることは 3.2.1 節で確認した．次に，陸上においてより自由な移動を実現するために，モデルが左右旋回運動を行う場合について検討した．具体的には，前進する静歩行パターンに位相差または速度比を加えて旋回歩行を目指す．3.2.2.1 節には位相差による手法を，3.2.2.2 節では速度比による手法をそれぞれ示す．しかし，速度比による手法では想定していた結果を得ることが出来なかった．そこで，モデルが安定した姿勢でその場回頭を行うパターンをベースとし，速度比を加える手法も検討した．この結果を 3.2.2.3 節に示す．

3.2.2.1. 位相差による手法

位相差を利用した旋回歩行の概要について図 3-15 に示す．ここでは，多足歩行型ロボットの静歩行パターンをベースとし，図 3-15 のようにモデル片側の脚に任意の位相差を加える．これによって，左右の脚で接地タイミングがずれるため，旋回歩行を行えるのではないかと考えた．ここで，任意の位相差については，ベースパターンの 1 周期をT[s]とし$T/n\ (n = 4,6,8)$を加えることとする．この時の T の値については作成した歩行パターンより平均値を求めて使用した．ベースパターンとなる静歩行パターンとして，接地脚域を決める角度：$\theta = 30$，45，60[deg]の 3 パターンを試した．使用した歩行パターンの制御パラメータを表 3-4 に示す．$\theta = 45$[deg]のパターンについては，静歩行となるように一意に決めたパラメータである[5]．$\theta = 30$，60[deg]についてはθの値を固定として，GA によって作成した．ここで，評価式には 3.2.1.1 節の(1)式を使用し静歩行を行うパラメータのみに進化するように右辺第四項の重みのみ$d = 1 \times 10^6$に変更した．また，表 3-4 のパラメータを用いてベースとなる歩行パターンを作成した．使用したプログラムを付録 1 プログラムリストの List.06 に示す．これらのベースパターンでモデルの右側となる脚に任意の位相差として$T/4$を加えて 100 秒間歩行させた．その結果を図 3-16 に示す．

図 3-16 では，モデルは左前方に大きく旋回しながら前進していた．また，どの歩行パターンでも腹部を地面に付けず安定な姿勢で移動できていることが確認できた．各歩行パターンはθの値が大きくなるにつれて旋回半径が小さくなる傾向が見られる．これは，接地脚領域が小さくなることでモデルは胴体位置をなるべく高く保つため，位相差を加えたことで接地する瞬間がずれる影響を大きく受けたのではないかと考えている．よって，ベースパターンとしては$\theta = 60$[deg]の静歩行パターンを用いることとする．

次に，任意の位相差として$T/n\ (n = 4,6,8)$をモデルの右側に加えて，先ほどと同様に 100 秒間歩行させた結果を図 3-17 に示す．図 3-17 では，図 3-16 と同様にモデルは左前方に大きく旋回しながら前進していた．脚の滑りなどによるランダム性の影響を受けてはいるが，モデルの旋回性能は加える位相差の大きさにほぼ比例する結果となった．しかし，各位相差において，T/8では腹部を地面に接触させず安定した姿勢で歩行できていたが，T/4およびT/6については時折腹部を地面に接触させながら歩行していたため安定な歩行とは言えない．また，最も旋回性能が良い結果であった位相差$T/4$の歩行では，回転半径は約 13[m]となっている．そのため，これ以上与える位相差を大きくしても，この手法では大きく旋回する歩行パターンしか作成できないと考えられる．そこで，次の手法として速度比を与えた場合についての検討を行った．

表 3-4 ベースとする静歩行パターンの制御パラメータ

	$\theta = 30$[deg]	$\theta = 45$[deg]	$\theta = 60$[deg]
ω_1：接地脚時の角速度[rad/s]	0.73	1.00	0.86
k：遊脚時の角速度の係数[-]	4.49	4.00	4.77
dt：位相差[s]	1.72	1.17	1.24
T：歩行周期[s]	4.14	2.74	2.49

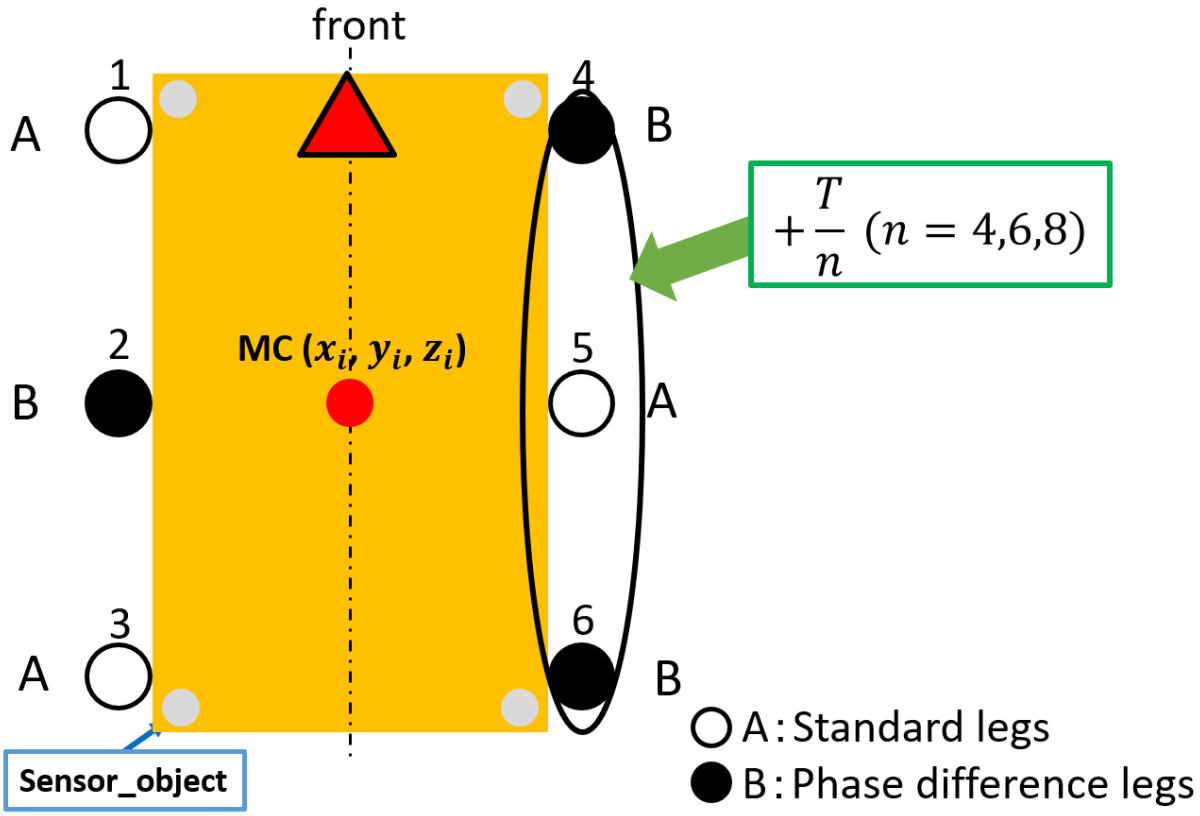

図 3-15 位相差による旋回歩行の概要図

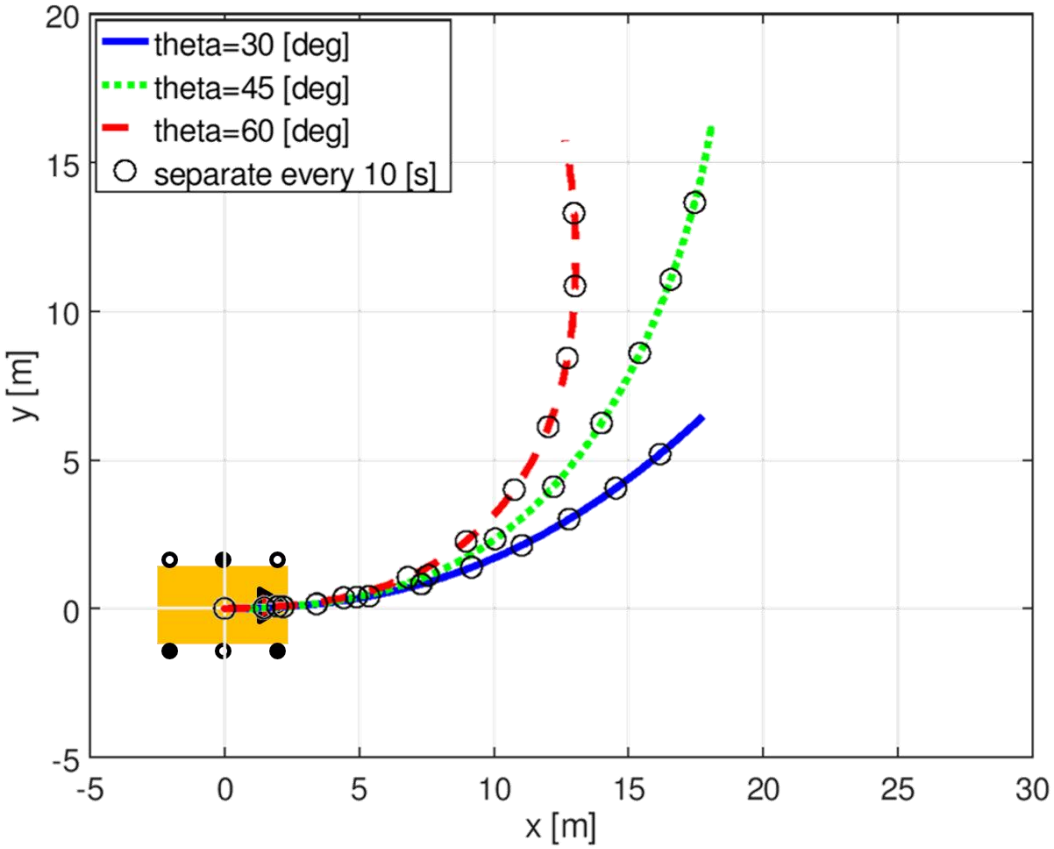

図 3-16 静歩行パターンの比較（位相差：T/4）

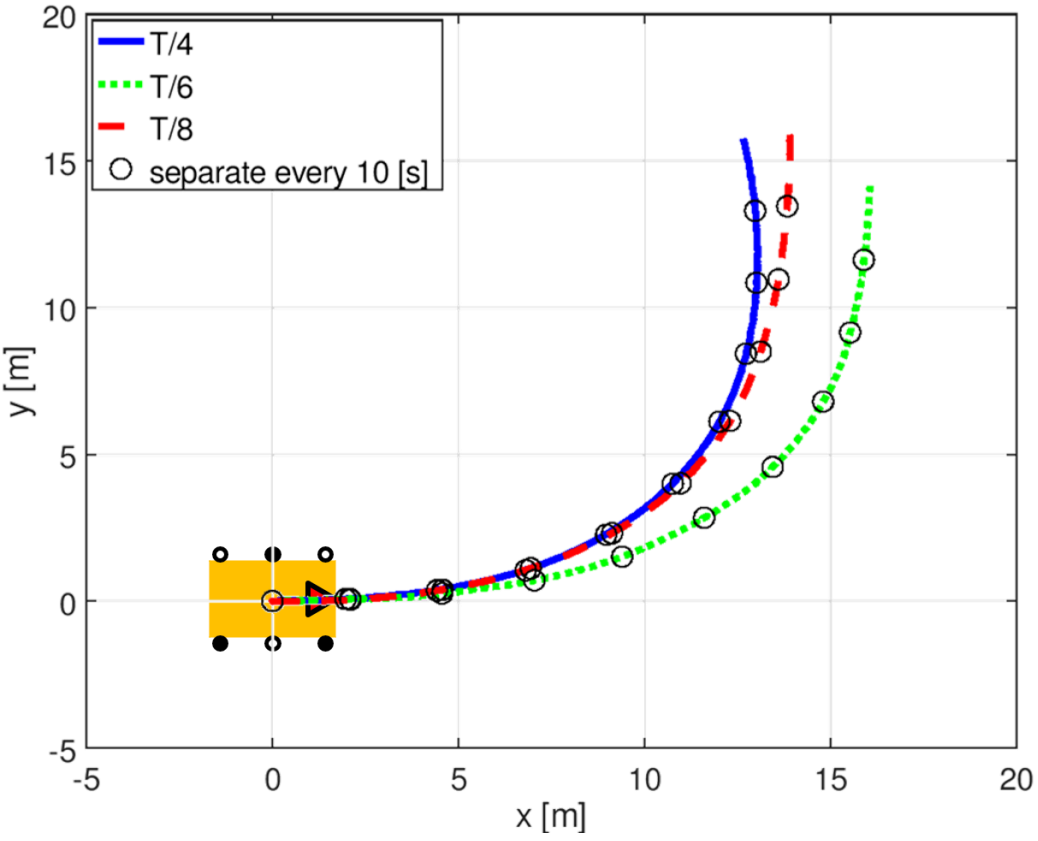

図 3-17 各位相差を加えた旋回歩行パターンの比較

3.2.2.2. 速度比による手法

ここでは，3.2.2.1 節に比べてより小さな旋回半径で旋回歩行できるパターンの作成を目標とする．旋回の手法としては，3.2.2.1 節の図 3-15 と同様に静歩行パターンをベースとし，モデル片側の脚に任意の速度比を与える．これにより，脚の接地時間を短くしモデルを傾かせることで旋回歩行を行う．3.2.2.1 節の表 3-4 の$\theta = 60$[deg]の静歩行パターンをベースとし，任意の速度比は 1.5 および 2.0 としてベースパターンの角速度に加える．また，速度比を加えたパターンは歩行周期：Tがベースパターンと同じになるようにθを設計する．ベースパターンの歩行周期は表 3-4 より$T = 2.49$[s]であるが，これは以下の(3)式のように考えられる．

$$\begin{cases} T = T_1 + T_2 \\ T_1 = (180 - 2\theta)\dfrac{\pi}{180} \times \dfrac{1}{\omega_1} \\ T_2 = (180 + 2\theta)\dfrac{\pi}{180} \times \dfrac{1}{\omega_2} \end{cases} \tag{3}$$

ここで，(3)式において，T_1は接地脚時の周期，T_2は遊脚時の周期をそれぞれ表している(図 3-18)．この(3)式をθについて解いた(4)式を用いて，角速度を任意の速度比倍した時のθを算出した．

$$\theta = \frac{T - \pi\left(\dfrac{1}{\omega_1} + \dfrac{1}{\omega_2}\right)}{\dfrac{\pi}{90}\left(\dfrac{1}{\omega_1} - \dfrac{1}{\omega_2}\right)} \tag{4}$$

各速度比を加えた旋回歩行パターンを作成し，100 秒間の歩行試験を行った．ここで，各速度比の歩行パターンの作成に使用した制御パラメータを表3-5に，シミュレーション結果を図3-19に示す．図 3-19 では，モデルは右前方に大きく旋回しながら前進していた．予想としては加えた速度比が大きいほど旋回性能も向上し，小さな旋回半径を描くように円形に旋回すると考えていた．しかし，実際には速度比(Speed ratio)1.5 および 2.0 を加えてもモデルの移動軌跡には大きな違いは見られなかった．また，移動軌跡自体も円形を描くことはなく，回転半径が 30[m]を超える非常に大回りとなる旋回歩行を行っていた．原因としては，脚の滑りによる影響などが考えられるが，詳細については不明である．この結果から，速度比の値を変更しても旋回性能の向上は見込めないため，次にベースとなる歩行パターンについて注目した．

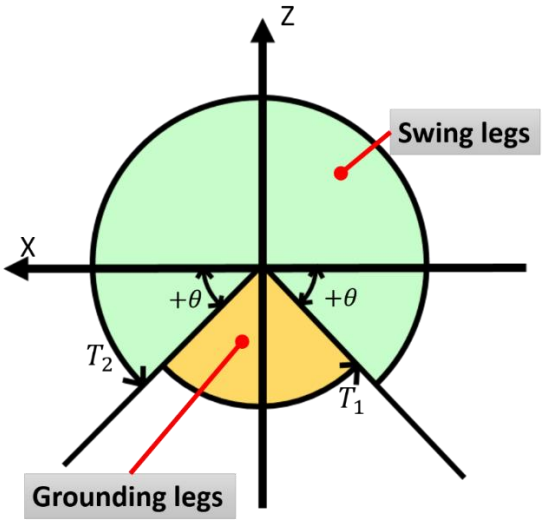

図 3-18 歩行周期の概要

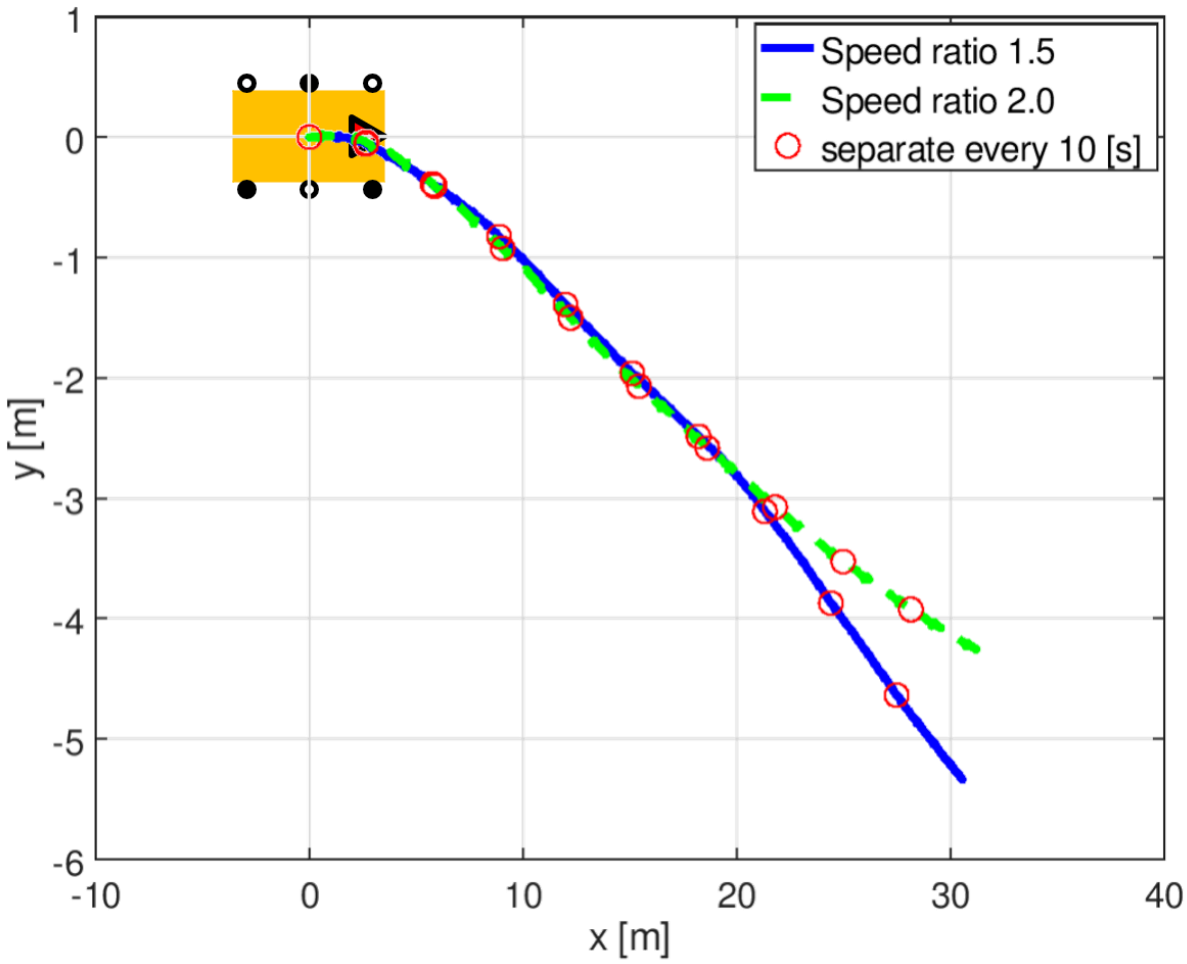

図 3-19 各速度比を使用した歩行パターンの比較(ベースパターン：前進する歩行パターン)

表 3-5 $\theta = 60$[deg]をベースパターンとした時の各速度比の制御パラメータ

	速度比 1.5	速度比 2.0
θ：接地脚域を定める角度[deg]	21.10	-17.80
ω_1：接地脚時の角速度[rad/s]	1.29	1.72
ω_2：遊脚時の角速度[rad/s]	6.14	8.19
dt：位相差[s]	1.24	1.24

3.2.2.3. その場回頭を行うパターンに速度比を加えた場合の検討

3.1 節の図 3-2 のモデルは図 3-20 に示すように，モデルの左右で脚に出力する角速度の正負を変えることでその場回頭に似た動作を行う．ここで，3.2.2.1 節の表 3-4 の$\theta = 45$[deg]のパラメータを用いて，モデル右側の脚に与える角速度を負とした歩行パターンを作成し，100 秒間の歩行試験を行った．この時のモデル移動軌跡を図 3-21 に示す．図 3-20 のように角速度を与えることでモデルは後方に進み，時計回りに旋回していく．また，胴体中心が開始地点(start point)から若干ズレてしまうものの，モデルは小さい回転半径で旋回していることが確認できる．よって，モデル左右の脚で与える角速度の正負を変えることで，その場回頭に似た動作が可能であると言える．これに速度比を組み合わせることで旋回性能の向上を図った．3.2.2.1 節の表 3-4 の$\theta = 60$[deg]をベースとし，図 3-22 に示すように角速度の正負と速度比を設定した．図 3-22 より，モデルの右側には任意の速度比を加え，さらに負の角速度を出力する．これによって，小さな旋回運動を行うことを目指す．また，制御パラメータとしては，3.2.2.2 節の表 3-4 と同じである．

各速度比の歩行パターンによるモデルの 20 秒間の歩行結果を図 3-22 に示す．図 3-21 と同様にモデルは後方に進み,時計回りに旋回する.ベースパターンとしてその場回頭のパターンを用いた結果,図 3-21 のように回転半径が約 0.5[m]と，図 3-19 とあまり変わらないが，小さく旋回歩行を行っていることが確認できた．また，通常であれば与える速度比に比例して旋回性能が向上すると考えられるが，今回の結果では逆の傾向が見られた．原因としては，速度比の上昇によってモデルの傾きが大きくなった分，脚の滑りによる影響も強くなり傾いた方向に胴体が流れたためだと考えている．今後は，希望の回転半径で回転するようにモデルに与える速度比を GA で設計する予定である．

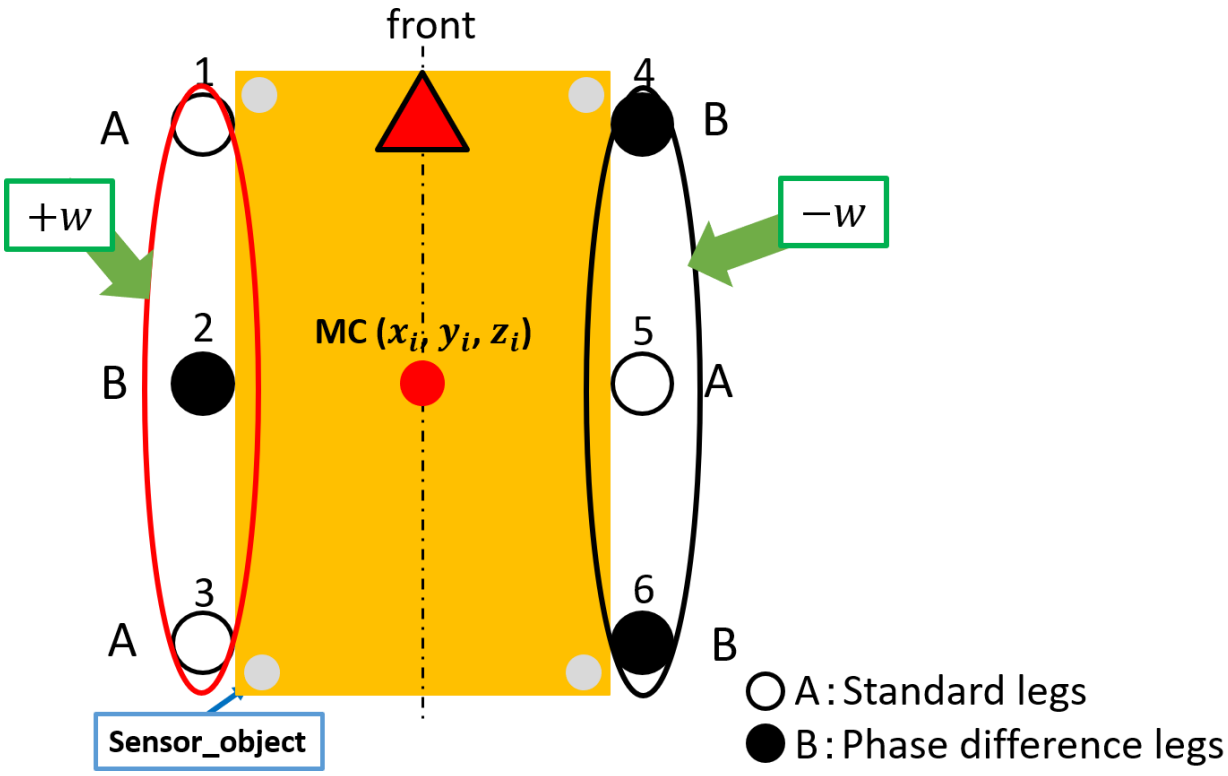

図 3-20 その場回頭パターンの概要

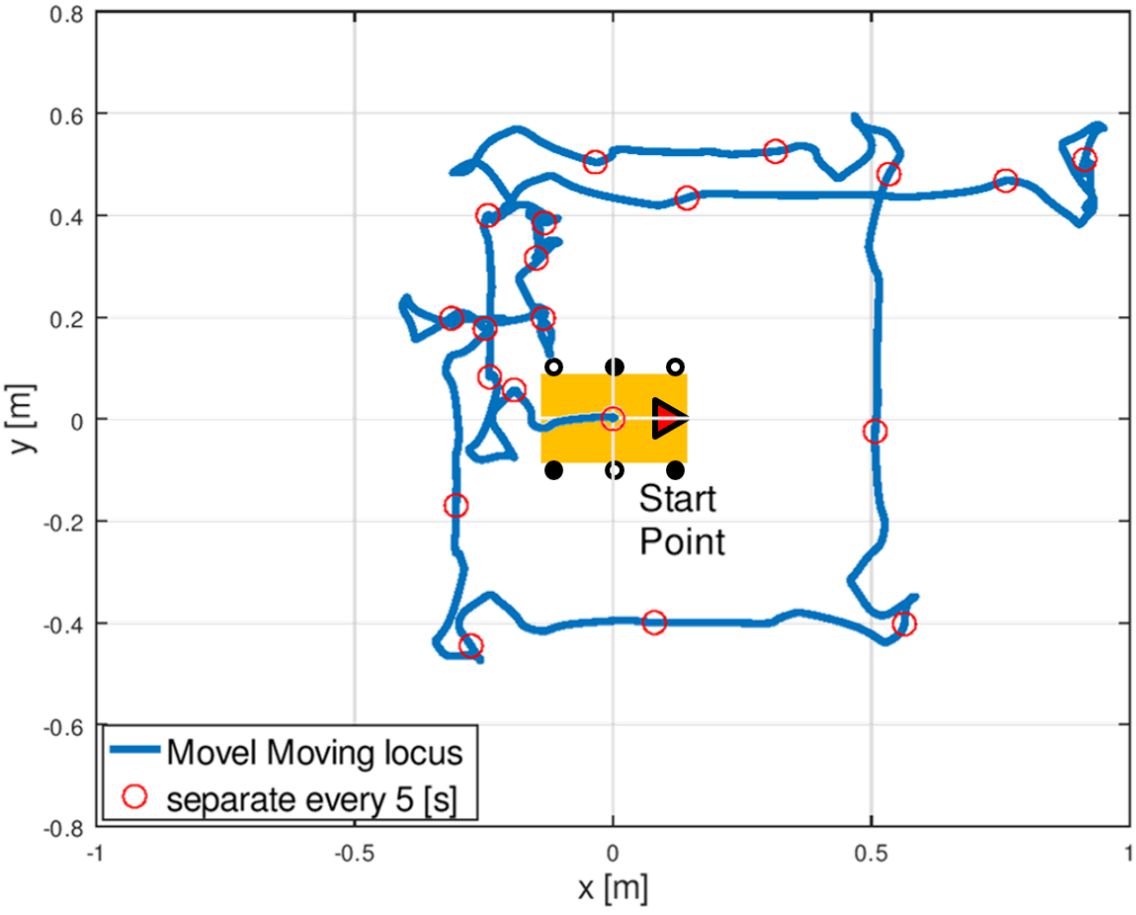

図 3-21 その場回頭時のモデル移動軌跡

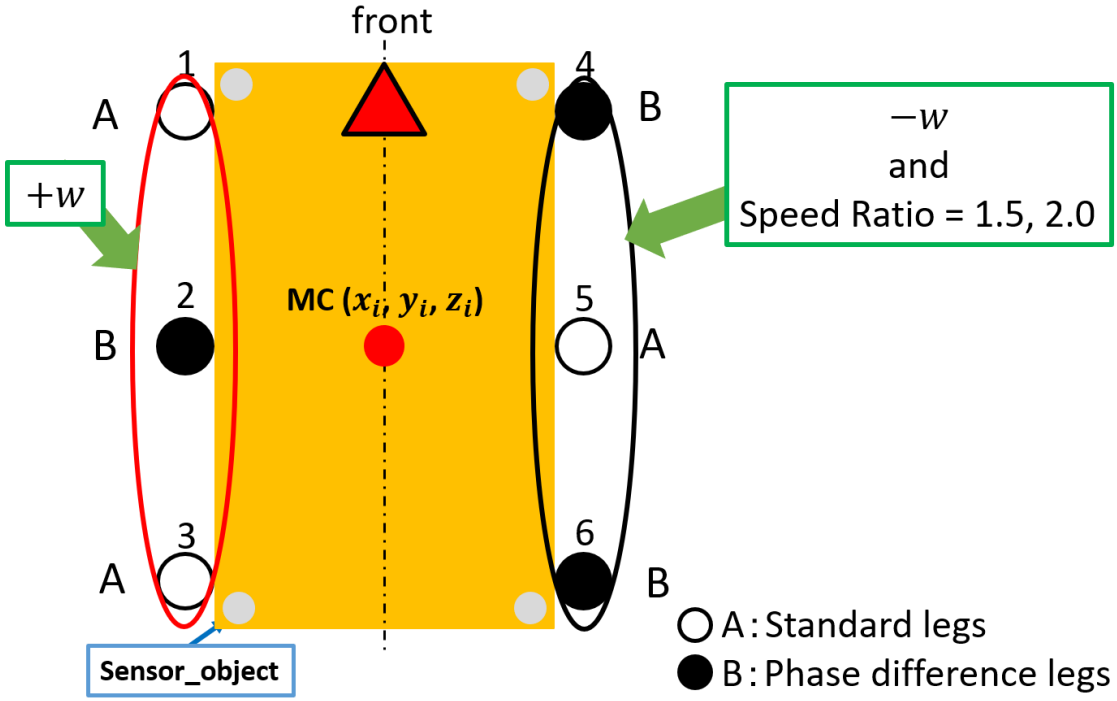

図 3-22 速度比＋その場回頭パターンの概要

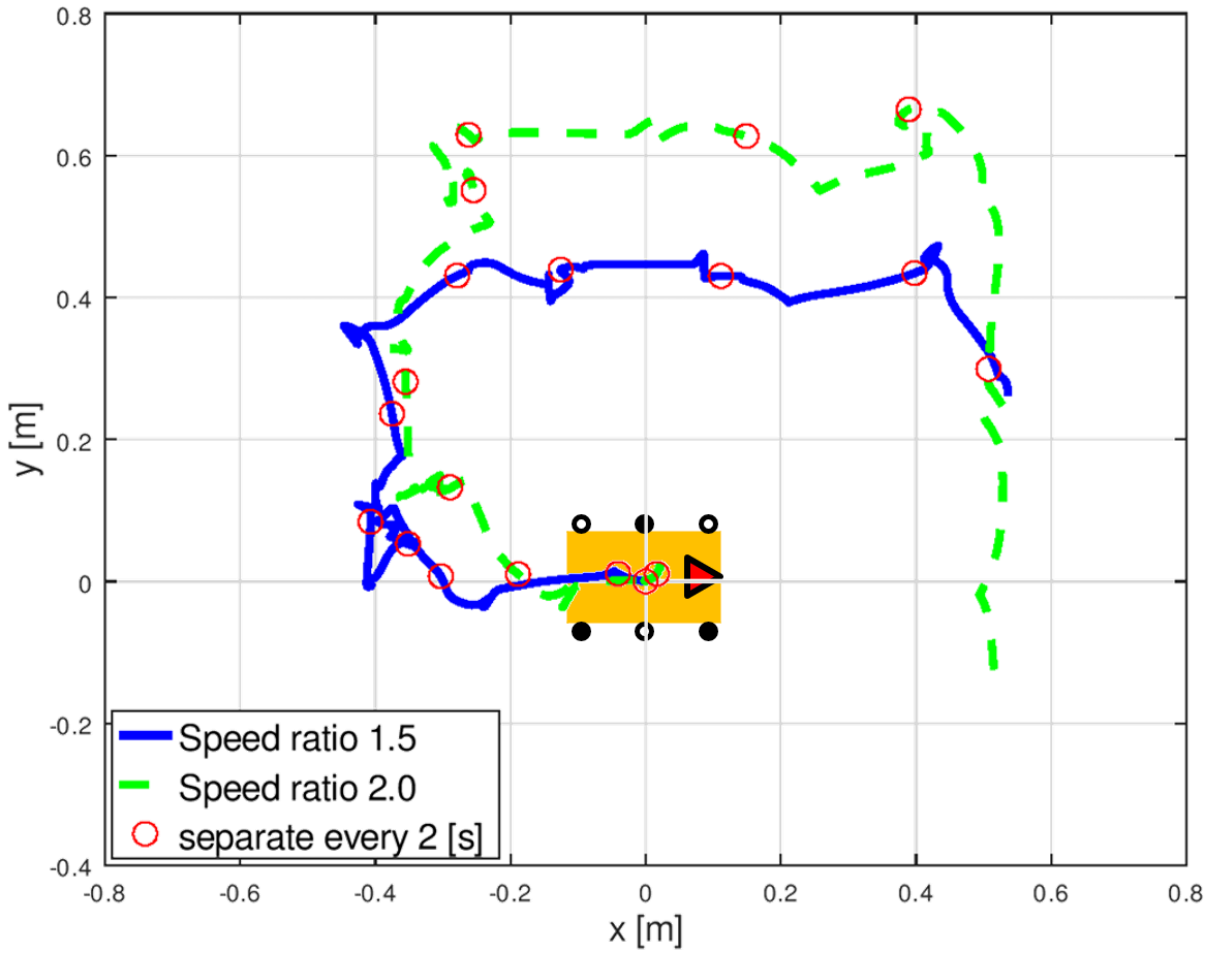

図 3-23 各速度比を使用した歩行パターンの比較(ベースパターン：その場回頭パターン)

3.2.3. GAによる静歩行パターンの検討結果のまとめ

3.2節では，3.1節の図3-2のモデルが陸上(平地)で安定な姿勢で歩行する手法を検討し，GAを用いた静歩行パターンの作成した．モデルの歩行では，各関節において接地脚時と遊脚時の角速度を制御する角速度制御法によって行った．GA については，この角速度制御に必要となるパラメータを設計することに使用した．また，本研究ではモデルの安定な姿勢での歩行として，腹部を地面に付けずに歩行することを目指した．

3.2.1節では，モデルが前進するための歩行パターンの作成を行った．まずは，このモデルが静的な歩行(トライポッド歩行)が行えることを確かめるため，モデルの各関節に与える制御パラメータを基準脚と位相差脚で同じものとした．これによって，モデルがトライポッド歩行により前進出来ることを確認した．次に，モデルが水域や沼地で動作することを考慮し，動作の自由度を増加させた場合について検討した．そのため，モデルに与える制御パラメータの内与える位相差:dtを関節ごとに設定した．検討の結果，GAでの評価式を適切なものにすることで，3.2.2.1節と同様にトライポッド歩行が行えることを確認した．また，検討の途中でモデルが実質5脚のみで歩行するようなユニークなパターンを見つけることが出来た．このようにGAを用いることで，こちらが意図していないようなユニークなパターンを作成できることを確認した．

3.2.2節では，陸上でモデルがより自由な移動を行うため，旋回歩行する手法について検討を行った．内容としては，ベースとなる静歩行パターンのモデル片側の脚のみに，位相差または速度比を加えて旋回することを目指した．3.2.2.1節より，位相差を用いた手法では，与える位相差を$T/8$（Tはベースとする歩行パターンの1周期)とすることで，安定な姿勢のまま大きく旋回する歩行パターンが作成できることを確認した．また，この時の回転半径は約15[m]となっていた．3.2.2.2節では，速度比によって 3.2.2.1 節よりも小さく旋回する歩行を目指したが，思ったような旋回歩行結果は得られなかった．そのため，その場回頭を行うパターンをベースとして速度比を与える場合について検討した．その結果，3.2.2.3節より回転半径が約0.5[m]の小さな旋回歩行パターンを作成することが出来た．

以上の結果より，3.2.1節では前進する静歩行パターンが，3.2.2節では旋回する静歩行パターンがそれぞれ作成できた．これらの歩行パターンを組み合わせることで，障害物などの回避や目標地点への移動など，陸上での自由な移動が行えると考えている．

3.3. 沼地・水域環境の再現方法の検討

本研究で提案する調査ロボットは干潟での動作を想定している．しかし，現状の Gazebo では干潟の様な陸域と水域の混ざり合った複合環境を作成することはできない．そのため，シミュレーションについても動作環境として水域や沼地を設定した場合について検討する必要がある．これについては，Gazebo で沼地・水域環境の代用となる環境を作成し，実際にシミュレーションを行った[5]．図 3-24 では沼地・水域環境として，液体や泥を微小な粒子の集合であると考え，ボールプールで代用した．このボールプールは，直径 0.1[m]の球を使用し，縦 3.0[m]，横 3.0[m]，高さ 0.2[m]のサイズとした．

この環境で 10 秒間の歩行シミュレーションを行った．ここでは，3.2.1.1 節の表 3-1 の制御パラメータを使用して，ボールプール内を前進させた．シミュレーションの結果，歩行完了までに実時間で約 6 分間(36 倍の時間)を要した．これは，シミュレーションに使用したオブジェクト数が非常に多く，PC 側への負荷が増大したためである．しかし，考え方を変えれば PC の計算処理速度が向上すれば，このボールプールを沼地・水域環境の代用として使用することができると言える．計算処理速度の向上方法としては，GPU を用いた並列処理やマルチスレッド化などを現在検討している．

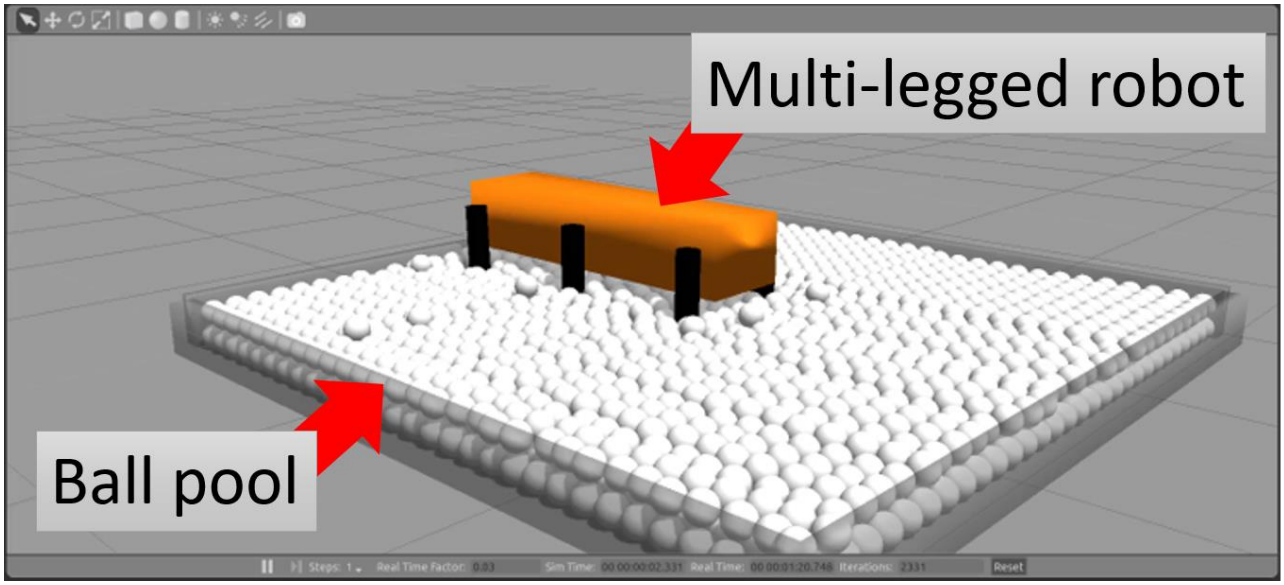

図 3-24 ボールプールを用いた歩行試験

4. 非線形同期による多足歩行型ロボットの運動制御手法

3 章において，GA による手法では適切な評価式を用いることで環境に合わせた歩行パターンを作成することができた．しかし，GA によって安定した歩行パターンを作成するためには膨大な試行回数を要する，周囲の環境が変化すると再計算が必要などの問題点がある．この詳細については 4.1 節に示す．これらの問題点を解決する方法として，非線形同期現象を多足歩行型ロボットの歩行に用いることを検討した．ここでは，ロボットの各脚を振動子として同期・結合させることで，ロボットを歩行させる．そのため，脚の運動を表す数理モデルとして結合 VDP（Van Der Pol）方程式を用いる．ここで，3.1 節の図 3-2 の様なロボットが静歩行（トライポッド歩行）を行うことためには，基準脚と位相差脚をそれぞれ結合させる必要がある．よって，結合数は 2 として検討を行った．また，結合数が 2 であれば試験モデルの解析が容易になる．

まずは，2 結合 VDP 方程式が安定な周期解であるかを，フロケの理論[7]によって確認した．詳細については 4.2 節に示す．また，3.1 節の図 3-2 の様なロボットが歩行を行う場合には周期解を逆相解で同期させる必要がある．ROS および Gazebo を用いて，2 つの単一振り子を持つモデル（2 脚モデル）を作成し，オープンループ制御系を作成し制御を行った．オープンループ制御系では，VDP 方程式とフロケの理論を用いて安定な周期解を作成できること，および周期解の同期状態を任意のタイミングで変更できることを確認した．この結果については，4.3 節に示す．しかし，オープンループ制御系では，周囲の環境が変化した場合，対応することが出来ない．そこで，様々な環境でも動作できるようにフィードバック制御系について検討した．この結果について，4.4 節に示す．また，2 脚モデルの制御結果についてのまとめを 4.5 節に示す．

4.1. GAの問題点

3章よりGAを用いることで，環境に合わせた歩行パターンを作成でき，なおかつ人が考えないようなユニークな歩行パターンまでも作成できることを確認した．しかし，この方法では，歩行に適した遺伝子を作成するまでに実時間で約4時間もかかる．これは，GAでは歩行に適さない遺伝子であっても動力学シミュレーションを行うためである．加えて，動力学シミュレーションでは10秒間の歩行を行うために，実時間で約60秒を要する．よって，試行回数が増えるのに比例してシミュレーション時間も増大してしまう．また，GAによる手法では歩行パターンを作成しているだけなので周囲の環境が変化すると再度シミュレーションを行う必要がある．そのため，環境の変化や突発的な外乱にも対応できる制御系を考える必要がある．上記の問題点より，現状ではGAによる歩行パターンの作成を多足歩行型ロボットの運動制御システムに使用することはあまり現実的なものではない．そこで，自然界で周囲の環境に適した移動方法をとるムカデの歩行に注目し，多足歩行型ロボットの運動制御に非線形同期現象を利用することについて検討する．

ムカデなどの多脚生物が歩行する場合，各脚を1つの振動子とすると各脚の振動子を結合し同期させることで歩行していると考えられる．そのため，全体の脚の動きは結合振動子の同期現象であると考える事ができる．また，実際の生物は周囲の環境の変化に適応して歩行を行っている．そこで，脚の運動を表す数理モデルとして，結合VDP方程式を用いる．結合VDP方程式は非線形項が0よりも大きい時に周期解（リミットサイクル）を持つことが知られている[7]．これによって，外乱が発生しても元の安定した周期に戻ることが出来るため，環境の変化に対応した歩行が行える．これを多足歩行ロボットに応用する．各脚を振動子として扱い結合・同期させることによって，周囲の環境変化にも対応できる歩行パターンの作成を目指す．

4.2. フロケの理論による周期解の安定性解析

まずは，VDP 方程式の周期解が安定なものであるかを解析する手法について検討を行った．本研究では,周期解の安定性解析の手法として,フロケの理論を用いることを検討した.フロケの理論とは,微分方程式の一般解よりフロケ乗数（特性乗数）:μを求めることで，μの全ての大きさが単根の 1 を除いて 1 未満であれば，その解は安定であると判別する手法である[7]．これにより，VDP 方程式の一般解が求まればフロケの理論から安定した解の判別を行うことができる．しかし，VDP 方程式は非線形の微分方程式であるため，一般解を求めることは困難である．そこで，VDP 方程式の近似解からフロケ乗数を求められることをシミュレーションによって確認した．使用する VDP 方程式としては，簡単なモデルであり解析も比較的容易な，以下の(5)式,(6)式の 2 結合 VDP 方程式[10]を考える．

$$\begin{cases}\ddot{x}_1-\varepsilon\left(1-\dot{x}_1{}^2\right)\dot{x}_1+\omega_0{}^2x_1+k_mx_2=0 & (5)\\ \ddot{x}_2-\varepsilon\left(1-\dot{x}_2{}^2\right)\dot{x}_2+\left(\omega_0{}^2+\delta\right)x_2+k_mx_1=0 & (6)\end{cases}$$

上記(5)式,(6)式は結合係数k_mによって相互に結合されている．また，上式においてω_0は固有振動数の初期値を，δは固有振動数のずれをそれぞれ表しており，$\omega_0{}^2>|\delta|$とする．ここで，k_mはδよりも十分大きいとし，非線形項の係数εを$\varepsilon=0$として線形解を導出する．その解より近似解を求めると以下の(7)式,(8)式が求められる[10]．

$$\begin{cases}x_1=A_1\sin(\Omega_1t+\varphi_1)+A_2\sin(\Omega_2t+\varphi_2) & (7)\\ x_2=A_1\sin(\Omega_1t+\varphi_1)-A_2\sin(\Omega_2t+\varphi_2) & (8)\\ \Omega_1=\sqrt{\omega_0+k_1},\Omega_2=\sqrt{\omega_0-k_2} & (9)\\ A_1=\sqrt{\left(\dfrac{x_1+x_2}{2}\right)^2+\left(\dfrac{\dot{x}_1+\dot{x}_2}{2\Omega_1}\right)^2} & (10)\\ A_2=\sqrt{\left(\dfrac{x_1-x_2}{2}\right)^2+\left(\dfrac{\dot{x}_1-\dot{x}_2}{2\Omega_1}\right)^2} & (11)\end{cases}$$

この(7)式,(8)式を用いてフロケ乗数を求める．ここで，(7)式,(8)式におけるφ_1，φ_2は未定数である．そこで，シミュレーションにより(5) 式,(6)式の数値解を求め，その数値解と近似式の誤差が最小となるφ_1，φ_2の組み合わせを同定する．(5)式,(6)式の数値解については，オイラー法では誤差が大きいため，本研究では 4 次のルンゲ=クッタ法を用いて算出した．φの組み合わせの同定方法としては，GA やニューラルネットワークなどによって求められるが，今回は未定数が少ないこともありφの範囲を 0.1[deg]刻みで変更していきメッシュ状にして求めた．ここで，シミュレーションに使用したプログラムを付録 1 プログラムリストの List.08 に示す.

本シミュレーションでは，同期パターンが同相解となるように，参考文献[10]より各パラメータを表 4-1 のものとした．また，VDP 方程式の初期値も$x_1(0)=0.16$，$\dot{x}_1(0)=0$，$x_2(0)=0.06$，$\dot{x}_2(0)=0$に設定した.以下の図 4-1 は,シミュレーションによって求めた(5)式,(6)式の数値解を表している．ここで，本研究では数値解x_1，x_2の 2 つの周期がほぼ同様となったときに同期状態であると判断した．図 4-1 より，数値解x_1，x_2は$t=5$[s]程度から同相で同期していることが確認できる．そのため，定常状態となる$t=5$~15[s]間で数値解と近似解の比較を行う．

シミュレーションの結果,数値解と近似解の誤差が最小となる組み合わせは$\varphi_1=102.3$[deg],$\varphi_2=119.6$[deg]の組み合わせで，誤差：eは$e=5.02$と非常に小さいものとなった．このφ_1，φ_2を用いた(7)式の近似解(x_{a1})と(5)式の数値解(x_{s1})を比較した．結果を図 4-2 に示す．図 4-2 からも，上記のφ_1，φ_2を用いることで数値解に近似する解が得られることが確認できる．また，同様に(8)式の近似

解と(5)式，(6)式の数値解を比較したが，こちらも数値解に近似した解が得られていることを確認した．

次に，フロケ乗数：μを算出し，この時の周期解の安定性解析を行う．フロケ乗数：μについては以下の(12)式より求められる．

$$\begin{cases} \mu = eig[c] \\ c = {V_{(0)}}^{-1} \cdot V_{(T)} \\ V_{(t)} = \begin{bmatrix} \sin(\Omega_1 t + \varphi_1) & \sin(\Omega_2 t + \varphi_2) \\ \sin(\Omega_1 t + \varphi_1) & -\sin(\Omega_2 t + \varphi_2) \end{bmatrix} \end{cases} \tag{12}$$

ここで，図 4-1 において定常状態における 1 周期：Tは$T = 0.89[\mathrm{s}]$である．また，(12)式中の$V_{(t)}$については(7)式, (8)式の基本行列である．試験結果のφ_1，φ_2を用いた近似解からフロケ乗数μを算出すると，以下の(13)式の様に単根の 1 を除いて全ての値が 1 未満になっていることが確認できた．

$$\mu = \begin{bmatrix} 0.68 \\ 1.00 \end{bmatrix} \tag{13}$$

よって，(13)式のフロケ乗数からも(7)式，(8)式の近似解は安定であると言える．しかし，解析に用いたのは VDP 方程式の近似解であるため，フロケ乗数については多少の丸め誤差が生じてしまうことに注意する必要がある．上記の結果から，(7)式，(8)式の近似解からフロケの理論による解の安定性判別は可能だと言える．

表 4-1 同期パターンが同相解となる制御パラメータ

ε	1
ω_0	2π
k_1, k_2	10
δ	0.5

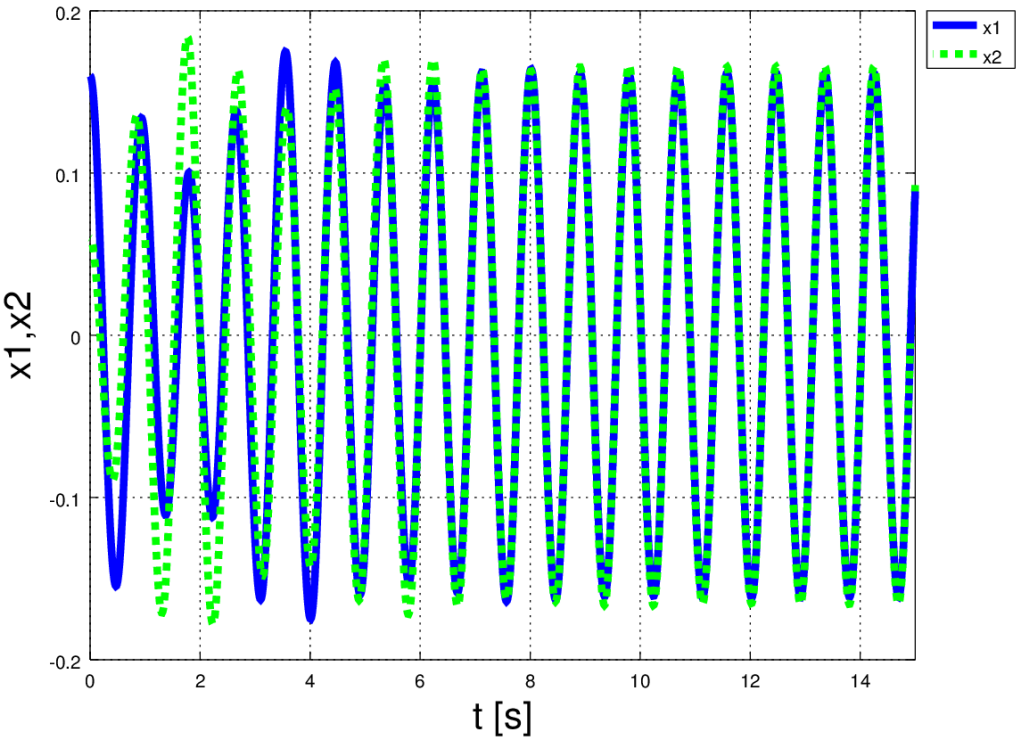

図 4-1 2 結合 VDP 方程式の数値解

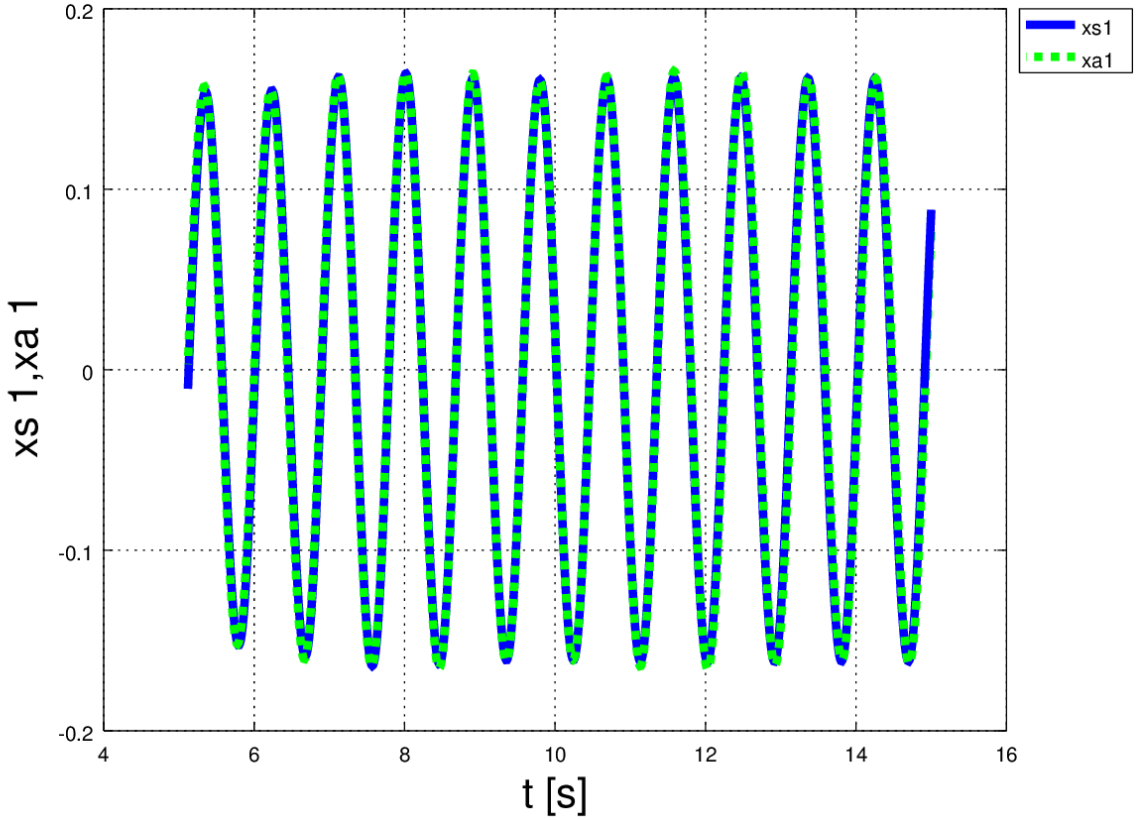

図 4-2 VDP 方程式の近似解と数値解の比較

4.3. オープンループ制御系

本節では，オープンループ制御系において 2 結合 VDP 方程式の周期解の同期パターンを任意のタイミングで変更する手法について検討した．また，周期解の安定性についてはフロケの理論によって確認した．

各脚の数理モデルとして，2 結合の VDP 方程式を使用する．2 結合 VDP 方程式には同相解と逆相解の 2 つの同期パターンがある[10]．しかし，VDP 方程式の同期パターンについては，初期値や結合係数の値によって変化する．また，本研究で提案するロボットは陸上では歩行するため，逆相解での同期が必要となる．よって，任意のタイミングで同期パターンを変更できる手法を考えなければならない．そこで，2 結合 VDP 方程式に周期外力を加えた以下の(14)式，(15)式[10]を使用する．

$$\begin{cases} \ddot{x}_1 - \varepsilon\left(1 - \dot{x}_1{}^2\right)\dot{x}_1 + \omega_0{}^2 x_1 + K_{m1} x_2 = 0 & (14) \\ \ddot{x}_2 - \varepsilon\left(1 - \dot{x}_2{}^2\right)\dot{x}_2 + (\omega_0{}^2 + \delta) x_2 + K_{m1} x_1 = K_{m2} F & (15) \end{cases}$$

(14)式および(15)式において，ω_0は固有振動数の初期値を，δは固有振動数のずれをそれぞれ表しており，$\omega_0{}^2 > |\delta|$の関係がある．これらの式は以下の(16)式より算出される結合係数K_{m1}，K_{m2}によって相互に結合されている．また，(16)式の各パラメータは参考文献[10]の値を参照した．

$$\begin{cases} K_{mn} = \dfrac{k_n}{1 + exp\left[a_n\left(A_{mj} - A_{tj}\right)\right]} \ (n = 1,2) \\ k_1 = k_1 = 10, a_1 = -100, a_1 = 100 \end{cases} \quad (16)$$

(16)式において，jの値は目標とする同期パターンによって決まり，A_{mj}およびA_{tj}は以下の式となる．

・同相解での同期($\varphi_d = 0, j = 1$)

$$\begin{cases} A_{m1} = \sqrt{\left(\dfrac{x_1 + x_2}{2}\right)^2 + \left(\dfrac{\dot{x}_1 + \dot{x}_2}{2\Omega_1}\right)^2} \\ A_{t1} = \sqrt{\dfrac{2}{3\Omega_1{}^2}} \\ \Omega_1 = \sqrt{\omega_0{}^2 + k_1} \end{cases} \quad (17)$$

・逆相解での同期($\varphi_d = \pi,\ j = 2$)

$$\begin{cases} A_{m2} = \sqrt{\left(\dfrac{x_1 - x_2}{2}\right)^2 + \left(\dfrac{\dot{x}_1 - \dot{x}_2}{2\Omega_2}\right)^2} \\ A_{t2} = \sqrt{\dfrac{2}{3\Omega_2{}^2}} \\ \Omega_2 = \sqrt{\omega_0{}^2 - k_1} \end{cases} \quad (18)$$

ここで，φ_dについては目標とする同期パターンの位相差を表し 0 またはπをとる．また，(16)式において Fは以下の(19)式で示す周期外力である．

$$F = -\sin(\varphi_d) K_{AG1}\, x_1 + \cos(\varphi_d)\, K_{AG2} \dot{x}_1 \quad (19)$$

ここで，(19)式中のK_{AG1}およびK_{AG2}はそれぞれ$x_1, \dot{x}_1$のオートコントロールのゲインであり，以下の(20)式で算出する．

$$\begin{cases} K_{AGx1} = \dfrac{1}{|x_1|} \\ K_{AG\dot{x}1} = \dfrac{1}{|\dot{x}_1|} \end{cases} \tag{20}$$

このゲインについては各振幅（x_1, $\dot{x}_1$）を半周期ごとに計測し，その逆数を取ったものである．この周期外力によって，結合 VDP 方程式の相互引き込み現象および強制引き込み現象を用いることで，同期パターンの切り替えを行う[9]．4.2 節の VDP 方程式の近似解(7)式,(8)式よりフロケ乗数が求められることは 4.2 節で確認した．また，VDP 方程式の近似解を使用することで，不安定となる周期解をあらかじめ取り除くことができ，GA の試行回数を減らし計算速度の向上も見込める．これらを踏まえて，周期外力を含む VDP 方程式とその近似解，およびフロケの理論によって安定な周期解が作成できるかを数値解析ソフトウェアの Octave を用いたシミュレーションによって確認した．ここで，未定数φ_1, φ_2については GA によって算出した．この時のシミュレーションプログラムを付録 1 プログラムリストの List.09 に示す．

世代数を 20，個体数を 100，サンプリングタイムを 1[msec]としてシミュレーションを行った．その結果，数値解と近似解の誤差:eが$e = 10.375$と十分に低くなる組み合わせとして，$\varphi_1 = 95.625$[deg]，$\varphi_2 = 322.03$[deg]が求まった．このφ_1, φ_2を用いたシミュレーション結果を図 4-3 に示す．図 4-3 では，任意のタイミングで同相解から逆相解へ同期パターンを変化させたときの結果である．この時，切り替えタイミングtを$t = 7$[s]とし，サンプリングタイムを 10[ms]に設定した．また，ε, ω_0については 4.2 節の表 4-1 を使用した．

図 4-3 より，時間$t = 3$[s]から 7[s]で同相解での同期が確認できた．この時のフロケ乗数μ_1に注目すると，値は 1.00 と 0.04 であるためフロケの理論より安定な解であるといえる．次に，$t = 11$[s]から 20[s]に注目すると数値解x_1，x_2は逆相解で同期していることが確認できる．この時のフロケ乗数μ_2の値は 0.36 と 1.02 となった．μ_2=1.02 という 1 以上の値については，VDP 方程式の近似解を用いたことによる誤差だと考えられる．そのため，逆相解でもフロケの理論的に安定な解であるといえる．したがって，フロケの理論よりオープンループ制御系で安定な周期解が作成できることを確認できた．また，同様に逆相解から同相解への同期パターンの切り替えもシミュレーションを行ったが，こちらでも安定な周期解が得られることを確認した．そのため，(14)，(15)式のように周期外力を加えた 2 結合 VDP 方程式を用いることで，オープンループ制御系でも任意のタイミングで同期パターンを変更できるといえる．

また，図 4-3 の結果を用いて物理シミュレータ上のモデルを動作させてみた．シミュレーションモデルとしては，1 つの胴体部を中空で固定し，1 脚につき 1 節を有する 2 脚モデル（図 4-4）を使用する．ここで，2 脚モデルの構成プログラムを付録 1 プログラムリストの List.10 に，制御器の構成プログラムを List.11 にそれぞれ示す．ここでは，　2 結合 VDP 方程式の近似解(7)，(8)式によって，角度および角速度の動作パターンを設計し，図 4-5 に示すようにモデルの joint1 と joint2 に出力を行った．角速度の動作パターンを出力した結果，設定した切り替えタイミング$t = 7$[s]で同期パターンが切り替わることを確認した．

しかし，オープンループ制御系では一意な歩行パターンを作成しているに過ぎず，現在の状態は考慮されていない．よって，GA による手法と同様に周囲の環境が変化した場合には再設計が必要となる．そのため，周囲の環境の変化に対応でき，安定した歩行が行える制御系として，次節の 4.4 節でフィードバック制御系について検討を行った．

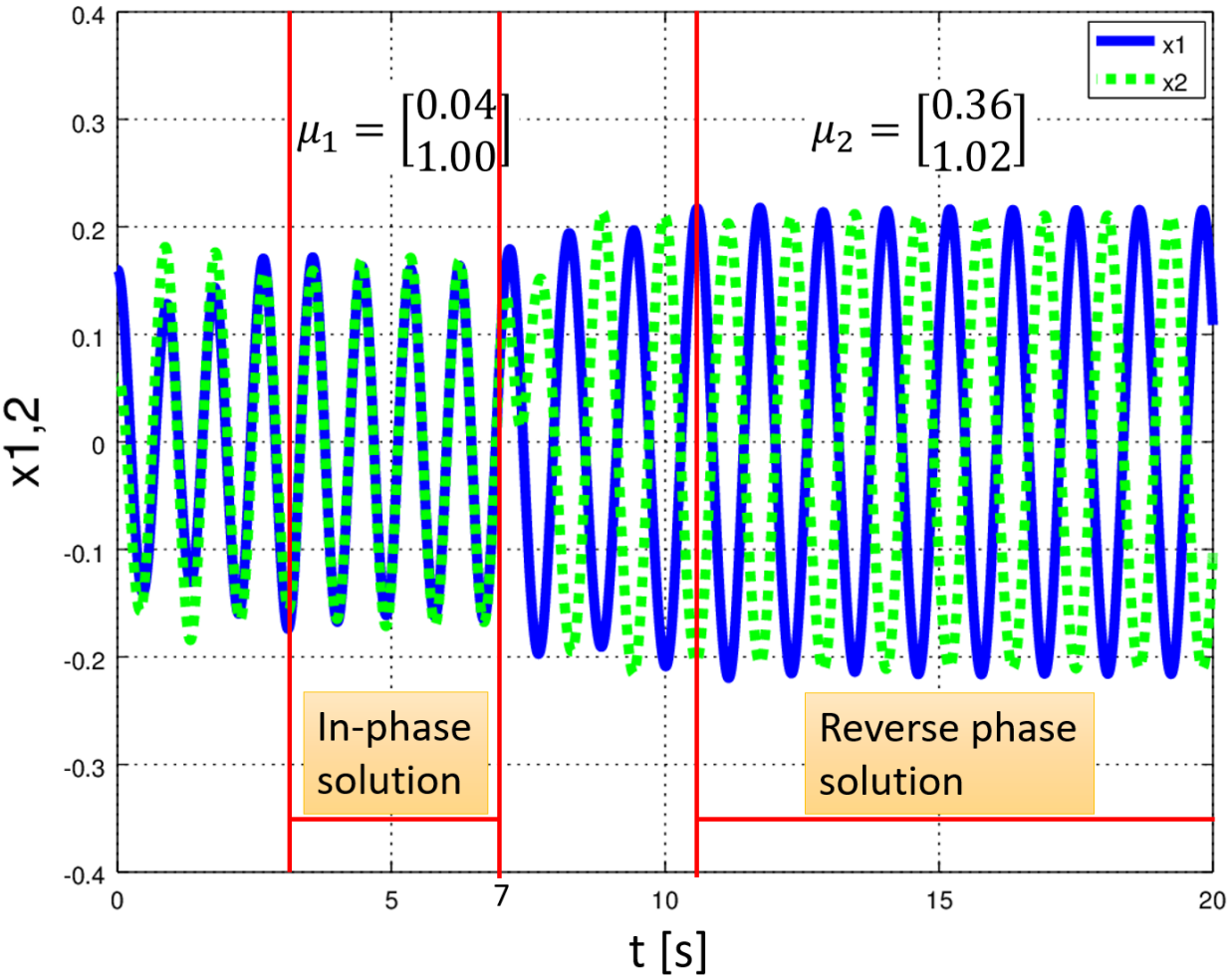

図 4-3 オープンループ制御系のシミュレーション結果

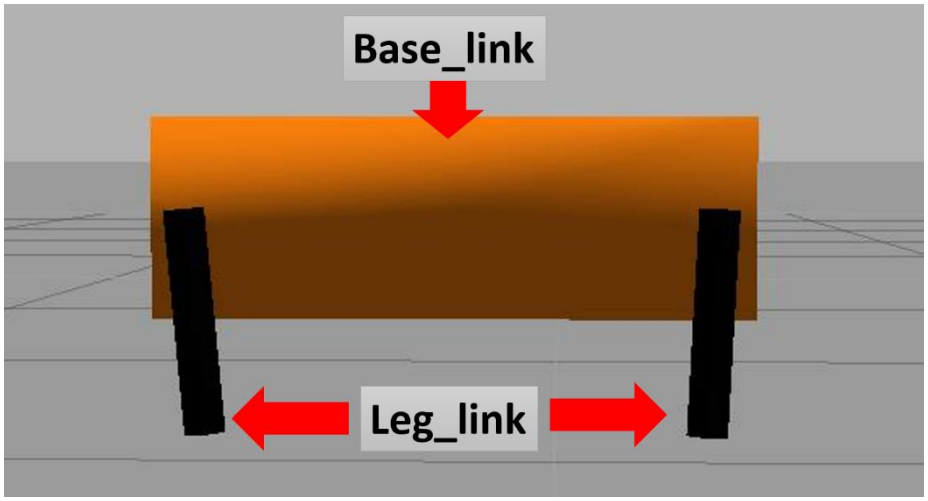

図 4-4 シミュレーションモデル(2 脚モデル)

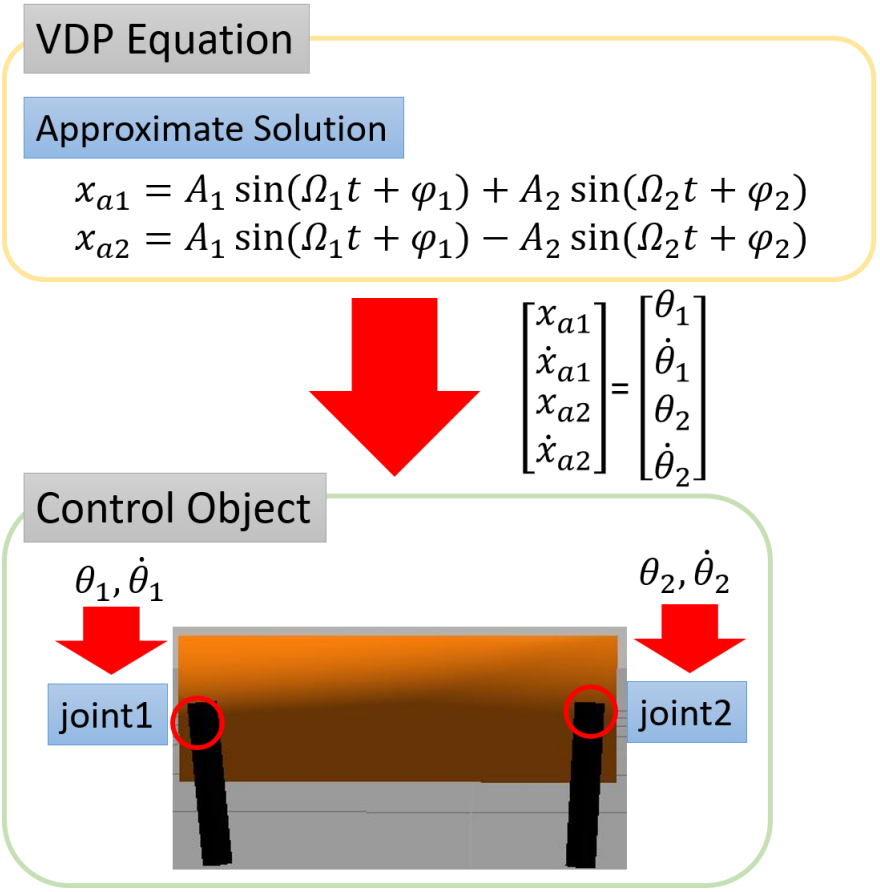

図 4-5 オープンループ制御系のシミュレーション概要

4.4. フィードバック制御系

周囲の環境の変化に対応でき，安定な歩行が可能な制御系として以下の図 4-6 のフィードバック制御系を考える．また，制御対象としては 4.3 節の図 4-4 のモデルを使用する．このモデルの 2 つの脚の角度：θおよび角速度：$\dot{\theta}$を取得し，制御入力値：u_iを算出する．ここで，各脚の運動モデルは以下の(21)式を使用した[11]．

$$\begin{cases} \ddot{\theta}_i + {\omega_0}^2 \theta_i = G u_i \ (i = 1,2) \\ G = \dfrac{1}{I + ml^2}, \omega_0 = \dfrac{mgl}{I + ml^2} \end{cases} \tag{21}$$

(21)式の各定数については参考文献[11]を参照し，表 4-2 に示した値を使用した．また，制御入力値：u_iについても参考文献[11]を参照し，以下の(22)式を使用した．

$$u_i = kv_i \cdot \alpha_i + F_i \tag{22}$$

(22)(21)式のkv_iについては参考文献[11]より次式の(23)式となる．

$$\begin{cases} kv_i = kp\left(Ed - \dfrac{2}{3}E\right) \\ Ed = 1.5 \times 10^{-5}, kp = 5.93 \end{cases} \tag{23}$$

ここで，Edは目標エネルギーを，Eについては(24)式で表せられる振り子の運動エネルギーである．

$$E = \frac{1}{2}(I + ml^2){\dot{\theta}_i}^2 \tag{24}$$

また，$\gamma = \frac{I+ml^2}{3Ed}$とおき，(23)式に(24)式を代入すると$kv_i$は(25)式のようになる．

$$kv_i = kp \cdot Ed\left(1 - \gamma \cdot {\dot{\theta}_i}^2\right) \tag{25}$$

次に，(22)式のα_iについては参考文献[11]より(26)式で表される．

$$\alpha_i = \dot{K}_{AGx_i} \cdot \dot{x}_i \tag{26}$$

ここで，(26)式の$\dot{K}_{AGx_i}$についてはオートコントロールのゲインであり，4.3 節の(20)式と同じである．次に(22)式の外力：Fについては，参考文献[11]より(27)式のように表される．

$$\begin{cases} F = -k_{m1}\left(\displaystyle\sum_{s=1}^{2} x_s - x_i\right) + k_{m2} \cdot F_{fi} \\ F_{f1} = 0, F_{f2} = -\sin\left(\varphi_d\right) \cdot K_{AGx_1} \cdot x_1 + \cos(\varphi_d) \cdot \dot{K}_{AGx_1} \cdot \dot{x}_1 \end{cases} \tag{27}$$

この(27)式によって，モデルの各脚(振動子)は互いに結合される．また，F_{fi}は周期外力であり，同期パターンの変更に用いる．k_{mi}については 4.3 節の(16)式を用いる．ここで，4.3 節と同様に(16)式における j の値は目標とする同期パターンによって決まり，A_{mj}およびA_{tj}は以下の(28)式, (29)式となる．また，φ_dについては目標とする同期パターンの位相差を表し 0 またはπをとる．

・同相解での同期($\varphi_d = 0, j = 1$)

$$\begin{cases} A_{m1} = \sqrt{\left(\dfrac{x_1 + x_2}{2}\right)^2 + \left(\dfrac{\dot{x}_1 + \dot{x}_2}{2{\Omega_1}'}\right)^2} \\ A_{t1} = \sqrt{\dfrac{2}{3\gamma {\Omega_1}'^2}} \\ {\Omega_1}' = \sqrt{{\omega_0}^2 + Gk_1} \end{cases} \tag{28}$$

・逆相解での同期($\varphi_d = \pi, j = 2$)

$$\begin{cases} A_{m2} = \sqrt{\left(\dfrac{x_1 - x_2}{2}\right)^2 + \left(\dfrac{\dot{x}_1 - \dot{x}_2}{2{\Omega_2}'}\right)^2} \\ A_{t2} = \sqrt{\dfrac{2}{3{\Omega_2}'^2}} \\ {\Omega_2}' = \sqrt{{\omega_0}^2 - Gk_1} \end{cases} \tag{29}$$

ここで，ω_0については以下の(30)式を使用した.

$$\omega_0 = \sqrt{\frac{mgl}{I + ml^2}} \tag{30}$$

また，x_iおよび$\dot{x}_i$については参考文献[11]より以下の VDP 方程式を使用した.

$$\begin{cases} \ddot{x}_i - \varepsilon\left(1 - {\dot{x}_i}^2\right) + \Omega^2 \cdot x_i = k_g \dot{\theta}_i \\ \Omega = 17.3, k_g = 20 \end{cases} \tag{31}$$

よって，(22)式の制御入力値 : u_iを制御対象へフィードバックすることで，周囲の環境が変化しても安定な周期を保つことが可能だと考えられる．作成したフィードバック制御プログラムを付録 1 プログラムリストの List.12 に示す.

制御シミュレーション結果を図 4-7 に示す. ここではシミュレーション時間tを$t = 0$~10 [s]とし，サンプリングタイムは 10[ms]に設定した．εについては 4.2 節の表 4-1 を使用した．また，同期パターンの切り替えタイミングとして,$t = 5$[s]で同相解から逆相解への切り替えを行った. 図 4-7 より，$t = 5$[s]で同相解から逆相解への変更が出来ていることを確認した．同様の設定で，逆相解から同相解の切り替えについてもシミュレーションを行い，同期パターンの制御が可能であることも確認した．よって，フィードバック制御系でも，任意のタイミングで同期パターンを制御することが可能であると言える.

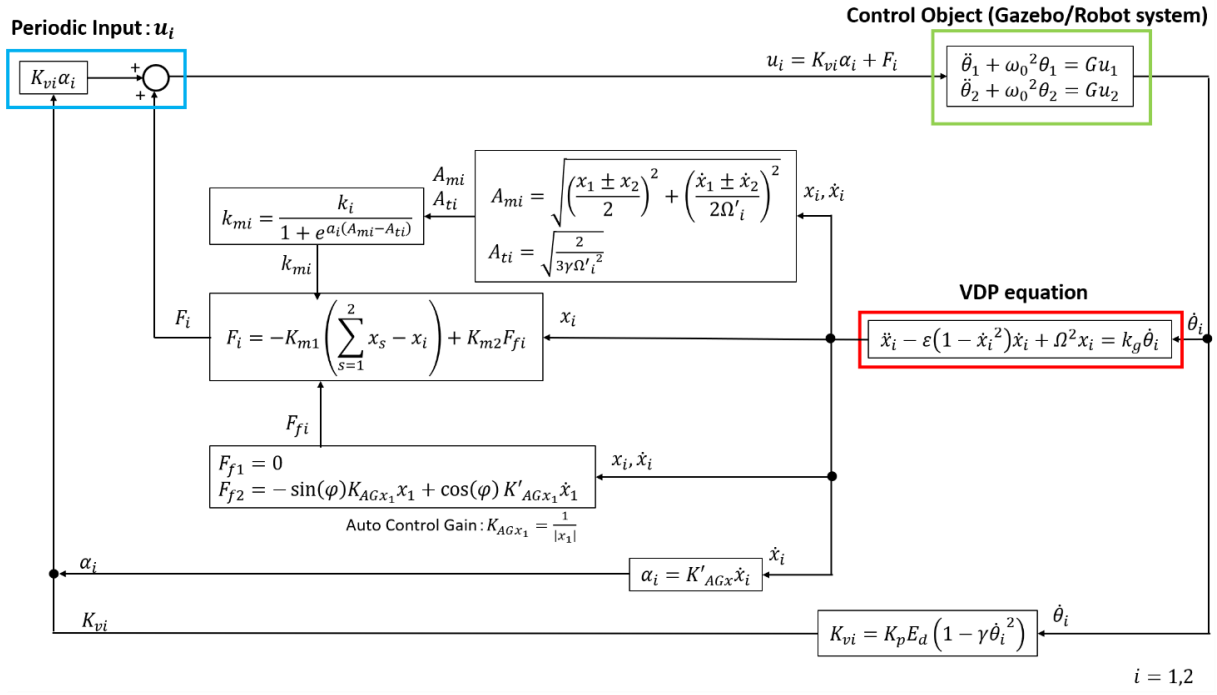

図 4-3 フィードバック制御系（2 結合モデル）

表 4-2 モデルの運動方程式における各定数値

m：モデル重量[kg]	0.12
I：脚の重心周りの慣性モーメント[kgm^2]	1.45×10^{-5}
l：脚の長さ[m]	0.011
g：重力加速度[m/s^2]	9.80

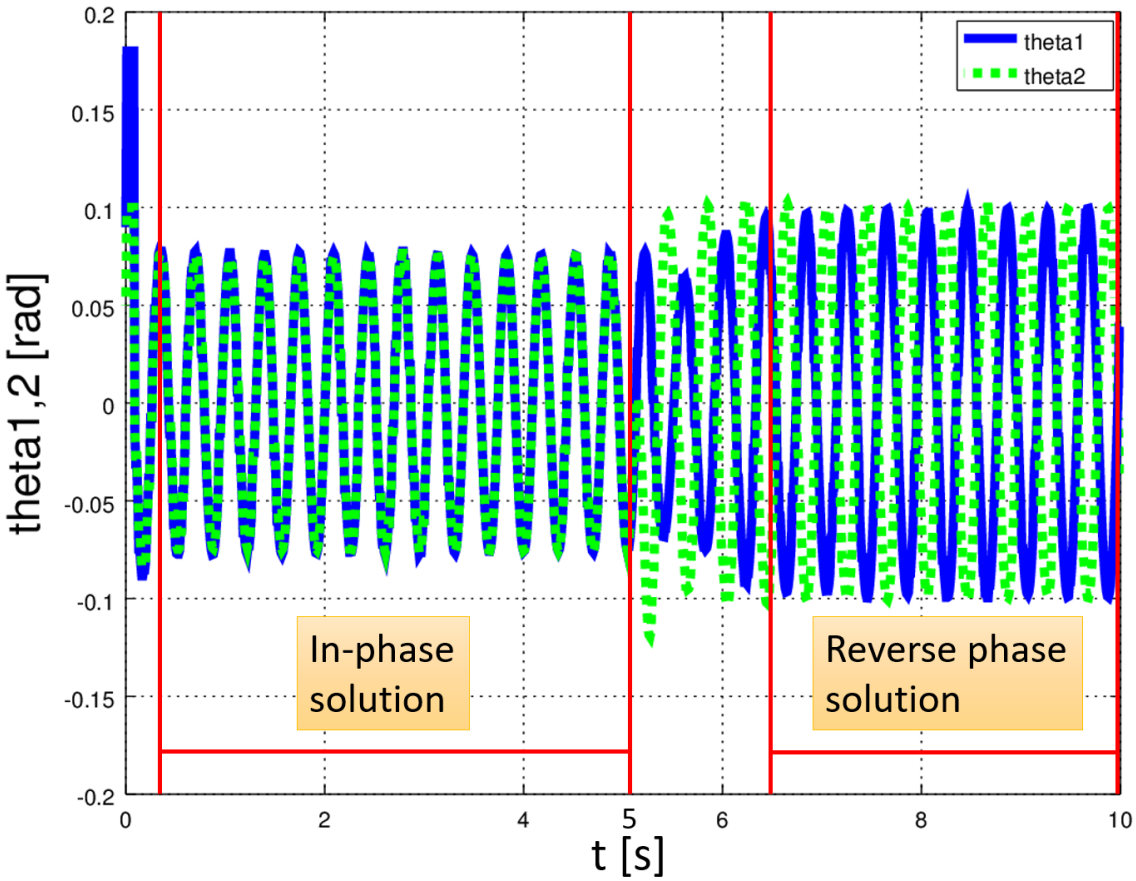

図 4-4 フィードバック制御システムのシミュレーション結果

4.5. 検討結果のまとめ

本章では，多足歩行ロボットの運動制御に 2 結合 VDP 方程式を用いることについて，オープンループ制御系とフィードバック制御系を作成しそれぞれ検討を行った．

オープンループ制御系では，フロケの理論より安定な周期解が作成できることを確認できた．また，周期外力を加えた 2 結合 VDP 方程式を用いることで，任意のタイミングで同期パターンを変更できることについても確認した．しかし，オープンループ制御系では一意な歩行パターンを作成しているに過ぎず，現在の状態は考慮されていない．よって，GA による手法と同様に周囲の環境が変化した場合には再設計が必要となるため，実際の運動制御としては現実的ではない．

そのため，周囲の環境の変化にも対応するため，フィードバック制御系について検討を行った．その結果，任意のタイミングで周期解の同期パターンを変更できることを確認した．これにより，2 結合 VDP 方程式を用いれば図 4-8 のように環境に合わせて同期パターンを切り替えることが出来る．したがって，干潟の移動も自由にできると考えられる．しかし，周期解の安定性については，VDP 方程式の近似解を用いていないためフロケの理論による判別が行えなかった．フィードバック制御系で周期解の安定性を判別する方法については現在検討中である．

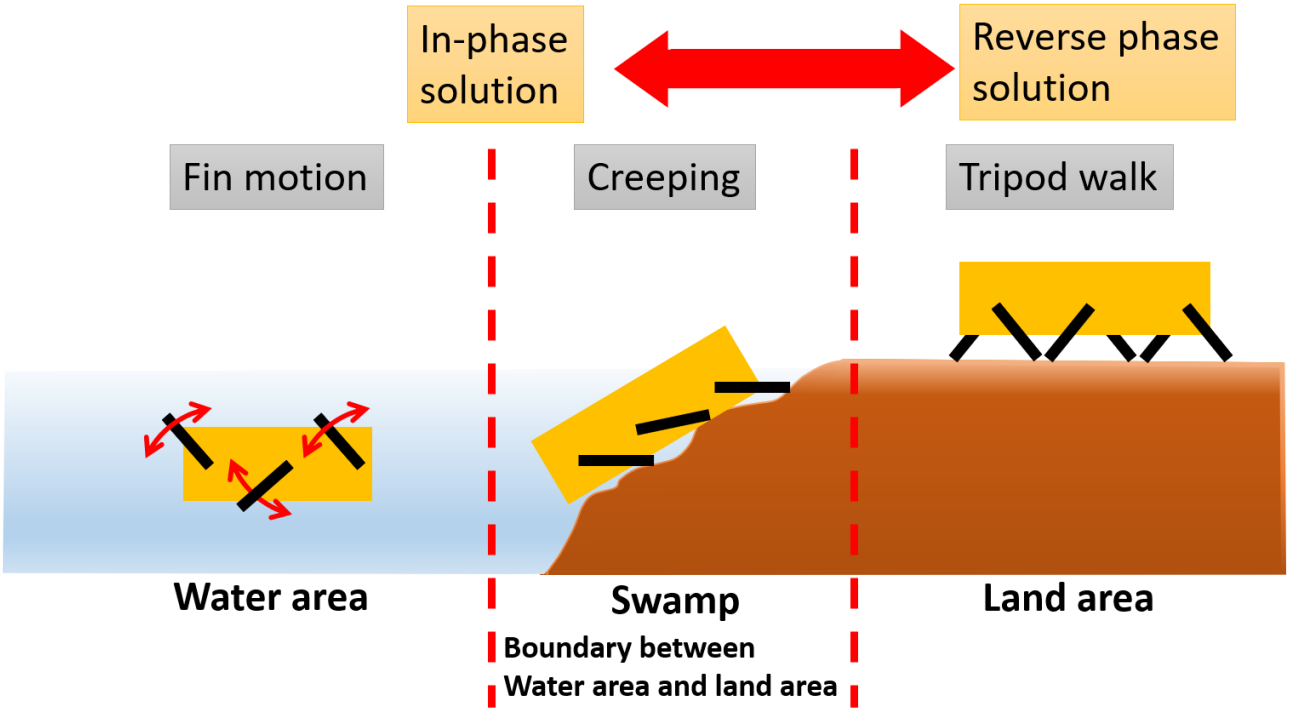

図 4-5 非線形同期を用いた多足歩行ロボットの運動制御の概要図

5. 多足歩行ロボットの実機の作成

3 章より，6 脚 6 節の多足歩行型ロボットが陸上で安定した歩行が可能であることは，シミュレーションによって確認した．次に，ROS の特徴を活用して，今回作成した制御システムによって実機の制御も可能であるかを試験する．そのため，陸上での動作を目的とした 6 脚 6 節の歩行型ロボットを作成する．これまでに，6 脚 6 節の歩行型ロボットの機構部分についてはほとんど作成が完了しており，詳細については 5.1 節に示す．現在は，モータ制御回路の作成を行っており，まずは，ロボットの 1 脚のみの制御を目的として 1 脚制御試験装置(5.2 節)を作成した．この装置を用いてマイコン側の制御プログラムの作成を行った．また，ROS の機能を用いたシリアル通信によって，PC とマイコン間で通信が行えることを確認した．この結果については，5.3 節に示す．

5.1 6 脚 6 節の歩行型ロボット（機構部分）の構成

作成した 6 脚 6 節の歩行型ロボット(機構部分)を図 5-1 に示す．全体のサイズとしては ROS でのシミュレーションモデルを参考に表 5-1 の値で設計した．胴体部分については，30[㎜]×30[㎜]のアルミフレーム角材を使用しメインフレームとした．脚部については，モータの軸を異径カップリングによって延長し，脚部パーツの取り付けを行った．モータの仕様については以下の表 5-2 に示す．このモータは，自重を支えられること，および今後搭載する機器が増えることを考慮して高トルクが得られる土佐電子製のエンコーダ付き DC モータ(PG188-E)を使用した．脚部パーツについては構成図を図 5-2 に示す．フレーム部分は，5[㎜]厚のアクリル板を図 5-2(正面図)の形状に加工したものを使用した．このフレームを 2 枚作成し，シャフトホルダーを挟むように取り付け，さらに地面と接する先端に保護用のゴムキャップを付けて脚部パーツを作成した．脚部の長さについては，フレームの材質がアクリルであることを考慮し，軸中心から約 0.20[m]とした．以上のような構成で 6 脚 6 節の歩行型ロボットの機構部分を作成した．ここで，作成した脚部フレームの設計図面および多足歩行ロボットの部品表を付録 2 に示す．

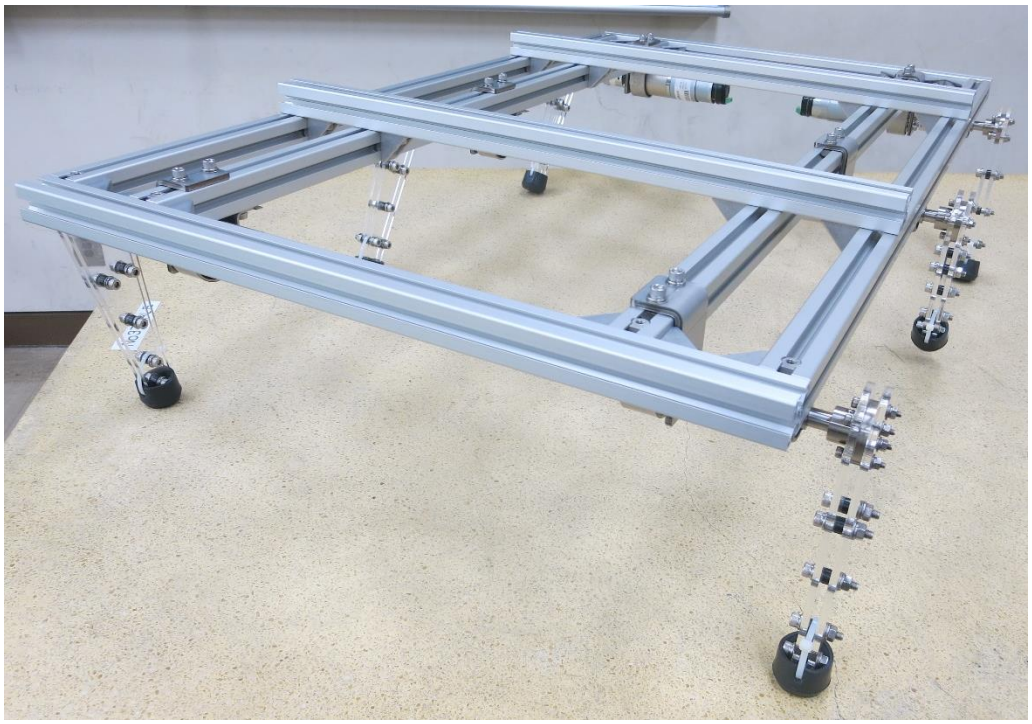

図 5-1 6脚6節の歩行型ロボット（機構部）

表 5-1 歩行ロボットのサイズ

全長［m］	1.03
全幅［m］	0.66
全高(足を下に向けた状態)［m］	0.25
総重量［kg］	18.20

表 5-2 DCモータ(PG188-E)の仕様

停動トルク[N・m]	44.80
電圧[V] (DC)	12.00
無負荷電流[A]	0.60
停動電流[A]	22.00
無負荷出力［W］	7.20
重さ[kg]	0.95
無負荷回転速度[rpm]	28.00
減速比	188.00
全長[mm]	188.32
軸径 ϕ ［mm］	10.00
モータ直径 ϕ ［mm］	45.00

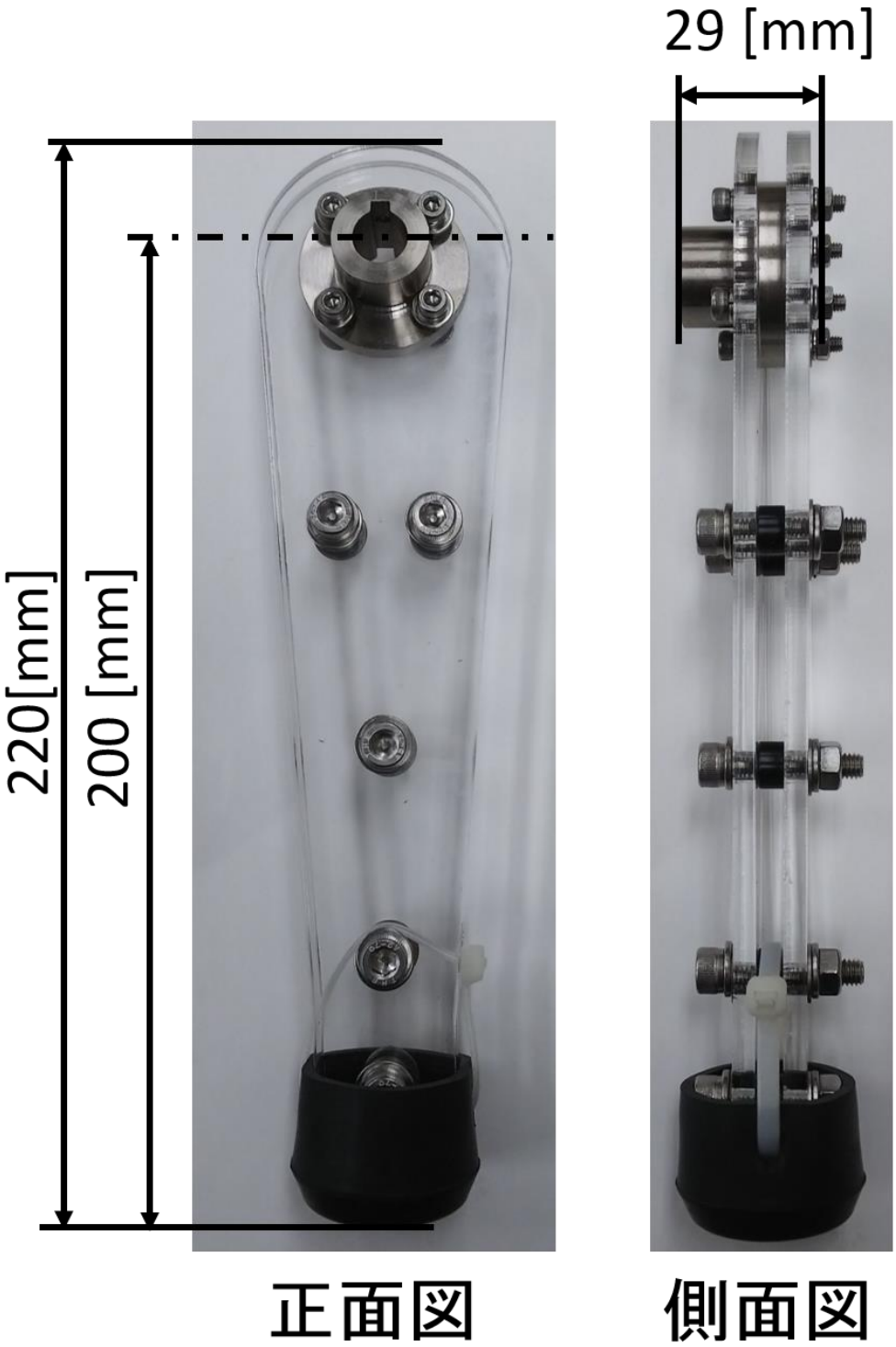

図 5-2 脚部フレームの構成図

5.2. 1脚制御試験装置を用いたモータ制御回路の作成

次に，脚部モータの制御回路の作成を行う．まずは，1脚分の制御のみを目的とした1脚制御試験装置を作成した．ここで，作成した装置を図5-3に，装置の概要を表すブロック線図を図5-4にそれぞれ示す．

この装置ではUSBシリアル変換基板を用いてPCと制御回路間のシリアル通信を行う．よって装置の主な制御回路としては，モータ制御回路，I2Cデジタル電流，電圧，電力計モジュール，モータドライバ回路，光センサ回路の4つで構成されている．また，モータ駆動電源としては安定化直流電源を使用し，モータ駆動電圧12[V]を供給した．モータの制御回路には，マイコンとしてATmega328P-PUを使用した．I2Cデジタル電流，電圧，電力計モジュール(INA226)はモータ駆動電源とモータドライバ間に配線した．これによって，モータドライバ回路へ供給される電圧[mV]，電流[mA]，電力[mW]を測定する．モータドライバ回路では，モータの無負荷時電流と停動電流の値を考慮し，最大定格電流が出来るだけ大きなものを選定し，最大定格電流が8.0[A]であるTOSHIBA製のDCモータドライバ(TB67H400AHG)を使用した．光センサ回路は赤外線を用いたフォトカプラ(RPR-220)を使用し，モータの原点位置を定めるために作成した．そのため，脚部フレームには赤外線が反射するように，通過個所を白いビニルテープで覆った．これらの回路については，回路図を付録3に示す．

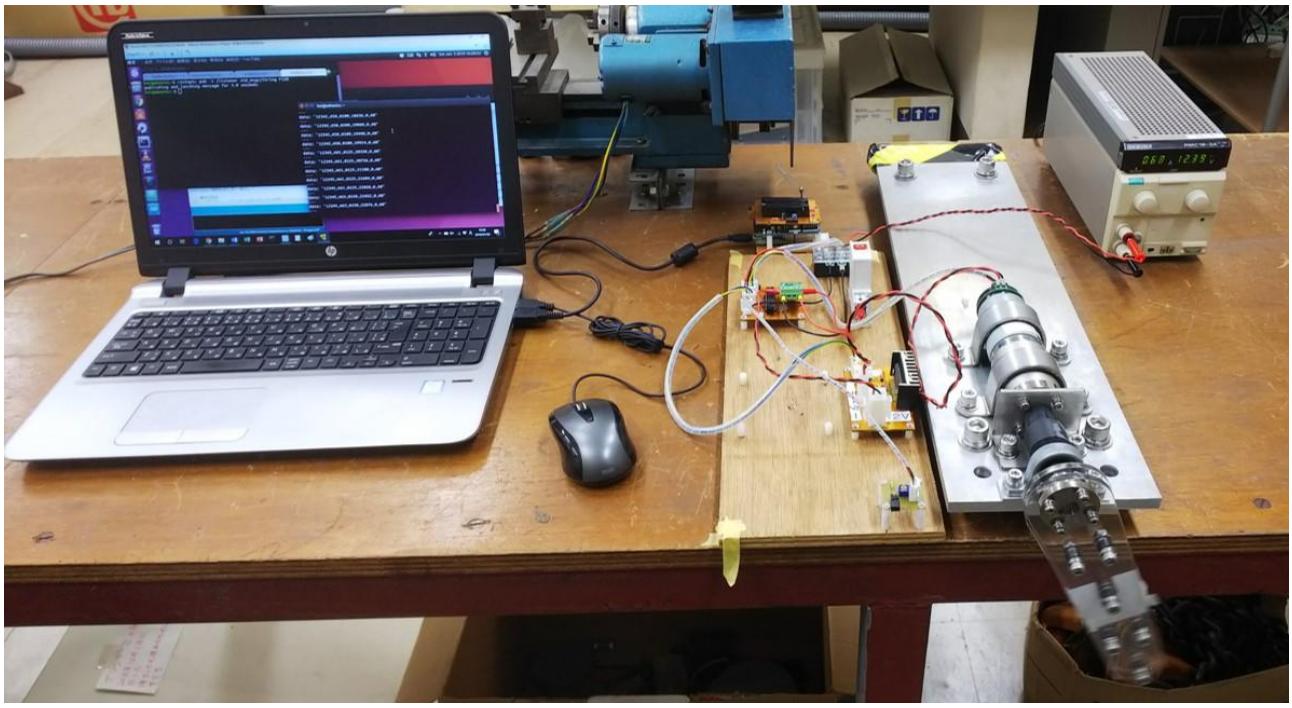

図 5-3 1 脚制御試験装置

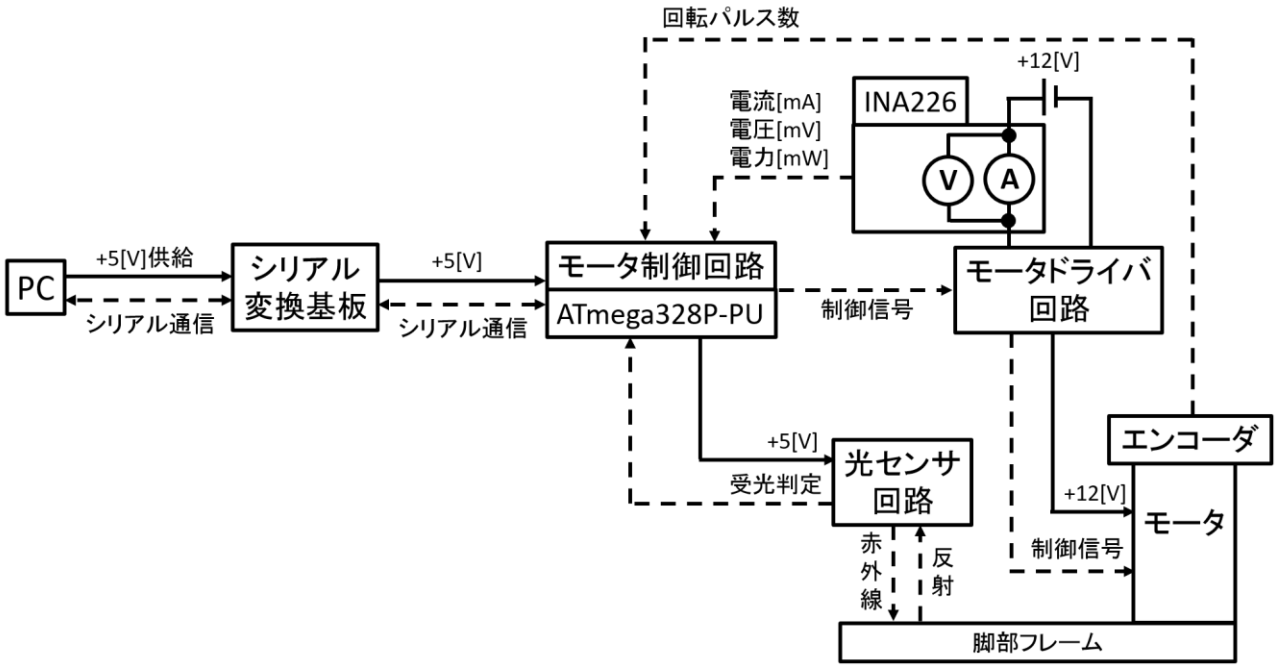

図 5-4 1 脚制御試験装置のブロック図

5.3. モータ制御プログラムの作成

5.2 節の図 5-3 の装置を用いて 1 脚の動作試験を行い，制御回路のマイコンで使用する制御プログラムの作成を行った．

この試験装置では，ROS によるシリアル通信機能を用いてモータの制御を行う．ここで，モータ制御の概要図を図 5-5 に示す．ROS のシリアル通信機能(ROS Serial)によってモータ制御回路側で通信用のノード(Serial Node)を作成する．この Serial Node と PC 側の通信用ノード(Control Node)を用いてシリアル通信を行う．制御回路側では，エンコーダよりモータの回転パルス数をカウントし，これを用いて回転角度:θを算出する．ここで,θの値は光センサの位置を基準として,0 から 360[deg]あるいは-360[deg]から 0[deg]の範囲となるように算出する．同時に，モータの角速度を算出するためにプログラム実行周期ごとのパルス数の差分値:$dpulse$も計算する．また，INA226 よりモータ駆動用電源(+12[V])とモータドライバ回路間の電圧：V[mV]，電流：I[mA]，電力：W[mW]も取得し，これらの値を PC 側に送信する．PC 側では，制御回路から送られてきたデータの受信，およびモータの回転速度指令値(duty 比)の送信を行う．

このような動作を行うための制御プログラムを，制御回路側では Arduino IDE を用いて，PC 側では python を用いてそれぞれ作成した．また，制御回路では Timer 関数を用いてサンプリングタイム 100[μsec]でエンコーダからのパルス数の取得を行い，50[msec]の間隔で PC 側への測定値の出力を行った．これらの制御プログラムを用いて，モータを正転方向で定速回転させ制御回路より所望のデータが取れるかを試験した．その結果，制御回路から INA226 の測定値，モータ回転角度および回転パルス数の差分値を PC 側で確認することができた．また，モータを逆転させた場合も同様に試験を行ったが，こちらでも問題なく制御回路からの出力値を確認することができた．この結果より，制御回路側のプログラムは完成できたと言える．ここで，作成したモータ制御システムおよびモータ制御回路のマイコンに搭載したプログラムを付録 1 のプログラムリスト List.13 に示す．

今後は，3.2.1.1 節の図 3-5 のように制御パラメータを定義し，角速度制御を行うことを目標として PC 側での制御プログラムの作成を行う必要がある．具体的な内容としては，取得した角度より現在の脚の状態を算出し，現在の脚が接地脚または遊脚状態であるかを判断しモータに出力する角速度を算出する．そして，算出した角速度を制御回路に出力するような制御を行う．また，制御プログラムの完成後には，これを 6 脚に拡張し，実機の歩行試験を行う予定である．

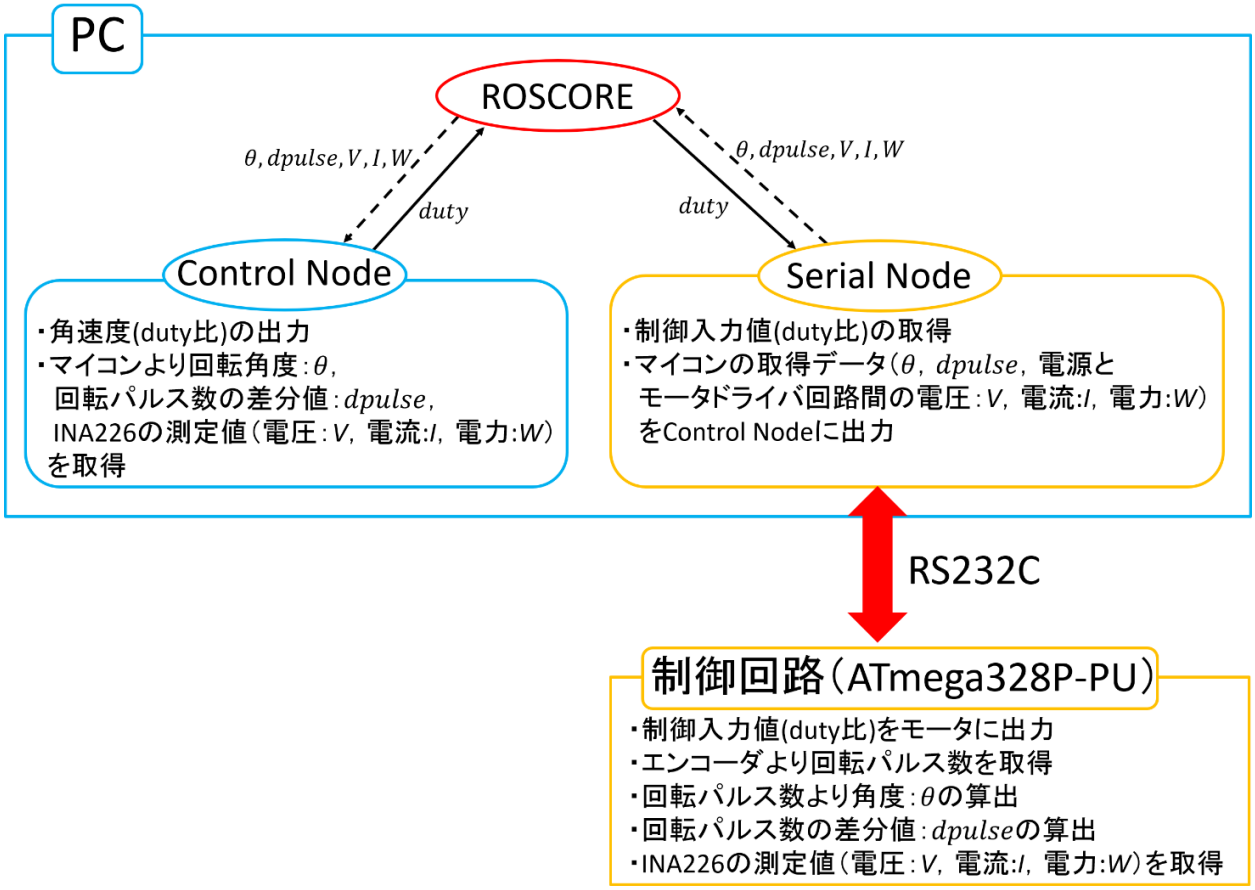
PC
ROSCORE
θ, dpulse, V, I, W
duty
θ, dpulse, V, I, W
duty
Control Node
・角速度(duty比)の出力
・マイコンより回転角度：θ，
回転パルス数の差分値：dpulse，
INA226の測定値（電圧：V，電流:I，電力:W）
を取得
Serial Node
・制御入力値(duty比)の取得
・マイコンの取得データ（θ，dpulse，電源と
モータドライバ回路間の電圧：V，電流:I，電力:W）
をControl Nodeに出力
RS232C
制御回路（ATmega328P-PU）
・制御入力値(duty比)をモータに出力
・エンコーダより回転パルス数を取得
・回転パルス数より角度：θの算出
・回転パルス数の差分値：dpulseの算出
・INA226の測定値（電圧：V，電流:I，電力:W）を取得

図 5-5 脚制御の概要図

6. μ-ASV

本章ではμ-ASV モデルが任意の目標座標まで移動し，目標座標近傍で定点保持およびヘディング制御を行うような運動制御手法(簡易 DPS 制御)について検討した．

まずは，シミュレーションモデルの作成を行った．通常の船体運動は複雑であるため，シミュレーションモデルでは陸上で船体運動に似た動作が行えるように，スラスタの代わりにオムニホイールを使用した．これによって，運動モデルの簡略化を図った．詳細については 6.1 節に示す．次に，目的とする簡易 DPS 制御のシステムについて検討した．また，制御システムの単純化を目的として，スライディングモード制御を制御手法として用いた．それぞれの概要については 6.2 節および 6.3 節にそれぞれ示す．

次に，このスライディングモード制御を用いて制御器設計を行う．ここでは，角度制御と直線距離制御の各制御器をそれぞれ個別に作成した．その結果を 6.4 節に示す．最後に，スライディングモード制御の特性を活かしてこれらの制御器を組み合わせて，簡易 DPS 制御システムの作成を行った．作成したシステムを用いた結果，モデルは任意の目標座標への到達を確認できた．この結果を 6.5 節に示す．

6.1. モデル構成

本研究で作成を予定しているμ-ASV の構成イメージを図 6-1 に示す．μ-ASV の実機としては，サーフボードと 3 機のスラスタ，通信装置などにより構成されるものを想定している．これにより，機体の価格を数万円程度の安価なものに抑えることができ，シンプルな構造であるため故障時の修復も容易となる．しかし，この機体と同じシミュレーションモデルを作成しても，船体運動は運動モデルが複雑であるため解析が困難となる．また，3.4 節でも触れたように現状の Gazebo では水域のシミュレーション環境が整っていない．そのため，図 6-2 に示すように船体モデルを台車モデルで近似し，運動モデルの簡略化を図った．

台車モデルは陸上で船体運動が再現できるものとして設計した．そのため，スラスタの代用として全方位へ移動できるオムニホイールを使用した．オムニホイールとは図 6-2 に示したもので，1 つの車輪に対し複数の樽型ローラが車軸に対して垂直にフリージョイント接続で取り付けられている．したがって，車軸に拘束されることなく自由な移動が可能となる．これにより，船体の運動を Gazebo 上で疑似的に再現した．また，通常の船体はラダーを搭載し船体角度を制御するが，本研究の船体モデルはバウスラスタ 1 機とメインスラスタ 2 機を搭載したものを考えている．それぞれのスラスタについては，バウスラスタは角度制御に，メインスラスタは前後の推進のみに使用する．そのため，モデルには前方に 1 つと後方に 2 つ，計 3 つのオムニホイールを搭載した．図 6-3 に作成した台車モデルを示す．このモデルは実機に使用する予定であるオムニホイールを参考に全長 0.980[m]，全高 0.323[m]，総重量 34.18[kg]とした．ここで，μ-ASV のシミュレーションモデルの構成プログラムについては付録 1 プログラムリストの List.14 に，各オムニホイールの制御コントローラについては List.15 にそれぞれ示す．

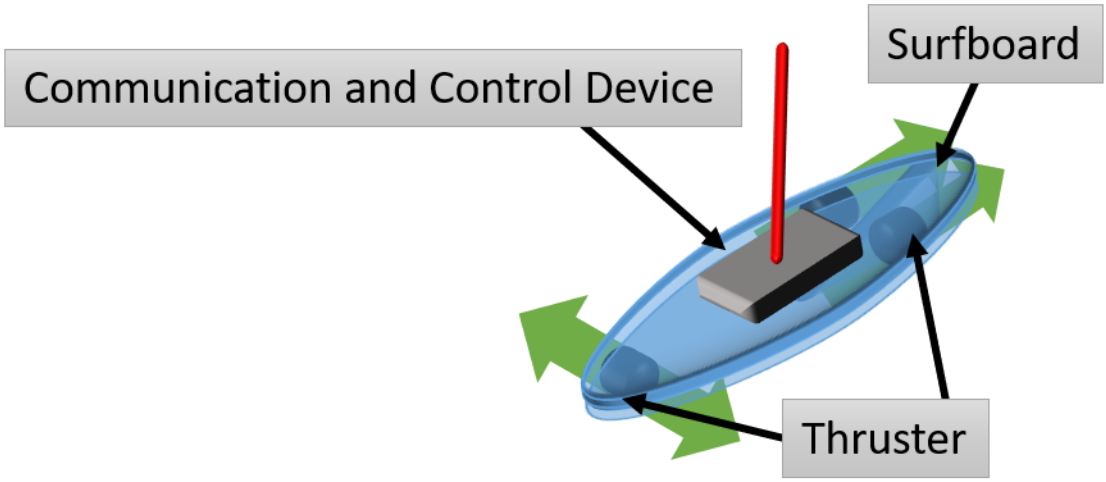

図 6-1 μ-ASV の実機構成イメージ

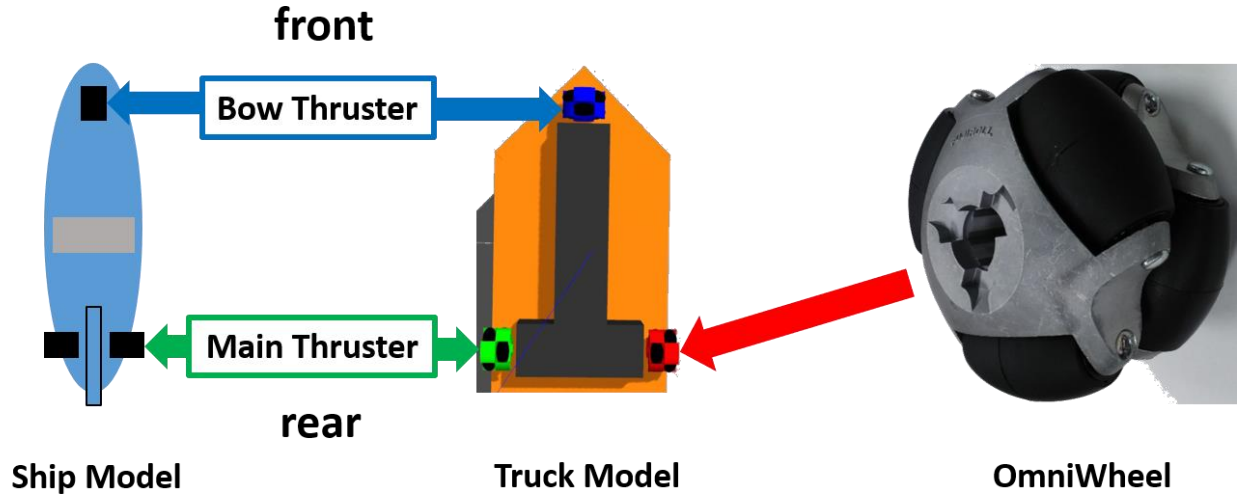

図 6-2 船体モデルの簡略化（台車モデルへの置き換え）

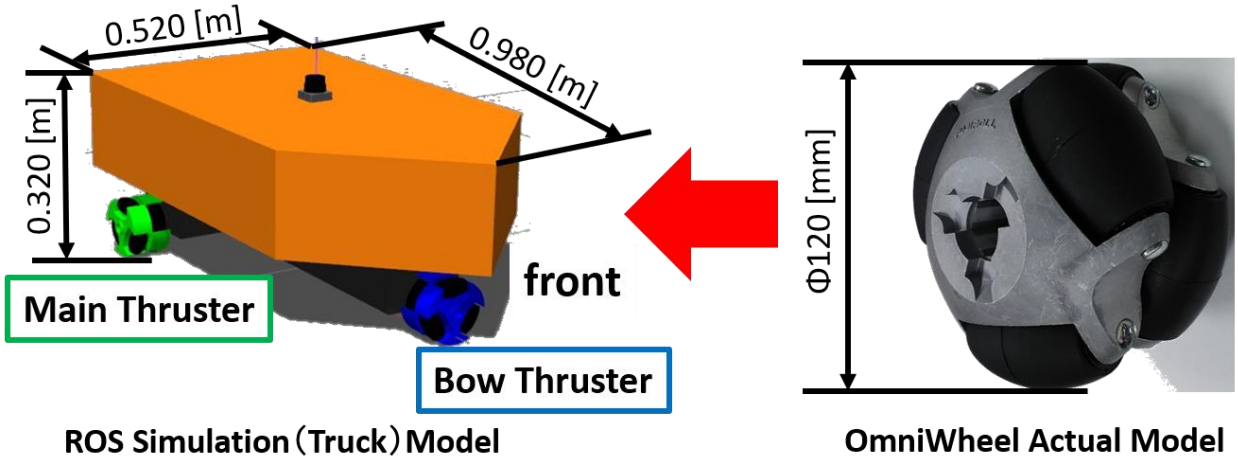

図 6-3 台車モデルの概要図

6.2. 簡易 DPS 制御の概要

本章で目指す台車モデルの運動制御システムの概要を図 6-4 に示す．本制御システムは，目標とする座標点が現在位置から数[m]程度の場所にあると想定している．台車モデルは現在位置からスタートし，任意の目標座標点 P に移動する．また，P の半径 1.5[m]に入った場合には，船体を North 方向(図 6-4 中では x 軸+方向)へ固定するヘディング制御を行う．このような簡易的な DPS(Dynamic Positioning System)制御を行うシステムの開発を目指す．ここで，台車モデルには以下の手順に従って簡易 DPS 制御を行う．

Step1 ：目標座標点 P の探査

・台車モデルの前方に±45[deg]の視界を設定し，点 P が視界内になければその場回頭を続ける．この時，台車モデルがその場回頭する方向は，点 P と現在地点のなす角度が最小となる方向へ回転する．

・簡易モデルの視界内に点 P が入った場合には step2 へ進む．

Step2 ：点 P への移動

・台車モデルは，前方のオムニホイールで船体の角度制御を，後方の 2 つのオムニホイールで前後推進をそれぞれ行う．よって，点 P への移動には角度制御と直線距離制御の 2 つの制御が必要となる．

・台車モデルが目標点の半径 1.5[m]のエリアに到達した場合には Step 3 へ進む．

Step3：ヘディング制御

・船体の目標制御角度を 0[deg]とし，船首を North 方向（図 6-4 中では x 軸＋方向）に向けて定点保持を行う．

Step2 で示したように，この運動制御にはモデルの角度制御および直線距離移動の制御が必要となる．そこで，具体的な制御手法としてスライディングモード制御（Sliding Mode Control）[6]について検討した．この結果については，6.4 節に示す．

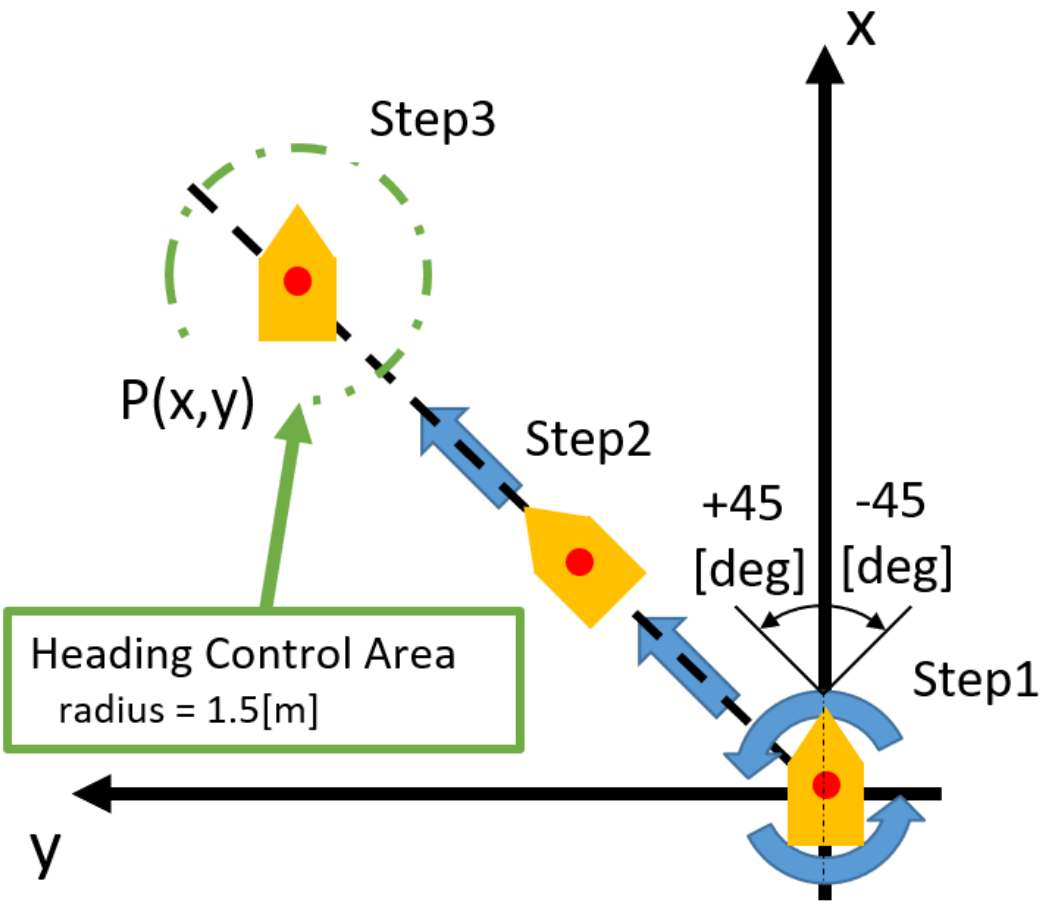

図 6-4 簡易 DPS 制御の概要

6.3. スライディングモード制御の概要

スライディングモード制御の概念図を図 6-5 に示す．スライディングモード制御とは，位相平面上にある任意の点 p から始まる軌跡を原点に収束させる制御法である．任意の点 p から始まる軌跡には(32)式の制御入力uを用いている．

$$\begin{cases} u = -K \cdot \mathrm{sgn}(\sigma) \\ \sigma = S \cdot X \end{cases} \tag{32}$$

ここで，(32)式の各変数はK：切換ゲイン，σ：切換関数，S：超平面，X：状態量である．切換関数σは，図 6-5 に示すように任意の位相平面上に設計した超平面Sを境界として極性が変わる．これによって制御入力uの極性も切り換わるため，任意の点 p から始まる軌跡を直線S上に引き込み，原点に近づけていく．そのため，これを船体角度および直線距離の制御に応用することで，モデルを目標座標点に収束させることが可能ではないかと考えている．また，スライディングモード制御のシステム上での挙動は以下の 3 つのモードに大別することができる[6]．

Mode1：到達モード

・任意の点 p から超平面Sに到達するまでの区間

Mode2：スライディングモード

・状態が超平面Sの直線上に拘束され原点に向かう

・システムのダイナミクスはSを記述するパラメータのみに唯一支配される

Mode3：定常モード

・最終的な周期状態（リミットサイクル，一定のオフセット状態など）

特に Mode2：スライディングモードの特性である，‘システムのダイナミクスはSを記述するパラメータのみに唯一支配される’[6]が本制御の大きな特徴である．これによって，スライディングモード制御はモデル誤差や外乱に対して優れたロバスト性を持つ制御法となっている．また，この優れたロバスト性を利用することで，別々に設計した制御器を組み合わせて使用することが可能となる．ここで，スライディングモード制御のシステム構成の概略図を図 6-6 に示す．今回の場合には，角度制御と直線距離制御の 2 つの制御を行うことになる．制御対象が不明な運動モデル$f_{(\theta,x)}$であるとすると，PID 制御などの一般的な制御手法では 1 つの制御器を用いて角度θと距離xを制御しなければならない．一方で，スライディングモードではロバスト性を利用することでモデル誤差などの影響を受けずに制御を行えるため，不明な運動モデル$f_{(\theta,x)}$であっても$f_{(\theta,x)} \fallingdotseq \{f_{(\theta)}, f_{(x)}\}$というように分けて考えることができる．そのため，制御を行う場合には角度θと距離xのそれぞれの制御器を作成すれば良くなり，制御器の設計が容易となる．

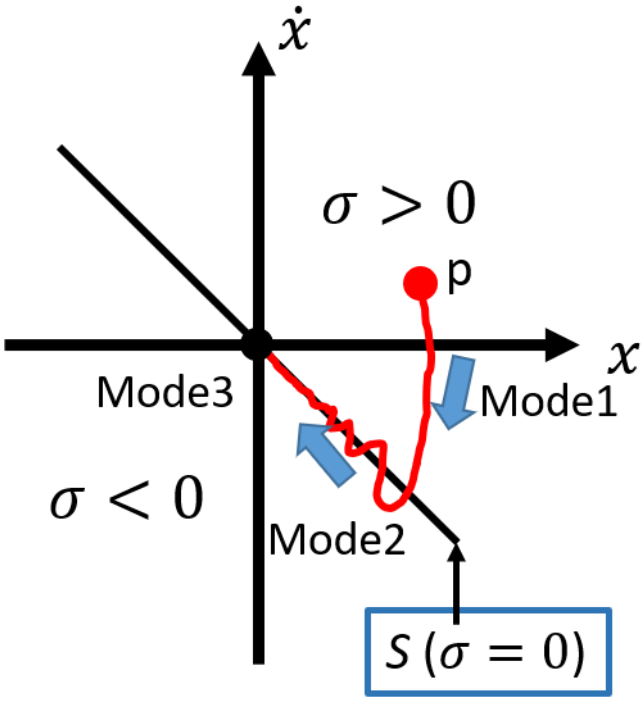

図 6-5 スライディングモード制御の概要図

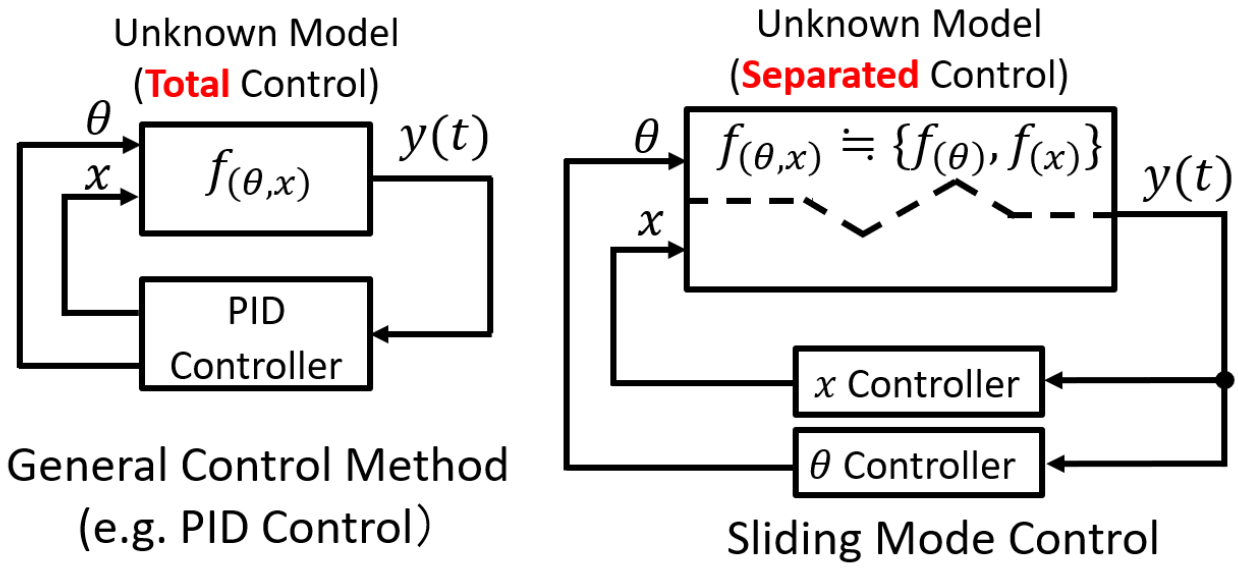

図 6-6 スライディングモード制御の制御システム構成

6.4. スライディングモード制御による制御器設計

スライディングモード制御を用いて，簡易 DPS 制御試験に必要となる角度制御および距離制御の各制御器を作成する．ここで，各制御器に必要なパラメータは，角度制御および直線距離移動の単独試験をそれぞれ行い，GA によって個別に設計した．角度制御器の設計については 6.4.1 節に，距離制御器の設計については 6.4.2 節にそれぞれ示す．

6.4.1. 角度制御器設計

角度制御の単独試験の概要を図 6-7 に示す．6.1 節にも示したように，角度制御ではモデル前方のオムニホイールのみを用いて制御を行う．運動モデルについては，前方のオムニホイールのみに制御入力を加えても，後方のフリージョイントで接続された樽型ローラが動いてしまうため，非線形性が強いのではないかと考えた．また，制御器の設計を簡単にするため，今回の検討ではスライディングモード制御の特性であるロバスト性の強さを活かしてモデルの角度制御を行うこととした．よって，今回は運動モデルを考慮せず，制御入力式としては以下の(33)式を使用する．

$$\begin{cases} u_1 = -K_1 \cdot sat(\alpha_1, \sigma_1) \\ sat(\alpha_1, \sigma_1) = \dfrac{\sigma_1}{|\sigma_1| + \alpha_1} \\ \sigma_1 = S_1 \cdot X_1 \\ X_1 = \left[(\theta - \theta_r), \dot{\theta}\right], S_1 = [S_1^{\theta}, 1] \end{cases} \tag{33}$$

ここで，(33)式の各変数はK_1：切換ゲイン，σ_1：切換関数，S_1：超平面，X_1：状態量，α_1：saturation 係数である．与える状態量X_1については，角度制御では目標角度：θ_rと現在角度：θの差分値：$(\theta - \theta_r)$と，モデル yaw 角(モデルの z 軸周り)方向の角速度：$\dot{\theta}$を用いる．そのため，超平面：S_1についても同じ次数となるように$S_1 = [S_1^{\theta}, 1]$とした．$K_1, \alpha_1, S_1^{\theta}$については未定数となるため，(34)式を用いて適応度：fが最大となるようなパラメータを GA により設計した．

$$\begin{cases} f = a\dfrac{1}{(\sum diff_{\theta})^2} + b \cdot count_{\theta} \\ a = 1.0 \times 10^8, b = 1.0 \times 10^1 \end{cases} \tag{34}$$

また，(34)式において，$diff_{\theta}(\fallingdotseq \theta - \theta_r)$：角度差分値（目標角度と現在モデル角度の差分値），$count_{\theta}$：角度差分値が 1[deg]以下の時の回数である．制御に必要なパラメータを求めるため，世代数を 10，個体数を 100 として GA シミュレーションを行った．設計した制御パラメータを表 6-1 に，GA シミュレーションに使用したプログラムを付録 1 のプログラムリスト List.16 にそれぞれ示す．

表 6-1 の制御パラメータを用いて角度制御の単独試験を行う．モデル初期位置は原点 O(0,0)とし目標点 P は P(5,5)に設定した．その時の結果として，試験時の角度差分値$(\theta - r\theta)$の時間変化を図 6-8 に示す．図 6-8 では，角度差分値の変動が少なくなった 2 秒以降の値を定常状態とした．定常状態の角度差分値に注目すると，チャタリングは 5[deg]程度までしか低減できていなかったが，本論文ではこの制御パラメータを角度制御に使用することとした．

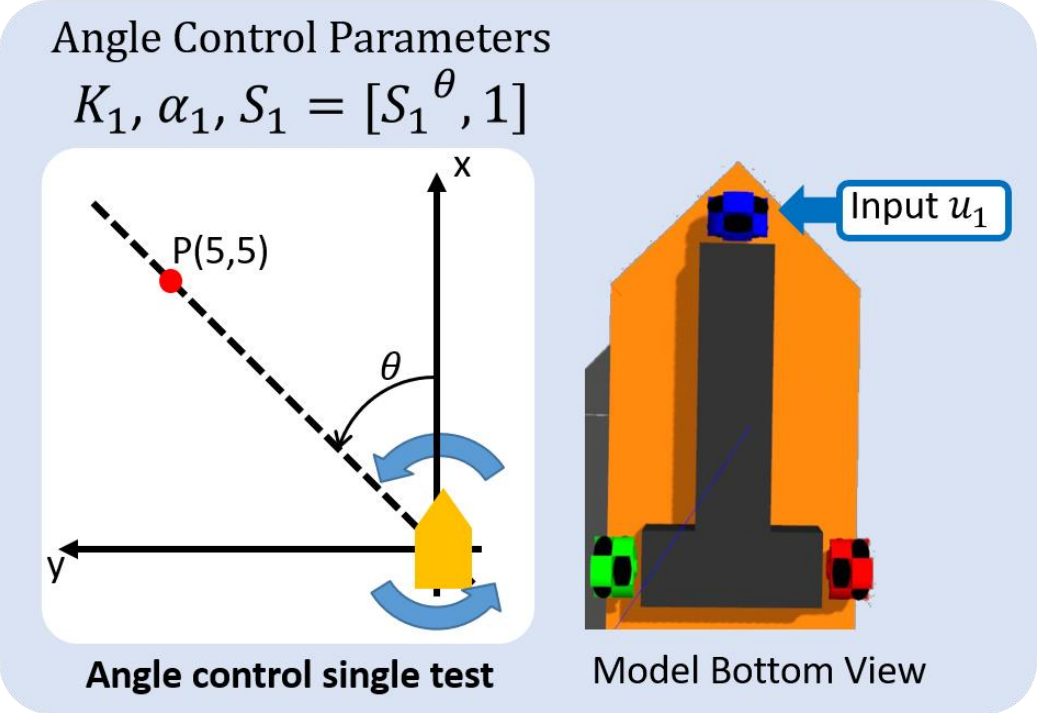

図 6-7 角度制御単独試験の概要

表 6-1 GA により作成した角度制御器の制御パラメータ

K_1	20.59
α_1	2.15×10^{-2}
$S_1{}^{\theta}$	26.24

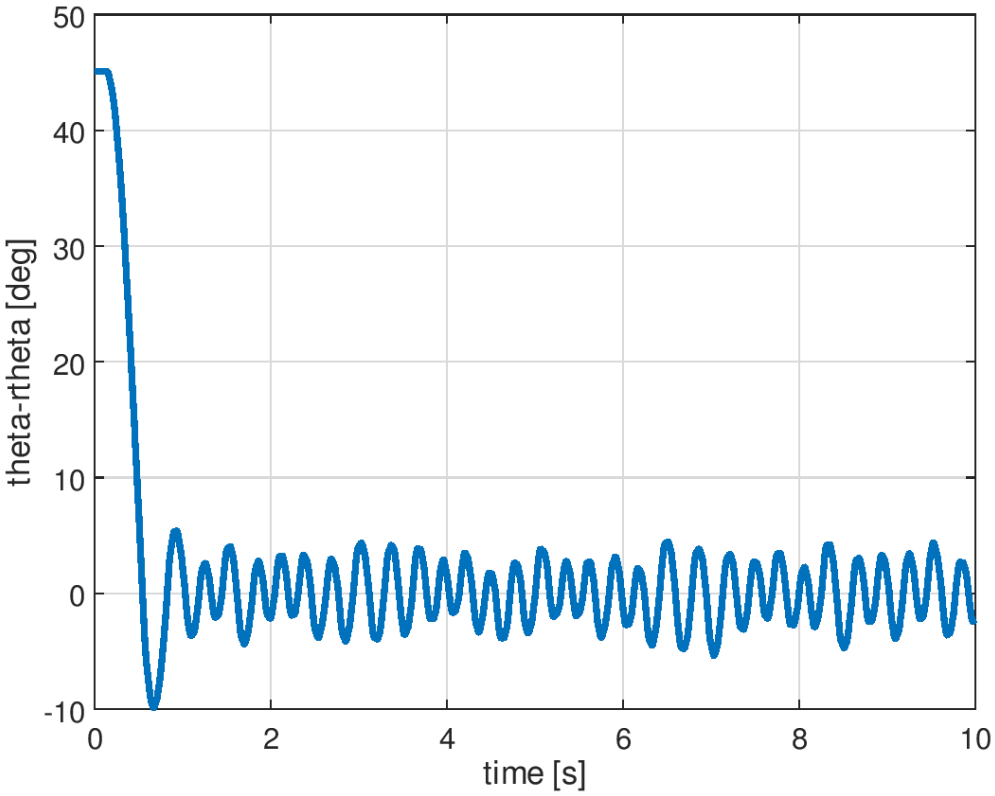

図 6-8　角度差分値：$(\boldsymbol{\theta} - \boldsymbol{\theta}_r)$の時間変化（角度制御単独試験時）

6.4.2. 距離制御器の作成

図 6-9 に，直線距離移動制御の単独試験の概要を示す．ここでは，制御入力値はモデル後方の 2 つのオムニホイールに同じ値を出力し前方への直進のみを行う．この時の運動モデルは直線移動のみの単純なものである．そこで，スライディングモード制御のロバスト性を利用し，RC 回路の状態空間モデルを 3 次にサーボ拡張した以下の(35)式で代用した．

$$\begin{cases} \dot{X}_2 = AX_2 + Bu_2 + D \cdot rd \\ \mathrm{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\dfrac{1}{\tau} & 0 \\ -1 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ \dfrac{1}{R} \\ 0 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{cases} \tag{35}$$

(35)式の行列 A の変数τについては，モデル後方のオムニホイール 2 つにステップ入力($u_s = 5[\mathrm{N}]$)を加えた時の動作特性より作図によって求めた．この時の動作特性として，モデル移動速度の時間変化を表したグラフを図 6-10 に示す．図 6-10 では無駄時間を含む応答となっているため，通常であればパデ近似などによって運動モデルを同定する必要がある．しかし，ここではスライディングモード制御のロバスト性を活用し，時定数は純粋な 1 次遅れ系のものとして算出した．したがって，無駄時間については考慮していない．図 6-10 より，出力応答の波形は振動しているため，その中間点を通る近似曲線を作図した．ここで，モデル最大移動速度 : $Y = 1.0[\mathrm{m/s}]$，無駄時間 : $L = 0.21[\mathrm{s}]$とした．これらの値より，縦軸の$0.632Y[\mathrm{m/s}]$から横軸に平行な補助線を引き，近似曲線と重なった時の時刻を : $\tau + L[\mathrm{s}]$とした．定規によって 0 から$\tau + L[\mathrm{s}]$までの距離 : $l_{\tau+L}$を測定した結果，$l_{\tau+L} = 20.5[\mathrm{mm}]$であった．また，同様に 0 から 1[s]までの距離 : l_{1s}は$l_{1s} = 45[\mathrm{mm}]$となった．よって，時刻 : 距離の対比より(36)式のように時定数 : τを求めた．この値をモデルの時定数として使用する．

$$(\tau + L) : 1 = l_{\tau+L} : l_{1s}$$

$$\tau = \frac{1 \times l_{\tau+L}}{l_{1s}} - L \fallingdotseq 0.245 \tag{36}$$

また，(35)式の行列 B の変数Rについては一意に決めた値として$R = 1$を使用する．

(35)式を運動モデルとしてスライディングモード制御を行う．制御入力 : uとしては定常状態におけるチャタリングを低減できるように，(33)式に等価線形入力 : u_{eq}を加えた(37)式を使用した．

$$\begin{cases} u_2 = u_{eq} - K_2 \cdot sat(\alpha, \sigma_2) \\ u_{eq} = -(S_2B)^{-1}S_2AX_2 - (S_2B)^{-1}D \cdot rd \\ sat(\alpha_2, \sigma_2) = \dfrac{\sigma_2}{|\sigma_2| + \alpha_2} \\ \sigma_2 = S_2 \cdot X_2 \\ X_2 = \left[x, \dot{x}, \int x\right], S_2 = [S_1^x, S_2^x, -1] \end{cases} \tag{37}$$

さらに，与える状態量 : X_2については，モデルの現在位置 : xと，モデル移動速度 : $\dot{x}$に加え，距離差分値の積算項 : $\int x$も使用する．したがって，超平面 : S_2についても同じ次数となるように$S_2 = [S_1^x, S_2^x, -1]$とする．また，(37)式中の変数 rd はモデルの初期位置から目標点 P までの直線距離である．ここで，スライディングモード制御の定義[6]より(38)式が言える[6]．

$$\dot{\sigma}_2 = S_2\dot{X}_2 = 0 \tag{38}$$

(38)式に(36)式, (37)式を代入しu_2について解くと，以下の(39)式が算出できる．

$$u_2 = -(S_2B)^{-1}S_2AX_2 - (S_2B)^{-1}D \cdot rd \tag{39}$$

この(39)式を(37)式の等価線形入力u_{eq}として使用した．

ここで，(35)式の運動モデルを(37)式によってスライディングモード制御が行えるかを，数値解析ソフトウェアの Octave を用いて検討した．この時，(35)式の状態空間モデルの行列 A については，以下の(40)式の安定余裕法を用いたA_εを使用する．また，(37)式の超平面 Sについては(40)式の値を用いて，LQ 関数により作成した．

$$\begin{cases} A_\varepsilon = A + \varepsilon I \\ Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1000 \end{bmatrix} \\ \alpha = 0.1, K = 1, rd = 5, \varepsilon = 1 \end{cases} \tag{40}$$

(40)式の行列 Qについては，物理シミュレータでの制御を考慮して，状態量 Xのサーボ項の重みを大きく設定して，シミュレーションを行った．数値シミュレーションに使用したプログラムを付録 1 のプログラムリストの List.18 に示す．目標点までの距離rdを$rd = 5$[m]として数値シミュレーションを行った．結果を図 6-11 に示す．(35)および(37)式を用いることで．目標値$rd = 5$[m]で状態量 xが収束していることが確認できる．よって，(35)式を運動モデルとして，(37)式を制御入力として距離制御に使用する．

(37)式において$K_2, \alpha_2, S_1^x, S_2^x$については未定数となるため，次式の(41)式を用いて適応度 : fが最大となるようなパラメータを GA により設計した．

$$\begin{cases} \mathrm{f} = \mathrm{c} \cdot \dfrac{1}{mp} + d \cdot \dfrac{1}{count_x} \\ c = 2.0 \times 10^5, d = 4.0 \times 10^4 \end{cases} \tag{41}$$

ここで，(41)式では mp は距離差分値（目標点とモデル現在位置の差分値）である．また，$count_x$は距離差分値が$rd \pm 0.1$[m]以外の値である時の回数で，定常状態におけるチャタリングの評価を行う項目である．ここで，GA シミュレーションに使用したプログラムを付録 1 プログラムリストの List.17 に示す．

6.4.2 節と同様に GA シミュレーションを行い，制御パラメータの作成を行った．作成した制御パラメータを表 6-2 に示す．また，このパラメータを用いて距離制御の単独試験を行った．モデル初期位置を原点 O(0,0)とし目標点 P(5,0)とした．試験結果として，モデル移動軌跡を図 6-12 に，モデル移動距離:xの時間変化を図 6-13 にそれぞれ示す．ここで，図 6-12 においてモデル前方方向は x 軸＋方向，モデル左方向は y 軸＋方向となる．移動軌跡を確認すると，モデルは目標点である P(5,0)まで直進し，P 付近で停止していることが確認できる．図 6-13 では，定常状態を距離差分値の変動が少なくなった 4 秒以降とし，この時の xの値は 0.01[m]以下と小さくチャタリングを十分に抑制できていることが確認できた．したがって，距離制御のパラメータは GA によって適したものを作成できたと言える．

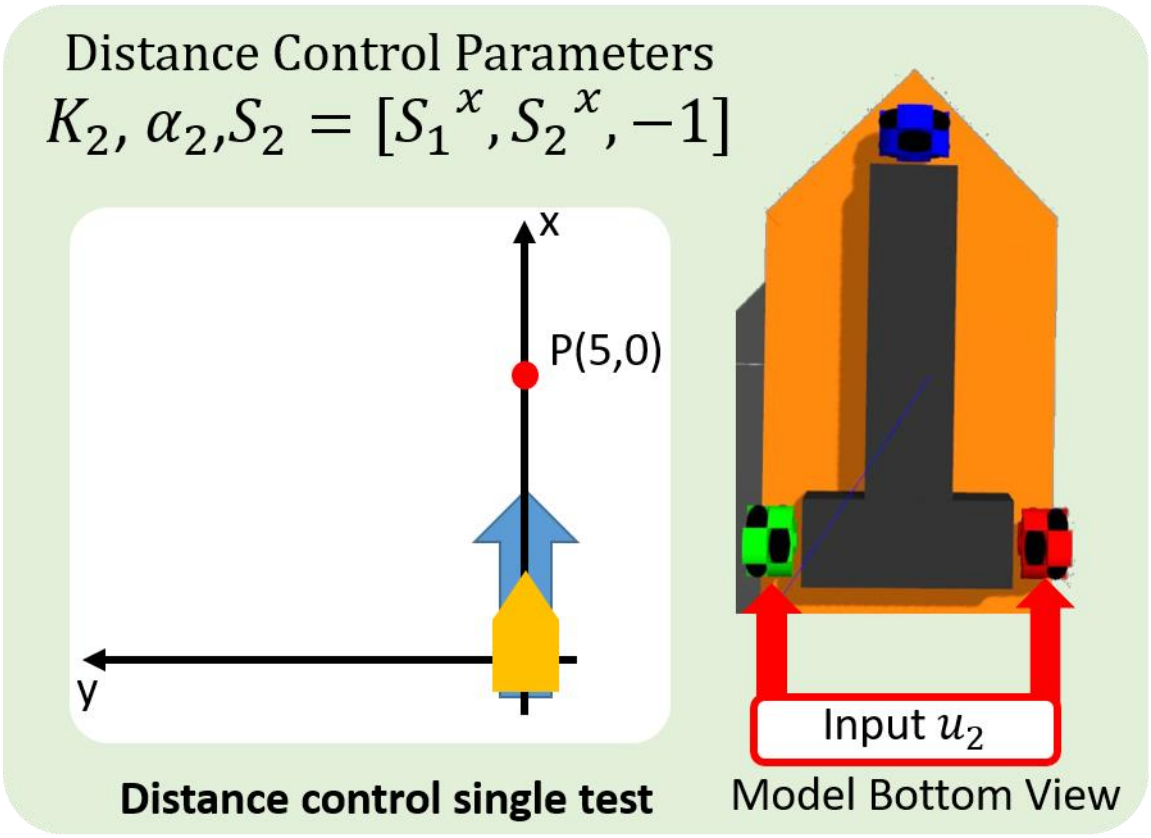

図 6-9 距離制御単独試験の概要

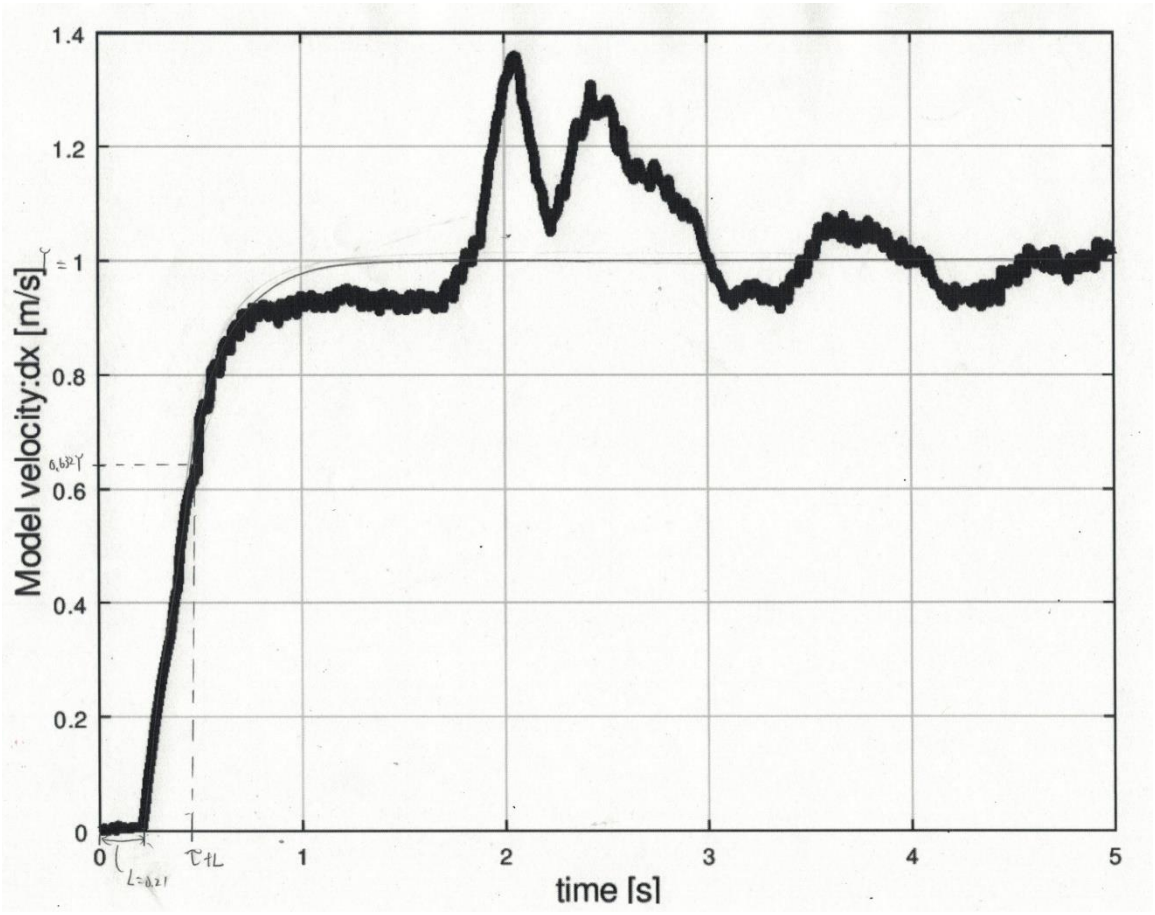

図 6-10 モデル後方の 2 つのオムニホイールにステップ入力(u_s)を加えた時のモデル動作特性

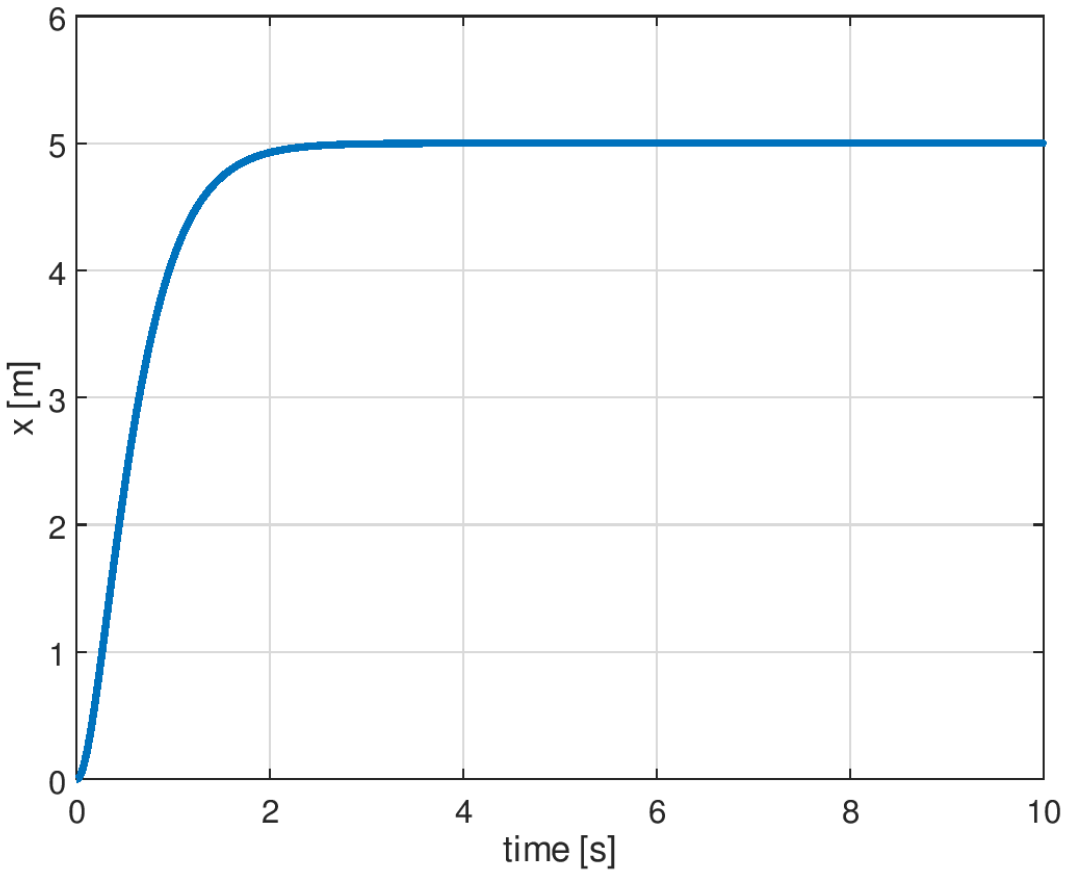

図 6-11 モデル移動距離 x の時間変化(Octave による数値シミュレーション結果)

表 6-2 GA によって作成した距離制御器の制御パラメータ

K_2	1.13×10^{-3}
α_2	5.49×10^{-1}
${S_1}^x$	6.81×10^{-2}
${S_2}^x$	5.74×10^{-4}

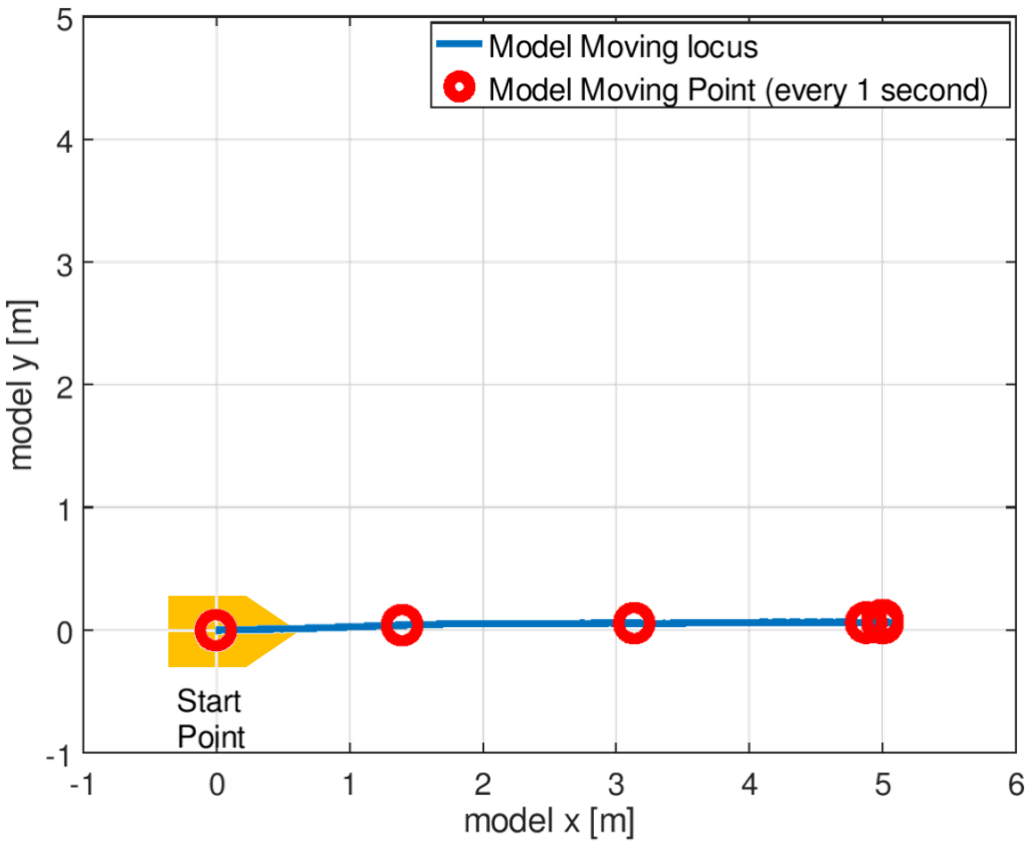

図 6-12 モデル移動軌跡（距離制御単独試験）

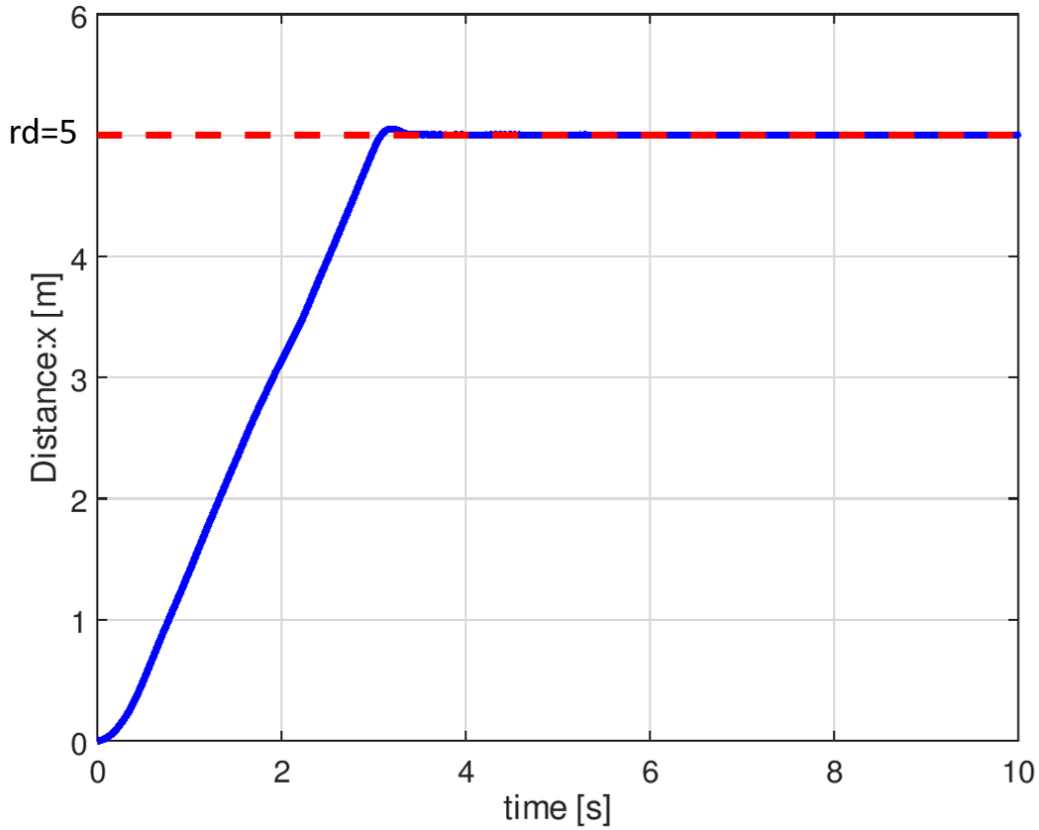

図 6-13 モデル移動距離：xの時間変化（距離制御単独試験）

6.5. 簡易 DPS 制御試験

6.4 節で作成した各制御器を用いて簡易 DPS 制御試験を行う．制御試験には，以下の表 6-3 の 2 つのパラメータを用いた．GA1 は 6.4 節で作成した制御パラメータである．GA2 は比較用のパラメータであり，6.4 節のように角度制御と距離制御で制御器を分けず一度に作成した．GA シミュレーションでは世代数を 10，個体数を 100 とし，モデル初期位置は原点 O(0,0)，目標点 P は P(5,5)に設定した．遺伝子の評価式は，6.4.1 節の(34)式と 6.4.2 節の(41)式を組み合わせた以下の(42)式を用いた．

$$\begin{cases} f = a\dfrac{1}{(\sum diff_\theta)^2} + b\cdot\dfrac{1}{count_\theta} + c\cdot\dfrac{1}{mp} + d\cdot\dfrac{1}{count_x} \\ a = 2.0\times10^7, b = 8.0\times10^3, c = 2.0\times10^5, d = 5.0\times10^3 \end{cases} \tag{42}$$

表 6-3 の各パラメータを用いて簡易 DPS 制御試験を行う．ここで，表 6-3 の GA2 のパラメータ作成時には目標点は P(5,5)として設計したが，今回の試験では他の点でも移動できることを確かめるため，モデル初期位置は原点 O(0,0)，目標点は P(5,-5)に設定した．各制御結果として，GA1 のパラメータ使用時のモデル移動軌跡を図 6-14 に，GA2 のパラメータ使用時のモデル移動軌跡を図 6-15 に，角度差分値：$(\theta-\theta_r)$の時間変化を図 6-16 に，モデル距離：xの時間変化を図 6-17 にそれぞれ示す．また，簡易 DPS 制御プログラムおよび GA2 の導出に使用したプログラムを付録 1 プログラムリストの List.18 にそれぞれ示す．

図 6-14 および図 6-15 より，GA1 と GA2 の制御結果では共に目標地点まで到達していることを確認した．しかし，ヘディング制御を行うエリアに入った後のモデルの挙動は，目標地点より大きく動くものとなっていた．一方で，図 6-16 では，値の変動が少なくなった 5.0[s]以降を定常状態とした．両制御結果共にヘディング制御時における定常状態のチャタリングに注目すると，モデル角度の振れが約 18[deg]から 20[deg]と大きいことが確認できる．そのため，ヘディング制御時においてモデルが大きく動き回る結果となったのである．この原因としては，6.4.1 節の角度制御結果において，制御設計を簡単にするため運動モデルを考慮しなかったことが原因であると考えられる．よって，安定したヘディング制御および定点保持を行うためには，距離制御と同様に運動モデルを考慮した制御を行う必要があると言える．また，図 6-17 より，距離制御については GA1 では約 0.2[m]，GA2 では約 1.8[m]のチャタリングが発生していた．GA2 については，GA によって角度制御と距離制御の制御パラメータを一度に作成したため，遺伝子の評価項目が多くなり進化方向に偏りが生まれたと考えられる．そのため，距離制御のパラメータが十分に進化できなかったのだと考えられる．

これまでの結果より，モデルの挙動としては GA1 と GA2 でほぼ同様なものであると言える．よって，スライディングモード制御によって個別に作成した制御器を併用しても制御結果に問題は無いと言える．また，GA による制御パラメータ設計についても，制御器を分けて作成することで評価式の複雑化を防ぎ設計を容易にできることが確認できた．

表 6-3 簡易 DPS 制御に使用した各制御パラメータ

		GA1	GA2
角度制御	K_1	20.59	11.84
	α_1	2.15×10^{-2}	3.14×10^{-1}
	${S_1}^{\theta}$	26.24	24. 94
距離制御	K_2	1.13×10^{-3}	1.26×10^{-3}
	α_2	5.49×10^{-1}	1.62×10^{-5}
	${S_1}^{x}$	6.81×10^{-2}	4.25×10^{-3}
	${S_2}^{x}$	5.74×10^{-4}	5.98×10^{-2}

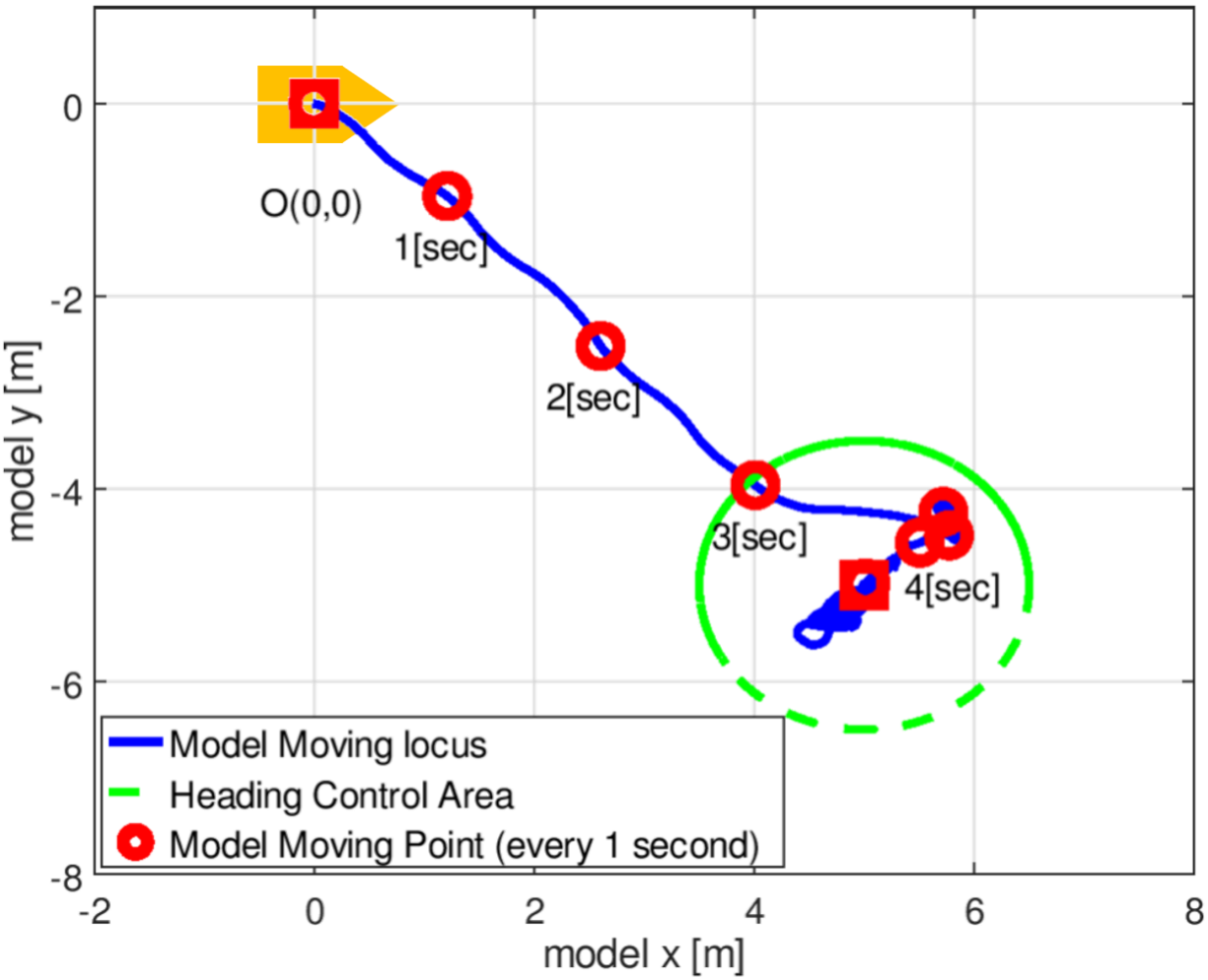

図 6-14 GA1 のパラメータ使用時のモデル移動軌跡

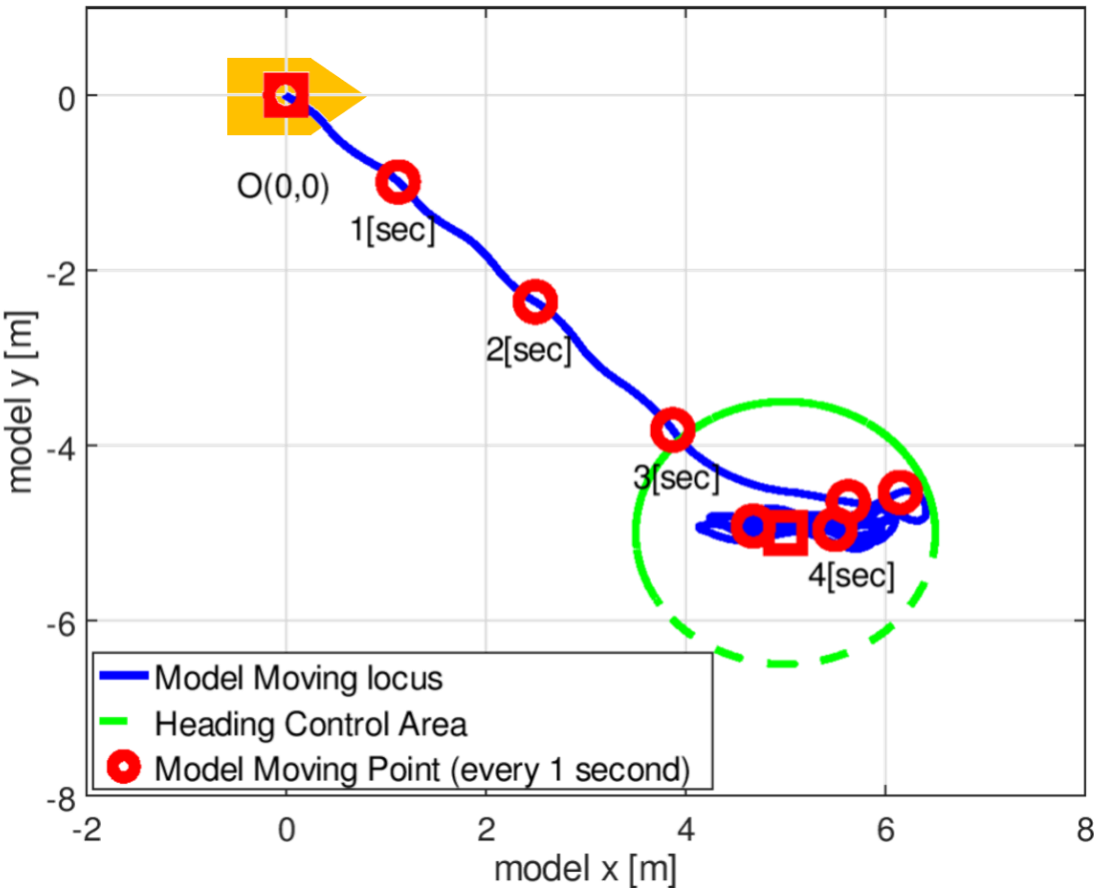

図 6-15 GA2 のパラメータ使用時のモデル移動軌跡

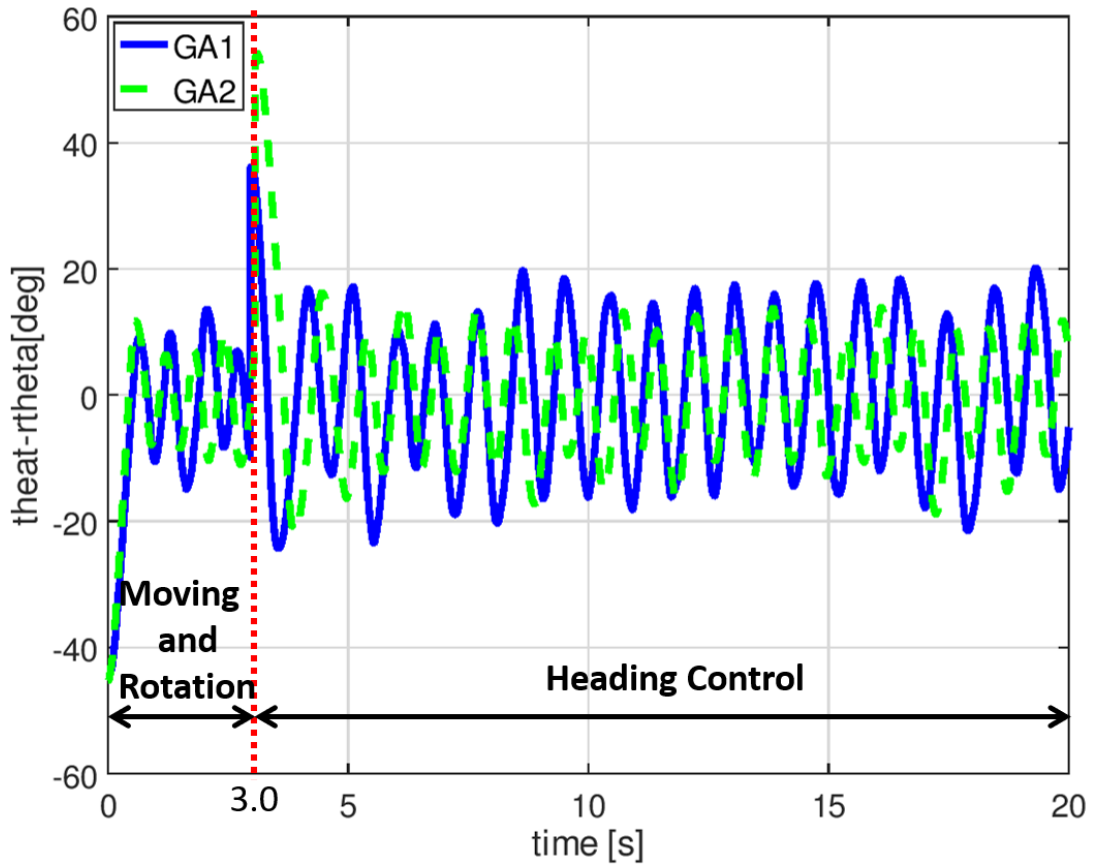

図 6-16 角度差分値：$(\theta - r\theta)$の時間変化（簡易 DPS 制御時）

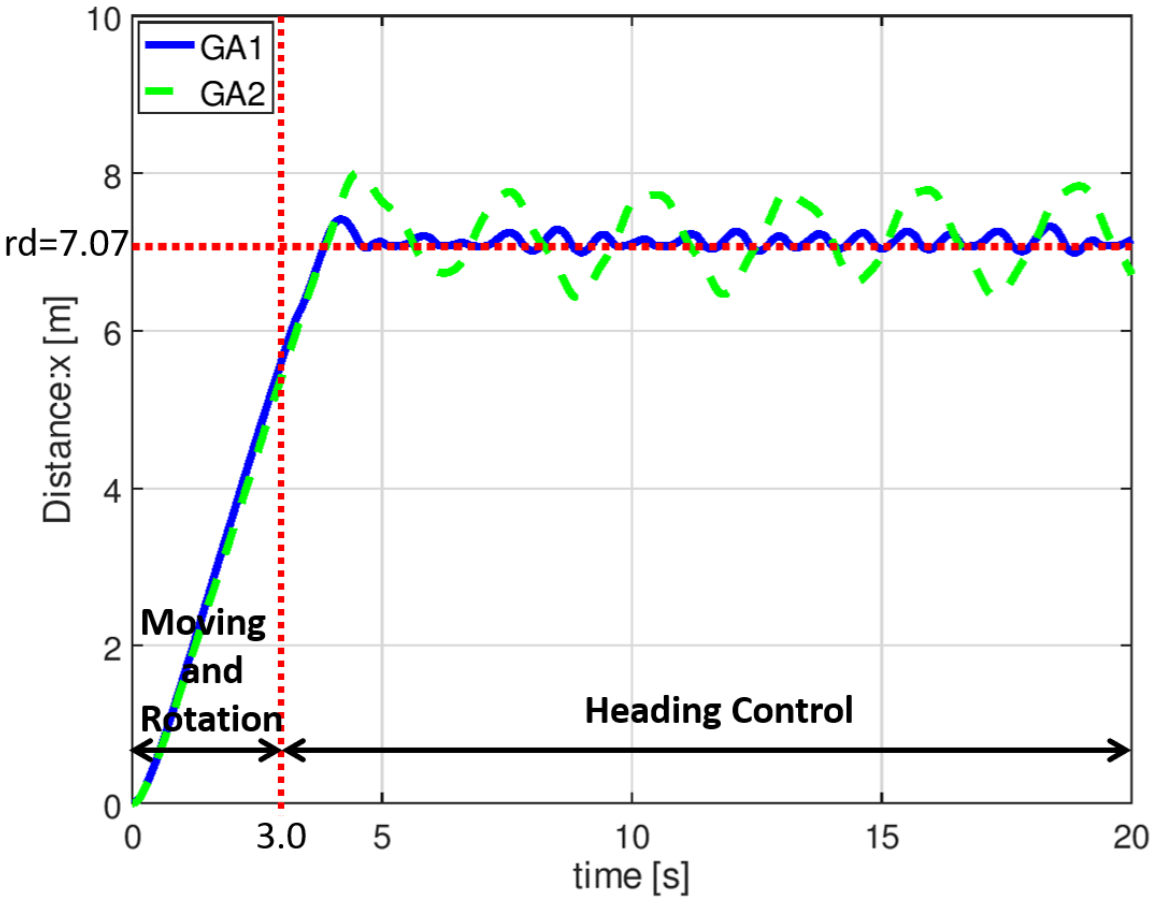

図 6-17 モデル移動距離:xの時間変化（簡易 DPS 制御時）

7. 結言

本研究では，干潟環境に対応した調査機器としてμ-ASV を搭載した多足歩行型ロボットを提案し，μ-ASV 部と多足歩行型ロボット部にモデルを分けて考え，それぞれの運動制御システムについて検討した．また，この検討にはロボット開発支援ソフトの ROS と物理シミュレータである Gazebo を使用した．

初めに，多足歩行型ロボットのモデル作成から取り組んだ．モデルとしては，不整地での安定性と修繕のしやすさを考慮し，6 脚 6 節を有する歩行ロボットモデルを採用した．まずは，遺伝的アルゴリズム(GA)によって歩行パターンを作成し，本モデルが陸上(平地)で安定な歩行を行えることを確認した．しかし，この時には遺伝子がトライポッド歩行を行うもののみに進化するように，各脚に与える制御パラメータを共通のものとしていた.そこで,本論文では干潟での自由な動作を目指し，1）ロボットの各関節の動作自由度を向上させる，２）左右への旋回歩行を行うパターンの作成法，の 2 点について検討を行った．１）については，各脚に与える位相差を個別に設定し，GA で使用する遺伝子の評価式を適切に設定することで環境に合わせた歩行パターンが作成できることを確認した．２）については，モデルの片側の脚に任意の位相差を与えることによって大きく旋回する歩行パターンが作成できることを確認した．また，その場回頭を行う歩行パターンをベースパターンとし，モデルの左右脚で速度比を付けることによって小さく旋回する歩行パターンを作成することができた．このため，周囲の環境によって歩行パターンを選択するような制御システムを作成できれば，陸上を自由に移動することが可能であると言える．

しかし，GA による手法では周囲の環境が変化した場合には再度歩行パターンを設計し直す必要があるなどの問題点がある．したがって，自然界の多足歩行生物の歩行に注目し，非線形同期を用いた歩行パターンの作成方法についても検討を行った．ここでは，多足歩行ロボットの各脚の運動を表す数理モデルとして 2 結合 VDP 方程式を用いた．そして，オープンループ制御系とフィードバック制御系を作成し，それぞれを用いて 2 脚モデルの制御を行った．オープンループ制御系では，フロケの理論より安定な周期解が作成できることを確認できた．また，周期外力を加えた 2 結合 VDP 方程式を用いることで，任意のタイミングで同期パターンを変更できることについても確認した．しかし，オープンループ制御系では一意な歩行パターンを作成しているに過ぎず，現在の状態は考慮されていない．よって，GA による手法と同様に周囲の環境が変化した場合には再設計が必要となるため，実際の運動制御としては現実的ではない．そこで，周囲の環境の変化にも対応するため，フィードバック制御系について検討を行った．その結果，任意のタイミングで周期解の同期パターンを変更できることを確認した．これにより，2 結合 VDP 方程式を用いれば環境に合わせて同期パターンを切り替えることが出来ると言えるため，干潟の移動も自由にできると考えられる．しかし，周期解の安定性については，VDP 方程式の近似解を用いていないためフロケの理論による判別が行えなかった．フィードバック制御系で周期解の安定性を判別する方法については現在検討中である．

次に，μ-ASV についての検討を行った．モデルとしては，現在の使用している物理シミュレータ(Gazebo)上で干潟環境の再現が難しいため，陸上で動作する台車モデルを作成した．そのため，全方位へ移動することができるオムニホイールをスラスタの代用とすることで船体運動を再現した．このモデルを用いて，任意の目標座標点に移動し，定点保持を行うような簡易的な DPS 制御システムの開発を行った．制御手法としては，制御器設計を容易にするため，スライディングモード制御を用い

た．また，制御器の設計に必要となるパラメータについては，GA によって作成した．作成した角度制御と距離制御の各制御器を用いることで，μ-ASV モデルで簡易 DPS 制御を行う制御システムを作成することができた．しかし，ヘディング制御時における角度制御のチャタリングが約 20[deg]と大きくなっている．この原因としては，6.4.1 節の角度制御結果において，制御設計を簡単にするため運動モデルを考慮しなかったことが原因であると考えられる．よって，安定したヘディング制御および定点保持を行うためには，距離制御と同様に運動モデルを考慮した制御を行う必要があると言える．しかし，制御結果としてはモデルの挙動は GA1 と GA2 でほぼ同様なものであると言える．よって，スライディングモード制御によって個別に作成した制御器を併用しても制御結果に問題は無いと言える．また，GA による制御パラメータ設計についても，制御器を分けて作成することで評価式の複雑化を防ぎ設計を容易にできることが確認できた．

上記のことから，多足歩行型ロボットおよびμ-ASV の基盤となる制御シミュレーションシステムは作成できたと言える．また，多足歩行型ロボットについては実機の開発を現在進めており，1 脚分のモータの制御については行えることを確認した．これを 6 脚に拡張し，3 章で作成した歩行パターンを用いて陸上での歩行試験を行うことが今後の課題だと言える．μ-ASV については，運動解析を容易にするためオムニホイールによって移動するモデルを作成した．そのため，今後はスラスタを搭載した実機の作成を行い，その動作特性について調査する．そして，その調査結果を用いて 5 章で作成した簡易 DPS 制御システムにフィードバックし，スラスタを搭載したモデル用にシステムを拡張することが課題と言える．これらの課題をクリアし，多足歩行型ロボットとμ-ASV の各運動制御システムを組み合わせて用いることで，本研究で提案する干潟調査用ロボットの運動制御も可能であると考えている．

参考文献

[1] 花輪伸一, “日本の干潟の現状と未来”, ‘地球環境 vol.11 No.2’, pp.235-244, 2006.

[2] 表允哲, 倉爪亮, 渡邊裕太, “諸説 ROS ロボットプログラミング”, ‘(http://irvs.github.io/rosbook_jp/)’, pp.41-49, pp.241-242, 2015.

[3] 銭飛, “ROS プログラミング”, ‘森北出版’, pp.1-16, 2016.

[4] 阿居院猛, 長尾智晴, “ジェネティック アルゴリズム”, ‘昭晃堂’, pp.1-16, 1996.

[5] 中村圭, 田原淳一郎, 齋藤幹大, “多足歩行ロボットの開発-ROS を使ったシミュレーション-”, ‘海洋理工学会 平成 29 年度春季大会 講演論文集’, pp.61-64, 2017.

[6] 野波健蔵, 田宏奇, “スライディングモード制御-非線形ロバスト制御の設計論-”, ‘コロナ社’, pp.24-71, 1994.

[7] 齋藤勝也, 宮崎倫子, “結合 van der Pol 方程式の安定性解析”, ‘数理解析研究所講究録 第 1637 巻’, pp.32-38, 2009.

[8] BenceMagyar, “ROS.org wiki ros_control”, ‘Open Source Robotics Foundation (http://wiki.ros.org/ros_control)’, 2016.

[9] Chris Prahacs, Aaron Saunders, Matthew K.Smith, Dave McMordie and Martin Buehler, “Towards Legged Amphibious Mobile Robotics”, ‘Proceedings of the Canadian Design Engineering Network Conference’, (http://ojs.library.queensu.ca/index.php/PCEEA/article/view/4043), 2004.

[10] 梶原秀一, 花島直彦, 青柳学, “引き込み現象を利用した結合 van der Pol 方程式の同期パターン制御”, ‘電子情報通信学会論文誌 A Vol.J98-A No.6’, pp.406-416, 2015.

[11] 梶原秀一, 花島直彦, 青柳学, “結合周期入力制御系の相互引き込み現象と同期パターン制御 -2 または 3 結合周期入力制御系の場合-”, ‘計測自動制御連合 Vol.53 No.5’, pp.308-318, 2017.

謝辞

本論文は，筆者が東京海洋大学大学院海洋科学技術研究科海洋システム工学専攻博士前期課程に在籍中の研究成果をまとめたものです．東京海洋大学学術研究院海洋電子機械工学部門の田原淳一郎准教授には指導教官として本研究の実施の機会を与えて頂き，その遂行にあたって終始丁寧なご指導を頂きました．ここに深謝の意を表します．また，同部門の章ふぇいふぇい教授，並びに小池雅和助教授には副査としてご助言を頂くとともに，時には本論文の細部にわたる部分の議論に参加して頂くこともあり，深く感謝しております．また，筆者が在籍していた電子制御研究室において，研究の内容について議論を交わし多くのアイデアを提供してくれた同期生の伊藤大智君に感謝の意を表します．ならびに，実験機材作成時の補助などを行ってくれた同研究室スタッフの齋藤幹大君，加藤哲君，川村大和君にも感謝します．最後に学生生活を送る上で学業に専念できるように支援してくれた両親に心からの感謝を申し上げます．

付録目次

付録１　プログラムリスト

各プログラムリストのソースリンクを示す.

List.01 多足歩行型ロボットモデルの構成プログラム

以下の No.1 から 5 で構成される.

No	ファイル名	備考
1	hexbug2_world.launch	多足歩行型ロボットモデルのスポーンおよびシミュレータの起動プログラム
2	hexbug2.world	シミュレーション環境設定ファイル
3	hexbug2.xacro	モデルの記述ファイル
4	rrbot.gazebo	モデルコンフィグファイル(ros_control_plugin の挿入, 各 Link への摩擦係数の設定など)
5	materials.xacro	モデルに使用する外見色の参照元ファイル
プログラムソース	https://www.dropbox.com/sh/477tjlkuxc70605/AABsFrVQ5pwXMBw7-cA3-uj3a?dl=0	

List.02 多足歩行型ロボットの各関節の制御コントローラ

以下の No.1 から 2 で構成される.

No	ファイル名	備考
1	rrbot_control.launch	制御コントローラの起動ファイル
2	rrbot_control.yaml	制御コントローラのコンフィグファイル
プログラムソース	https://www.dropbox.com/sh/la5kxdgfnljrvx8/AAALbVcV7OQ21e4aFbnJ_UNoa?dl=0	

List.03 GA シミュレーションプログラム(トライポッド歩行のみ)

以下の No.1 から 15 で構成される（一部のプログラムは List.01 および List.02 を参照).

No	ファイル名	備考
1	ga_simulation_ver7.m	GA シミュレーションメインプログラム
2	mk_animal.m	任意の行数と列数をもつ bit 配列(シミュレーションの初期遺伝子)の作成プログラム
3	output_csv_ver2.m	作成した遺伝子(2 進数)を制御パラメータ(10 進数)に変換するプログラム
4	tripod_3.bsh	モデルのスポーン，制御コントローラの起動，歩行プログラムの実行プログラム
5	ga_ps_del.bsh	Gazebo に関係するプロセスの強制終了を行うプログラム
6	ps_del_5.bsh	指定文字列のプロセスを強制終了を行うプログラム
7	hex_tripod_ver5.py	角速度制御による歩行プログラム
8	change_velocity.py	接地脚時と遊脚時の角速度を変更する関数プログラム
9	delta_t.py	設定された位相差(待ち時間)まで脚を停止させる関数プログラム
10	fitness_calculate_ver4.m	遺伝子の評価プログラム
11	z_skip.m	csv ファイルの空行を 0 で埋めるプログラム
12	s_animal.m	遺伝子の淘汰を行うプログラム
13	mk_pair2.m	一点交叉を行うランダムなペアを作成するプログラム
14	cross_animal.m	遺伝子の一点交叉を行うプログラム
15	mu_animal.m	遺伝子の突然変異を行うプログラム
プログラムソース	https://www.dropbox.com/sh/v30n5l5ditfdtu2/AACYNV83Llv7iFT_J5Jn7zxta?dl=0	

List.04 GA シミュレーションプログラム(各関節の自由度を増加,動歩行)

以下の No.1 から 5 で構成される（一部のプログラムは List.01 から List.03 を参照).

No	ファイル名	備考
1	ga_mud_simulation_ver1.m	GA シミュレーションのメインプログラム
2	output_csv_ver3.m	遺伝子を制御パラメータに変換するプログラム
3	mud_walk.bsh	モデルのスポーン，制御コントローラの起動，モデルの歩行を一度に行うプログラム
4	mud_walk_ver1.py	角速度制御による歩行プログラム
5	fitness_calculate_ver5.m	遺伝子の評価を行うプログラム
プログラムソース	https://www.dropbox.com/sh/vxf1ew2vdjydlaa/AADNpYmIvj74FpidZPH4rFZTa?dl=0	

List.05 GA シミュレーションプログラム(各関節の自由度を増加,5 脚歩行)

以下の No.1 から 4 で構成される（一部のプログラムは List.01 から List.04 を参照).

No	ファイル名	備考
1	ga_mud_simulation_program_ver2.m	GA シミュレーション実行プログラム
2	ga_mud_simulation_ver4_continue.m	GA シミュレーションメイン関数プログラム
3	mud_walk_ver3.py	角速度制御による歩行プログラム
4	fitness_calculate_ver9.m	遺伝子の評価プログラム
プログラムソース	https://www.dropbox.com/sh/k86zcnxfvne6ld0/AABtm4s2b8ps2KNsYlK17yEKa?dl=0	

List.06 位相差による旋回歩行

以下の No.1 から 9 で構成される（一部のプログラムは List.01 から List.05 を参照).

No	ファイル名	備考
1	ga_tripod_simulation_ver1.m	GA シミュレーション実行プログラム
2	hexbug2_gatripodsim_r1.m	GA シミュレーションメイン関数プログラム
3	output_csv_3params.m	遺伝子(2 進数)を制御パラメータ(10 進数)に変換するプログラム
4	hexbug2_make_walkpattern_r4_2.m	制御パラメータより歩行パターン（角速度の動作パターン）を作成するプログラム
5	change_velocity.m	角速度の切り換えプログラム
6	delta_t.m	設定された位相差(待ち時間)まで脚を止めるプログラム
7	hexbug2_gatripod.bsh	モデルのスポーン，制御コントローラの起動，歩行までを一度に行うプログラム
8	hexbug2_csvread_gatripod_r1.py	歩行パターンに従って歩行するプログラム
9	hexbug2_make_walkpattern_r4.m	任意の位相差をモデル右側の脚に加えた歩行パターンを作成するプログラム
プログラムソース	https://www.dropbox.com/sh/5g69ob9buwsmadd/AADHbxh7_3IZg5A8IOyaz9Qoa?dl=0	

List.07 速度比による旋回歩行

以下の No.1 から No.4 で構成される（一部のプログラムは List.01 から List.06 を参照).

No	ファイル名	備考
1	make_mixpattern_r3.m	任意の速度比を加えた歩行パターンを作成するプログラム
2	change_velocity_r3.m	角速度を切り換えるプログラム
3	deltat_r3.m	設定された位相差(待ち時間)まで脚を停止させるプログラム
4	hexbug2_csvread_mixtripod_r2.py	作成した歩行パターンで歩行するプログラム
プログラムソース	https://www.dropbox.com/sh/ebdgc6lp4ad2fap/AAC-aifQVNrASZ_seh1Ud1f5a?dl=0	

List.08 未定数φ_1，φ_2の算出プログラム

以下の No.1 から No.4 で構成される（各プログラム名で GA と入っているが，本リストのプログラムでは GA ではなく For ループでパラメータを求めている）.

No	ファイル名	備考
1	vdp_ga_sim_r2.m	シミュレーションの実行プログラム
2	vdp_ga_r4_3.m	シミュレーションのメインプログラム
3	switch_matrix_r2.m	切換行列の作成
4	Runge_Kutta.m	ルンゲ=クッタ 4 次法での計算を行うプログラム
プログラムソース	https://www.dropbox.com/sh/9xrjfghcuek50fb/AAA7xCy3imY86IhjyITUKpDFa?dl=0	

List.09 安定な周期解を作成するオープンループ制御プログラム

以下の No.1 から No.9 で構成される（一部のプログラムは List.03 を参照）.

No	ファイル名	備考
1	test_switch_r4.m	GA シミュレーションメインプログラム
2	numerical_solution_switch_r2.m	数値解を計算するプログラム
3	zero_cross_check_1cyc_r4.m	1 周期を計算するプログラム
4	steady_state_time_r3.m	平均周期を求めるプログラム
5	phi_change_dec_r2.m	遺伝子を制御パラメータに変換するプログラム
6	approximate_solution_r3.m	近似解を求めるプログラム
7	approximate_solution_check_r2.m	GA で作成した最も高い適応度のパラメータを用いて近似解の再計算を行うプログラム
8	floquet_r2.m	フロケ乗数を算出するプログラム
9	twolegs_csvread_walk_r1.py	作成したパターンをモデルに出力するプログラム
プログラムソース	https://www.dropbox.com/sh/k9ugwjvebu50er5/AAAh4QJ_Yu4OXTfSqZGKuljza?dl=0	

List.10 2脚モデルの構成プログラム

以下の No.1 から 3 で構成される（一部のプログラムは List.01 を参照).

No	ファイル名	備考
1	rrbot_world.launch	モデルのスポーンおよびシミュレータの起動を行うファイル
2	rrbot.xacro	モデルの記述ファイル
3	rrbot.gazebo	モデルに ros_control_plugin の挿入および各 Link に 摩擦係数を設定
プログラムソース	https://www.dropbox.com/sh/xp95zmt27qcmqay/AAAjMv1-VjKgNWOWZUJwggxXa?dl=0	

List.11 2脚モデルの各関節の制御コントローラ

以下の No.1 から No.2 で構成される.

No	ファイル名	備考
1	rrbot_control.launch	制御コントローラの起動を行うプログラム
2	rrbot_control.yaml	制御コントローラのコンフィグファイル
プログラムソース	https://www.dropbox.com/sh/e5rg37vwivkt5jy/AABSaBJDG6m8kdQmYFoLwLE4a?dl=0	

List.12 フィードバック制御プログラム

以下の No.1 から No.3 で構成される.

No	ファイル名	備考
1	huriko2_syncro_control_r5.py	フィードバック制御系での 2 脚モデル制御プログラム
2	runge_kutta.py	ルンゲ=クッタ法 4 次の計算プログラム
3	zero_cross_check3.py	1 周期を計算するプログラム
プログラムソース	https://www.dropbox.com/sh/niu1yhz6w3wevpq/AAAOgCo1sixfDOuWOPJIEsiba?dl=0	

List.13 1脚制御プログラム

以下の No.1 のプログラムを使用する.

No	ファイル名	備考
1	motor_control2_r2_9.ino	1脚試験装置の制御回路に書き込むプログラム (モータの制御およびモータの回転角度，電源電圧などの値を取得し，PC に送信する)
プログラムソース	https://www.dropbox.com/sh/g0s61u88pst9uqb/AAAd6Or_PvxOYKtXizN6oiPTa?dl=0	

List.14 μ-ASV モデルの構成プログラム

以下の No.1 から No.5 で構成される.

No	ファイル名	備考
1	asv_world.launch	モデルのスポーンおよびシミュレータの起動を行うプログラム
2	asvomni_cs.world	シミュレーション環境の設定ファイル
3	asvomni_original.xacro	μ-ASV モデルの記述ファイル
4	rim_original.xacro	オムニホイールの車輪部の記述ファイル
5	roller_original.xacro	オムニホイールの樽型ローラ部の記述ファイル
プログラムソース	https://www.dropbox.com/sh/b242xha78ylrvr0/AAD3yOZTjnhHCR44_kpFhXhaa?dl=0	

List.15 μ-ASV モデルの制御コントローラ

以下の No.1 から No.2 で構成される.

No	ファイル名	備考
1	asvomni_control.launch	μ-ASV モデルの制御コントローラ
2	asvomni_control.yaml	制御コントローラのコンフィグファイル
プログラムソース	https://www.dropbox.com/sh/4snye04t315idfu/AAAo7jcAO_vatMii3gCZuhIIa?dl=0	

List.16 角度制御器設計の GA プログラム

以下の No.1 から 11 で構成される（一部のプログラムは List.03 を参照）.

No	ファイル名	備考
1	gasmcsim_main.m	GA シミュレーション実行プログラム
2	ga_smc_simulation.m	GA シミュレーションメインプログラム
3	output_csv_smc.m	遺伝子を制御パラメータに変換するプログラム
4	asvomni_smc.bsh	モデルのスポーン，制御コントローラの起動，制御プログラムの実行までを一度に行うプログラム
5	asvomni_slidingcontrol_new_1.py	角度制御単独試験プログラム
6	csv_read.py	csv ファイルを読み込むプログラム
7	csv_write.py	csv ファイルに書き込むプログラム
8	get_modelstate.py	モデルの状態を取得するプログラム
9	calcu_smw_sat2.py	スライディングモード制御の制御出力の計算プログラム
10	calcu_degree.py	目標制御角度の計算プログラム
11	fitness_calculate_smc2.m	適応度の計算プログラム
プログラムソース	https://www.dropbox.com/sh/rcn3bztl6jrzdvg/AAAIKYXfnZEN-AO7QFLu9AO2a?dl=0	

List.17 距離制御器設計の GA プログラム

以下の No.1 から 5 で構成される（一部のプログラムは List.03 および List.16 を参照）

No	ファイル名	備考
1	asvomni_slidingcontrol_new_2_3.py	距離制御単独試験プログラム
2	calcu_smw_ueq.py	スライディングモード制御の制御出力値の計算プログラム
3	decide_VecUeq2.py	目標地点までの角度を算出するプログラム
4	init_param2.py	モデルの初期位置を算出するプログラム
5	make_ueq.py	等価線形入力を算出するプログラム
プログラムソース	https://www.dropbox.com/sh/kbbhfj66il5aq2f/AAA6-zASCSnmgskRAuPdlK2aa?dl=0	

List.18 簡易 DPS 制御の GA プログラム

以下の No.1 から 4 で構成される（一部のプログラムは List.03 および List.16,17 を参照）

No	ファイル名	備考
1	asvomni_slidingcontrol_new_2_4.py	簡易 DPS 制御プログラム
2	calcu_visib2.py	目標点が視界内にあるかを判別するプログラム
3	decide_VecUeq2.py	現在角度から目標点までの角度差分値を算出するプログラム
4	heading_control2.py	ヘディング制御への切り替えを行うプログラム
プログラムソース	https://www.dropbox.com/sh/efm2Opj38guwf6p/AAAQUp5uCPwlMnOD8mPzN-pca?dl=0	

付録２　多足歩行型ロボットの設計図および部品表

多足歩行型ロボットに使用した部品表および，自作部品の cad 図面のソースを以下のリンクに示す.

		ソースリンク
使用部品表		https://www.dropbox.com/s/2cb7asqagzfilf9/%E5%A4%9A%E8%B6%B3%E6%AD%A9%E8%A1%8C%E5%9E%8B%E3%83%AD%E3%83%9C%E3%83%83%E3%83%88_%E9%83%A8%E5%93%81%E8%A1%A8.pdf?dl=0
自作部品 CAD 図面	脚部フレーム	https://www.dropbox.com/s/8iai5boepc845gc/leg_frame_r3.pdf?dl=0
	異径カップリング	https://www.dropbox.com/s/x3w6iwh7c9546hf/capring_rib_zaguri_tap.pdf?dl=0

付録３　モータ制御回路図

1 脚制御試験装置に使用した回路の回路図，実体配線図，部品表のソースを以下に示す.

・モータ制御回路

	ソースリンク
回路図	https://www.dropbox.com/s/d3bb4e0gf9u1cup/MotorControl_r3.pdf?dl=0
実体配線図	https://www.dropbox.com/s/0nge0wparfk703b/MotorControl_r3_pcb.pdf?dl=0
部品表	https://www.dropbox.com/s/ao0zd8oks90y3kt/MotorControl_%E9%83%A8%E5%93%81%E8%A1%A8.pdf?dl=0

・モータドライバ回路

	ソースリンク
回路図	https://www.dropbox.com/s/k1be0nb7kxwoj2v/MotorDriver_r2.pdf?dl=0
実体配線図	https://www.dropbox.com/s/jt1fojqaom0ulhn/MotorDriver_r2_pcb.pdf?dl=0
部品表	https://www.dropbox.com/s/6zu012e6bh3v69q/MotorDriver_%E9%83%A8%E5%93%81%E8%A1%A8.pdf?dl=0

・光センサ回路

	ソースリンク
回路図	https://www.dropbox.com/s/k4l7114pn7uysti/PhotoSensor.pdf?dl=0
実体配線図	https://www.dropbox.com/s/u7i1a80yyfvqrcf/PhotoSensor_pcb.pdf?dl=0
部品表	https://www.dropbox.com/s/8dqa2a1i68ux5lj/PhotoSensor_%E9%83%A8%E5%93%81%E8%A1%A8.pdf?dl=0

付録４　学会参加履歴

筆者が研究室配属となった 2016 年 4 月から 2019 年 2 月現在までに参加した学会の履歴を以下に示す.

１）第 49 回 計測自動制御学会
発表題目　：遺伝的アルゴリズムによる多足歩行移動システムの開発
著　者　　：中村圭，田原淳一郎，伊藤大智，齋藤幹大
掲載論文集：第 49 回 計測自動制御学会北海道支部学術講演会 論文集，pp.83～84 (B01)
発表内容　：口頭発表
学会開催日：2017 年 02 月 22 日～23 日

２）海洋理工学会 H29 春季大会
発表題目　：干潟用多足歩行ロボットの開発-ROS を使ったシミュレーション-
著　者　　：中村圭，田原淳一郎，齋藤幹大
掲載論文集：海洋理工学会 平成 29 年度春季大会 講演論文集，pp.61～64 (A4)
発表内容　：口頭発表
学会開催日：2017 年 06 月 08 日～09 日

３）第 22 回 知能メカトロニクスワークショップ（IMEC 2017)
発表題目　：非線形同期および GA による多足歩行ロボットの移動システムの開発
著　者　　：中村圭，田原淳一郎
掲載論文集：第 22 回 知能メカトロニクスワークショップ　講演概要集，p.20 (3A1-5)
発表内容　：口頭発表
学会開催日：2017 年 08 月 26 日～28 日

４）第 60 回 自動制御連合講演会
発表題目　：干潟調査用の多足歩行ロボットの開発
著　者　　：中村圭，田原淳一郎
掲載論文集：FrSP1-9（論文集無し）
発表内容　：ポスター発表
学会開催日：2017 年 11 月 10 日～12 日

５）AROB 23rd(2018)

発表題目　：Motion control of multi legged walking robot for tideland research using the nonlinear synchronization phenomena

著　者　　：Kei Nakamura, Junichiro Tahara, Masakazu Koike, Feifei Zhang, Mikihiro Saito

掲載論文集：Program AROB 23rd 2018, p.34 (GS7-2)

発表内容　：口頭発表

学会開催日：2018 年 01 月 18 日～20 日

６）海洋理工学会 H30 秋季大会

発表題目　：干潟環境における海洋調査ロボットの開発

著　者　　：中村圭, 田原淳一郎, 加藤哲, 馬場尚一朗

掲載論文集：海洋理工学会 平成 30 年度秋季大会 講演論文集, pp.63～66 (A1)

発表内容　：口頭発表

学会開催日：2018 年 10 月 18 日～19 日

７）AROB 24th(2019)

発表題目　：Development of the environment survey robot corresponding to tideland

著　者　　：Kei Nakamura, Junichiro Tahara, Masakazu Koike, Feifei Zhang, Mikihiro Saito

掲載論文集：Program AROB 24th 2019, p.36 (GS7-1)

発表内容　：口頭発表

学会開催日：2019 年 01 月 22 日～25 日