東海大学大学院平成２８年度博士論文

剛性可変機構を有する
多関節グリッピングハンド
に関する研究

指導　小金澤　鋼一　教授

東海大学大学院総合理工学研究科
総合理工学専攻

玉本　拓巳

目次

第 1 章

序論

1.1　災害対応ロボットへの取り組み

1995 年に発生した阪神淡路大震災をきっかけに，日本国内では地震や津波などの自然災害による被害に対してロボットを活用しようとする動きが大きくなり，レスキューロボットの開発が本格的に開始した [1]．災害現場では，火災や家屋の倒壊，さらには水害や土砂崩れなど, 二次災害の発生する可能性が非常に大きく，救助者には危険が付きまとう．ロボットを救助活動に活用することで，人がこのような二次災害に巻き込まれる可能性を排除し，さらに人には進入不可能な経路を通ることで，迅速に，より広範囲に活動ができるという利点がある．

また 2011 年に発生した東日本大震災と呼ばれる地震と, それに伴う津波によりもたらされた福島第一原子力発電所事故は，災害現場にロボットを投入する必要性をさらに強く認識させた [2]．これらのことから，レスキューロボットに求められる作業は単なる人命救助のみでなく，密閉空間の調査や危険物処理など多岐に渡るようになり，災害対応ロボットと呼ばれるようになる [3]．特に国外では，軍事やテロ対策の手段としても注目されている．

文部科学省は 2002〜2006 年にかけて，「首都圏や京阪神などの大都市圏において大地震が発生した際の，人的・物的被害を大幅に軽減するための科学・技術基盤の確立」を目的とした，大都市大震災軽減化特別プロジェクト (大大特) を実施した [4]．そのプロジェクト内で，「レスキューロボット等次世代防災基盤の開発」が課題の一つとして挙げられている [5]．また現在進行中のプロジェクトとして，内閣府の革新的研究開発推進プログラム (ImPACT) で実施されている，タフ・ロボティクス・チャレンジが挙げられる [6]．これは，「極限の災害現場でも，へこたれず，タフに仕事ができる遠隔自律ロボットの実現」を目的とし，屋外ロボットに必要な基盤技術を発展させるためのプロジェクトである．これらのことから，災害対応ロボットに対して国がいかに力を入れているかを理解することができる．

実際に国内で運用されている災害対応ロボットをいくつか紹介する．図 1.1 は東京消防庁に配備されている，遠隔操作タイプのレスキューロボット「ロボキュー」である [7]．二つのマニピュレータを操作することで，服を掴むことで救助者をロボット内へ収容したり，危険物の回収などを行うことができる．コントローラにマニピュレータが受けた力をフィードバックさせることができるが，その感覚を利用して作業をこなすには高い熟練度が必要である．図 1.2 は株式会社テムザックが開発した，遠隔操作可能なレスキューロボット「援竜」である [8]．ロボキューと同様，二つのマニピュレータを有している．コントローラは操縦者の動きに同期するような機能を持ち，直感的な操作を実現することで，長時間にわたる救助活動でも負担が小さくなるという特長がある．これらに代表される災害対応ロボットにおいて，調査以外に何かしらの作業を目的に持つ場合，その成功率や作業効率は，操縦者の熟練度およびマニピュレータ・ハンドの性能に大きく依存するといえる．

図 1.1　救出ロボット：ロボキュー [7]

図 1.2　レスキューロボット：T-53 援竜 (左)，T-52 援竜 (右) [8]

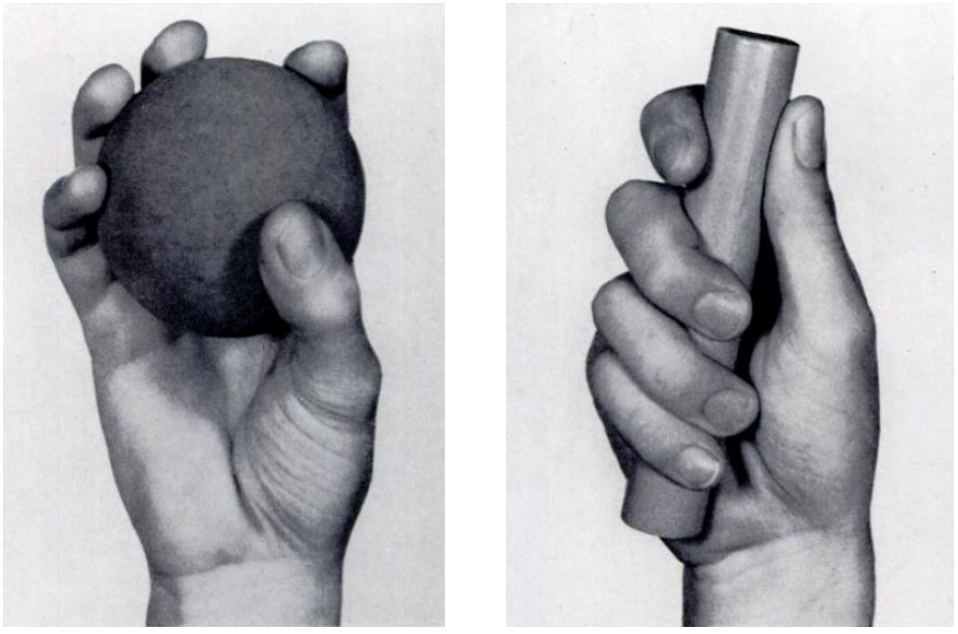
1.2　求められるロボットハンド

1.2.1　ロボットハンドのエンドエフェクタ

現場で作業を行う災害対応ロボットにおいて，最も基本となる作業の一つに瓦礫の撤去や要救助者の運搬などが挙げられることから，エンドエフェクタには把持を行うことを主目的とするハンドが選択されるのが一般的である．

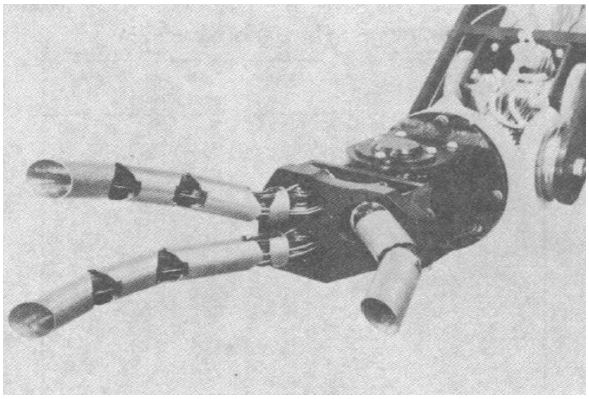
ロボットハンドによる把持は図 1.3 に示すように，その形態から大きく指先把持と包み込み把持に分類することができ，Napier の二大分類として知られている [9]．指先把持はその後の器用な操作へ繋げることを前提とした把持形態であり，精密把持 (precision grip) とも呼ばれる．一方，包み込み把持は頑丈な把握を目的とする把持形態であり，握力把持 (power grip) とも呼ばれる．マニピュレーションに関する研究は，主にこの分類を元に議論されている．

ロボットハンドの研究開発として，1979 年に電総研 (現産総研) から発表された，3 指で合計 11 自由度を持つ多指ロボットハンド [10] を皮切りに，4 指で合計 16 自由度を持つ The Utah/MIT Hand [11]，5 指で合計 12 自由度を持つ Robonaut hand [12,13]，同じく 5 指で合計 20 自由度を持つ Gifu Hand [14,15] など，多指多自由度を持ち，巧みな把持操作を行うロボットハンドの研究が盛んになった [16]．

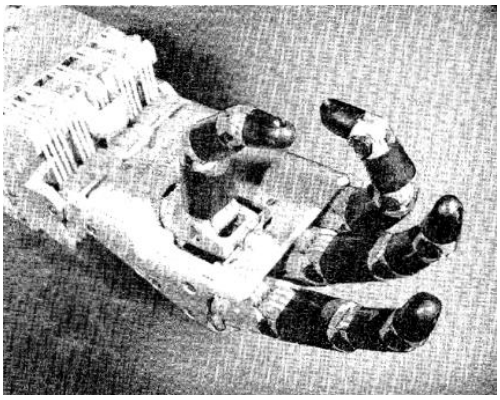


(a) 指先把持　　(b) 包み込み把持

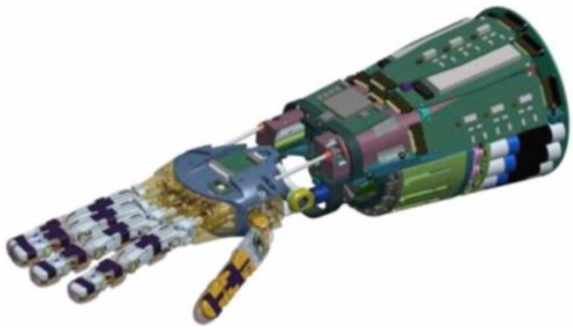
図 1.3　Napier の二大分類 [9]



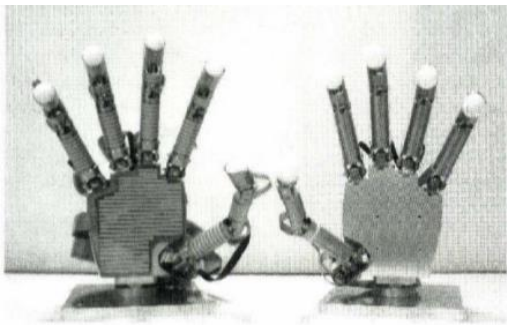
(a) 電総研のハンド



(b) The Utah/MIT Hand



(c) The Robonaut 2 hand



(d) Gifu Hand Ⅲ

図 1.4　さまざまな多指ロボットハンド [10, 11, 13, 15]

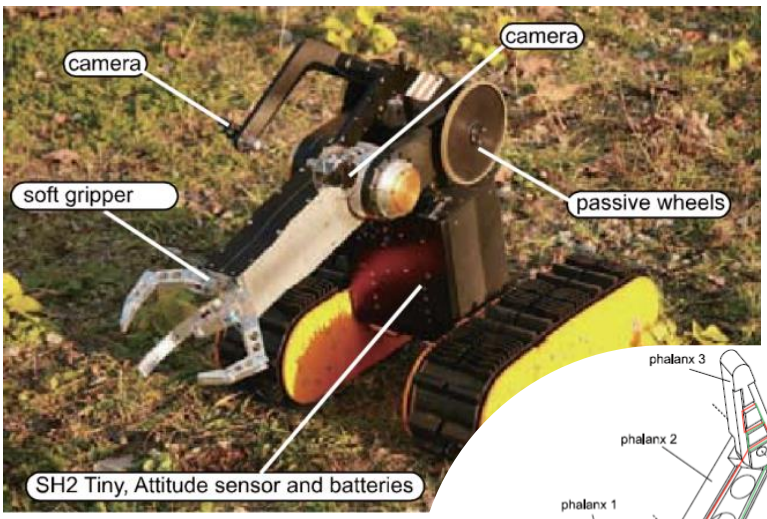
こうしたロボットハンドの研究が活発になった当初，マニピュレーションに関する研究は指先把持を中心に展開されてきた [17–19].

一方，包み込み把持に関しては Salisbury により，多関節指のリンク部分と対象物の接触を許容することにより，接触点数を増やすことでアクチュエータへの負荷を軽減させ，さらに外乱に対するロバスト性を得ることなどが報告された [20]．また Trinkle ら [21] がこの把持形態に対して Envelope Grasp（包み込み把持）と呼称し，この呼び名が定着した．その後，包み込み把持の研究は，把持のロバスト性に関する研究が主流となっている [22–26].

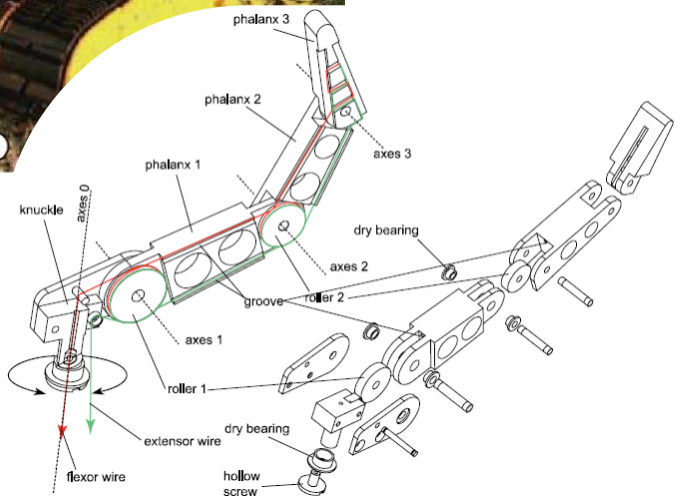
このように，包み込み把持は対象物との多点接触という特徴を持ち，対象物にある程度の外乱が働いても把持し続けることが可能である．また指先把持と比較して，対象物の形状やサイズも厳密に認識している必要はない．これらのことは多くの研究により報告されており，災害対応ロボットのように屋外で作業するロボットにおいては非常に重要なメリットと考えられる．よって災害対応ロボットのエンドエフェクタとしては，包み込み把持の機能性が高いハンドが適していると考える．

リンクが剛体であると仮定すれば，関節数が多いほど接触点数は多くなる．その特徴を活かした包み込み把持に特化した多関節ロボットハンド，グリッピングハンド（またはグリッパとも呼ぶ）をエンドエフェクタとして採用した例がある [27,28]．図 1.5 は，大大特プロジェクトで開発された災害対応ロボット「HELIOS VIII」および搭載されているグリッパ機構を示している．このロボットに搭載されているグリッパ機構は，次節で触れるソフトグリッパ機構をベースとしており，把持対象物の形状や材質に対する適応性が高く，多様な目標物を把持することができる．つまり，対象物の形状に沿った柔らかい把持を可能とし，そのことにより把持力の分散などが期待できる．

このような柔らかい把持は，人を搬送する場合や危険物の回収時など，把持対象に過度なダメージを与えてはいけない場合に非常に有効であり，現在運用されている災害対応ロボットに備わっていない特徴でもある．まだ運用実績はないようだが，今後の活躍が期待できる試みであると共に，災害対応ロボットが備えるべきエンドエフェクタの形の提唱として，有意義な研究である．



(a) HELIOS VIII



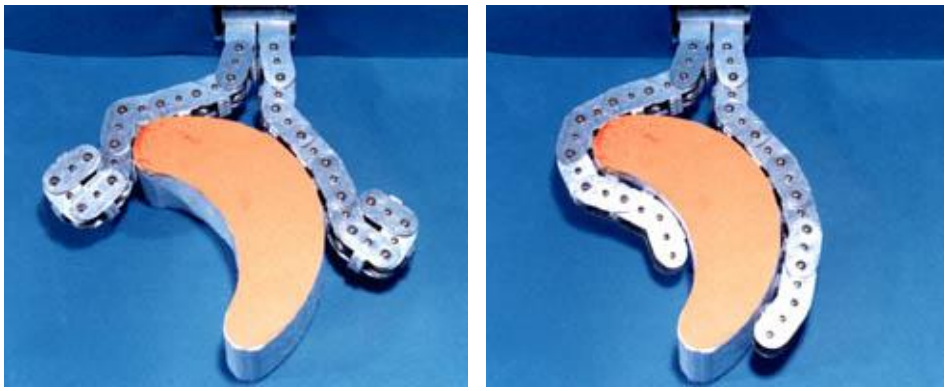
(b) HELIOS VIII のグリッパ機構

図 1.5　大大特プロジェクトで開発された HELIOS VIII およびグリッパ機構 [28]

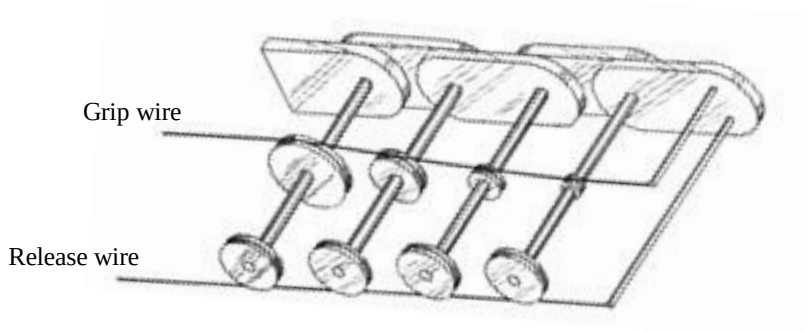
1.2.2　包み込み把持に特化したロボットハンド

前節で紹介した HELIOS VIII に搭載されているグリッパ機構のベースとなっている Soft Gripper [29, 30] は，広瀬らによって開発され，この分野におけるパイオニアとしてよく知られている．

図 1.6 は，Soft Gripper の包み込み把持の様子と内部の機構を示している．動作には根元から先端まで張られた 2 本のワイヤを用いており，Grip wire を牽引することで，各関節に把持方向に屈曲するトルクが発生し，包み込み把持を行う．また Release wire を牽引することで，逆方向のトルクが発生し，物体を解放する．Release wire の張力は各関節に等しいトルクを与えるが，Grip wire の張力は，各関節の回転中心からワイヤまでの距離 (モーメントアーム) を変化させることで，関節ごとに与えるトルクが異なる．図 1.6(b) では，指先に近づくに従い，ワイヤを通しているプーリ径が小さくなる様子が確認できる．よって，Release wire に一定の張力を与えた状態で Grip wire の牽引力を徐々に大きくすると，第 1 関節が最初に把持方向に屈曲するよう設計されている．なお，各ワイヤに対しそれぞれモータ 1 基を使用している．



(a) 包み込み把持の様子



(b) 内部の機構

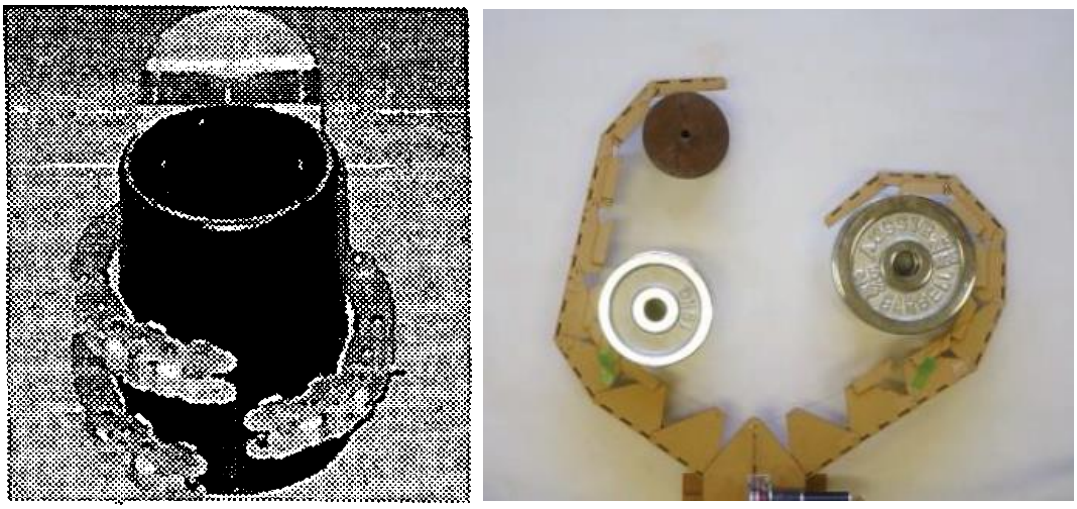
図 1.6　Soft Gripper

Soft Gripper に代表される，テンドンシステム (腱駆動方式) を用いたグリッパでは多くの場合，根元側の関節に発生するトルクを大きくすることで，対象物と非接触時には関節の屈曲動作が根元から起きるように設計されている．そして物体と接触した場合は，接触点よりも根元側の関節に屈曲を妨げるトルクが発生するため，接触点より先端の関節が屈曲する．この原理により，図 1.6 のような物体形状になじむ包み込み把持を可能としている．

このモーメントアームを調節するテンドンシステムの発想は多くの研究に採用されており [31–35]，先に紹介した災害対応ロボットのみならず，図 1.7 に示す，吉田らによる深海探査ロボット TAKO gripper [36] や，多賀らによるデブリの捕獲を目的としたグリッパ [37] などがある．

TAKO gripper の基本構造は Soft Gripper と同じであり，把持用と解放用の 2 本のベルトを，根元に設置したアクチュエータにより制御することで包み込み把持を行う．ただし，アクチュエータはモータ 1 基のみで動作させており，一方のベルトの牽引と同時に，もう一方のベルトの繰り出しを行う仕組みを取っている．

デブリ捕獲用グリッパは，根元から先端までワイヤを 1 本のみ通しており，把持方向の屈曲にはこのワイヤを牽引することでトルクを発生させ，解放方向には各関節に設置したトーションバネによりトルクを発生させている．多賀らによるこのグリッパは，Soft Gripper に基づく機構としては最も単純なものであると考えられ，また，その簡素さを生かしたグリッパ自体の収納性に注力しており，非常にコンパクトに折りたたむことが可能という特徴を持つ．



(a) TAKO gripper　　(b) デブリ捕獲用グリッパ

図 1.7　テンドンシステムを利用したグリッパ [36, 37]

このように多岐に渡るフィールドで求められる機構となっているが，その理由は主に以下の 3 点の特徴を持つためだと考えられる．

- 物体形状へなじむ柔らかい把持が可能である．
- 多自由度に対して劣駆動で動作可能である．
- センシングを必要としないシンプルな機構である．

特に 3 つ目の項目に関して，宇宙や深海，そして災害現場などの極限環境下では，衝撃や熱，電磁波といった電気/電子機器にとっての危険因子が多く存在する (図 1.8)．圧力センサなどを用いた力覚システムを利用して対象物の手繰りや把持力を制御することはよくある手法だが，基礎機能である把持をそれらのセンサに依存していた場合，センサの故障により把持物に危険が生じてしまう恐れがある．それに対し，紹介したテンドンシステムのグリッパは，どれもセンサに依存することなく包み込み把持の機能を十分に果たしている．つまり，本質的な安全を確保することが出来ているといえる．

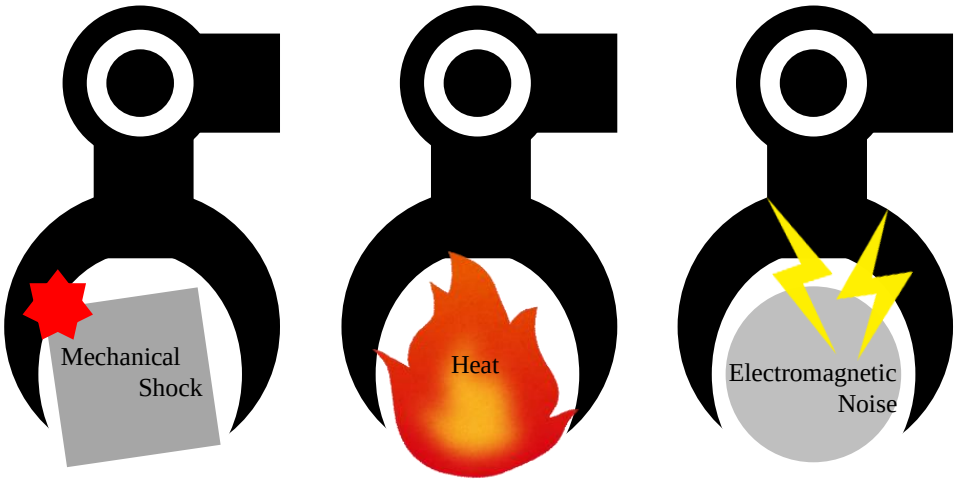


図 1.8　災害現場などで想定される事態

このように，いくつかの特筆すべき利点を備えているが，これらを運用するに当たり問題となる点が大きく二つ存在する．

一つ目は，姿勢の不安定性である．例えばモータによりワイヤを牽引しているとして，モータが一定角度回転した場合を考える．物体と非接触時でかつ水平状態であれば，あらかじめ設計した，最も大きなトルクが発生する関節が一定角度屈曲する．しかし，把持対象物以外の物体との接触が発生したり，あらかじめ決めた姿勢でアプローチできなかった場合に，設計した関節以外が屈曲してしまう事態が起こ

りえる．これはモータの回転力が各間接に分配されるシステムであるがために起きる問題である．つまり，モータの回転力という 1 つの入力が，各関節の回転という複数の出力に繋がっているためである．このことはさらに，各関節間で角度・トルクが相互作用していると解釈することもできる．

このテンドンシステムの概念をチャンバとシリンダを用いて図 1.9 に示す．錘は全て均一の重さであり，その質量により各関節のシリンダが可動域下限 (Release 方向) にある状態を基準とする．モータのシリンダを下に押し込めば，最も大きな力が発生する第 1 関節が Grip 方向へ動く．ここで，シリンダの動きは関節の屈曲，シリンダ直径をモーメントアームの大きさ，チャンバ内圧力により発生する力が屈曲用のワイヤ牽引により発生するトルク，錘による力は Soft Gripper に当てはめるなら Release wire が発生するトルクとみなせる．この図からも分かるとおり，各シリンダはチャンバを介して力を伝達することができ，いずれかの関節が Grip 方向への変位を持っている場合，それらが相互作用することは容易に想像できる．

二つ目は，床や壁に密着している物体に対して，アプローチ可能な角度が限定されてしまうことである．ピンチング動作を想定した機構ではないため，対象物の裏側に障害物がある場合，把持を達成する術を持たないことが原因である．ピンチング動作を行えない理由は，指先が接触した後にモータの回転力を伝達しようとすると，そのトルクは指先ではなく負荷の小さい他の関節へ伝わってしまうためである．

これらの理由から，従来提案されてきたテンドンシステムでは災害対応ロボットとして不十分である．また包み込み把持を行うハンドとして，材質そのものを柔らかくしたハンドに関しても報告が多数されている [38–42]．しかし，これらに関しては災害現場で使用するには最大把持力や損傷といったアクシデントが懸念されるため，そのような場面では頑丈な骨格を持ったハンドが望まれる．

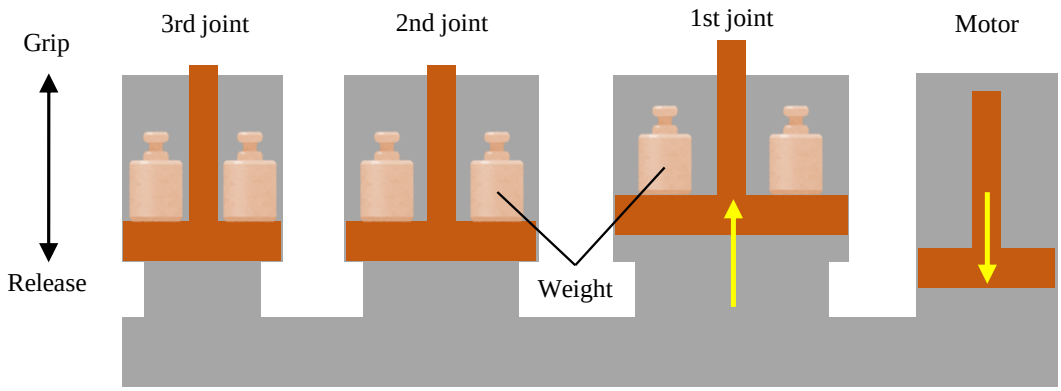


図 1.9　一般的なテンドンシステムの概念

1.3　本研究の目的

本研究では，災害対応ロボットに取り付けるエンドエフェクタとして有効であると考えられる，多関節グリッピングハンドの開発を目的とする．既に述べたように，従来提案されてきた包み込み把持に特化した多関節グリッピングハンドは，多関節であることと劣駆動であることを両立させるため，関節間で変位や力の相互作用が存在した．それが原因となり，姿勢を把握できなくなる状態が存在することを示唆した．またこうしたハンドに適応可能で，災害現場などの極限環境下で有効なセンシング手法についてもあまり議論されてこなかった．遠隔操作や人工知能で運用される可能性が高い災害対応ロボットおよびそのハンドにおいては，人の目視に頼ることができないため，ある程度のセンシングが必要であると考えられる．

よって本研究では，従来のグリッピングハンドが有している利点を引き継ぎつつ，これらの課題をクリアするハンド機構を提案する．既に述べた内容と重複する部分もあるが，本論文で提案する多関節グリッピングハンドが満たすべき要件について，以下のように 5 つの項目を設定する．

・物体形状に沿った柔らかい把持を可能とする

災害対応ロボットはしばしば，人の搬送や危険物の回収など，把持対象に過度な衝撃や負荷を与えてはいけない場面に遭遇する．物体形状になじむ把持は接触点数を増やし，把持対象物にかかる負荷の軽減が可能であることから，重要な特徴であるといえる．本項目は従来の提案機構で既に達成されている特徴であり，継承すべきである．

・シンプルな制御で動作可能な劣駆動機構である

災害現場などの極限環境下では，衝撃や熱，電磁波といった，センシングに必要な電気/電子部品にとっての危険因子が多く存在する．よって，それらの影響を受けやすい機器に頼らず動作可能であることで，本質的な安全を確保できる．本項目はそのために必要であり，また前の項目と同様に，従来の提案機構で既に達成されている特徴であり，継承すべきである．

・状況に応じて関節剛性の制御を可能とする

把持対象物がある程度の未知情報を持っている災害現場において，把持している対象物から受ける力は，ハンドが壁や床と接触することで生じる力，不整地を走行する際の振動および重力によって受ける力などと，機構上では同

質の力として認識され，メカニカルに区別することは現実的ではない．しかし，柔らかい把持を実現するには，把持対象物から受ける力に対しては柔軟な関節が必要であり，把持するためにアプローチする段階では，正確な動作を阻害する力に対して頑丈でなくてはならない．よって，現在の状況によって，関節剛性を調節する必要があることを提唱する．ロボットの剛性可変機構に関しては様々な研究が行われているが [43]- [45]，包み込み把持に特化したグリッピングハンドに適応させることに関しては，これまで議論はあまりされてきていない．

・本質安全を確保した力覚システムを有する

災害対応ロボットは，遠隔操作や人工知能によるオートでの制御で運用される可能性が高いので，人の目視に頼ることができない．よってある程度のセンシングが必要であると考えられるが，上で述べたように本質的な安全を損なってはならない．例えば，指部表面に圧力センサを設置して対象物にかかる負荷を制御することなどは，一般的なセンサの運用方法であるが，センサに不備が生じた場合は対象物に危険が生じてしまう．そこで，外界接触部にセンサを取り付けないという条件に基づく力覚システムの必要性を提唱する．こうしたセンサの運用方法については，これまで議論がほとんどされてきていないが，必要な特徴であると考える．

・包み込み把持に加えピンチングを可能とする

様々な場面で対象物を確実に把持するためには，時にピンチングを要求される場面がある．対象物が床や壁に密着しており，かつアプローチ可能な角度が限定されてしまうような場合である．ワイヤ・プーリを用いたテンドンシステムに代表される従来の提案機構では，多関節という特徴を際立たせた場合，ピンチング動作を行うことができない．ピンチングが可能であることは必要な特徴であり，かつその後の包み込み把持への移行をスムーズに行えることが望ましい．

これら 5 項目を満たすことで，災害対応ロボットとして，瓦礫の撤去・要救助者の搬送，危険物の除去など，把持を必要とする作業をより効果的に達成することができると期待する．

1.4　本論文の構成

本論文は「剛性可変機構を有する多関節グリッピングハンドに関する研究」と題し，全 5 章によって構成されている．各章の概要については以下の通りである．

第 1 章では研究目的までを示した．国の災害対応ロボットへの取り組みを紹介し，それらのロボットが備えるべきエンドエフェクタについて考察した．そこから多関節グリッピングハンドが有効であるという結論に至り，従来のグリッピングハンドの問題点を明らかにした．また，それらを踏まえて目指すべきハンドのコンセプトを目的として挙げた．

第 2 章は，本研究の基礎となるグリッパ機構について説明する．本研究では 2 入力 1 出力の差動機構を重要な要素として使用している．差動機構を利用したロボットハンドの研究例から，それらの利点，また本研究への応用方法について説明し，この差動機構を用いた 1 関節の駆動原理，剛性可変機構，連鎖駆動についてそれぞれ説明する．

第 3 章では，2 章で提示したグリッパ機構の動力学シミュレーション手法について述べる．物体把持を行うシミュレーションについて，運動要素として挙げられるリンク機構，モータ，把持物の運動方程式の導出および計算手法について述べる．また把持物とハンド機構の接触処理について述べた上で，最も基礎的な機能である包み込み把持のシミュレーションを行い，考察する．

第 4 章では実際に製作した実験機の概要，実験で必要な各種センサの扱い，制御系についてまず述べ，そして実験と動力学シミュレーションによる評価について論じる．評価は 1 章で挙げた目的に沿って，物体形状へなじむ包み込み把持および把持力の分散，剛性可変機構，力覚システム，ピンチング能力についてそれぞれ行う．

第 5 章では本研究の結論を述べると共に，今後の展望について述べる．

第 2 章

グリッピングハンド機構の提案

2.1　はじめに

本章では，序論で述べた要件を満たすような関節機構について提案する．差動歯車機構というシステムを用いるが，その概要および差動歯車機構を利用している研究について紹介し，それらの利点を示すと共に，本研究にどのように応用するのかを説明する．

また，提案した関節機構を劣駆動で運用するためのシステムについても検討し，最終的に，少数のアクチュエータで駆動し，関節剛性の調節も可能な，多関節グリッピングハンド機構について提案する．

また機構としては常時搭載ではなく拡張的なものになるが，ピンチングから包み込み把持への移行をスムーズに行うための指先機構についても提案する．

2.2　求められる関節機構

序論で定めた，多関節グリッピングハンドが満たすべき要件を再度掲げる．

(a) 物体形状に沿った柔らかい把持を可能とする
(b) シンプルな制御で動作可能な劣駆動機構である
(c) 状況に応じて関節剛性の制御を可能とする
(d) 本質安全を確保した力覚システムを有する
(e) 包み込み把持に加えピンチングを可能とする

これらを基に関節機構について検討するが，はじめに 1 関節について詳細に考察する．前提条件として，序論で述べたように各関節の変位およびトルクが相互作用しないような機構を提案する．

まず (a) を満たすために，関節はバックドライバビリティを持つ必要がある．ここで (b) にあるように劣駆動であることを考慮すれば，関節数よりも少数であるアクチュエータの特性や制御に依存して再現されるものではなく，関節毎に独立した機構により再現されることが望まれる．例えば，図 2.1 は入力軸 (モータ) と出力軸 (関節) の間に弾性体であるバネを用いることで，その機能を再現している．モータ・バネを合わせて，バックドライバビリティを有するアクチュエータと見なすことができる．しかし，バックドライバビリティの機能を再現しているのはバネであるので，単一のモータに対してバネ・関節機構を並列に接続すれば，関節毎に独立したバックドライバビリティを有する多関節機構を構成することができる．

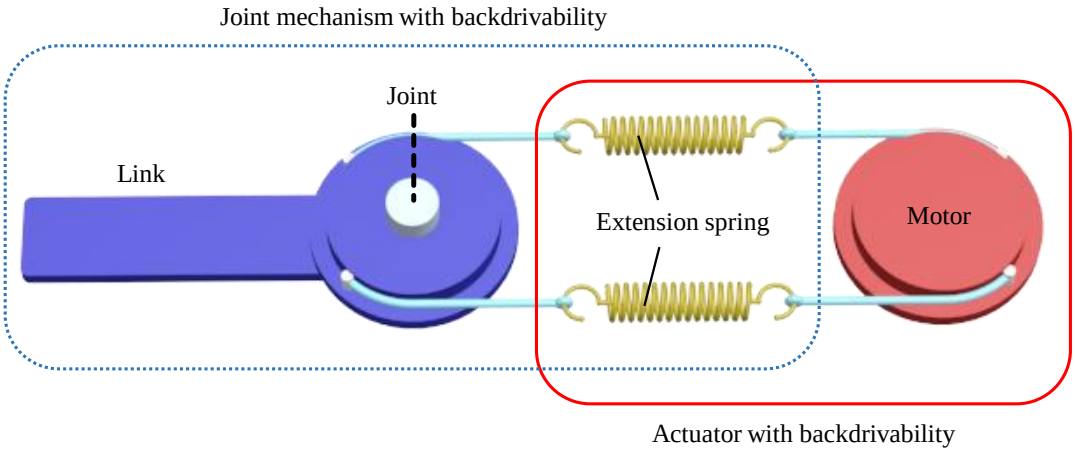


図 2.1　バックドライバビリティを持つ関節機構

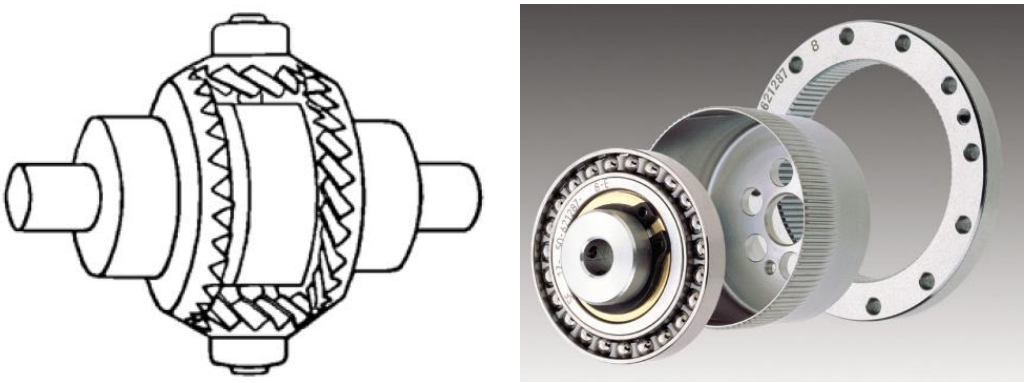
(d) の力覚システムは，バックドライブ量，図 2.1 ならばバネの伸び量を計測することで，現在の関節角度および外部負荷によりリンクに発生しているトルクを知ることができる．これは，バックドライバビリティを再現する部分がアクチュエータから切り離されていることで得られる利点である．対象物との接触部に圧力センサなどを用いたシステムと等価な効果を得ることが期待でき，また計測部分の保護もそれらに対して容易であるため，センサの故障などを予防することが可能である．

重要なのは (c) の関節剛性の制御であり，(d) のピンチング動作の実現はこの可否に依存するものである．(c) を満たすためには 1 関節の出力に対して「角度の制御」と「関節剛性の制御」という 2 つの入力が必要である．図 2.1 で再現するならば，モータと関節の軸間距離を制御する必要があるが，そのような機構はシステムの複雑化を招き，また新たな問題を引き起こしかねないので好ましくない．その他の 2 入力 1 出力の機構が望まれる．

2.3　差動歯車機構

2.3.1　差動機構の概要

2 入力 1 出力の機構として有名なのが，差動歯車機構 [46] である．差動機構やデフとも呼ばれ，よく知られている用途として，自動車がカーブを曲がる場合に左右の車輪に回転速度の差が生じるが，エンジンからの動力を左右の車輪，2 種類の回転速度に振り分ける役割を担っている．自動車の例ではかさ歯車を利用した機構であり，その他にも減速機としてよく用いられる遊星歯車機構やハーモニックドライブ [47] も差動歯車機構の 1 つである．



(a) かさ歯車を利用した差動機構　　(b) ハーモニックドライブ

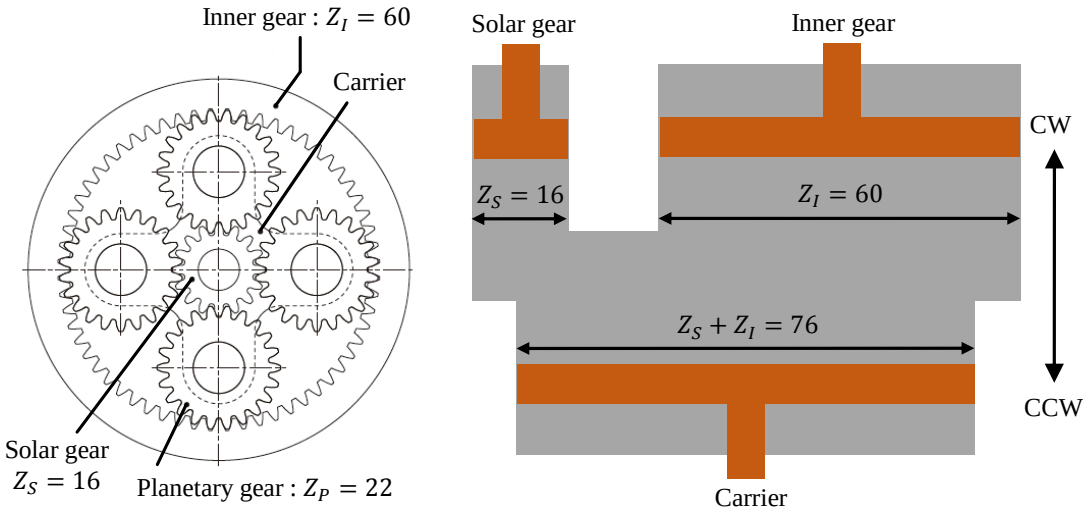
図 2.2　差動歯車機構の例 [46, 47]

いずれも 3 つの軸を持ち，それらの軸に入力されるトルクにより 1 つの状態が定まる．2 入力 1 出力と述べたが，最も回転しやすい軸が出力となるため，その時々により入力・出力軸は切り替わる．

遊星歯車機構を例にその関係を考察する．図 2.3 に機構の構成およびチャンバとシリンダを用いた動作の概念図を示す．遊星歯車機構で注目すべき 3 つの軸は，太陽歯車 (S : Solar gear)，内歯車 (I : Inner gear)，遊星歯車 (P : Planetary gear) の公転と同期するキャリア (C : Carrier) である．動作の関係式は次のようになる．

$$(Z_S)\,\theta_S = (Z_S + Z_I)\,\theta_C - (Z_I)\,\theta_I \tag{2.1}$$

ここで θ および Z は，それぞれ添え字の要素の回転角度および歯数を表す．3 つのシリンダ (= 軸) に対しそれぞれ入力があり，歯数によって定まる係数がかけられ，圧力 (= 入力の大きさを比較できる数値) が得られる．ここで最も圧力の小さいシリンダが出力方向への変位量を持つ．仮に 1 つのシリンダが完全に固定されている場合，残り 2 つのシリンダの入出力変位は比例関係となる．



(a) 構成　　(b) 動作の概念

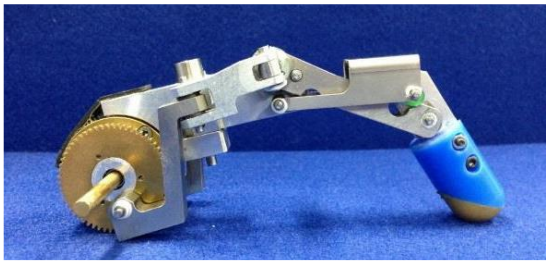
図 2.3　遊星歯車機構の構成 [48] と動作の概念

2.3.2　差動機構を利用したロボットハンド

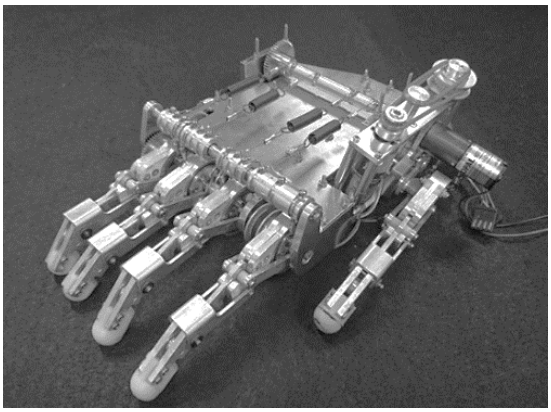
遊星歯車機構をロボットハンドに利用した研究として，人間と同等の機構自由度を持つが劣駆動で動作する，竹田らの指関節機構 [49] や，その内転外転方向の自由度を無くした指機構を用いた，佐藤らの 5 指ハンド [50,51] があり，図 2.4 に示す．これらは遊星歯車機構の要素の 1 つである太陽歯車にモータからの動力を与え，内歯車およびキャリアを指の第 1 関節，第 2 関節に連結している．キャリアの回転をバネにより拘束することで第 1 関節 (= 内歯車) から屈曲し，物体との接触などにより屈曲が阻害されると，第 2 関節 (= キャリア) が屈曲を開始する．

この機構は 1 つの遊星歯車機構の動作が複数の関節に影響を及ぼすことから，その関節間で変位およびトルクの相互作用が発生することが予想できる．人の指を模した機構として一定の効果を得られているようだが，さらに多関節化した場合に，ワイヤ・プーリを用いたテンドンシステムと本質的には同様の問題を持っている．

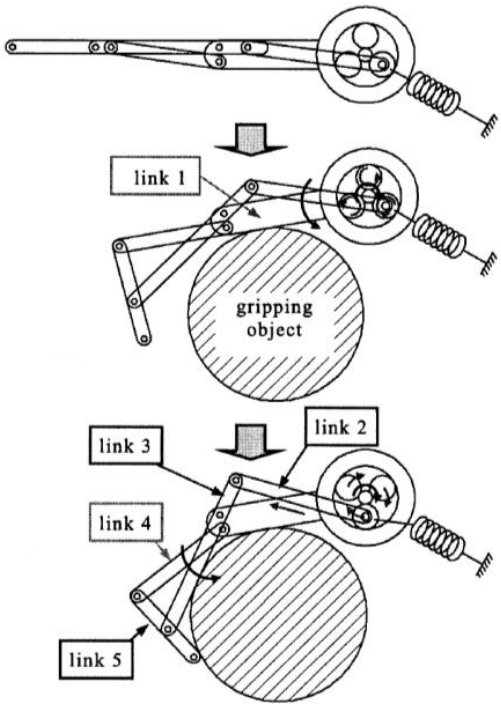
しかし，第 1 関節のみに焦点を当てると，キャリア，つまり第 2 関節がバックドライバビリティの出力先として機能していると解釈することができる．その結果に注目すると，遊星歯車機構はバックドライバビリティを持つ関節機構の要素として期待が持てる．



(a) 指関節機構



(b) 5指ハンド

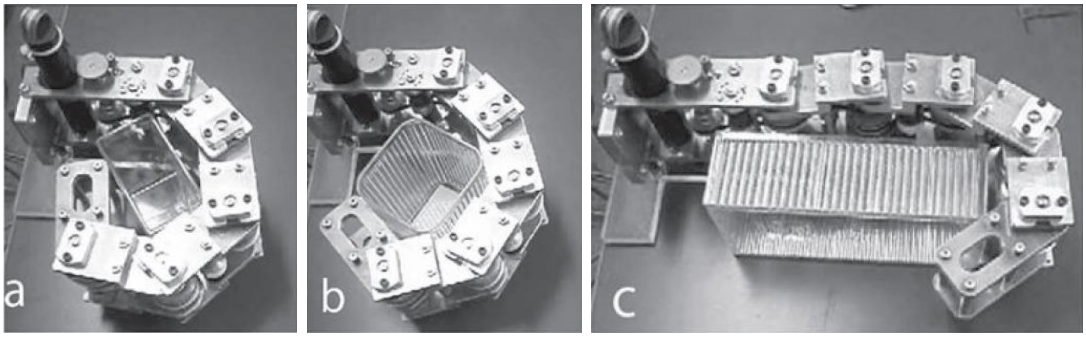


(c) 指関節機構

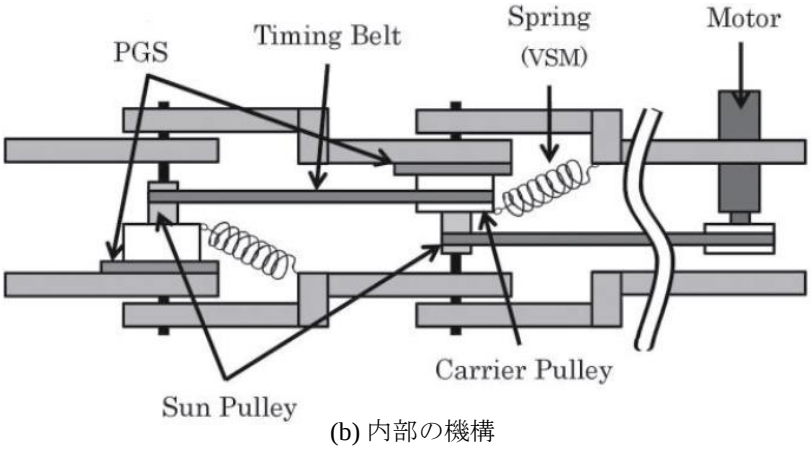
図 2.4　遊星歯車機構を利用した指関節機構および 5 指ハンド [49–51]

これに対し内田らは，遊星歯車機構を各関節に配置した多関節グリッパ [52] について報告し，また下野らによって，関節の剛性可変機構を搭載した改良機 [53] が報告された．これは，第 1 関節の太陽歯車へモータの動力を伝え，先端側リンクを内歯車に接続し，キャリアがバネを介して根本側リンクと接続されている．基本的にキャリアの回転が拘束されるため，内歯車 (= 先端側リンク) が回転する．仮に内歯車の回転が阻害された場合は，キャリアが回転し，それがバックドライバビリティとしての機能を果たす．内歯車とキャリアに生じる相対角度を，先端側関節の太陽歯車へ入力することで，劣駆動を成立させている．バックドライバビリティの機能を果たすバネが各関節付近に設置されていることから，目的とするような力覚システムの搭載は難しいと予想されるが，多関節グリッピングハンドとしてのその他の機能を構造的には満たしている．

しかし報告によれば，根本側の関節角度の影響をわずかに受けることで，物体を把持している場合には指先関節へトルクが伝わりづらくなることが理論上避けられない点や，機構の複雑さから 1 関節が持つ質量が肥大化し，水平状態でしか動作がままならない点などの問題が指摘された．



(a) 包み込み把持の様子



(b) 内部の機構

図 2.5　遊星歯車機構を用いた多関節グリッパおよび機構概要 [52, 53]

2.3.3　提案する差動機構

前項で示した複数の研究例より，差動歯車機構は目的とする関節機構に対し十分な効果を発揮できると予想する．しかし遊星歯車機構では，歯数比の選択肢が狭いことや，各関節の遊星歯車機構の間で動力を伝達する時に，必ず何か別の伝達要素を経由しなければならず，機構の肥大化が避けられない．そこで，極めてシンプルな差動歯車機構である，図 2.6 に示すような 2 段の減速機構を提案する．以降，差動歯車機構とは本機構を指すものとする．動作の関係式を次に示す．

$$(Z_l^2 - Z_s^2)\,\theta_L = (Z_l^2)\,\theta_C - (Z_s^2)\,\theta_A \tag{2.2}$$

ここで θ_L，θ_A，θ_C は，それぞれリンク，歯車 A，歯車 C の角度を示す．Z_l，Z_s は，差動歯車機構を 2 種類の歯数で構成した場合の大小それぞれの歯数である．よって，各歯車 (A，C，E1，E2) の歯数に対して次の式が成り立つ．

$$Z_A = Z_{E1} = Z_s = 20 \tag{2.3}$$
$$Z_C = Z_{E2} = Z_l = 30 \tag{2.4}$$

ここで歯数をそれぞれ 20 と 30 と置いたが，これは後の章で実際に使用している歯数である．しかし，差動歯車機構の動作の特徴を説明する上で，厳密な歯数は必要としない．ただし $Z_s = Z_l$ の場合は，式 (2.2) 左辺の値が 0 となり，いかなる場合でもリンクが回転しなくなってしまうため，必ず $Z_s \neq Z_l$ を満たす必要がある．

式 (2.2) において，歯数は設計変数であり既知であるので，未知数は各 θ の 3 つとなる．これは遊星歯車機構と同じであり，その構造と照らし合わせると，歯車 A を太陽歯車，歯車 C を内歯車，リンクをキャリア，また歯車 E (E1 と E2 による段歯車) を遊星歯車とみなすことができる．

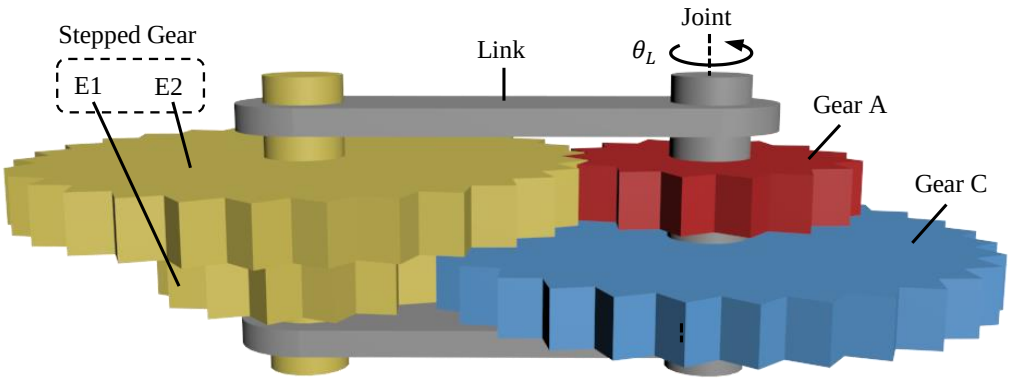


図 2.6　差動歯車機構

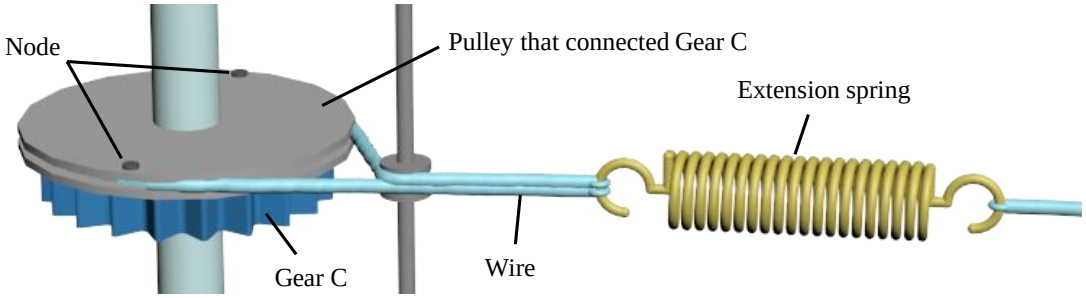


図 2.7　バックドライバビリティ生成機構

関節機構としては，歯車 A にモータからの動力を伝え，歯車 C へ引張バネからの力を伝える．なお，引張バネは図 2.7 に示すように回転を拘束する力として作用させる．リンクは重力や慣性力，外部との接触力といった力を受ける．設計条件として，モータからの入力最大値が，予想されるバネ力やリンクへ与えられる外力の入力値よりも大きな値をとるようにすることで，モータを角度制御することにより歯車 A は任意の角度を得るものとする．つまり，リンクおよび歯車 C への入力の大小によって，動作モードを概略的に分けることができる．動作モードは大きく 3 つ存在し，図 2.8 に示す．各動作の説明は次の通りである．

Motion1 歯車 A をモータにより回転させ，「リンクへの入力 < 歯車 C への入力」の場合の動作である．リンクに重力や外部との接触力といった環境外乱が働かない，またはその力が小さい場合，リンクへの入力が最も小さくなり，歯車 A の変位は全てリンクへと伝達される．

Motion2 歯車 A をモータにより回転させ，「歯車 C への入力 < リンクへの入力」の場合の動作である．何かしらの外力によりリンクの回転が阻害されており，その入力が，歯車 C と繋がるバネからの入力よりも大きい場合，歯車 A の変位は全て歯車 C へ伝わる．歯車 C の回転に伴い，バネには変位量に応じたポテンシャルエネルギーが蓄えられる．

Motion3 歯車 A の回転を停止させた場合の動作である．リンクに何かしらの外力が働きトルクが生じた場合や，バネが歯車 C を回転させるポテンシャルエネルギを蓄えている場合，それらの大小により，リンク・歯車 C の一方が出力となり動作する．力が釣り合う場合は停止状態となる．

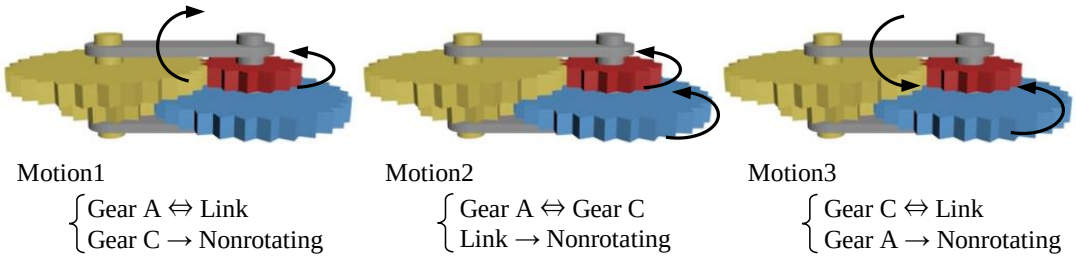


図 2.8　差動歯車機構の動作モード

この機構は歯車 C と繋がるバネがバックドライバビリティとして機能しており，このバネ力を制御することで，関節剛性の制御も可能である．バネの一端が自由であり機構的な拡張が可能であることから，提案する関節機構として必要な機能を満たせることが見込まれる．

また，差動歯車機構の動作の関係式として既に示した次の式について，その導出過程を説明する．

$$(Zl^2 - Zs^2)\,\theta_L = (Zl^2)\,\theta_C - (Zs^2)\,\theta_A \tag{2.2}$$

まず，噛み合っている 2 つの歯車の組み合わせ (A と E2，C と E1) について，それぞれ以下の式を立てることができる．

$$\theta_{E1} = \frac{Z_C}{Z_{E1}}(\theta_L - \theta_C) \tag{2.5}$$

$$\theta_{E2} = \frac{Z_A}{Z_{E2}}(\theta_L - \theta_A) \tag{2.6}$$

ここで，θ_{E1}，θ_{E2} は歯車 E1，E2 の角度を，Z は各添え字の歯車における歯数を表す．E1 と E2 は段歯車であることから，

$$\theta_{E1} = \theta_{E2} \tag{2.7}$$

が成立する．この式に式 (2.5)，式 (2.6) を代入することにより，次の式を得る．

$$(Z_{E2}Z_C - Z_{E1}Z_A)\,\theta_L = (Z_{E2}Z_C)\,\theta_C - (Z_{E1}Z_A)\,\theta_A \tag{2.8}$$

先に示した式 (2.2) は歯数を 2 種類しか使用しないという仮定の下で成り立つ式であり，式 (2.8) が一般化された式である．

2.4　多関節列駆動機構への応用

提案した差動歯車機構は，多関節の劣駆動機構へ発展されなければならない．基本的には，各関節に差動歯車機構を用いた関節機構を配置すれば良いのだが，ここで重要なのが，バックドライバビリティを再現しているバネの設置場所である．力覚システムとして応用するのであれば，このバネの伸縮量を計測しなければならない．衝撃・熱・電磁波といった因子に影響を受けにくいシステムのためには，実際に駆動し，物体と接触する指部ではなく，第 1 関節よりも根元側である基部に設置されるべきである．これらのことから，差動歯車機構の要素である歯車 A・歯車 C に対して (図 2.6 参照)，それぞれ基部から，モータの動力・バネによる動力を伝達する必要がある．

そこで，根元側に位置する関節角度の影響を受けない，動力の伝達機構について検討する．次の式は差動歯車機構の関係式である，式 (2.2) を変形させたものである．ここで，添え字の i は根元から数えた関節番号を意味する．

$$\theta_{Li} = \frac{Zl^2}{Zl^2 - Zs^2}\theta_{Ci} - \frac{Zs^2}{Zl^2 - Zs^2}\theta_{Ai} \tag{2.9}$$

根元側の関節角度の影響を受けないということは，根元側の関節が回転した場合においても $\theta_{Li} = 0$ を維持する．そこで，例として第 3 関節のみの屈曲動作について焦点を当て，図 2.9 に示すような歯車列を構成する．図に示した歯車列は大小 2 種類の歯数の歯車のみで構成されているものとする．第 3 関節の歯車 A(A3) に直接動力を伝達する歯車を A3’，また歯車 C(C3) のそれを C3’ とする．歯車 C3’ に段歯車を用いれば，第 1 関節から A3’・C3’ 両歯車へ至る歯数の並びを図のように揃えることが可能となる．よって第 1 関節，第 2 関節の屈曲により各歯車に回転が生

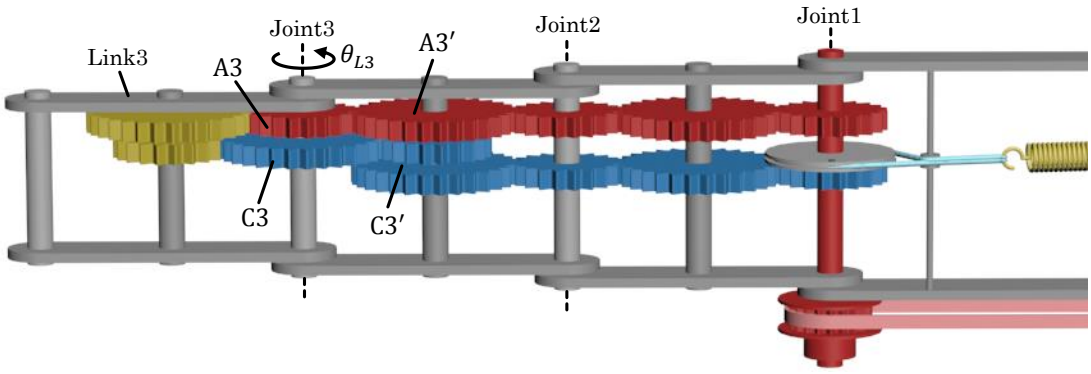


図 2.9　各差動歯車機構への動力伝達

じても，そのことに起因する A3’・C3’ の相対角度は常に 0°となる．この両歯車の角度を θ_x と置くと，差動歯車機構の歯車 A 及び C への伝達は次のように求められる．

$$\theta_{A3} = -\frac{Z_l}{Z_s}\theta_x \tag{2.10}$$

$$\theta_{C3} = -\frac{Z_s}{Z_l}\theta_x \tag{2.11}$$

これらを差動歯車機構の式 (2.9) に代入すると，

$$\theta_{L3} = 0 \tag{2.12}$$

となる．このことから，第 3 関節のリンク角度は，第 1 関節に位置する 2 つの歯車の相対角度によって決定するといえる．

この機構を 3 関節分の動作に拡張したモデルを図 2.10 に示す．図中の同軸上で

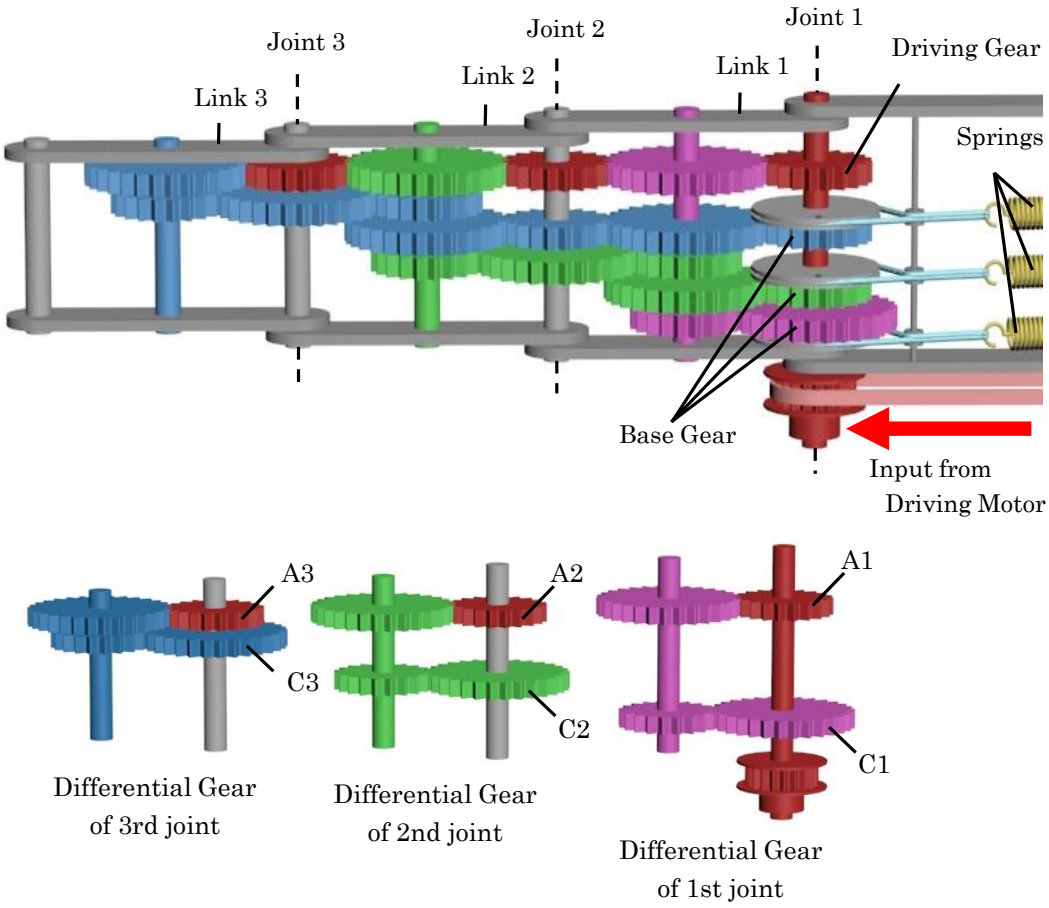


図 2.10　3 関節の機構モデル

同色で示された要素は連結していることを意味する．また，同図には理解の一助のために各関節の差動歯車機構を抜き出したものも示している．第 1 関節に位置し，各差動歯車機構の歯車 A へ動力を伝達する歯車をドライビングギアと呼び，同じく第 1 関節に位置し，各歯車 C へ動力を伝達する歯車をベースギアと呼ぶものとする．ドライビングギアは 1 つのみであり，屈曲動作を行うためのモータ (ドライビングモータ) の動力を第 1 関節から順に，全ての関節へと伝達する．ベースギアは関節ごとに独立して存在し，それぞれ引張バネで回転を拘束するような作用を受けている．

リンク角度は，ドライビングギアとベースギア角度によって求めることができ，その式は次のようになる．

$$\theta_{L1} = \frac{Zl^2}{Zl^2 - Zs^2}\theta_{B1} - \frac{Zs^2}{Zl^2 - Zs^2}\theta_D \tag{2.13}$$

$$\theta_{Li} = \frac{Zs^2}{Zl^2 - Zs^2}\theta_{Bi} - \frac{Zs^2}{Zl^2 - Zs^2}\theta_D \qquad (i > 1) \tag{2.14}$$

ここで，θ_{Bi} は各関節に対応するベースギア角度，θ_D はドライビングギア (またはドライビングモータ) の角度である．

図 2.10 からも分かる通り，第 1 関節の式 (2.13) は式 (2.9) で示した差動歯車機構の式そのものである．第 2 関節以降は，ベースギアから歯車 C への伝達で段歯車を介しているため，ベースギアの項が異なっている．よって，リンクに重力や外部との接触力といった外力が発生しておらず，全てのベースギア角度 θ_{Bi} が 0 である場合，同じ式になることが確認できる．

これらのことから，全関節の基準となる屈曲角度はドライビングギアの角度によって決まり，それは全ての関節において同一角度であると言える．

2.5　剛性可変機構

第 1 関節より根本側である基部について，図 2.11 に示す．モータは 1 指につき 2 基使用する．屈曲動作用のドライビングモータの動力は，ベルトとプーリを介し，ドライビングギアへと伝達する．

もう一方の VSM モータは関節剛性を制御するモータである．VSM とは，Variable Stiffness Mechanism (剛性可変機構) の略である．VSM 軸を回転させることで，ワイヤとプーリを介して，バックドライバビリティを再現している引張バネのもう一端の位置を制御する．

このシステムによりベースギアに発生するトルクは非線形となる．VSM モータで制御する変位量を基準バネ伸び量とし，例として，その量が 2[mm] と 8[mm] の場合のベースギア角度と生じるトルクについての関係を図 2.12 に示す．なお，バネ係数は 1[N/mm] とする．これらの剛性値 S は，角度の微小変化とトルクの微小変化の比，

$$S = \Delta T_B / \Delta \theta_B \tag{2.15}$$

で表され，角度の微小変化を $0 \rightarrow 5[deg]$ とすれば，2[mm] の場合は $S = 1.88[Nmm/deg]$，8[mm] の場合は $S = 6.68[Nmm/deg]$ となり，基準伸び量に伴い関節剛性が制御される．

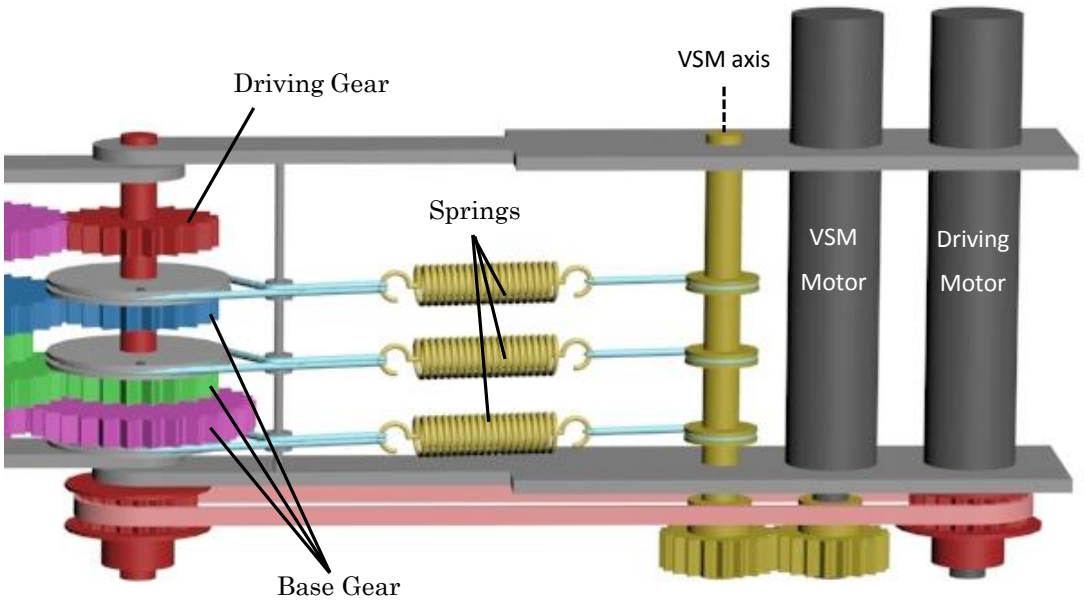


図 2.11　剛性可変機構

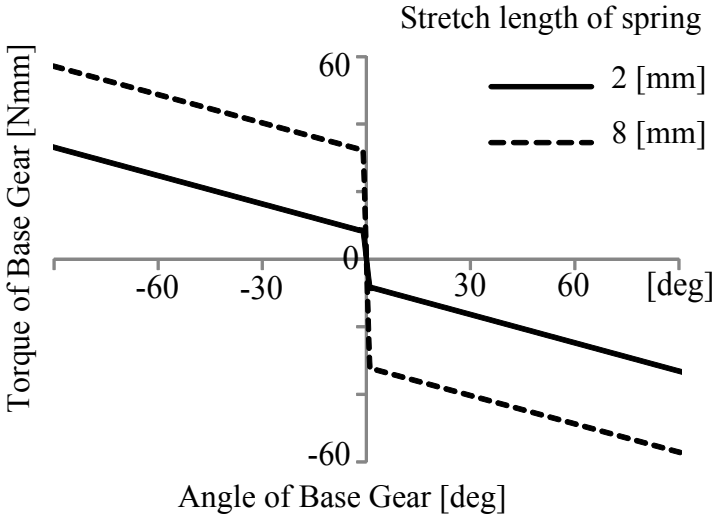


図 2.12　ベースギア角度と発生するトルクの関係

2.6　ピンチングのための指先機構

複数の指機構を運用する場合，指先でつまみあげるピンチングが可能となる．ピンチングは地面に接している物体を上から拾い上げるなど，屋外で運用するには必須の機能であると考える．しかし本機構は，器用なハンドリングではなく包み込み把持を行うことを重視しているため，ピンチングから包み込み把持へいかにスムーズに移行するかが重要である．手繰りによりそれを達成する研究は存在するが，本研究では機構的にその機能を備えることを目指す．

図 2.13 にその機構の概要を示す．提案した劣駆動機構は，各関節の差動歯車機構の要素を直列につなぐことで，モータの動力を各関節へ伝えている．つまり，先端関節の差動歯車機構からさらに先へ，モータの動力を伝えることが可能であることを意味する．図 2.6 で示した歯車 E の動力をプーリとベルトを介して，自由に回転するよう取り付けたローラへ伝える．リンクが屈曲動作中はこのローラは同方向へ回転するため，結果的に物体を内側へ引き寄せる働きをする．またモータからベースギアへ動力が伝わる場合は必ず歯車 E を経由するため，ドライビングモータが物体を包み込む方向へ駆動している場合は，リンクが停止しているか屈曲しているかに関わらず内側へ引き寄せる働きをする．

この機構により，ピンチングから包み込み把持への移行がスムーズに行えることが期待できる．また本機構は実験的なものであり，ピンチングの機能を評価する場合のみ搭載する機構とする．

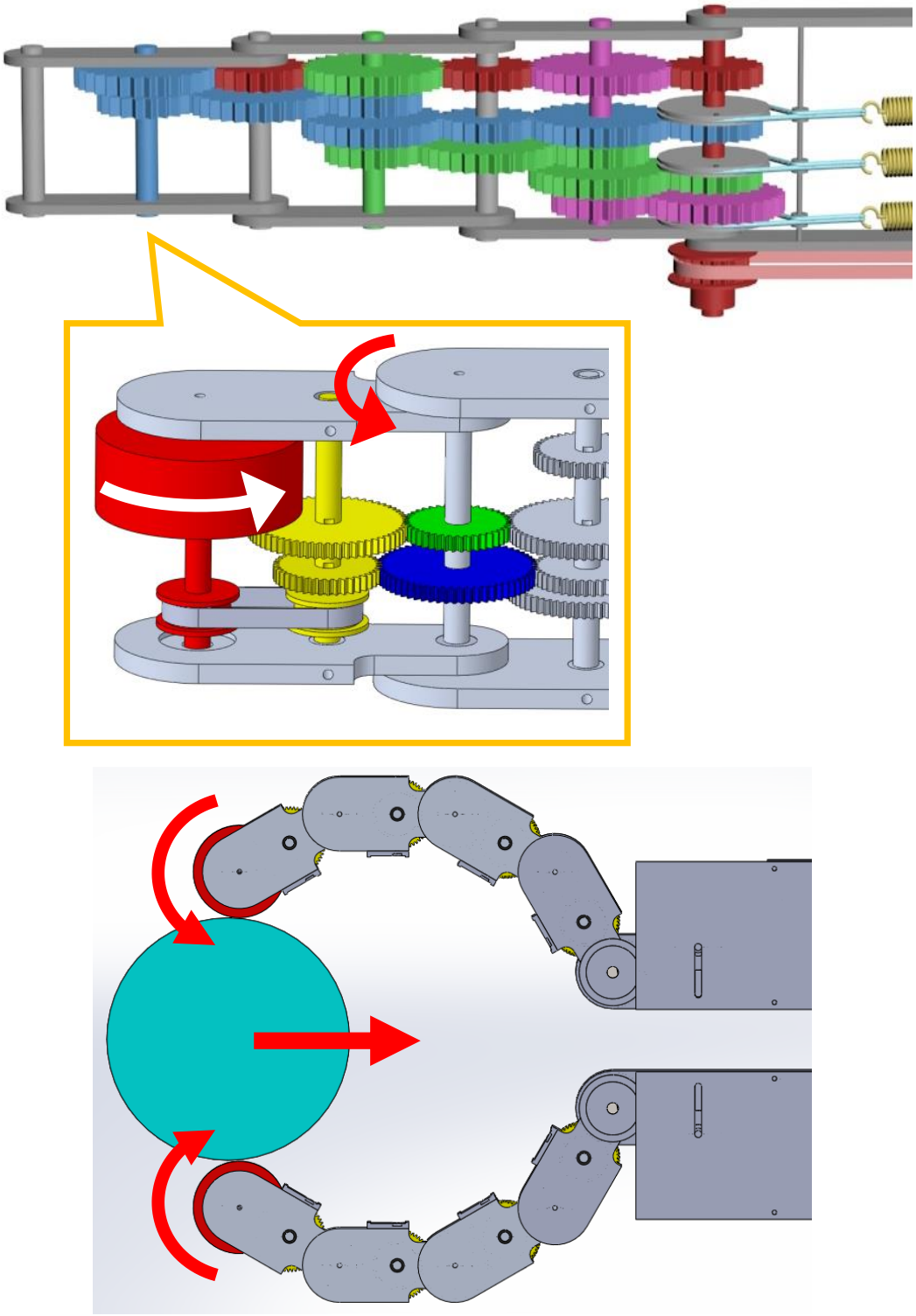


図 2.13　指先ローラ機構

2.7　まとめ

本章では，序論で述べた多関節グリッピングハンドが満たすべき以下の要件に沿った関節機構の提案を行った．

(a) 物体形状に沿った柔らかい把持を可能とする
(b) シンプルな制御で動作可能な劣駆動機構である
(c) 状況に応じて関節剛性の制御を可能とする
(d) 本質安全を確保した力覚システムを有する
(e) 包み込み把持に加えピンチングを可能とする

一関節に求められる機能は，角度を制御可能であることと剛性を制御可能であることの 2 つである．この 2 入力の要求を満たすため，差動歯車機構を導入した．使用する差動機構は 2 入力 1 出力という特徴を持ち，入力の一つはモータからのトルクを与え，もう一方はバネによる受動的なトルクを与える．能動的な角度制御を行う場合はモータトルクがそのまま関節へ出力されるが，関節が外部との干渉により固定された，または過剰に回転した場合に，バネにエネルギーを蓄えるように動作する．バネの復元力が外部干渉に対する抵抗力となり，バネ力を制御することで関節剛性を可変とする．本機構は劣駆動で多関節に拡張可能であり，目的とする要件を満たす機構の提案を行った．

第 3 章

動力学シミュレーションの導入

3.1　はじめに

本章では，提案した機構について評価するためのシミュレーション手法についての説明と，最も基礎となる動作である屈曲動作と包み込み把持のシミュレーションについて述べる．

動力学シミュレーションを導入する理由は，実機を用いての実験では，接触力など厳密な測定が困難な情報が存在し，それらの予測のためや，またバネ係数などのパラメータ設定に役立てるためである．本章では評価は基礎的な動作のみにとどめ，詳細なシミュレーションによる評価は，後の章で実機による実験と比較しながら示すものとする．

シミュレーション手法に関しては，運動方程式の導出には基本的にラグランジュ法を用いており，各運動要素に関してその導出過程を示す．また，運動要素同士の接触などといった干渉の扱いについても説明する．

3.2　対象とする運動モデル

図 3.1 に 1 指および 2 指のシミュレーションモデルを示す．シミュレーションにおける運動要素として，指部であるリンク機構，モータ，把持物が挙げられ，それらをモデル化した図である．$\theta_1 \sim \theta_i$ は基部側リンクに対する各リンクの相対角度，θ_D，θ_{VSM} はそれぞれドライビングモータ，VSM モータ角度，G_x，G_y，θ_G は物体座標および角度である．モータは数値のみで表現され，またリンク機構および把持物は，点・円・直線のいずれかの要素の組み合わせで構成される．

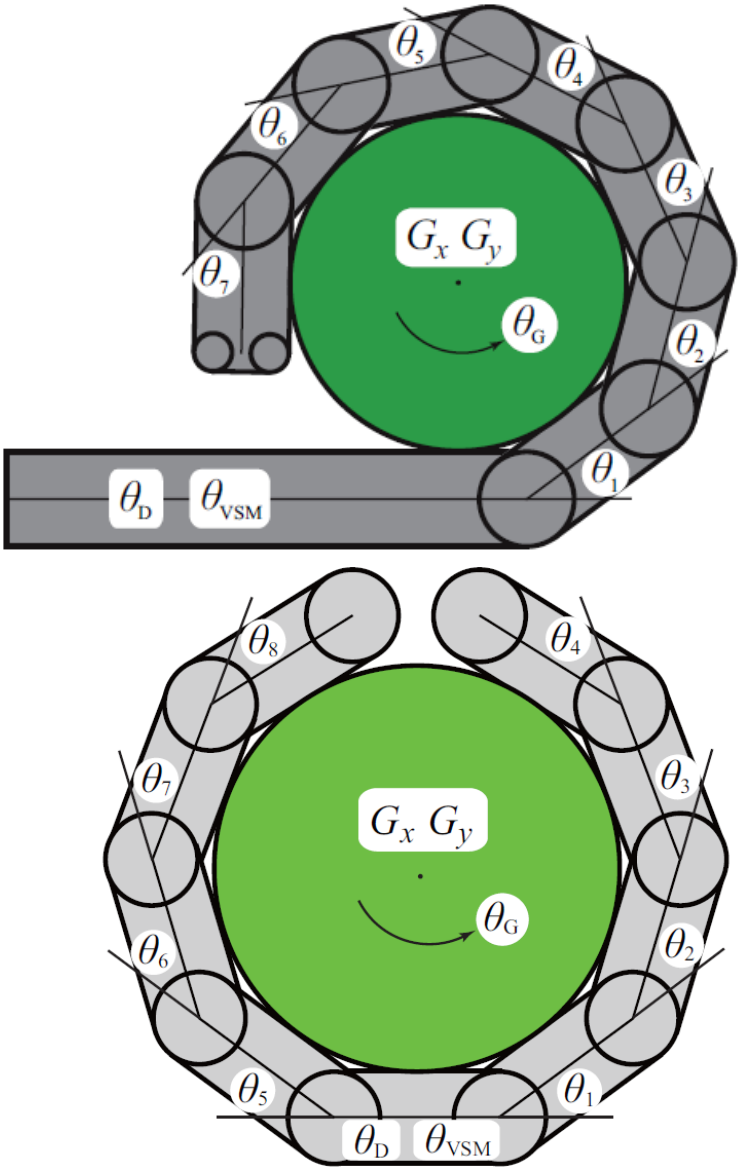


図 3.1　シミュレーションモデルと運動学的変数

3.3 運動方程式の導出

3.3.1 リンク機構の運動方程式の導出

以下，指部の運動方程式の導出について説明する．使用するパラメータ記号について次に示す．なお，i は関節角度を意味する．

l_i	リンク長	P_i	リンク重心座標
m_i	リンク質量	$\dot{P}_i$	リンク並進速度
I_i	リンク慣性モーメント	c	リンク回転散逸係数
J_i	関節座標	d	リンク並進散逸係数

本モデルは，直交座標系よりも極座標系を用いることで問題が簡単になるため，このような場合に利点があり導出が容易なラグランジュ（Lagrange）の運動方程式を用いる．指の相対角度を表す一般化座標 $\theta_1 \sim \theta_7$ を q，それに対応する一般化力を Q とし，関節座標 J_i，リンク重心座標 P_i およびその並進速度 $\dot{P}_i$ を求める．

$$q = [\theta_1 \ \ \theta_2 \ \ \theta_3 \ \ \theta_4 \ \ \theta_5 \ \ \theta_6 \ \ \theta_7]^T \tag{3.1}$$

$$Q = [Q_1 \ \ Q_2 \ \ Q_3 \ \ Q_4 \ \ Q_5 \ \ Q_6 \ \ Q_7]^T \tag{3.2}$$

$$J_1 = [x0 \ \ y0]^T \tag{3.3}$$

$$J_i = J_{i-1} + l\,[cos\theta_{i-1} \ \ sin\theta_{i-1}]^T \quad (2 \leqq i) \tag{3.4}$$

$$P_i = J_{i-1} + l\,[cos\theta_{i-1} \ \ sin\theta_{i-1}]^T\,/2 \tag{3.5}$$

$$\dot{P}_i = \partial P_i / \partial t \tag{3.6}$$

ここで $x0$，$y0$ は任意の定数である．また，すでに説明した通り指機構はバネを有しており，バネのポテンシャルエネルギが存在する．しかし使用している引張バネには初期張力が存在し，理論上バネにより発生しているトルクが不連続に変化する場合（BaseGear が $0[deg]$ 付近の場合）がある．このような不連続に変化するエネルギを数式で表現することは困難なため，バネにより発生している力は外力 Q として計算に含めることにする．その計算手順については本項最後に説明する．

続いて，運動エネルギ T，ポテンシャルエネルギ U，散逸エネルギ V 及びラグラ

ンジアン L を求める．T はリンク慣性力を，U は $-Y$ 軸方向の重力を，V は並進と回転の減衰力を含む．尚，L は運動エネルギ T とポテンシャルエネルギ U により与えられるパラメータである．

$$T = \sum_{i=1}^{7} \{ \frac{m_i}{2} (\dot{P}_{xi}^2 + \dot{P}_{yi}^2) + \frac{I_i}{2} \dot{\theta}_i^2 \} \tag{3.7}$$

$$U = \sum_{i=1}^{7} mgP_{yi} \quad (g = 9.81) \tag{3.8}$$

$$L = T - U \tag{3.9}$$

$$V = \sum_{i=1}^{7} \{ \frac{d}{2} (\dot{P}_{xi}^2 + \dot{P}_{yi}^2) + \frac{c}{2} \dot{\theta}_i^2 \} \tag{3.10}$$

求めた L 及び V をラグランジュの運動方程式である式 (3.11) 代入する．具体的には式 (3.12) のような形になる．

$$\frac{d}{dt} (\frac{\partial L}{\partial \dot{\boldsymbol{q}}}) - \frac{\partial L}{\partial \boldsymbol{q}} + \frac{\partial V}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{Q} \tag{3.11}$$

$$\begin{bmatrix} \frac{d}{dt} (\frac{\partial L}{\partial \dot{\theta}_1}) - \frac{\partial L}{\partial \theta_1} + \frac{\partial V}{\partial \dot{\theta}_1} \\ \vdots \\ \frac{d}{dt} (\frac{\partial L}{\partial \dot{\theta}_7}) - \frac{\partial L}{\partial \theta_7} + \frac{\partial V}{\partial \dot{\theta}_7} \end{bmatrix} = \begin{bmatrix} Q_1 \\ \vdots \\ Q_1 \end{bmatrix} \tag{3.12}$$

この式において，左辺の各成分第 1 項には加速度行列 $\ddot{q} = [\ddot{\theta}_1 \cdots \ddot{\theta}_1]$ の成分が含まれており，これを解く形に変形する．まず，$\ddot{q}$ 成分の係数で構成される行列である，慣性項（システム）行列 M を導く．M_{ni} を式 (3.12) 左辺の第 n 成分における $\ddot{\theta}_i$ の係数とすると，次のように表現できる．

$$M = \begin{bmatrix} M_{11} & \cdots & M_{17} \\ \vdots & \ddots & \vdots \\ M_{71} & \cdots & M_{77} \end{bmatrix} \tag{3.13}$$

上式により，式 (3.12) を次のように表現できる．

$$M\ddot{q} + H_q = Q \tag{3.14}$$

$$\ddot{q} = M^{-1}(Q - H_q) \tag{3.15}$$

ここで H_q は式 (3.12) の左辺各要素の $\ddot{q}$ 成分を含まない項をまとめた行列である．数値計算ではこの運動方程式を利用する．

また，バネによる外力 Q の計算手順について説明する．図 2.11 において，i 関節に対応するベースギアとそれに接続されたプーリ角度および発生しているトルクを θ_{Bi}，T_{Bi} とし，プーリ径は一律で r_B とする．また VSM 軸とそれに接続されているプーリ角度を θ_V とし，プーリ径は一律で r_V とする．バネ係数を k_i とすれば，ベースギアの回転により生じるトルクは次の式で表される．

$$T_{Bi} = -r_B k_i (r_B \theta_{Bi} + r_V \theta_V) \tag{3.16}$$

上式で求めたベースギアに発生するトルクを各リンクに発生するトルクへと変換する式は，ベースギアとリンク角度の関係を表す式 (2.13)，式 (2.14) より，次のように表される．

$$Q_1 = \frac{Zl^2 - Zs^2}{Zl^2} T_{B1} \tag{3.17}$$

$$Q_i = \frac{Zl^2 - Zs^2}{Zs^2} T_{Bi} \qquad (i > 1) \tag{3.18}$$

3.3.2　モータの運動方程式の導出

導出過程はリンクと同じであるため，式をいくつか示すだけにとどめる．一般化座標及び一般化力，各エネルギは次の通りである．なお，説明の混乱を避けるため，同じ性質を示すパラメータは前項と同じ記号を用いている．

$$q = [\theta_D \quad \theta_{VSM}]^T \tag{3.19}$$

$$Q = [Q_D \quad Q_{VSM}]^T \tag{3.20}$$

$$T = \frac{I}{2}(\dot{\theta}_D^2 + \dot{\theta}_{VSM}^2) \tag{3.21}$$

$$U = 0 \tag{3.22}$$

$$V = \frac{c}{2}(\dot{\theta}_D^2 + \dot{\theta}_{VSM}^2) \tag{3.23}$$

この後の過程は式 (3.11)-(3.15) に示した通りである．尚，2 つのモータは運動方程式の上では互いに干渉しないため，慣性項行列 M は対角行列となる．

3.3.3　把持物の運動方程式の導出

剛体の場合の運動方程式の導出はリンク，モータと同様であるため，こちらも式をいくつか示すだけにとどめる．一般化座標及び一般化力，各エネルギは次の通りである．同じ性質を示すパラメータは前項と同じ記号を用いている．

$$q = [G_x \quad G_y \quad G_\theta]^T \tag{3.24}$$

$$Q = [Q_x \quad Q_y \quad Q_\theta]^T \tag{3.25}$$

$$T = \frac{m}{2}(\dot{G}_x^2 + \dot{G}_y^2) + \frac{I}{2}\dot{G}_\theta^2 \tag{3.26}$$

$$U = mgG_y \tag{3.27}$$

$$V = \frac{d}{2}(\dot{G}_x^2 + \dot{G}_y^2) + \frac{c}{2}\dot{G}_\theta^2 \tag{3.28}$$

把持物に関しても，一般化座標の 3 つの要素は互いに干渉しないため，慣性項行列 M は対角行列である．

3.3.4　柔軟な把持物の運動方程式の導出

続いて弾性物体の場合について説明する．運動方程式の導出には非線形有限要素法を用いる．ここでいう非線形とは形状的非線形のことであり，物体の変形に伴い，剛性マトリクスが時間経過と共に変化することを指す．

$$M\ddot{U} + C\dot{U} + KU = F \tag{3.29}$$

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & 0.5-\nu \end{bmatrix} \tag{3.30}$$

$$K = S \cdot B^T \cdot D^T \cdot B \tag{3.31}$$

$$M = \rho \cdot N^T \cdot N \tag{3.32}$$

$$C = \alpha M + \beta K \tag{3.33}$$

上式で表現され，U は節点座標，$M \cdot C \cdot K$ はそれぞれ質量マトリクス・散逸マトリクス，剛性マトリクスである．F は各節点が受けるXおよびY方向の外力である．これら3つのマトリクスは要素ごとに計算された後に合成される．まず各要素の節点座標を基に要素内の物理量を表す形状関数を定義し，そのパラメータによりNマトリクス（N）を計算する．次にNマトリクスからBマトリクスを導く．これはひずみと変位を結びつけるマトリクスである．続いて，物体特性からDマトリクス（D）を計算する．これは応力とひずみを結びつけるマトリクスである．剛性マトリクス，質量マトリクスはこれらの値を利用して計算される．

減衰マトリクスである C は，計算の安定化に優れたレイリー減衰により求める．レイリー減衰は質量マトリクス・剛性マトリクスの固有値から計算されるパラメータを使用することとする．なお，減衰率の指定が必要であり，0.9 とした．

3.4　モデル同士の接触処理

3.4.1　接触判定

物体とリンクの接触判定について説明する．図 3.2 はその過程を表す．リンクを構成する辺の基準点 O_i を原点とし，その辺の長手方向が X 軸となるようにモデルを回転させ，回転後の物体座標 G' により接触の有無を評価する．以下の 2 式を

満たす場合は接触状態にあると判断する．尚，l_i は辺の長さ，r は物体半径を意味する．

$$0 < G'_x < l_i \tag{3.34}$$

$$G'_y < r \tag{3.35}$$

式 (3.34) は，接触位置がリンクの長さ以内に収まっているかを評価する．式 (3.35) は，物体とリンクの距離が接触する距離であるかを評価する．この評価を各辺につき行う．

先端リンクの頂点が物体と接触した場合を，図 3.3 に示す．この場合は，次の 1 式を満たす場合に接触状態にあると判断する．尚，r_t はリンク先端の丸み半径を意味する．

$$r_t + r < l_x \tag{3.36}$$

物体が多角形である場合，物体を構成する各頂点に対して同様の評価を行う．この場合，r は各頂点の丸み半径を意味する．

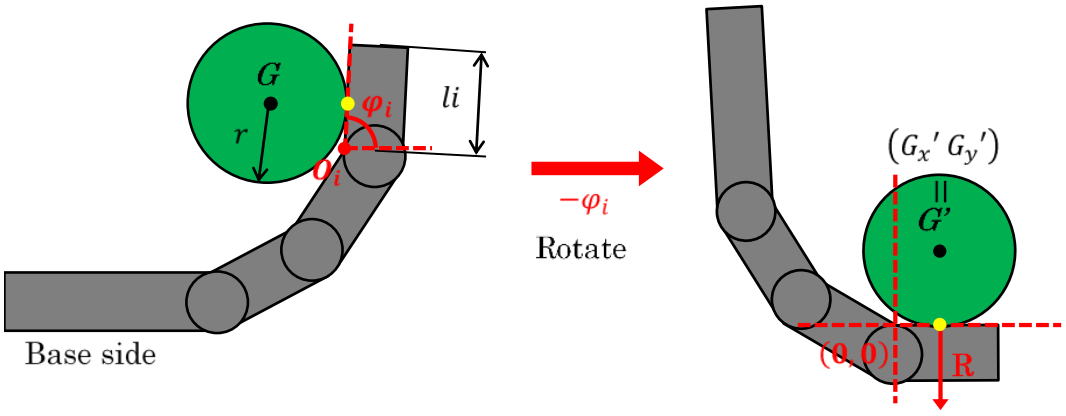


図 3.2　接触処理

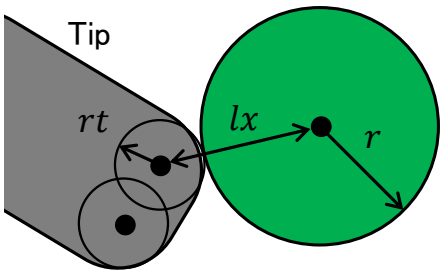


図 3.3　指先の接触処理

また，多角形の物体は構成要素に辺を持つことになり，辺がリンク先端の頂点と接触することが想定される．その場合は図 3.2 に示した接触処理と同様に考える．まず物体を構成する辺の基準点を中心に，辺の長手方向が X 軸となるようモデルを回転する．その際のリンク先端の丸み半径の中心座標により判断する．それは式 (3.34)-(3.36) と同じように，接触位置が辺の長さ以内に収まっているか，物体とリンクの距離が接触する距離であるかを評価する．これらの手法を用いることで，直線と円より構成させる物体形状に対して接触判定を行うことが可能である．

3.4.2 接触反力の反映

接触判定により接触していると判断される場合の反力については，ペナルティ法（Penalty method）を用いる．ペナルティ法とは物体を弾性体とみなす手法であり，物体への貫入を許容して，反力 R はその量に比例した力となる．以下に式を示す．

$$|R| = kd \tag{3.37}$$

d は物体への貫入量であり，長さである．これは接触判定で用いたパラメータから計算可能である．k は物体のヤング率であるが，実際の材料定数を用いると数値計算を行うには過大であるため，経験と試行に基づき計算が安定する値を用いる．

R の方向については，辺と頂点の接触である場合は辺と直交するベクトルであり，頂点と頂点の接触である場合は 2 つの座標を結ぶベクトルである．ここで求めた接触反力は，既に示した運動方程式における一般化力 Q に含める．Q は各リンクへのトルクを要素に持つ行列であるため，接触反力 R から各リンクに発生するトルクを計算する必要がある．次に示すように，外積により求める．

$$Q_i += P_{ci} \times R \quad (1 \leqq i \leqq i_c) \tag{3.38}$$

$$Q_i += 0 \quad (ic < i) \tag{3.39}$$

P_{ci} は J_i（i 番目の関節座標）から接触点までのベクトルであり，i_c は接触したリンク番号である．この計算を各接触点につき行う．$Q_i +=$ となっているように，接触以外の外力が働いている場合や接触点が複数ある場合は，加算していく必要があることに注意しなければならない．

物体についても同様に，運動方程式における一般化力に含める．

3.5　数値計算の手順

数値積分法には 4 次のルンゲクッタ法（Runge-Kutta method）を用いた．以下に式を示す．

$$k1 = f([q_i \quad \dot{q}_i]^T) \tag{3.40}$$

$$k2 = f([q_i \quad \dot{q}_i]^T + k1/2) \tag{3.41}$$

$$k3 = f([q_i \quad \dot{q}_i]^T + k2/2) \tag{3.42}$$

$$k4 = f([q_i \quad \dot{q}_i]^T + k3) \tag{3.43}$$

$$[q_{i+1} \quad \dot{q}_{i+1}]^T = \Delta t(k1 + k2 + k3 + k4)/6 \tag{3.44}$$

Δt は刻み時間である．$f(x)$ は関数であり，一般化座標及び速度（$[q_i \quad \dot{q}_i]^T$）を引数とし，速度と加速度（$[\dot{q}_i \quad \ddot{q}_i]^T$）を返す．速度に関しては与えられた値をそのまま返し，加速度は運動方程式により得た解を返す．

計算のフローチャートを次に示す．

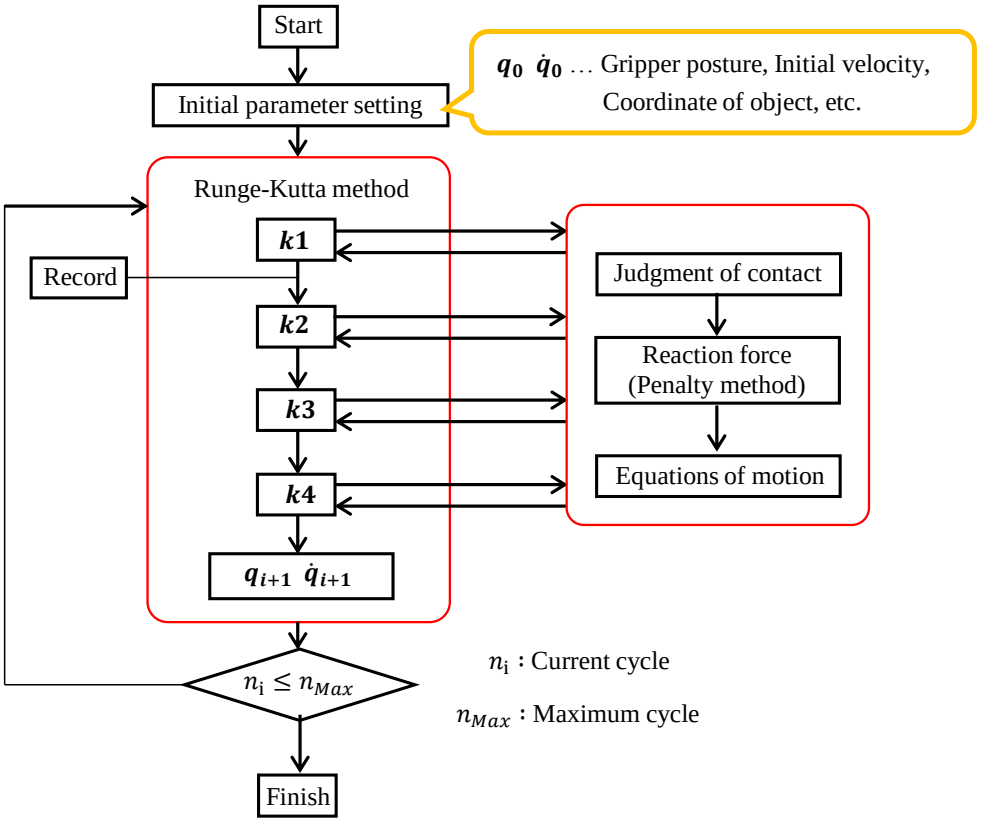

Start
Initial parameter setting
q_0 $\dot{q}_0$ … Gripper posture, Initial velocity, Coordinate of object, etc.
Runge-Kutta method
$k1$
$k2$
$k3$
$k4$
q_{i+1} $\dot{q}_{i+1}$
Record
Judgment of contact
Reaction force (Penalty method)
Equations of motion
$n_i \leq n_{Max}$
n_i : Current cycle
n_{Max} : Maximum cycle
Finish

図 3.4　数値計算の流れ

3.6　基礎動作の検証

最も基本的な動作である屈曲動作および包み込み把持のシミュレーションを行った．シミュレーションに用いたグリッパのパラメータを表 3.1 に示す．

屈曲動作について，ドライビングモータを 50[deg] まで回転させた場合の結果を図 3.5 に示す．水平状態における動作では，リンクに作用する外力が存在しないため，モータの動力がほぼ全てリンクへと伝わっている．リンク自体が持つ慣性力のみが屈曲を妨げる力として働き，このシミュレーションにおけるベースギアの最大回転角度は 0.1[deg] に収まっており，なじみ動作はほぼ起きていないと言える．

包み込み把持のシミュレーションを 3 種類の形状に対して行った結果を，図 3.6 から図 3.8 に示す．またシミュレーションパラメータを表 3.2 に示す．各時刻における把持の様子と，リンクおよびドライビングモータ角度の推移，そしてベースギア角度の推移を示している．なお，図中の矢印は接触力を表している．物体と接触するまではベースギアはほぼ回転しておらず，物体と接触後に回転を始める．そしてそのタイミングで，リンクの角度変位が収束しいることが確認できる．その後，ドライビングモータの目標角度到達に合わせてベースギアの変化も収まる．示した 3 種類の実験結果については，物体形状になじむような把持が達成できている．

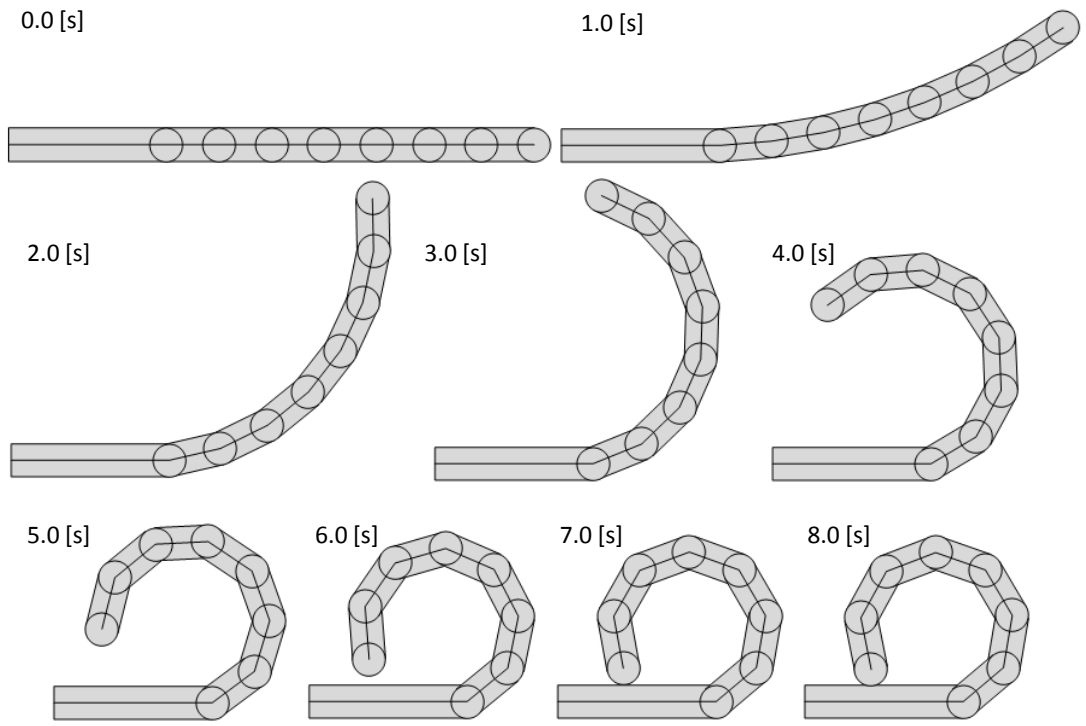


図 3.5　シミュレーション結果：7 関節モデル-基本動作

表 3.1　シミュレーションにおけるグリッパ機構のパラメータ

DOF	7
Length of link	50[mm]
Width of link	30[mm]
Weight of link	100[g]
Moment of inertia	$50 \times 10^{-6}[kgm^2]$
Damping of rotation	0.03
Teeth number of Gears	30 & 20

表 3.2　シミュレーションパラメータ

Simulation time	10[s]
Calculation step / sec.	1200
Weight of Object	100[g]
Driving Motor	0→70[deg]
VSM Motor	20[deg]

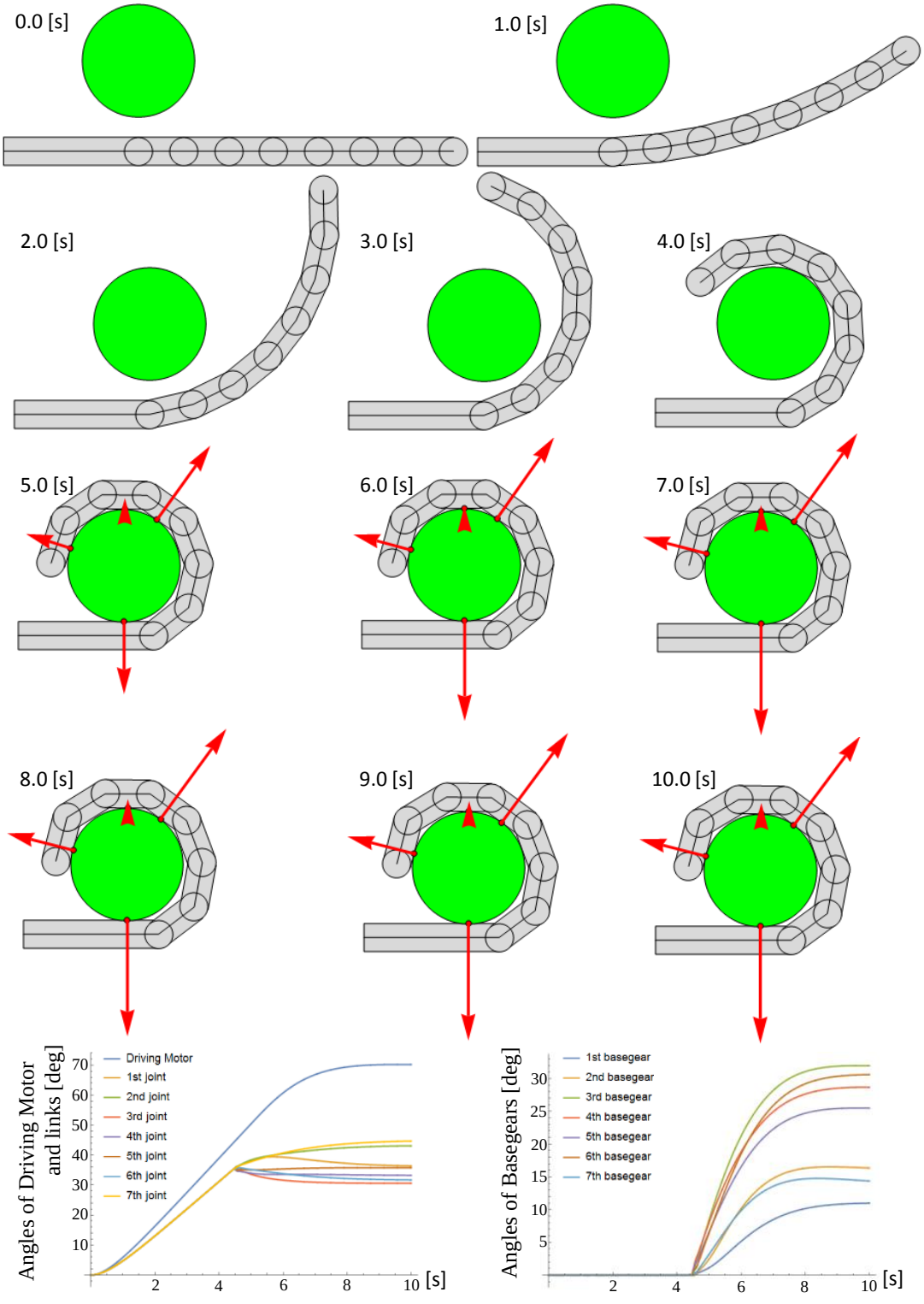


図 3.6　シミュレーション結果：7 関節モデル-円形物体

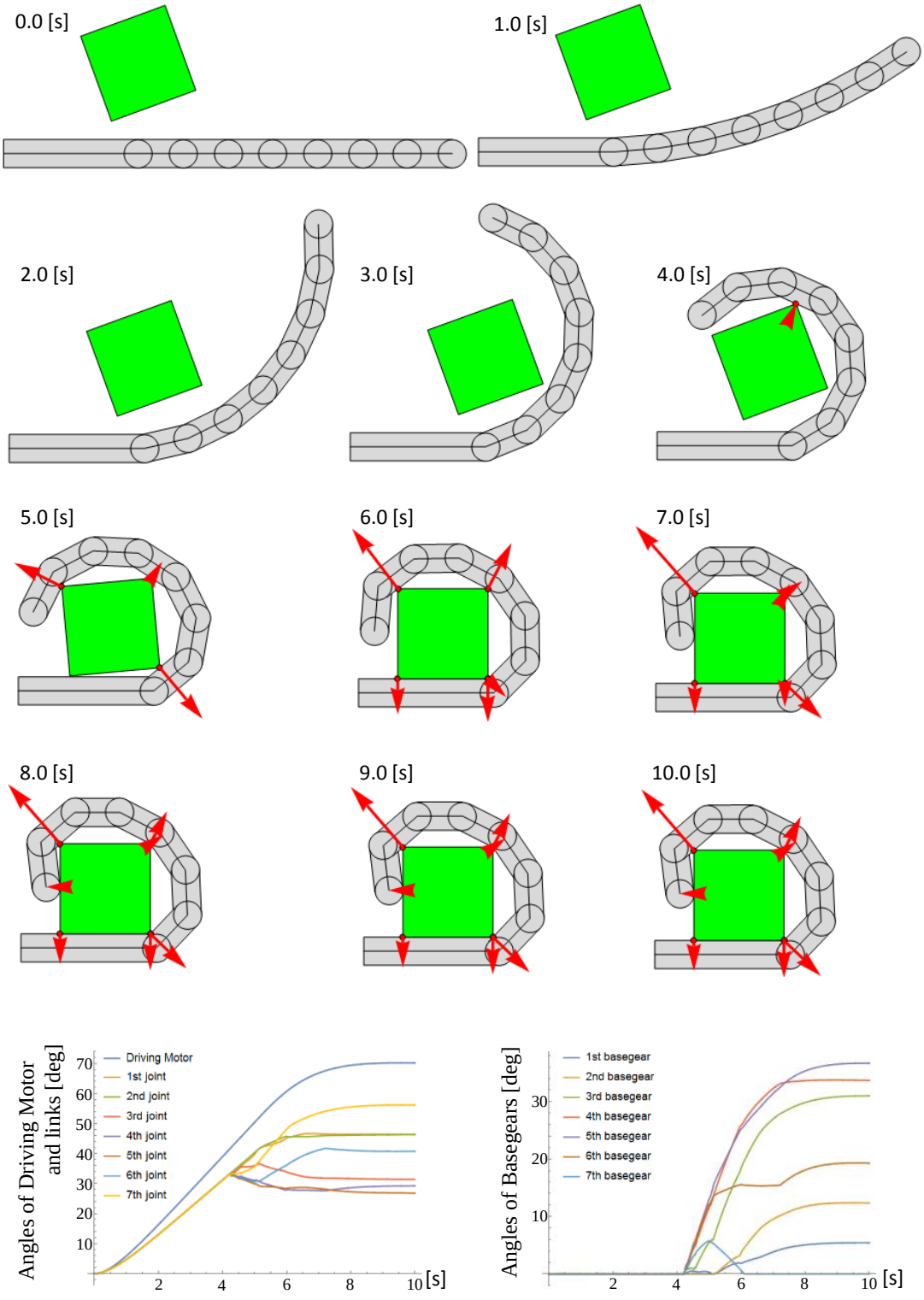


図 3.7　シミュレーション結果：7 関節モデル-正方形物体

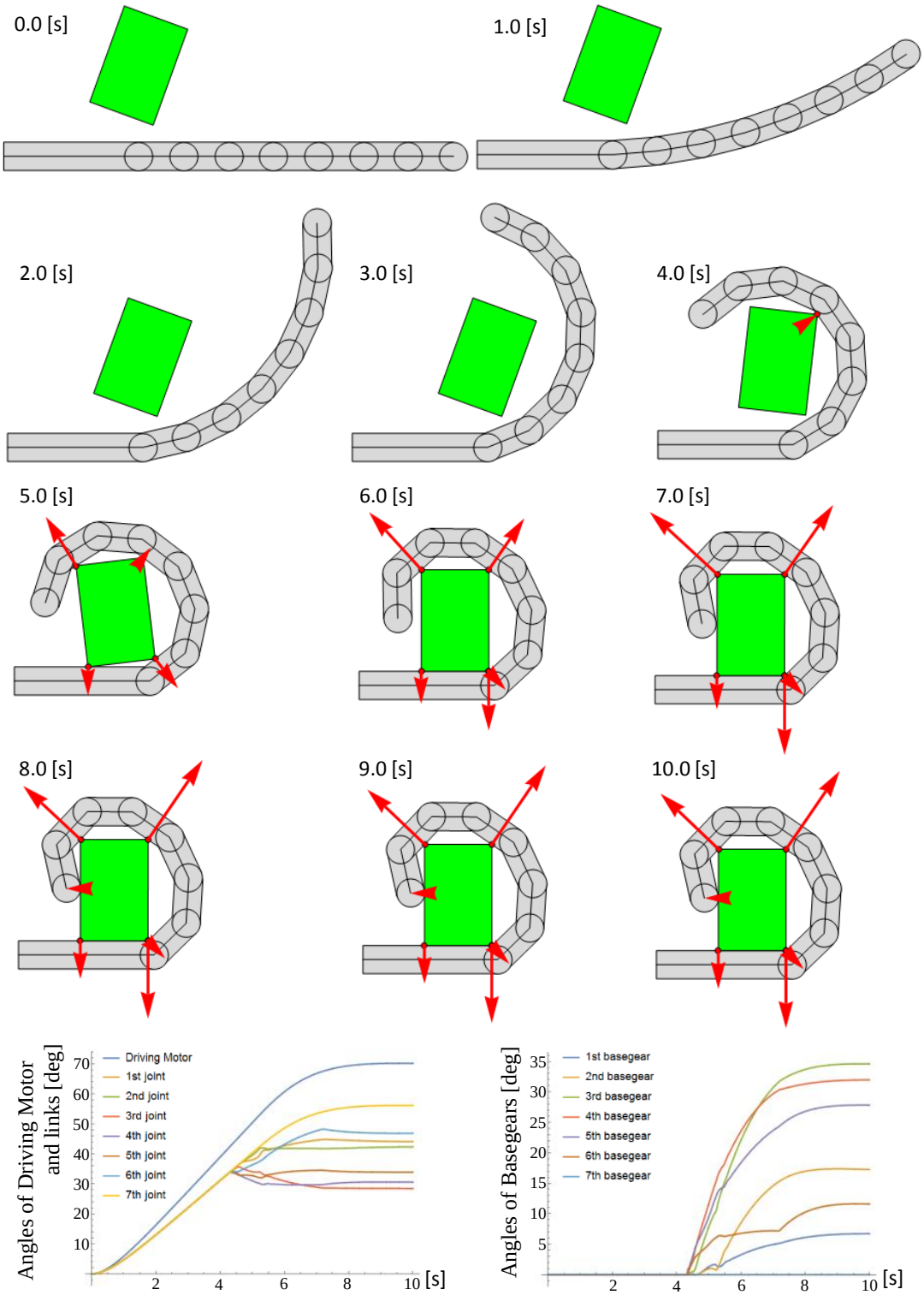


図 3.8　シミュレーション結果：7 関節モデル-長方形物体

3.7　まとめ

本章では，動力学シミュレーション手法の説明を主に行った．各運動要素である，リンク機構，モータ，把持物の運動方程式の導出過程を示し，また把持物とリンクの接触という要素同士の干渉について，その扱いを示した．

運動方程式の導出にはラグランジュ法を主に用い，弾性物体に関しては非線形有限要素法を用いる．接触処理には，判定を行う要素間に発生している貫入量に比例する反力が生じる，ペナルティ法を用いる．

これらの手法を元に，前章で示したグリッピングハンド機構による基礎的な動作の評価を行った．評価は単純な屈曲動作と，いくつかの形状に対する包み込み把持である．シミュレーション結果から，非常に単純な制御で物体形状になじむ包み込み把持が達成できることが確認された．

第 4 章

実験およびシミュレーションによる評価

4.1　はじめに

本章では，これまでに説明した機構を基に設計したハンドの概要についてまず述べる．本機構は，包み込み把持に関しては 1 指でも十分に機能を満たせるものであると見込まれる．そこで，1 指モデルについてまず製作した．その後，ピンチングなど多指モデルならではの動作を目指して，2 または 3 本の指を組み合わせたモデルを製作した．これらの実機について説明した後，ベースギア角度の測定や接触力測定，それらを統合した制御系について説明する．

実験を行うための環境について一通り述べた後で，序論での目的に沿って評価を行う．まずシミュレーションでは既に試みた基本的な屈曲動作および包み込み把持について検証する．その後，剛性可変機構，ベースギアを測定することによる力覚システムについて評価し，関節剛性のパターンと把持力の分布について検討する．最後に，指先機構によるピンチングからの包み込み把持について考察する．

4.2　実機および制御系

4.2.1　1 指モデルの概要

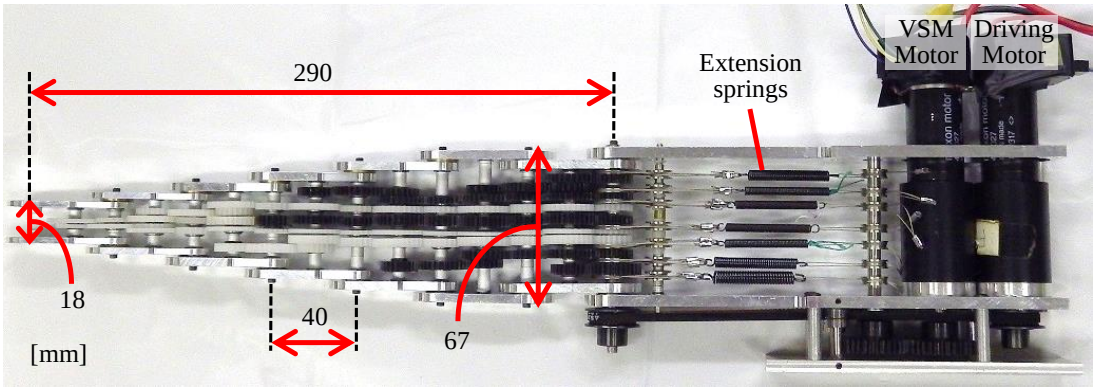
1 本指で運用するモデルとして，7 関節を持つ実機を製作した．表 4.1 に各設計パラメータを，図 4.1 に実機概要を示す．関節間距離は 40[mm]，指部全長は 290[mm]，指部重量は 680[g] である．モータは，ドライビングモータ，VSM モータ共に 20[W] の DC モータを用い，246 分の 1 のギアヘッドを取り付けている．センサは，基本的にはモータ回転角の計測のみに用いられ，分解能 500 のロータリエンコーダをモータに直結させている．

また，各関節剛性に影響を与える引張バネのバネ係数は，実験内容により異なるパターンを用いることもあるが，表 4.1 に示した並びを基本のパターンとする．このバネ係数のパターンは，通常の屈曲動作時にリンク慣性力の影響を受けやすい基部側の関節ほど，より関節剛性が大きくなるようにしたものである．このバネ係数パターンにおける屈曲動作の様子を図 4.1(b) に示している．素早い屈曲動作となるよう制御しているが，全ての関節がほぼ同一角度を保っていることを確認でき，自らの慣性力によるなじみ動作を引き起こさずに屈曲している．

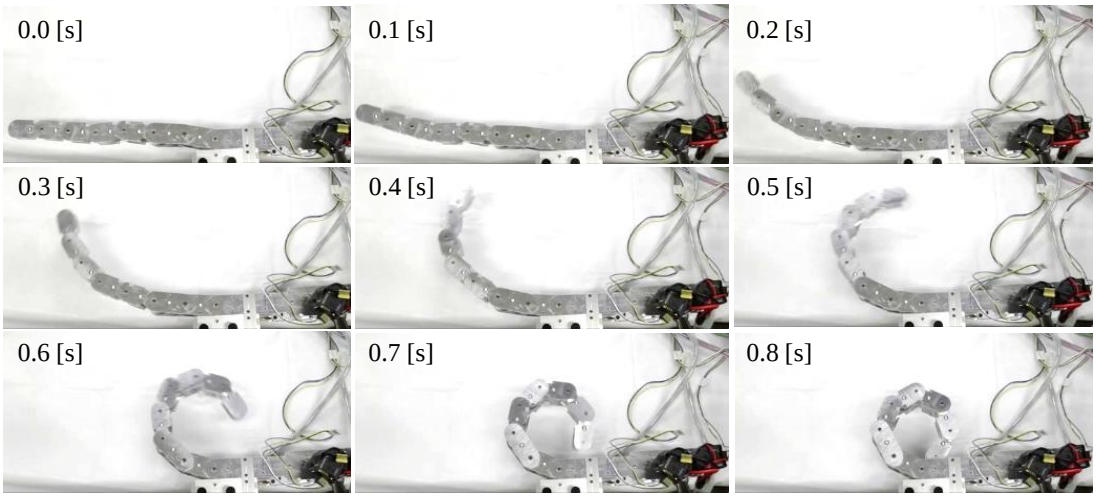
また，図 4.1(c) に VSM モータによる関節剛性を変化させている様子を示しているが，本実験機においてバネの基準伸び量の最大値は，およそ 15[mm] である．

表 4.1　1 指モデルの指機構設計パラメータ

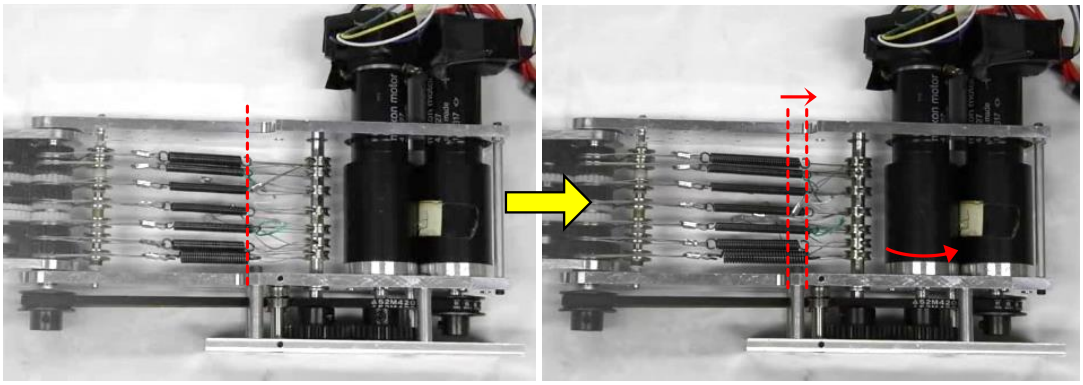
Number of joint	7
Teeth number of Gears	20，30
Length of finger part	290 [mm]
Length of between the joint	40 [mm]
Width of finger part	26 [mm]
Height of finger part	18∼67 [mm]
Weight of finger part	680 [g]
Motor	Maxon RE25×2 (Blushless 20W) Gear head GP32A (246:1)
Encoder	Maxon HEDS5540 (Resolution:500)
Spring modulus	1st : 3.50　2nd : 1.99　3rd : 1.99　4th : 1.50 5th : 1.50　6th : 0.98　7th : 0.98　[N/mm]



(a) 機体側面



(b) 屈曲動作の様子



(c) 関節剛性を調節する様子

図 4.1　1 指モデルの実機概要

4.2.2　多指モデルの概要

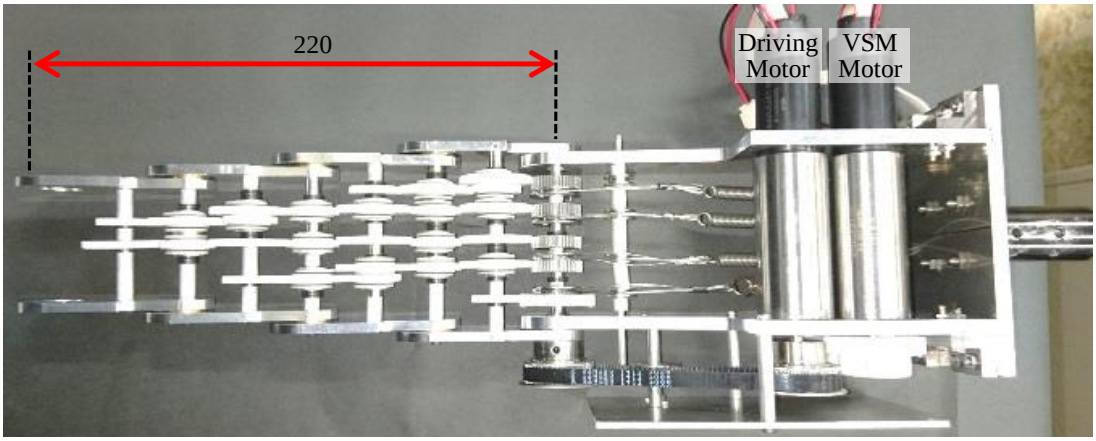
複数本の指で運用するモデルとして，4 関節を持つ実機を製作した．表 4.2 に各設計パラメータを，図 4.2 に実機概要を示す．本論文では，多指モデルとしては主に図 4.2(d) に示すような 2 指モデルについて議論する．その他のモデルとして，図 4.2(e) に示すような 3 指モデルを組むことができるが，簡単な包み込み把持の実験を示すにとどめる．

設計パラメータは，指部全長 220[mm]，指部重量 450[g] である．また 1 指モデルと同じく 20[W] の DC モータを用い，基本的なバネ係数のパターンは，基部側の関節ほど関節剛性が大きくなるものとする．

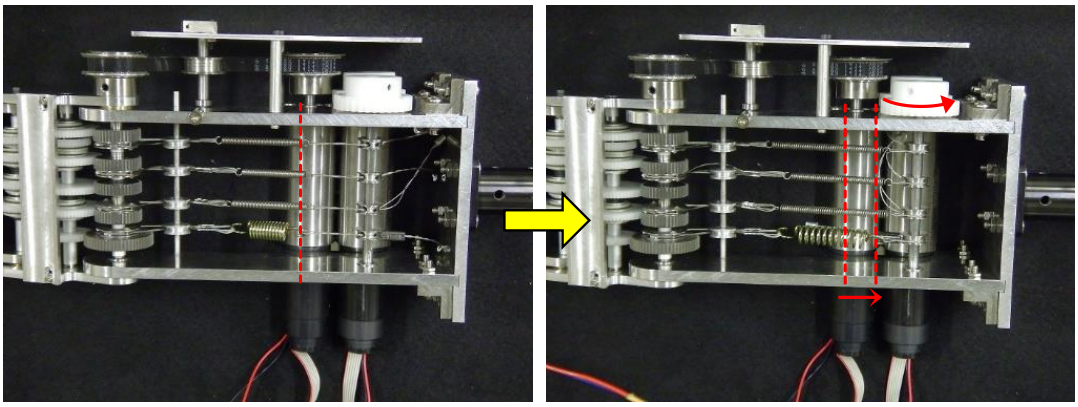
複数の指を組み合わせる場合は，図 4.2(a) における右端が固定板であり，土台となるフレームに接続する．取り付ける際の角度などは手動で調節できる仕組みとなっている．

表 4.2　多指モデルの指機構設計パラメータ

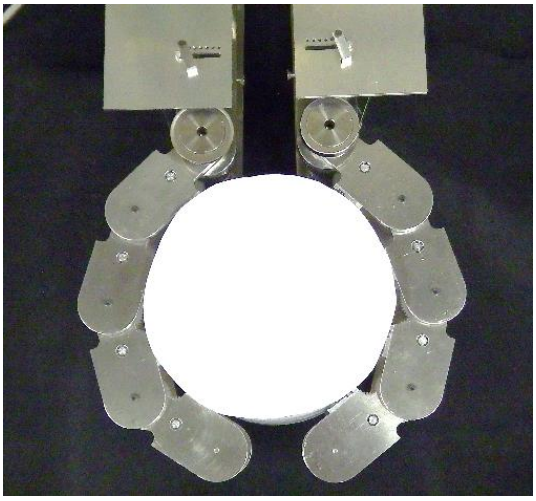
Number of joint	4
Teeth number of Gears	20，30
Length of finger part	220 [mm]
Length of between the joint	50 [mm]
Width of finger part	30 [mm]
Height of finger part	50∼85 [mm]
Weight of finger part	450 [g]
Motor	Faulhaber×2 (Blushless 20W) Gear head (415:1)
Encoder	Faulhaber (Resolution:500)
Spring modulus	1st : 3.50　2nd : 0.90　3rd : 0.50　4th : 0.30　[N/mm]



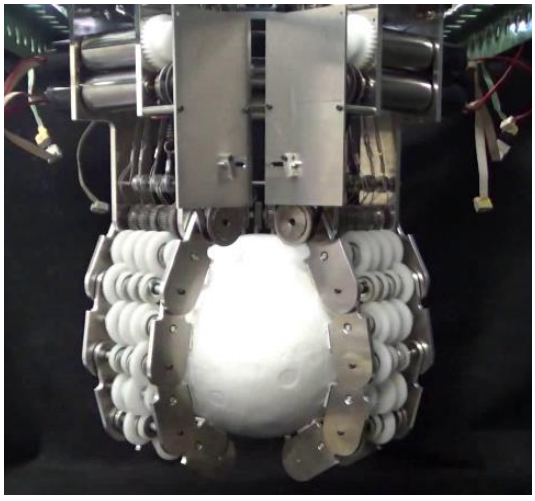
(a) 機体側面



(b)関節剛性を調節する様子



(d) 2指モデルによる包み込み把持



(e) 3指モデルによる包み込み把持

図 4.2　多指モデルの実機概要

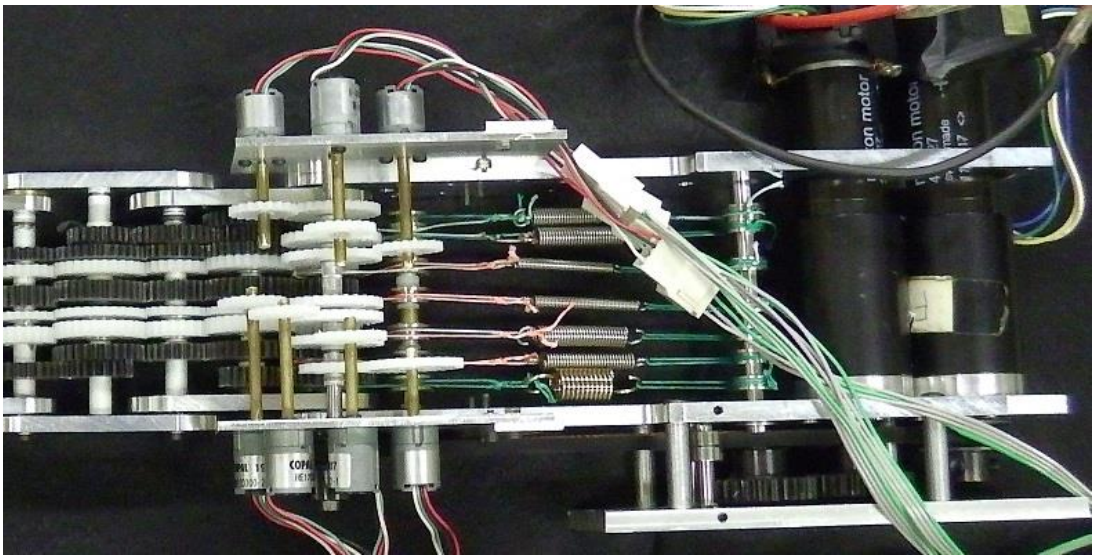
4.2.3　ベースギア角度測定用のエンコーダ

主に力覚システムの実験を行う際に，各関節が引き起こしているなじみ動作を計測するため，ベースギアの角度をロータリエンコーダにより測定する．エンコーダおよびその設置方法を図 4.3 に，仕様を表 4.3 に示す．COPAL 社製の RE12D を用いる．第 1 関節付近にエンコーダ用のフレームを設置し，関節数に等しい数を取り付ける．ベースギアの角度は歯車を経由して測定される．

表 4.3　ロータリエンコーダの仕様：COPAL 社製 RE12D

Output waveform	Square wave
Output phase	A, B
Resolution	300
Maximum response frequency	10kHz
Output signal	4.5V minimum / 0.5V maximum

(a)　ロータリエンコーダ : COPAL RE12D



(b) エンコーダをベースギアへ接続する様子

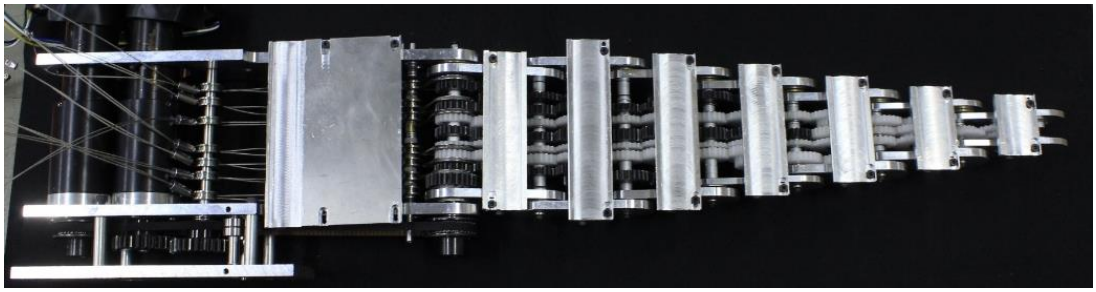
図 4.3　ベースギア角度計測用のロータリエンコーダ

4.2.4　接触力測定用の圧力センサ

把持物との接触力を測定するために，圧力センサを利用する．センサおよび設置方法について図 4.4 に示す．取り付けには，図 4.4(a) に示すようにリンク側面に圧力センサが収まる溝を掘った接触版を設置し，その上に貼り付ける．センサの感圧部には，できるだけ均等に力がかかるように，その形状に沿うように厚さ 1[mm] の樹脂材の円板を貼り付けている．

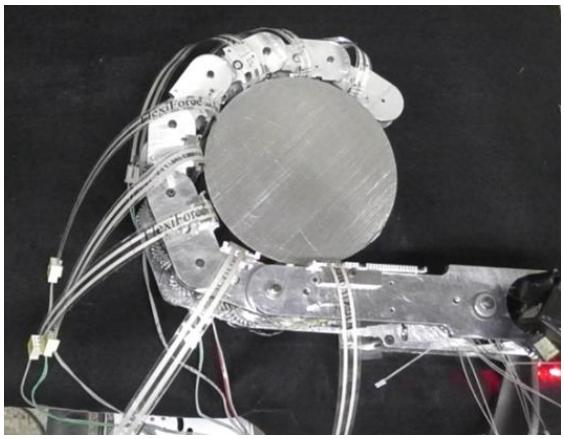
圧力センサは SparkFun 製 FlexiForce A201-1 を用いる．圧力により抵抗値が変化するタイプのセンサで，無負荷時はおよそ 20[MΩ]，最大荷重 (0.453[kg]) 時は 20[kΩ] となる．仕様の詳細を表 4.4 に示す．また，抵抗値の変化を電圧値で読み取る回路を用い，それを図 4.5 に示す．これにより，およそ 0 ～ 400[g] の荷重を 0 ～ 10[V] の出力として計測することが出来る．

(a) 圧力センサ



(b) リンクへの接触板の取り付け

(c)接触板への圧力センサの取り付け

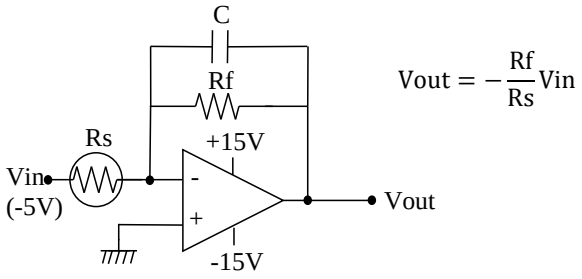


(d) 接触力測定の様子

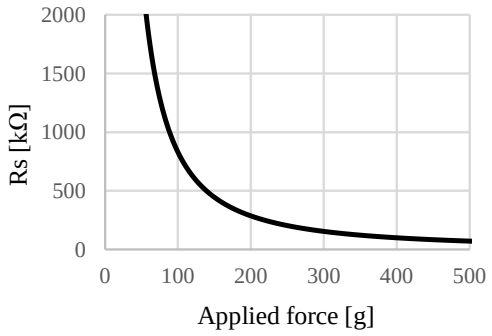
図 4.4　接触力測定用の圧力センサ

表 4.4　圧力センサの仕様：SparkFun 製 FlexiForce A201-1

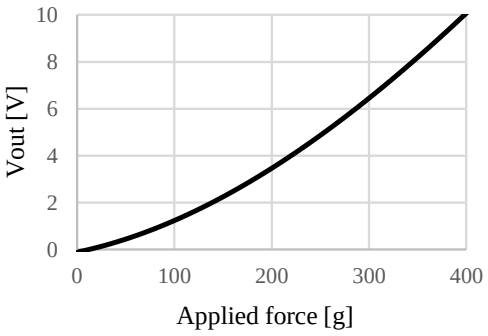
Length	184[mm]
Width	14[mm]
Thickness	0.08[mm]
Diameter of measuring surface	9[mm]
Connecting pin	3pin(Middle pin is open)
Linearity error	±5%
Response time	5[μs]
Resistance value with no-load	20[MΩ]
Resistance value with maximum load	20[kΩ]
Resistance value with overload	5[kΩ]



(a) 圧力測定用回路



(b) 荷重に伴うセンサ抵抗値



(c) 荷重に伴う出力電圧

図 4.5　接触力測定用回路

4.2.5　制御系

ここまで取り上げてきた，モータ，ロータリエンコーダ，圧力センサなどを使用するための制御系について説明する．フローチャートを図 4.6 に示す．制御プログラムは，PC の Windows 上で動作するプログラムを作成した．制御プログラムを基点に考えると，大きな流れとして，モータへの出力，ロータリエンコーダからの入力，圧力センサからの入力の 3 つに分けられる．それぞれの流れについて，図 4.6 に沿って説明する．

まず，モータへの出力について説明する．USB-シリアル変換モジュールである FTDI 社の FT232RL(秋月電子版) を USB 接続することにより，SPI 通信が可能となる．SPI 信号を Linear Technology 社のシリアル D/A コンバータ LTC1660CN に入力することで，任意のアナログ電圧へと変換される．ここで，PIC マイコンである PIC16F1503 に，アナログ電圧を PWM へ変換するプログラムを組んでおく．モータへ直接出力するモータドライバ IC である，東芝セミコンダクター社の TB6643KQ は，PWM 駆動方式に対応しており，デューティ比によりモータの回転速度を制御することが出来る．これらの流れによりモータの駆動は達成される．

次に，エンコーダからの入力について説明する．プログラム上でパルスをカウントするために，PC に Interface 社のカウンタボードである PCI-6205C を搭載している．このカウンタボードは差動入力である必要があるため，エンコーダからの信号をラインドライバ IC などを利用して変換しなくてはならない．本カウンタボードでは，ノット回路を用いることでラインドライバ IC の代替とすることが出来る．図 4.6 中にも示しているが，エンコーダから出力される A 相，B 相の信号 (High/Low) を A+，B+ にそのまま出力し，反転した信号を A-，B-へ出力すれば良い．カウンタボードはこれら 4 つの信号を受け取ることで，正確にパルスをカウント可能である．

最後に，圧力センサからの入力について説明する．既に説明した圧力センサを用いた回路により，測定値は電圧へと変換されている．PC に A/D 変換ボードである Interface 社の PCI-3133 を搭載しており，これを利用することで，プログラム上で電圧値を読み取ることができる．

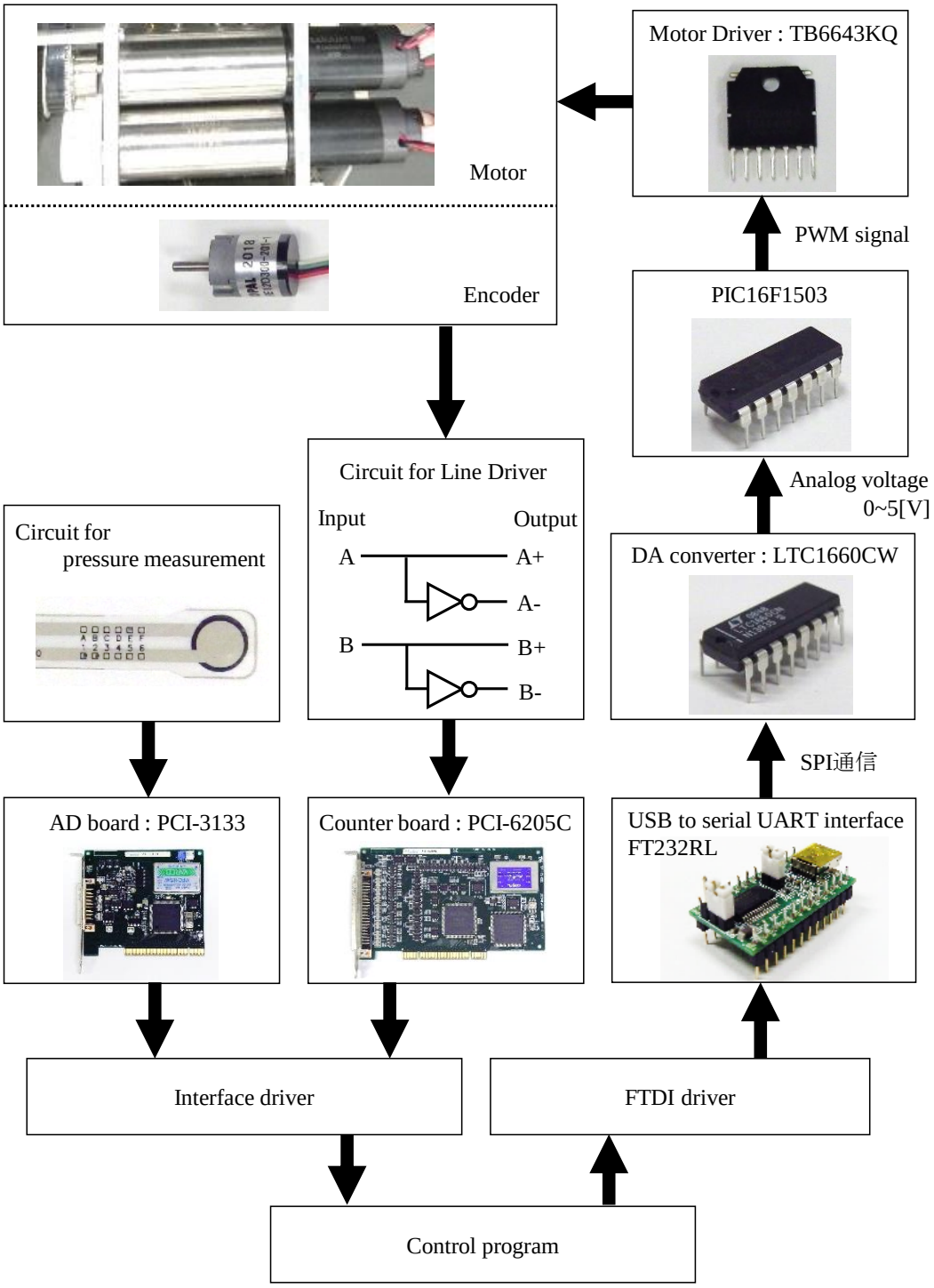


図 4.6　制御系のフローチャート

4.3 物体形状になじむ包み込み把持

製作した実験機を用いて，実際に物体形状へなじむ包み込み把持を試みた．いくつかの対象物に対し同一の制御で把持を試みる．

初めに 1 指モデルについて報告する．VSM モータは 20[deg] を維持するよう終始一定で駆動させ，ドライビングモータは 70[deg] まで駆動させる．この際，モータの制御には PD 制御を用いるが，最大電圧を指定することで最大各速度を制限している．把持物のパラメータを表 4.5 に示し，それぞれの把持物に対する実験結果を図 4.7 から図 4.12 に示す．

表 4.5　把持物パラメータ

(a)	12 [g]	ϕ90 [mm]	plastic
(b)	50 [g]	70 × 70 [mm]	plastic
(c)	145 [g]	140 × 60 [mm]	aluminum
(d)	30 [g]	120 × 90 [mm]	plastic
(e)	20 [g]	ϕ85 [mm]	sponge
(f)	20 [g]	90 × 90 [mm]	sponge

実験結果では，動力学シミュレーションと同じく物体形状になじむ把持を理論通りに行えることが確認できた．図 4.7 では (a) の円形物体を把持している．通常の屈曲動作でも指部は円形に近い姿勢をとるため，スムーズになじむことができている．図 4.8 から図 4.10 では正方形または長方形の物体を把持している．それぞれ物体形状に合わせて姿勢が変化しているが，特に (c) の物体を把持している図 4.9 では，接触後に第 1 関節の屈曲が停止し，それにより基部付近の関節が直線に沿って把持が出来ていると言える．また図 4.11，図 4.12 はそれぞれ柔らかい物体を把持した場合の結果である．円形物体の場合はもちろん，正方形の物体に対しても，極端な変形を生じさせずに把持を行えている．この実験により，物体形状になじむ包み込み把持が非常に簡単な制御で実現できていると言える．

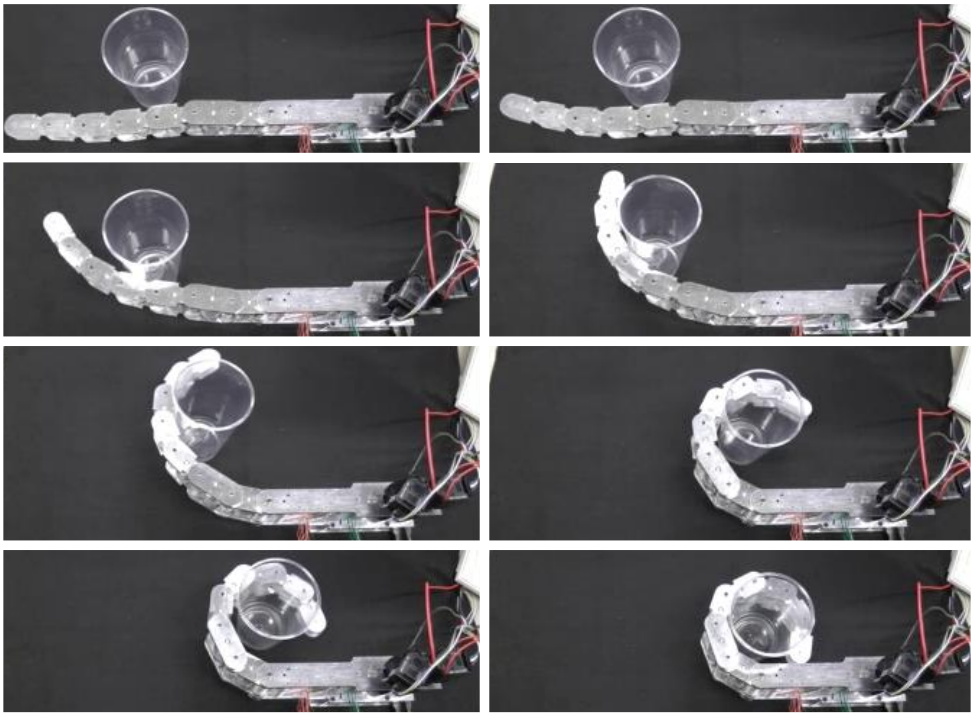


図 4.7　包み込み把持：1 指モデル - 把持物パラメータ (a)

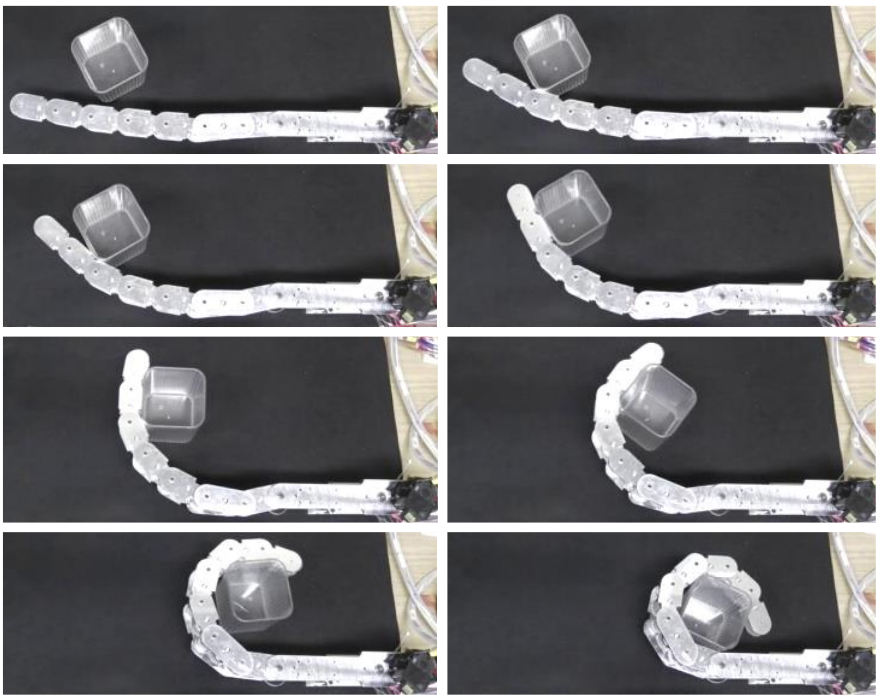


図 4.8　包み込み把持：1 指モデル - 把持物パラメータ (b)

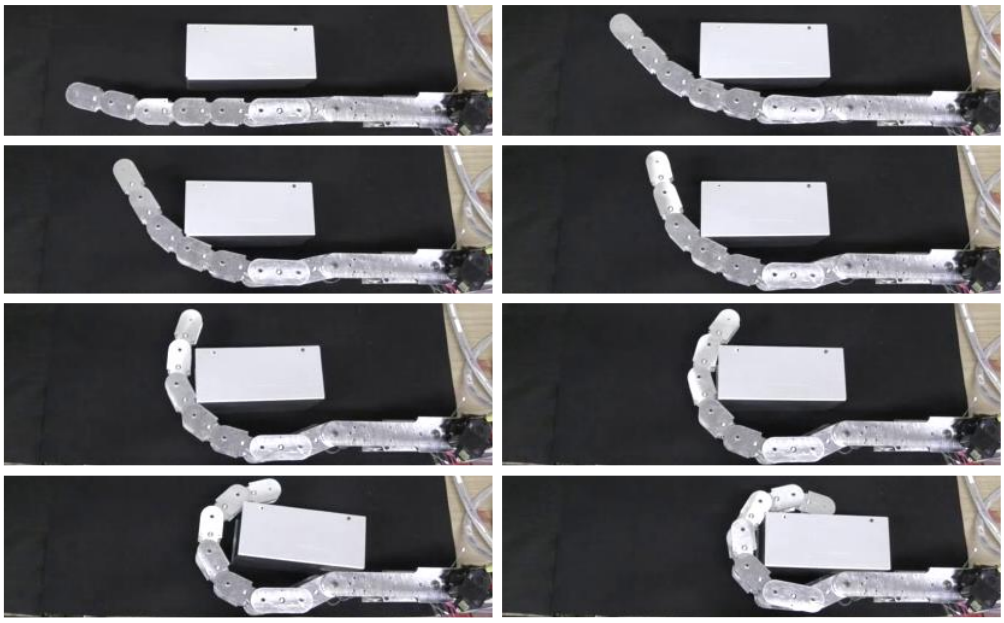


図 4.9　包み込み把持：1 指モデル - 把持物パラメータ (c)

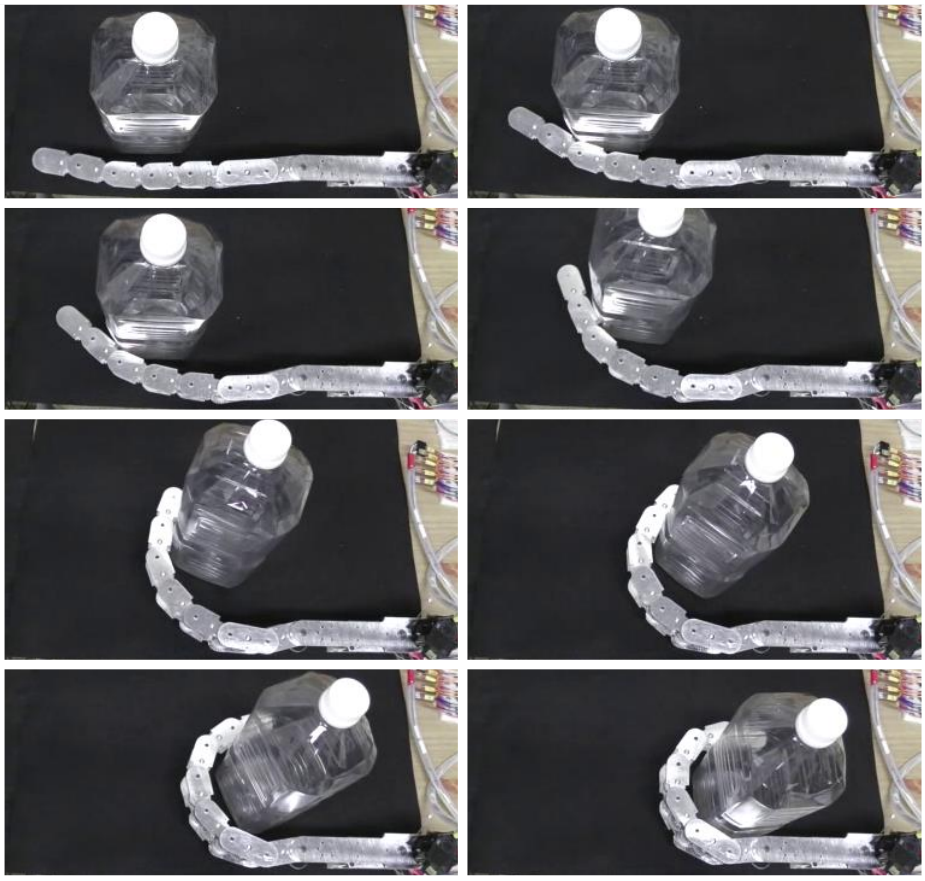


図 4.10　包み込み把持：1 指モデル - 把持物パラメータ (d)

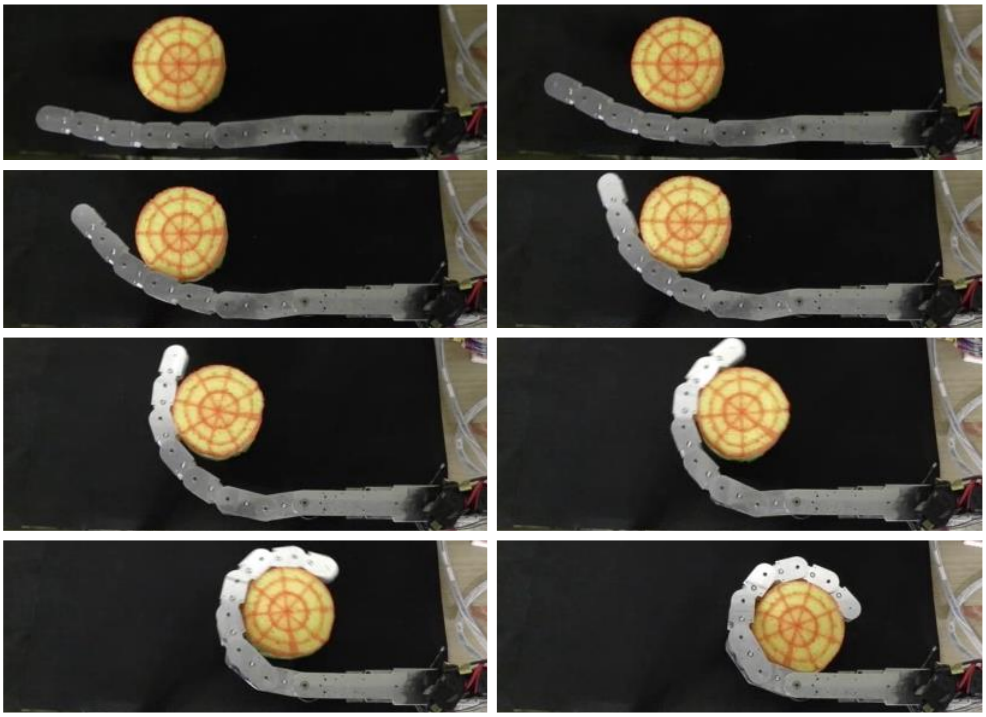


図 4.11　包み込み把持：1 指モデル - 把持物パラメータ (e)

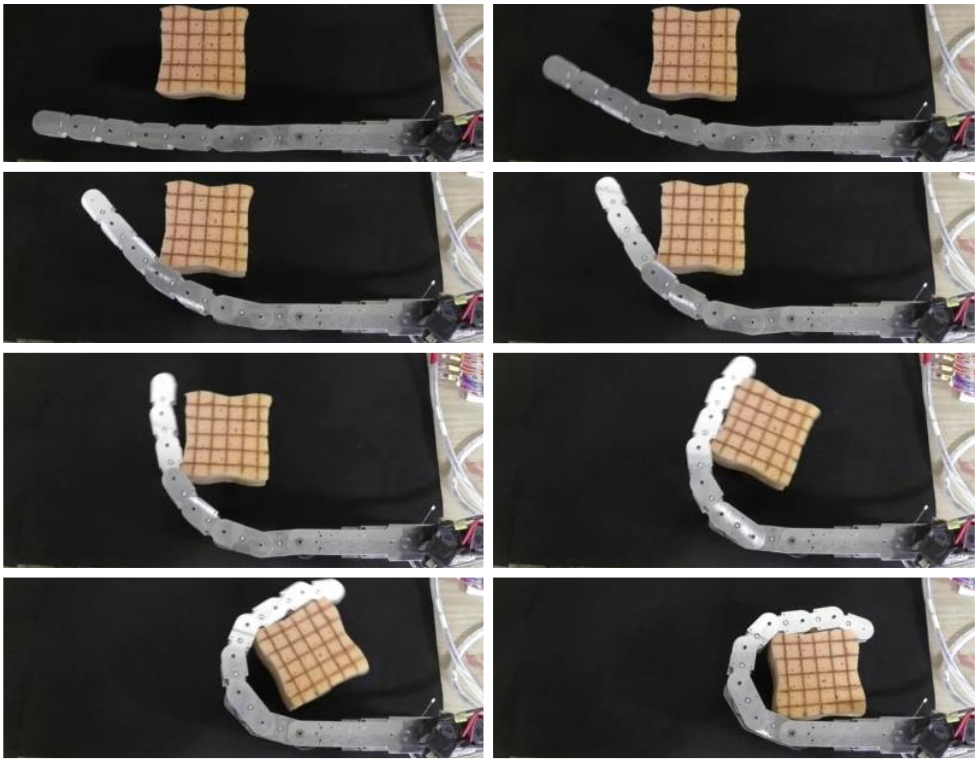


図 4.12　包み込み把持：1 指モデル - 把持物パラメータ (f)

次に 2 指モデルについて報告する．いくつかの形状に対して把持を行い，その結果を図 4.13 から図 4.16 に示す．VSM モータは 20[deg] を維持するよう終始一定で駆動させ，ドライビングモータを適宜制御する．モータ制御には PD 制御を用いている．また把持物のパラメータを表 4.6 に示す．

表 4.6　把持物パラメータ

(g)	35 [g]	ϕ150 [mm]	styrofoam
(h)	75 [g]	230 × 50 [mm]	carton
(i)	300 [g]	ϕ80 [mm]	mug

図 4.13 は 2 指により球状の物体を把持した様子である．1 指モデルと同様，円形状の物体に対しては順調になじむことが確認できる．図 4.14 は同じく 2 指により長方形の物体を把持した様子である．これら 2 種類の把持は，ドライビングモータに左右の指に対して同一の目標角度，60[deg] を与えているのみである．そのような制御でも，大きく形状が異なる物体に対して，それぞれの形状になじむ把持を達成している．

図 4.15 は 2 指で丁度包み込むには小さいサイズの物体を把持する様子である．先に片方の指で手繰り寄せ，後からもう一方の指で逆の指ごと把持している．この把持は操作者が目視により状態を確認しながら行ったものである．1 指でもある程度の手繰り寄せや把持が行える構造だからこそ，このような把持が可能であるといえる．

図 4.16 は 3 指による動作であり，摘み上げから包み込み把持へ移行する様子を示している．こちらの制御も操作者が目視により確認しながら行っているものである．リンクと把持物の間に大きな摩擦が働かなければ，単純な目標角度を与えるだけの制御でも同様のことが可能である．

これらの結果より，本機構は物体形状に沿った柔らかい把持が達成されていることを確認したが，ここまで示した結果はいずれも VSM モータを 20[deg] としている．関節剛性を変えた場合の効果も確認する必要がある．そこで次節では剛性可変機構について検証する．

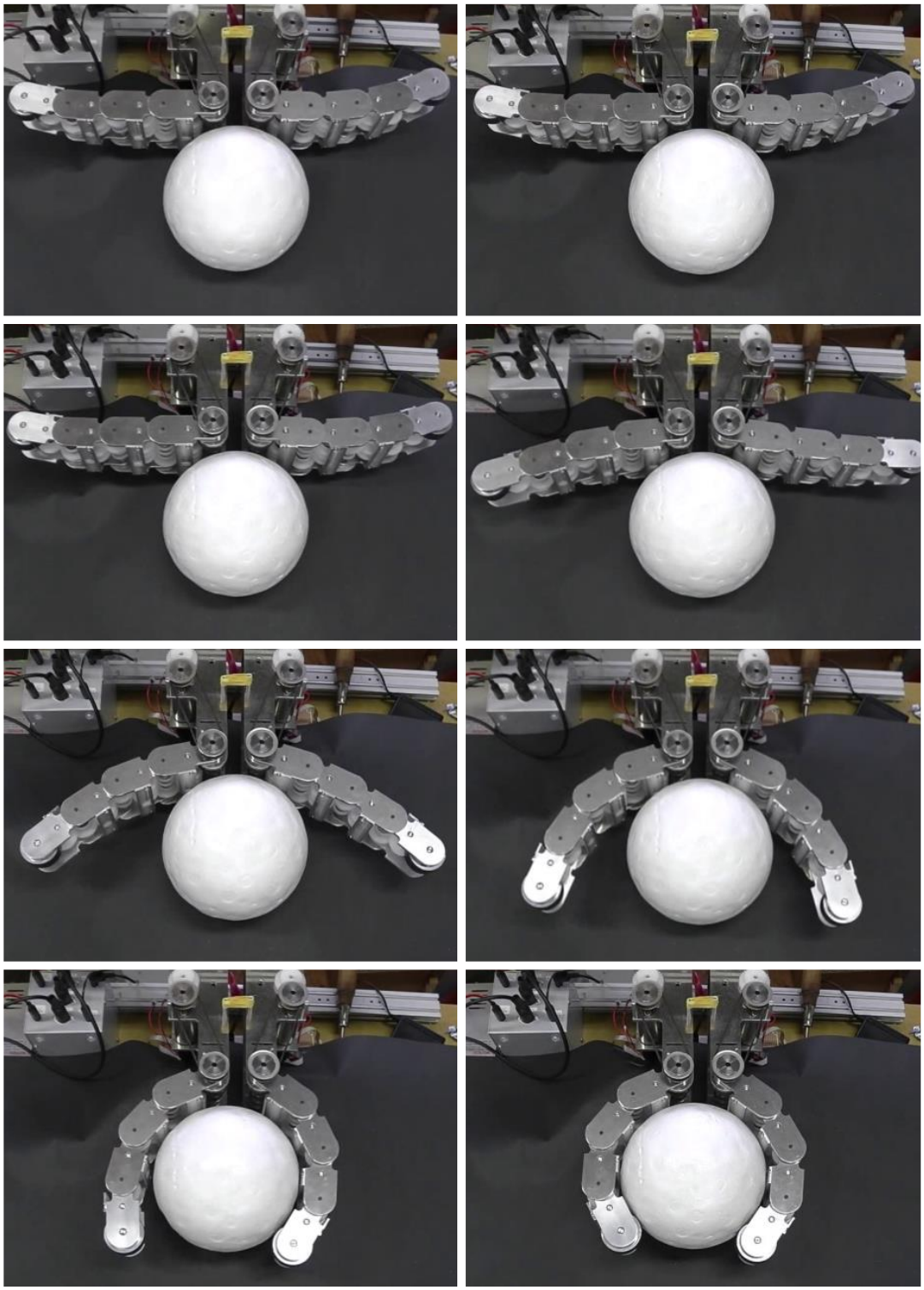


図 4.13　包み込み把持：2 指モデル - 把持物パラメータ (g)

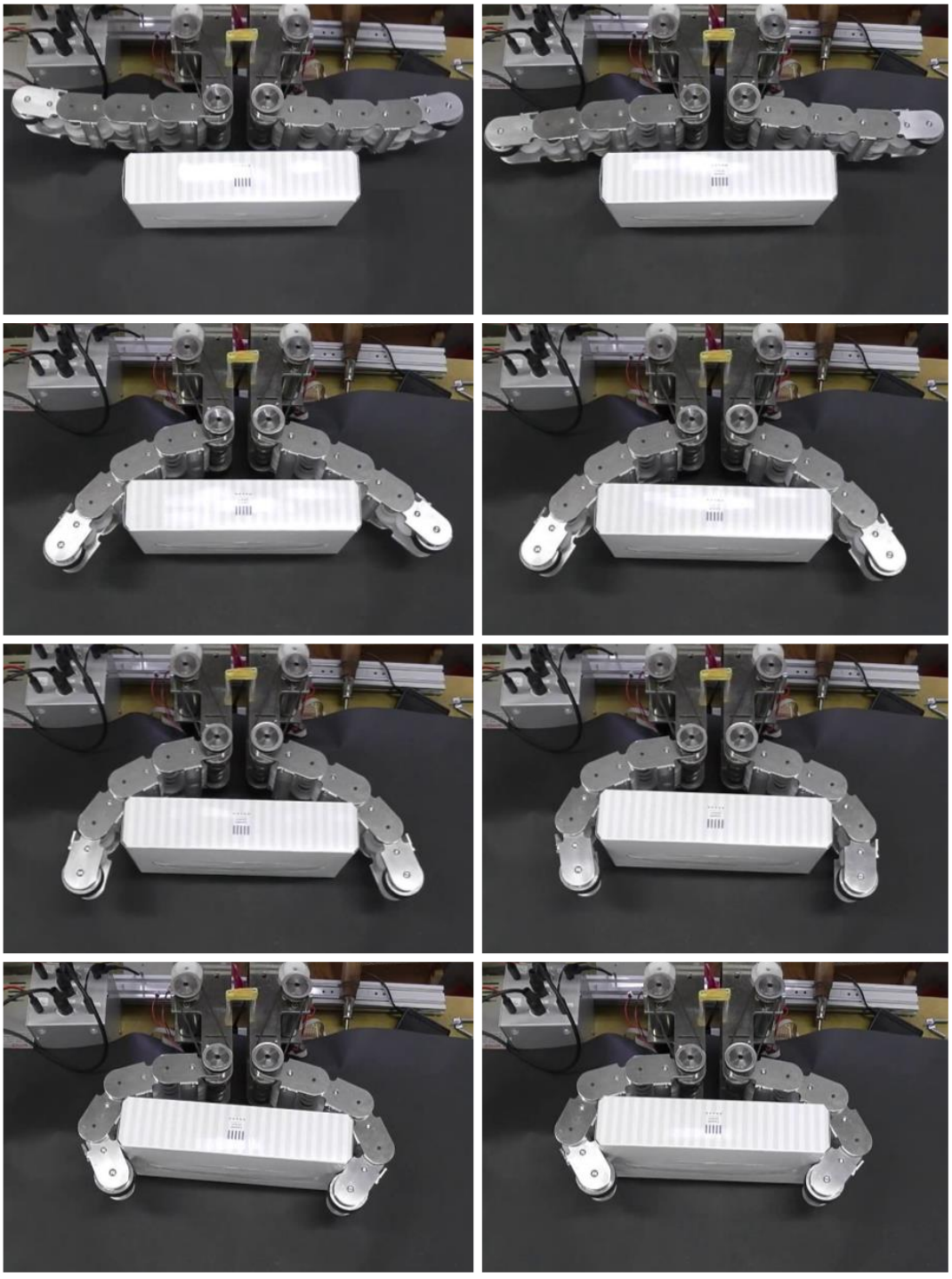


図 4.14　包み込み把持：2 指モデル - 把持物パラメータ (h)

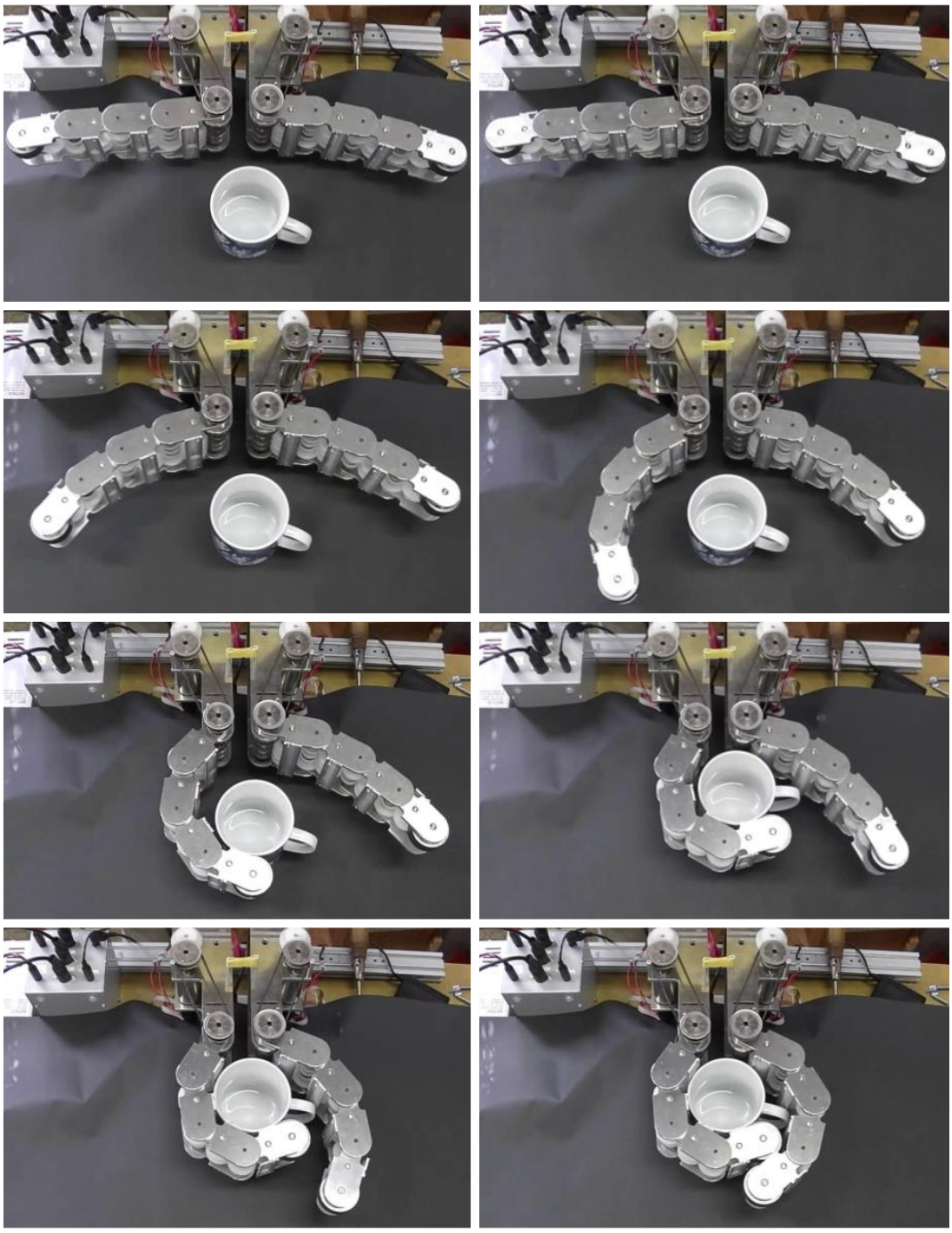


図 4.15　包み込み把持：2 指モデル - 把持物パラメータ (i)

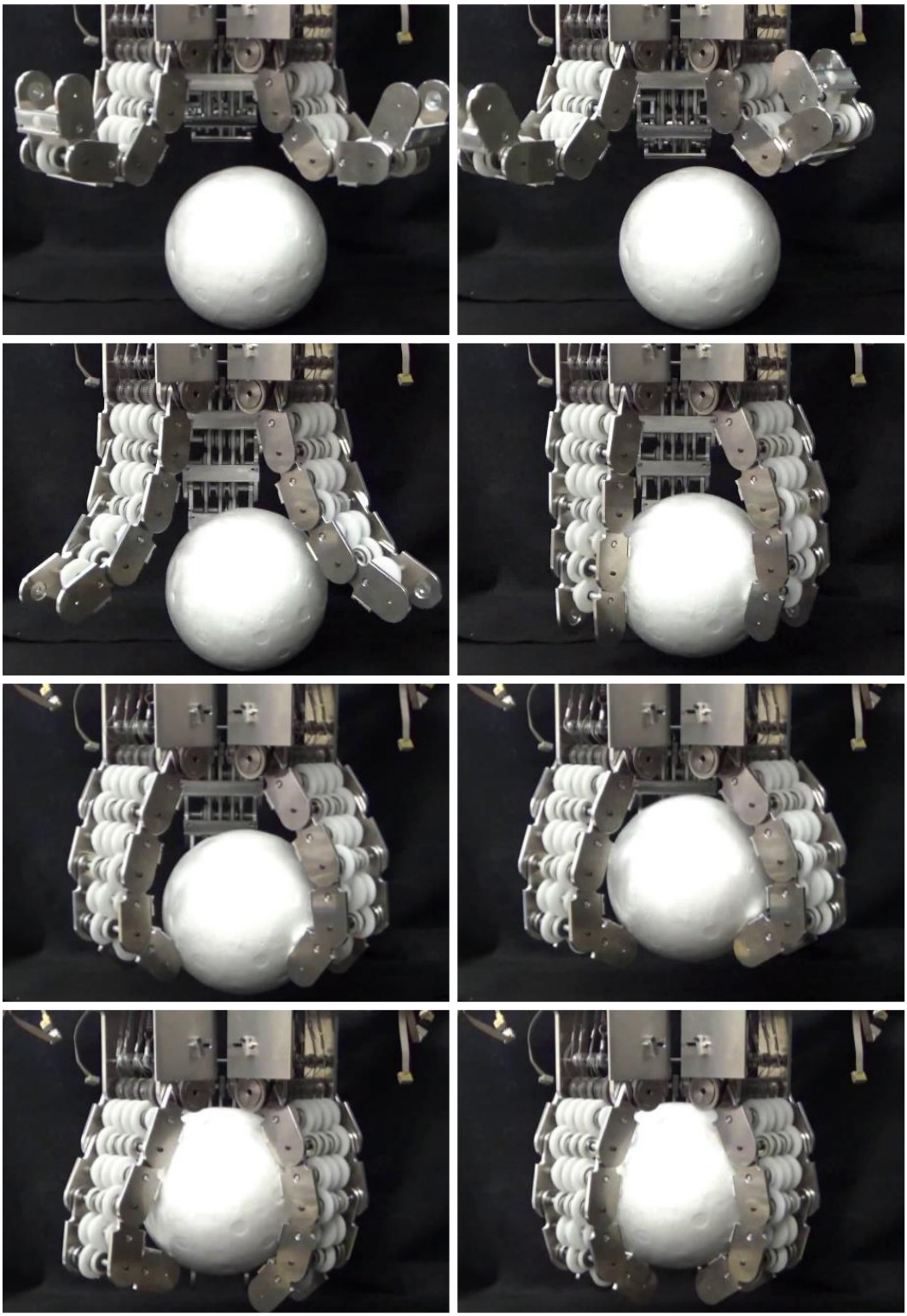


図 4.16　包み込み把持：3 指モデル - 把持物パラメータ (g)

4.4　関節の剛性可変機構の効果

関節の剛性可変機構により意図した効果が得られるかどうかを確認するために，2 つの実験について低剛性時と高剛性時における結果を比較した．一方は指部を重力方向に垂らした状態でのすくう動作であり，もう一方は柔軟物体の把持である．両実験共，ドライビングモータは同一の角度制御を行い，VSM モータにより剛性を変化させた際の比較を行う．

まず，すくう動作の実験について，物体の重量は 320[g] とし，剛性は駆動直後から一定に保つ．VSM モータ角度は，低剛性時が 20[deg]，高剛性時が 80[deg] とした．低剛性時の結果を図 4.17 に，高剛性時の結果を図 4.18 に示す．尚，実験結果はそれぞれの動作における各時間の状態を表したものである．

図 4.17 で示した低剛性時の把持では，物体重量を支えきれずに落としてしまう結果となった．また屈曲後，指部重心が左方向へ移動することにより根本側の関節に反時計回りのモーメントが発生し，指示した角度とは逆に曲がっていることが分かる．図 4.18 で示した高剛性時の把持では，物体をすくう形での把持を成功させ，根本側関節が逆方向へ回転する傾向も小さくなっていることが確認できる．この結果より，リンクに重力によるトルクが作用する中でも，屈曲動作及び物体把持を行えるだけの剛性を持たせられることが確認できた．

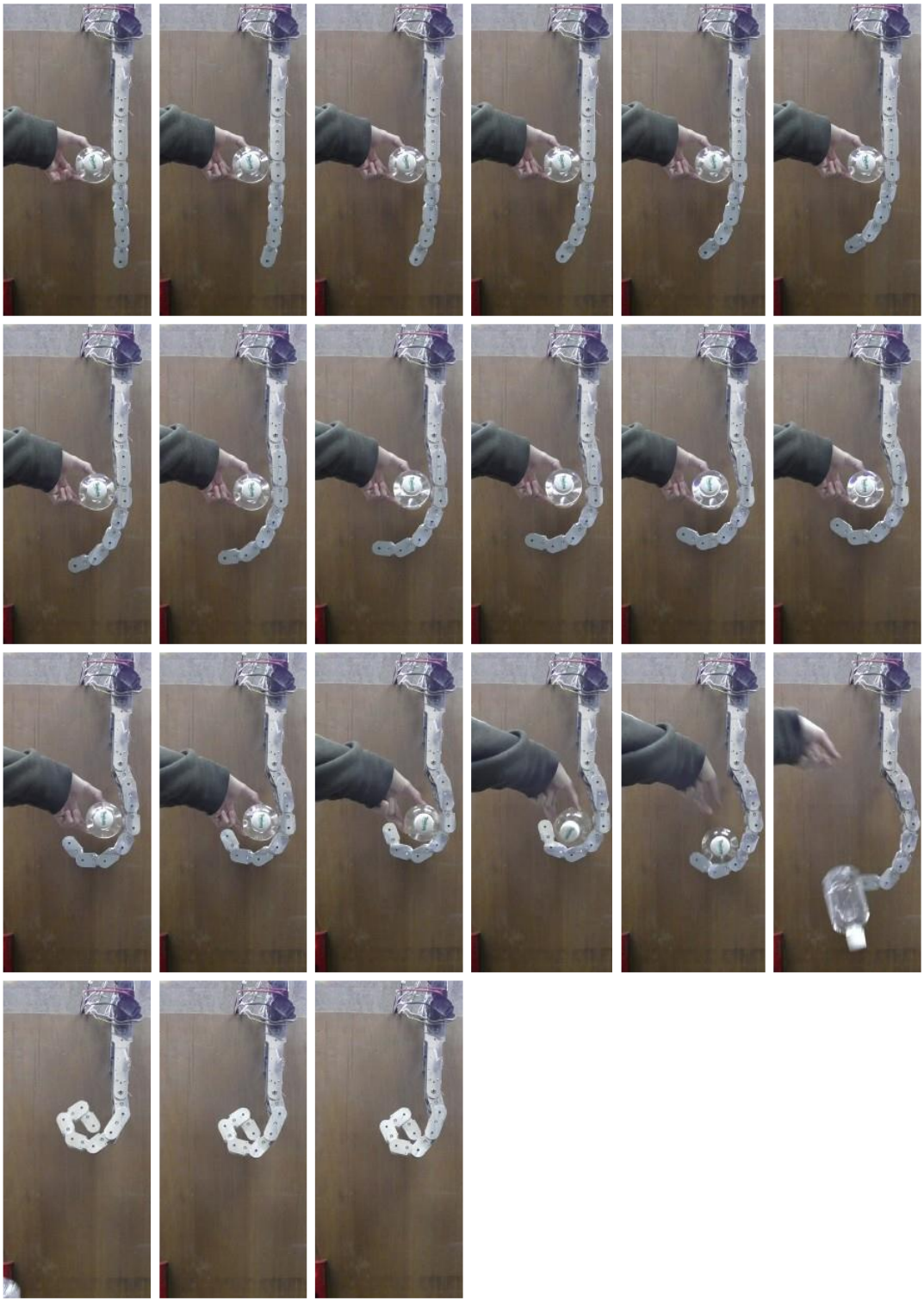


図 4.17　すくい動作：7 関節モデル-低剛性

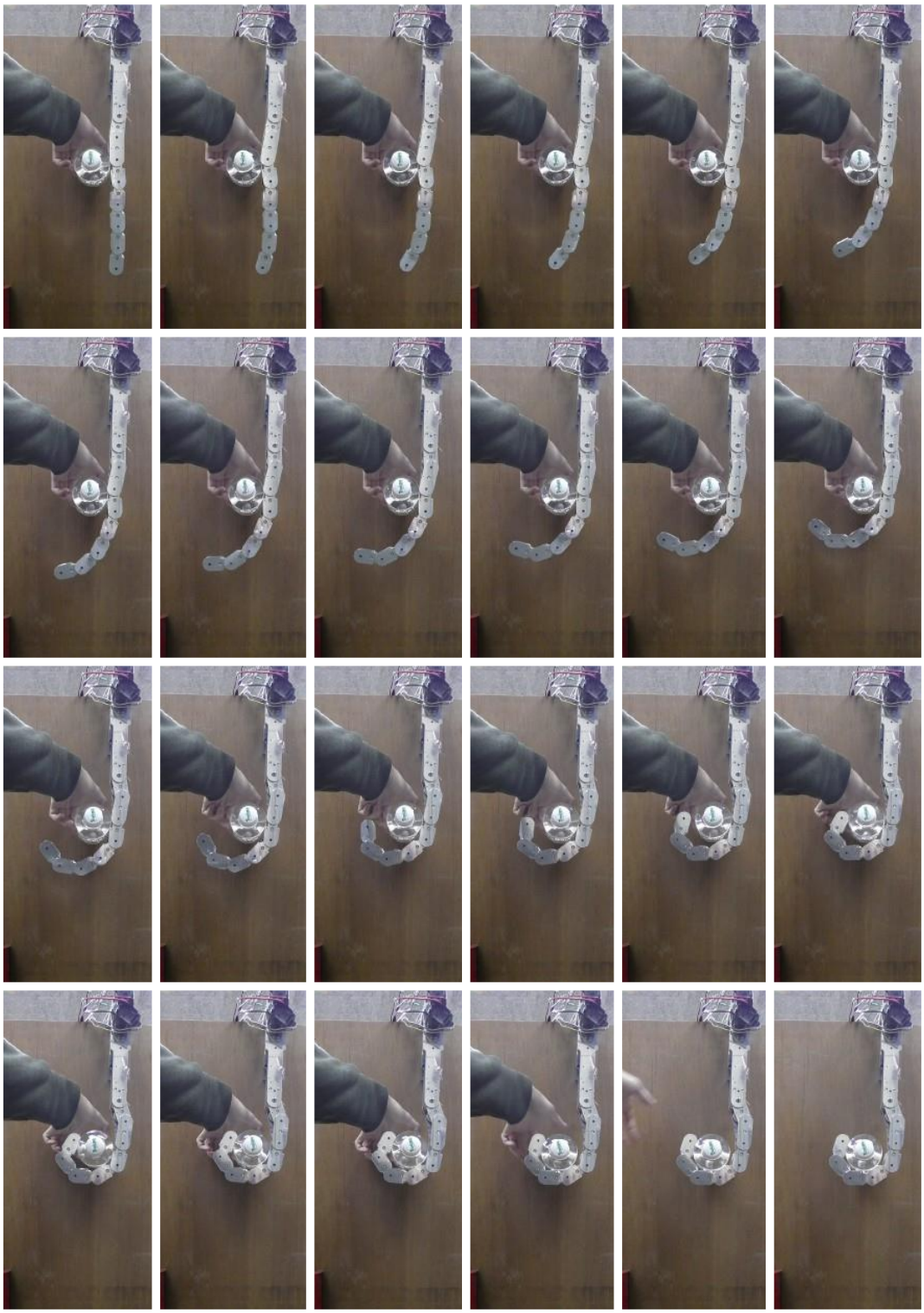


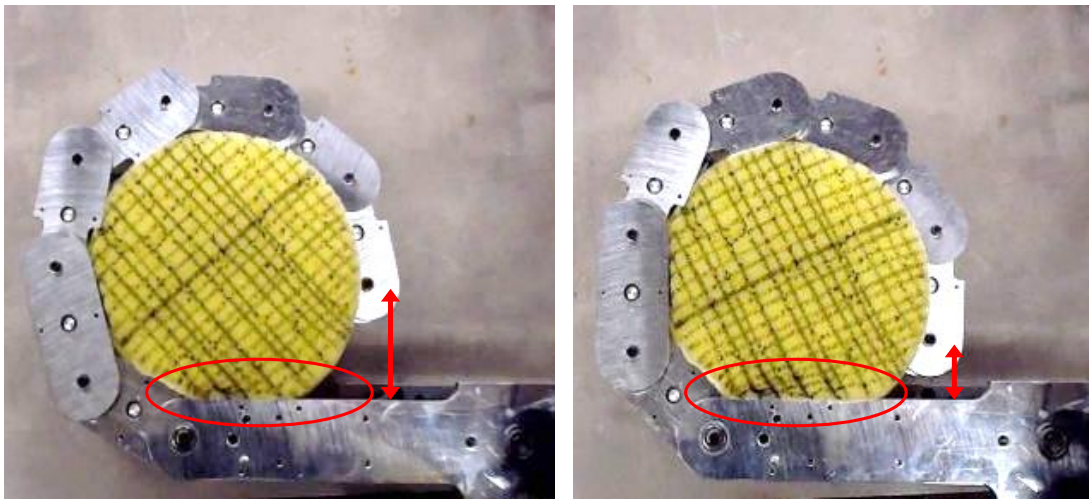
図 4.18　すくい動作：7 関節モデル-高剛性

次に柔軟物体の把持について，低剛性と高剛性それぞれの結果を確認した．まず低剛性で把持した状態から，駆動モータを同角度に維持したまま剛性のみ変化させる．把持対象はスポンジとし，剛性変化前と変化後におけるそれぞれの状態を図 4.19 に示す．

実験結果より，低剛性時にほとんど潰さずに把持できていた物体に対し，高剛性時では変形を引き起こしてしまっていることが分かる．図中の枠で囲んだ部分のスポンジとリンクの接触面積や，指先リンクとベースリンクの距離等からそのことが確認できる．これは高剛性の状態は関節剛性が大きく，グリッパの姿勢は物体との接触力の影響を受けにくくなっている為である．この結果より，低剛性時には，弾性物体に対して柔軟な把持が可能であることが確認できた．

これら 2 つの実験より，剛性可変機構の有効性を示すことができた．それは，リンクに外部からのトルクが発生する中では高剛性にすることで安定した動作を可能とし，弾性物体を把持する際には低剛性にすることで柔軟な把持を可能とする．これらを状況に合わせて調整することで臨機応変な動作を実現できることである．

本実験において，ドライビングモータと VSM モータの制御角度は，操縦者の目視により経験的に決めている．グリッパ単体で状況に合わせて最適な把持動作をとろうとする場合，接触を検知するシステムを有する等，何かしらのフィードバックが必要になると言えるだろう．



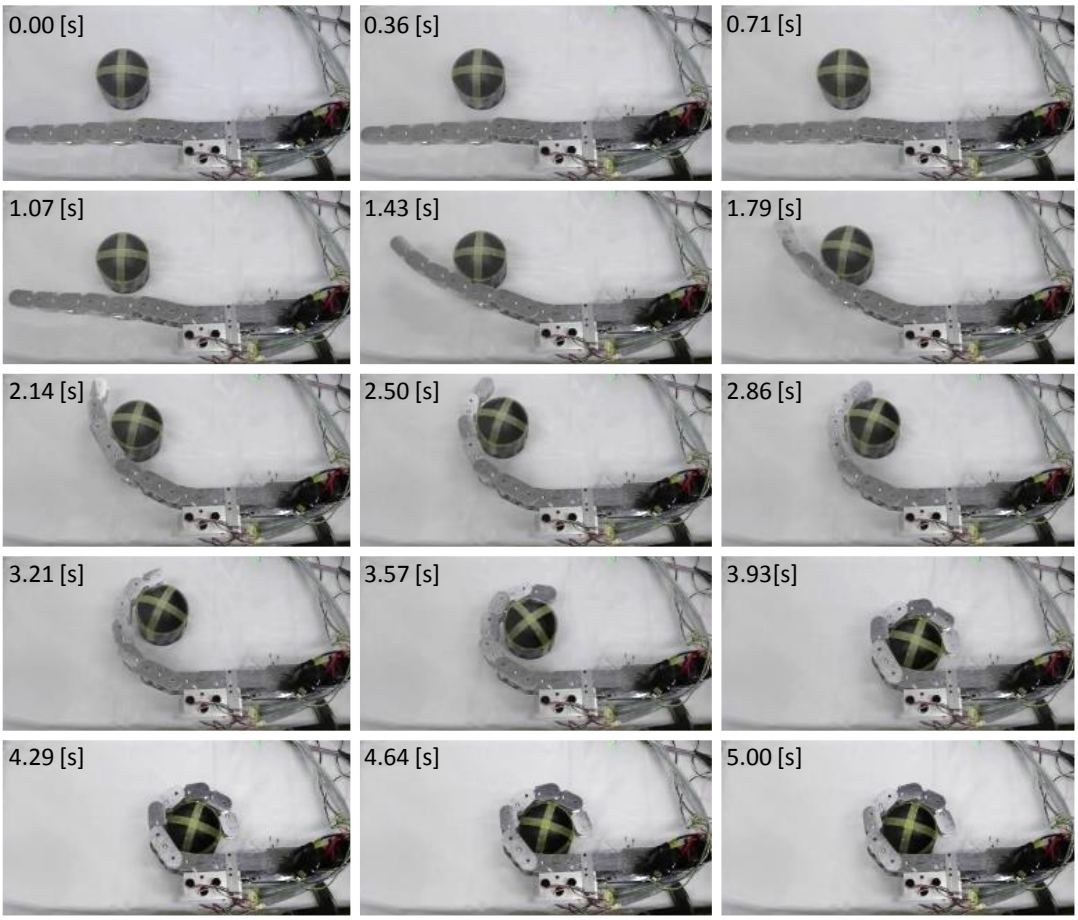
Low stiffness state　　High stiffness state

図 4.19　柔軟物体把持：7 関節モデル

4.5　力覚システムの評価

最適な把持動作を得るためにフィードバックを利用する手法として，ベースギアの回転角を計測することを提案する．ベースギアは，外部からの力によりリンクに対してトルクが発生していない場合には，バネによりその回転が拘束されている．この角度を読み取ることにより，各関節が外部から与えられるトルクに対して引き起こしているなじみ動作をある程度予測することが可能となる．その結果，物体との接触や把持状況の把握につながるのではないかと期待する．角度の計測には関節数分のロータリエンコーダを使用し，設置方法については既に説明した通りである．実際に把持を行う際にはエンコーダと干渉しない方向へ屈曲させるものとする．本節では，物体を指部中腹へ配置した場合と基部近辺に配置した場合について，実験結果とシミュレーション結果を示す．それぞれを配置 A，配置 B とする．

まず配置 A，指部中腹へ物体を配置した場合について，実験結果を図 4.20 に，シミュレーション結果を図 4.21 に示す．それぞれ，計測中の各時間における状態，実際に計測したベースギアとドライビングモータ角度の時間変化を表すグラフを示している．ドライビングモータを駆動させた時刻を 0[s] とする．動画の記録より，この実験では 1.2[s] で物体と接触し，3.8[s] で包み込み把持が完了したことが確認できた．ベースギアの推移を表したグラフより，接触タイミングの 1.2[s] で第 1 関節がなじみ動作を引き起こしている．このことから新たな外力が働いたと推測でき，対象物と接触したと分かる．続いて，3.8[s] から全関節のベースギアが一斉に回転を始める．これは，本来リンクへ伝わるはずだった動力がほぼ全てベースギアへ伝わっている状態であり，リンクが屈曲方向へ姿勢を変化できない状態であることを示している．よって，包み込み把持が完了したと推測できる．その後モータが停止するまで，さらになじみ動作を引き起こしている．また，図 4.21 に示したシミュレーション結果では，およそ 0.5[s] で物体と接触し，それと同時に第 1 関節のベースギアが回転を始めている．そして，およそ 2.5[s] に包み込み把持が完了しており，そのタイミングで全関節のベースギアが回転を始めている．各タイミングの時刻は異なるが，接触時に第 1 関節がなじみ動作を引き起こし，把持完了のタイミングで全関節が一斉になじみ動作を引き起こす．その後モータ停止まで増加し続けるという点で，実験結果とシミュレーション結果が一致することを確認できた．実験結果とシミュレーション結果の間に生じているタイミングのずれは，機構上の摩擦やバックラッシュが原因で実験結果に遅延が起きていると考えられる．



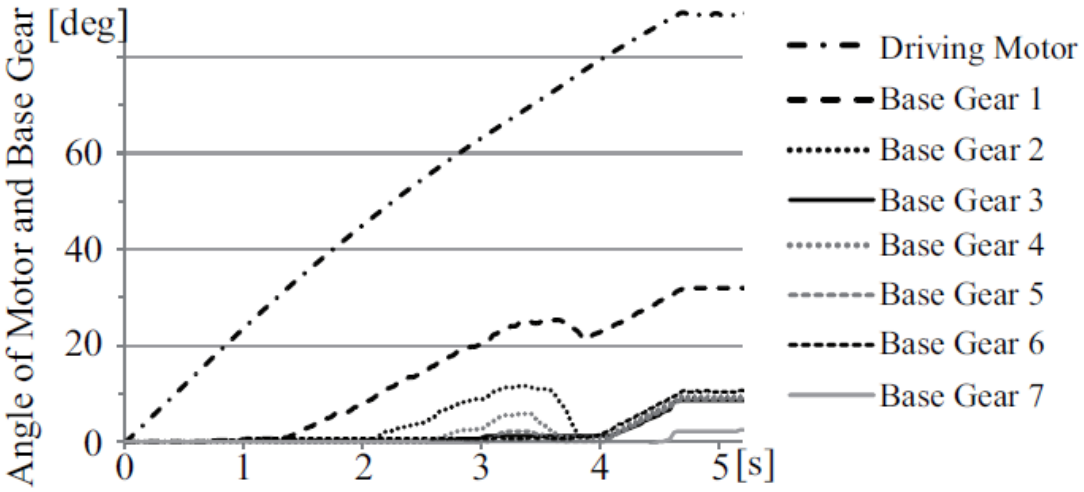


図 4.20　接触情報の検出：配置 A の実験結果

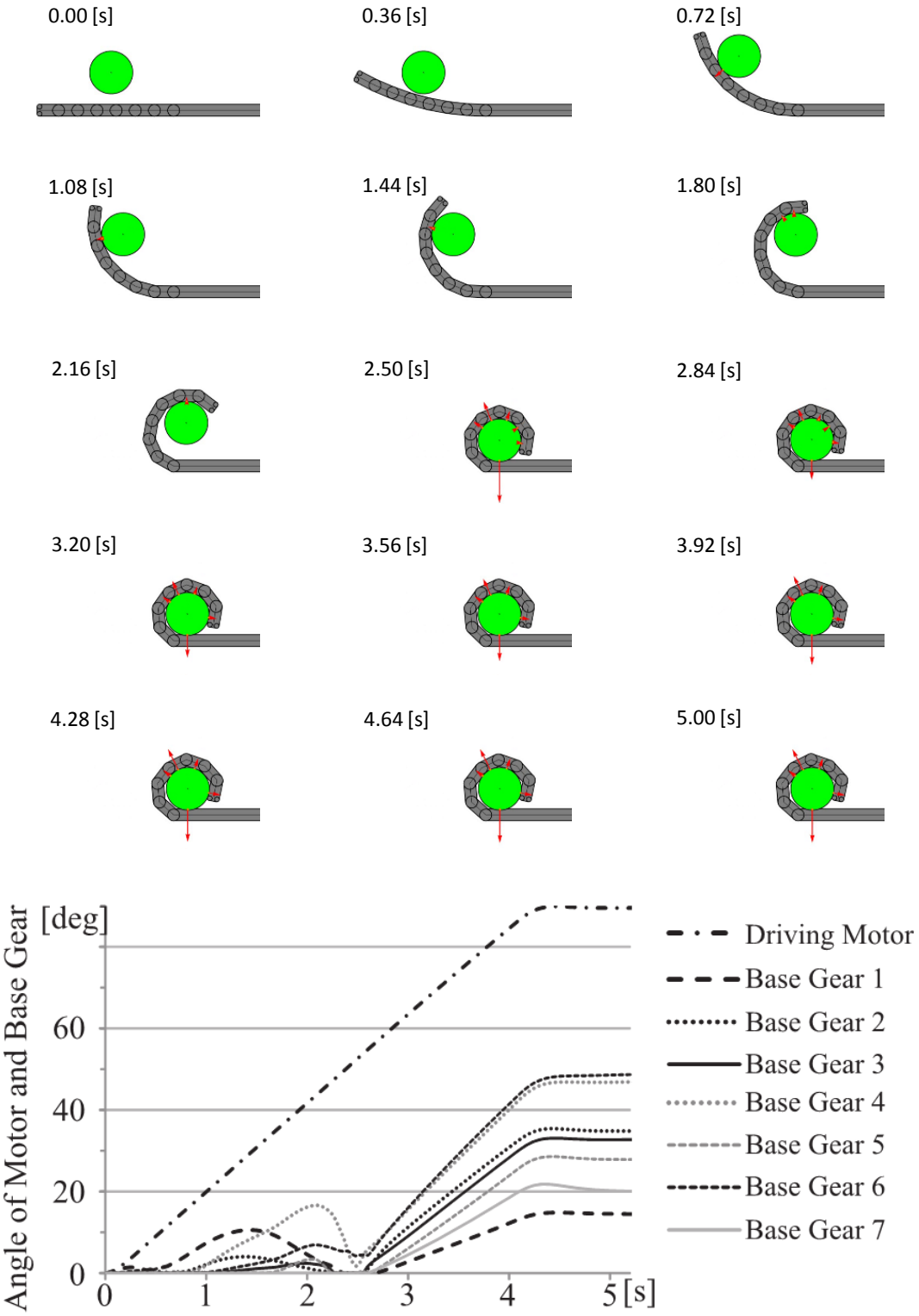


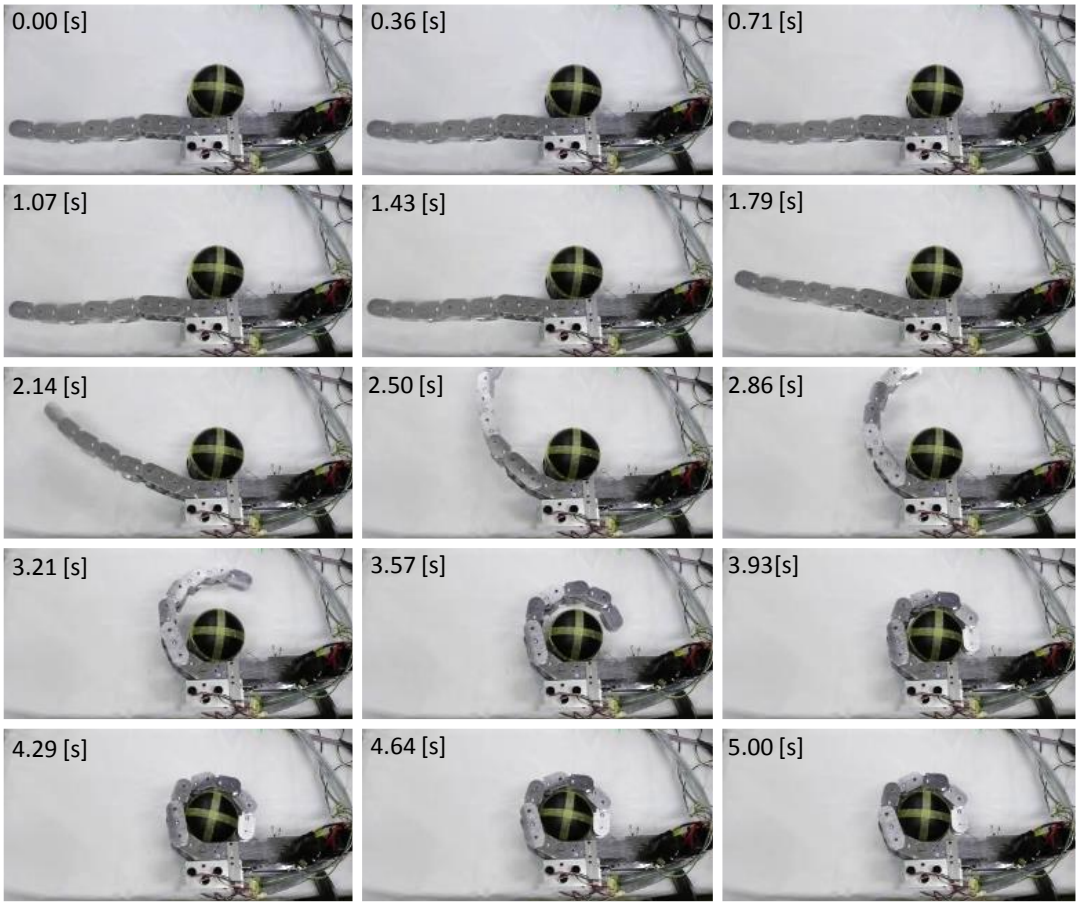
図 4.21　接触情報の検出：配置 A のシミュレーション結果

続いて，把持物を配置 B，第 1 関節付近へ配置した場合について，実験結果を図 4.22 に，シミュレーション結果を図 4.23 に示す．

先ほどと同じくドライビングモータを駆動させた時刻を 0[s] とする．動画の記録より，3.5[s] で物体と接触するのとほぼ同時に包み込み把持が完了したことが確認できた．ベースギアの推移を表したグラフより，およそ 3.5[s] で全関節のベースギアが一斉に回転を始めていることから，包み込み把持が完了し，リンクがそれ以上屈曲できない状態であると分かる．2.5[s] で第一関節がなじみ動作を引き起こしているが，これは摩擦が動作に影響を及ぼしており，第一関節が一定角度以上になると回転を妨げる様に作用してしまうことが確認されている．

図 4.23 に示したシミュレーションにおいては，接触及び把持完了のタイミングはおよそ 2.5[s] である．なお，配置 B でも実験結果とシミュレーション結果の間にタイミングのずれが生じているが，配置 A と同様に，機構上の摩擦やバックラッシュが原因で実験結果に遅延が起きていると考えられる．

以上の結果及び考察より，接触及び包み込み把持完了のタイミングといった現在の把持状況の把握は，各関節に対応するベースギア角度を測定することにより可能であると言える．また，それぞれのタイミングを得るだけであれば，ロータリエンコーダのような高精度なセンサではなく，ひずみゲージなどを用いて引張バネの伸び量を監視すれば達成できるものと思われる．よって提案した力覚システムは，今後よりコンパクトになり実装しやすくなるものであると期待する．



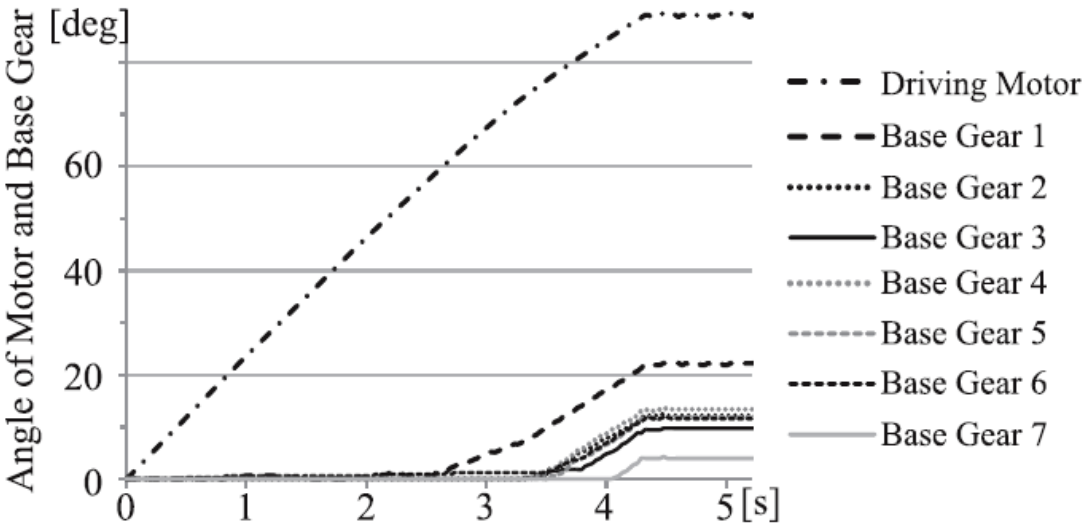


図 4.22　接触情報の検出：配置 B の実験結果

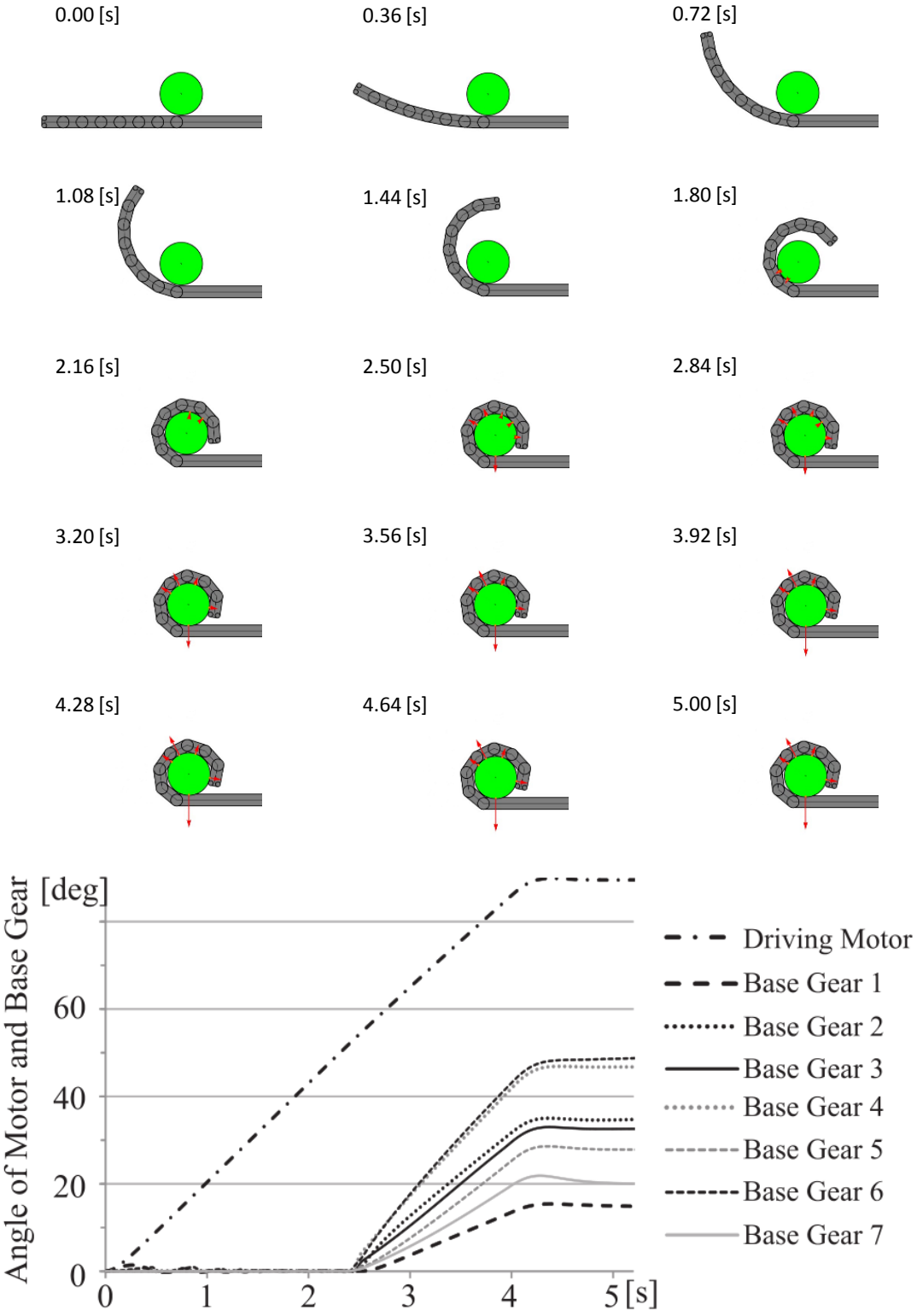


図 4.23　接触情報の検出：配置 B のシミュレーション結果

4.6　関節剛性パターンに伴う把持力の分布

包み込み把持を行う際，その把持力は多くの場合において複数に均等に分散されていたほうが良い．把持力，つまり接触力の分布は，本機構においては関節剛性のバランスに依存する．つまり，ベースギアを拘束している引張バネのバネ係数の並びに大きく左右される．

そこで，以下の 2 種類のバネ係数パターンについて検討する．パターン (1) は，表 4.2 でも示した基本となるバネ係数パターンである．これは高速で屈曲動作を行う場合や，重力影響下で動作する場合に，慣性力や重力といった外力の影響を受けやすい関節ほど，大きい剛性を持つパターンである．パターン (2) は，ピンチングのための指先機構無しでもある程度引き寄せを行えるパターンである．指先の関節剛性を高めることは，ピンチング時の引き寄せや，1 指で引っ掛けての手繰り寄せなどに対して有効な手段である．そこでパターン (2) は，パターン (1) を元に指先 2 関節のバネ係数を変更したものとした．

表 4.7　バネ係数パターン [N/mm]

	1st joint	2nd joint	3rd joint	4th joint
(1)	3.50	0.90	0.50	0.30
(2)	3.50	0.90	0.90	1.50

図 4.24 に剛体とみなせる物体の接触力について，実験結果とシミュレーション結果を示している．図中の矢印は接触力を表している．いずれの結果も左右の指のドライビングモータを 60[deg] まで回転させ，VSM モータは 20[deg] を維持した場合のものである．パターン (1) では指部の全てのリンクで接触し，接触力もある程度均一に分散しているのに対し，パターン (2) では 4 点接触となり，物体が受ける力が偏ってしまっている．

図 4.25 に柔軟物体を把持した場合の変形と，有限要素法による応力分布をシミュレーションした結果を示す．物体の各要素の色は応力の大きさを表現しており，赤色に近いほど大きい応力が発生しており，黄色に近いほど無負荷状態に近い状態である．これらの結果から，パターン (2) では大きな応力が発生するポイントが存在するのに対し，パターン (1) ではよく分散していることが分かる．また，物体全体に生じている変形からも，その傾向を見て取ることができる．

これらの実験およびシミュレーションの結果から，2 指により挟み込むような形での包み込み把持による把持力の分散には，基部側の関節剛性が大きいパターン (1) のようなバランスが最も効果的であるといえる．

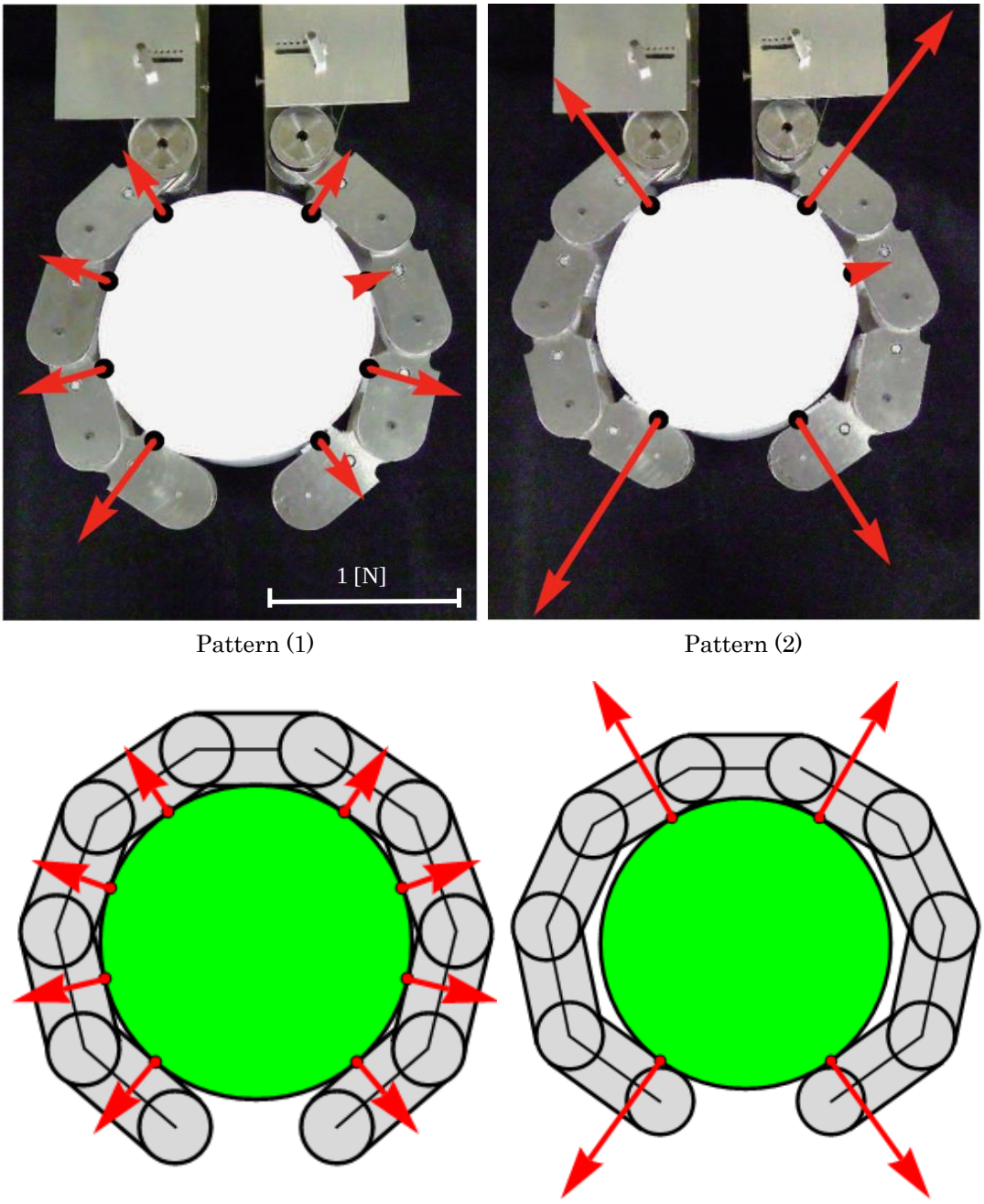
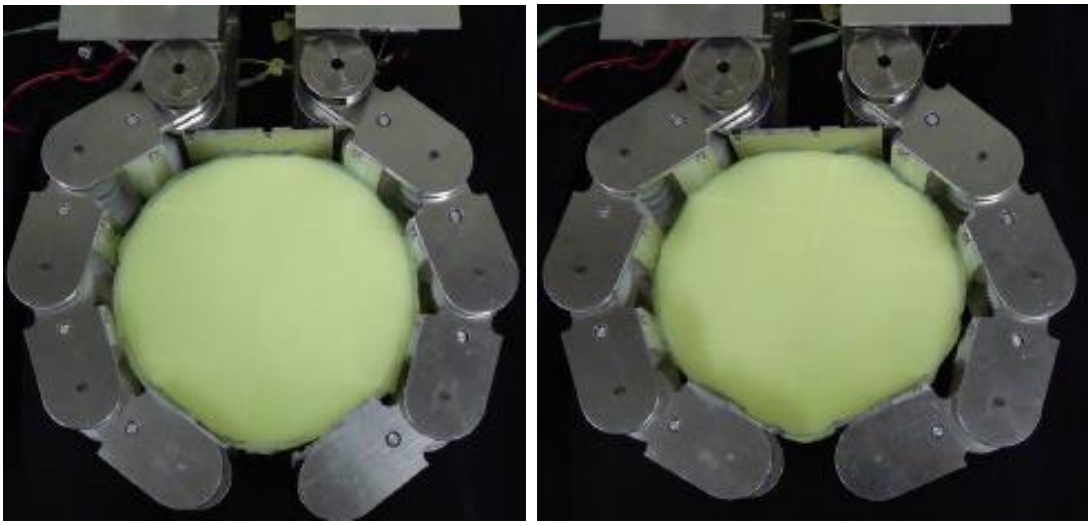
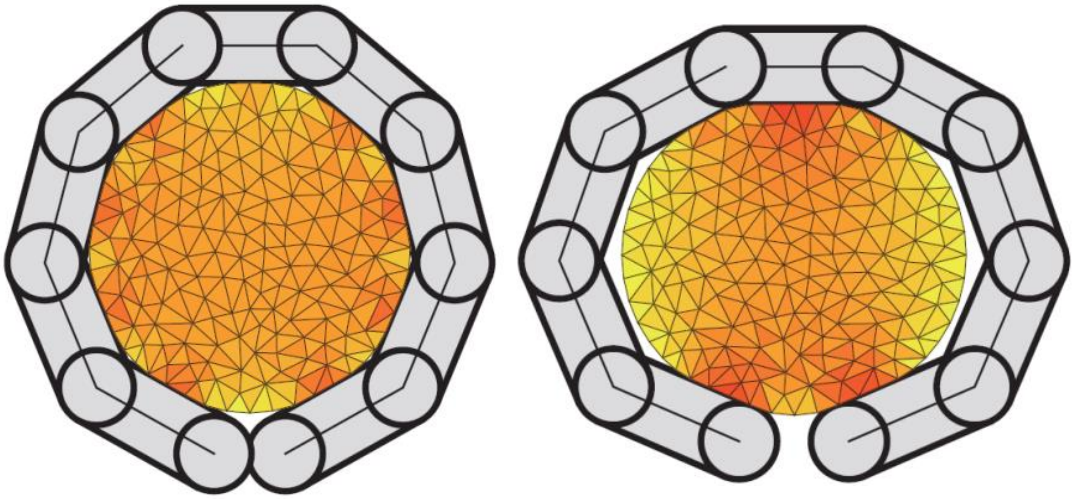


図 4.24　剛体の場合の接触力分布



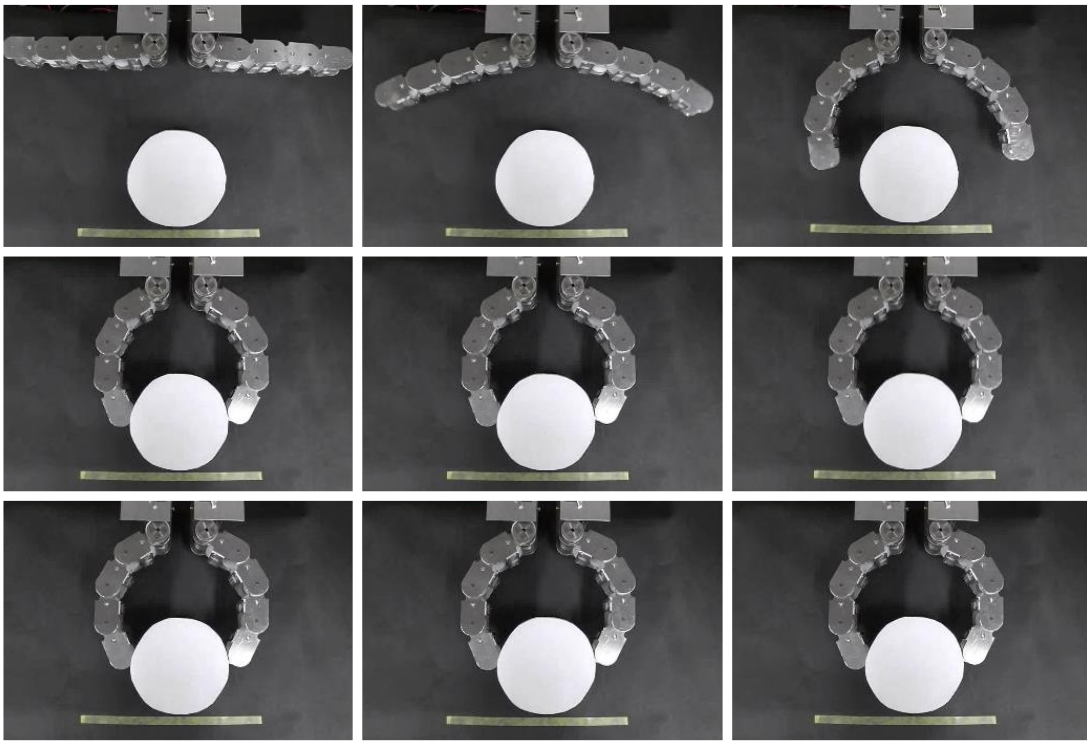
Pattern (1)　　Pattern (2)



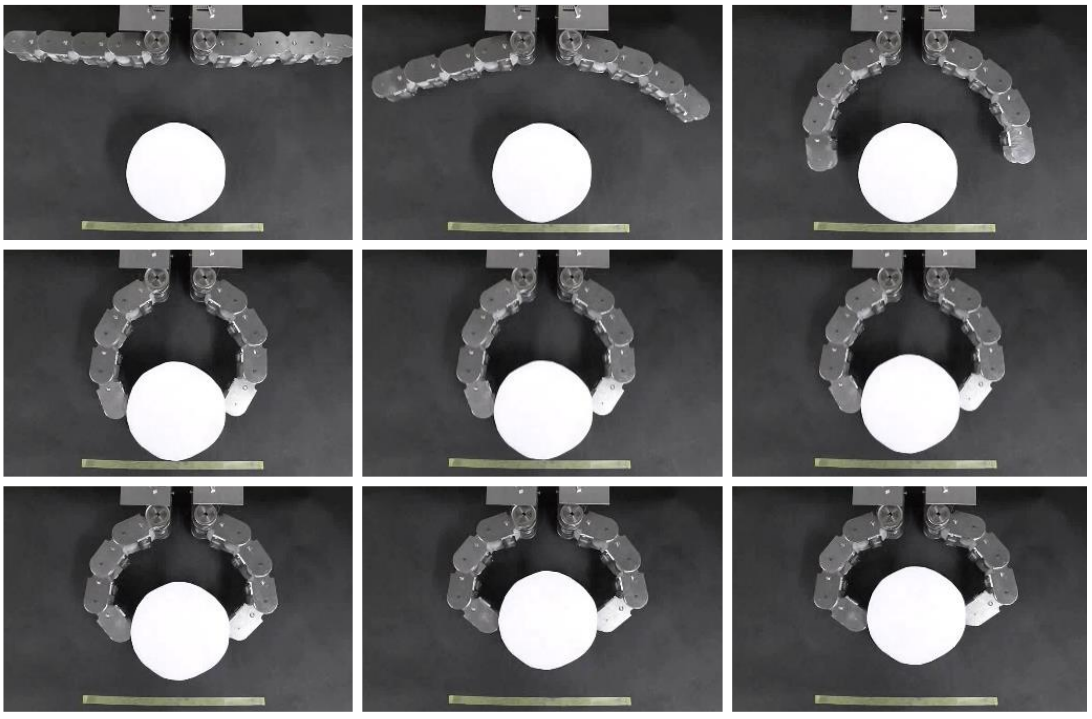
Pattern (1)　　Pattern (2)

図 4.25　弾性体の場合の応力分布

また，2 種類のバネ係数パターンでピンチングによる引き寄せを試みた場合，どのような違いが生じるかを示す．図 4.26 にそれらの結果を示す．共にドライビングモータには一定の目標角度を与えただけの制御だが，パターン (1) では指先が触れてから物体の位置がほとんど変化していないのに対し，パターン (2) では基部側へ大きく引き寄せていることが確認できる．しかし，指先と把持物がすべらないと仮定すると，この状態から包み込み把持へ移行するには何かしらのハンドリングが必要になる．このことから，前に述べたピンチング時に物体の引きよせをスムーズに行う指先機構が望まれる．



Pattern (1)



Pattern (2)

図 4.26　ピンチングによる引き寄せ動作

4.7　ピンチングから包み込み把持への移行

本節では，ピンチング時に物体の引きよせをスムーズに行う指先機構について評価する．実際に指先にローラを取り付けて把持を行った結果を図 4.27 に示す．なお，バネ係数はパターン (1) による動作である．

指先機構を搭載していない結果である図 4.26 では，ピンチング後に物体を動かすことができていなかったが，本結果ではピンチング後に物体を引き寄せ包み込み把持まで行えることを確認した．さらに，これらのピンチング，引き寄せ，包み込み把持に至る一連の動作をドライビングモータの単純な制御で実現できているといえ，期待に沿った効果が得られた．

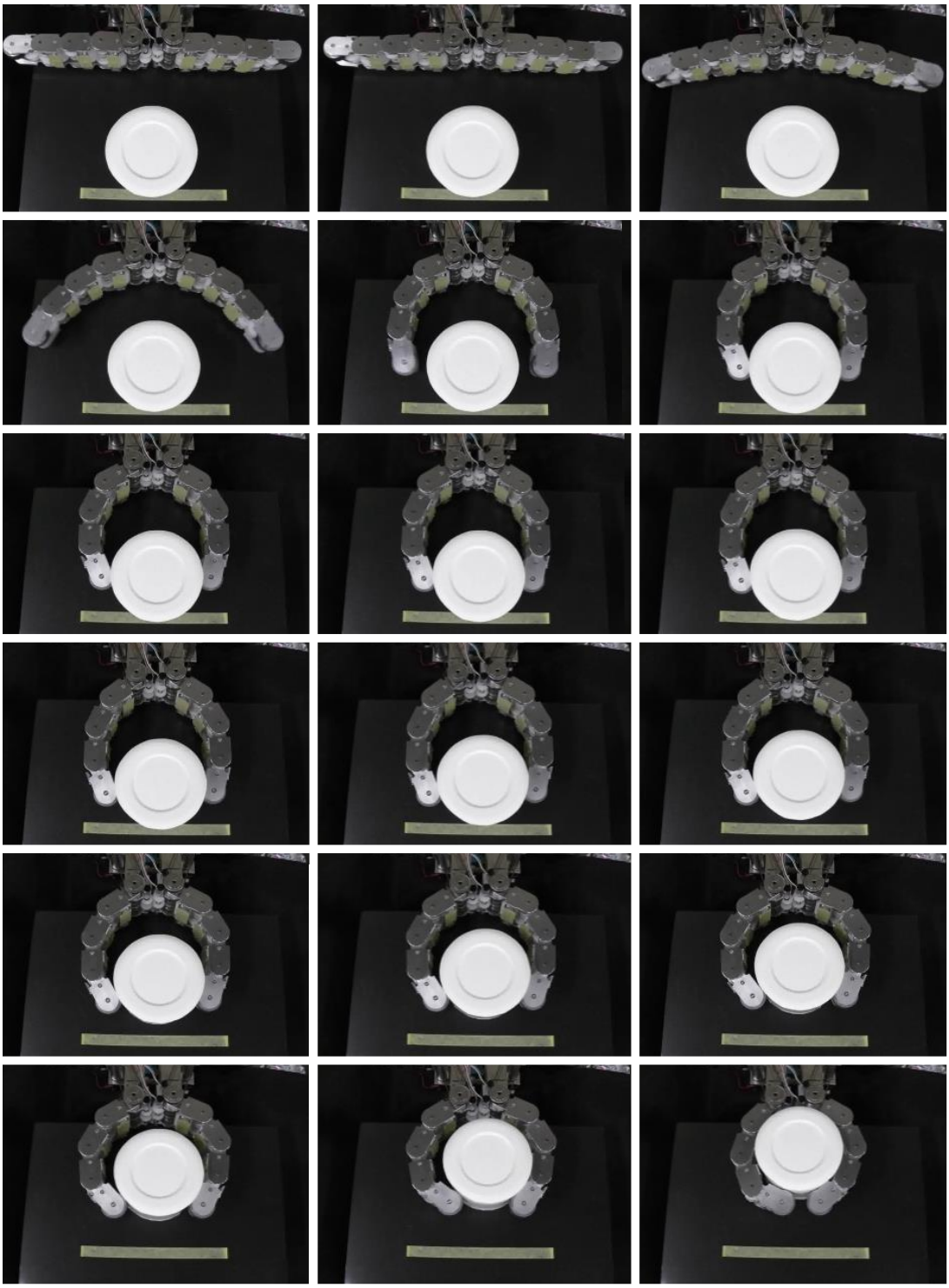


図 4.27　指先ローラ機構によるピンチングと引き寄せ動作

4.8　まとめ

本章では，実際に製作した実験機とシミュレーションによる評価を行った．序論で示した目的のうち，「物体形状に沿った柔らかい把持を可能とする」という項目に関して，1 指モデル，2 指モデルによる実験により，それぞれ達成できていることを確認した．「本質安全を確保した力覚システムを有する」という点に関して，現在の把持状況を，第 1 関節に位置するベースギアの角度を測定することで，把握可能であることを確認した．「包み込み把持に加えピンチングを可能とする」という項目に関して，指先にローラを設置する機構を実際に運用し，その効果によりピンチングから物体を引き寄せ，また包み込み把持にスムーズに移行することを確認した．「状況に応じて関節剛性の制御を可能とする」という項目に関して，重力の影響下にある場合の把持や柔軟物体の包み込み把持実験から，低剛性，高剛性それぞれに有利な点があることを示し，剛性可変機構の有効性を示した．

第 5 章

結論

5.1　本論文のまとめ

本研究は,「剛性可変機構を有する多関節グリッピングハンドに関する研究」と題し，災害現場で救助や捜索活動などを行う災害対応ロボットに搭載することを目指したグリッピングハンドの開発を目的とした.

第 1 章では，現在運用されている災害対応ロボットを紹介し，その作業内容から包み込み把持という把持形態が適していると論じた．これは把持対象物との接触を，指先だけでなくリンクに対しても積極的に許容することで，接触点数を増やし，より確実な把持が達成される．そして，この包み込み把持を主目的に持つハンドを災害対応ロボットのツールとして利用する場合の利点及び課題を示した．それらのことから，本研究で開発するグリッピングハンドが満たすべき項目を 5 つ挙げ，次のように定めた.「物体形状に沿った柔らかい把持を可能とする」,「シンプルな制御で動作可能な劣駆動機構である」,「状況に応じて関節剛性の制御を可能とする」,「本質安全を確保した力覚システムを有する」,「包み込み把持に加えピンチングを可能とする」.

第 2 章では，第 1 章で定めた項目を満たすようなハンド機構の提案を行った．1 関節に求められる機能は，角度を制御可能であることと剛性を制御可能であることの 2 つである．この 2 入力の要求を満たすため，差動歯車機構を導入した．使用する差動機構は 2 入力 1 出力という特徴を持ち，入力の一つはモータからのトルクを与え，もう一方はバネによる受動的なトルクを与える．能動的な角度制御を行う場合はモータトルクがそのまま関節へ出力されるが，関節が外部との干渉により固定された，または過剰に回転した場合に，バネにエネルギーを蓄えるように動作する．バネの復元力が外部干渉に対する抵抗力となり，バネ力を制御することで関節剛性を可変とする．本機構は劣駆動で多関節に拡張可能であり，目的とする要件を満たす機構の提案を行った.

第 3 章では提案したグリッピングハンド機構の動力学シミュレーション手法について論じた．動力学シミュレーションは，実機では厳密な測定が困難な接触力の予測や，バネ係数などのパラメータ設計に役立てることを主な目的とし，導入した．各運動要素である，リンク機構，モータ，把持物の運動方程式の導出過程を示し，また把持物とリンクの接触という要素同士の干渉について，その扱いを示した．運動方程式の導出にはラグランジュ法を主に用い，弾性物体に関しては非線形有限要素法を用いる．接触処理には，判定を行う要素間に発生している貫入量に比例する反力が生じる，ペナルティ法を用いる．これらの手法を元に，基礎となる屈曲動作および包み込み把持のシミュレーション結果をいくつか提示し，非常に単純な制御で物体形状になじむ包み込み把持が達成できることを確認した．

第 4 章では，まず実験システムについて説明した．製作した実機として，7 関節を持つ 1 指モデルと，4 関節を持つ 2 指または 3 指モデルを提示し，専用の実験で使用する各種センサの取り付け方法，制御プログラムと実機を繋いでいる制御系について説明した．実験およびシミュレーションによる評価を行い，物体形状になじむ柔らかい包み込み把持が可能であること，またそれがシンプルな制御で達成されていること，剛性可変機構および力覚システムの有効性，さらにピンチングが達成可能であることを確認した．

5.2　本研究の成果

本論文では多関節グリッピングハンドの機構を新たに提案し，その評価を行った．提案した多関節グリッピングハンドの新規性は，関節ごとに独立して物体形状になじむことが可能であり，さらに関節剛性を制御する機構を有することで，従来の研究では両立することが困難であった，物体形状になじむ柔らかい把持と，安定した姿勢の維持が必要なすくう動作やピンチングといった把持形態を達成したことにある．また，本質的な安全を有する力覚システムの実現も，この分野においては新たな取り組みであるといえる．

製作したハンドは災害現場など，対象物の形状が未知であり，またハンドに対して様々な外力が加わる可能性の高い場所での使用において有効であることが示唆された．このことは，従来これらのフィールドで実用的な動作を行うには不十分であった，多関節グリッピングハンドを使用するための道筋を示した．

5.3　本研究の展望

本研究の成果について示したが，実際に運用するには提案したハンドはまだいくつかの課題を持っている．

一つは，指部表面に関する議論を行っておらず，実際の運用にあたってはその柔軟性，摩擦特性などの議論が必須であると考えられ，今後率先して取り組むべき課題である．

また，把持対象物の多様化も挙げられる．本論文では，グリッピングハンドのサイズに対して，おおむね丁度収まる大きさの対象物を取り上げてきた．しかし，摩擦などを利用して大きな物体を把持することや，板や紙のように薄い物体を把持することを達成すれば，より実用的なハンドとなるだろう．それらの実現の手段として，関節剛性のパターンを調節することなどが考えられ，今後の拡張の一つの案として挙げられる．

謝辞

本研究を行うにあたり，また，学部・修士に在籍時から御指導・御鞭撻をして頂きました本学工学部機械工学科・小金澤鋼一教授に心より厚く御礼申し上げます．

本論文をまとめるにあたり，本学工学部・奥山淳教授，山本佳男教授，甲斐義弘教授，本学情報理工学部・稲葉毅教授には，研究内容および論文に対するご意見を頂きまして，心より厚く御礼申し上げます．

実機製作時において，加工について御指導して頂きました先生方に心より厚く御礼申し上げます．

これまで研究および学生生活を送る上で，様々な面でサポートして頂きました，本学工学部機械工学科の先生方に心より厚く御礼申し上げます．

学部在籍時に先輩として様々なことを教えてくださった下野さんに，心より厚く御礼申し上げます．

そして，研究において同班として共に研究を築いてきた，佐山君，野村君，多くのメンバーに心より感謝致します．また，様々な面で御支援下さいました小金澤研究室の先輩・同輩・後輩の全ての皆様に心より感謝致します．

参考文献

[1] 松野文俊, 阪神淡路大震災を振り返って, 日本ロボット学会誌, Vol.28, No.2, pp.138-141, 2010.

[2] 田所諭, レスキューロボットと福島第一原発事故, 日本ロボット学会誌, Vol.30, No.10, pp.1017-1021, 2012.

[3] 徳田献一, レスキューロボットに見る災害対応の取り組み, 和歌山大学防災研究教育センター紀要, Vol.1, pp.21-26, 2015.

[4] 文部科学省, 大都市大震災軽減化特別プロジェクト, http://www.mext.go.jp/a_menu/kaihatu/jishin/04031203.htm, アクセス 2016/10/1.

[5] 田所諭, 大大特プロジェクトの目的と概要, 日本ロボット学会誌, Vol.22, No.5, pp.544-545, 2004.

[6] 革新的研究開発推進プログラム (ImPACT), タフ・ロボティクス・チャレンジ, http://www.jst.go.jp/impact/program/07.html, アクセス 2016/10/1.

[7] 東京消防庁, ロボキュー, http://www.tfd.metro.tokyo.jp/ts/soubi/robo/05.htm, アクセス 2016/10/1.

[8] 株式会社テムザック, 新型レスキューロボット, http://www.tmsuk.co.jp/lineup/t53_enryu/index.html, アクセス 2016/10/1.

[9] J.R.Napier, The prehensile movements of the human hand, The Journal of Bone and Joint Surgery, Vol.38, No.4, pp.902-913, 1956.

[10] Okada.T, Object-Handling System for Manual Industry, IEEE Transactions on Systems, Man, and Cybernetics, Vol.9, No.2, pp.79-89, 1979.

[11] S.Jacobsen, E.Iversen, D.Knutti, R.Johnson, K.Biggers, Design of the Utah/M.I.T. Dextrous Hand, Proceedings of the IEEE International Conference on Robotics and Automation, pp.1520-1532, 1986.

[12] C.S.Lovchik, M.A.Diftler, The Robonaut hand: a dexterous robot hand for space, Proceedings of IEEE International Conference on Robotics and Automation, pp.907-912, 1999.

[13] L.B.Bridgwater, C.A.Ihrke, M.A.Diftler, M.E.Abdallah, N.A.Radford, J.M.Rogers, S.Yayathi, R.S.Askew, D.M.Linn, The Robonaut 2 hand - designed to do work with tools, Proceedings of IEEE International Conference on Robotics and Automation, pp.3425-3430, 2012.

[14] H.Kawasaki, T.Komatsu, K.Uchiyama, Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II, IEEE/ASME Transactions on Mechatronics, Vol.7, No.3, pp.296-303, 2002.

[15] 川崎, 毛利, 伊藤, 下村, 松波, 花田, 東, 人間型ロボットハンド Gifu Hand III, 日本ロボット学会誌, Vol.22, No.1, pp.55-56, 2004.

[16] 下条誠, 人工の手の研究開発動向, バイオメカニズム学会誌, Vol.38, No.1, pp.3-10, 2014.

[17] 松岡, 長谷川, 本田, 桐下, 作業状態観測と評価に基づく多関節多指ハンド物体操作システム, 日本ロボット学会誌, Vol.17, No.5, pp.696-703, 1999.

[18] 並木, 石川, 視覚フィードバックを用いた最適把握行動, 日本ロボット学会誌, Vol.18, No.2, pp.103-111, 2000.

[19] 横小路, 坂本, 吉川, カメラ画像を併用した多指ハンドにより操られる物体の位置姿勢推定法-ソフトフィンガー型ハンドによる操り制御への適用-, 日本ロボット学会誌, Vol.18, No.3, pp.401-410, 2000.

[20] 金子真, 包み込み把握 Enveloping Grasp, 日本ロボット学会誌, Vol.18, No.6, pp.782-785, 2000.

[21] J.Trinkle, J.Abel, R.Paul, Enveloping, frictionless, planar grasping, Proceedings of IEEE International Conference on Robotics and Automation, pp.246-251, 1987.

[22] M.Kaneko, Y.Tanaka, T.Tsuji, Scale-dependent grasp, Proceedings of IEEE International Conference on Robotics and Automation, pp.2131-2136, 1996.

[23] 八島, 山口, 久保, 多指ハンドのリンク内接触による操り, 日本機械学会論文集 C 編, Vol.64, No.622, pp.2074-2080, 1998.

[24] 小俣, 永田, 多指ハンドによるパワーグラスプの力学的特性, 日本ロボット学会誌, Vol.13, No.4, pp.525-531, 1995.

[25] 金子, 東森, 辻, 包み込み把握の遷移安定性, 日本ロボット学会誌, Vol.16, No.5, pp.712-720, 1998.

[26] 原田, 金子, 古寺, 辻, 複数対象物の Active Force Closure, 日本ロボット学会誌, Vol.17, No.8, pp.1158-1166, 1999.

[27] 田所諭, III-4 レスキューロボット等次世代防災基盤技術の開発, 大都市大震災軽減化特別プロジェクト総括成果報告書, 文部科学省, pp.189-228, 2007.

[28] M.Guarnieri, I.Takao, E.F.Fukushima, S.Hirose, HELIOS VIII search and rescue robot: Design of an adaptive gripper and system improvements, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1775-1780, 2007.

[29] 広瀬茂男, 生物機械工学，工業調査会, pp.191-199，1987.

[30] S.Hirose, Y.Umetani, The development of soft gripper for the versatile robot hand, Mechanism and machine theory, Vol.13, No.3, pp.351-359, 1987.

[31] 東森, 金子, 単一ワイヤ駆動ロボットハンドの Dynamic Preshaping, 日本ロボット学会誌, Vol.22, No.8, pp.997-1003, 2004.

[32] N. Dechev. W. L. Clegjhorn, S. Naumann, Multiple Finger, Passive Adaptive Grasp Prosthesis Hand, Mechanism Machine Theory, Vol.36, No.10, pp.1157-1173, 2001.

[33] B. Massa, S. Roccella, M. C. Carrozza, P. Dario, Design and Development of an Underactuated Prosthetic Hand, Proceedings of IEEE International Conference on Robotics and Autom, pp.3374-3379, 2002.

[34] N. Yamano, S. Takamuku, K. Hosoda, Development of Underactuated Humanoid Robot Hand for Adaptable Grasp, Proceeding of the 2008 JSME Conference on Robotics and Mechatronics, pp.——, 2008.

[35] M. Wassink, R. Carloni, S. Stramigioli, Port-Hamiltonian Analysis of a Novel Robotic Finger Concept for Minimal Actuation Variable Impedance Grasping, Proceedings of IEEE International Conference on Robotics and Automation, pp.771-776, 2010.

[36] 吉田, 安孫子, OTA, 水中作業ロボット用多関節グリッパの開発, 日本機械学会論文集 C 編, Vol.68, No.675, pp.3373-3380, 2008.

[37] 多賀, 中西, 小田, 高収納効率と広い把持範囲を持つワイヤ駆動型把持機構の検討, 日本機械学会ロボティクス・メカトロニクス講演会予稿集, 2A2-04a7, 2016.

[38] 井上, 平井, 柔軟指による把持物体の姿勢制御, 日本機械学會論文集 C 編, Vol.75, No.757, pp.2537-2546, 2009.

[39] 井上, 平井, 柔軟指による物体把持と操作における力学の実験的解明, 日本ロボット学会誌, Vol.25, No.6, pp.951-959, 2007.

[40] 天瀬, 西岡, 安田, 螺旋状巻付き型極軽量ソフトアクチュエータの開発：多様な螺旋形状の生成方法の提案, ロボティクス・メカトロニクス講演会講演概要集,

2A1-N05, 2014.

[41] 阿妻, 吉留, 河原崎, 小型フレキシブルロボットハンドの開発, ロボティクス・メカトロニクス講演会講演概要集, 1P2-D10, 2015.

[42] S.Takamuku,A.Fukuda,K.Hosoda, Repetitive grasping with anthropomorphic skin-covered hand enables robust haptic recognition, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.3212-3217, 2008.

[43] K. Koganezawa, A Mechanical Musculo-Skeletal System for a Human-Shaped Robot Arm, Actuators 2014, Vol.3, pp.124-141, 2014.

[44] R. Schiavi, G. Grioli, S. Sen, A. Bicchi, VSA-II: a Novel Prototype of Variable Stiffness Actuator for Safe and Performing Robots Interacting with Humans, Proceedings of the IEEE International Conference on Robotics and Automation, pp.2171-2176, 2008.

[45] S. Wolf, G. Hirzinger, A New Variable Stiffness Design: Matching Requirements of the Next Robot Generation, Proceedings of the IEEE International Conference on Robotics and Automation, pp.1741-1746, 2008.

[46] 矢田恒二, 歯車応用機構の設計, 機械技術協会, pp.3-21, 2011.

[47] ハーモニック・ドライブ・システムズ, ハーモニックドライブの原理, https://www.hds.co.jp/products/hd_theory/, アクセス 2016/10/1.

[48] 小笠原工業株式会社, 遊星歯車機構, https://www.khkgears.co.jp/gear_technology/intermediate_guide/KHK388.html, アクセス 2016/10/1.

[49] K.Takeda, H.Chiba, K.Koganezawa, The Artificial Finger using The Double Planetary Gear System, Proceedings of the 8th Asian Conference on Multibody Dynamics, ROB-2(3), 2016.

[50] 小金澤鋼一, 遊星歯車機構を用いた 5 指ハンド-最小アクチュエータによる「手」の日常動作の実現-, 電子情報通信学会技術研究報告, ヒューマン情報処理, Vol.112, No.483, pp.87-92, 2013.

[51] Y.Sato, K.Koganezawa, Five finger robot hand with a planetary gear system, Proceedings of the 8th Asian Conference on Multibody Dynamics, ROB-2(2), 2016.

[52] 内田, 下野, 小金澤, 多関節グリッパの新機構, 第 28 回日本ロボット学会学術講演会予稿集, 1M1-3, 2010.

[53] 下野, 玉本, 伊藤, 小金澤, 剛性可変機構を有する多関節グリッパ, 計測自動制御学会論文集, Vol.49, No.1, pp.11-17, 2013.

[54] 下野, 小金澤, 第 12 回計測自動制御学会システムインテグレーション部門講演会, 3J2-1, 2011.

付録

付録 A

Mathematica によるプログラム

- シミュレーションベース部分 -

(Mathematica のバージョン：11.0.1)

Mathematica の設定/初期処理

```
(* ログビューワーを閉じる - 2回目以降の全コード実行用 *)
LAB`LogClose[];
(* ダイアログを全て閉じる - 2回目以降の全コード実行用 *)
LAB`Simulation["close", All];
LAB`Simulation["close:analysis", All];
(* 履歴保存数を0に設定する *)
$HistoryLength = 0;
(* コンテキストが[Global],[LAB]であるシンボルを除去する *)
If[Length@Names["Global`*"] != 0, Remove["Global`*"]];
If[Length@Names["LAB`*"] != 0, Remove["LAB`*"]];
(* FEMパッケージのインストール *)
Needs["NDSolve`FEM`"];
(* Cコンパイラパッケージをロード *)
Needs["CCompilerDriver`"]
Needs["CCompilerDriver`GenericCCompiler`"]
(* Cコードの操作に使うデフォルトのCコンパイラを設定する *)
$CCompiler = {
   "Compiler" -> CCompilerDriver`GenericCCompiler`GenericCCompiler,
   "CompilerInstallation" -> "C:\\TDM-GCC-64",
   "CompilerName" -> "x86_64-w64-mingw32-gcc.exe"
   };
(* コンパイル方法のデフォルト設定 - "WVM" or "C" *)
$CompilationTarget = "C";
(* このノートブックのディレクトリを取得 *)
LAB`NotebookDirectory = NotebookDirectory[];
(* コンパイルデータを保管するビルドディレクトリ *)
LAB`BuildDirectory = $UserBaseDirectory <>
   "\\ApplicationData\\CCompilerDriver\\BuildFolder";
(* ログビューワーを使用する *)
LAB`LogEnabled = 1;
(* ログビューワーの初期値 *)
LAB`LogProcess = "Finished";
(* 画像の登録 *)
LAB`Image["update"] = ***;
LAB`Image["setting"] = ***;
LAB`Image["cross"] = ***;
```

各コンテンツのリストを作成

```
(* データ群のモデル = データモデル *)
LAB`Contents["data model"] = {
   "config", "run", "device", "system"
   };
(* 実行状況の種類 - データモデル[run]に属する *)
LAB`Contents["run state"] = {
   "set", "standby", "simulation", "abort", "analysis", "re-new"
```

```
    };
(* デバイスモデル - データモデル[device]に属する *)
LAB`Contents["device model"] = {
    "link", "motor", "rigid", "elastic"
    };
(* システムモデル - データモデル[system]に属する *)
LAB`Contents["system model"] = {
    "dgs-vsm"
    };
(* デバイスモデル内で,接触が起こりうるモデル *)
LAB`Contents["contact model"] = {
    "link", "rigid", "elastic"
    };
(* 接触が起こりうるモデル内で,自身の構成要素同士が接触しないモデル *)
LAB`Contents["single body model"] = {
    "rigid", "elastic"
    };
(* 数値計算手法 *)
LAB`Contents["numerical integration"] = {
    "Euler",
    "Midpoint",
    "Heun",
    "Heun3",
    "Kutta3",
    "RK4",
    "Kutta3/8"
    };
```

[関数定義] LAB`LogStatus
    function : ログビューワーの状態を取得
    argument1 : Null
    return   : 1 → 開いている
                2 → 閉じている

```
(* ログビューワーの状態を取得 *)
LAB`LogStatus[___] := Module[{
    status, ret
    },
    (* ログビューワーのプロセスの状態を取得 *)
    status = ProcessStatus@LAB`LogProcess;
    (* 状態を数値に変換 *)
    ret = Switch[status,
       "Running", 1,(* 実行中 *)
       "Finished", 0,(* 終了済み *)
       _, 0 (* 一度も実行していない *)
     ];
    (* 戻り値 *)
    Return@ret
]
```

[関数定義] LAB`LogOpen
    function : ログビューワーを開く
    argument1 : Null
    return   : Null

```
LAB`LogOpen[___] := Module[{
    path = LAB`NotebookDirectory <> "LogViewer.exe",
    process
    },
    (* 既にオープンしている場合は終了 *)
    If[LAB`LogStatus[] === 1, Return@Null];
```

```
92      (* プログラムを開く - ウィンドウ位置,サイズを指定 *)
93      process = StartProcess[{path, "window", "-1", "-1", "800", "335"}];
94      Pause[0.5];
95      (* グローバル変数に登録 *)
96      LAB`LogProcess = process;
97      (* 項目を設定する *)
98      process = StartProcess[{path, "column", "Level", "Time", "Event"}];
99      While[ProcessStatus@process === "Running"];
100     (* 項目の幅を設定する *)
101     process = StartProcess[{path, "width", "80", "70", "610"}];
102     While[ProcessStatus@process === "Running"];
103     (* 戻り値 *)
104     Return@Null;
105 ]
```

[関数定義] LAB`LogClose
    function : ログビューワーを閉じる
    argument1 : Null
    return   : Null

```
106 LAB`LogClose[___] := Module[{
107     },
108     (* オープンしていない場合は終了 *)
109     If[! LAB`LogStatus[] === 1, Return@Null];
110     (* プロセスを終了する *)
111     KillProcess[LAB`LogProcess];
112     (* 戻り値 *)
113     Return@Null;
114 ]
```

[関数定義] LAB`LogOutput
    function : ログの出力
    argument1 : 1 列目 - ログレベル（通常, 警告, エラー,etc.）
    argument2 : 2 列目 - 日時表示の有無
    argument3 : 3 列目 - イベント文字列（リストで複数指定も可能）
    return   : Null

```
115 LAB`LogOutput[Level_, bTime_, Str__] := Module[{
116     ToAscii, process, output
117     },
118     (* ログを未使用の設定になっている場合は終了 *)
119     If[LAB`LogEnabled === 0, Return@Null];
120     (* オープンしていない場合はオープン *)
121     If[! LAB`LogStatus[] === 1, LAB`LogOpen[]];
122     (* 出力文字列, 複数ある場合は連結 *)
123     output = StringJoin[ToString@# & /@ {Str}];
124     (* UTF16に変換, 特殊文字は自作の変換ルールが必要 *)
125     ToAscii[str_] := StringReplace[
126         ToString[str, CharacterEncoding -> "ASCII"], {
127             "<-"                -> "\\:2190",
128             "\\[UpArrow]"       -> "\\:2191",
129             "->"                -> "\\:2192",
130             "\↓"     -> "\\:2193",
131             "\●" -> "\\:25CF"
132         }];
133     (* プログラムの実行 *)
134     process = StartProcess[{
135         LAB`NotebookDirectory <> "LogViewer.exe",
136         "wolfram",
137         ToAscii@Level,
138         ToAscii@If[bTime === True,
```

```
139             DateString[{"Hour24", ":", "Minute", ":", "Second"}], " "],
140         ToAscii@output
141     }];
142     (* プログラムの処理が終了するまで待機 *)
143     While[ProcessStatus@process === "Running"];
144     (* 戻り値 *)
145     Return@Null;
146 ];
```

[関数定義] LAB`Log/LAB`Warning/LAB`Error/LAB`EmptyLog
function : ログの出力（簡易版）
argument1 : 出力する文字列 - String
return : Null

```
147 LAB`Log[Str__]       := LAB`LogOutput["information", True,  Str];
148 LAB`Warning[Str__]   := LAB`LogOutput["warning",     True,  Str];
149 LAB`Error[Str__]     := LAB`LogOutput["error",       True,  Str];
150 LAB`EmptyLog[Str__] := LAB`LogOutput[" ",           False, Str];
```

[関数定義] LAB`NewNoteContents
function : 渡されたコンテンツを表示するノートブックを新規で開く
argument1 : 表示データ - 文字列データ/ボックスデータ/セルデータ/これらを要素に持つリスト
argument2 : 表示オプション
- 文字列 -> ウィンドウタイトル
- リスト -> ウィンドウサイズ {幅, 高さ, 左端からの距離, 上端からの距離}
return : ノートブックハンドル

```
151 LAB`NewNoteContents[Data_, Option___] := Module[{
152     f, data, title, list, size, margin, handle
153     },
154     (*-------------------------------------------------------------------------*)
155     (* 表示データの型により形式を変換する関数 *)
156     f[x_] := Which[
157         (* 文字列データの場合 - 入力形式に変換 *)
158         StringQ@x, Cell[BoxData[x], "Input"],
159         (* ボックスデータの場合 - 入力形式に変換 *)
160         MatchQ[Head@x, BoxData], Cell[x, "Input"],
161         (* セルデータの場合 - そのまま *)
162         MatchQ[Head@x, Cell], x,
163         (* その他 - そのまま *)
164         True, x
165      ];
166     (*-------------------------------------------------------------------------*)
167     (* 表示データを変換 *)
168     data = If[ListQ@Data, f[#] & /@ Data, f[Data]];
169     (* オプションを取得 - タイトル,ウィンドウサイズ,位置 *)
170     title = FirstCase[{Option, "New Notebook"}, _String];
171     list = FirstCase[{Option, {Automatic}}, _List];
172     If[Length[list] == 1, size = {list[[1]], Automatic}];
173     If[Length[list] > 1, size = {list[[1]], list[[2]]}];
174     margin = {{Automatic, Automatic}, {Automatic, Automatic}};
175     If[Length[list] > 2, margin[[1, 1]] = list[[3]]];(* 左端からの距離 *)
176     If[Length[list] > 3, margin[[2, 2]] = list[[4]]];(* 上端からの距離 *)
177     (*-------------------------------------------------------------------------*)
178     (* ノートブックを開く *)
179     handle = CreateDocument[
180         data,
181         WindowTitle -> title,
182         WindowFrame -> "ModelessDialog",
183         WindowSize -> size,
184         WindowMargins -> margin
```

```
    ];
    (* 戻り値 *)
    Return@handle;
]
```

[関数定義] LAB‘NewNoteSymbol
    function : 渡されたシンボルに関連付けられた定義情報を表示するウィンドウを新規で開く
    argument1 : シンボル（主にハンドルとして用いている変数） - Symbol
    argument2 : 表示オプション
                - 文字列   -> ウィンドウタイトル
                - リスト   -> ウィンドウサイズ {幅, 高さ, 左端からの距離, 上端からの距離}
    return   : ノートブックハンドル

```
LAB‘NewNoteSymbol[Data_, Option___] := Module[{
    title, list, size, margin, handle, code, data
    },
    (*--------------------------------------------------------------------------*)
    (* オプションを取得 - タイトル&ウィンドウサイズ *)
    title = "Definition information associated with the symbol";
    title = FirstCase[{Option, title}, _String];
    list = FirstCase[{Option, {Automatic}}, _List];
    If[Length[list] == 1, size = {list[[1]], Automatic}];
    If[Length[list] > 1, size = {list[[1]], list[[2]]}];
    margin = {{Automatic, Automatic}, {Automatic, Automatic}};
    If[Length[list] > 2, margin[[1, 1]] = list[[3]]];(* 左端からの距離 *)
    If[Length[list] > 3, margin[[2, 2]] = list[[4]]];(* 上端からの距離 *)
    (* ノートブックを開く *)
    handle = CreateNotebook[
        "Default",
        WindowTitle   -> title,
        WindowFrame   -> "ModelessDialog",
        WindowSize    -> size,
        WindowMargins -> margin
    ];
    (*--------------------------------------------------------------------------*)
    (* 実行するコードの文字列を定義 *)
    code = "Save[\"stdout\"," <> ToString[Data] <> "];";
    (* ノートブックに展開 - 展開したセルを選択 *)
    NotebookWrite[
        handle,
        Cell[BoxData[code], "Input", Background -> LightYellow],
        All];
    (* セルを評価 *)
    SelectionEvaluate[handle];
    (* ノートの先頭を選択 *)
    SelectionMove[handle, Before, Notebook];
    (*--------------------------------------------------------------------------*)
    (* 戻り値 *)
    Return@handle;
]
```

[関数定義] LAB‘LocalFilePath
    function : 指定したファイルの拡張子を補完し, ファイルが存在するかをチェックする
    argument1 : ファイル名（拡張子は任意）   - String
    argument2 : 拡張子（. は任意）   - String
    argument3 : オプション
                - find   : 指定ファイルが存在する場合は 1 を返す
                - notfind : 指定ファイルが存在しない場合は 1 を返す
    return   : 成功 : ファイルのパス / オプションによる数値
                失敗 : 0

```
LAB`LocalFilePath[File_, Ext_, Option___] := Module[{
    file, path, ext,
    option = ToLowerCase@Flatten@{Option}
    },
    (*--------------------------------------------------------------------------*)
    (* 引数の型が不正であれば0を返す *)
    If[! StringQ@File, Return@0];
    If[! StringQ@Ext, Return@0];
    (*--------------------------------------------------------------------------*)
    (* 拡張子を取得 *)
    ext = Which[
        (* 未指定である場合 *)
        Ext === "", "",
        (* ファイル名に含む拡張子と指定した拡張子が同じである場合 *)
        FileExtension[File] === Ext, "",
        (* 拡張子の先頭に"."を含む場合 *)
        StringMatchQ[Ext, ".*"], Ext,
        (* いずれにも当てはまらない場合 *)
        True, "." <> Ext
    ];
    (* ファイル名を取得 *)
    file = File <> ext;
    (* パスを取得, ドライブが指定されていない場合は相対パスとする *)
    path = If[StringMatchQ[file, "*:*"],
        file,
        LAB`NotebookDirectory <> file
    ];
    (*--------------------------------------------------------------------------*)
    (* オプション : 指定ファイルが存在する場合は1を返す *)
    If[MemberQ[option, "find"], If[FileExistsQ@path, Return@1]];
    (* オプション : 指定ファイルが存在しない場合は1を返す *)
    If[MemberQ[option, "notfind"], If[! FileExistsQ@path, Return@1]];
    (*--------------------------------------------------------------------------*)
    (* 戻り値-パスを返す *)
    Return@path;
]
```

[関数定義] LAB`CheckFileName
    function : ファイル名が正しいか確認する
    argument1 : チェックするファイル名
    return   : 1 : 問題がない
                 0 : 文字列でない
                -1 : 禁止文字を含む
                -2 : 使用禁止の単語である
                -3 : 実体のない文字列 ("", " ") である

```
LAB`CheckFileName[Value_] := Module[{
    banned
    },
    (* 文字列でない - 戻り値:0 *)
    If[! StringQ@Value, Return@0];
    (* 名前に禁止文字を含む - 戻り値:-1 *)
    banned = {"\\", "/", ":", "*", "?", "\"", "<", ">", "|"};
    If[! StringFreeQ[Value, banned], Return[-1]];
    (* 名前が禁止単語である - 戻り値:-2 *)
    banned = {};
    If[! AnyTrue[(Value == #) & /@ banned, TrueQ], Return[-2]];
    (* 名前が空白文字である - 戻り値:-3 *)
    If[StringDelete[Value, Whitespace] == "", Return[-3]];
    (* 全てのチェックにパス - 戻り値:1 *)
    Return@1;
]
```

```
[関数定義] LAB`DataConvert
    function  : 指定されたオプションに従いデータの形式を変換
    argument1 : 変換元データ
    argument2 : 変換オプション - 複数指定可
                - int       : 整数に変換 (Listable)       "2.5" / 2.5 -> 2 , 2.51 -> 3
                - float     : 小数に変換 (Listable)       "3"   / 3   -> 3.
                - mm        : 10^(-3) 倍する (Listable)
                - g         : 10^(-3) 倍する (Listable)
                - degree    : 180/π 倍する (Listable)
                - radian    : π/180 倍する (Listable)
                - micro     : 10^(-6) 倍する (Listable)
                - list      : 文字列リストを正式なリストに変換  "{1,2,3}" -> {"1","2","3"}
                - str       : 文字列化                          {1,2,3} -> "{1,2,3}"
                - string    : 文字列化                          {1,2,3} -> "{1,2,3}"
                - strd      : 文字列化 & 空白全削除             {1,2,3} -> "{1,2,3}"
                - stre      : リスト要素を文字列化             {1,{2,3}} -> {"1",{"2","3"}}
                - strjoin   : リスト要素を文字列化して結合     {1,{2,3}} -> "123"
                - sriffle:,     : リスト要素間に次の文字を挿入し，文字列化して結合 [,]
                - sriffle:n     : リスト要素間に次の文字を挿入し，文字列化して結合 [\n]
                - sriffle:nt    : リスト要素間に次の文字を挿入し，文字列化して結合 [\n\t]
                - sriffle:,nt   : リスト要素間に次の文字を挿入し，文字列化して結合 [,\n\t]
    return    : 変換されたデータ
```

```
278 LAB`DataConvert[Data_, Option__] := Module[{
279     data = Data, rule, f, s,
280     option = ToLowerCase@Flatten@{Option}
281     },
282     (*-------------------------------------------------------------------------*)
283     (* リストの要素のみを文字列化する関数を定義 *)
284     f[x_] := If[ListQ@x, f[#] & /@ x,
285         If[StringQ@x, x, ToString[x, InputForm]]];
286     (* リストごと文字列化する関数を定義 *)
287     s[x_] := If[ListQ@x,
288         StringJoin["{", Riffle[s[#] & /@ x, ","], "}"],
289         If[StringQ@x, x, ToString[x, InputForm]]];
290     (*-------------------------------------------------------------------------*)
291     Do[
292         Switch[option[[n]],
293         (* 整数/小数に変換 *)
294         "int",          data = Round@ToExpression@data,
295         "float",        data = 1.0*ToExpression@data,
296         (* 桁を調節 *)
297         "mm",           data = data*10^(-3),
298         "g",            data = data*10^(-3),
299         "degree",       data = data*Degree,
300         "radian",       data = data/Degree,
301         "micro",        data = data*10^(-6),
302         (* リスト文字列を正式なリストに変換 "{1,2,3}" -> {"1","2","3"} *)
303         "list", If[StringQ@data,
304             rule = {"{" -> "{\"", "}" -> "\"}", "," -> "\",\""};
305             data = StringTrim@ToExpression@StringReplace[data, rule]],
306         (* 文字列化 *)
307         "str",          data = s[data],
308         "string",       data = s[data],
309         (* 文字列化 : 空白全削除 *)
310         "strd",         data = StringDelete[s[data], Whitespace],
311         (* 文字列化 : リスト要素のみ {1,{2,3}} -> {"1",{"2","3"}} *)
312         "stre",         data = f@data,
313         (* 文字列化 : 全リスト要素の結合 {1,{2,3}} -> "123" *)
314         "strjoin",      data = StringJoin@f@data,
315         (* 文字列化 : リスト要素間に特定の文字列を挿入し,結合 *)
316         "sriffle:,",    data = StringJoin@f@Riffle[data, ","],
317         "sriffle:n",    data = StringJoin@f@Riffle[data, "\n"],
```

```
318         "sriffle:nt",  data = StringJoin@f@Riffle[data, "\n\t"],
319         "sriffle:,nt", data = StringJoin@f@Riffle[data, ",\n\t"],
320         (* 色つき文字に変換 *)
321         "red", data =
322             "\*StyleBox[\"" <> s[data] <> "\",FontColor->RGBColor[1, 0, 0]]",
323         "blue", data =
324             "\*StyleBox[\"" <> s[data] <> "\",FontColor->RGBColor[0, 0, 1]]",
325         (* 何れにも該当しない場合 *)
326         _, Null
327         ], {n, Length@option}
328     ];
329     (*-------------------------------------------------------------------------*)
330     (* 戻り値 *)
331     Return@data;
332 ]
```

[関数定義] LAB‘Digit
function : 小数値を指定桁で丸める
argument1 : 対象の小数値
argument2 : 小数点以下の桁数
return : 丸めた数値

```
333 LAB‘Digit[x_, digit_] := N[Round[x*10^digit]/10^digit];
```

[関数定義] LAB‘CheckValue
function : 値が指定した型と一致するか確認する，必要に応じて修正
argument1 : 値
argument2 : 値の型（文字列またはヘッダーを指定）
- int : 整数値
- float : 小数値
- bool : True/False
- string/str : 文字列
- replacement : 強制で規定値

argument3 : 規定値
return : 値

```
334 LAB‘CheckValue[Value_, Type_, Pre_] := Module[{
335     type = Type
336     },
337     (*-------------------------------------------------------------------------*)
338     (* 型が文字列で指定されている場合 *)
339     If[StringQ@type,
340     Switch[ToLowerCase@type,
341         (* int : 整数値 - 数値以外->規定値, 小数->変換, 整数->そのまま *)
342         "int",
343         If[! NumberQ@Value, Return@Pre];
344         If[! IntegerQ@Value, Return@Round@Value];
345         Return@Value,
346         (* float : 小数値 - 数値以外->規定値, 整数->変換, 小数->そのまま *)
347         "float",
348         If[! NumberQ@Value, Return@Pre];
349         If[IntegerQ@Value, Return[1.0*Value]];
350         Return@Value,
351         (* bool : True/False - T/F以外->規定値, T/F->そのまま *)
352         "bool",
353         If[! BooleanQ@Value, Return@Pre];
354         Return@Value,
355         (* replacement : 強制的に規定値に修正 *)
356         "replacement", Return@Pre,
357         (* 型をヘッダーに置き換える = 処理をヘッダー指定に回す *)
358         "string", type = String,
```

```
359         "str", type = String,
360         (* 該当なし *)
361         _, Return@Null
362     ]];
363     (*-------------------------------------------------------------------------*)
364     (* 型がヘッダーで指定されている場合 *)
365     If[Head[Value] === type,
366         (* 一致する->そのまま *)
367         Return@Value,
368         (* 一致しない->修正値 *)
369         Return@Pre
370     ];
371     (*-------------------------------------------------------------------------*)
372     Return@Null;
373 ]
```

[関数定義] LAB‘IniToList
function : INI ファイルのデータ読込
argument1 : ファイル名（拡張子は任意） - String
return : 成功 : リストデータ - {{グループ名, アイテム名, 値},{グループ名, アイテム名, 値},...}
失敗 : 0

[INI] ファイルの記述
[***] -> グループ名
*** = *** &オプション 1 &オプション 2 ; コメント
-> アイテム名 & 値 ※ オプションは関数 [LAB‘DataConvert] のもの
;*** -> コメント

```
374 LAB‘IniToList[File_] := Module[{
375     path, data, line, sec, key, value, option, type, list = {}
376     },
377     (*-------------------------------------------------------------------------*)
378     (* ファイルの絶対パスを取得 - 失敗した場合は0を返す *)
379     path = LAB‘LocalFilePath[File, "ini", "notfind"];
380     If[! StringQ@path, Return@0];
381     (*-------------------------------------------------------------------------*)
382     (* INIファイルを読み込みリスト化する - 1行=1要素 *)
383     data = Import[path, "List"];
384     (* デフォルトのグループ名 *)
385     sec = "";
386     (*-------------------------------------------------------------------------*)
387     (* 正式なリストの生成 : 1行ごとに処理 *)
388     Do[
389         (* n行目のデータ *)
390         line = data[[n]];
391         Which[
392             (* グループ名の行 : グループ名を取得 *)
393             StringMatchQ[line, "[*]"], sec = StringTake[line, {2, -2}];,
394             (* コメント行 : 何もしない *)
395             StringMatchQ[line, ";*"], Null;,
396             (* アイテム名&値の行 *)
397             StringMatchQ[line, "*=*"],
398                 (* アイテム名とアイテム値に分離 *)
399                 {key, value} = StringSplit[line, "=", 2];
400                 (* アイテム値に含まれるコメント&オプションを除外 *)
401                 value = StringSplit[value, ";" | "&"][[1]];
402                 (* アイテム名&値の前後の余白を除外 *)
403                 {key, value} = StringTrim@{key, value};
404                 (* 行内にオプションが指定してある場合は取得し反映させる *)
405                 If[StringMatchQ[line, "*&*"],
406                 type = StringCases[line, RegularExpression["&([A-z]+)"] -> "$1"];
407                 value = LAB‘DataConvert[value, type]];
```

```
408             (* グループ名&アイテム値を小文字化し,リストに追加 *)
409             AppendTo[list, {ToLowerCase@sec, ToLowerCase@key, value}];
410       (*  *)
411       ], {n, Length@data}
412   ];
413   (*--------------------------------------------------------------------*)
414   (* 戻り値 *)
415   Return@list;
416 ]
```

[関数定義] LAB‘ListToHandle
function : リストデータをハンドルに関連付ける
- Handle[グループ名][アイテム名] = アイテム値
- Handle["Group"] = グループ名のリスト
argument1 : ハンドル
argument2 : リストデータ - {{グループ名, アイテム名, 値},{グループ名, アイテム名, 値},...}
argument3 : グループ名 (任意, 指定しなければ全て)
return : 成功 : 1
失敗 : 0

```
417 LAB‘ListToHandle[Handle_, Data_, Group___] := Module[{
418   groups
419   },
420   (* データの型が不正であれば0を返す *)
421   If[! ListQ@Data, Return@0];
422   (* グループリストの取得:指定がなければリストデータから全グループを取得 *)
423   groups = Flatten@{Group};
424   groups = Select[groups, StringQ];
425   If[Length@groups == 0, groups = Union@Data[[All, 1]]];
426   (* データを関連付け *)
427   Do[If[MemberQ[groups, Data[[n, 1]]],
428     Handle[Data[[n, 1]]][Data[[n, 2]]] = Data[[n, 3]]
429   ], {n, Length@Data}];
430   (* グループ名のリストを作成 : 重複分を削除 *)
431   If[! ListQ@Handle["Group"], Handle["Group"] = {}];
432   Handle["Group"] = Union@Join[Handle["Group"], groups];
433   (* 戻り値 *)
434   Return@1;
435 ]
```

[関数定義] LAB‘IniToHandle
function : INI ファイルのデータを読込み, ハンドルに関連付ける
- Handle[グループ名][アイテム名] = アイテム値
- Handle["Group"] = グループ名のリスト
- Handle["Source"] = Ini ファイルのパス
argument1 : ハンドル
argument2 : ファイル名 (拡張子は任意) - String
argument3 : グループ名 (任意, 指定しなければ全て)
return : 成功 : 1
失敗 : 0

```
436 LAB‘IniToHandle[Handle_, File_, Group___] := Module[{
437   list, ret
438   },
439   (* Iniファイルからリストデータを取得 *)
440   list = LAB‘IniToList[File];
441   If[list === 0, Return@0];
442   (* リストデータをハンドルデータに関連付け *)
443   ret = LAB‘ListToHandle[Handle, list, Group];
444   If[ret === 0, Return@0];
445   (* Iniファイルのパスを登録 *)
```

```
    Handle["Source"] = LAB`LocalFilePath[File, "ini"];
    (* 戻り値 *)
    Return@1;
]
```

[関数定義] LAB`GetHandle
function : 指定モデル/グループ名のハンドルを取得
argument1 : データモデル （リスト, All, Null, Automatic）
argument2 : グループ名 （リスト, All, Null）
return : 成功 : ハンドルリスト/ハンドル（グループ名指定でハンドルが存在しない場合は Null）
失敗 : 0

```
LAB`GetHandle[Model_, Name_] := Module[{
    getHndles, selectHandle, hList = {}, ret
    },
    (*=============================================================================*)
    (* データモデルが登録された指定文字から始まるシンボル *)
    getHndles[str_] := Module[{tmp},
        tmp = Select[Names[str <> "*"], Head@ToExpression@# === Symbol &];
        tmp = ToExpression@tmp;
        tmp = Select[tmp, #["data model"] === str &];
        Union@tmp
    ];
    (* リストから指定したグループ名のハンドル/Nullを選択する関数 *)
    selectHandle[list_, name_] := Module[{tmp},
        tmp = Select[list, #["name"] === name &];
        If[Length@tmp > 0, Return@tmp[[1]], Return@Null]
    ];
    (*=============================================================================*)
    (* 対象となるハンドルリストを作成 *)
    Which[
        (* モデルが単独指定 *)
        StringQ@Model, hList = getHndles[Model],
        (* モデルが複数指定 *)
        ListQ@Model, hList = Flatten[getHndles[#] & /@ Model],
        (* モデルが未指定 *)
        Model === Automatic || Model === Null || Model === All,
        hList = Flatten[getHndles[#] & /@ LAB`Contents["data model"]]
    ];
    (*=============================================================================*)
    (* グループ名での絞り込み *)
    Which[
        (* 名前が単独指定 *)
        StringQ@Name, Return@selectHandle[hList, Name],
        (* 名前複数指定 *)
        ListQ@Name, Return[selectHandle[hList, #] & /@ Name],
        (* 名前未指定 *)
        Name === All || Name === Null,
        Return@hList
    ];
    (*=============================================================================*)
    (* 戻り値 *)
    Return@0;
]
```

[関数定義] LAB`UsingHandle
function : ハンドルを使用中であるかチェック
argument1 : チェックするハンドル
argument2 : オプション（未使用）
return : リアルタイムで使用中の実行用ハンドル [run*] の数

```
LAB`UsingHandle[Handle_, Option___] := Module[{
   ret, rets = {}, hRuns = {}, hConfigs = {}, hlist, str1, str2,
   option = ToLowerCase@Flatten@{Option}
   },
   (*=============================================================================*)
   (* データモデルが未知のモデルである場合終了 *)
   If[! MemberQ[LAB`Contents["data model"],
      Handle["data model"]],
      Return@0];
   (* 全ての計算設定ハンドルを取得:存在しなければ全て未使用と判断 *)
   hlist = LAB`GetHandle["config", All];
   If[Length@hlist == 0, Return@0];
   (*=============================================================================*)
   (* 関連付けられている実行処理用ハンドル[run*]を取得 *)
   Switch[Handle["data model"],
      (* run *)
      "run", hRuns = {Handle},
      (* config *)
      "config", hRuns = {Handle["handle:run"]},
      (* device *)
      "device",
         hConfigs = Select[hlist, MemberQ[#["device list"], Handle] &];
         hRuns = #["handle:run"] & /@ hConfigs,
      (* system *)
      "system",
         hConfigs = Select[hlist, MemberQ[#["system list"], Handle] &];
         hRuns = #["handle:run"] & /@ hConfigs
   ];
   (*=============================================================================*)
   (* ハンドルの使用状況を取得 *)
   rets = Switch[#["state"],
      "set", 2,
      "standby", 0,
      "re-new", 0,
      "simulation", 3,
      "abort", 3,
      "analysis", 4,
      _, 0] & /@ hRuns;
   (*=============================================================================*)
   (* 戻り値 *)
   Return@Length@Select[rets, # > 0 &];
]
```

[関数定義] LAB`SortHandle
function : データモデルを基にハンドルリストを並び替える
argument1 : ハンドルリスト
return : 並び替えたハンドルリスト

```
LAB`SortHandle[HList_, ___] := Module[{
   f
   },
   (* 並びえの順位を返す関数 *)
   f[x_] := Which[
      x["data model"] === "device" && x["device model"] === "link",      1,
      x["data model"] === "device" && x["device model"] === "motor",     2,
      x["data model"] === "device" && x["device model"] === "rigid",     3,
      x["data model"] === "device" && x["device model"] === "elastic",   4,
      x["data model"] === "device",  50,
      x["data model"] === "system" && x["system model"] === "dgs-vsm", 51,
      x["data model"] === "system", 100,
      x["data model"] === "config", 101,
      x["data model"] === "run",    102,
```

```
548         True, 200
549     ];
550     (* 戻り値 *)
551     Return@SortBy[HList, f];
552 ]
```

[関数定義] LAB‘SortGroup
function : データモデルを基にハンドルのグループ名リストを並び替える
argument1 : INI ファイルから読み込んだハンドルデータ
return : グループ名リスト

```
553 LAB‘SortGroup[Handle_, ___] := Module[{
554     f
555     },
556     (* 並びえの順位を返す関数 *)
557     f[x_] := Module[{
558         group = Handle[x],
559         model = Handle[x]["data model"]
560         }, Which[
561             model === "device" && group["device model"] === "link",      1,
562             model === "device" && group["device model"] === "motor",     2,
563             model === "device" && group["device model"] === "rigid",     3,
564             model === "device" && group["device model"] === "elastic",   4,
565             model === "device",  50,
566             model === "system" && group["system model"] === "dgs-vsm", 51,
567             model === "system", 100,
568             model === "config", 101,
569             model === "run",    102,
570             True, 200
571         ]];
572     (* 戻り値 *)
573     Return@SortBy[Handle["Group"], f];
574 ]
```

[関数定義] LAB‘NumericalIntegration["Euler"]
function : 数値積分を行う関数を定義する - オイラー法
argument1 : 速度, 座標から加速度を返す関数
argument2 : 刻み時間
return : 定義した関数
- argument : 現在値のリスト - {q,dq}
- return : Δｔ秒後の値のリスト - {q’,dq’}

```
575 LAB‘NumericalIntegration["Euler", Inside_, Step_] := (
576     Return@Compile[{{now, _Real, 2}}, Module[{k1},
577         k1 = Inside[now];
578         (* return *)
579         now + Step*k1
580     ]];
581 )
```

[関数定義] LAB‘NumericalIntegration["Midpoint"]
function : 数値積分を行う関数を定義する - 中点法 - 改良オイラー法
argument1 : 速度, 座標から加速度を返す関数
argument2 : 刻み時間
return : 定義した関数
- argument : 現在値のリスト - {q,dq}
- return : Δｔ秒後の値のリスト - {q’,dq’}

```
582 LAB‘NumericalIntegration["Midpoint", Inside_, Step_] := (
```

```
    Return@Compile[{{now, _Real, 2}}, Module[{k1, k2},
       k1 = Inside[now];
       k2 = Inside[now + Step*k1/2];
       (* return *)
       now + Step*k2
    ]];
)
```

[関数定義] LAB‘NumericalIntegration["Heun"]
    function : 数値積分を行う関数を定義する - ホイン法 - 修正オイラー法
    argument1 : 速度, 座標から加速度を返す関数
    argument2 : 刻み時間
    return   : 定義した関数
                - argument : 現在値のリスト - {q,dq}
                - return   : Δ t 秒後の値のリスト - {q’,dq’}

```
LAB‘NumericalIntegration["Heun", Inside_, Step_] := (
    Return@Compile[{{now, _Real, 2}}, Module[{k1, k2},
       k1 = Inside[now];
       k2 = Inside[now + Step*k1];
       (* return *)
       now + Step*(k1 + k2)/2
    ]];
)
```

[関数定義] LAB‘NumericalIntegration["Heun3"]
    function : 数値積分を行う関数を定義する - ホインの 3 次公式
    argument1 : 速度, 座標から加速度を返す関数
    argument2 : 刻み時間
    return   : 定義した関数
                - argument : 現在値のリスト - {q,dq}
                - return   : Δ t 秒後の値のリスト - {q’,dq’}

```
LAB‘NumericalIntegration["Heun3", Inside_, Step_] := (
    Return@Compile[{{now, _Real, 2}}, Module[{k1, k2, k3},
       k1 = Inside[now];
       k2 = Inside[now + Step*k1/3];
       k3 = Inside[now + Step*k2*2/3];
       (* return *)
       now + Step*(k1 + 3*k3)/4
    ]];
)
```

[関数定義] LAB‘NumericalIntegration["Kutta3"]
    function : 数値積分を行う関数を定義する - クッタの 3 次公式
    argument1 : 速度, 座標から加速度を返す関数
    argument2 : 刻み時間
    return   : 定義した関数
                - argument : 現在値のリスト - {q,dq}
                - return   : Δ t 秒後の値のリスト - {q’,dq’}

```
LAB‘NumericalIntegration["Kutta3", Inside_, Step_] := (
    Return@Compile[{{now, _Real, 2}}, Module[{k1, k2, k3},
       k1 = Inside[now];
       k2 = Inside[now + Step*k1/2];
       k3 = Inside[now - Step*k1 + Step*k2*2];
       (* return *)
       now + Step*(k1 + 4*k2 + k3)/6
    ]];
```

```
)
```

[関数定義] LAB‘NumericalIntegration["RK4"]
    function : 数値積分を行う関数を定義する - 4 次のルンゲクッタ法
    argument1 : 速度, 座標から加速度を返す関数
    argument2 : 刻み時間
    return   : 定義した関数
                - argument : 現在値のリスト - {q,dq}
                - return   : Δ t 秒後の値のリスト - {q',dq'}

```
LAB‘NumericalIntegration["RK4", Inside_, Step_] := (
   Return@Compile[{{now, _Real, 2}}, Module[{k1, k2, k3, k4},
      k1 = Inside[now];
      k2 = Inside[now + Step*k1/2];
      k3 = Inside[now + Step*k2/2];
      k4 = Inside[now + Step*k3];
      (* return *)
      now + Step*(k1 + 2*k2 + 2*k3 + k4)/6
   ]];
)
```

[関数定義] LAB‘NumericalIntegration["Kutta3/8"]
    function : 数値積分を行う関数を定義する - クッタの 3/8 公式
    argument1 : 速度, 座標から加速度を返す関数
    argument2 : 刻み時間
    return   : 定義した関数
                - argument : 現在値のリスト - {q,dq}
                - return   : Δ t 秒後の値のリスト - {q',dq'}

```
LAB‘NumericalIntegration["Kutta3/8", Inside_, Step_] := (
   Return@Compile[{{now, _Real, 2}}, Module[{k1, k2, k3, k4},
      k1 = Inside[now];
      k2 = Inside[now + Step*k1/3];
      k3 = Inside[now - Step*k1/3 + Step*k2];
      k4 = Inside[now + Step*k1 - Step*k2 + Step*k3];
      (* return *)
      now + Step*(k1 + 3*k2 + 3*k3 + k4)/8
   ]];
)
```

[関数定義] LAB‘LagrangeEquation["link"]
    function : ラグランジュの運動方程式より加速度を計算する関数を定義する - 多関節リンク
    argument1 : 必要なアイテム値が保存されている変数群のハンドル
    return   : 定義した関数 : return = f[arg1, arg2, arg3]
                - arg1~3   : 一般化座標, 一般化速度, 一般化力のリスト [List]
                - return   : 一般化加速度のリスト [List]

```
LAB‘LagrangeEquation["link", Handle_] := Module[{
   dof, l, r, m, i, c, d, q0, xy, g, t, Q, Qi, q, qi, dq, ddq,
   absq, unitVector, relativeJ, relativeP, absJ, absP, dJ, dP,
   T, U, V, Mv, L, Eq, Hq, rule
   },
   (* パラメータ読み込み *)
   dof = Round@Handle["dof"];
   l   = Handle["length of link"];
   r   = l*Handle["rate of center"];
   m   = Handle["mass"];
   i   = Handle["moment of inertia"];
   c   = Handle["damping of rotation"];
```

```
    d   = Handle["damping of translation"];
    xy  = {0, 0};
    q0  = 0;
    g   = Handle["g"];
    (* 一般化力・一般化座標 *)
    Q   = Array[Qi, dof];
    q   = Array[qi[t], dof];
    dq  = D[q, t];
    ddq = D[dq, t];
    (* 関節絶対角度 *)
    absq = Accumulate[q] + q0;
    (* リンク長手方向単位ベクトル *)
    unitVector = {Cos[#], Sin[#]} & /@ absq;
    (* 関節相対座標・重心位置相対座標：第一関節は原点からの座標 *)
    relativeJ = Prepend[unitVector*l, xy];
    relativeP = unitVector*r;
    (* 関節絶対座標・重心位置絶対座標 *)
    absJ = Accumulate[relativeJ];
    absP = Drop[absJ, -1] + relativeP;
    (* 関節座標速度・重心座標速度 *)
    dP = D[absP, t];
    (* 運動・ポテンシャル・散逸エネルギ *)
    T = Total[m*dP^2/2 + i*dq^2/2, Infinity];
    U = g*m*absP[[All, 2]] // Total;
    V = Total[d*dP^2/2 + c*dq^2/2, Infinity];
    (* ラグランジアン *)
    L = T - U;
    (* ラグランジュの運動方程式 *)
    Eq = D[D[L, #] & /@ dq, t] - (D[L, #] & /@ q) + (D[V, #] & /@ dq);
    (* 慣性項行列 *)
    Mv = Coefficient[Eq, #] & /@ ddq;
    (* 遠心力・コリオリ力項行列 *)
    Hq = Eq /. Thread[ddq -> 0];
    (* 関数の作成 - 文字列で作成し,式に展開する *)
    rule = Thread[Join[q, dq, Q] -> Join[
        Table[ "q[[" <> ToString[j] <> "]]", {j, dof}],
        Table["dq[[" <> ToString[j] <> "]]", {j, dof}],
        Table[ "Q[[" <> ToString[j] <> "]]", {j, dof}]]];
    Mv = StringJoin[StringSplit[ToString[InputForm[Mv /. rule]], "\""]];
    Hq = StringJoin[StringSplit[ToString[InputForm[Hq /. rule]], "\""]];
    Q  = StringJoin[StringSplit[ToString[InputForm[Q  /. rule]], "\""]];
    (* 戻り値 *)
    Return@ToExpression[
        "Compile[{{q,_Real,1},{dq,_Real,1},{Q,_Real,1}}," <>
        "Inverse[" <> Mv <> "].(" <> Q <> "-" <> Hq <> ")]"
    ];
]
```

[関数定義] LAB‘LagrangeEquation["motor"]
    function : ラグランジュの運動方程式より加速度を計算する関数を定義する - モータ
    argument1 : 必要なアイテム値が保存されている変数群のハンドル
    return   : 定義した関数 : return = f[arg1, arg2, arg3]
                - arg1~3   : 一般化座標, 一般化速度, 一般化力のリスト [List]
                - return   : 一般化加速度のリスト [List]

```
LAB‘LagrangeEquation["motor", Handle_] := Module[{
    i, c, t, Q, q, q1, dq, ddq, T, U, V, L, Eq, Mv, Hq, rule
    },
    (* パラメータ読み込み *)
    i = Handle["moment of inertia"];
    c = Handle["damping of rotation"];
    (* 一般化力・一般化座標 *)
```

```
    q = q1[t]; dq = D[q, t]; ddq = D[dq, t];
    (* 運動・ポテンシャル・散逸エネルギ *)
    T = i*dq^2/2; U = 0; V = c*dq^2/2;
    (* ラグランジアン *)
    L = T - U;
    (* ラグランジュの運動方程式 *)
    Eq = D[D[L, dq], t] - D[L, q] + D[V, dq];
    (* 慣性項 *)
    Mv = Coefficient[Eq, ddq];
    (* 遠心力・コリオリ力項 *)
    Hq = Eq /. ddq -> 0;
    (* 関数の作成 - 文字列で作成し,式に展開する *)
    rule = Thread[{q, dq, Q} -> {"q[[1]]", "dq[[1]]", "Q[[1]]"}];
    Mv = StringJoin[StringSplit[ToString[InputForm[Mv /. rule]], "\""]];
    Hq = StringJoin[StringSplit[ToString[InputForm[Hq /. rule]], "\""]];
    Q  = StringJoin[StringSplit[ToString[InputForm[Q  /. rule]], "\""]];
    (* 戻り値 *)
    Return@ToExpression[
       "Compile[{{q,_Real,1},{dq,_Real,1},{Q,_Real,1}}," <>
       "{((" <> Q <> ")-(" <> Hq <> "))/(" <> Mv <> ")}]"
    ];
]
```

[関数定義] LAB‘LagrangeEquation["rigid"]
    function : ラグランジュの運動方程式より加速度を計算する関数を定義する - 剛体
    argument1 : 必要なアイテム値が保存されている変数群のハンドル
    return   : 定義した関数 : return = f[arg1, arg2, arg3]
                - arg1~3   : 一般化座標, 一般化速度, 一般化力のリスト [List]
                - return   : 一般化加速度のリスト [List]

```
LAB‘LagrangeEquation["rigid", Handle_] := Module[{
    m, i, c, d, t, Q, Qx, Qy, Qθ, q, qx, qy, qθ, dq, ddq,
    T, U, V, Mv, L, Eq, Hq, g, rule
    },
    (* パラメータ読み込み *)
    m = Handle["mass"];
    i = Handle["moment of inertia"];
    c = Handle["damping of rotation"];
    d = Handle["damping of translation"];
    (* 一般化力・一般化座標 *)
    Q   = {Qx, Qy, Qθ};
    q   = {qx[t], qy[t], qθ[t]};
    dq  = D[q, t];
    ddq = D[dq, t];
    (* 運動・ポテンシャル・散逸エネルギ *)
    T = Total[m*dq[[;; 2]]^2/2 + i*dq[[3]]^2/2, Infinity];
    U = 0;
    V = Total[d*dq[[;; 2]]^2/2 + c*dq[[3]]^2/2, Infinity];
    (* ラグランジアン *)
    L = T - U;
    (* ラグランジュの運動方程式 *)
    Eq = D[D[L, #] & /@ dq, t] - (D[L, #] & /@ q) + (D[V, #] & /@ dq);
    (* 慣性項行列 *)
    Mv = Coefficient[Eq, #] & /@ ddq;
    (* 遠心力・コリオリ力項行列 *)
    Hq = Eq /. Thread[ddq -> 0];
    (* 関数の作成 - 文字列で作成し,式に展開する *)
    rule = Thread[Join[q, dq, Q] -> Join[
       Table[ "q[[" <> ToString[j] <> "]]", {j, 3}],
       Table["dq[[" <> ToString[j] <> "]]", {j, 3}],
       Table[ "Q[[" <> ToString[j] <> "]]", {j, 3}]]];
    Mv = StringJoin[StringSplit[ToString[InputForm[Mv /. rule]], "\""]];
```

```
    Hq = StringJoin[StringSplit[ToString[InputForm[Hq /. rule]], "\""]];
    Q  = StringJoin[StringSplit[ToString[InputForm[Q  /. rule]], "\""]];
    (* 戻り値 *)
    Return@ToExpression[
        "Compile[{{q,_Real,1},{dq,_Real,1},{Q,_Real,1}}," <>
        "Inverse[" <> Mv <> "].(" <> Q <> "-" <> Hq <> ")]"
    ];
]
```

[関数定義] LAB‘RectanglePoint
function : 幅, 高さから長方形座標を計算
argument1 : 幅
argument2 : 高さ
argument3 : 基準点 - 中心を原点とした座標指定{x,y} or 数値指定
- 0:中心, 1:左下, 2:左上, 3:右上, 4:右下,
- 1~5 の小数:各辺上の相当する場所 (例:2.3->上辺を 3:7 で分割した位置)
argument4 : 角度 - ラジアン
return : 座標リスト - {{x1,y1},{x2,y2},{x3,y3},{x4,y4}}

```
LAB‘RectanglePoint[{Width_, Height___}, Ref_, Angle___] := Module[{
    list, w = Width, h = Height, ref, angle
    },
    (* 高さの取得 *)
    h = If[{Height} === {}, w, {Height}[[1]]];
    (* 左下=1を基準点とした場合の座標 *)
    list = {{0, 0}, {0, h}, {w, h}, {w, 0}};
    (* 基準座標を取得 *)
    If[VectorQ[Ref], ref = Ref];(* {x,y}指定 *)
    If[NumberQ[Ref] && Ref == 0, ref = {w/2, h/2}];(* 中心 *)
    If[NumberQ[Ref] && 1 <= Ref <= 5, Which[
        Ref == 1, ref = {0, 0},(* 左下の頂点 *)
        Ref == 2, ref = {0, h},(* 左上の頂点 *)
        Ref == 3, ref = {w, h},(* 右上の頂点 *)
        Ref == 4, ref = {w, 0},(* 右下の頂点 *)
        Ref == 5, ref = {0, 0},(* 左下の頂点 *)
        1 < Ref < 2, ref = {0, (Ref - 1)*h},(* 左辺 *)
        2 < Ref < 3, ref = {(Ref - 2)*w, h},(* 上辺 *)
        3 < Ref < 4, ref = {w, h - (Ref - 3)*h},(* 右辺 *)
        4 < Ref < 5, ref = {w - (Ref - 4)*w, 0},(* 下辺 *)
        True, ref = {w/2, h/2}(* 中心 *)
    ]];
    (* 基準座標系にずらす *)
    list = (# - ref) & /@ list;
    (* 角度を取得,回転させる *)
    If[! {Angle} === {},
        angle = {Angle}[[1]];
        list = RotationMatrix[angle].# & /@ list;
    ];
    (* 戻り値 *)
    Return@list;
]
```

[関数定義] LAB‘PolygonAngle
function : ポリゴン座標から各辺の角度を計算
argument1 : ポリゴン座標リスト - 複数ポリゴンを指定可
return : リスト - ポリゴン各辺の角度

```
LAB‘PolygonAngle[Polygon_] := Module[{
    f, list
    },
    (* ポリゴンに対して各辺の角度を返す関数を作成 *)
```

```
    f[x_] := Module[{polygon = Append[x, x[[1]]]},
        Table[ArcTan[#[[1]], #[[2]]] &@(polygon[[n + 1]] - polygon[[n]])
            , {n, Length@x}]
    ];
    (* 対称のポリゴンが一つか複数かで分岐 - ネストの深さで判定 *)
    Switch[Length@Dimensions@Polygon,
        2, list = f[#] &@Polygon,(* 一つの場合 *)
        3, list = f[#] & /@ Polygon(* 複数の場合 *)
    ];
    (* 戻り値 *)
    Return@list;
]
```

[関数定義] LAB‘PolygonLength
function : ポリゴン座標から各辺の長さを計算
argument1 : ポリゴン座標リスト - 複数ポリゴンを指定可
return : リスト - ポリゴン各辺の長さ

```
LAB‘PolygonLength[Polygon_] := Module[{
    f, list
    },
    (* ポリゴンに対して各辺の長さを返す関数を作成 *)
    f[x_] := Module[{polygon = Append[x, x[[1]]]},
        Table[Norm@(polygon[[n + 1]] - polygon[[n]]), {n, Length@x}]
    ];
    (* 対称のポリゴンが一つか複数かで分岐 - ネストの深さで判定 *)
    Switch[Length@Dimensions@Polygon,
        2, list = f[#] &@Polygon,(* 一つの場合 *)
        3, list = f[#] & /@ Polygon(* 複数の場合 *)
    ];
    (* 戻り値 *)
    Return@list;
]
```

[関数定義] LAB‘Autocomplete["All"]
function : ハンドルに登録されている全グループのパラメータを自動補完
argument1 : 全データのハンドル / 文字列の場合は INI ファイルから読み込む
return : 登録に成功したグループ名リスト

```
LAB‘Autocomplete["All", Handle_, ___] := Module[{
    ret = {}, handle, groups, tmp
    },
    (*=========================================================================*)
    (* ハンドルを取得 *)
    Which[
        (* ハンドル指定 *)
        Head@Handle === Symbol, handle = Handle,
        (* INIファイルパス指定 *)
        StringQ@Handle, LAB‘IniToHandle[handle, Handle];
    ];
    (* ハンドルの取得に失敗した場合は0を返す *)
    If[! Head@handle === Symbol, Return@0];

    (*=========================================================================*)
    (* グループ名リストを取得 : デバイス->システム->計算設定 の順にソート *)
    groups = handle // LAB‘SortGroup;
    (* 各グループの登録 : 成功したグループ名を戻り値リストに追加 *)
    Do[
        tmp = LAB‘Autocomplete["Auto", handle, groups[[n]]];
        If[! tmp === 0, AppendTo[ret, groups[[n]]]];
        , {n, Length@groups}];
```

```
    (* 戻り値 *)
    Return@ret;
]
```

[関数定義] LAB`LAB`Autocomplete["Auto"]
    function : パラメータを自動補完，必要なパラメータが登録されているかをチェック
    argument1 : 全データのハンドル / 文字列の場合は INI ファイルから読み込む
    argument2 : グループ名　- String
    return   : 成功 : グループの独立ハンドル
                失敗 : 0

```
LAB`Autocomplete["Auto", Handle_, Group_, ___] := Module[{
    handle, datamodel, contentmodel, hContent, ret
    },
    (*=========================================================================*)
    (* ハンドルを取得 *)
    Which[
        (* ハンドル指定 *)
        Head@Handle === Symbol, handle = Handle,
        (* INIファイルパス指定 *)
        StringQ@Handle, LAB`IniToHandle[handle, Handle];
    ];
    (* ハンドルの取得に失敗した場合は0を返す *)
    If[! Head@handle === Symbol, Return@0];

    (*=========================================================================*)
    (* データモデルを取得 - 失敗した場合は0を返す *)
    datamodel = handle[Group]["data model"];
    If[! MemberQ[LAB`Contents["data model"], datamodel], Return@0];
    (* コンテンツのモデルを取得 - 失敗した場合は0を返す *)
    contentmodel = Switch[datamodel,
        "device", handle[Group]["device model"],
        "system", handle[Group]["system model"],
        "config", "config",
        _, Return@0];
    (* コンテンツの独立ハンドルを作成 - 既に登録済みの場合はそのまま *)
    hContent = LAB`GetHandle[All, Group];
    If[hContent === Null, hContent = Unique[datamodel]];
    (* 開始ログ *)
    LAB`Log["● ", Group,
        " : [", contentmodel, "]の登録を行います : ", hContent];
    (* ハンドルを使用中の場合は処理を停止 *)
    If[LAB`UsingHandle[hContent] > 0,
        LAB`Warning["↓ ハンドルを使用中のため処理を中断します"]; Return@0];

    (*=========================================================================*)
    (* モデルに応じた関数を実行 *)
    ret = LAB`Autocomplete[contentmodel,
        handle, hContent, Group, Automatic];
    (* 成功した場合 - ハンドルのソースを登録 *)
    If[ret === hContent,
        hContent["Source"] = If[StringQ@#, #, "---"] &@handle["Source"];
        Return@hContent;
    ];
    (* 失敗した場合 *)
    Return@0;
]
```

[関数定義] LAB`Autocomplete["Modification"]
    function : 指定アイテムの値が正しいか確認する，必要に応じて修正/規定値の使用
                LAB`Autocomplete["Registration"] で呼び出される

```
argument1 : Habdle[アイテム名]=アイテム値 となるハンドル
argument2 : アイテム名
argument3 : 値の型 ※ 関数 [CheckValue] で使用可能な型 (int,float,str,replacement)
            {***,num}   - list[num]:* に変換
            list:***    - いずれかの型のリスト (要素数は自動判別)
            list[*]:*** - いずれかの型のリスト (要素数指定)
argument4 : 規定値
return    :  1 : 問題がない場合
             0 : 修正を行った場合
            -1 : 型の指定が不正である場合 - 値に Null が代入された場合
            -2 : 規定値の要素数が不正である場合
```

```
LAB`Autocomplete["Modification", Handle_, Item_, Type_, Pre_] := Module[{
   compare, type = Type, pre = Pre, new, elements = 0, tmp,
   old = Handle[Item]
   },
   (*=============================================================================*)
   (* 値を比較する関数 *)
   compare[new_, old_] := Module[{bool},
   (* 一致しているかの評価 *)
   bool = (TrueQ[new == old] || new === old);
   (* 戻り値  1:一致, -1:new=Null(Typeが不正), 0:不一致(正常に修正) *)
   Which[bool === True, 1, new === Null, -1, True, 0]];
   (*=============================================================================*)
   (* 型が{リスト}である場合は修正 *)
   If[ListQ@type,
      type = "list[" <> ToString@type[[2]] <> "]:" <> ToString@type[[1]]];
   (* 型を小文字に修正 *)
   type = ToLowerCase@type;
   (*=============================================================================*)
   (* 型がヘッダーで指定されている -> そのまま関数[CheckValue]を呼び出す *)
   If[! StringQ@type,
      Handle[Item] = new = LAB`CheckValue[old, type, pre];
      Return@compare[new, old];
   ];
   (* 型が文字列で"replacement"である -> 強制置換 *)
   If[type === "replacement",
      Handle[Item] = Pre;
      Return@1;
   ];
   (* 型が文字列で"list"を含まない -> そのまま関数[CheckValue]を呼び出す *)
   If[! StringMatchQ[type, "*list*"],
      Handle[Item] = new = LAB`CheckValue[old, type, pre];
      Return@compare[new, old];
   ];
   (*=============================================================================*)
   (* 対象がリストである場合 -> 型に"list"が含まれる場合 *)
   (* リスト要素数を取得する *)
   (*-----------------------------------------------------------------------------*)
   (* 型の文字列内で "list[***]" のように要素数が指定されている場合 *)
   If[StringContainsQ[type, "list[" ~~ ___ ~~ "]"],
      elements = StringDelete[type, {___ ~~ "list[", "]" ~~ ___}];
      If[StringMatchQ[elements, Characters["0123456789"] ..],
         (* "***"の部分が整数 -> その整数値 *)
         elements = ToExpression@elements,
         (* "***"の部分が文字 -> Handle["***"]の値, 整数でなければ-2を返す *)
         elements = Handle[elements];
         If[! IntegerQ@elements, Return[-2]];
      ]];
   (*-----------------------------------------------------------------------------*)
   (* 型の文字列内で要素数が指定されていない場合 *)
   (*     優先順位 - アイテム値の要素数 -> 修正値の要素数 -> 1 *)
   If[elements == 0, elements = Length@old];
```

```
    If[elements == 0, elements = Length@pre];
    If[elements == 0, elements = 1];
    (*-------------------------------------------------------------------*)
    (* 型からリスト情報を削除する *)
    type = StringDelete[
        type, {"list[" ~~ ___ ~~ "]", "list", ":", Whitespace}];
    (* 規定値がリストでない場合に, 同一要素でリスト化する *)
    If[! ListQ@pre, pre = ConstantArray[pre, elements]];
    (* 規定値の要素数が変換要素数に対し不足している場合, -2を返す *)
    If[Length@pre < elements, Return[-2]];
    (* アイテム値がリストでない場合に, 同一要素でリスト化する *)
    tmp = Handle[Item];
    If[! ListQ@old, tmp = ConstantArray[old, elements]];
    (* アイテム値の要素数が変換要素数に対し不足している場合, Nullで埋める *)
    If[Length@tmp < elements,
        tmp = Join[tmp, ConstantArray[Null, elements - Length@tmp]];
    ];
    (* 関数[CheckValue]を呼び出す *)
    Handle[Item] = new = Table[
        LAB`CheckValue[tmp[[n]], type, pre[[n]]], {n, elements}];
    Return@compare[new, old];
]
```

[関数定義] LAB`Autocomplete["Registration"]
    function : コンテンツのアイテム値を登録する
    argument1 : 全データのハンドル / 文字列の場合は INI ファイルから読み込む
    argument2 : コンテンツのハンドル
    argument3 : グループ名 (デバイス名)   - String
    argument4 : アイテム情報のリスト - {{名前, 型, 規定値}...}
    return   : 成功 : 1
                失敗 : 0

```
LAB`Autocomplete["Registration", Handle_, hContent_, Group_, Items_] :=
Module[{
    name, type, pre, old, ret
    },
    (*===================================================================*)
    (* パラメータの登録 *)
    (hContent[#] = Handle[Group][#]) & /@ Items[[All, 1]];

    (*===================================================================*)
    (* パラメータの修正 *)
    Do[
        (* アイテムの名前,型,規定値を取得 *)
        {name, type, pre} = Items[[n]];
        (* 修正前の値を保存 - old=Handle[Group][name]と定義される *)
        old = hContent[name];
        (* アイテム値を修正する関数の実行 *)
        Switch[
            ret = LAB`Autocomplete["Modification", hContent, name, type, pre],
            (* 問題がない *)
            1, Null,
            (* 修正が行われた *)
            0, LAB`Warning[
                "↓ パラメータ修正 [", name, "] ",
                If[Head@old === Handle[Group], "Null", old],
                " -> ", hContent[name]],
            (* エラーコードが返された *)
            _, LAB`Error[
                "↓ パラメータ修正に失敗 [", name, "] return code=", ret]
        ], {n, Length@Items}
    ];
```

```
1001
1002    (*=========================================================================*)
1003    (* コンテンツのアイテムリストを作成 *)
1004    If[! ListQ@hContent["Items"], hContent["Items"] = {}];
1005    If[! MemberQ[hContent["Items"], #],
1006       AppendTo[hContent["Items"], #]] & /@ Items[[All, 1]];
1007
1008    (*=========================================================================*)
1009    (* アイテムを羅列する文字列を作成 *)
1010    hContent["str:Items"] = LAB`DataConvert[{
1011       hContent, "[\"", #, "\"]=",
1012       Which[
1013          StringQ@#, ToString[#, InputForm],
1014          ListQ@#, StringDelete[ToString[#, InputForm], Whitespace],
1015          True, ToString[#]
1016       ] &@hContent[#],
1017       ";\n"
1018    } & /@ hContent["Items"], "strjoin"];
1019
1020    (*=========================================================================*)
1021    (* 戻り値 *)
1022    Return@1;
1023 ]
```

[関数定義] LAB`Autocomplete["Cells"]
function : パラメータを一括表示するセルリストを作成
argument1 : コンテンツのハンドル
argument2 : セルを構成するデータ - {{表示コード 1, オプション..},{表示コード 2, オプション..}..}
return : セルリスト

```
1024 LAB`Autocomplete["Cells", Cells__] := Module[{
1025    list, code, option, style, bgcolor, flame, flamecolor, font, fontsize
1026    },
1027    (*=========================================================================*)
1028    (* 空のリストを作成 *)
1029    list = {};
1030    (* 各要素ごとの処理 *)
1031    Do[
1032       (* 表示コード,オプションの取得 *)
1033       If[ListQ@{Cells}[[n]],
1034          code = {Cells}[[n, 1]]; option = ToLowerCase@{Cells}[[n, 2 ;;]];,
1035          code = {Cells}[[n]]; option = {};
1036       ];
1037       (* セルのスタイル,デフォルトの背景色&フォントを取得 *)
1038       {style, bgcolor, flame, flamecolor, font, fontsize} = Which[
1039          (* セクション *)
1040          MemberQ[option, "section"], {{"Section", Bold, Italic},
1041             LightRed, {{0, 0}, {0, 1.5}}, Red, "Consolas", 20},
1042          (* サブセクション *)
1043          MemberQ[option, "subsection"], {{"Subsection", Bold},
1044             LightGreen, {{0, 0}, {0, 1.5}}, Green, "Consolas", 14},
1045          (* サブサブセクション *)
1046          MemberQ[option, "subsubsection"], {{"Subsubsection", Bold},
1047             LightBlue, {{0, 0}, {0, 1.5}}, Blue, "Consolas", 14},
1048          (* 入力 = デフォルト *)
1049          True, {{"Input", Bold},
1050             LightYellow, 0, Black, "Courier New", 13}
1051       ];
1052       (* リストに追加 *)
1053       AppendTo[list, Cell[
1054          BoxData[code], style, Background -> bgcolor,
1055          CellFrame -> flame, CellFrameColor -> flamecolor,
```

```
1056            FontFamily -> font, FontSize -> fontsize
1057        ]];
1058    , {n, Length@{Cells}}];
1059    (*=========================================================================*)
1060    (* 戻り値 *)
1061    Return@list;
1062 ]
```

[関数定義] LAB`Autocomplete["link"]
    function : デバイスのパラメータを自動補完 - 多関節リンク
    argument1 : 全データのハンドル
    argument2 : デバイスハンドル
    argument3 : グループ名（デバイス名）   - String
    return   : デバイス用独立ハンドル（device*）

```
1063 LAB`Autocomplete["link", Handle_, hDevice_, Group_, ___] := Module[{
1064    list, mm = N[10^(-3)], g = N[10^(-3)], μ = N[10^(-6)],
1065    dof, l, h, θ, pos, basepolygon,
1066    toStr, rules
1067    },
1068    list = #[[2 ;;, 1 ;; 3]] &@ {
1069       {(* アイテム名 *),          (* 型 *),           (* 基準値 *) },
1070       {"data model",              "replacement",      "device"     },
1071       {"device model",            "replacement",      "link"       },
1072       {"name",                    "replacement",      Group        },
1073       {"dof",                     "int",              2            },
1074       {"length of link",          "list[dof]:float",  50*mm        },
1075       {"rate of center",          "list[dof]:float",  0.5          },
1076       {"basic height",            "list[dof]:float",  30*mm        },
1077       {"mass",                    "list[dof]:float",  100*mm       },
1078       {"moment of inertia",       "list[dof]:float",  50*μ         },
1079       {"damping of rotation",     "list[dof]:float",  0.03         },
1080       {"damping of translation",  "list[dof]:float",  0            },
1081       {"ref pos",                 "list[2]  :float",  {0, 0}*mm    },
1082       {"ref angle",               "float",            0*Degree     },
1083       {"base length",             "float",            100*mm       },
1084       {"base height",             "float",            30*mm        },
1085       {"base angle",              "float",            0*Degree     },
1086       {"g",                       "float",            0            }
1087    };
1088
1089    (*=========================================================================*)
1090    (* アイテム値の登録 *)
1091    LAB`Autocomplete["Registration", Handle, hDevice, Group, list];
1092    (* よく使うパラメータを変数登録 *)
1093    dof = hDevice["dof"];
1094    l   = hDevice["length of link"];
1095    h   = hDevice["basic height"];
1096    θ   = hDevice["ref angle"] + hDevice["base angle"];
1097    pos = hDevice["ref pos"];
1098
1099    (*=========================================================================*)
1100    (* ポリゴン,ポイントの取得:ベーシック *)
1101    hDevice["polygon"] =
1102       Table[LAB`RectanglePoint[{l[[n]], h[[n]]}, 1.5], {n, dof}];
1103    hDevice["polygon:j"]    = Range@dof;
1104    hDevice["line enable"]  = Table[{0, 1, 0, 1}, {n, dof}];
1105    hDevice["point"]        = Table[{0, 0}, {n, dof}];
1106    hDevice["point:j"]      = Range@dof;
1107    hDevice["point enable"] = Table[If[n == 1, 1, 0], {n, dof}];
1108    hDevice["point radius"] = Table[h[[n]]/2, {n, dof}];
1109    (* ポリゴン,ポイントの取得:ベース *)
```

```
    basepolygon = (pos + #) & /@ LAB`RectanglePoint[
       {hDevice["base length"], hDevice["base height"]}, 3.5, θ];
    AppendTo[hDevice["polygon"], basepolygon];
    AppendTo[hDevice["polygon:j"], 0];
    AppendTo[hDevice["line enable"], {0, 1, 0, 1}];
    AppendTo[hDevice["point"], #] & /@ basepolygon;
    AppendTo[hDevice["point:j"], #] & /@ {0, 0, 0, 0};
    AppendTo[hDevice["point enable"], #] & /@ {0, 0, 0, 0};
    AppendTo[hDevice["point radius"], #] & /@ {0, 0, 0, 0};
    (* ポリゴン,ポイントの取得:チップ *)
    AppendTo[hDevice["point"], {l[[-1]], 0}];
    AppendTo[hDevice["point:j"], dof];
    AppendTo[hDevice["point enable"], 1];
    AppendTo[hDevice["point radius"], h[[-1]]/2];
    (* ライン情報を取得 *)
    hDevice["line angle"]  = LAB`PolygonAngle@hDevice["polygon"];
    hDevice["line length"] = LAB`PolygonLength@hDevice["polygon"];
    (* 関節ラインの補足 *)
    hDevice["base j0"] =
       {pos + RotationMatrix[θ].{-hDevice["base length"], 0}};
    (* パラメータ補完:初期値 *)
    hDevice["q0"]  = ConstantArray[0, dof];
    hDevice["dq0"] = ConstantArray[0, dof];
    (* パラメータ補完:制御対象リスト *)
    hDevice["controls"] = {};

    (*=========================================================================*)
    (* 運動方程式の定義 *)
    hDevice["eq:ddq"] = LAB`LagrangeEquation["link", hDevice];
    LAB`Log["↓ ラグランジュ法による運動方程式を登録しました"];
    (* 初期値を一括セットする関数 *)
    hDevice["eq:preset"][run_] := (
       run[hDevice, "q0"]  = hDevice["q0"];
       run[hDevice, "dq0"] = hDevice["dq0"];
    );

    (*=========================================================================*)
    (* 補助関数,データの生成 *)
    toStr[x_] := LAB`DataConvert[hDevice[x], "strd"];
    rules = {
       "$handle"   -> ToString@hDevice,
       "$dof"      -> toStr["dof"],
       "$angle"    -> toStr["ref angle"],
       "$pos"      -> toStr["ref pos"],
       "$j0"       -> toStr["base j0"],
       "$length"   -> toStr["length of link"],
       "$polygon"  -> toStr["polygon"],
       "$jPolygon" -> toStr["polygon:j"],
       "$nPolygon" -> ToString@Length@hDevice["polygon"],
       "$point"    -> toStr["point"],
       "$jPoint"   -> toStr["point:j"],
       "$nPoint"   -> ToString@Length@hDevice["point"],
       "$radius"   -> toStr["point radius"]
    };

    (*=========================================================================*)
    (* 関数定義 : 一般化座標 -> 関節座標 *)
    hDevice["str:q2j"] = StringReplace[
(* ■■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■■ *)
"Compile[{{q,_Real,1}
},With[{
    dof=$dof,pos=$pos,l=$length,j0=$j0,
    absq=$angle+Accumulate[q]
},
```

```
Join[j0,FoldList[Plus,pos,
   Table[l[[n]]*{Cos[absq[[n]]],Sin[absq[[n]]]},{n,dof}]
   ]]
]]", rules];
(* ■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■ *)
   hDevice["eq:q2j"] = ToExpression@hDevice["str:q2j"];

   (*=====================================================================*)
   (* 関数定義 : 一般化座標 -> ポイント座標&半径 {x,y,r} *)
   hDevice["str:q2point"] = StringReplace[
(* ■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■ *)
"Compile[{{q,_Real,1}
},With[{
   pos=$pos,
   l=$length,
   absq=$angle+Accumulate[q],
   point=$point,
   jPoint=$jPoint,
   radius=$radius
},Module[{
   j=FoldList[Plus,pos,
      Table[l[[n]]*{Cos[absq[[n]]],Sin[absq[[n]]]},{n,$dof}]],
   jnum
},
   Table[jnum=Round[jPoint[[n]]];
      If[jnum==0,
         Append[point[[n]],radius[[n]]],
         Append[j[[jnum]]+{
            point[[n,1]]*Cos[absq[[jnum]]]-point[[n,2]]*Sin[absq[[jnum]]],
            point[[n,2]]*Cos[absq[[jnum]]]+point[[n,1]]*Sin[absq[[jnum]]]
            },radius[[n]]]
      ],{n,$nPoint}]
]]]", rules];
(* ■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■ *)
   hDevice["eq:q2point"] = ToExpression@hDevice["str:q2point"];

   (*=====================================================================*)
   (* 関数定義 : 一般化座標 -> ポリゴン座標 *)
   hDevice["str:q2polygon"] = StringReplace[
(* ■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■ *)
"Compile[{{q,_Real,1}
},With[{
   dof=$dof,
   pos=$pos,
   l=$length,
   absq=$angle+Accumulate[q],
   polygon=$polygon,
   jPolygon=$jPolygon
},Module[{
   j=FoldList[Plus,pos,
      Table[l[[n]]*{Cos[absq[[n]]],Sin[absq[[n]]]},{n,$dof}]],
   jnum
},
   Table[jnum=jPolygon[[n]];
    If[jnum==0, polygon[[n,k]],
      j[[jnum]]+{
       polygon[[n,k,1]]*Cos[absq[[jnum]]]-polygon[[n,k,2]]*Sin[absq[[jnum]]],
       polygon[[n,k,2]]*Cos[absq[[jnum]]]+polygon[[n,k,1]]*Sin[absq[[jnum]]]}
     ],{n,$nPolygon,1,-1},{k,4}]
]]]", rules];
(* ■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■ *)
   hDevice["eq:q2polygon"] = ToExpression@hDevice["str:q2polygon"];

   (*=====================================================================*)
```

```
1238    (* 編集用セルデータの作成[関数] *)
1239    hDevice["Cells"] := LAB`Autocomplete["Cells",
1240       {Group <> " [" <> ToString@hDevice <> "]", "subsection"},
1241       {"Parameter",    "subsubsection"}, hDevice["str:Items"],
1242       {"eq:q2j",       "subsubsection"}, hDevice["str:q2j"],
1243       {"eq:q2point",   "subsubsection"}, hDevice["str:q2point"],
1244       {"eq:q2polygon", "subsubsection"}, hDevice["str:q2polygon"]
1245    ];
1246    (* 戻り値 *)
1247    Return@hDevice;
1248 ]
```

[関数定義] LAB`Autocomplete["motor"]
    function : デバイスのパラメータを自動補完 - モータ
    argument1 : 全データのハンドル
    argument2 : デバイスハンドル
    argument3 : グループ名（デバイス名）   - String
    return   : デバイス用独立ハンドル（device*）

```
1249 LAB`Autocomplete["motor", Handle_, hDevice_, Group_, ___] := Module[{
1250    list, mm = N[10^(-3)], g = N[10^(-3)], ret, old, name, type, pre
1251    },
1252    list = #[[2 ;;, 1 ;; 3]] &@ {
1253       {(* アイテム名 *),      (* 型 *),     (* 基準値 *)},
1254       {"data model",          "replacement", "device"   },
1255       {"device model",        "replacement", "motor"    },
1256       {"name",                "replacement", Group      },
1257       {"dof",                 "replacement",   1        },
1258       {"moment of inertia",   "float",         1        },
1259       {"damping of rotation", "float",         1        },
1260       {"target",              "float",         0        },
1261       {"target-max",          "float",       100        },
1262       {"target-min",          "float",      -100        },
1263       {"target-move",         "float",         0.1      },
1264       {"method",              "string",      "P"        },
1265       {"kp",                  "float",         1        },
1266       {"kd",                  "float",         0        },
1267       {"ki",                  "float",         0        },
1268       {"maximum output",      "float",         5        }
1269    }
1270
1271    (*=========================================================================*)
1272    (* アイテム値の登録 *)
1273    LAB`Autocomplete["Registration", Handle, hDevice, Group, list];
1274    (* パラメータ補完:初期値 *)
1275    hDevice["q0"]  = ConstantArray[0, hDevice["dof"]];
1276    hDevice["dq0"] = ConstantArray[0, hDevice["dof"]];
1277    (* パラメータ補完:制御対象リスト *)
1278    hDevice["controls"] = {"target"};
1279
1280    (*=========================================================================*)
1281    (* 運動方程式の定義 *)
1282    hDevice["eq:ddq"] = LAB`LagrangeEquation["motor", hDevice];
1283    LAB`Log["↓ ラグランジュ法による運動方程式を登録しました"];
1284    (* 初期値を一括セットする関数 *)
1285    hDevice["eq:preset"][run_] := (
1286       run[hDevice, "q0"]     = hDevice["q0"];
1287       run[hDevice, "dq0"]    = hDevice["dq0"];
1288       run[hDevice, "target"] = hDevice["target"];
1289    );
1290    (* トルク計算式 - 制御方式に従う *)
1291    With[{
```

```
      kp = hDevice["kp"], kd = hDevice["kd"], ki = hDevice["ki"],
      max = hDevice["maximum output"]
      },
      Switch[hDevice["method"],
      (* P制御 *)
      "P", hDevice["eq:q2Q"] = Compile[{{q, _Real, 1}, {target, _Real}},
         Module[{out},
            out = ({target*Degree} - q)*kp;
            If[out[[1]] >  max, out = {max}];
            If[out[[1]] < -max, out = {-max}];
            out
         ]];
      ];
   ];
   (*=========================================================================*)
   (* 編集用セルデータの作成[関数] *)
   hDevice["Cells"] := LAB`Autocomplete["Cells",
      {Group <> " [" <> ToString@hDevice <> "]", "subsection"},
      {"Parameter", "subsubsection"}, hDevice["str:Items"]
   ];
   (* 戻り値 *)
   Return@hDevice;
]
```

[関数定義] LAB`Autocomplete["rigid"]
    function : デバイスのパラメータを自動補完 - 剛体
    argument1 : 全データのハンドル
    argument2 : デバイスハンドル
    argument3 : グループ名(デバイス名)   - String
    return   : デバイス用独立ハンドル(device*)

```
LAB`Autocomplete["rigid", Handle_, hDevice_, Group_, ___] := Module[{
   list, sublist, mm = N[10^(-3)], g = N[10^(-3)], μ = N[10^(-6)],
   ret, old, name, type, pre, toStr, rules, form
   },
   list = #[[2 ;;, 1 ;; 3]] &@ {
      {(* アイテム名 *),          (* 型 *),          (* 基準値 *)},
      {"data model",              "replacement",    "device"    },
      {"device model",            "replacement",    "rigid"     },
      {"name",                    "replacement",    Group       },
      {"dof",                     "replacement",    3           },
      {"mass",                    "float",          100*g       },
      {"moment of inertia",       "float",          1*μ         },
      {"damping of rotation",     "float",          0.001       },
      {"damping of translation", "float",           0.3         },
      {"pos",                     "list[2]:float",  {0, 0}*mm   },
      {"angle",                   "float",          0*Degree    },
      {"form",                    "string",         "round"     }
   };
   sublist["round"] = #[[2 ;;, 1 ;; 3]] &@ {
      {(* アイテム名 *),          (* 型 *),          (* 基準値 *)},
      {"radius",                  "float",          50*mm       }
   };
   sublist["rectangle"] = #[[2 ;;, 1 ;; 3]] &@ {
      {(* アイテム名 *),          (* 型 *),          (* 基準値 *)},
      {"radius",                  "float",           5*mm       },
      {"height",                  "float",          50*mm       },
      {"width",                   "float",          50*mm       }
   };

   (*=========================================================================*)
   (* アイテム値の登録 *)
```

```
1346    LAB`Autocomplete["Registration", Handle, hDevice, Group, list];
1347    form = hDevice["form"];
1348    LAB`Autocomplete["Registration", Handle, hDevice, Group, sublist[form]];
1349    (* パラメータ補完:形状関連 *)
1350    Switch[form,
1351       "round",
1352          hDevice["polygon"]      = {};
1353          hDevice["point"]        = {{0, 0}};
1354          hDevice["point enable"] = {1};
1355          hDevice["point radius"] = {hDevice["radius"]};
1356          hDevice["line angle"]   = {};
1357          hDevice["line length"]  = {};
1358          hDevice["line enable"]  = {};,
1359       "rectangle",
1360          hDevice["polygon"] =
1361             LAB`RectanglePoint[{hDevice["width"], hDevice["height"]}, 0];
1362          hDevice["point"]        = hDevice["polygon"];
1363          hDevice["point enable"] = {1, 1, 1, 1};
1364          hDevice["point radius"] = {0, 0, 0, 0};
1365          hDevice["line angle"]   = LAB`PolygonAngle@hDevice["polygon"];
1366          hDevice["line length"]  = LAB`PolygonLength@hDevice["polygon"];
1367          hDevice["line enable"]  = {1, 1, 1, 1};
1368    ];
1369    (* パラメータ補完:初期値 *)
1370    hDevice["q0"]       = Join[hDevice["pos"], {hDevice["angle"]}];
1371    hDevice["dq0"]      = ConstantArray[0, hDevice["dof"]];
1372    (* パラメータ補完:制御対象リスト *)
1373    hDevice["controls"] = {};
1374
1375    (*=========================================================================*)
1376    (* 運動方程式の定義 *)
1377    hDevice["eq:ddq"] = LAB`LagrangeEquation["rigid", hDevice];
1378    LAB`Log["↓ ラグランジュ法による運動方程式を登録しました"];
1379    (* 初期値を一括セットする関数 *)
1380    hDevice["eq:preset"][run_] := (
1381       run[hDevice, "q0"]  = hDevice["q0"];
1382       run[hDevice, "dq0"] = hDevice["dq0"];
1383    );
1384
1385    (*=========================================================================*)
1386    (* 補助関数,データの生成 *)
1387    toStr[x_] := LAB`DataConvert[hDevice[x], "strd"];
1388    rules = {
1389       "$handle"   -> ToString@hDevice,
1390       "$point"    -> toStr["point"],
1391       "$polygon"  -> toStr["polygon"],
1392       "$radius"   -> toStr["point radius"],
1393       "$nPolygon" -> ToString@Length@hDevice["polygon"],
1394       "$nPoint"   -> ToString@Length@hDevice["point"]
1395    };
1396
1397    (*=========================================================================*)
1398    (* 関数定義 : 一般化座標 -> ポイント座標 *)
1399    hDevice["str:q2point"] = StringReplace[
1400     "Compile[{{q,_Real,1}
1401     },With[{
1402       point=$point,
1403       radius=$radius
1404     },
1405     Table[{q[[1]],q\
1406 [[2]],radius[[\
1407 n]]}+{
1408       point[[n,1]]*Cos[q\
1409 [[3]]]-point[[\
```

```
n,2]]*Sin[q[[3\
]]],
      point[[n,2]]*Cos[q\
[[3]]]+point[[\
n,1]]*Sin[q[[3\
]]],
      0},{n,$nPoint}]
    ]]", rules];
  hDevice["eq:q2point"] = ToExpression@hDevice["str:q2point"];

   (*=====================================================================*)
   (* 関数定義 : 一般化座標 -> ポリゴン座標 *)
   hDevice["str:q2polygon"] = StringReplace[
(* ■■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■ *)
"Compile[{{q,_Real,1}
},With[{
   polygon=$polygon
},
   Table[{q[[1]],q[[2]]}+{
      polygon[[n,1]]*Cos[q[[3]]]-polygon[[n,2]]*Sin[q[[3]]],
      polygon[[n,2]]*Cos[q[[3]]]+polygon[[n,1]]*Sin[q[[3]]]
      },{n,$nPolygon}]
]]", rules];
(* ■■■■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■■■■ *)
   If[Length@hDevice["polygon"] == 0,
      hDevice["eq:q2polygon"][_] = {},
      hDevice["eq:q2polygon"] = ToExpression@hDevice["str:q2polygon"];
   ];

   (*=====================================================================*)
   (* 編集用セルデータの作成[関数] *)
   hDevice["Cells"] := LAB`Autocomplete["Cells",
      {Group <> " [" <> ToString@hDevice <> "]", "subsection"},
      {"Parameter",    "subsubsection"}, hDevice["str:Items"],
      {"eq:q2point",   "subsubsection"}, hDevice["str:q2point"],
      {"eq:q2polygon", "subsubsection"}, hDevice["str:q2polygon"]
   ];
   (* 戻り値 *)
   Return@hDevice;
]
```

[関数定義] LAB`Autocomplete["elastic"]
    function : デバイスのパラメータを自動補完 - 弾性体
    argument1 : 全データのハンドル
    argument2 : デバイスハンドル
    argument3 : グループ名（デバイス名）   - String
    return   : デバイス用独立ハンドル（device*）

[関数定義] LAB`Autocomplete["dgs-vsm"]
    function : システムのパラメータを自動補完 - 差動機構
    argument1 : 全データのハンドル
    argument2 : システムハンドル
    argument3 : グループ名（システム名）   - String
    return   : システム用独立ハンドル（system*）

```
LAB`Autocomplete["dgs-vsm", Handle_, hSystem_, Group_, ___] := Module[{
   list, mm = N[10^(-3)], g = N[10^(-3)],
   ret, old, item, name, type, pre, dof, conectItem, sConect, hConect,
   END := Goto["END"]
   },
```

```
1456    list = #[[2 ;;, 1 ;; 3]] &@ {
1457       {(* アイテム名 *),  (* 型 *),        (* 基準値 *), □     },
1458       {"data model",      "replacement",   "system",      □     },
1459       {"system model",    "replacement",   "dgs-vsm",     □     },
1460       {"name",            "replacement",   Group,         □     },
1461       {"dof",             "replacement",   dof,           □     },
1462       {"conect-link",     "string",        "",            □     },
1463       {"conect-drive",    "string",        "",            □     },
1464       {"conect-vsm",      "string",        "",            □     },
1465       {"method",          "string",        "standard",    □     },
1466       {"spring modulus",  {"float", dof},  1,           "[N/mm]"},
1467       {"initial tension", {"float", dof},  0.4,         "[N]"   },
1468       {"initial length",  {"float", dof},  0,           "[mm]"  },
1469       {"1st pulley r",    {"float", dof},  4,           "[mm]"  },
1470       {"vsm pulley r",    {"float", dof},  4,           "[mm]"  },
1471       {"zl",              "int",          30,             □     },
1472       {"zs",              "int",          20,             □     },
1473       {"direction",       "int",           1,             □     }
1474    }
1475
1476    (*=========================================================================*)
1477    (* システムを接続するデバイスの名前, ハンドルを取得 *)
1478    conectItem = {"conect-link", "conect-drive", "conect-vsm"};
1479    Do[
1480       item = conectItem[[n]];
1481       (* デバイス名を取得 *)
1482       sConect[item] = Handle[Group][item];
1483       If[! StringQ@sConect[item] || sConect[item] == "",
1484          LAB`Warning["↓ パラメータ取得に失敗 [", item, "]"];
1485          LAB`Warning["↓ 処理を中断します"];
1486          END];
1487       (* デバイスハンドルを取得 *)
1488       hConect[item] = LAB`GetHandle["device", sConect[item]];
1489       If[hConect[item] === Null,
1490          LAB`Warning["↓ デバイス[", sConect[item], "]の登録が必要です"];
1491          LAB`Warning["↓ 処理を中断します"];
1492          END];
1493       (* 成功ログ *)
1494       LAB`Log["↓ デバイスを接続しました : ",
1495          hConect[item], " [", sConect[item], "]"];
1496       (* 接続するデバイスの数だけ繰り返す *)
1497    , {n, Length@conectItem}];
1498
1499    (*=========================================================================*)
1500    (* 自由度の取得 *)
1501    dof = hConect["conect-link"]["dof"];
1502    (* アイテム値の登録 *)
1503    LAB`Autocomplete["Registration", Handle, hSystem, Group, list];
1504    (* パラメータ補完:接続デバイスハンドル *)
1505    hSystem["handle-link"]  = hConect["conect-link"];
1506    hSystem["handle-drive"] = hConect["conect-drive"];
1507    hSystem["handle-vsm"]   = hConect["conect-vsm"];
1508    (* パラメータ補完:制御対象リスト *)
1509    hSystem["controls"]     = {};
1510
1511    (*=========================================================================*)
1512    (* トルク計算式 - システム方式に従う *)
1513    With[{
1514       zl = hSystem["zl"], zs = hSystem["zs"],
1515       k = hSystem["spring modulus"],
1516       it = hSystem["initial tension"], il = hSystem["initial length"],
1517       r1 = hSystem["1st pulley r"], rv = hSystem["vsm pulley r"],
1518       dir = hSystem["direction"]
1519       },
```

```
    Switch[hSystem["method"],
       (* 標準 *)
       "standard",
          hSystem["eq:q2Q"] = Compile[{
             {q, _Real, 1}, {drive, _Real, 1}, {vsm, _Real, 1}
             },
             Module[{qbase, stretch, tension, τbase, τlink},
                qbase = (zl^2 - zs^2)/zs^2*q - drive[[1]]*dir;(*[degree]*)
                qbase[[1]] *= (zs^2/zl^2);
                stretch = il + r1*Abs[qbase] + rv*vsm[[1]];(*[mm]*)
                tension = it + k*stretch;(*[N]*)
                τbase = -Sign[qbase]*r1*10^(-3)*tension;(*[Nm]*)
                τlink = (zl^2 - zs^2)/zs^2*τbase;
                τlink[[1]] *= (zs^2/zl^2);
                τlink
             ]];
    ]];
    (*=========================================================================*)
    (* 編集用セルデータの作成[関数] *)
    hSystem["Cells"] := LAB`Autocomplete["Cells",
       {Group <> " [" <> ToString@hSystem <> "]", "subsection"},
       {"Parameter", "subsubsection"}, hSystem["str:Items"]
    ];
    (* 戻り値 *)
    Return@hSystem;
    Label["END"];
    Return@0;
]
```

[関数定義] LAB`Autocomplete["config"]
    function : 各グループを統合した計算に必要なデータを生成
    argument1 : 全データのハンドル
    argument2 : 計算設定ハンドル
    argument3 : グループ名(計算設定名)   - String
    argument4 : 計算に組み込むデバイス,システムのハンドルリスト / Automatic / Null
    return   : 計算設定用独立ハンドル

```
LAB`Autocomplete["config", Handle_, hConfig_, Group_, Contents_, ___] :=
Module[{
   list, hContents = {}, tmp, handle, ret, old, name, type, pre
   },
   list = #[[2 ;;, 1 ;; 3]] &@ {
      {(* アイテム名 *),      (* 型 *),       (* 基準値 *)},
      {"data model",          "float",         "config"   },
      {"name",                "replacement",   Group      },
      {"contents",            "string",        "{}"       },
      {"target time",         "int",              5       },
      {"integration method", "float",          "RK4"      },
      {"calc/sec",            "int",           1200       },
      {"calc/rec",            "int",              1       },
      {"left",                "replacement",     -0.3     },
      {"right",               "replacement",      0.3     },
      {"bottom",              "replacement",     -0.27    },
      {"top",                 "replacement",      0.28    },
      {"size",                "replacement",    500       }
   }
   (*=========================================================================*)
   (* コンテンツリストの作成 - デバイス,システム *)
   Which[
      (* リストが指定されている場合 *)
      ListQ@Contents, tmp = Flatten@Contents,
      (* Automatic/Nullが指定されている場合 *)
```

```
      Contents === Automatic || Contents === Null,
         tmp = LAB`DataConvert[Handle[Group]["contents"], "list"]
   ];
   (* コンテンツのハンドルリストを作成 *)
   Do[Which[
      (* ハンドルを直接指定している場合はそのまま追加 *)
      Head@tmp[[n]] === Symbol,
      handle = tmp[[n]];
      name = handle["name"];
      AppendTo[hContents, handle];
      LAB`Log["↓ コンテンツを登録しました : ", handle, " [", name, "]"],
      (* デバイス/システム名を指定している場合はハンドルを取得し追加 *)
      StringQ@tmp[[n]],
      name = tmp[[n]];
      handle = LAB`GetHandle[Null, tmp[[n]]];
      AppendTo[hContents, handle];
      LAB`Log["↓ コンテンツを登録しました : ", handle, " [", name, "]"]
   ], {n, Length@tmp}];

   (*=========================================================================*)
   (* アイテム値の登録 *)
   LAB`Autocomplete["Registration", Handle, hConfig, Group, list];
   (* 補完パラメータの登録 *)
   hConfig["step"] = N[1/hConfig["calc/sec"]];
   hConfig["device list"] = LAB`SortHandle@
      Select[hContents, #["data model"] === "device" &];
   hConfig["system list"] = LAB`SortHandle@
      Select[hContents, #["data model"] === "system" &];

   (*=========================================================================*)
   (* 編集用セルデータの作成[関数] *)
   hConfig["Cells"] := LAB`Autocomplete["Cells",
      {Group <> " [" <> ToString@hConfig <> "]", "subsection"},
      {"Parameter", "subsubsection"}, hConfig["str:Items"]
   ];
   (* 戻り値 *)
   Return@hConfig;
]
```

[関数定義] LAB`String["rendering"]
function : 現在値リストからグラフィックスを生成する関数のコードを文字列で生成する
argument1 : 計算設定が保存されている変数群のハンドル
return : 文字列化された関数コード

```
LAB`String["rendering", hConfig_] := Module[{
   hRun = hConfig["handle:run"],
   str, nDevs, hDevs, sDevs, nDofs, nRanges, sRanges, strJoin, sStep,
   left = hConfig["left"], right = hConfig["right"],
   bottom = hConfig["bottom"], top = hConfig["top"],
   size = hConfig["size"], per, list, pos
   },
   (* 横幅を100等分した長さ *)
   per = (right - left)/100;
   (* 関数[LAB`String]にハンドルを順に渡し,生成された文字列を結合する関数 *)
   strJoin[model_, list_, option___] := StringJoin@Table[
      LAB`String[model, list[[n]], n, option], {n, Length@list}];
   (* デバイスの数,ハンドル,名前,自由度,自由度の始点-終点番号のリスト *)
   nDevs = Length@hConfig["device list"];
   hDevs = hConfig["device list"];
   sDevs = #["name"] & /@ hDevs;
   nDofs = #["dof"] & /@ hDevs;
   nRanges = Table[
```

```
1629        Total[#[[;; i]]] + {1, #[[i + 1]]}, {i, nDevs}] &@Prepend[nDofs, 0];
1630    sRanges =
1631        LAB`DataConvert[{#[[1]], ";;", #[[2]]}, "strjoin"] & /@ nRanges;
1632    (* 計算の刻み時間 *)
1633    sStep = ToString@hConfig["step"];
1634
1635    (* 関数コードの生成 *)
1636    str = StringReplace[
1637 (* ■■■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■■■ *)
1638 "$run[\"Rendering\"]=Function[{arg,Option},Module[{
1639    $arg2q,
1640    text=\"text\"/.Option,
1641    Cplist=\"Cplist\"/.Option,
1642    Fvlist=\"Fvlist\"/.Option,
1643    j
1644 },
1645 If[text===\"text\",text=\"\"];
1646 If[!ListQ@Cplist,Cplist={}];
1647 If[!ListQ@Fvlist,Fvlist={}];
1648
1649 (* グラフィック生成 *)
1650 Graphics[{" <>
1651 (*-----------------------------------------------------------------*)
1652 (* 多関節リンク *)
1653 LAB`DataConvert[Table[If[hDevs[[n]]["device model"] === "link", {
1654 "\n(* " <> LAB`DataConvert[hDevs[[n]]["name"], "blue"] <> " *)
1655 LightGray,EdgeForm[{Thickness[0.002],Black}],
1656 Polygon[#]&@", hDevs[[n]], "[\"eq:q2polygon\"][q", n, "],
1657 Disk[#[[;;2]],#[[3]]]&/@", hDevs[[n]], "[\"eq:q2point\"][q", n, "],
1658 Black,
1659 Line[", hDevs[[n]], "[\"eq:q2j\"][q", n, "]],"
1660 }, ""], {n, nDevs}], "strjoin"] <>
1661 (*-----------------------------------------------------------------*)
1662 (* モータ *)
1663 LAB`DataConvert[Table[If[hDevs[[n]]["device model"] === "motor", {
1664 list = Select[hDevs, #["device model"] === "motor" &];
1665 pos = (Length@list - FirstPosition[list, hDevs[[n]]][[1]])*3.5*per;
1666 "\n(* " <> LAB`DataConvert[hDevs[[n]]["name"], "blue"] <> " *)
1667 Text[Style[SetPrecision[q", n, "[[1]]/Degree//N,3],13],",
1668    "{", right - 11*per, ",", bottom + 1*per + pos, "},{-1,-1}],
1669 Text[Style[\":\",Black,13],",
1670    "{", right - 12*per, ",", bottom + 1*per + pos, "},{1,-1}],
1671 Text[Style[\"", hDevs[[n]]["name"], "\",Black,13],",
1672    "{", right - 14*per, ",", bottom + 1*per + pos, "},{1,-1}],"
1673 }, ""], {n, nDevs}], "strjoin"] <>
1674 (*-----------------------------------------------------------------*)
1675 (* 剛体 *)
1676 LAB`DataConvert[Table[If[hDevs[[n]]["device model"] === "rigid", {
1677 "\n(* " <> LAB`DataConvert[hDevs[[n]]["name"], "blue"] <> " *)
1678 Green,EdgeForm[{Thickness[0.002],Black}],
1679 Polygon[#]&@", hDevs[[n]], "[\"eq:q2polygon\"][q", n, "],
1680 Disk[#[[;;2]],#[[3]]]&/@", hDevs[[n]], "[\"eq:q2point\"][q", n, "],"
1681 }, ""], {n, nDevs}], "strjoin"] <>
1682 (*-----------------------------------------------------------------*)
1683 (* 接触 *)
1684 LAB`DataConvert[{
1685 "\n(* " <> LAB`DataConvert["contact force", "blue"] <> " *)
1686 Red,Thick,
1687 Disk[#,0.0025]&/@Cplist,
1688 Arrow[#]&/@Transpose[{Cplist,0.10*Fvlist+Cplist}],"
1689 }, "strjoin"] <>
1690 (*-----------------------------------------------------------------*)
1691 (* 補足テキスト:左上 *)
1692 LAB`DataConvert[{
```

```
"\n(* " <> LAB`DataConvert["time", "blue"] <> " *)
Text[Style[ToString[text],Black,13],",
   "{", left + 3 per, ",", top - 3 per, "},{-1,1}]"
}, "strjoin"] <>
(*------------------------------------------------------------*)
"\n},
(* グラフィックス設定 *)
PlotRange->{{" <>
   ToString@left <> "," <> ToString@right <> "},{" <>
   ToString@bottom <> "," <> ToString@top <> "}},
ImageSize->" <> ToString@size <> ",
Axes->False,
Frame->True
]]];", {
(* ■■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■ *)
   (* 実行中処理用ハンドル *)
   "$run" -> ToString@hRun,
   (* qの振り分け *)
   "$arg2q" -> LAB`DataConvert[
      Table[{"q", n, "=arg[[", sRanges[[n]], "]]"}, {n, nDevs}], "sriffle:,"]
   }];
   (* 戻り値 *)
   Return@str
]
```

[関数定義] LAB`String["1step"]
function : 1ステップ計算する関数のコードを文字列で生成する
argument1 : 計算設定が保存されている変数群のハンドル
return : 文字列化された関数コード

```
LAB`String["1step", hConfig_] := Module[{
   str, hRun, strJoin,
   nDevs, hDevs, sDevs, nDofs, nRanges, sRanges, hList
   },
   (* 関数[LAB`String]にハンドルを順に渡し,生成された文字列を結合する関数 *)
   strJoin[model_, list_, mode_] := StringJoin@Table[
      LAB`String[model, list[[n]], n, mode, hConfig], {n, Length@list}];
   (* 実行処理ハンドル, 記録ハンドル *)
   hRun = hConfig["handle:run"];
   (* デバイスの数,ハンドル,名前,自由度,自由度の始点-終点番号のリスト *)
   nDevs = Length@hConfig["device list"];
   hDevs = hConfig["device list"];
   sDevs = #["name"] & /@ hDevs;
   nDofs = #["dof"] & /@ hDevs;
   nRanges = Table[
      Total[#[[;; i]]] + {1, #[[i + 1]]}, {i, nDevs}] &@Prepend[nDofs, 0];
   sRanges =
      LAB`DataConvert[{#[[1]], ";;", #[[2]]}, "strjoin"] & /@ nRanges;
   (* デバイス+システムのハンドル *)
   hList = Join[hDevs, hConfig["system list"]];

   (* 関数コードの生成 *)
   str = StringReplace[
(* ■■■■■■■■■■↓文字列による関数定義↓■■■■■■■■■■ *)
"$run[\"1step\"]=Compile[{{arg,_Real,2}
},With[{
(* 置換変数 *)
   $arg2q,
   $arg2dq," <>
   strJoin["link",    hList, "with"] <>
   strJoin["motor",   hList, "with"] <>
   strJoin["dgs-vsm", hList, "with"] <>
```

```
1749    strJoin["rigid",   hList, "with"] <>
1750    strJoin["elastic", hList, "with"] <> "
1751    (* 計算カウント *)
1752    count=Round[$run[\"count\"]+1]
1753 },Module[{
1754 (* 局所変数 *)
1755    ddq,jnum,θ,origin,vector,deform,Cp,Fm,Fv,
1756    Cplist={{0.,0.}},Fmlist={0.},Fvlist={{0.,0.}}," <>
1757    strJoin["link",    hList, "module"] <>
1758    strJoin["motor",   hList, "module"] <>
1759    strJoin["dgs-vsm", hList, "module"] <>
1760    strJoin["rigid",   hList, "module"] <>
1761    strJoin["elastic", hList, "module"] <> "
1762    $Q=0
1763 },
1764 (*-------------------------------------------------------------------------*)"
1765 <>
1766 strJoin["link",    hList, "code"] <>
1767 strJoin["motor",   hList, "code"] <>
1768 strJoin["dgs-vsm", hList, "code"] <>
1769 strJoin["rigid",   hList, "code"] <>
1770 strJoin["elastic", hList, "code"] <>
1771 StringJoin@Table[
1772    LAB`String["contact", hList[[n1]], n1, hList[[n2]], n2, "code"],
1773    {n1, Length@hList}, {n2, n1, Length@hList}] <> "\n
1774 (*-------------------------------------------------------------------------*)
1775 (* 記録 *)
1776 If[$run[\"rec\"],
1777    $run[\"rec\"]=False;
1778    $rec4q" <>
1779    strJoin["link",    hList, "record"] <>
1780    strJoin["motor",   hList, "record"] <>
1781    strJoin["dgs-vsm", hList, "record"] <>
1782    strJoin["rigid",   hList, "record"] <>
1783    strJoin["elastic", hList, "record"] <> "
1784    $run[\"rec:Cp\"][count]=Cplist;
1785    $run[\"rec:Fm\"][count]=Fmlist;
1786    $run[\"rec:Fv\"][count]=Fvlist;
1787    ];
1788 (* 加速度 *)
1789 ddq=Join[
1790    $eq4ddq
1791    ];
1792 (* 戻り値 *)
1793 {arg[[2]],ddq}
1794 ]],{
1795 (* コンパイルオプション *)
1796    {$run[\"count\"],_Integer},
1797    {$run[\"rec\"],True|False}," <>
1798    strJoin["link",    hList, "option"] <>
1799    strJoin["motor",   hList, "option"] <>
1800    strJoin["dgs-vsm", hList, "option"] <>
1801    strJoin["rigid",   hList, "option"] <>
1802    strJoin["elastic", hList, "option"] <> "
1803    {_Symbol[\"eq:q2Q\"][___],_Real,1},
1804    {_Symbol[\"eq:ddq\"][___],_Real,1}
1805 },CompilationTarget->\"C\"]", {
1806 (* ■■■■■■■■■■■■■■■■■↑↑■■■■■■■■■■■■■■■■■ *)
1807    (* 実行中処理用ハンドル *)
1808    "$run" -> ToString@hRun,
1809    (* 引数をq,dqに振り分け, Qを0で初期化 *)
1810    "$arg2q" -> LAB`DataConvert[
1811       Table[{"q", n,"=arg[[1,",sRanges[[n]],"]]"},{n,nDevs}],"sriffle:,"],
1812    "$arg2dq" -> LAB`DataConvert[
```

```
      Table[{"dq",n,"=arg[[2,",sRanges[[n]],"]]"},{n,nDevs}],"sriffle:,"],
   "$Q=0" -> LAB`DataConvert[
      Table[{"Q", n,"=",ToString@ConstantArray[0.,nDofs[[n]]]},{n, nDevs}],
      "sriffle:,", "strd"],
   (* qを記録 *)
   "$rec4q" -> LAB`DataConvert[Table[{
      hRun, "[\"rec:q\",\"", sDevs[[n]], "\"][count]=q", n, ";"
      }, {n, nDevs}], "sriffle:nt"],
   (* 加速度を計算する関数 *)
   "$eq4ddq" -> LAB`DataConvert[
      Table[{hDevs[[n]], "[\"eq:ddq\"][q", n, ",dq", n, ",Q", n, "]",
         "(* ", sDevs[[n]], " *)"}, {n, nDevs}], "sriffle:,nt"]
   }];
   (* 戻り値 *)
   Return@str
]
```

[関数定義] LAB`String["link"]
    function : LAB`String["1step"] 内での各デバイスの処理を行う関数コードの文字列を生成
                LAB`String["1step"] から呼び出される
    argument1 : デバイスパラメータが保存されている変数群のハンドル
    argument2 : デバイスの通し番号 - Number
    argument3 : 取得する文字列データの内容   - String (With, Module, Record, Code)
    return   : 文字列化された関数コードの一部

```
LAB`String["link", hDevice_, n_, Mode_, ___] := Module[{
   mode = ToLowerCase@Mode, name, dof, toStr
   },
   (* データモデル="device", デバイスモデル="link" でなければ "" を返す*)
   If[! hDevice["data model"]   === "device", Return[""]];
   If[! hDevice["device model"] === "link",   Return[""]];
   (* 他の引数の型のチェック *)
   If[! IntegerQ@n,   Return[""]];
   If[! StringQ@Mode, Return[""]];
   (* 良く使うデバイスパラメータの取得 *)
   name = LAB`DataConvert[hDevice["name"],"blue"]<>" [="<>ToString@n<>"]";
   dof = hDevice["dof"];
   (* 各モードごとの処理 *)
   Switch[mode,
      (*=============================================================*)
      (* 置換変数 *)
      "with",
      toStr[x_] := LAB`DataConvert[hDevice[x], "strd"];
      Return@StringReplace[LAB`DataConvert[{{
         {"\n\t(* ",         name,                       " *)\n\t"},
         {"dof$●=",          dof,                        ","      },
         {"q0$●=",           toStr["ref angle"],         ","      },
         {"x0y0$●=",         toStr["ref pos"],           ","      },
         {"link$●=",         toStr["length of link"],    ",\n\t"  },
         {"polygon$●=",      toStr["polygon"],           ",\n\t"  },
         {"nPolygon$●=",     Length@hDevice["polygon"], ",\n\t"  },
         {"jPolygon$●=",     toStr["polygon:j"],         ",\n\t"  },
         {"lineEnable$●=",   toStr["line enable"],       ",\n\t"  },
         {"angle$●=",        toStr["line angle"],        ",\n\t"  },
         {"length$●=",       toStr["line length"],       ",\n\t"  },
         {"point$●=",        toStr["point"],             ",\n\t"  },
         {"nPoint$●=",       Length@hDevice["point"],    ",\n\t"  },
         {"jPoint$●=",       toStr["point:j"],           ",\n\t"  },
         {"pointEnable$●=", toStr["point enable"],      ",\n\t"  },
         {"radius$●=",       toStr["point radius"],      ","      }
      }}, "strjoin"], {"●" -> ToString@n}],
      (*=============================================================*)
```

```
        (* 局所変数 *)
        "module",
        Return@StringReplace[LAB`DataConvert[{
            "\n\t", "absq●,jpos●,polygon●,angle●,point●,"
            }, "strjoin"], {"●" -> ToString@n}],
        (*============================================================*)
        (* 実行コード *)
        "code",
        Return@StringReplace[LAB`DataConvert[{
"\n(* ", name, " *)
absq●=q0$●+Accumulate[q●];
jpos●=FoldList[Plus,x0y0$●,
    Table[link$●[[n]]*{Cos[absq●[[n]]],Sin[absq●[[n]]]},{n,dof$●}]];
polygon●=Table[jnum=jPolygon$●[[n]];
    If[jnum==0,polygon$●[[n,i]],jpos●[[jnum]]+{
         polygon$●[[n,i,1]]*Cos[absq●[[jnum]]]
        -polygon$●[[n,i,2]]*Sin[absq●[[jnum]]],
         polygon$●[[n,i,2]]*Cos[absq●[[jnum]]]
        +polygon$●[[n,i,1]]*Sin[absq●[[jnum]]]
    }],{n,nPolygon$●},{i,4}];
angle●=Table[jnum=jPolygon$●[[n]];
    If[jnum==0,angle$●[[n]],absq●[[jnum]]+angle$●[[n]]
    ],{n,nPolygon$●}];
point●=Table[jnum=jPoint$●[[n]];
    If[jnum==0,point$●[[n]],jpos●[[jnum]]+{
         point$●[[n,1]]*Cos[absq●[[jnum]]]
        -point$●[[n,2]]*Sin[absq●[[jnum]]],
         point$●[[n,2]]*Cos[absq●[[jnum]]]
        +point$●[[n,1]]*Sin[absq●[[jnum]]]
    }],{n,nPoint$●}];"
        }, "strjoin"], {"●" -> ToString@n}],
        (*============================================================*)
        (* 記録 *)
        "record",
        Return@StringReplace[LAB`DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*============================================================*)
        (* コンパイルオプション *)
        "option",
        Return@StringReplace[LAB`DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*============================================================*)
        (* 該当なし *)
        _, Return@""
    ]
]
```

[関数定義] LAB`String["motor"]
    function : LAB`String["1step"] 内での各デバイスの処理を行う関数コードの文字列を生成
                LAB`String["1step"] から呼び出される
    argument1 : デバイスパラメータが保存されている変数群のハンドル
    argument2 : デバイスの通し番号 - Number
    argument3 : 取得する文字列データの内容 - String (With, Module, Record, Code)
    argument4 : 計算設定が保存されている変数群のハンドル
    return   : 文字列化された関数コードの一部

```
LAB`String["motor", hDevice_, n_, Mode_, hConfig_, ___] := Module[{
    mode = ToLowerCase@Mode, name, dof, toStr,
    hRun = hConfig["handle:run"]
    },
```

```
    (* データモデル="device", デバイスモデル="motor" でなければ "" を返す*)
    If[! hDevice["data model"]   === "device", Return[""]];
    If[! hDevice["device model"] === "motor",  Return[""]];
    (* 他の引数の型のチェック *)
    If[! IntegerQ@n,   Return[""]];
    If[! StringQ@Mode, Return[""]];
    (* 良く使うデバイスパラメータの取得 *)
    name = LAB`DataConvert[hDevice["name"],"blue"]<>" [="<>ToString@n<>"]";
    (* 各モードごとの処理 *)
    Switch[mode,
       (*============================================================*)
       (* 置換変数 *)
       "with",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 局所変数 *)
       "module",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 実行コード *)
       "code",
       Return@StringReplace[LAB`DataConvert[{
          "\n(* ", name, " *)\n",
          "Q●+=$device[\"eq:q2Q\"][q●,$run[$device,\"target\"]];"
       }, "strjoin"], {"●" -> ToString@n,
          "$device" -> ToString@hDevice,
          "$run"    -> ToString@hRun}],
       (*============================================================*)
       (* 記録 *)
       "record",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* コンパイルオプション *)
       "option",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 該当なし *)
       _, Return@""
    ]
]
```

[関数定義] LAB`String["rigid"]
    function : LAB`String["1step"] 内での各デバイスの処理を行う関数コードの文字列を生成
                LAB`String["1step"] から呼び出される
    argument1 : デバイスパラメータが保存されている変数群のハンドル
    argument2 : デバイスの通し番号 - Number
    argument3 : 取得する文字列データの内容   - String (With, Module, Record, Code)
    return   : 文字列化された関数コードの一部

```
LAB`String["rigid", hDevice_, n_, Mode_, ___] := Module[{
    mode = ToLowerCase@Mode, name, dof, toStr
    },
    (* データモデル="device", デバイスモデル="rigid" でなければ "" を返す*)
    If[! hDevice["data model"]   === "device", Return[""]];
```

```
    If[! hDevice["device model"] === "rigid",  Return[""]];
    (* 他の引数の型のチェック *)
    If[! IntegerQ@n,   Return[""]];
    If[! StringQ@Mode, Return[""]];
    (* 良く使うデバイスパラメータの取得 *)
    name = LAB`DataConvert[hDevice["name"],"blue"]<>" [="<>ToString@n<>"]";
    (* 各モードごとの処理 *)
    Switch[mode,
       (*============================================================*)
       (* 置換変数 *)
       "with",
       toStr[x_] := LAB`DataConvert[hDevice[x], "strd"];
       Return@StringReplace[LAB`DataConvert[{{
          {"\n\t(* ",         name,                    " *)\n\t"},
          {"point$●=",       toStr["point"],          ",\n\t"  },
          {"nPoint$●=",      Length@hDevice["point"], ",\n\t"  },
          {"pointEnable$●=", toStr["point enable"],   ",\n\t"  },
          {"radius$●=",      toStr["point radius"],   ","      }
       },If[Length@hDevice["polygon"] != 0, {
          {"\n\tpolygon$●=", toStr["polygon"],        ",\n\t"  },
          {"nPolygon$●=",    Length@hDevice["polygon"],",\n\t"  },
          {"lineEnable$●=",  toStr["line enable"],    ",\n\t"  },
          {"angle$●=",       toStr["line angle"],     ",\n\t"  },
          {"length$●=",      toStr["line length"],    ","      }
       } , ""]
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 局所変数 *)
       "module",
       Return@StringReplace[LAB`DataConvert[{
          "\n\t",
          "pos●,polygon●,angle●,point●,"
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 実行コード *)
       "code",
       Return@StringReplace[LAB`DataConvert[{
"\n(* ", name, " *)
pos●={q●[[1]],q●[[2]]};
point●=Table[pos●+{
   point$●[[n,1]]*Cos[q●[[3]]]-point$●[[n,2]]*Sin[q●[[3]]],
   point$●[[n,2]]*Cos[q●[[3]]]+point$●[[n,1]]*Sin[q●[[3]]]
   },{n,nPoint$●}];" <>
If[Length@hDevice["polygon"] != 0, "
polygon●=Table[pos●+{
   polygon$●[[n,1]]*Cos[q●[[3]]]-polygon$●[[n,2]]*Sin[q●[[3]]],
   polygon$●[[n,2]]*Cos[q●[[3]]]+polygon$●[[n,1]]*Sin[q●[[3]]]
   },{n,nPolygon$●}];
angle●=q●[[3]]+angle$●;", ""]
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 記録 *)
       "record",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* コンパイルオプション *)
       "option",
       Return@StringReplace[LAB`DataConvert[{
          ""
       }, "strjoin"], {"●" -> ToString@n}],
       (*============================================================*)
       (* 該当なし *)
```

```
        _, Return@""
    ]
]
```

[関数定義] LAB‘String["elastic"]
    function : LAB‘String["1step"] 内での各デバイスの処理を行う関数コードの文字列を生成
                LAB‘String["1step"] から呼び出される
    argument1 : デバイスパラメータが保存されている変数群のハンドル
    argument2 : デバイスの通し番号 - Number
    argument3 : 取得する文字列データの内容   - String (With, Module, Record, Code)
    return   : 文字列化された関数コードの一部

```
LAB‘String["elastic", hDevice_, n_, Mode_, ___] := Module[{
    mode = ToLowerCase@Mode, name, dof, toStr
    },
    (* データモデル="device", デバイスモデル="rigid" でなければ "" を返す*)
    If[! hDevice["data model"]   === "device",  Return[""]];
    If[! hDevice["device model"] === "elastic", Return[""]];
    (* 他の引数の型のチェック *)
    If[! IntegerQ@n,   Return[""]];
    If[! StringQ@Mode, Return[""]];
    (* 良く使うデバイスパラメータの取得 *)
    name = LAB‘DataConvert[hDevice["name"],"blue"]<>" ["<>ToString@n<>"]";
    (* 各モードごとの処理 *)
    Switch[mode,
        (*=================================================================*)
        (* 置換変数 *)
        "with",
        toStr[x_] := LAB‘DataConvert[hDevice[x], "strd"];
        Return@StringReplace[LAB‘DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*=================================================================*)
        (* 局所変数 *)
        "module",
        Return@StringReplace[LAB‘DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*=================================================================*)
        (* 実行コード *)
        "code",
        Return@StringReplace[LAB‘DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*=================================================================*)
        (* 記録 *)
        "record",
        Return@StringReplace[LAB‘DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*=================================================================*)
        (* コンパイルオプション *)
        "option",
        Return@StringReplace[LAB‘DataConvert[{
            ""
        }, "strjoin"], {"●" -> ToString@n}],
        (*=================================================================*)
        (* 該当なし *)
        _, Return@""
    ]
]
```

[関数定義] LAB‘String["dgs-vsm"]
function : LAB‘String["1step"] 内での各システムの処理を行う関数コードの文字列を生成
LAB‘String["1step"] から呼び出される
argument1 : システムパラメータが保存されている変数群のハンドル
argument2 : システムの通し番号 - Number
argument3 : 取得する文字列データの内容 - String (With, Module, Record, Code)
argument4 : 計算設定が保存されている変数群のハンドル
return : 文字列化された関数コードの一部

```
2087 LAB‘String["dgs-vsm", hSystem_, n_, Mode_, hConfig_, ___] := Module[{
2088    mode = ToLowerCase@Mode, name, dof, toStr, link, drive, vsm, hList
2089    },
2090    (* データモデル="system", デバイスモデル="dgs-vsm" でなければ "" を返す*)
2091    If[! hSystem["data model"]   === "system",  Return[""]];
2092    If[! hSystem["system model"] === "dgs-vsm", Return[""]];
2093    (* 他の引数の型のチェック *)
2094    If[! IntegerQ@n,   Return[""]];
2095    If[! StringQ@Mode, Return[""]];
2096    (* 良く使うデバイスパラメータの取得 *)
2097    name  = LAB‘DataConvert[hSystem["name"],"blue"]<>" [="<>ToString@n<>"]";
2098    hList = Join[hConfig["device list"], hConfig["system list"]];
2099    link  = ToString@FirstPosition[hList, hSystem["handle-link"]][[1]];
2100    drive = ToString@FirstPosition[hList, hSystem["handle-drive"]][[1]];
2101    vsm   = ToString@FirstPosition[hList, hSystem["handle-vsm"]][[1]];
2102
2103    (* 各モードごとの処理 *)
2104    Switch[mode,
2105       (*================================================================*)
2106       (* 置換変数 *)
2107       "with",
2108       Return@StringReplace[LAB‘DataConvert[{
2109          ""
2110       }, "strjoin"], {"●" -> ToString@n}],
2111       (*================================================================*)
2112       (* 局所変数 *)
2113       "module",
2114       Return@StringReplace[LAB‘DataConvert[{
2115          ""
2116       }, "strjoin"], {"●" -> ToString@n}],
2117       (*================================================================*)
2118       (* 実行コード *)
2119       "code",
2120       Return@StringReplace[LAB‘DataConvert[{
2121          "\n(* ", name, " *)\n",
2122          "Q[●1]+=", hSystem,
2123          "[\"eq:q2Q\"][q[●1],q[●d],q[●v]];"
2124       }, "strjoin"], {"[●1]" -> link,
2125          "[●d]" -> drive,
2126          "[●v]" -> vsm}],
2127       (*================================================================*)
2128       (* 記録 *)
2129       "record",
2130       Return@StringReplace[LAB‘DataConvert[{
2131          ""
2132       }, "strjoin"], {"●" -> ToString@n}],
2133       (*================================================================*)
2134       (* コンパイルオプション *)
2135       "option",
2136       Return@StringReplace[LAB‘DataConvert[{
2137          ""
2138       }, "strjoin"], {"●" -> ToString@n}],
2139       (*================================================================*)
2140       (* 該当なし *)
```

```
        _, Return@""
    ]
]
```

[関数定義] LAB‘String["contact"]
    function : LAB‘String["1step"] 内での接触処理を行う関数コードの文字列を生成
                LAB‘String["1step"] から呼び出される
    argument1 : デバイスハンドル
    argument2 : デバイスの通し番号 - Number
    argument3 : デバイスハンドル
    argument4 : デバイスの通し番号 - Number
    argument5 : 取得する文字列データの内容   - String (Code)
    return   : 文字列化された関数コードの一部

```
LAB‘String["contact", hD1_, n1_, hD2_, n2_, Mode_, ___] := Module[{
    mode = ToLowerCase@Mode, tmp, name1, name2, x = 0.02, k = 5000
    },
    (* データモデル="device" でなければ "" を返す *)
    If[! hD1["data model"] === "device", Return[""]];
    If[! hD2["data model"] === "device", Return[""]];
    (* 接触が発生するデバイスモデルでなければ "" を返す *)
    tmp = LAB‘Contents["contact model"];
    If[! MemberQ[tmp, hD1["device model"]], Return[""]];
    If[! MemberQ[tmp, hD2["device model"]], Return[""]];
    (* 同一モデル指定で自己接触が存在しない場合モデルは "" を返す *)
    If[hD1 == hD2 && hD1["device model"] === #, Return[""]
       ] & /@ LAB‘Contents["single body model"];
    (* 他の引数の型のチェック *)
    If[! IntegerQ@n1, Return[""]];
    If[! IntegerQ@n2, Return[""]];
    If[! StringQ@Mode, Return[""]];
    (* 良く使うデバイスパラメータの取得 *)
    name1 = LAB‘DataConvert[hD1["name"],"blue"]<>" [="<>ToString@n1<>"]";
    name2 = LAB‘DataConvert[hD2["name"],"blue"]<>" [="<>ToString@n2<>"]";
    (* 各モードごとの処理 *)
    Switch[mode,
       (*=============================================================*)
       (* 実行コード *)
       "code",
       Return@LAB‘DataConvert[{"\n
(*-------------------------------------------------------------------------*)
(* contact : ", name1, " & ", name2, " *)" <> Which[
(* link[1] & rigid[2] *)
hD1["device model"] == "link" && hD2["device model"] == "rigid",
    LAB‘String["contact:link & rigid", hD1, n1, hD2, n2, x, k],
(* rigid[1] & link[2] *)
hD1["device model"] == "rigid" && hD2["device model"] == "link",
    LAB‘String["contact:link & rigid", hD2, n2, hD1, n1, x, k],
(* link[1] & elastic[2] *)
hD1["device model"] == "link" && hD2["device model"] == "elastic",
    LAB‘String["contact:link & elastic", hD1, n1, hD2, n2, x, k],
(* elastic[1] & link[2] *)
hD1["device model"] == "elastic" && hD2["device model"] == "link",
    LAB‘String["contact:link & elastic", hD2, n2, hD1, n1, x, k],
(* 該当なし *)
True, ""]
}, "strjoin"],
       (*=============================================================*)
       (* 該当なし *)
       _, Return@""
    ]
]
```

[関数定義] LAB‘String["contact:link & rigid"]
    function : LAB‘String["1step"] 内での接触処理を行う関数コードの文字列を生成
                LAB‘String["contact"] から呼び出される
    argument1 : link ハンドル
    argument2 : link の通し番号
    argument3 : rigid ハンドル
    argument4 : rigid の通し番号
    argument5 : 接触判定の最大貫入量
    argument6 : ペナルティ係数
    return   : 文字列化された関数コードの一部

```
2192 LAB‘String["contact:link & rigid", hD1_, n1_, hD2_, n2_, x_, k_, ___] :=
2193 Module[{
2194    },
2195    StringReplace[
2196 (*-----------------------------------------------------------------*)
2197 (* リンク-ライン & 剛体-ポイント *)
2198 "\n(* link-line & rigid-point *)
2199 Do[If[lineEnable$●1[[k,i]]==pointEnable$●2[[n]]!=0,
2200     θ=angle●1[[k,i]];
2201     origin=polygon●1[[k,i]];
2202     (* ラインとポイントの貫入量より接触を評価 *)
2203     vector={{Cos[-θ],-Sin[-θ]},{Sin[-θ],Cos[-θ]}}.(point●2[[n]]-origin);
2204     deform=radius$●2[[n]]-vector[[2]];
2205     If[0<vector[[1]]<length$●1[[k,i]]&&0<=deform<●x,
2206        (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
2207        Cp=origin+{Cos[θ],Sin[θ]}*vector[[1]];
2208        Fm=deform*●k;
2209        Fv={Sin[θ],-Cos[θ]}*Fm;
2210        PrependTo[Cplist,Cp];
2211        PrependTo[Fmlist,Fm];
2212        PrependTo[Fvlist,Fv];
2213        (* 一般化力へ反映 *)
2214        Q●1+=Table[
2215           If[j<=jPolygon$●1[[k]],(Cp-jpos●1[[j]]).{Fv[[2]],-Fv[[1]]},0]
2216           ,{j,dof$●1}];
2217        Q●2+={-Fv[[1]],-Fv[[2]],(Cp-pos●2).{-Fv[[2]],Fv[[1]]}};
2218    ]
2219 ],{k,nPolygon$●1},{i,4},{n,nPoint$●2}];" <>
2220 (*-----------------------------------------------------------------*)
2221 (* リンク-ポイント & 剛体-ポイント *)
2222 "\n(* link-point & rigid-point *)
2223 Do[If[pointEnable$●1[[n1]]==pointEnable$●2[[n2]]!=0,
2224    (* ポイントとポイントの貫入量より接触を評価 *)
2225    deform=
2226       radius$●2[[n2]]+radius$●1[[n1]]-Norm[point●2[[n2]]-point●1[[n1]]];
2227    If[0<deform<●x,
2228       (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
2229       vector=Normalize[point●2[[n2]]-point●1[[n1]]];
2230       Cp=point●1[[n1]]+vector*radius$●1[[n1]];
2231       Fm=deform*●k;
2232       Fv=-vector*Fm;
2233       PrependTo[Cplist,Cp];
2234       PrependTo[Fmlist,Fm];
2235       PrependTo[Fvlist,Fv];
2236       (* 一般化力へ反映 *)
2237       Q●1+=Table[
2238          If[j<=jPoint$●1[[n1]],(Cp-jpos●1[[j]]).{Fv[[2]],-Fv[[1]]},0]
2239          ,{j,dof$●1}];
2240       Q●2+={-Fv[[1]],-Fv[[2]],(Cp-pos●2).{-Fv[[2]],Fv[[1]]}};
2241     ]
2242 ],{n1,nPoint$●1},{n2,nPoint$●2}];" <>
2243 (*-----------------------------------------------------------------*)
```

```
(* リンク-ポイント & 剛体-ライン *)
If[Length@hD2["polygon"] == 0, "",
"\n(* link-point & rigid-line *)
Do[If[lineEnable$●2[[n2]]==pointEnable$●1[[n1]]!=0,
    θ=angle●2[[n2]];
    origin=polygon●2[[n2]];
    (* ラインとポイントの貫入量より接触を評価 *)
    vector={{Cos[-θ],-Sin[-θ]},{Sin[-θ],Cos[-θ]}}.(point●1[[n1]]-origin);
    deform=radius$●1[[n1]]-vector[[2]];
    If[0<vector[[1]]<length$●2[[n2]]&&0<=deform<●x,
       (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
       Cp=point●1[[n1]]+{Sin[θ],-Cos[θ]}*radius$●1[[n1]];
       Fm=deform*●k;
       Fv={-Sin[θ],Cos[θ]}*Fm;
       PrependTo[Cplist,Cp];
       PrependTo[Fmlist,Fm];
       PrependTo[Fvlist,Fv];
       (* 一般化力へ反映 *)
       Q●1+=Table[
          If[j<=jPoint$●1[[n1]],(Cp-jpos●1[[j]]).{Fv[[2]],-Fv[[1]]},0]
          ,{j,dof$●1}];
       Q●2+={-Fv[[1]],-Fv[[2]],(Cp-pos●2).{-Fv[[2]],Fv[[1]]}};
    ]
],{n1,nPoint$●1},{n2,nPolygon$●2}];"]
(*-------------------------------------------------------------*)
    , {
       "●1" -> ToString@n1,
       "●2" -> ToString@n2,
       "●x" -> ToString[x],
       "●k" -> ToString[k]
    }]
]
```

[関数定義] LAB‘Simulation["set"]
    function : 渡された計算設定ハンドルの初期化を行う
    argument1 : 計算設定が保存されている変数群のハンドル
    argument2 : モード
                - new   : ハンドルを新規登録する
                - reset   : ハンドルを初期化する
                - restore : 指定ハンドルを復元する - Mode[[2]]:
    argument3 : 復元する実行用ハンドル（モードが"restore"の場合のみ有効）
    return   : 成功 : 1
                失敗 : 0

```
LAB‘Simulation["set", hConfig_, Mode_, Option___] := Module[{
    hRun = hConfig["handle:run"],
    mode = ToLowerCase@Mode,
    hRestore = If[Length@{Option} >= 1, {Option}[[1]], Null],
    tmp, device, backup
    },
    (*=============================================================================*)
    (* 開始ログ *)
    LAB‘Log["● シミュレーションの準備を行います : ", hConfig];
    (* 実行用ハンドルを使用中の場合は処理を停止 *)
    If[LAB‘UsingHandle[hRun] > 0,
       LAB‘Warning["↓ ハンドルを使用中のため処理を中断します"];
       Return@0];
    (* モードの補正 - 初期化モードでシンボルが不正の場合は新規モードに変更 *)
    If[mode === "reset" && ! Head@hRun === Symbol, mode = "new"];
    (*=============================================================================*)
    (* 実行ハンドルの取得, その他のモード毎処理 *)
    Which[
```

```
2294        (*-----------------------------------------------------------*)
2295        (* 新規モード *)
2296        mode === "new", (
2297            (* 実行処理用ハンドルを定義,データモデル&状態を登録 *)
2298            hRun = hConfig["handle:run"] = Unique@"run";
2299            hRun["data model"] = "run";
2300            hRun["state"] = "set";
2301            FinishDynamic[];
2302            LAB`Log["↓ 実行用ハンドルを新しく定義しました : ", hRun];
2303            (* 目標時間のセット *)
2304            hRun["target:str"] = ToString@hConfig["target time"];
2305            (* 初期値,制御値のセット - run*[#]["q0"], run*[#]["dq0"], etc. *)
2306            #["eq:preset"][hRun] & /@ hConfig["device list"];
2307            (* 1ステップ計算用関数 *)
2308            hRun["str:1step"] = LAB`String["1step", hConfig];
2309            ToExpression@hRun["str:1step"];
2310            LAB`Log["↓ 1ステップ計算用関数を定義しました"];
2311            (* 数値積分関数 *)
2312            hRun["Integration"] = LAB`NumericalIntegration[
2313                hConfig["integration method"], hRun["1step"], hConfig["step"]];
2314            LAB`Log["↓ 数値積分関数を定義しました - ",
2315                hConfig["integration method"], ", fps=", hConfig["calc/sec"]];
2316            (* 描画関数 *)
2317            hRun["str:Rendering"] = LAB`String["rendering", hConfig];
2318            ToExpression@hRun["str:Rendering"];
2319            LAB`Log["↓ 描画関数を定義しました"];
2320        ),
2321        (*-----------------------------------------------------------*)
2322        (* 初期化モード *)
2323        mode === "reset", (
2324            (* 退避データの項目 *)
2325            backup = {"1step", "Integration", "Rendering",
2326                "str:1step", "str:Rendering", "target:str"};
2327            (* データの退避 - 1ステップ/数値積分/描画関数,初期値,制御値 *)
2328            (tmp[#] = hRun[#]) & /@ backup;
2329            Do[device = hConfig["device list"][[n]];
2330                (tmp[device, #] = hRun[device, #]) & /@ {"q0", "dq0"};
2331                (tmp[device, #] = hRun[device, #]) & /@ device["controls"];
2332                , {n, Length@hConfig["device list"]}];
2333            ClearAll[#] &@hRun;
2334            (* 退避データの復元 *)
2335            (hRun[#] = tmp[#]) & /@ backup;
2336            Do[device = hConfig["device list"][[n]];
2337                (hRun[device, #] = tmp[device, #]) & /@ {"q0", "dq0"};
2338                (hRun[device, #] = tmp[device, #]) & /@ device["controls"];
2339                , {n, Length@hConfig["device list"]}];
2340            ClearAll[#] &@tmp;
2341            (* データモデル&状態を登録 *)
2342            hRun["data model"] = "run";
2343            hRun["state"] = "set";
2344            FinishDynamic[];
2345            LAB`Log["↓ 実行用ハンドルは規定値を使用します : ", hRun];
2346        ),
2347        (*-----------------------------------------------------------*)
2348        (* 復元モード *)
2349        mode === "restore", (
2350            If[! Head@hRestore === Symbol,
2351                LAB`Warning["↓ 復元データが存在しません"];
2352                Return@0];
2353            (* ハンドル&状態を登録 *)
2354            hRun = hConfig["handle:run"] = hRestore;
2355            hRun["state"] = "set"; FinishDynamic[];
2356            LAB`Log["↓ 実行用ハンドルを復元します : ", hRun];
2357        ),
```

```
2358        (*-----------------------------------------------------------*)
2359        (* 該当なし *)
2360        True,
2361           LAB`Error["↓ 計算準備のモードが不正のため中断します"];
2362           Return@0
2363        (*-----------------------------------------------------------*)
2364     ];
2365     (*=========================================================================*)
2366     (* 新規モード/初期化モードの場合の追加処理 *)
2367     If[mode === "new" || mode === "reset",
2368        (* 各パラメータの初期化 *)
2369        hRun["count"] = 0;          (* 現在カウント値 *)
2370        hRun["time"] = 0;           (* 現在時間 *)
2371        hRun["date:run"] = "---"; (* 最終実行時刻 *)
2372        (* 目標時間 *)
2373        If[! StringQ@hRun["target:str"] ||
2374           ! NumberQ@ToExpression@hRun["target:str"],
2375           hRun["target:str"] = "0"];
2376        (* 一般化座標の初期値,現在値 *)
2377        hRun["q0"] = Flatten[hRun[#, "q0"] & /@ hConfig["device list"]];
2378        hRun["dq0"] = Flatten[hRun[#, "dq0"] & /@ hConfig["device list"]];
2379        hRun["qdq"] = {hRun["q0"], hRun["dq0"]};
2380        LAB`Log["↓ 初期値を登録しました"];
2381        (* 現在の描画を取得 *)
2382        hRun["rendering", 0] = hRun["Rendering"][hRun["q0"], {}];
2383        (* その他パラメータの取得 *)
2384        hRun["fps"] = hConfig["calc/sec"];
2385        (* GUIパラメータを初期化 *)
2386        hRun["show:count"] = 0;           (* 表示カウント値 *)
2387        hRun["show:auto"] = True;         (* 表示カウント値 *)
2388        hRun["locator:enabled"] = False; (* 初期値用ロケータの表示 *)
2389        hRun["locator:sub"] := {};        (* ロケータと重ねるグラフィック *)
2390        hRun["locator:other"] := {};      (* 初期値編集用のグラフィック *)
2391     ];
2392     (*=========================================================================*)
2393     (* 実行状態をスタンバイに設定 *)
2394     hRun["state"] = "standby";
2395     FinishDynamic[];
2396     (* 戻り値 *)
2397     Return@1;
2398  ]
```

[関数定義] LAB`Simulation["start"]
function : 渡された計算設定ハンドルの情報に従ってシミュレーションを開始する
argument1 : 計算設定が保存されている変数群のハンドル
return : 成功 : 1
失敗 : 0

```
2399  LAB`Simulation["start", hConfig_, ___] := With[{
2400     hRun = hConfig["handle:run"], step = hConfig["step"]
2401     }, Module[{
2402     time
2403     },
2404     (*=========================================================================*)
2405     (* 開始ログ *)
2406     LAB`Log["● シミュレーションを開始します : ", hConfig];
2407     (* 実行用ハンドルの状態がスタンバイでない場合は処理を停止 *)
2408     If[! hRun["state"] === "standby",
2409        LAB`Warning["↓ 計算の準備が完了していません"]; Return@0];
2410     (* 目標時間, 目標カウント(整数)を取得 *)
2411     hRun["target:time"] = ToExpression@hRun["target:str"];
2412     hRun["target:count"] = Floor[hConfig["calc/sec"]*hRun["target:time"]];
```

```
    (* 目標時間が数値でない, 計算済み時間より小さい場合は終了 *)
    If[! NumberQ@hRun["target:time"],
       LAB`Warning["↓ 目標時間が不正です"]; Return@0];
    If[TrueQ[hRun["target:count"] <= hRun["count"]],
       LAB`Warning["↓ 指定した目標時間は既に計算済みです"]; Return@0];
    (* 目標時間ログ *)
    LAB`Log["↓ カウント ", hRun["count"] + 1,
       " から ", hRun["target:count"], " まで計算を行います"];
    (*=========================================================================*)
    (* 最新描画を有効にする *)
    hRun["show:auto"] = True;
    (* 実行状態をシミュレーションに設定 *)
    hRun["state"] = "simulation";
    FinishDynamic[];
    (* 繰り返し計算 - 計算にかかった時間を取得する *)
    time = AbsoluteTiming[Do[
       (* 中断するか評価 - 計算状態が"abort"の場合はループを抜ける *)
       If[hRun["state"] === "abort", Break[]];
       (* 記録を有効化, 数値積分, 描画を更新, 計算(済)カウントを更新 *)
       hRun["rec"] = True;
       hRun["qdq"] = hRun["Integration"][hRun["qdq"]];
       hRun["rendering", n] = hRun["Rendering"][hRun["qdq"][[1]], {
          "text"   -> n*step,
          "Cplist" -> Most@hRun["rec:Cp"][n],
          "Fvlist" -> Most@hRun["rec:Fv"][n]
          }];
       hRun["count"] = n;
       (* ループの始点,終点 *)
       , {n, hRun["count"] + 1, hRun["target:count"]}]
    ][[1]];
    (* 終了ログ *)
    time = " - 所要時間 : " <> ToString@time <> " [s]";
    Switch[hRun["state"],
       "simulation", LAB`Log["↓ 計算が正常に終了しました", time],
       "abort", LAB`Log["↓ 計算が中断されました", time]];
    (* 計算済み時間を登録, 実行日時を登録 *)
    hRun["time"] = ToString@LAB`Digit[hRun["count"]*step, 2];
    hRun["date:run"] = DateString[{"Hour24", ":", "Minute", ":", "Second"}];
    (*=========================================================================*)
    (* 実行状態をスタンバイに設定 *)
    hRun["state"] = "standby";
    FinishDynamic[];
    (* 戻り値 *)
    Return@1;
]]
```

[関数定義] LAB`Simulation["analysis"]
    function : シミュレーション結果の処理を行う
    argument1 : 結果処理を行う実行処理ハンドル
    return   : 成功 : 1
                失敗 : 0

```
LAB`Simulation["analysis", hRun_, ___] := Module[{
    mode = ToLowerCase@Mode
    },
    (*=========================================================================*)
    (* 開始ログ *)
    LAB`Log["● 実行結果の解析を行います : ", hRun];
    (* 実行用ハンドルを使用中の場合は処理を停止 *)
    If[LAB`UsingHandle[hRun] > 0,
       LAB`Warning["↓ ハンドルを使用中のため処理を中断します"]; Return@0];
    (* 既に解析済みの場合は終了 *)
```

```
    If[hRun["date:run"] === hRun["date:analysis"],
       LAB`Log["↓ 既に解析済みのため処理を終了します"]; Return@1];
    (* 計算が1ステップも行われていない場合は終了 *)
    If[! NumberQ@hRun["count"] || hRun["count"] === 0,
       LAB`Log["↓ 計算が実行されていないため処理を終了します"]; Return@1];
    (* 実行状態を解析に設定 *)
    hRun["state"] = "set";
    (*=============================================================================*)
    (* 解析対象の実行時間を取得 *)
    hRun["date:analysis"] = hRun["date:run"];
    (*=============================================================================*)
    (* 実行状態をスタンバイに設定 *)
    hRun["state"] = "standby";
    FinishDynamic[];
    (* 戻り値 *)
    Return@1;
]
```

[関数定義] LAB`Simulation["open"]
function : コントロールダイアログを開く
argument1 : 計算設定ハンドル
return : 成功 : 開いたノートブックのハンドル
失敗 : 0

```
LAB`Simulation["open", hConfig_, Option___] := With[{
    hRun = hConfig["handle:run"], w = 785, h = 500,
    option = ToLowerCase@Flatten@{Option}
    },
    (*=============================================================================*)
    (* 実行用ハンドルが登録されていない場合はエラー終了 *)
    If[! Head@hRun === Symbol,
       LAB`Error["● 実行用ハンドルが未登録のためウィンドウを開けません"];
       Return@0;];
    (* 既に開いている場合は閉じる *)
    LAB`Simulation["close", hConfig];

    (*=============================================================================*)
    (* 新規でノートブックを開き, ノートブックハンドルを取得 *)
    hConfig["handle:note"] = CreateDialog[
     TableForm[{
       (* グラフィックス描画 *)
       LAB`Simulation["form:rendering", hConfig],
       (* パラメータ設定タブ *)
       TabView[{
          "control" -> LAB`Simulation["form:control", hConfig],
          "preset"  -> LAB`Simulation["form:preset",  hConfig],
          "config"  -> LAB`Simulation["form:config",  hConfig],
          "result"  -> LAB`Simulation["form:result",  hConfig]
       }, FrameMargins -> 8, ImageSize -> {w - 525, 1000(*h-25*)}]
    }, TableDirections -> Row, TableAlignments -> Top],
    (* サイズ *)
    WindowSize -> {w, h},
    (* 余白 - {{left, right}, {bottom, top}} *)
    WindowMargins -> {{0, Automatic}, {Automatic, 0}},
    (* タイトル *)
    WindowTitle -> hConfig["name"] <>
       " [" <> ToString@hConfig <> ", " <> ToString@hRun <> "]",
    (* フレームに表示する要素 *)
    WindowFrameElements -> {"CloseBox","ZoomBox","MinimizeBox","ResizeArea"},
    WindowElements -> {"MagnificationPopUp"}
    ];

```

```
    (*=========================================================================*)
    (* 戻り値 *)
    Return@hConfig["handle:note"];
]
```

[関数定義] LAB‘Simulation["close"]
    function : コントロールダイアログを閉じる
    argument1 : 計算設定ハンドル / All
    return   : 成功 : 1
                失敗 : 0

```
LAB‘Simulation["close", hConfig_, ___] := Module[{
    hConfigs
    },
    (* 計算設定ハンドルにAllが指定されている場合 *)
    If[hConfig === All,
        hConfigs = LAB‘GetHandle["config", All];
        If[MemberQ[Notebooks[], #["handle:note"]],
            NotebookClose@#["handle:note"]] & /@ hConfigs;
        Return@1
    ];
    (* 計算設定ハンドルが指定されている場合 *)
    If[MemberQ[Notebooks[], hConfig["handle:note"]],
        NotebookClose@hConfig["handle:note"];
        Return@1
    ];
    (* 戻り値 *)
    Return@0
]
```

[関数定義] LAB‘Simulation["form:rendering"]
    function : コントロールダイアログ内の表示要素を生成する
                LAB‘Simulation["open"] 内で呼び出されれる
    argument1 : 計算設定ハンドル
    return   : 生成したテーブルフォーム

```
LAB‘Simulation["form:rendering", hConfig_, ___] := With[{
    hRun = hConfig["handle:run"],
    left = hConfig["left"], right = hConfig["right"],
    bottom = hConfig["bottom"], top = hConfig["top"],
    size = hConfig["size"]
    },
    (* 最新グラフィック表示のチェックボックス値 *)
    hRun["show:auto"] =
        If[NumberQ@hRun["show:count"] && NumberQ@hRun["count"]
            && hRun["show:count"] != hRun["count"], False, True];
    (* 各パラメータを初期化 *)
    hRun["show:count"] = 0;(* 表示カウント値 *)
    hRun["show:auto"] = True;(* 表示カウント値 *)
    hRun["locator:enabled"] = False;(* 初期値用ロケータの表示 *)
    hRun["locator:sub"] := {};(* ロケータと重ねるグラフィック *)
    hRun["locator:other"] := {};(* 初期値編集用のグラフィック *)

    (* フォームを生成 *)
    TableForm[{
        (*-------------------------------------------------------------*)
        TableForm[{"        ",
            (* スライダー : Automatic がチェックされていれば無効 *)
            Slider[Dynamic@hRun["show:count"],
            {0, Dynamic@(If[NumberQ@#, #, 0] &@hRun["count"]), 1},
            Enabled -> Dynamic[! hRun["show:auto"]], ImageSize -> 280],
```

```
2570            (* 選択カウント/現在カウント = スライダー選択値/スライダー最大値 *)
2571            Dynamic@(
2572               hRun["show:count"] = If[hRun["show:auto"],
2573                  hRun["count"], Min[hRun["show:count"], hRun["count"]]];
2574               Row[{
2575                  Text@If[NumberQ@#, #, 0] &@hRun["show:count"], " / ",
2576                  Text@If[NumberQ@#, #, 0] &@hRun["count"]}]),
2577            (* チェックボックス - 常に最新のグラフィックを表示 *)
2578            Row[{Checkbox[Dynamic@hRun["show:auto"]], " Automatic"}]
2579         }, TableDirections -> Row],
2580         (*-------------------------------------------------------------*)
2581         (* グラフィック - 2つのグラフィックを重ねる *)
2582         Dynamic@Show[
2583            (* 標準のグラフィック - 本物/ダミー *)
2584            If[Head@# === Graphics, #,
2585               Graphics[{}, Frame -> True, ImageSize -> size,
2586                  PlotRange -> {{left, right}, {bottom, top}}]
2587            ] &@hRun["rendering", hRun["show:count"]],
2588            (* 初期値編集用グラフィック *)
2589            Graphics[{
2590               If[TrueQ@hRun["locator:enabled"], {
2591                  hRun["locator:other"],
2592                  hRun["locator:sub"],
2593                  Locator@Dynamic@hRun["locator:pos"]
2594               }]
2595            },
2596            PlotRange -> {{left, right}, {bottom, top}},
2597            Axes -> False, Frame -> True, ImageSize -> size]
2598         ]
2599         (*-------------------------------------------------------------*)
2600      }, TableAlignments -> Left, TableSpacing -> 2]
2601   ]
```

[関数定義] LAB‘Simulation["form:control"]
function : コントロールダイアログ内の表示要素を生成する
LAB‘Simulation["open"] 内で呼び出されれる
argument1 : 計算設定ハンドル
return : 生成したテーブルフォーム

```
2602   LAB‘Simulation["form:control", hConfig_, ___] := With[{
2603      hRun = hConfig["handle:run"],
2604      hlist = hConfig["device list"]
2605      }, With[{
2606      hElastic = Select[ToExpression@hlist, #["device model"] === "elastic" &],
2607      hRigid   = Select[ToExpression@hlist, #["device model"] === "rigid" &],
2608      hMotor   = Select[ToExpression@hlist, #["device model"] === "motor" &]
2609      },
2610      (* パラメータの補正 *)
2611      If[! StringQ@hRun[#], hRun[#] = "0"] &@"target:str";
2612      If[! NumberQ@hRun[#], hRun[#] = -1] &@"count";
2613      (* フォームを生成 *)
2614      Return@TableForm[{
2615      (*-------------------------------------------------------------*)
2616      (* ボタン *)
2617      TableForm[{
2618         (* new *)
2619         Dynamic@Button["new", (
2620            LAB‘Simulation["close", hConfig];
2621            LAB‘Simulation["set",   hConfig, "new", "open"];
2622            LAB‘Simulation["open",  hConfig];
2623            ),
2624            Method   -> "Queued",
```

```
2625            Enabled -> Switch[hRun["state"],
2626               "standby", True,
2627               "re-new",  True,
2628               _,         False]
2629         ],
2630         (* analysis *)
2631         Dynamic@Button["analysis", (
2632            LAB`Simulation["analysis", hRun];
2633            LAB`Simulation["open:analysis", hRun];
2634            ),
2635            Method  -> "Queued",
2636            Enabled -> NumberQ@hRun["count"] && hRun["count"] > 0 &&
2637               Switch[hRun["state"],
2638                  "standby", True,
2639                  _,         False]
2640         ],
2641         (* Edit function *)
2642         ActionMenu["preferences", {
2643            "1step" :> (LAB`NewNoteContents[
2644               hRun["str:1step"], "1step", {1000}];
2645            ),
2646            "rendering" :> (LAB`NewNoteContents[
2647               hRun["str:Rendering"], "rendering", {1000}];
2648            ),
2649            "open : current directory" :> (
2650               StartProcess[{"explorer", LAB`NotebookDirectory}];
2651            ),
2652            "open : build directory" :> (
2653               StartProcess[{"explorer", LAB`BuildDirectory}];
2654            )
2655         }]
2656         (*  *)
2657      }, TableDirections -> Row, TableSpacing -> 0.3],
2658      (*-------------------------------------------------------------*)
2659      (* 計算状況を操作 *)
2660      TableForm[{
2661         (* reset *)
2662         Dynamic@Button["reset", (
2663            LAB`Simulation["set", hConfig, "reset"]
2664            ),
2665            Method  -> "Queued",
2666            Enabled -> Switch[hRun["state"],
2667               "standby", True,
2668               _,         False]
2669         ],
2670         (* start *)
2671         Dynamic@Button["start", (
2672            LAB`Simulation["start", hConfig]
2673            ),
2674            Method  -> "Queued",
2675            Enabled -> Switch[hRun["state"],
2676               "standby", True,
2677               _,         False]
2678         ],
2679         (* abort *)
2680         Dynamic@Button["abort", (
2681            hRun["state"] = "abort"
2682            ),
2683            Method  -> "Preemptive",
2684            Enabled -> Switch[hRun["state"],
2685               "simulation", True,
2686               _,            False]
2687         ],
2688         (* state *)
```

```
2689        " state -> ",
2690        Dynamic["[" <> ToString@hRun["state"] <> "]"]
2691        (*  *)
2692     },
2693     TableDirections -> Row,
2694     TableSpacing    -> 0.2
2695     ],
2696     (*------------------------------------------------------------*)
2697     "----------------------------------",
2698     (* 時間パラメータ *)
2699     TableForm[{
2700        (* 現在時間, 目標時間, +-ボタン *)
2701        {  "  Time",
2702           (* 現在時間 *)
2703           Dynamic@StringTake[
2704              ToString@SetPrecision[hConfig["step"]*hRun["count"], 4] <>
2705              "          ", 10],
2706           "->",
2707           (* 目標時間 *)
2708           InputField[Dynamic@hRun["target:str"], String,
2709              ImageSize        -> {50, 17},
2710              Alignment        -> {Left, Center},
2711              ContinuousAction -> True],
2712           (* +-ボタン *)
2713           Row[{
2714              Button["+", hRun["target:str"] = ToString[
2715                 If[NumberQ@#, # + 1, 0] &@ToExpression@hRun["target:str"]],
2716                 ImageSize    -> {16, 16},
2717                 FrameMargins -> 0,
2718                 Method       -> "Preemptive"],
2719              Button["-", hRun["target:str"] = ToString[
2720                 If[NumberQ@#, # - 1, 0] &@ToExpression@hRun["target:str"]],
2721                 ImageSize    -> {16, 16},
2722                 FrameMargins -> 0,
2723                 Method       -> "Preemptive"]
2724           }]
2725        },
2726        (* 現在カウント, 目標カウント *)
2727        {  "  Count",
2728           (* 現在カウント *)
2729           Dynamic@(If[NumberQ@#, #, 0] &@hRun["count"]),
2730           "->",
2731           (* 目標カウント *)
2732           Dynamic@Floor[hConfig["calc/sec"]*
2733              If[NumberQ@#, #, 0] &@ToExpression@hRun["target:str"]]
2734        }
2735     },
2736     TableSpacing    -> {0.5, 1},
2737     TableAlignments -> {Left, Center}
2738     ],
2739     (*------------------------------------------------------------*)
2740     "----------------------------------",
2741     (* 制御項目 *)
2742     TableForm[
2743        Join[{{
2744           Style[" motor target angle", FontFamily -> "Helvetica"], "",
2745           Style["[deg]", FontFamily -> "Helvetica"]
2746           }}, {
2747           (* 名前ボタンリスト *)
2748           Tooltip[Button[
2749              Style[StringTake[#["name"] <> "                    ", 19],
2750                 FontFamily -> "Helvetica", FontColor -> Black],
2751              Null,
2752              Alignment  -> Left,
```

```
2753                ImageSize  -> {110, 18},
2754                Enabled    -> False,
2755                Background -> LightGray, FrameMargins -> 2,
2756                Appearance -> "Palette"
2757                 ], #],
2758             (* スライダー *)
2759             Slider[Dynamic@hRun[#, "target"],
2760                {#["target-min"], #["target-max"], #["target-move"]},
2761                ImageSize -> 80],
2762             (* 数値 *)
2763             Dynamic@hRun[#, "target"]
2764             } & /@ hMotor
2765          ], TableSpacing -> {0.5, 0.7}
2766       ]
2767       (*------------------------------------------------------------*)
2768       },
2769       TableAlignments -> Left,
2770       TableSpacing    -> 0.5
2771       ];
2772   ]]
```

[関数定義] LAB‘Simulation["form:preset"]
function : コントロールダイアログ内の表示要素を生成する
LAB‘Simulation["open"] 内で呼び出されれる
argument1 : 計算設定ハンドル
return : 生成したテーブルフォーム

```
2773   LAB‘Simulation["form:preset", hConfig_, ___] := With[{
2774      hRun = hConfig["handle:run"], mm = 10^(-3),
2775      hlist = hConfig["device list"]
2776      }, With[{
2777      hElastic = Select[ToExpression@hlist, #["device model"] === "elastic" &],
2778      hRigid   = Select[ToExpression@hlist, #["device model"] === "rigid" &],
2779      hMotor   = Select[ToExpression@hlist, #["device model"] === "motor" &]
2780      },
2781      DynamicModule[{
2782      locator, x, y, θ, getDefault, setDefault
2783      },
2784      (* 初期値を取得する関数 & 実行 *)
2785      getDefault[] := (
2786         ({x[#], y[#], θ[#]} =
2787            LAB‘Digit[hRun[#, "q0"]/{mm, mm, Degree}, 1]) & /@ hRigid;
2788         (θ[#] = hRun[#, "q0"][[1]]/Degree) & /@ hMotor;
2789      );
2790      getDefault[];
2791      (* 初期値を変更する関数 *)
2792      setDefault[] := (
2793         (hRun[#, "q0"] = {x[#], y[#], θ[#]}*{mm, mm, Degree}) & /@ hRigid;
2794         (hRun[#, "q0"] = {θ[#]}*Degree) & /@ hMotor;
2795      );
2796
2797      (* フォームを生成 *)
2798      TableForm[{
2799         (*====================================================================*)
2800         (* ボタン *)
2801         TableForm[{
2802            (*------------------------------------------------------------*)
2803            (* 設定ボタン *)
2804            Button[LAB‘Image["setting"], (
2805               LAB‘NewNoteContents[{}, "preset"];
2806               ),
2807               Method       -> "Preemptive",
```

```
2808            ImageSize    -> {22, 22},
2809            Alignment    -> Center,
2810            FrameMargins -> 1
2811         ],
2812         (*-----------------------------------------------------------*)
2813         (* apply & reset - 初期値を変更しリセットする *)
2814         Button["apply & reset", (
2815            setDefault[];
2816            LAB`Simulation["set", hConfig, "reset"];
2817            ),
2818            ImageSize -> {Automatic, 22}
2819         ],
2820         (*-----------------------------------------------------------*)
2821         (* cancel - ロケールを無効化,変更前の初期値を取得 *)
2822         Button["cancel", (
2823            hRun["locator:enabled"] = False;
2824            locator = Null;
2825            getDefault[];
2826            ),
2827            ImageSize -> {Automatic, 22}
2828         ]
2829         (*  *)
2830      }, TableDirections -> Row, TableSpacing -> 0.3],
2831      (*=============================================================*)
2832      "---------------------------------",
2833      (* rigid *)
2834      TableForm[Join[{If[
2835         Length@hRigid == 0,
2836            " \"rigid\" does not exist",
2837            {Style[" \"rigid\"", FontFamily -> "Helvetica"],
2838            Style[" x [mm]    y [mm]    θ [deg]",
2839            FontFamily -> "Helvetica"]}
2840         ]}, {
2841         (*-----------------------------------------------------------*)
2842         (* 名前ボタン *)
2843         Tooltip[Button[
2844            Style[StringTake[#["name"] <> "               ", 15],
2845            FontFamily -> "Helvetica"], (
2846            (* 初期描画を表示する *)
2847            hRun["show:auto"] = False;
2848            hRun["show:count"] = 0;
2849            (* 選択中か非選択中かで分岐 *)
2850            If[locator === #,
2851               (* 選択中 -> 非選択 - ロケール表示を無効化 *)
2852               hRun["locator:enabled"] = False;
2853               locator = Null;,
2854               (* 非選択 -> 選択中 - ロケール表示を有効化 *)
2855               hRun["locator:enabled"] = True;
2856               locator = #;
2857               (*-------------------------------------------------*)
2858               (* その他のグラフィックの表示する関数を定義 *)
2859               hRun["locator:other"] := Module[{list, angle, pos},
2860                  list = {
2861                     pos = {x[#], y[#]}*{mm, mm};
2862                     angle = If[NumberQ@#, #, 0] &@θ[#]*Degree;
2863                     Opacity[0],
2864                     EdgeForm[{Thickness[0.004], Red, Dashing[Tiny]}],
2865                     Table[Disk[pos + #["point"][[n]][[;; 2]],
2866                        #["point radius"][[n]]], {n, Length@#["point"]}],
2867                     Polygon[(pos+RotationMatrix@angle.#) & /@ #["polygon"]]
2868                  } & /@ hRigid;
2869                  Delete[list, #] &@FirstPosition[hRigid, #][[1]]
2870               ];
2871               (*-------------------------------------------------*)
```

```
2872                (* ロケール座標を取得 *)
2873                If[! NumberQ@x[#], x[#] = 0];
2874                If[! NumberQ@y[#], y[#] = 0];
2875                hRun["locator:pos"] = {x[#], y[#]}*mm;
2876                (*-----------------------------------------------------*)
2877                (* ロケールに追従するグラフィック用の関数を定義 *)
2878                hRun["locator:sub"] := Module[{pos, angle}, {
2879                   pos = hRun["locator:pos"];
2880                   angle = If[NumberQ@#, #, 0] &@@θ[#]*Degree;
2881                   Opacity[0],
2882                   EdgeForm[{Thickness[0.004], Red}],
2883                   Table[Disk[pos + #["point"][[n]][[;; 2]],
2884                      #["point radius"][[n]]], {n, Length@#["point"]}],
2885                   Polygon[(pos+RotationMatrix@angle.#) & /@ #["polygon"]]
2886                }]
2887                (*-----------------------------------------------------*)
2888             ];
2889             ),
2890             Alignment    -> Left,
2891             ImageSize    -> {91, 17},
2892             Background   -> Dynamic@If[locator === #, LightRed, LightGray],
2893             Method       -> "Preemptive",
2894             FrameMargins -> 2,
2895             Appearance   -> "Palette"
2896          ], #],
2897          (*-------------------------------------------------------------*)
2898          (* x,y,θ *)
2899          Dynamic@If[locator === #,
2900             (* 選択されている項目 *)
2901             If[hRun["locator:enabled"] === False,
2902                locator = Null,
2903                {x[#], y[#]} = LAB`Digit[hRun["locator:pos"]/mm, 1]
2904             ];
2905             Row[{
2906                InputField[Dynamic@x[#],
2907                   ImageSize -> {47, 17},
2908                   Alignment -> {Left, Center},
2909                   Enabled   -> False],
2910                InputField[Dynamic@y[#],
2911                   ImageSize -> {47, 17},
2912                   Alignment -> {Left, Center},
2913                   Enabled   -> False],
2914                InputField[Dynamic@θ[#],
2915                   ImageSize -> {44, 17},
2916                   Alignment -> {Left, Center},
2917                   Enabled   -> True]
2918             }],
2919             (* 選択されていない項目 *)
2920             Row[{
2921                InputField[Dynamic@x[#],
2922                   ImageSize -> {47, 17},
2923                   Alignment -> {Left, Center}],
2924                InputField[Dynamic@y[#],
2925                   ImageSize -> {47, 17},
2926                   Alignment -> {Left, Center}],
2927                InputField[Dynamic@θ[#],
2928                   ImageSize -> {44, 17},
2929                   Alignment -> {Left, Center}]
2930             }]
2931          ]
2932          (*-------------------------------------------------------------*)
2933          } & /@ hRigid
2934       ],
2935       TableSpacing -> {0, 0},
```

```
      TableAlignments -> {Left, Center}
      ],
      (*====================================================================*)
      "----------------------------------",
      (* motor *)
      TableForm[Join[{If[Length@hMotor == 0,
         " \"motor\" does not exist",
         {Style[" \"motor\"", FontFamily -> "Helvetica"], "",
            Style["[deg]", FontFamily -> "Helvetica"]}
         ]}, {
         (* 名前ボタンリスト *)
         Tooltip[Button[
            Style[StringTake[#["name"] <> "                    ", 19],
               FontFamily -> "Helvetica", FontColor -> Black],
            Null,
            Alignment    -> Left,
            ImageSize    -> {110, 18},
            Enabled      -> False,
            Background   -> LightGray,
            FrameMargins -> 2,
            Appearance   -> "Palette"
         ], #],
         (* スライダー *)
         Slider[Dynamic@θ[#],
            {#["target-min"], #["target-max"], #["target-move"]},
            ImageSize -> 71],
         (* 入力フィールド *)
         InputField[Dynamic@θ[#],
            ImageSize -> {44, 17},
            Alignment -> {Left, Center},
            Enabled   -> False]
         (*  *)
         } & /@ hMotor
      ],
      TableSpacing    -> {0.2, 0.5},
      TableAlignments -> {Left, Center}
      ]
      (*====================================================================*)
   },
   TableAlignments -> Left,
   TableSpacing -> 0.5
   ]
]]]
```

[関数定義] LAB‘Simulation["form:config"]
    function : コントロールダイアログ内の表示要素を生成する
                LAB‘Simulation["open"] 内で呼び出されれる
    argument1 : 計算設定ハンドル
    return   : 生成したテーブルフォーム

```
LAB‘Simulation["form:config", hConfig_, ___] := With[{
   hRun    = hConfig["handle:run"],
   hDevice = LAB‘SortHandle@LAB‘GetHandle["device", All],
   hSystem = LAB‘SortHandle@LAB‘GetHandle["system", All]
   }, DynamicModule[{
   enable, delete, update, str = ""
   },
   (* 削除対象の状況を取得 *)
   (delete[#] = False) & /@ hDevice; (delete[#] = False) & /@ hSystem;
   (* 更新対象の状況を取得 *)
   (update[#] = False) & /@ hDevice; (update[#] = False) & /@ hSystem;
   (* デバイスハンドル・システムハンドルのリスト登録状況を取得 *)
```

```
2991    (enable[#] = MemberQ[hConfig["device list"], #]) & /@ hDevice;
2992    (enable[#] = MemberQ[hConfig["system list"], #]) & /@ hSystem;
2993
2994    (* フォームを生成 *)
2995    TableForm[{
2996       (*=====================================================================*)
2997       (* ボタン *)
2998       TableForm[{
2999          (* 設定ボタン *)
3000          Button[LAB`Image["setting"], (
3001             LAB`NewNoteContents[hConfig["Cells"], hConfig["name"]];
3002             ),
3003             Method       -> "Preemptive",
3004             ImageSize    -> {22, 22},
3005             Alignment    -> Center,
3006             FrameMargins -> 1
3007          ],
3008          (* apply & new : ハンドルリストを取得し,新規で初期化する *)
3009          Button["apply & new", (
3010             LAB`Simulation["close", hConfig];
3011             If[delete[#], enable[#] = False; ClearAll@#] & /@ hDevice;
3012             If[delete[#], enable[#] = False; ClearAll@#] & /@ hSystem;
3013             hConfig["device list"] =
3014                LAB`SortHandle@Select[hDevice, enable[#] &];
3015             hConfig["system list"] =
3016                LAB`SortHandle@Select[hSystem, enable[#] &];
3017             LAB`Simulation["set", hConfig, "new", "open"];
3018             LAB`Simulation["open", hConfig];
3019             ),
3020             Method    -> "Queued",
3021             ImageSize -> {Automatic, 22},
3022             Enabled   -> Switch[hRun["state"],
3023                "standby", True,
3024                "re-new",  True,
3025                _,         False]
3026          ],
3027          (* cancel : 変更項目を元に戻す *)
3028          Button["cancel", (
3029             (delete[#] = False) & /@ hDevice;
3030             (delete[#] = False) & /@ hSystem;
3031             (enable[#] = MemberQ[hConfig["device list"], #]) & /@ hDevice;
3032             (enable[#] = MemberQ[hConfig["system list"], #]) & /@ hSystem;
3033             ),
3034             ImageSize -> {Automatic, 22}
3035          ],
3036          (* 状態を知らせる文字列 *)
3037          "",
3038          Dynamic@str
3039          (*  *)
3040       },
3041       TableDirections -> Row,
3042       TableSpacing    -> 0.3
3043       ],
3044       "----------------------------------",
3045       (*=====================================================================*)
3046       (* デバイス一覧の説明 *)
3047       Row[{
3048          "device list   ",
3049          Button["",
3050             Null,
3051             ImageSize  -> {20, 13},
3052             Enabled    -> False,
3053             Appearance -> "Palette",
3054             Background -> LightRed
```

```
3055            ],
3056            " : in use   ",
3057            Style["name", FontFamily -> "Helvetica", Red, Bold],
3058            " : updated",
3059            Button["",
3060               Null,
3061               ImageSize  -> {1, 18},
3062               Enabled    -> False,
3063               Appearance -> "Palette",
3064               Background -> White
3065            ]
3066         }],
3067         (* デバイス一覧 *)
3068         Grid[{
3069            (*-----------------------------------------------------------------*)
3070            (* 使用チェック *)
3071            Checkbox[Dynamic@enable[#]], "",
3072            (*-----------------------------------------------------------------*)
3073            (* ハンドル *)
3074            Button[Style[StringTake[ToString@#, 7 ;;],
3075                  FontColor -> Black, FontFamily -> "Helvetica"],
3076               (Null),
3077               Alignment -> Left,
3078               ImageSize -> {40, 17},
3079               Enabled -> False,
3080               FrameMargins -> 2,
3081               Appearance -> "Palette",
3082               Background ->
3083                  If[MemberQ[hConfig["device list"], #], LightRed, White]
3084            ],
3085            (*-----------------------------------------------------------------*)
3086            (* 名前 *)
3087            Button[Style[#["name"], FontFamily -> "Helvetica",
3088                  FontColor  -> Dynamic@If[update[#], Red,  Black],
3089                  FontWeight -> Dynamic@If[update[#], Bold, Plain]],
3090               (Null),
3091               Alignment -> Left,
3092               ImageSize -> {125, 17},
3093               Enabled -> False,
3094               FrameMargins -> 2,
3095               Appearance -> "Palette",
3096               Background ->
3097                  If[MemberQ[hConfig["device list"], #], LightRed, White]
3098            ],
3099            "",
3100            (*-----------------------------------------------------------------*)
3101            TableForm[{
3102               (* 更新ボタン *)
3103               Tooltip[Button[LAB`Image["update"], (
3104                  If[LAB`UsingHandle[#] === 0,
3105                     If[!LAB`Autocomplete["Auto",#["Source"],#["name"]]===0,
3106                        update[#] = True; hRun["state"] = "re-new";
3107                        str = Style["Please apply!", Red, Bold];
3108                     ]
3109                  ];
3110                  ),
3111                  Method       -> "Preemptive",
3112                  ImageSize    -> {17, 17},
3113                  Alignment    -> Center,
3114                  FrameMargins -> 0
3115               ],
3116               #["Source"]
3117               ],
3118               (* 設定ボタン *)
```

```
3119            Button[LAB`Image["setting"], (
3120               LAB`NewNoteContents[#["Cells"], #["name"]];
3121               ),
3122               Method       -> "Preemptive",
3123               ImageSize    -> {17, 17},
3124               Alignment    -> Center,
3125               FrameMargins -> 0
3126            ],
3127            (* 削除ボタン *)
3128            Button[LAB`Image["cross"], (
3129               If[delete[#], delete[#] = False,
3130               If[LAB`UsingHandle[#] === 0,
3131                  delete[#] = True;
3132                  enable[#] = False]];
3133               ),
3134               Method       -> "Preemptive",
3135               ImageSize    -> {17, 17},
3136               Background   -> Dynamic@If[delete[#], Red, Automatic],
3137               Alignment    -> Center,
3138               FrameMargins -> 0
3139            ]
3140         },
3141         TableDirections -> Row,
3142         TableSpacing    -> 0
3143         ]
3144         (*-----------------------------------------------------------*)
3145      } & /@ hDevice,
3146      Alignment -> {Left, Center},
3147      Dividers  -> {False, All},
3148      Spacings  -> {0.1, 0.2}
3149      ],
3150      (*=============================================================*)
3151      (* システム一覧の説明 *)
3152      Row[{
3153         "system list   ",
3154         Button["",
3155            Null,
3156            ImageSize  -> {20, 13},
3157            Enabled    -> False,
3158            Appearance -> "Palette",
3159            Background -> LightRed
3160         ],
3161         " : in use   ",
3162         Style["name", FontFamily -> "Helvetica", Red, Bold],
3163         " : updated",
3164         Button["",
3165            Null,
3166            ImageSize  -> {1, 18},
3167            Enabled    -> False,
3168            Appearance -> "Palette",
3169            Background -> White
3170         ]
3171      }],
3172      (* システム一覧 *)
3173      Grid[{
3174         (*-----------------------------------------------------------*)
3175         (* 使用チェック *)
3176         Checkbox[Dynamic@enable[#]],
3177         (*-----------------------------------------------------------*)
3178         (* ハンドル *)
3179         Button[Style[StringTake[ToString@#, 7 ;;],
3180               FontColor -> Black, FontFamily -> "Helvetica"],
3181            (Null),
3182            Alignment    -> Left,
```

```
3183            ImageSize    -> {40, 17},
3184            Enabled      -> False,
3185            FrameMargins -> 2,
3186            Appearance   -> "Palette",
3187            Background   ->
3188               If[MemberQ[hConfig["system list"], #], LightRed, White]
3189         ],
3190         (*------------------------------------------------------------*)
3191         (* 名前 *)
3192         Button[Style[#["name"], FontFamily -> "Helvetica",
3193               FontColor -> Dynamic@If[update[#], Red, Black],
3194               FontWeight -> Dynamic@If[update[#], Bold, Plain]],
3195            (Null),
3196            Alignment    -> Left,
3197            ImageSize    -> {125, 17},
3198            Enabled      -> False,
3199            FrameMargins -> 2,
3200            Appearance   -> "Palette",
3201            Background   ->
3202               If[MemberQ[hConfig["system list"], #], LightRed, White]
3203         ],
3204         (*------------------------------------------------------------*)
3205         TableForm[{
3206            (* 更新ボタン *)
3207            Tooltip[Button[LAB`Image["update"], (
3208               If[LAB`UsingHandle[#] === 0,
3209                  If[!LAB`Autocomplete["Auto",#["Source"],#["name"]]===0,
3210                     update[#] = True;
3211                     hRun["state"] = "re-new";
3212                     str = Style["Please apply!", Red, Bold];
3213                  ]
3214               ];
3215               ),
3216               Method       -> "Preemptive",
3217               ImageSize    -> {17, 17},
3218               Alignment    -> Center,
3219               FrameMargins -> 0
3220            ],
3221            #["Source"]
3222            ],
3223            (* 設定ボタン *)
3224            Button[LAB`Image["setting"], (
3225               LAB`NewNoteContents[#["Cells"], #["name"]];
3226               ),
3227               Method       -> "Preemptive",
3228               ImageSize    -> {17, 17},
3229               Alignment    -> Center,
3230               FrameMargins -> 0
3231            ],
3232            (* 削除ボタン *)
3233            Button[LAB`Image["cross"], (
3234               If[delete[#], delete[#] = False,
3235                  If[LAB`UsingHandle[#] === 0,
3236                     delete[#] = True;
3237                     enable[#] = False
3238                  ]
3239               ];
3240               ),
3241               Method       -> "Preemptive",
3242               ImageSize    -> {17, 17},
3243               Background   -> Dynamic@If[delete[#], Red, Automatic],
3244               Alignment    -> Center,
3245               FrameMargins -> 0
3246            ]
```

```
3247            },
3248            TableDirections -> Row,
3249            TableSpacing    -> 0
3250            ]
3251            (*------------------------------------------------------------*)
3252        } & /@ hSystem,
3253        Alignment -> {Left, Center},
3254        Dividers -> {False, All},
3255        Spacings -> {0.1, 0.2}
3256        ]
3257        (*============================================================*)
3258    },
3259    TableAlignments -> Left,
3260    TableSpacing -> 0.5
3261    ]
3262 ]]
```

[関数定義] LAB‘Simulation["form:result"]
function : コントロールダイアログ内の表示要素を生成する
LAB‘Simulation["open"] 内で呼び出されれる
argument1 : 計算設定ハンドル
return : 生成したテーブルフォーム

```
3263 LAB‘Simulation["form:result", hConfig_, ___] := With[{
3264    }, DynamicModule[{
3265    hRuns
3266    },
3267    (* ハンドルリストを取得 *)
3268    hRuns = Select[LAB‘GetHandle["run", All], IntegerQ@#["count"] &];
3269
3270    (* フォームを生成 *)
3271    TableForm[{
3272       (*------------------------------------------------------------*)
3273       (* リスト更新ボタン, オールクリアボタン *)
3274       TableForm[{
3275          (* list update : ハンドルリストを取得 *)
3276          Button["list update", (
3277             hRuns = Select[LAB‘GetHandle["run", All], IntegerQ@#["count"] &];
3278             ),
3279             Method -> "Preemptive"
3280          ],
3281          (* all clear : 現在のハンドル以外をクリア,ハンドルリストを取得 *)
3282          Button["all clear", (
3283             If[! hConfig["handle:run"] === #, ClearAll[#]] & /@ hRuns;
3284             hRuns = Select[LAB‘GetHandle["run", All], IntegerQ@#["count"] &];
3285          ),
3286             Method -> "Preemptive"
3287          ]
3288          (*  *)
3289       },
3290       TableDirections -> Row,
3291       TableSpacing    -> 0.5
3292       ],
3293       (*------------------------------------------------------------*)
3294       (* リスト *)
3295       Dynamic@Grid[Join[{{
3296          Style[" restore ",  FontFamily -> "Helvetica"],
3297          Style[" last run ", FontFamily -> "Helvetica"],
3298          Style[" time    ",  FontFamily -> "Helvetica"],
3299          Style[" count ",    FontFamily -> "Helvetica"]
3300          }}, {
3301          (* 復元ボタン *)
```

```
3302            Button[Style[ToString@#, FontFamily -> "Helvetica", Blue], (
3303               LAB`Simulation["close", hConfig];
3304               LAB`Simulation["set",   hConfig, "restore", #];
3305               LAB`Simulation["open",  hConfig];
3306               ),
3307               Alignment    -> Left,
3308               ImageSize    -> {50, 15},
3309               Background   -> If[hConfig["handle:run"] === #, LightRed, White],
3310               Enabled      -> If[hConfig["handle:run"] === #, False, True],
3311               FrameMargins -> 2,
3312               Appearance   -> "Palette",
3313               Method       -> "Queued"
3314            ],
3315            (* 登録日時, 計算時間, 計算カウント *)
3316            Style[#["date:run"], FontFamily -> "Helvetica"],
3317            Tooltip[
3318               Style[#["time"], FontFamily -> "Helvetica"],
3319               Magnify[#["rendering", #["count"]], 0.5]],
3320            Style[#["count"], FontFamily -> "Helvetica"],
3321            (* クリアボタン *)
3322            Button[LAB`Image["cross"], (
3323               ClearAll[#] &@#;
3324               hRuns = Select[LAB`GetHandle["run", All], IntegerQ@#["count"] &];
3325               ),
3326               Method       -> "Preemptive",
3327               ImageSize    -> {15, 15},
3328               Enabled      -> If[hConfig["handle:run"] === #, False, True],
3329               Alignment    -> Center,
3330               FrameMargins -> 0
3331            ]
3332            (*  *)
3333            } & /@ Reverse@hRuns
3334         ],
3335         Alignment -> {{Left, Left, Left, Right}, Center},
3336         Dividers  -> {Automatic, {2 -> True}},
3337         Frame     -> {All, False}
3338         ]
3339         (*------------------------------------------------------------*)
3340      },
3341      TableAlignments -> Left,
3342      TableSpacing    -> 2
3343      ]
3344   ]]
```

[関数定義] LAB`Simulation["open:analysis"]
    function : シミュレーション結果ダイアログを開く
    argument1 : 実行処理ハンドル
    return   : 成功 : 開いたノートブックのハンドル
                失敗 : 0

```
3345   LAB`Simulation["open:analysis", hRun_, Option___] := With[{
3346      w = 1050, h = 700,
3347      option = ToLowerCase@Flatten@{Option}
3348      },
3349      (*=========================================================================*)
3350      (* 実行用ハンドルが登録されていない場合はエラー終了 *)
3351      If[! Head@hRun === Symbol,
3352         LAB`Error["● 実行用ハンドルが未登録のためウィンドウを開けません"];
3353         Return@0;];
3354      (* 解析結果が生成されていない場合は終了 *)
3355      If[! StringQ@hRun["date:analysis"] ||
3356         hRun["date:analysis"] === "---",
```

```
3357        LAB`Warning["● 解析結果が生成されていないためウィンドウを開けません"];
3358        Return@0;];
3359    (* 既に開いている場合は閉じる *)
3360    LAB`Simulation["close:analysis", hRun];
3361
3362    (*=========================================================================*)
3363    (* 新規でノートブックを開き, ノートブックハンドルを取得 *)
3364    hRun["handle:note"] = CreateDialog[
3365        TableForm[{
3366            (* グラフィックス描画 *)
3367            LAB`Simulation["form:animation", hRun],
3368            (* パラメータ設定タブ *)
3369            TabView[{
3370                "control" -> ""
3371            },
3372            FrameMargins -> 8,
3373            ImageSize -> {w - 540, 1000}
3374            ]
3375        },
3376        TableDirections -> Row,
3377        TableAlignments -> Top
3378        ],
3379        (* 背景色 *)
3380        Background -> LightBlue,
3381        (* サイズ *)
3382        WindowSize -> {w, h},
3383        (* 余白 - {{left, right}, {bottom, top}} *)
3384        WindowMargins -> {{Automatic, Automatic}, {0, Automatic}},
3385        (* タイトル *)
3386        WindowTitle -> "analysis [" <> ToString@hRun <> ", date : " <>
3387            hRun["date:analysis"] <> "]",
3388        (* フレームに表示する要素 *)
3389        WindowFrameElements ->
3390            {"CloseBox", "ZoomBox", "MinimizeBox", "ResizeArea"},
3391        WindowElements ->
3392            {"MagnificationPopUp", "HorizontalScrollBar", "VerticalScrollBar"}
3393    ];
3394    (*=========================================================================*)
3395    (* 戻り値 *)
3396    Return@hRun["handle:note"];
3397 ]
```

[関数定義] LAB`Simulation["close:analysis"]
function : シミュレーション結果ダイアログを閉じる
argument1 : 実行処理ハンドル / All
return : 成功 : 1
失敗 : 0

```
3398 LAB`Simulation["close:analysis", hRun_, ___] := Module[{
3399     hRuns
3400     },
3401     (* 計算設定ハンドルにAllが指定されている場合 *)
3402     If[hRun === All,
3403         hRuns = LAB`GetHandle["run", All];
3404         If[MemberQ[Notebooks[], #["handle:note"]],
3405             NotebookClose@#["handle:note"]
3406         ] & /@ hRuns;
3407         Return@1
3408     ];
3409     (* 計算設定ハンドルが指定されている場合 *)
3410     If[MemberQ[Notebooks[], hRun["handle:note"]],
3411         NotebookClose@hRun["handle:note"];
```

```
3412        Return@1
3413    ];
3414    (* 戻り値 *)
3415    Return@0
3416 ]
```

[関数定義] LAB‘Simulation["form:animation"]
    function : シミュレーション結果ダイアログ内の表示要素を生成する
                LAB‘Simulation["open:analysis"] 内で呼び出されれる
    argument1 : 実行処理ハンドル
    return   : 生成したテーブルフォーム

```
3417 LAB‘Simulation["form:animation", hRun_, ___] := With[{
3418    count = hRun["count"],
3419    fps   = hRun["fps"],
3420    time  = hRun["count"]/hRun["fps"] // N
3421    },
3422    TableForm[{
3423    (*--------------------------------------------------------------------*)
3424    Manipulate[
3425       hRun["rendering", n], {
3426          (* 操作する変数, 初期位置, コントロールラベル *)
3427          {n, 1, ""},
3428          (* 最小値, 最大値, 刻み幅 *)
3429          0, count, 1,
3430          (* アニメーションタイプ : Manipulator / Animator *)
3431          ControlType -> Animator,
3432          (* 表示時に実行中にするかどうか *)
3433          AnimationRunning -> False,
3434          (* 毎秒表示枚数 *)
3435          AnimationRate -> fps
3436       },
3437       (* ステップ / 時間 *)
3438       Column[{
3439          Row@{Text["  step : "], Dynamic@Text[n],       " / ", Text[count]},
3440          Row@{Text["  time : "], Dynamic@Text@N[n/fps], " / ", Text[time] }
3441       }],
3442       (* フレームの表示/非表示 *)
3443       Paneled -> False,
3444       (* フレームとの余白 *)
3445       FrameMargins -> 0,
3446       ImageMargins -> 0
3447    ]
3448    (*--------------------------------------------------------------------*)
3449    },
3450    TableAlignments -> Left,
3451    TableSpacing    -> 2
3452    ]
3453 ]
```

付録 B

Mathematica によるプログラム

- シミュレーションの実行例 -

Mathematica 上で実行するコード（付録 A の続き）

```
LAB‘Autocomplete["All", "mjg.ini"];
config = LAB‘GetHandle[Null, "calc setting"];
LAB‘Simulation["set",  config, "new"];
LAB‘Simulation["open", config];
```

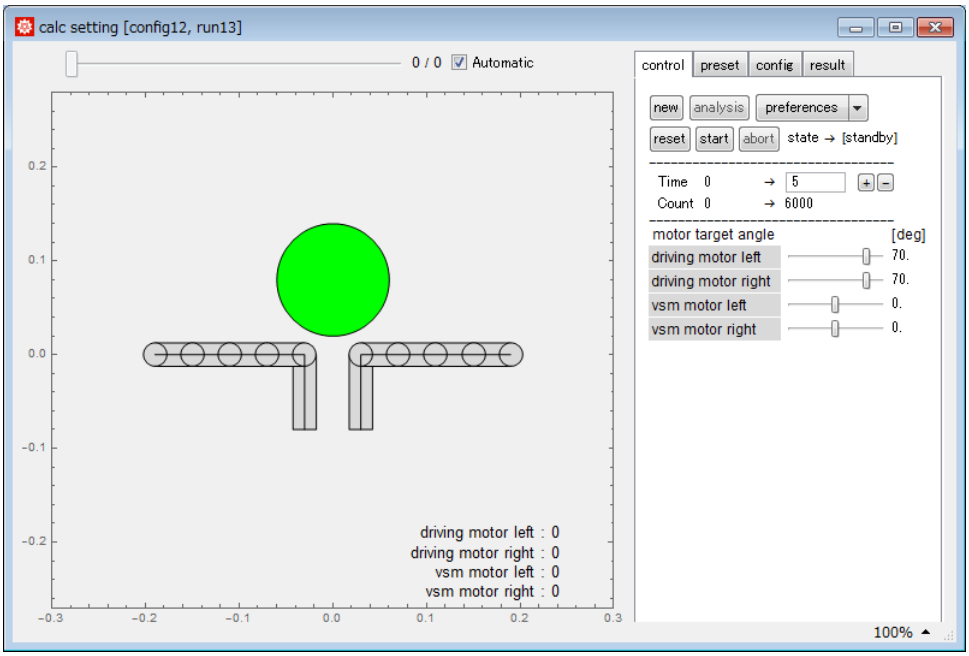
計算パラメータが記述されている INI ファイル

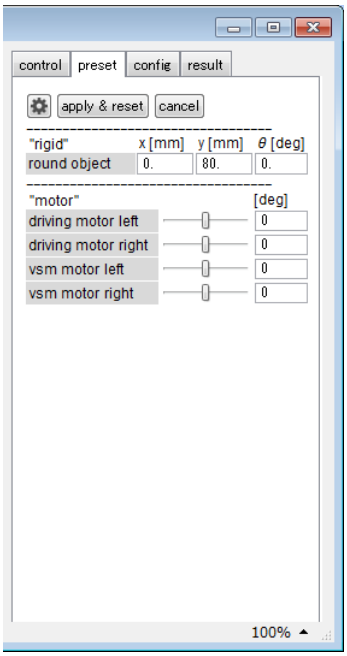
```
[calc setting]
data model              =config
target time             =5                      &int
integration method      =RK4
calc/sec                =1200                   &int
calc/rec                =1                      &int
contents                ={right finger,driving motor right,vsm motor right,
    dgs-vsm right,left finger,driving motor left,vsm motor left,dgs-vsm left
    ,round object}
[right finger]
data model              =device
device model            =link
dof                     =4                      &int
length of link          ={40,40,40,40}          &float  &mm
rate of center          ={0.5,0.5,0.5,0.5}      &float
basic height            ={25,25,25,25}          &float  &mm
mass                    ={100,100,100,100}      &float  &g
moment of inertia       ={50,50,50,50}          &float  &micro
damping of rotation     ={0.03,0.03,0.03,0.03}  &float
damping of translation  ={0,0,0,0}              &float
ref pos                 ={30,0}                 &float  &mm
ref angle               =0                      &float  &degree
base length             =80                     &float  &mm
base height             =25                     &float  &mm
base angle              =90                     &float  &degree
[left finger]
data model              =device
device model            =link
dof                     =4                      &int
length of link          ={40,40,40,40}          &float  &mm
rate of center          ={0.5,0.5,0.5,0.5}      &float
basic height            ={25,25,25,25}          &float  &mm
mass                    ={100,100,100,100}      &float  &g
moment of inertia       ={50,50,50,50}          &float  &micro
damping of rotation     ={0.03,0.03,0.03,0.03}  &float
damping of translation  ={0,0,0,0}              &float
ref pos                 ={-30,0}                &float  &mm
ref angle               =180                    &float  &degree
base length             =80                     &float  &mm
base height             =25                     &float  &mm
base angle              =-90                    &float  &degree
[driving motor right]
```

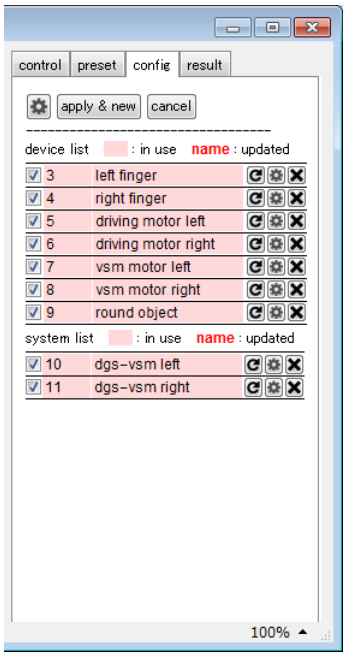
```
data model              =device
device model            =motor
moment of inertia       =1                      &float
damping of rotation     =1                      &float
target                  =70                     &float
target-max              =100                    &float
target-min              =-100                   &float
target-move             =0.1                    &float
method                  =P
kp                      =1                      &float
kd                      =0                      &float
ki                      =0                      &float
maximum output          =5                      &float
[vsm motor right]
data model              =device
device model            =motor
moment of inertia       =1                      &float
damping of rotation     =1                      &float
target                  =0                      &float
target-max              =100                    &float
target-min              =-100                   &float
target-move             =0.1                    &float
method                  =P
kp                      =1                      &float
kd                      =0                      &float
ki                      =0                      &float
maximum output          =5                      &float
[driving motor left]
data model              =device
device model            =motor
moment of inertia       =1                      &float
damping of rotation     =1                      &float
target                  =70                     &float
target-max              =100                    &float
target-min              =-100                   &float
target-move             =0.1                    &float
method                  =P
kp                      =1                      &float
kd                      =0                      &float
ki                      =0                      &float
maximum output          =5                      &float
[vsm motor left]
data model              =device
device model            =motor
moment of inertia       =1                      &float
damping of rotation     =1                      &float
target                  =0                      &float
target-max              =100                    &float
target-min              =-100                   &float
target-move             =0.1                    &float
method                  =P
kp                      =1                      &float
kd                      =0                      &float
ki                      =0                      &float
maximum output          =5                      &float
[dgs-vsm right]
data model              =system
system model            =dgs-vsm
conect-link             =right finger
conect-drive            =driving motor right
conect-vsm              =vsm motor right
method                  =standard
spring modulus          ={1.0, 1.0, 1.0, 1.0}   &float ;[N/mm]
initial tension         ={0.4, 0.4, 0.4, 0.4}   &float ;[N]
```

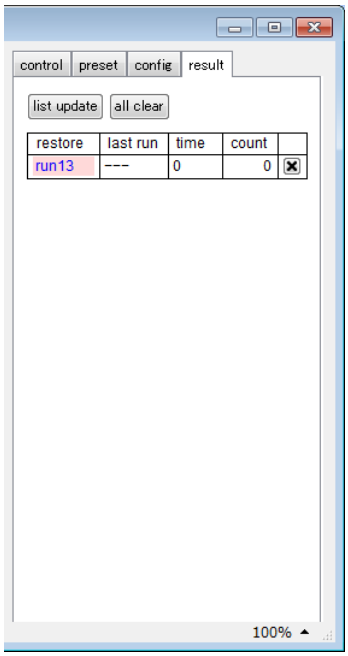
```
initial length          ={0.0, 0.0, 0.0, 0.0}   &float ;[mm]
1st pulley r            ={4.0, 4.0, 4.0, 4.0}   &float ;[mm]
vsm pulley r            ={4.0, 4.0, 4.0, 4.0}   &float ;[mm]
zl                      =30                     &int
zs                      =20                     &int
direction               =1                      &int
[dgs-vsm left]
data model              =system
system model            =dgs-vsm
conect-link             =left finger
conect-drive            =driving motor left
conect-vsm              =vsm motor left
method                  =standard
spring modulus          ={1.0, 1.0, 1.0, 1.0}   &float ;[N/mm]
initial tension         ={0.4, 0.4, 0.4, 0.4}   &float ;[N]
initial length          ={0.0, 0.0, 0.0, 0.0}   &float ;[mm]
1st pulley r            ={4.0, 4.0, 4.0, 4.0}   &float ;[mm]
vsm pulley r            ={4.0, 4.0, 4.0, 4.0}   &float ;[mm]
zl                      =30                     &int
zs                      =20                     &int
direction               =-1                     &int
[round object]
data model              =device
device model            =rigid
mass                    =100                    &float  &g
moment of inertia       =1                      &float  &micro
damping of rotation     =0.001                  &float
damping of translation  =0.3                    &float
pos                     ={0,80}                 &float  &mm
angle                   =0                      &float  &degree
form                    =round
radius                  =60                     &float  &mm
```

シミュレーションプログラムで生成される GUI









INI ファイルの情報から自動で生成されるコード - 1 ステップを計算する関数
※ 改行位置やインデントは修正

[関数定義] run*["1step"]
function : 1 ステップを計算する関数
argument1 : 一般化座標, 一般化速度のリスト
return : 一般化速度, 一般化加速度のリスト

```
run13["1step"] = Compile[{{arg, _Real, 2}
}, With[{
    (* 置換変数 *)
   q1 = arg[[1, 1 ;; 4]],
   q2 = arg[[1, 5 ;; 8]],
   q3 = arg[[1, 9 ;; 9]],
   q4 = arg[[1, 10 ;; 10]],
   q5 = arg[[1, 11 ;; 11]],
   q6 = arg[[1, 12 ;; 12]],
   q7 = arg[[1, 13 ;; 15]],
   dq1 = arg[[2, 1 ;; 4]],
   dq2 = arg[[2, 5 ;; 8]],
   dq3 = arg[[2, 9 ;; 9]],
   dq4 = arg[[2, 10 ;; 10]],
   dq5 = arg[[2, 11 ;; 11]],
   dq6 = arg[[2, 12 ;; 12]],
   dq7 = arg[[2, 13 ;; 15]],
   (* left finger [=1] *)
   dof$1 = 4,
   q0$1 = 3.141592653589793,
   x0y0$1 = {-0.03, 0.},
   link$1 = {0.04, 0.04, 0.04, 0.04},
   polygon$1 = {
      {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
      {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
      {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
      {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
      {{-0.0175, -0.08}, {-0.0425, -0.08},
       {-0.042499999999999996, 7.654042494670958*^-19},
       {-0.017499999999999998, -7.654042494670958*^-19}}},
   nPolygon$1 = 5,
   jPolygon$1 = {1, 2, 3, 4, 0},
   lineEnable$1 = {{0, 1, 0, 1}, {0, 1, 0, 1}, {0, 1, 0, 1},
      {0, 1, 0, 1}, {0, 1, 0, 1}},
   angle$1 = {
      {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
      {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
      {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
      {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
      {3.141592653589793, 1.5707963267948966,
       -6.123233995736767*^-17, -1.5707963267948966}},
   length$1 = {
      {0.025, 0.04, 0.025, 0.04},
      {0.025, 0.04, 0.025, 0.04},
      {0.025, 0.04, 0.025, 0.04},
      {0.025, 0.04, 0.025, 0.04},
      {0.025, 0.08, 0.024999999999999998, 0.08}},
   point$1 = {
      {0, 0}, {0, 0}, {0, 0}, {0, 0}, {-0.0175, -0.08}, {-0.0425, -0.08},
      {-0.042499999999999996, 7.654042494670958*^-19},
      {-0.017499999999999998, -7.654042494670958*^-19}, {0.04, 0}},
   nPoint$1 = 9,
   jPoint$1 = {1, 2, 3, 4, 0, 0, 0, 0, 4},
   pointEnable$1 = {1, 0, 0, 0, 0, 0, 0, 0, 1},
   radius$1 = {0.0125, 0.0125, 0.0125, 0.0125, 0, 0, 0, 0, 0.0125},
```

```
56      (* right finger [=2] *)
57      dof$2 = 4,
58      q0$2 = 0.,
59      x0y0$2 = {0.03, 0.},
60      link$2 = {0.04, 0.04, 0.04, 0.04},
61      polygon$2 = {
62         {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
63         {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
64         {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
65         {{0, -0.0125}, {0, 0.0125}, {0.04, 0.0125}, {0.04, -0.0125}},
66         {{0.042499999999999996, -0.08}, {0.017499999999999995, -0.08},
67          {0.017499999999999998, 7.654042494670958*^-19},
68          {0.042499999999999996, -7.654042494670958*^-19}}},
69      nPolygon$2 = 5,
70      jPolygon$2 = {1, 2, 3, 4, 0},
71      lineEnable$2 = {{0, 1, 0, 1}, {0, 1, 0, 1}, {0, 1, 0, 1},
72         {0, 1, 0, 1}, {0, 1, 0, 1}},
73      angle$2 = {
74         {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
75         {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
76         {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
77         {1.5707963267948966, 0., -1.5707963267948966, 3.141592653589793},
78         {3.141592653589793, 1.5707963267948966,
79          -6.123233995736767*^-17, -1.5707963267948966}},
80      length$2 = {
81         {0.025, 0.04, 0.025, 0.04},
82         {0.025, 0.04, 0.025, 0.04},
83         {0.025, 0.04, 0.025, 0.04},
84         {0.025, 0.04, 0.025, 0.04},
85         {0.025, 0.08, 0.024999999999999998, 0.08}},
86      point$2 = {
87         {0, 0}, {0, 0}, {0, 0}, {0, 0},
88         {0.042499999999999996, -0.08}, {0.017499999999999995, -0.08},
89         {0.017499999999999998, 7.654042494670958*^-19},
90         {0.042499999999999996, -7.654042494670958*^-19}, {0.04, 0}},
91      nPoint$2 = 9,
92      jPoint$2 = {1, 2, 3, 4, 0, 0, 0, 0, 4},
93      pointEnable$2 = {1, 0, 0, 0, 0, 0, 0, 0, 1},
94      radius$2 = {0.0125, 0.0125, 0.0125, 0.0125, 0, 0, 0, 0, 0.0125},
95      (* round object [=7] *)
96      point$7 = {{0, 0}},
97      nPoint$7 = 1,
98      pointEnable$7 = {1},
99      radius$7 = {0.06},
100     (* 計算カウント *)
101     count = Round[run13["count"] + 1]
102 }, Module[{
103     (* 局所変数 *)
104     ddq, jnum, θ, origin, vector, deform, Cp, Fm, Fv,
105     Cplist = {{0., 0.}}, Fmlist = {0.}, Fvlist = {{0., 0.}},
106     absq1, jpos1, polygon1, angle1, point1,
107     absq2, jpos2, polygon2, angle2, point2,
108     pos7, polygon7, angle7, point7,
109     Q1 = {0., 0., 0., 0.},
110     Q2 = {0., 0., 0., 0.},
111     Q3 = {0.},
112     Q4 = {0.},
113     Q5 = {0.},
114     Q6 = {0.},
115     Q7 = {0., 0., 0.}
116 },
117     (*--------------------------------------------------------------------*)
118     (* left finger [=1] *)
119     absq1 = q0$1 + Accumulate[q1];
```

```
120     jpos1 = FoldList[Plus, x0y0$1,
121        Table[link$1[[n]]*{Cos[absq1[[n]]], Sin[absq1[[n]]]}, {n, dof$1}]];
122     polygon1 = Table[jnum = jPolygon$1[[n]];
123        If[jnum == 0,
124           polygon$1[[n, i]], jpos1[[jnum]] + {
125              polygon$1[[n, i, 1]]*Cos[absq1[[jnum]]] -
126              polygon$1[[n, i, 2]]*Sin[absq1[[jnum]]],
127              polygon$1[[n, i, 2]]*Cos[absq1[[jnum]]] +
128              polygon$1[[n, i, 1]]*Sin[absq1[[jnum]]]
129           }
130         ], {n, nPolygon$1}, {i, 4}];
131     angle1 = Table[jnum = jPolygon$1[[n]];
132        If[jnum == 0,
133           angle$1[[n]],
134           absq1[[jnum]] + angle$1[[n]]
135        ], {n, nPolygon$1}];
136     point1 = Table[jnum = jPoint$1[[n]];
137        If[jnum == 0,
138           point$1[[n]],
139           jpos1[[jnum]] + {
140              point$1[[n, 1]]*Cos[absq1[[jnum]]] -
141              point$1[[n, 2]]*Sin[absq1[[jnum]]],
142              point$1[[n, 2]]*Cos[absq1[[jnum]]] +
143              point$1[[n, 1]]*Sin[absq1[[jnum]]]
144           }
145        ], {n, nPoint$1}];
146     (* right finger [=2] *)
147     absq2 = q0$2 + Accumulate[q2];
148     jpos2 = FoldList[Plus, x0y0$2,
149        Table[link$2[[n]]*{Cos[absq2[[n]]], Sin[absq2[[n]]]}, {n, dof$2}]];
150     polygon2 = Table[jnum = jPolygon$2[[n]];
151        If[jnum == 0,
152           polygon$2[[n, i]], jpos2[[jnum]] + {
153              polygon$2[[n, i, 1]]*Cos[absq2[[jnum]]] -
154              polygon$2[[n, i, 2]]*Sin[absq2[[jnum]]],
155              polygon$2[[n, i, 2]]*Cos[absq2[[jnum]]] +
156              polygon$2[[n, i, 1]]*Sin[absq2[[jnum]]]
157           }
158        ], {n, nPolygon$2}, {i, 4}];
159     angle2 = Table[jnum = jPolygon$2[[n]];
160        If[jnum == 0,
161           angle$2[[n]],
162           absq2[[jnum]] + angle$2[[n]]
163        ], {n, nPolygon$2}];
164     point2 = Table[jnum = jPoint$2[[n]];
165        If[jnum == 0,
166           point$2[[n]],
167           jpos2[[jnum]] + {
168              point$2[[n, 1]]*Cos[absq2[[jnum]]] -
169              point$2[[n, 2]]*Sin[absq2[[jnum]]],
170              point$2[[n, 2]]*Cos[absq2[[jnum]]] +
171              point$2[[n, 1]]*Sin[absq2[[jnum]]]
172           }
173        ], {n, nPoint$2}];
174     (* driving motor left [=3] *)
175     Q3 += device5["eq:q2Q"][q3, run13[device5, "target"]];
176     (* driving motor right [=4] *)
177     Q4 += device6["eq:q2Q"][q4, run13[device6, "target"]];
178     (* vsm motor left [=5] *)
179     Q5 += device7["eq:q2Q"][q5, run13[device7, "target"]];
180     (* vsm motor right [=6] *)
181     Q6 += device8["eq:q2Q"][q6, run13[device8, "target"]];
182     (* dgs-vsm left [=8] *)
183     Q1 += system10["eq:q2Q"][q1, q3, q5];
```

```
184     (* dgs-vsm right [=9] *)
185     Q2 += system11["eq:q2Q"][q2, q4, q6];
186     (* round object [=7] *)
187     pos7 = {q7[[1]], q7[[2]]};
188     point7 = Table[pos7 + {
189        point$7[[n, 1]]*Cos[q7[[3]]] - point$7[[n, 2]]*Sin[q7[[3]]],
190        point$7[[n, 2]]*Cos[q7[[3]]] + point$7[[n, 1]]*Sin[q7[[3]]]
191     }, {n, nPoint$7}];
192
193     (*-------------------------------------------------------------------------*)
194     (* contact : left finger [=1] & left finger [=1] *)
195
196     (*-------------------------------------------------------------------------*)
197     (* contact : left finger [=1] & right finger [=2] *)
198
199     (*-------------------------------------------------------------------------*)
200     (* contact : left finger [=1] & round object [=7] *)
201     (* link-line & rigid-point *)
202     Do[If[lineEnable$1[[k, i]] == pointEnable$7[[n]] != 0,
203         θ = angle1[[k, i]]; origin = polygon1[[k, i]];
204        (* ラインとポイントの貫入量より接触を評価 *)
205        vector = {
206           {Cos[-θ], -Sin[-θ]}, {Sin[-θ], Cos[-θ]}}.(point7[[n]] - origin);
207        deform = radius$7[[n]] - vector[[2]];
208        If[0 < vector[[1]] < length$1[[k, i]] && 0 <= deform < 0.02,
209           (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
210           Cp = origin + {Cos[θ], Sin[θ]}*vector[[1]];
211           Fm = deform*5000;
212           Fv = {Sin[θ], -Cos[θ]}*Fm;
213           PrependTo[Cplist, Cp];
214           PrependTo[Fmlist, Fm];
215           PrependTo[Fvlist, Fv];
216           (* 一般化力へ反映 *)
217           Q1 += Table[
218              If[j <= jPolygon$1[[k]],
219                 (Cp - jpos1[[j]]).{Fv[[2]], -Fv[[1]]}, 0
220              ], {j, dof$1}];
221           Q7 += {-Fv[[1]], -Fv[[2]], (Cp - pos7).{-Fv[[2]], Fv[[1]]}};
222        ]
223     ], {k, nPolygon$1}, {i, 4}, {n, nPoint$7}];
224     (* link-point & rigid-point *)
225     Do[If[pointEnable$1[[n1]] == pointEnable$7[[n2]] != 0,
226        (* ポイントとポイントの貫入量より接触を評価 *)
227        deform = radius$7[[n2]] + radius$1[[n1]]
228                 - Norm[point7[[n2]] - point1[[n1]]];
229        If[0 < deform < 0.02,
230           (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
231           vector = Normalize[point7[[n2]] - point1[[n1]]];
232           Cp = point1[[n1]] + vector*radius$1[[n1]];
233           Fm = deform*5000;
234           Fv = -vector*Fm;
235           PrependTo[Cplist, Cp];
236           PrependTo[Fmlist, Fm];
237           PrependTo[Fvlist, Fv];
238           (* 一般化力へ反映 *)
239           Q1 += Table[
240              If[j <= jPoint$1[[n1]],
241                 (Cp - jpos1[[j]]).{Fv[[2]], -Fv[[1]]}, 0
242              ], {j, dof$1}];
243           Q7 += {-Fv[[1]], -Fv[[2]], (Cp - pos7).{-Fv[[2]], Fv[[1]]}};
244        ]
245     ], {n1, nPoint$1}, {n2, nPoint$7}];
246
247     (*-------------------------------------------------------------------------*)
```

```
248     (* contact : right finger [=2] & right finger [=2] *)
249
250     (*-------------------------------------------------------------------*)
251     (* contact : right finger [=2] & round object [=7] *)
252     (* link-line & rigid-point *)
253     Do[If[lineEnable$2[[k, i]] == pointEnable$7[[n]] != 0,
254        θ = angle2[[k, i]];
255        origin = polygon2[[k, i]];
256        (* ラインとポイントの貫入量より接触を評価 *)
257        vector = {
258           {Cos[-θ], -Sin[-θ]}, {Sin[-θ], Cos[-θ]}}.(point7[[n]] - origin);
259        deform = radius$7[[n]] - vector[[2]];
260        If[0 < vector[[1]] < length$2[[k, i]] && 0 <= deform < 0.02,
261           (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
262           Cp = origin + {Cos[θ], Sin[θ]}*vector[[1]];
263           Fm = deform*5000;
264           Fv = {Sin[θ], -Cos[θ]}*Fm;
265           PrependTo[Cplist, Cp];
266           PrependTo[Fmlist, Fm];
267           PrependTo[Fvlist, Fv];
268           (* 一般化力へ反映 *)
269           Q2 += Table[
270              If[j <= jPolygon$2[[k]],
271                 (Cp - jpos2[[j]]).{Fv[[2]], -Fv[[1]]}, 0
272              ], {j, dof$2}];
273           Q7 += {-Fv[[1]], -Fv[[2]], (Cp - pos7).{-Fv[[2]], Fv[[1]]}};
274        ]
275     ], {k, nPolygon$2}, {i, 4}, {n, nPoint$7}];
276     (* link-point & rigid-point *)
277     Do[If[pointEnable$2[[n1]] == pointEnable$7[[n2]] != 0,
278        (* ポイントとポイントの貫入量より接触を評価 *)
279        deform = radius$7[[n2]] + radius$2[[n1]]
280                 - Norm[point7[[n2]] - point2[[n1]]];
281        If[0 < deform < 0.02,
282           (* Cp:接触点座標, Fm:接触力, Fv:接触力ベクトル rigid->link *)
283           vector = Normalize[point7[[n2]] - point2[[n1]]];
284           Cp = point2[[n1]] + vector*radius$2[[n1]];
285           Fm = deform*5000;
286           Fv = -vector*Fm;
287           PrependTo[Cplist, Cp];
288           PrependTo[Fmlist, Fm];
289           PrependTo[Fvlist, Fv];
290           (* 一般化力へ反映 *)
291           Q2 += Table[
292              If[j <= jPoint$2[[n1]],
293                 (Cp - jpos2[[j]]).{Fv[[2]], -Fv[[1]]}, 0
294              ], {j, dof$2}];
295           Q7 += {-Fv[[1]], -Fv[[2]], (Cp - pos7).{-Fv[[2]], Fv[[1]]}};
296        ]
297     ], {n1, nPoint$2}, {n2, nPoint$7}];
298
299     (*-------------------------------------------------------------------*)
300     (* contact : round object [=7] & round object [=7] *)
301
302     (*-------------------------------------------------------------------*)
303     (* 記録 *)
304     If[run13["rec"],
305        run13["rec"] = False;
306        run13["rec:q", "left finger"][count] = q1;
307        run13["rec:q", "right finger"][count] = q2;
308        run13["rec:q", "driving motor left"][count] = q3;
309        run13["rec:q", "driving motor right"][count] = q4;
310        run13["rec:q", "vsm motor left"][count] = q5;
311        run13["rec:q", "vsm motor right"][count] = q6;
```

```
312         run13["rec:q", "round object"][count] = q7;
313         run13["rec:Cp"][count] = Cplist;
314         run13["rec:Fm"][count] = Fmlist;
315         run13["rec:Fv"][count] = Fvlist;
316     ];
317     (* 加速度 *)
318     ddq = Join[
319         device3["eq:ddq"][q1, dq1, Q1](* left finger *),
320         device4["eq:ddq"][q2, dq2, Q2](* right finger *),
321         device5["eq:ddq"][q3, dq3, Q3](* driving motor left *),
322         device6["eq:ddq"][q4, dq4, Q4](* driving motor right *),
323         device7["eq:ddq"][q5, dq5, Q5](* vsm motor left *),
324         device8["eq:ddq"][q6, dq6, Q6](* vsm motor right *),
325         device9["eq:ddq"][q7, dq7, Q7](* round object *)
326     ];
327     (* 戻り値 *)
328     {arg[[2]], ddq}
329 ]], {
330     (* コンパイルオプション *)
331     {run13["count"], _Integer},
332     {run13["rec"], True | False},
333     {_Symbol["eq:q2Q"][___], _Real, 1},
334     {_Symbol["eq:ddq"][___], _Real, 1}
335 }, CompilationTarget -> "C"]
```

INI ファイルの情報から自動で生成されるコード - 一般化座標からグラフィックを生成する関数
※ 改行位置やインデントは修正

[関数定義] run*["Rendering"]
function : 一般化座標からグラフィックを生成
argument1 : 一般化座標
return : 生成したグラフィック

```
336 run13["Rendering"] = Function[{arg, Option},
337 Module[{
338     q1 = arg[[1 ;; 4]],
339     q2 = arg[[5 ;; 8]],
340     q3 = arg[[9 ;; 9]],
341     q4 = arg[[10 ;; 10]],
342     q5 = arg[[11 ;; 11]],
343     q6 = arg[[12 ;; 12]],
344     q7 = arg[[13 ;; 15]],
345     text = "text" /. Option,
346     Cplist = "Cplist" /. Option,
347     Fvlist = "Fvlist" /. Option,
348     j
349 },
350     If[text === "text", text = ""];
351     If[! ListQ@Cplist, Cplist = {}];
352     If[! ListQ@Fvlist, Fvlist = {}];
353
354     (* グラフィック生成 *)
355     Graphics[{
356         (* left finger *)
357         LightGray,
358         EdgeForm[{Thickness[0.002], Black}],
359         Polygon[#] &@device3["eq:q2polygon"][q1],
360         Disk[#[[;; 2]], #[[3]]] & /@ device3["eq:q2point"][q1],
361         Black,
362         Line[device3["eq:q2j"][q1]],
363         (* right finger *)
364         LightGray,
```

```
      EdgeForm[{Thickness[0.002], Black}],
      Polygon[#] &@device4["eq:q2polygon"][q2],
      Disk[#[[;; 2]], #[[3]]] & /@ device4["eq:q2point"][q2],
      Black,
      Line[device4["eq:q2j"][q2]],
      (* driving motor left *)
      Text[Style[SetPrecision[q3[[1]]/Degree // N, 3], 13],
         {0.23399999999999999, -0.201}, {-1, -1}],
      Text[Style[":", Black, 13],
         {0.22799999999999998, -0.201}, {1, -1}],
      Text[Style["driving motor left", Black, 13],
         {0.21599999999999997, -0.201}, {1, -1}],
      (* driving motor right *)
      Text[Style[SetPrecision[q4[[1]]/Degree // N, 3], 13],
         {0.23399999999999999, -0.222}, {-1, -1}],
      Text[Style[":", Black, 13],
         {0.22799999999999998, -0.222}, {1, -1}],
      Text[Style["driving motor right", Black, 13],
         {0.21599999999999997, -0.222}, {1, -1}],
      (* vsm motor left *)
      Text[Style[SetPrecision[q5[[1]]/Degree // N, 3], 13],
         {0.23399999999999999, -0.24300000000000002}, {-1, -1}],
      Text[Style[":", Black, 13],
         {0.22799999999999998, -0.24300000000000002}, {1, -1}],
      Text[Style["vsm motor left", Black, 13],
         {0.21599999999999997, -0.24300000000000002}, {1, -1}],
      (* vsm motor right *)
      Text[Style[SetPrecision[q6[[1]]/Degree // N, 3], 13],
         {0.23399999999999999, -0.264}, {-1, -1}],
      Text[Style[":", Black, 13],
         {0.22799999999999998, -0.264}, {1, -1}],
      Text[Style["vsm motor right", Black, 13],
         {0.21599999999999997, -0.264}, {1, -1}],
      (* round object *)
      Green,
      EdgeForm[{Thickness[0.002], Black}],
      Polygon[#] &@device9["eq:q2polygon"][q7],
      Disk[#[[;; 2]], #[[3]]] & /@ device9["eq:q2point"][q7],
      (* contact force *)
      Red,
      Thick,
      Disk[#, 0.0025] & /@ Cplist,
      Arrow[#] & /@ Transpose[{Cplist, 0.10*Fvlist + Cplist}],
      (* time *)
      Text[Style[ToString[text], Black, 13],
         {-0.282, 0.262}, {-1, 1}]
   },
   (* グラフィックス設定 *)
   PlotRange -> {{-0.3, 0.3}, {-0.27, 0.28}},
   ImageSize -> 500,
   Axes -> False,
   Frame -> True
]]];
```