東京大学
情報理工学系研究科 創造情報学専攻
博士論文

実体設計評価機構を備えた
ロボットの身体行動創成支援システムの構成法

Configuration method of body and behavior creation support system for robot with design and embodiment evaluation mechanism

野田 晋太朗
Noda, Shintaro

指導教員　稲葉雅幸教授

2017 年 12 月

概要

本研究ではロボット開発における歩行制御系といった運動モデルのパラメタ生成、仕様を設計に落とし込む工程でのリンク長といった身体モデルのパラメタ決定、行動実現時に身体や環境、運動モデルのパラメタを実物のロボットに合わせて修正する作業をそれぞれ探索問題としてとらえ支援する身体行動創成支援システムの構成法について論じる。ロボット開発前の設計段階では、運動モデルと身体モデルが与えられたとしてそれぞれのパラメタを探索することで設計を助ける問題が支援例として考えられる。ロボット開発後の行動実現段階では、実際にロボットを動作させ運動の結果を確認し、それに応じて人間がロボットの身体モデル、環境モデルのパラメタを修正することによって運動を作り直す工程が必要であるが、この工程も身体環境モデルのパラメタを探索する問題としてとらえることが可能である。これらの探索問題は従来の運動モデルのパラメタ探索のみ考慮していた探索手法から身体環境モデルのパラメタ探索も含む拡張が行われている点で大きく異なっている。

このような問題を効率よく解くためには非線形最適化手法や進化計算、動力学シミュレータといった各種の探索手法が必要である。姿勢の探索に限れば勾配法が有効であることが知られているが、環境との干渉回避には不向きでありサンプリング手法との組み合わせが必要となる。また幾何的な条件だけでなく力学的な条件を考慮し関節負荷トルクを最小化するような問題まで考慮するには探索手法の高速化も課題である。あるいは接触遷移を伴うような運動では、接触を保つ姿勢列を扱うために局所探索手法が有効であり、微分が難しい接触状態探索を別の問題としてグラフ探索手法などにより解くほうが効率的である。さらに制御系のパラメタや身体環境モデルのパラメタなど勾配法により探索できないパラメタが多数含まれる場合は進化計算が有効であり、動力学シミュレータや高速評価可能な運動モデルも必要になる。

本論文で構成する身体行動創成支援システムは、問題の性質に従い使い分けるための各種探索手法を有するとともに、ロボット完成前の設計段階と完成後の行動実現段階のそれぞれを支援するために、それら探索手法を使い分ける機構（実体設計評価機構）を有する。本論文の各章では設計支援、行動実現支援をそれぞれ異なる探索手法で解く例を示し、実際に脚型ロボットを開発するなかで実証的に支援システムを評価する。検証内容の中でも特に、従来開発の難しかった受動機械要素の設計支援と、受動機械要素を用いた運動モデルのパラメタ生成、実ロボットの身体環境運動モデルのパラメタ修正により準受動歩行行動を実現し、本支援システムがこれまでにないロボットを創成する力をもつことを示す。本支援システムはロボット開発を効率化し設計の自由度を広げるものである。

目次

第 1 章　序論 1
1.1　背景と目的 1
1.2　ロボット開発支援に有効な探索問題についての考察 3
1.2.1　全自由度空間を探索範囲とする探索 4
1.2.2　探索範囲を狭めない条件つき探索 5
1.2.3　実ロボットでの探索された行動の忠実な再現 6
1.2.4　リンク質量等の身体環境モデルのパラメタ探索 7
1.2.5　ロボットの運動から身体まで探索するシステム 7
1.3　本論文の構成 8

第 2 章　ロボットの身体行動創成支援システム構成法 10
2.1　はじめに 10
2.2　探索空間から見たロボット開発の流れと既存研究 10
2.3　等身大脚型準受動歩行ロボット設計開発を例としたロボット開発の流れ . . . 15
2.3.1　仕様から自明でない身体モデルのパラメタ探索 16
2.3.2　設計と異なる実体身体環境モデルのパラメタ推定と動作再生成 18
2.3.3　パラメタの空間を定義するモデルと設計の制約である仕様 21
2.4　身体行動創成支援システムの構造 21
2.4.1　身体運動モデルのパラメタ生成と実体の身体環境モデルのパラメタ修正からなる実体設計評価機構 21
2.4.2　身体行動創成支援システムで扱うモデル空間 23
2.4.2.1　身体環境モデル空間 24
2.4.2.2　運動モデル空間 24
2.4.2.3　物理演算器モデル空間 24
2.4.2.4　運動結果空間 24
2.4.3　実体設計評価機構が扱う探索問題の分類 25
2.4.3.1　運動モデル空間での探索 26
2.4.3.2　身体環境モデル空間での探索 27
2.4.3.3　身体環境運動モデル空間での探索 28

2.5 各章における実体設計評価機構の使い方と意義 28
2.6 おわりに 30

第 3 章 全身姿勢探索による片足支持に適した股関節間距離の決定 31
3.1 はじめに 31
3.2 オートエンコーダを用いた冗長埋め込みによる高速な初期値非依存探索法 . . 32
3.2.1 冗長埋め込みと探索空間削減 33
3.2.1.1 冗長埋め込み： タスク空間とタスク冗長空間からロボットの状態空間への写像 33
3.2.1.2 探索空間削減： 探索空間を状態空間からタスク冗長空間へ . 34
3.2.2 Task-State Map の学習実験： 単腕・双腕・全身のキネマティクス . . 37
3.2.2.1 学習設定とアルゴリズム 37
3.2.2.2 ネットワークパラメタに対する Task-State Map 性能評価 . 38
3.2.3 探索実験: 狭い隙間からの脱出と遮蔽物裏へのリーチング 43
3.2.3.1 狭い空間からの脱出タスク 43
3.2.3.2 遮蔽物裏へのリーチングタスク 45
3.2.4 学習を用いた初期値非依存な探索高速化手法についてのまとめ 46
3.3 低計算コストな擬似関節負荷トルク勾配を用いた高速な低負荷姿勢探索法 . . 47
3.3.1 接触のモデルと実現可能なロボットの状態 48
3.3.1.1 接触状態のモデルと滑り・転がりに関する不等式拘束 48
3.3.1.2 実現可能なロボットの状態 50
3.3.2 問題設定: 姿勢最適化問題の定式化とアルゴリズム 52
3.3.2.1 負荷指標と姿勢最適化 52
3.3.2.2 逆運動学と接触力最適化の繰り返し解法 54
3.3.3 関節負荷トルク勾配の定義と計算量 57
3.3.3.1 厳密な関節負荷トルク勾配: 定義と計算量 57
3.3.3.2 擬似関節負荷トルク勾配: 接触力近似による計算高速化 . . . 60
3.3.3.3 計算時間の測定 64
3.3.3.4 関節負荷トルク勾配の零点について 64
3.3.4 擬似負荷トルク勾配を用いた姿勢探索性能の評価実験 65
3.3.4.1 実験に用いる姿勢探索アルゴリズム 66
3.3.4.2 進化計算アルゴリズムを用いたテストデータ生成 67
3.3.4.3 姿勢探索性能の評価実験の実験結果 67
3.3.5 擬似関節負荷トルク勾配についてのまとめ 73
3.3.6 関節負荷トルク勾配計算法についての補足 74
3.3.6.1 １自由度回転ジョイント系の高速な同次変換行列勾配計算法 74
3.3.6.2 空間に固定された点のヤコビ行列 75
3.4 全身姿勢探索による片足支持に適した股関節間距離の決定 76

3.5 おわりに 78

第 4 章 接触遷移行動探索による着座支持棒長さの決定 79

4.1 はじめに 79
4.2 複数の接触遷移方式探索機能を有する行動生成法 80
4.2.1 複数の接触遷移方式探索機能を有する行動生成手法の構成 82
4.2.2 複数の接触遷移方式探索機能を有する行動生成手法の実装 83
4.2.2.1 接触状態選択について 83
4.2.2.2 接触状態選択の探索アルゴリズム 87
4.2.2.3 動作生成について 88
4.2.3 複数の接触遷移方式探索機能を用いた行動生成実験 93
4.2.3.1 立ち上がり行動生成実験 93
4.2.3.2 シートへの座り行動生成 95
4.2.4 実等身大ヒューマノイドにおける滑り接触遷移を用いた摩擦パラメタ推定と着座行動実験 96
4.2.5 複数の接触遷移方式探索機能を有する行動生成法についてのまとめ . . 100
4.3 動力学シミュレーションを用いた接触行動同時探索法 101
4.3.1 高速計算可能な簡易動力学シミュレータ構成 102
4.3.1.1 動力学シミュレータの全体構成 102
4.3.1.2 順運動学 102
4.3.1.3 逆動力学 106
4.3.1.4 順動力学 109
4.3.1.5 接触判定と接触力計算 114
4.3.1.6 センサシミュレーション 116
4.3.1.7 速度評価実験 118
4.3.2 動歩行探索のための軽量な歩行モデル 120
4.3.2.1 既存の動歩行モデルのまとめ 120
4.3.2.2 ステートマシン型歩行モデル 122
4.3.2.3 歩き初めと歩き終わりを考慮したステート遷移 122
4.3.2.4 位置制御のための線形補間 123
4.3.2.5 ヨー回転のフードバック補償 124
4.3.2.6 足平と体幹の回転補正 124
4.3.2.7 受動軸の振りぬき速度を変更可能な遊脚軌道 124
4.3.2.8 重心位置計算に置ける姿勢推定アルゴリズム 125
4.3.2.9 フィードバックゲインの関節軸方向による次元削減 125
4.3.3 動力学シミュレーションを用いた動歩行行動制御探索法 126
4.3.3.1 人間によるパラメタの決定と歩行実験 126
4.3.3.2 省エネルギー歩行のための評価関数 130

4.3.3.3　実現可能な歩行生成のための条件 133
4.3.3.4　動力学シミュレーション上での進化計算による動歩行行動制御探索実験 134
4.3.3.5　歩行の安定性について 135
4.4　接触遷移行動探索による着座支持棒長さの決定 137
4.5　おわりに 139

第 5 章　運動結果距離最小化による歩行行動の忠実再現 141
5.1　はじめに 141
5.2　受動軸を備える脚型ロボットの身体構成・ソフトウェア構成 141
5.2.1　ハードウェア構成 141
5.2.1.1　概要 141
5.2.1.2　既存ロボット JAXON との比較 144
5.2.1.3　関節配置 145
5.2.1.4　電装系 146
5.2.1.5　リンクの質量・重心・慣性テンソル 146
5.2.2　制御フロー 146
5.2.3　オープンソース制御系を用いた動歩行実験と消費エネルギー量 150
5.3　動力学演算を用いた実体の身体環境モデルのパラメタ修正 152
5.3.1　実体モデルパラメタ修正に基づく行動の忠実再現法 152
5.3.1.1　探索空間：実体の身体環境モデルのパラメタについて 152
5.3.1.2　評価関数：再現度評価のための運動結果距離関数について . 153
5.3.2　実体モデルパラメタ修正に基づく行動の忠実再現実験 153
5.3.2.1　発見的方法による実体の身体環境モデルのパラメタ推定と行動再生成実験 154
5.3.2.2　実体の身体環境モデルのパラメタ推定と行動再生成実験 . . . 157
5.4　おわりに 160
5.5　補足：動力学演算を用いた実体の身体環境モデルのパラメタ修正実験ログ . . 161
5.5.1　歩行動作の各ステート姿勢と歩行の様子 161
5.5.2　実行された運動結果ログ 163
5.5.2.1　シミュレーションでの膝曲げ歩行 164
5.5.2.2　実ロボットでの膝曲げ歩行 165
5.5.2.3　実体の身体環境モデルのパラメタ推定後のシミュレーションでの膝曲げ歩行 166
5.5.2.4　シミュレーションでの膝曲げ歩行重心補正 20 mm 167
5.5.2.5　実ロボットでの膝曲げ歩行重心補正 20 mm 168
5.5.2.6　実体の身体環境モデルのパラメタ推定後のシミュレーションでの膝曲げ歩行重心補正 20 mm 169

5.5.2.7 シミュレーションでの最適化された膝伸ばし歩行 170
5.5.2.8 実ロボットでの最適化された膝伸ばし歩行 171
5.5.2.9 実体の身体環境モデルのパラメタ推定後のシミュレーションでの最適化された膝伸ばし歩行 172
5.5.2.10 シミュレーションでの再度最適化された膝伸ばし歩行 173
5.5.2.11 実ロボットでの再度最適化された膝伸ばし歩行 174
5.5.2.12 実体の身体環境モデルのパラメタ推定後のシミュレーションでの再度最適化された膝伸ばし歩行 175
5.5.3 実体の身体環境モデルのパラメタ推定計算時の距離関数グラフ 176
5.5.4 推定された実体の身体環境モデルのパラメタ 177

第 6 章 身体運動モデルのパラメタ同時探索による準受動歩行支持バネ係数決定と行動生成制御 183
6.1 はじめに 183
6.2 省エネルギー歩行のための拘束解放可変機構 184
6.2.1 受動軸の有無からみた動歩行研究と受動歩行 184
6.2.2 受動歩行を能動歩行ロボットへ応用するための課題とクラッチ機構 . . 186
6.2.3 アクチュエータと角位置センサを有しソフトウェアにより受動軸を追従制御する水平ピンクラッチ機構 187
6.2.4 一軸試験機の開発 188
6.2.4.1 体積について 188
6.2.4.2 強度について 189
6.2.4.3 ピンの傾斜について 189
6.2.5 PD 制御を用いた同期制御評価予備実験 190
6.2.6 同期精度向上のためのカルマンフィルタを用いたリンク位置の予測 . . 191
6.2.6.1 動作モデル 192
6.2.6.2 センサモデル 192
6.2.6.3 拡張カルマンフィルタ 192
6.2.6.4 追従制御実験の結果 193
6.2.7 等身大脚型ロボットへのクラッチ機構適用に向けた展望 194
6.2.8 ML1 への拘束解放可変機構の適用 196
6.3 拘束解放可変機構を用いた歩行行動探索法と実験 198
6.3.1 クラッチの拘束解放を切り替えながら歩行する能動受動歩行 198
6.3.2 体幹リンクを有するロボットの受動歩行モデル 202
6.3.3 バネとクラッチを用いた準受動歩行行動探索実験 202
6.3.3.1 運動モデル空間と身体環境モデル空間からなる探索空間について 202
6.3.3.2 準受動歩行行動の探索実験 203

6.3.4 実ロボットによるバネとクラッチを用いた準受動歩行行動実験 206
6.3.5 実ロボットによる手動ブレーキを用いた低自由度歩行行動実験 208
6.4 環境モデル空間を変化させながら行動生成器探索するロバスト歩行生成実験 . 210
6.4.1 未知外力の加わる環境でのロバスト歩行探索法 210
6.5 おわりに 211

第 7 章 結論 212
7.1 成果と結論 212
7.2 まとめと概要 212

発表文献と研究活動 215

参考文献 219

第 1 章

序論

1.1 背景と目的

本論文ではロボット開発を探索により一部自動化し効率化することを目的とする。開発者が仕様を設計に落とし込む作業や運動モデルのパラメタ生成、行動実現時の身体モデルのパラメタや運動モデルのパラメタ修正といった各工程を探索問題としてとらえ、シミュレーションや非線形最適化手法といった各種探索手法の提供とその組み合わせによってロボット開発を支援するシステムを構成することを目的とし、その評価として具体的にいくつかの事例について実際に脚型ロボット開発を支援することで実証的に支援システムの評価を行い、従来の人間の試行錯誤による作業では困難だったロボットの設計と行動実現を達成する。まず序論では、従来解かれてきた探索問題を分類しまとめるとともに、それらの適用範囲と解き方について確認し、本論文における各章の立ち位置を探索手法と探索範囲という観点からまとめる。

等身大ロボット開発には多くの時間と費用がかかることが知られる [1]。一方でロボット開発を効率的に行う手法の確立は実現されていない。ロボットの身体設計を評価するためには目的となる行動を如何に効率よく行うことができるかを評価する必要があるが，一般的な行動探索を確実かつ高速に行う手法の確立が行われていないからである。過去の研究では，安定化制御や二足歩行，上半身を用いたマニピュレーションタスクに至るまで多様な問題が個別の手法で解かれているが，それぞれの手法は一長一短があり全ての問題に適した手法は存在しない．すなわち，広く一般的な問題を解くためには，これら多様な手法をロボットの幾何モデル・動力学モデルとあわせて利用可能な統合開発環境が必要となるのである．このような統合開発環境の必要性については，近年 Darpa Robotics Challenge (DRC) [2] を始めとするロボット実用化に向けての機運のなかで，多くのロボット研究者が共通に抱き始めているものであると感じる．2014 年に Moulard らは RobOptim[3] と名付けられたフレームワークを開発している．RobOptim はロボットの幾何モデルと動力学計算，各種の最適化手法ライブラリを統合したものであり，Moulard らが研究に用いてきた最適化ライブラリ群をまとめたものであった．さらに 2016 年，Tedrake らは Drake[4] と名付けられた解析ツールキットを開発している．Drake は RobOptim と同様にロボット幾何モデルと動力学計算，各種の最適化手法ライブラリを統合し，加えて，動力学シミュレーション，逆運動学計算実装等の解析ツールを有する．

以上のように多様な問題に対処可能な統合開発環境の研究が始まりつつあるが，真にロボットの行動性能を評価するためには，単純な動作の探索だけではなく，目的の状態を達成するためのリンクの接触変化を伴う行動の探索を行う必要がある．また，ロボットの動作生成，行動生成に求められる手法は Drake, RobOptim が有する非線形最適化ライブラリにとどまらない．例えば，衝突回避軌道の探索にはランダムサンプリング手法 [5][6][7] が有効であるし，進化計算手法 [8][9][10] も計算時間はかかるものの局所解に陥らず実行可能解を求めるという問題については有効である．あるいは，近年成長著しい機械学習手法 [11][12][13] と，それを用いた動作生成の高速化も今後求められる技術と考えられる．

ここまでは行動評価手法の適用範囲に関する問題（一般性）であるが、評価する行動が複雑化するにつれ高速性も課題となる。例えば行動の評価に応じてロボットの身体パラメタを探索し、ロボット身体そのものを探索するためには評価を高速に繰り返す必要がある。さらに動力学シミュレーション世界で探索された行動を実世界で実現する（再現性）ためには、探索時に用いたモデルと実ロボットモデル間に存在するモデル化誤差の同定も必要である。以下に，考えられる問題の性質と問題に有効と考えられる手法をまとめる．

最適性： バランスや出力上限を考慮しロボットの運動性能を限界まで利用する動作生成問題については、勾配を用いた非線形最適化手法 [14] が有効であることが知られている [15][16][17]。

可解性： 環境との干渉やバランスを考慮し実行可能な動作を探索するという問題については、同じく最適化手法を用いるものもある [18][19] が、一般の環境が非凸であるために局所解から脱出可能なランダムサンプリング手法 [6][7]，や進化計算手法 [8][9][10] が有効である．

高速性： 高速に解を求めることが要求される問題では、学習 [20] やデータベース化 [21][22] による高速化手法。モデルリダクション [23] が用いられる．また問題の性質によっては等式不等式制約を線形化することで，高速計算可能な二次計画問題 [24][25][26] や一次計画問題として解く手法も有効である．

汎用性： 以上の手法を組み合わせて，より広範囲な問題を統一的に解くための試みがある．

- グラフ探索やサンプリング手法を用いて接触状態を選択し，それを満たすような動作を探索することで全体の行動を生成する手法として ‘contact-before-motion’ [27][28][29][30][31][32][33][34] と呼ばれるアプローチが用いられる．さらに、‘contact-before-motion’ を拡張し，滑りや転がりも利用した行動生成を可能にする手法として ‘planner-before-motions’ [35][36] がある．
- アニメーションの研究では、複雑な接触力計算・運動学計算の式を立てることなしに、汎用の動力学シミュレーター [37][38][39] 上で繰り返し運動を計算し探索する手法 [40][41][42] が用いられ、問題の評価関数を与えるだけで、究めて汎用的に動作生成が可能であることが示されている。一方で探索にかかる計算時間は動力学シミュレータの速度に比例し一般に低速である．

本研究では情報システム工学研究室で開発されている Euslisp [43][44] をこのような各種機

能を備える統合開発環境の一つであると考え用いる。Euslisp では、各ロボットモデルの表示から運動学計算、動力学計算を扱うことができる。また非線形最適化手法 [14] や二次計画問題 [24][25]、ランダムサンプリング手法 [5]、進化計算ライブラリ [8][9][10]、深層学習ライブラリ [45] を呼び出すことができる。また本研究では動力学シミュレータも実装を行い Euslisp のロボットモデルを用いて高速に計算できるものを開発した。それらを用いて可能となる一般性の高い行動生成手法 [35][36] やシミュレーション上での繰り返し計算と探索を用いた動作生成手法についても本論文で述べる。

さて、このような統合環境ができることで、それを用いてヒューマノイドロボットの性能を効率よく使いきりロボットの身体パラメタまで含む探索を行う研究にようやく着手できる。本研究の目的はロボットの運動モデルのパラメタ探索のみならず身体モデル、環境モデルのパラメタ探索も行うことで、ロボット完成前の段階における設計支援からロボット完成後の行動実現支援まで扱うロボット創成支援システムの構成法について論じるとともに、実際に脚型ロボットを支援システムを用いて開発し行動実現していくなかで支援システムの評価を行うことである。

1.2 ロボット開発支援に有効な探索問題についての考察

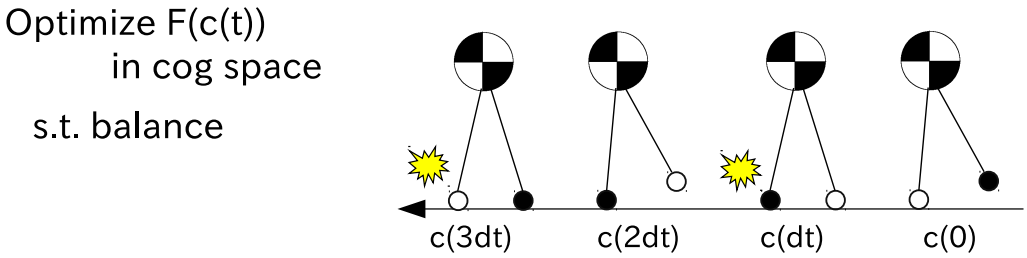


図 1.1. Heuristical model reduction approach (e.g., Center of Gravity model for dynamic walking) has been sometimes used.

ヒューマノイドロボットの行動生成問題は、運動方程式に代表される等式条件と関節負荷トルク上限といった不等式条件を満たす探索問題として、あるいは消費エネルギーといった評価関数も考慮し最適化問題として定式化できることが知られている。例えば二足動歩行の研究 [46][47] ではロボットを重心質点モデルとして近似し、与えられた足の置き方を考慮しつつ転倒が起こらないよう重心軌道を探索することで実時間での歩行行動探索が実現されているととらえることができる（図 1.1）。しかしながら、そのような過去の研究が最適化によりロボットの持つ性能を使いきっているかと問われると未だ課題は多いと考える。先の歩行の例で課題を並べてみると、一つ: ロボットの全自由度ではなく重心の自由度のみを探索しているという問題を簡単に思いつくことができる。二つ: 足の置き方、あるいは接触遷移という条件が与えられていることにも課題がある。足の運び方を計画する問題はフットステッププランニングとして知られているが、その多くは計画された足運びをどうやって実現するのかを考慮しない。

滑らせるのか転がせるのか持ち上げて下ろすのかといった情報も必要である。より直接的な条件としては運動方程式以外の制約を課さないことであり、動力学シミュレーションの出力する全運動のなかで行動を探索することで一般の接触遷移を伴う行動を探索することができる。三つ: 実ロボットで行動実現する際に発見的に決定されるパラメタがしばしば存在する。これは質量といったロボットの身体パラメタがシミュレーション世界と実世界とで誤差を含んでいることに由来するものである。実ロボットにとって最適化された行動は実ロボットのパラメタを用いて最適化することで初めて得られ、そのためには実ロボットのパラメタをシミュレーション世界に取り込みつつ行動を最適化する必要がある。最後に四つ: 実ロボットの持つ身体パラメタも行動に対して最適化することができる。特定の行動のみを行うロボットであれば全身体パラメタを最適化でき、特定の行動に特化した機構を持つロボットであればその機構のパラメタが探索空間に含まれることでよりよい行動が得られる。本論文で目指す身体行動創成支援システムとは、一つ目二つ目の全身自由度・接触遷移を考慮した行動生成に加えて、三つ目四つ目の身体環境モデルのパラメタ推定と探索まで扱うものをさす。本論文で解くべき問題を具体的に見るため以上四つの問題について詳しくまとめる。

1.2.1　全自由度空間を探索範囲とする探索

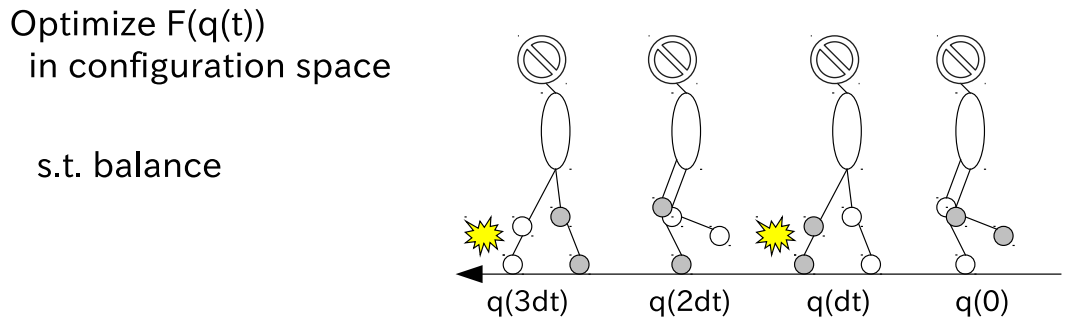


図 1.2. Better walking motion than CoG model reduction will be achieved by searching in whole-body configuration space.

図 1.2 に示すように全可動域空間で同じ評価関数 F を最適化することにより運動軌道求めることで、一部自由度空間のみでの探索よりさらに評価の高い行動生成が実現できる。ヒューマノイドはルートリンクが地面に固定されておらず、浮遊リンクとなるため、ロボットの剛体としての振る舞いを表現するパラメタ数は全アクチュエータの自由度にルートリンクの位置姿勢に関する六自由度を加えたものとなる。また、浮遊リンクの六自由度はロボットのアクチュエータへの入力のように自由に制御するための入力を持たない。このようなシステムを劣駆動系と呼ぶ。劣駆動系であるヒューマノイドの全自由度を利用した探索には二通りの方法が考えられる。一つは全アクチュエータの入力空間と浮遊リンクの六自由度を合わせた空間で軌道を探索し、その際に動作が動力学的に正しくなるという条件を満たすように探索を行う方法である。すなわち全アクチュエータの位置・速度・加速度と浮遊リンクの位置・速度・加速度は多

リンク系の運動方程式を満たさなければならない。運動方程式にはロボットの環境から受ける外力の項も含まれるため、探索空間に接触反力・反モーメントも含め摩擦等の条件を考慮して探索するか、反力の探索を部分問題として別に解きながら全体の運動を探索する方法が考えられる。この方法は、接触状態を固定にしたりあらかじめ与えたりすることで高速計算可能な非線形最適化手法を用いて解くことが可能であり [16][34][17] 全身軌道の最適化でも数十分から数時間程度の時間で求めることが可能になっている。もう一つの方法は汎用の動力学シミュレータを用いて、アクチュエータの入力空間のみで探索を行う方法であり、浮遊リンクの運動や接触反力の計算は動力学シミュレータが行うことにより常に動力学的に正しい運動のみを探索することができる [41][40]。動力学シミュレータのを用いる場合のメリットは、複雑な式を立てる必要なく評価関数と条件だけを定義すれば動作を表現できることにある。一方で、その汎用性ゆえに適切な条件が設定されなければ不適切な解がでてしまうというデメリットも考えられる。また探索は動力学シミュレーションの速度に比例するため高速なシミュレータが必要である。

1.2.2 探索範囲を狭めない条件つき探索

図 1.3. More general motions than walking includes whole-body motion and contact transition in addition to leg contacts.

これまでは歩行を例に挙げて考えてきたが、ヒューマノイドに求められる行動は歩行のみに留まらない。移動行動だけをとってみても、歩行、階段昇降、脚立上り、崖上りなど、より一般性の高い行動生成が必要である（図 1.3）。脚のみに限らずリンクの接触変化を伴う運動を探索する方法については、問題を接触状態の探索（歩行におけるフットステッププランニング）と運動の探索（歩行における歩行軌道生成）に分割する方法が有効であることが知られている（‘contact-before-motion’: [27][28]）また筆者の過去の研究 [35][36] では、加えて接触状態の遷移方法の探索問題も解くことで滑りや転がり遷移を利用したより一般性の高い行動が探索できることが示されている。このような一般の行動ができるロボットを開発する場合には、設計段階でアクチュエータの出力上限といった行動実現に必要とされるスペックを予測することが困難であるために、歩行でも必要だった転倒しないという条件を満たすことに加え自己衝突しない条件やアクチュエータの出力上限、力センサの破壊応力に基づく接触反力・反モーメント

の上限も考慮しなければ実現可能な行動は生成できない。そのためにはすべてを式で書き下し探索を行う前述の方法に加えて、前章で述べた動力学シミュレーションを用いる手法も、問題の複雑さゆえにその汎用性が有効であるといえる。例えば野球のスイング動作を最適化する場合には、まず重心を後ろに移動させ前足を軽く浮かせてから重心を前に移動させつつバットをふり、振り抜いたのちに後ろ脚が弧をえがいて滑るといった複雑な接触変化を伴う。これを式により表現することは手間であるが、シミュレーションを用いた方法ではバットの振り抜き速度を評価関数として転ばない条件で動作を探索すれば評価値の高い行動が求められると期待できる。

1.2.3　実ロボットでの探索された行動の忠実な再現

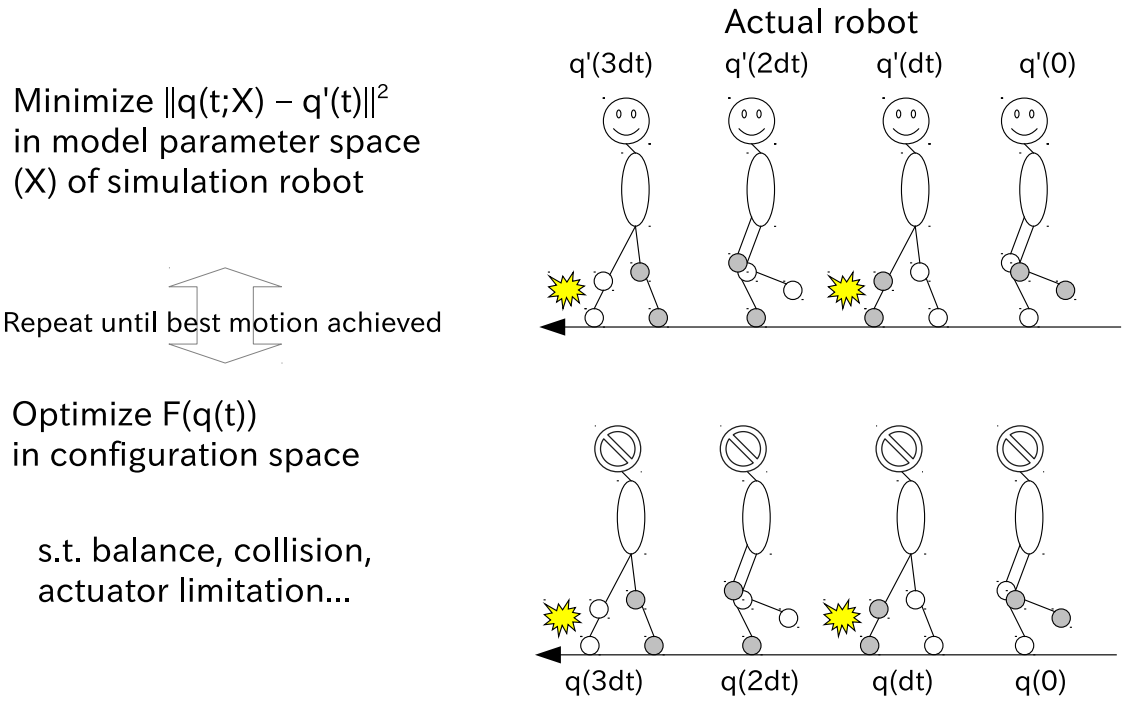


図 1.4. Optimized motion is not optimized if the motion could not be achieved by actual robot faithfully as simulation.

ここまでは探索空間と探索条件に着目し一般化した解き方であり、与えられたモデル上の探索としてはこれ以上の拡張は不可能である。しかしモデルが完全に与えられない場合、変更が可能な場合にはさらなる拡張が考えられる。一つは探索された行動をシミュレーション通り忠実に実現する問題である（図 1.4）。実ロボットでの行動実現が探索に用いたシミュレーションでの挙動と大きくずれていたのでは最適化された行動が実現できたとは言えない。図 1.4 に示すように、実ロボットでの行動とシミュレーションでの行動の差を小さくするようにロボットのモデルを探索し、そのモデルを用いて再度シミュレーション空間での最適化を行い実ロボットに適用することを繰り返す必要がある。一度の実験で完全なモデルが推定できる問題なら

ば、この繰り返しは一度で十分である。しかし前述の通り劣駆動系であるロボットでは、全自由度の入力を制御できず全状態を測定することは困難であり、ロボットと独立した外部に固定設置された測距センサ等を用いなければモデルを一意に推定できる情報は得られない。本研究ではロボットに搭載可能なセンサのみを用いて不完全な推定を複数回繰り返すことでモデルを推定するアプローチをとる。

1.2.4 リンク質量等の身体環境モデルのパラメタ探索

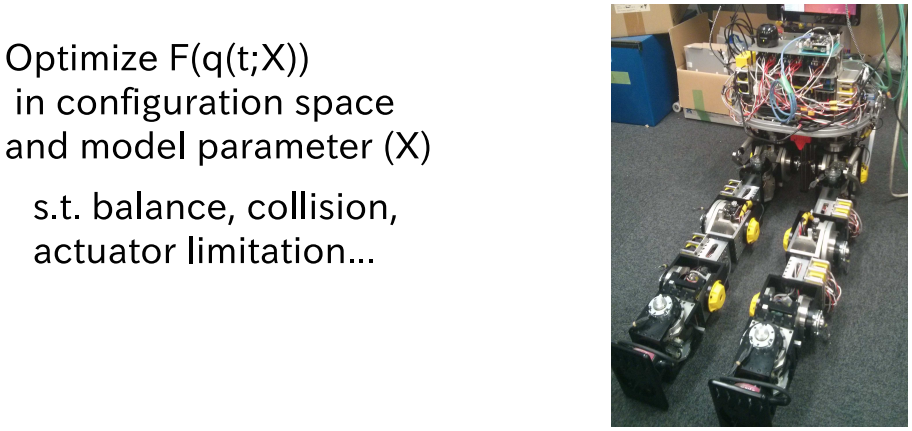


図 1.5. Robot model could be optimized in addition to whole-body configuration. Right robot is a developed robot in this paper by using body and behavior creation support system.

最後に身体と環境のモデルが変更可能であるような問題への拡張として、実ロボットの持つ身体パラメタを行動に対して最適化する課題を取り上げる。特定の行動のみを行うロボットであれば全身体パラメタを探索でき、特定の行動に特化した機構を持つロボットであればその機構のパラメタが探索空間に含まれることでより評価値の高い行動が得られる。図 1.5 に示すような、ロボットの全自由度・接触探索に加えてロボットの身体モデルのパラメタが同時に探索できれば、身体と行動が複雑に相互作用するような探索問題も扱うことが可能である。本論文では、バネによる体幹の支持を行う準受動歩行を取り上げ、歩行とバネの相互作用を考慮して探索することで歩行に適したバネ係数と準受動歩行行動の制御と動作生成を行う。

1.2.5 ロボットの運動から身体まで探索するシステム

以上、探索問題を四つに分けて考察した。分割は、まず探索空間をロボットのアクチュエータ自由度空間と身体環境モデル空間とに分け、後者を探索しない問題を探索空間と探索条件の二点で分け、さらに後者の身体環境モデル空間探索を考慮する問題を推定と最適化という目的関数の性質によって二点に分けた。前者は動作モデルのパラメタ探索であり、後者は身体環境モデルのパラメタ推定と探索である。次の 2 章では、もう一度具体例を交えて探索問題の分類

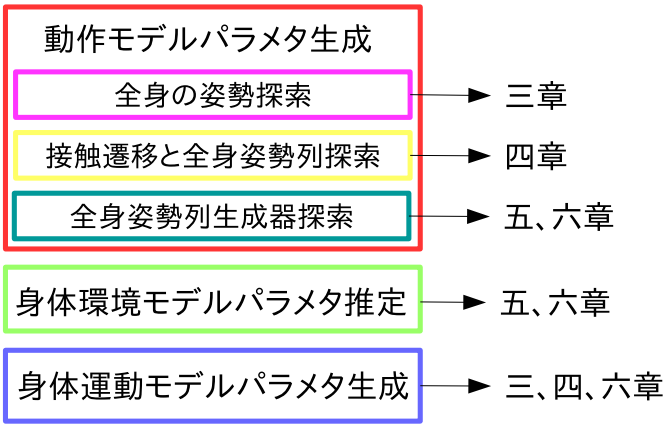


図 1.6. Overview of all chapters with respect to search problem.

を行うとともにロボット開発のなかで以上の問題がどのように用いられるかを検討し身体行動創成支援のためのシステムが解く問題と各章のつながりについてまとめる。さらに 3 章以降では支援システムを用いて具体的にロボット開発支援を行ってゆき、実際に等身大脚型ロボットの開発・行動実現することで評価を行う。

1.3　本論文の構成

本論文は全 7 章から構成される．以下に各章の概要を述べる．

1 章：ロボット開発における設計段階、運動生成、行動実現が失敗したときの運動の再生成といった各問題を探索問題としてとらえ、探索問題を探索空間や制約、身体モデルのパラメタ探索の考慮といった視点から分類していき、それぞれに有効なシミュレータや探索アルゴリズムといったツールが存在することをまとめ、各章がそういった問題の性質から見たときにどのように異なる目的を持つものであるかをまとめた。

2 章：具体例から考察しロボット開発における支援として、仕様を設計に落とし込む工程を自動化するための身体運動モデルのパラメタ探索、ロボット完成後の行動実現時に必要となる運動再生成のための実体の身体環境モデルのパラメタ推定という二種類の支援を行うことでロボット開発を支援する身体行動創成支援システムについてまとめる。また支援システムにおける探索手法と問題の性質による違いについてもまとめ各章で解く問題について支援システム内での立ち位置を明確にする。

3 章：リンク長の設計を支援する問題を姿勢探索手法を用いて解く。全身自由度を用いた姿勢探索手法の課題として計算速度の問題を取り上げ、局所解にとらわれない干渉回避姿勢探索手法の高速化，関節負荷トルク上限といった力学的制約も考慮した低負荷姿勢探索手法の高速化について述べる．

4 章：接触点の設計を支援する問題を接触遷移を伴う姿勢探索手法を用いて解き、摩擦推定と着座行動再生成のオンラインでの繰り返しにより行動実現を支援する。接触遷移を伴う

行動生成手法として最適化手法とグラフ探索を組み合わせたもの、動力学シミュレーションと進化計算を組み合わせたものそれぞれについて実装と実験を行う。前者については接触遷移方法として単一のものに限らず，滑りや転がりも拡張考慮した行動生成手法を扱い，実ロボットを用いた滑り着座行動実験を示す．後者についてはさらに制御パラメタの探索も含めた動歩行生成実験を行う。

5 章： 行動実現のための推定が必要となる身体環境モデルのパラメタのうち一部のみがセンサ情報により計測可能であるような問題として歩行時の全身動力学モデル・接触モデルのパラメタ推定を扱い、実ロボットを用いた性能評価実験を行う。また、本研究で開発した脚型ロボットの身体構成、センサと電装系構成、制御ソフトウェア構成についてもここでまとめる。

6 章： 受動機械要素の設計を支援し行動実現ではそれを推定する問題として、受動軸に備えられたバネ係数を省エネルギー歩行が可能になるよう探索により決定し、行動実現においてはバネ係数と動力学モデル・接触モデルパラメタ探索を行うことで準受動歩行行動実現まで行う。また、本研究で開発した脚型ロボットのバネとクラッチを用いた歩行補助機構についてもここでまとめる。

7 章： 本研究の成果をまとめ、課題について言及する。

第 2 章

ロボットの身体行動創成支援システム構成法

2.1 はじめに

序論では、ロボット開発には手作業の試行錯誤では解決することが難しい問題が存在し、それらを探索問題と捉え自動化するロボット開発支援システムが必要であることをまとめた。またそのような探索問題について従来の研究は運動モデルのパラメタ探索に限るものがほとんどであり、身体や環境が変化することを考慮しないものであった。本章ではロボット開発の流れ全体を本論文で開発した等身大準受動脚型ロボットを具体例として見ていくことで身体行動創成支援システムが解く問題の範囲と限界についてまず確認する。次に支援システムが探索によりロボット開発を支援する際に解く問題について運動モデル空間だけでなく身体環境モデル空間まで探索するために必要な探索手法と使い分けについてまとめ、三章以降で解く問題の範囲と意義を確認する。

2.2 探索空間から見たロボット開発の流れと既存研究

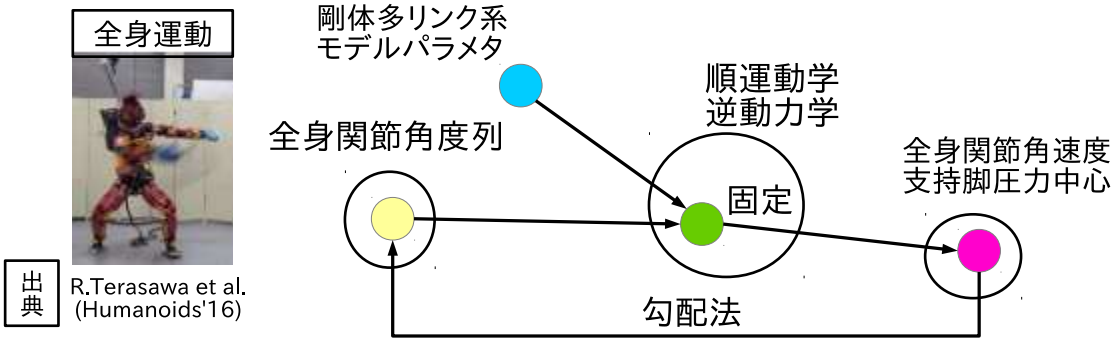


図 2.1. Search in whole-body configuration [17].

これまでのロボット開発は探索空間の拡充により発展してきたという見方ができる．ロボットの体が与えられるものとして，そこで様々なタスクが可能となるようにソフトウェアを開発していくなかで，それまでにできていなかったような行動が実現できたということは，それまで解かれていなかった探索自由度を用いて探索しているためという理由が一つありえるということである．図 2.1 は等身大人型ロボット JAXON の全身を用いたラケットのスイング動作を表している．エンドエフェクタの位置姿勢のみに着目した運動と比べて全身の運動を探索することでより高速なスイング動作が達成されている [17].

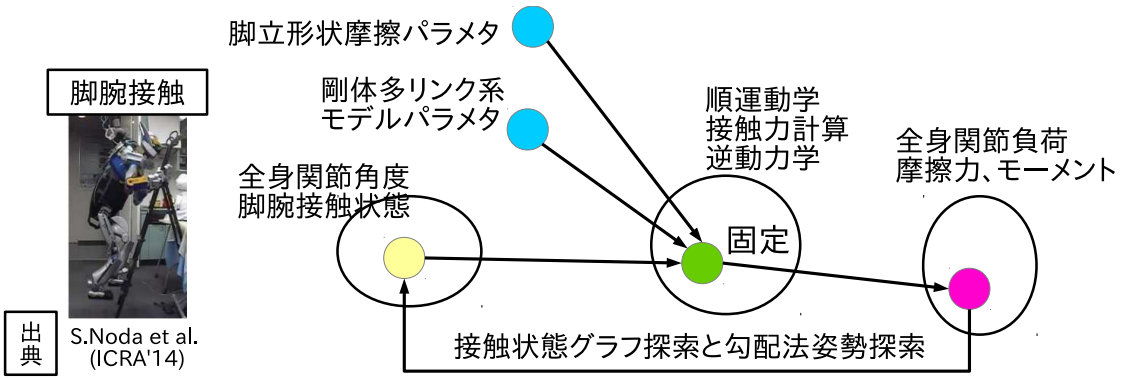


図 2.2. Search in whole-body configuration and contact states [48].

全身自由度を使った動作からさらに探索空間を拡充する例としては接触状態があげられる．接触の切り替わらない動作と異なり探索空間には接触状態を含める手法が有効である．それにより接触を切り替えない探索では生成できなかった脚立登り [48] のような動作も生成できるようになる（図 2.2）.

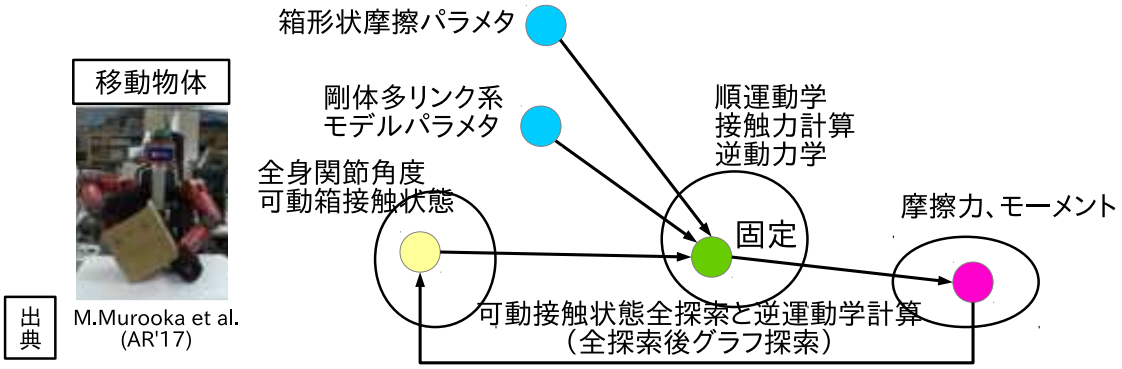


図 2.3. Search in whole-body configuration and contact states which is placed in a movable object [49].

あるいは接触する対象が動くとしたら，接触状態に加えて動く接触対象の状態も探索空間に入る．これはマニピュレーション研究 [49] が当てはまるものである（図 2.3）．ここまで見てきた探索空間の拡充では図中の一番左側にある黄色丸が入った空間，運動モデル空間が広がっ

ていることが分かる．それにより運動を評価するための運動学や動力学計算といった物理演算部（緑丸が入った空間）が変化し探索のためのアルゴリズムが変わっていく．

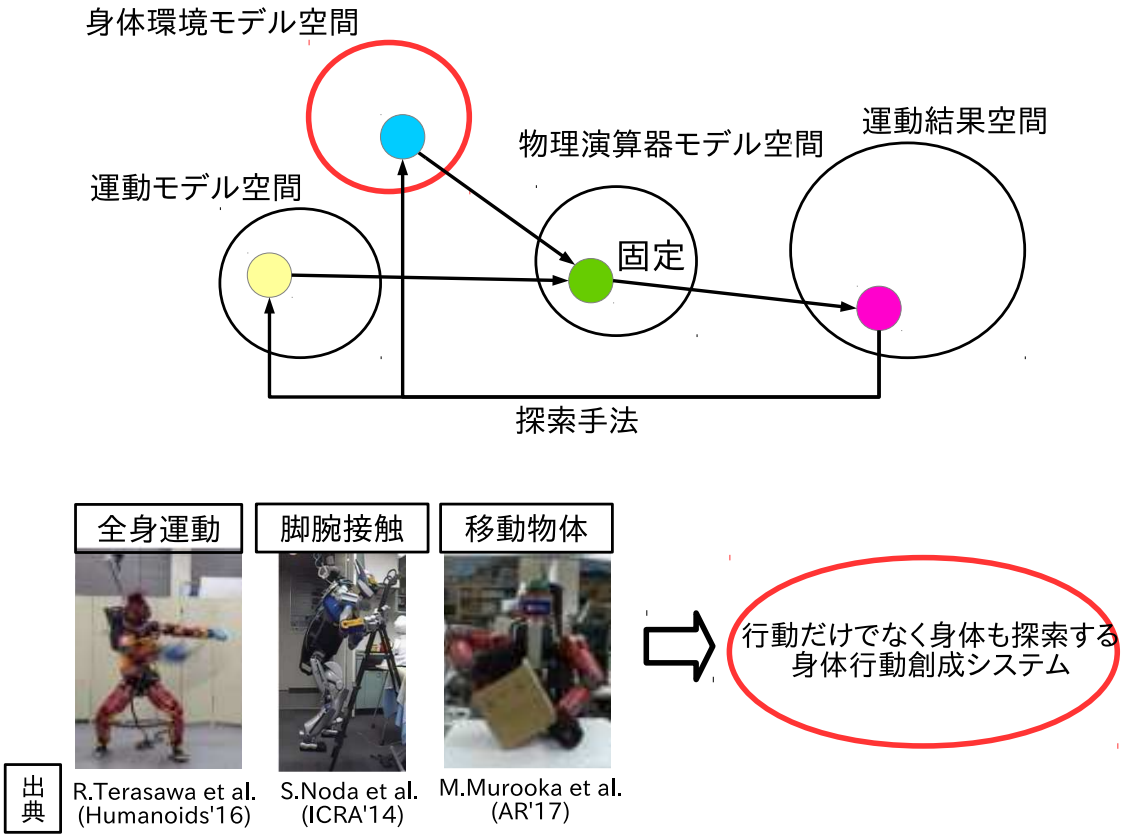


図 2.4. Search in not only motion model space but also body model space [17], [48], [49].

このような従来型のロボット開発の流れ，ロボットの体があってそれを用いて様々なタスクを実現していくような開発工程とは異なる流れとして，従来探索空間として持ちいられてきた運動モデル空間にくわえて身体環境モデル空間での探索も扱う問題を本研究では解いていく（図 2.4）．身体環境モデル空間とはロボットの身体や環境のモデルの具体的なパラメタが含まれる空間のことである．例えばロボットの身体モデルのパラメタとしてはリンクの質量や重心，慣性テンソルといったもの，環境モデルのパラメタとしては地面の傾きや固さといったものが含まれる．身体環境モデル空間を探索するということはロボットの身体が変化していくような問題や，環境が変わっていくような問題を扱うということである．このような探索空間の追加により表現できる運動結果空間は拡大し，扱える問題の種類が増える．従来すでに二種類の成果がでている．

一つは身体環境モデルのパラメタ推定である．もっとも簡単な例としては現実の運動結果から身体環境モデルパラメタが一意に計算できるような例が考えられる．しかし一意に計算できなかったとしても身体環境モデル空間での探索問題として考えることでセンシングできている一部のパラメタから残りのパラメタを推定するような問題を考えることができる．図 2.5 では

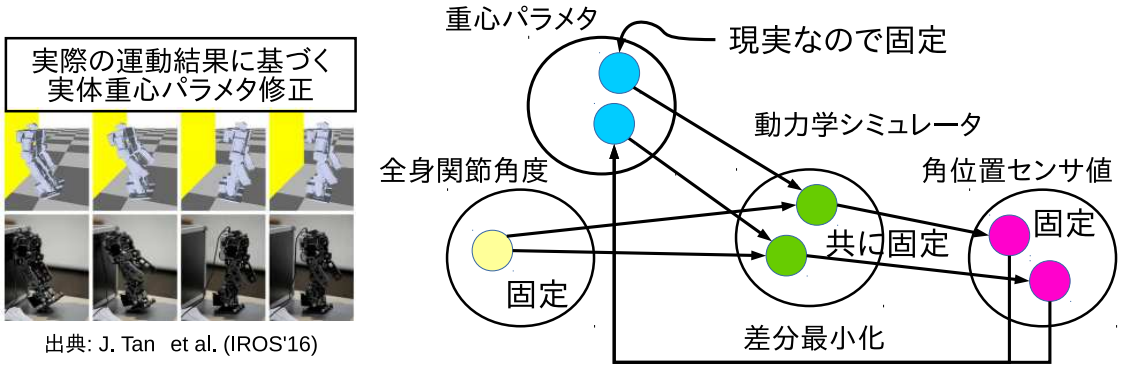


図 2.5. Body model parameter estimation by searching body model space [50].

角位置センサだけが分かるとして，全身の関節角度を動力学シミュレータと現実のロボットに与えることで結果をを比較し，それを最小化するような探索によりリンクの重心パラメタを探索する研究 [50] について示したものである．

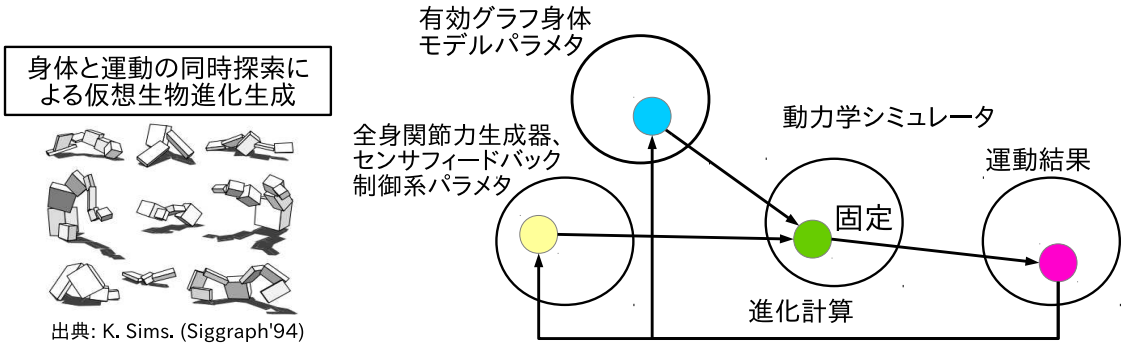


図 2.6. Body model parameter search by searching body model space [51].

もう一つは評価関数に基づく身体環境モデルのパラメタ最適化探索である．図 2.6 では動力学シミュレーション空間において有効グラフで表された身体モデルのパラメタと制御系パラメタを探索することで環境に適した運動を獲得する仮想生命体モデルを進化計算により生成するという興味深い研究 [51] について示したものである．

以上のような身体環境モデル空間での探索も行う身体行動創成支援システムについて最後に図 2.7 で示す．ここまで例で見てきたとおり，支援システムには身体モデルの推定と生成の二種類の応用があった．推定とは実際のロボットの実体があって，その身体モデルパラメタの値が分からないような場合にそれを探索により予測するということであり実体の身体モデルのパラメタ探索である．一方生成とは実体であるロボットが無い状態でコンピュータの上で身体環境モデルのパラメタを変えていき評価の高いものを見つけるという問題でありロボットを設計している段階での設計の身体モデルのパラメタ探索である．実体の身体モデルのパラメタ探索において，運動モデルのパラメタ探索と組み合わせる方法はシンプルな方法としては身体モデ

ルのパラメタを少し変えて運動モデルのパラメタを探索するということを繰り返すといったものが考えられる．このような探索法は運動モデルのパラメタ探索手法が既にあるような場合に効率がよい．また両空間を同時に探索することも探索手法によっては可能である．進化計算が有効であることは先行研究 [51] ですでに示されている．一方実体の身体モデルのパラメタ探索に運動モデルのパラメタ探索を組み合わせる方法は交互に行う前者だけである．これは実体の身体モデルのパラメタには真値が存在しそれが動かせないことに由来する．真値とかけ離れた身体環境モデルパラメタを用いて運動モデルのパラメタを生成することは無意味であり，まず身体環境モデルのパラメタを推定し，それを用いて運動モデルのパラメタを修正するという順番が必要である．これら実体の身体運動モデルのパラメタ修正，設計の身体運動モデルのパラメタ生成を実現する仕組みを実体設計評価機構と本論文では呼ぶことにし，その構成法について具体的に各章にまとめる．

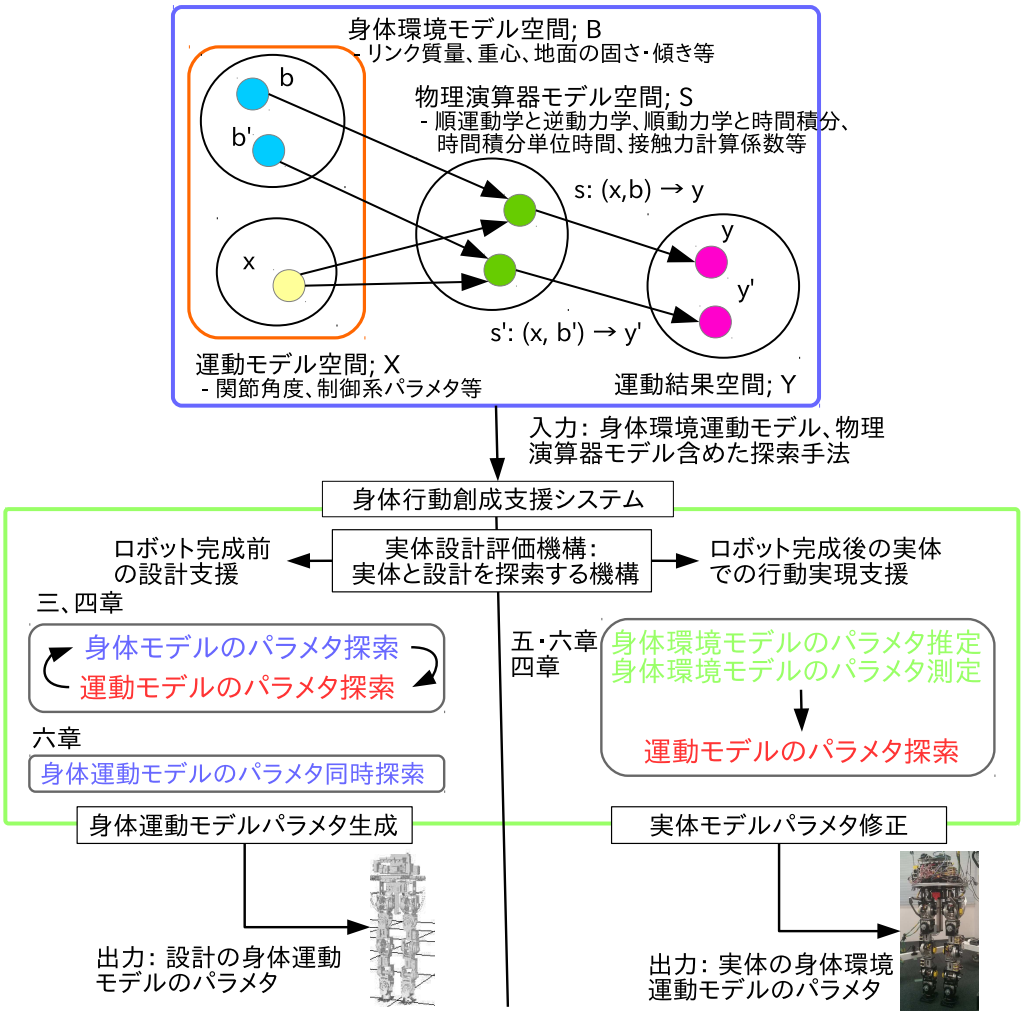


図 2.7. Two scenarios to use body and environment model search; generation of body and motion model parameter for virtual robot, and correction of body, environment, and motion model parameter for actual robot.

2.3 等身大脚型準受動歩行ロボット設計開発を例としたロボット開発の流れ

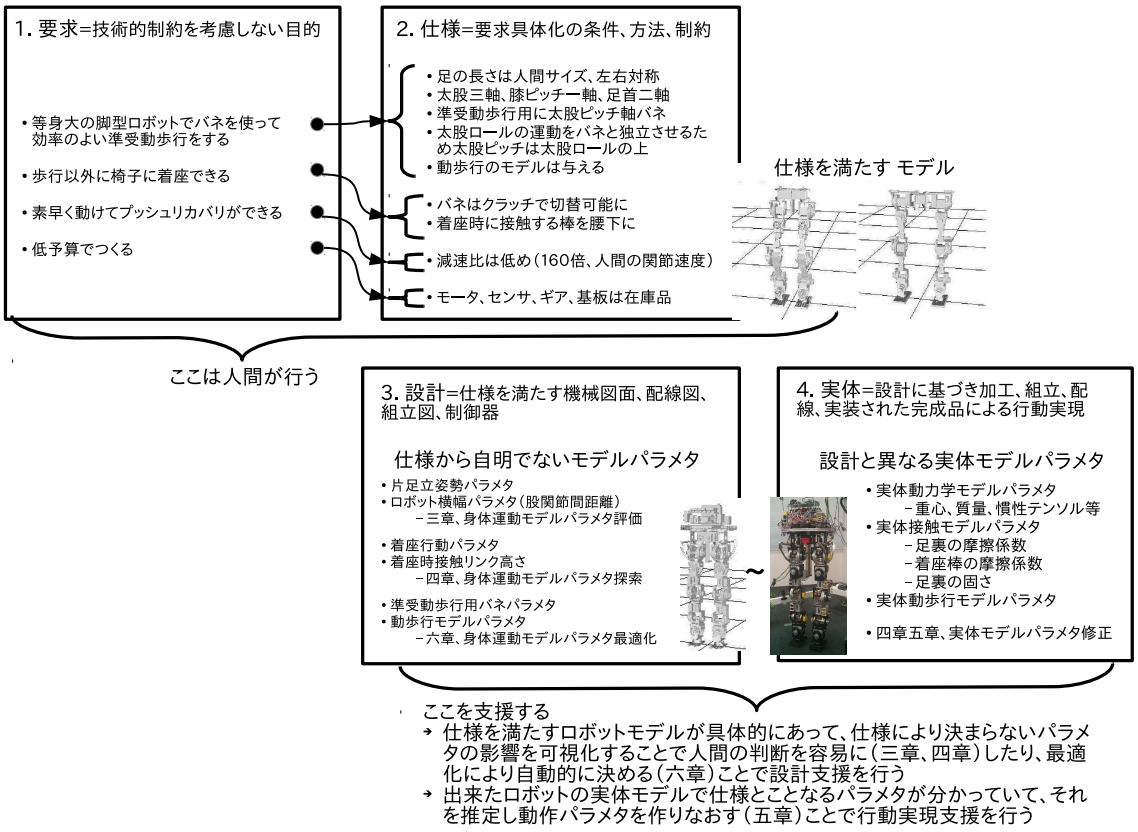


図 2.8. A story of robot development

本章ではロボットの開発の流れについて本論文で設計開発するロボットを実例にまとめるが，本論文の目的はここに示すような要求を満たすロボットを開発することではなく，運動モデルパラメタだけでなく身体環境モデルのパラメタも探索できるような身体行動創成支援システムについて論じることである．さて本研究ではロボット開発を四段階に分けて考える．一段階目としてロボットには技術的な制約のない要求があるところから開発が始まる．二段階目は要求を人間が具体的な条件や制約として書き直した仕様であり，三段階目として実体の作り方動かし方を一意に決める機械図面や制御器まで落とし込むことで，実際にロボットの完成品を用いて行動実現を試みる四段階目にはいっていくことになる．具体例として本研究で開発した準受動脚型ロボットを取り上げる（図 2.8）．要求として以下の四つを例として考える．

- 等身大の脚型ロボットでバネを使って効率のよい準受動歩行をする
- 歩行以外に椅子に着座できる
- 素早く動けてプッシュリカバリできる

- 低予算で作る

これらの要求は実現可能であるかを考慮しないものであるが，要求を具体化するために実現可能で具体的な条件として定義しなおしたものが仕様である．等身大の脚型ロボットであるという要求は，足の長さが人間サイズであり，足が左右対称であるという仕様に変換される．この際には人間の経験に基づく知識も用いられる．例えば脚型ロボットという要求は関節配置は膝にピッチ軸が一つと足首，太股に二自由度以上といった大雑把な仕様で満たされるものであるが，歩行するためには脚の逆運動学が解けた方がよくそのためには六自由度以上あったほうがよい．さらに太股を三軸にすることで歩けることが過去の研究・経験により分かっているといった知識から関節自由度数が決定される．本研究の立場では仕様とは要求から人間が決められるものであるとし，そこに支援は行わない．これはロボット開発の自由度を制限し支援システムにより出来上がるロボットがあるモデルを満たすものに限定させるためである．仕様が完成しそれを設計に起こしていく際，この例ではいくつかの仕様から決定されないパラメタが残ることが分かった．例えば両足の距離パラメタは決まっていない．このような仕様が厳密に決定しないパラメタの決定を探索により支援することが本研究で提案する支援システムの一つ目の目的である．また仕様から決定されないパラメタを人間の経験と支援システムによって決定し設計が完了し実体が完成したとしても目的とする行動実現はできないことも分かった．これは設計通りに実体化することが難しい実体身体環境モデルのパラメタが存在するためである．このような設計と異なる実体パラメタの推定を探索により支援することが二つ目の目的である．以下の章では以上二つの目的について本論文で解く問題を具体例としてまとめる．

2.3.1　仕様から自明でない身体モデルのパラメタ探索

具体例を以下にまとめる．

- 身体モデル
  - リンク長（股関節間距離）
  - 接触点（着座支持棒高さ）
  - 受動機械要素（準受動歩行用バネ係数）
- 運動モデル
  - 着座行動モデルパラメタ
  - 準受動歩行モデルパラメタ

3 章ではロボットの横幅を決める問題を効率のよい歩行をするという要求を目的関数として決定する．そのためにまず足配置が与えられたとして関節負荷がもっとも小さくなるような姿勢の探索問題を考える．関節負荷を考慮しつつ全身の自由度を用いた姿勢を探索する問題であり，探索空間は全身の関節角度ベクトル，探索の制約条件は両足が地面についており重心が一方の接触面に十分近いこと，運動学・動力学を考慮した探索は剛体多リンク系のモデルで行うことができ，探索の目的関数は全身関節負荷トルクの二乗和，探索手法は勾配法が有効である

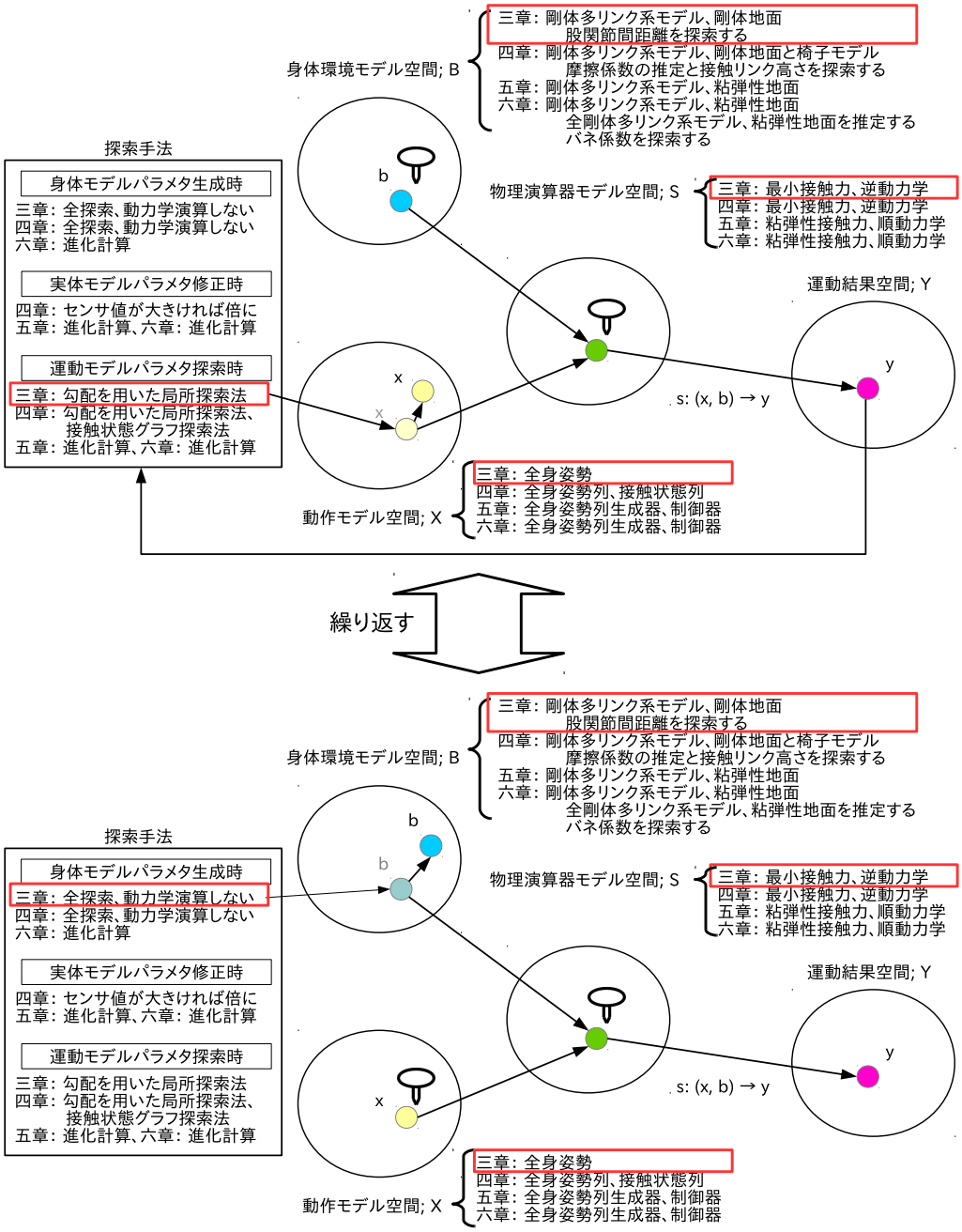


図 2.9. Definition of model and search method to change body and search motion model parameter used in chapter 3.

と知られている．このように仕様により決定されないパラメタの探索には，探索に用いるモデルと探索手法を定義する必要がある（図 2.9）．勾配法による姿勢探索は高速であるため，図 2.9 のように股関節間距離を網羅的に変化させながら姿勢探索を行い全探索することが可能であった．4 章では着座行動時の接触点を着座行動ができることという要求に従い決定するが探索の方法は同様である．

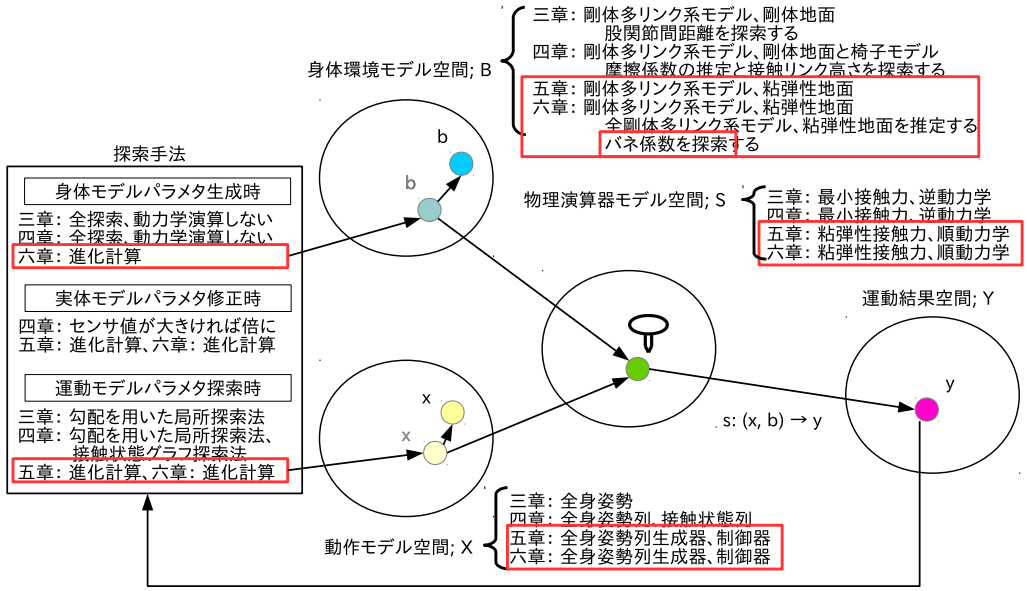


図 2.10. Definition of model and search method to search motion and body model parameter used in chapter 6.

6 章では準受動歩行に必要なバネパラメタを効率のよい準受動歩行を行うという要求を目的関数として決定する．他の例と同様にモデルと探索空間を定義すると図 2.10 のようになる．歩行動作生成器と制御系のモデルが与えられ，剛体多リンク系のモデル，固さと傾きを持つ地面モデル，動力学モデルと進化計算手法を用いることで問題を解いた．図 2.9 との違いは動作モデル空間と身体環境モデル空間が交互に動くか同時に動くかという点である．これは動作モデル空間単体での探索に時間がかかるため網羅的に身体モデルのパラメタを変化させることが現実的に難しいという理由による．動作モデルのパラメタ探索に時間がかかる理由はここで扱う問題が制御系パラメタや地面との不連続接触を扱うために勾配法のような高速な手法を用いることが難しいためである．本論文では探索手法は与えられる入力として扱うが，問題の性質によって有効な探索手法は大きく制限されることが探索手法を他の空間とは別に分けて書いている理由である．

2.3.2　設計と異なる実体身体環境モデルのパラメタ推定と動作再生成

支援システムにより決定した設計と異なる実体の身体環境モデルのパラメタを以下に具体的にまとめる．

- 身体環境モデル
  - 着座接触リンク摩擦係数
  - 準受動歩行用バネ係数
- 運動モデル
  - 着座行動モデルパラメタ

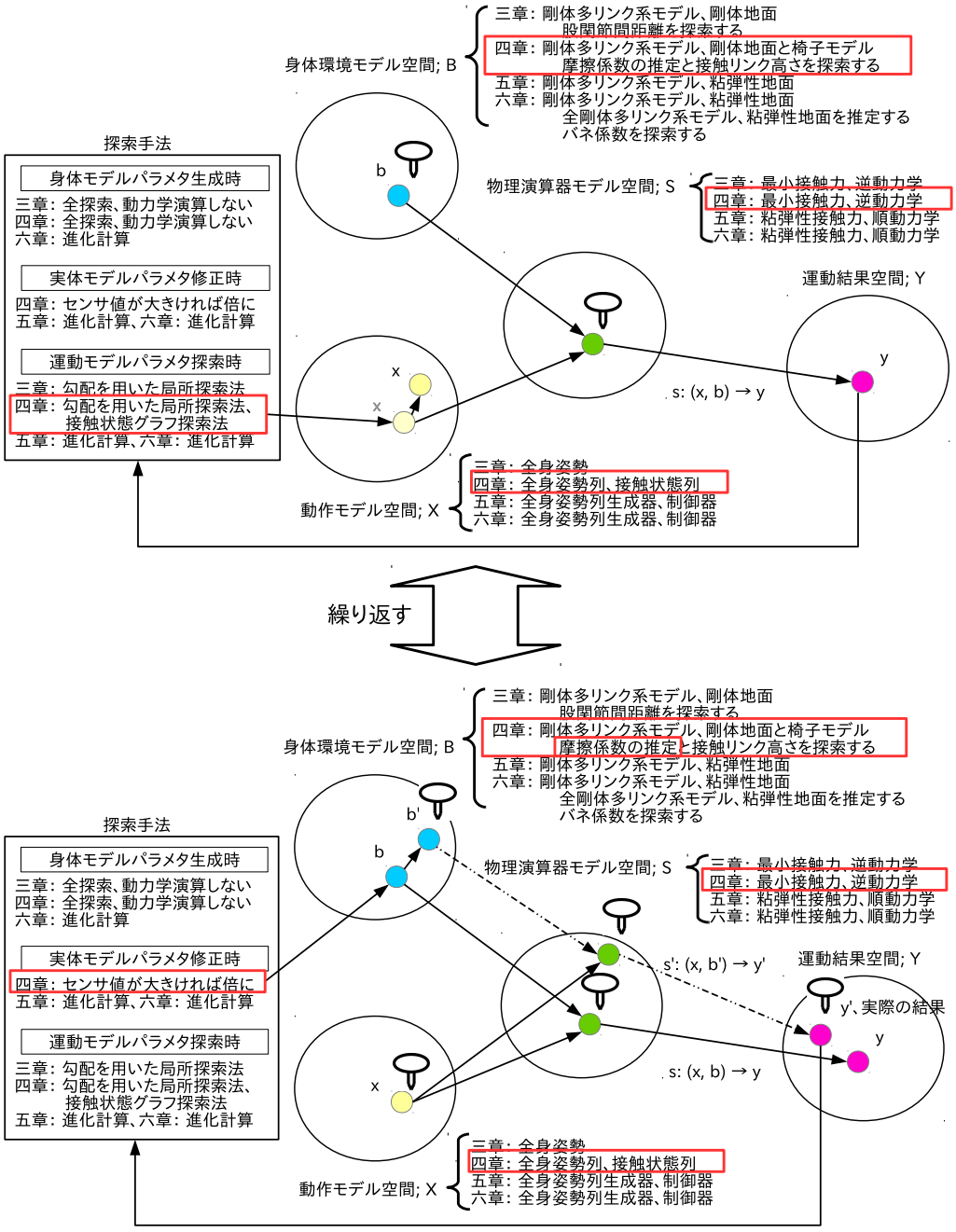


図 2.11. Definition of model and search method to estimate body and regenerate motion model parameter used in chapter 4.

– 準受動歩行モデルパラメタ

4 章では着座接触リンク摩擦係数を推定しながら着座行動モデルパラメタを探索しなおすことで着座行動実現を行う．推定する摩擦係数は力センサを有する本論文のロボットでは直接測定が可能である．そのような場合には単純に動作を実ロボットで実行し測定値を摩擦係数として用いて再度動作生成を行うことで正しい実体運動モデルのパラメタを得ることができる（図

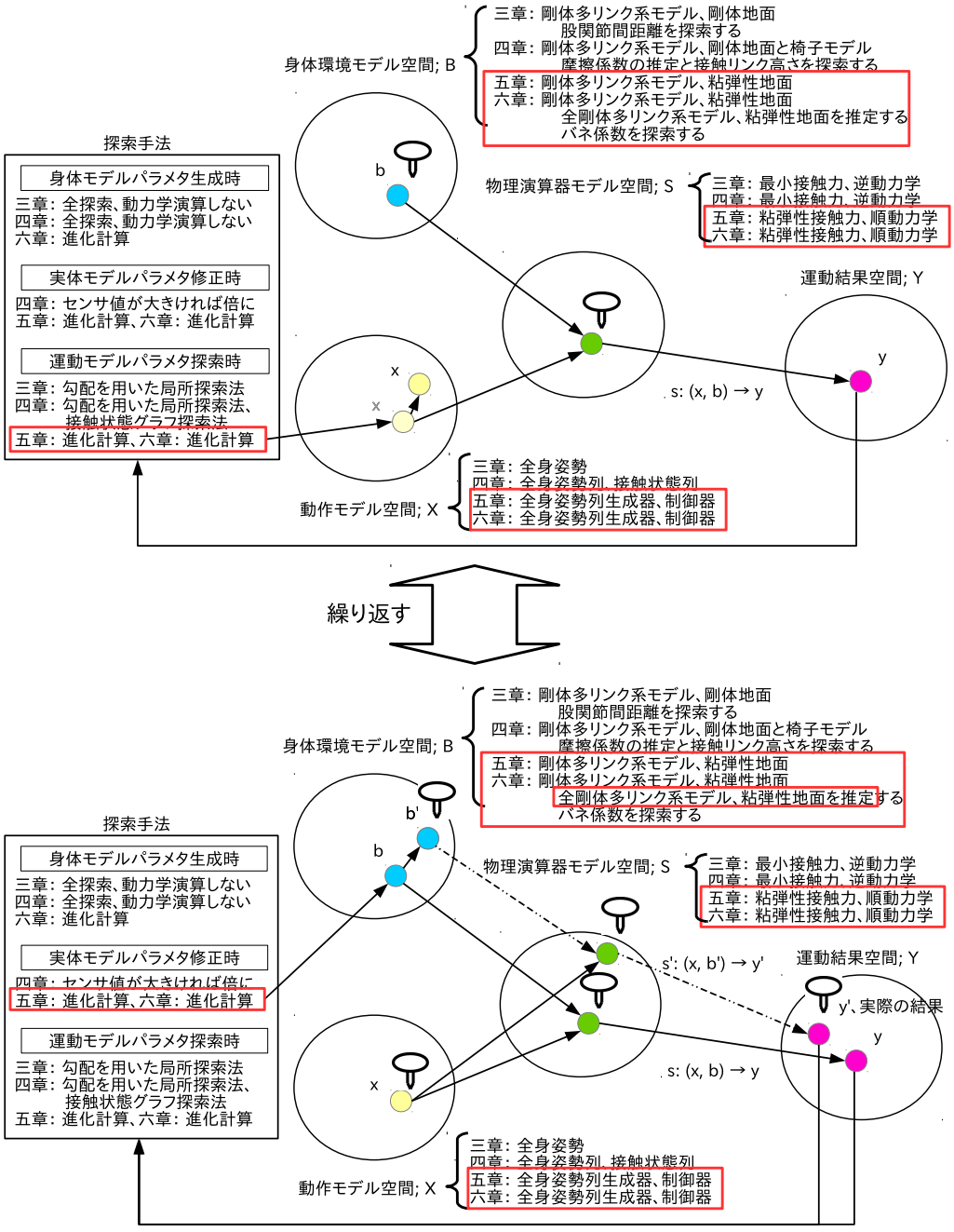


図 2.12. Definition of model and search method to estimate body and regenerate motion model parameter used in chapter 6.

2.11)．図中 ′ で区別される記号は実体モデルのパラメタを表している．s' は現実世界での物理現象と同じ結果を返す物理演算器モデルのパラメタであり実際には存在しない．

6 章ではバネ係数を推定しながら準受動歩行モデルパラメタを再生成する．図 2.11 との違いは設計と異なる実体の身体環境モデルのパラメタがセンサにより直接計測できないことである．図 2.12 のように動力学シミュレーションの結果 y と現実の結果 y' の差を最小化するこ

とで計測できる値から推定する問題を解く．

2.3.3 パラメタの空間を定義するモデルと設計の制約である仕様

本 2.3 章ではパラメタの空間であるモデルと有効な探索手法が与えられたとしてパラメタを探索する問題を扱った．支援システムの入力はモデルであり出力はパラメタである．類似した話題として本 2.3 章冒頭では仕様は要求から人間が決めることのできる設計の制約であるとまとめた．仕様を設計に落とす段階で一意に決定されない部分についてモデルがあればパラメタを探索支援できるシステムが本論文で開発するものであるが，モデルは仕様で完全に決まるとは限らないものである．例えば接触力を計算し運動モデルパラメタを探索するには接触力計算法のモデルを決定する必要があるが，それは要求と仕様から明らかではない．したがって本支援システムは仕様作成を支援しないという限界を持つとともに，設計もモデルを与える部分を人間が行わなければならないという限界がある．もちろんモデルとして要求から考慮すべき現象を網羅した物理演算器モデル，身体環境モデル，運動モデルとそこを探索可能な探索手法が定義できればあらゆる支援が可能となるが，本論文で扱うのは一部であり支援可能な範囲には限界がある．またモデルを限定することで生成されるロボットに制約を加えるという意図もある．省エネルギーで移動するという要求に対し車輪ロボットではなく脚型ロボットのみを探索するという使い方ができることは実用的である．

2.4 身体行動創成支援システムの構造

2.4.1 身体運動モデルのパラメタ生成と実体の身体環境モデルのパラメタ修正からなる実体設計評価機構

本論文で開発する身体行動創成支援システムについて具体例から確認を行ってきた．人間により行われるロボット開発において，ロボット完成前の設計時にモデルが定義されたがモデルパラメタの一部が仕様から定まらない場合，ロボットができた後に実体の身体環境モデルのパラメタが設計通りではなかった場合にそれらの探索と推定を行うことで支援するシステムがあればロボット開発を効率化することが可能である．またそのような計算は身体環境モデル，運動モデル，物理演算器モデルを与え，その中での探索アルゴリズムを決定すれば可能であることを具体例より確認した．この流れは図にすると図 2.13 のようになる．支援システムが解く二つの問題を身体運動モデルパラメタ生成，実体モデルパラメタ修正と呼ぶ．身体運動モデルパラメタ生成はロボット完成前の仕様が決めない身体モデルのパラメタ探索を自動化するものである．この時実体である完成したロボットは存在しないため探索は常に設計の身体環境運動モデルのパラメタについて行われる．また探索の仕方として高速に運動モデルのパラメタ生成が可能な姿勢探索といった問題では身体モデルのパラメタを網羅的に変更しながら運動モデルのパラメタを探索する手法が有効である．逆に低速である場合には二つを同時に探索するといった有効な探索手法の選択が必要となる．実体モデルパラメタ修正ではロボット完成後に設計と実体でモデルパラメタに誤差が含まれている分を修正し，それに基づき運動モデルのパラ

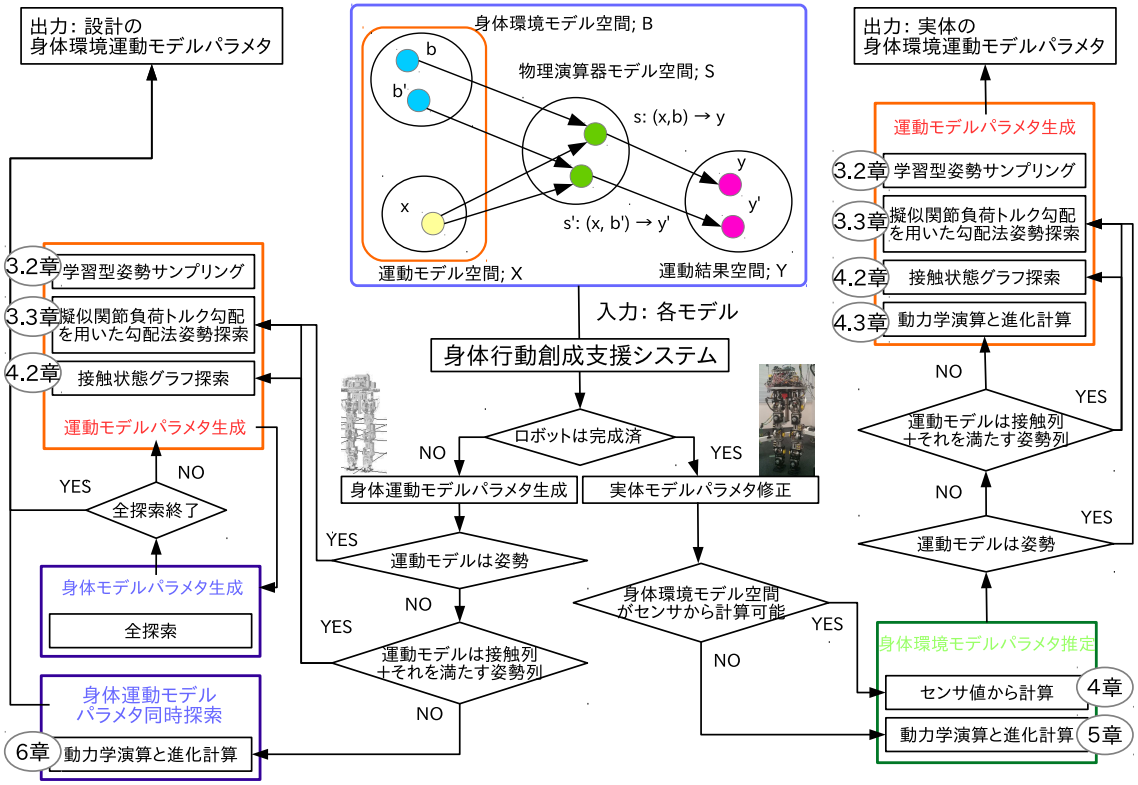


図 2.13. Two scenarios to use body and environment model search; generation of body and motion model parameter for virtual robot, and correction of body, environment, and motion model parameter for actual robot.

メタも再生成することで行動実現を支援する．この時設計モデルパラメタを変更することは実体に影響を与えないため無意味であり探索は常に実体モデルパラメタについて行われる．身体運動モデルのパラメタ生成では二つの空間を交互に解くか同時に解くかの使い分けが必要であったが実体モデルパラメタ修正では同時に解くことは行わない．これは実体の身体環境モデルパラメタに真値が存在するためであり，間違った実体身体環境モデルパラメタを用いて運動モデルパラメタを生成することは無意味だからである．以上の二つの問題はそれぞれ実体の評価と設計の評価を行う仕組みであるため実体設計評価機構と呼ぶ．以上二問題の他に実際に開発したロボットの結果をみて設計モデルパラメタを再度修正しロボットを作り直すという問題も考えられる．本支援システムでは身体運動モデルパラメタ探索で目的となる動作がロボット完成前に評価されており，実体モデルパラメタ修正で行動実現支援することでそのような時間面・金銭面でのコストが高い実体化後再設計が起こらないよう支援する．しかし再設計が必要となる場合は二つの評価機構のどちらで失敗したかによって二種類の原因が考えられる．一つ実体モデルパラメタ修正での失敗．すなわちは機械加工誤差，組み立て誤差によりロボットの開発時に大きな誤差が入ることで実体の身体環境モデルのパラメタ修正後に再生成された運動モデルのパラメタが設計段階と大きく異なってしまうような場合である．このようなハードウェア製作に問題がある場合に支援システムが可能な支援内容としては，誤差が乗ることが分

かっていればそれを身体運動モデルのパラメタ探索で考慮することが可能である．例えば歩行を目的とするロボットでリンク質量が 10% ずれることが分かっていれば，その誤差の範囲で歩行可能となるよう運動モデルのパラメタ探索を行うことができる．本論文で開発したロボットではこのような大きすぎる誤差はなかったため実験は行っていないが，地面の傾きに対してロバストな歩行生成が可能であること等は確認できており問題として扱うことは可能である．もう一つの失敗原因は設計時の身体運動モデルのパラメタ生成に失敗している場合であり動作生成モデルが現実を反映していない場合，すなわち物理演算器の誤差である．本論文では剛体多リンク系の運動モデルが近似的に成り立つ問題を扱っていたため物理演算器は既知として扱ったが，新しい現象を扱うためには物理演算器モデルのパラメタ推定も必要となりうると予想している．

また図 2.13 は問題の性質によって本論文で開発した支援システムの持つ評価機構を使い分けるブロック図になっている．まずロボットが完成後が前かで設計と実体のどちらの身体環境運動モデルパラメタを生成するのかが切り替わる．図中左側のブロック図は設計モデルパラメタを生成するものであり，高速計算が可能な姿勢探索法（3 章）を用いることができるか，接触遷移姿勢列探索（4 章）を用いることのできる場合には上側の繰り返し探索に入る．いずれでも無い場合には動力学演算と進化計算を用いた同時探索法（6 章）にはいる．動力学演算と進化計算を用いた手法は汎用的であるので姿勢探索等にも用いることが可能であるが一般に進化計算は勾配法に比べて低速であり効率が悪い．また接触状態を維持するような運動モデルのパラメタ生成には十分細かい姿勢を表現できる運動モデルを定義する必要がある．一方勾配法を用いる方法ではすべての姿勢に逆運動学問題をとくことで接触位置姿勢を正確に満たすことが可能である．次に図中右側のブロック図を見てみる．こちらは実体モデルパラメタを生成するものであり，初めに身体環境モデルのパラメタがセンサから計測可能であるかを確認し，可能であるならセンサ値からの計算（4 章）を用いる．不能であるならば動力学演算と進化計算を用いて運動結果の差を最小化するような問題を解く（6 章）．前者は高速でありリアルタイムでの実体身体環境運動モデル修正も可能であることが 4 章では示される．その後修正された実体の身体環境モデルのパラメタを用いて運動モデルのパラメタ修正を行う．この時の分岐は設計時の運動モデルのパラメタ生成器の選択ブロック図と近い．姿勢の探索であれば勾配法や学習型姿勢サンプリング手法，接触を満たす姿勢列探索であれば接触遷移グラフ探索（4 章），どちらでもなければ動力学演算と進化計算の組み合わせ（5,6 章）である．

2.4.2 身体行動創成支援システムで扱うモデル空間

身体環境運動モデルのパラメタから実際に起こる運動結果を得るための写像が実体と設計の両モデルパラメタについて定義されるとする．ある運動モデルのパラメタと身体環境モデルのパラメタが決まった時，運動結果を計算し評価することでその身体環境運動モデルパラメタを評価する．また実ロボットで同じ運動モデルのパラメタを用いた動作実験を行い比較することで実体身体環境モデルのパラメタを推定する．身体行動創成支援システムで扱う四つのモデル空間について以下にまとめる．

2.4.2.1　身体環境モデル空間

ロボットの物理演算では関節リンクの質量，重心，慣性行列，リンク間相対位置姿勢，関節軸ベクトル，リンク外形モデル各頂点座標といったロボットのモデルを構成するパラメタが必要となる．さらにモータドライバの用いる関節位置制御のゲインや関節の粘性摩擦係数などセンサモデルも含める．また身体だけでなく環境との相互作用によって決まる接触力計算時に用いる地面の固さや粘性といったもの，摩擦係数，地面の傾斜などもあわせて身体環境モデル空間と呼ぶことにする．

2.4.2.2　運動モデル空間

ロボットの全身アクチュエータの軌道や姿勢，それらをセンサ情報から計算する行動生成器のパラメタが含まれる空間を運動モデル空間と呼ぶ．本論文における運動モデルパラメタ探索とは，所謂モーションプランニングの問題として従来から研究されてきたものを指す．また本研究では運動モデルのパラメタは現実と仮想で同じものを用いることができると仮定する．これは運動モデルパラメタ生成に用いたコンピュータと，ロボットに搭載されたコンピュータがまったく同じ計算を行うことができるためである．

2.4.2.3　物理演算器モデル空間

ある身体環境モデルのパラメタ，運動モデルのパラメタが与えられることで一意に運動結果を計算する物理演算の方法が含まれる空間を物理演算器モデル空間と呼ぶ．この空間には順動力学と逆動力学といった運動方程式の解き方の区別や，順動力学計算後の積分に用いるタイムステップ，接触力モデルのパラメタ等が含まれる．接触力モデルについては [52] に詳しい．本論文で用いる動力学シミュレーションでは接触力モデルとしてペナルティ法を用いており，接触点の地面への侵入距離 x と速度 v，地面の弾性 k と粘性 d を用いて接触力 $f = kx^a + dx^b v^c$ のように計算を行う．本論文で用いる動力学シミュレータでは $a = c = 1, b = 0$ であるが，$a = b = 3/2, c = 1$ とするモデル [53] なども考案されており，この a, b, c ような物理演算に用いるパラメタをこの空間で表現している．実体モデルパラメタ推定では現実と仮想の運動結果を比較することで身体環境モデルのパラメタを推定するが，これは現実の物理法則とまったく同じ計算が可能であるような物理演算器が存在するとしてそれが推定に用いる物理演算器と十分近いことを仮定している．

2.4.2.4　運動結果空間

物理演算の結果であるロボットの振る舞いを運動結果空間と呼ぶことにする．振る舞いとは，ロボットの運動を決定するパラメタであり，回転リンクからなる剛体多リンク系では各関節の角位置・角速度・角加速度，ルートリンクの位置姿勢・速度・加速度，全接触点とそこに加わる力などが考えられる．物理演算ではその全ての値が計算できるが，実世界で計測できるのは通常，一部のみである．運動結果は身体環境，運動，物理演算器の各モデルのパラメタが与えられることで一意に決まるものであるため，モデルという表現を用いず運動結果空間と

呼ぶ.

2.4.3 実体設計評価機構が扱う探索問題の分類

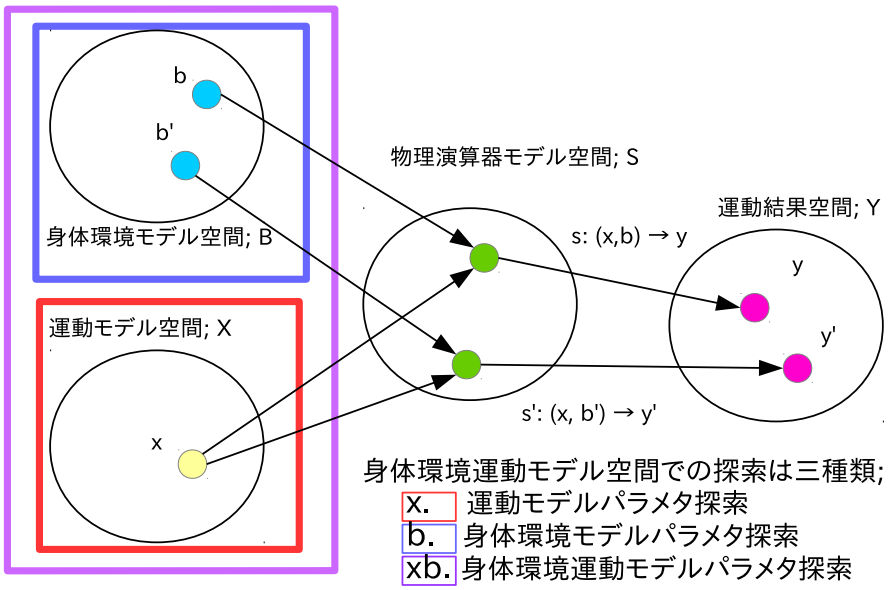


図 2.14. There exist 3 patterns of search problem; x) motion model parameter search, b) body and environment model search, and xb) both search.

図 2.15. There exist 3 patterns of search problem, 2 patterns of objective (e.g., optimization and identification), and their 6 combinations. However, we consider that motion model parameter is the same in real world and virtual world, 2 out of 6 combinations in esitimation problem have no sense. Hense, there are 4 problems; x1) optimize motion model parameter, b1) optimize body and environment parameter, b2) estimate body and environment parameter, and xb) optimize body and environment and motion model parameter.

図 2.14 のように支援システムでは身体環境運動モデルのパラメタから物理演算器を通して運動結果を得る計算を行い身体環境運動モデルのパラメタを探索する．本支援システムの実体

設計評価機構はこの身体環境モデル空間，運動モデル空間での探索を扱うものであるが，物理演算器モデル空間での探索は行わないとする．これは本研究で扱う歩行動作や着座行動の物理演算器は常に与えられるということであり，モデル化が十分検討されてない運動では物理演算器モデル空間での探索も視野に入れる必要がある．さて，二つの探索空間を考慮する場合，その探索方法は探索空間の組み合わせに基づいて三通りに分類できる．一つ身体環境モデル空間での探索，二つ運動モデル空間での探索，三つ両空間での同時探索である（図 2.14）．さらに探索の評価関数について，身体環境運動モデルのパラメタ最適化と実体モデルのパラメタ推定の二つを考慮すると全組み合わせは六通りに分類される（図 2.15）．ただし本研究では運動モデルのパラメタを仮想と現実で共通して用いることができると仮定しているため，その推定問題は考える必要がない．したがってそれを除いた全四通りの問題を考える．図 2.13 と見比べると，身体運動モデルのパラメタ生成は身体環境モデル空間と運動モデル空間での探索を組み合わせたもの．実体モデルパラメタ修正は身体環境モデル空間での推定と運動モデル空間での探索を組み合わせたものである．

2.4.3.1　運動モデル空間での探索

身体環境モデル空間，物理演算器モデル空間を固定し，運動モデル空間 $x \in X$ を探索する問題は所謂モーションプランニング，動作生成問題として古くから盛んに取り組まれてきた問題である．図 2.9，2.11，2.12 の上側の図のように探索は進む．式（2.1）に式を示す．

$$\begin{aligned} Optimize_{\ x} \quad & f(s(x,b)) \\ s.t. \quad & x \in X, s \in S, b \in B \end{aligned} \tag{2.1}$$

f は評価関数，Optimize の下付き添字は探索空間を表す意味で用いた．動作生成に限らず探索問題は評価関数・探索空間・制約条件により表現され評価関数は問題によって変わるため除外すると探索空間と制約条件に着目した小分類も考慮できる．特に接触の条件の扱いはロボットの行動探索問題の解き方を大きく変えるため接触遷移の有無による考察は有益である．

2.4.3.1.1　探索空間： 全身の自由度を用いた探索

剛体多リンク系モデルでは全アクチュエータの自由度とルートリンクの位置姿勢により全身の状態を表現することが可能であり，それら全体がもっとも広い探索空間となるが，ルートリンクが環境に固定されない等ルートリンクの位置姿勢を完全には決められないロボットでは冗長な自由度は運動方程式により決まる値となるため探索空間から除外するか，探索空間はそのままに運動方程式を制約条件とした探索を行う必要がある．また脚のみに限らず手や膝といった全身リンクの接触を扱う場合には転倒や関節負荷トルクの計算のために接触力が必要となる．接触力を扱う場合は探索空間に接触力を含める方法と [15][16]，接触力がロボットのコンフィギュレーションから一意に計算できるとする方法 [48][35] が考えられ，これらが物理演算器 $s \in S$ である．

2.4.3.1.2 制約条件： 探索範囲を狭めない接触制約

接触状態が変化するような行動生成問題では，全身のバランスや関節負荷トルク制限など不等式の拘束を考慮する必要がある．さらに固定された接触状態や，ベタ足のみでの歩行といった単一の接触状態遷移を用いた行動では，ロボットのもつ自由度の一部が無視されている．探索範囲を狭めない制約付き探索とは，関節負荷トルク制限といったハードウェアの持つ制限は考慮しつつ，探索範囲を狭めてしまうような接触状態の過度な制約を取り払った条件で，ロボットのコンフィギュレーション空間を探索することで達成できる．これはより一般的には，動力学シミュレーションの出力する全運動のなかで行動生成器を探索することである．

2.4.3.2 身体環境モデル空間での探索

受動歩行器では，身体環境モデルを探索することが歩行行動を探索することであるが，受動軸以外に能動軸を有するロボットでは運動モデルのパラメタを固定した状態で身体環境モデルのパラメタのみを変化させて探索しても意味のある行動を得ることは難しい．しかし，身体環境モデル空間での探索は運動モデル空間での探索と繰り返し行うことで二つの応用が考えられる．一つは身体環境モデル空間を全探索により変化させ固定してから運動モデル空間探索することで行動実現が可能な身体モデルのパラメタを探索する問題であり，身体環境モデルのパラメタを固定することで運動モデル空間探索が高速化できる場合に効率のよい方法である．もう一つは実体の運動結果，運動モデルのパラメタが分かっているとして同じ運動モデルのパラメタを用いて物理演算を行い運動結果を比較し差を最小化することで実体の身体環境モデルのパラメタを推定するという使い方である．以下，それぞれについてまとめる．

2.4.3.2.1 身体モデルのパラメタ生成： 運動モデル空間との繰り返し探索

身体環境モデルのパラメタを固定することで行動生成が効率よく行える場合がある．例えば関節負荷トルクを最小化するような姿勢探索手法については多数先行研究が存在し高速化や高精度化の工夫が議論されてきている．本論文でも探索時間と探索成功率や評価値の関係が詳細に調べ高速確実に解を得られるソルバを実装した．これら既存の探索器を身体環境モデルを変化させながら用いることで行動生成可能なモデルを獲得したり高評価な行動が可能となるようモデル探索したりといった応用が考えられる．

2.4.3.2.2 実体の身体環境モデルのパラメタ推定： 実機による計画動作の忠実実行

ロボットの行動が計算機上で求められたとして，それを現実世界で同じように実現するためには誤差の問題がある．本章ではロボットの身体環境モデルのパラメタの違いにより生じる物理演算と現実での運動結果のずれを低減する手法について概要を述べる．目的は物理演算の結果を現実で忠実に再現することであるが，もちろん誤差を零にすることは不可能である．しかし誤差を計算することができれば，その誤差を減らすような計算が可能であるし，その行動がどの程度忠実に実現できたのかを評価し後の研究につなげることができる．現実での

身体環境モデルのパラメタを $b' \in B$，仮想世界では $b \in B$，現実での物理演算器を $s' \in S$，仮想世界では $s \in S$ とすると，現実で起こる運動結果 $s' : (x, b') \to y' \in Y$ と物理演算のそれ $s : (x, b) \to y \in Y$ は同じではない．これはロボットの物理演算器と身体環境モデルのパラメタが現実と異なるためである．本研究では現実に十分近い物理演算器が与えられるとして，ロボットの身体環境モデルのパラメタ探索を行うことで誤差を吸収する．図 2.12 の下図のような流れで計算を行う．運動モデルパラメタ $p \in P$ を固定して身体環境モデルのパラメタ $b \in B$ を変化させることで運動結果 $y, y' \in Y$ の差を最小化する．以下の式（2.2）を解く．

$$\begin{aligned} Minimize_{\ b} \quad & ||s(x,b) - s'(x,b')||^2 \\ s.t. \quad & x \in X, s, s' \in S, b, b' \in B \end{aligned} \tag{2.2}$$

関連研究として [54] [55] [56] ではルートリンクの三次元位置姿勢を含む全関節リンクの一般化座標と，ルートリンクにかかる全外力を測定あるいは零とすることでロボットの運動方程式を表現する全動力学パラメタを同定できることが示されており，計算速度の面でも優れている．対して本章では，運動方程式の動力学パラメタに加え環境との接触力計算・摩擦力計算に用いるゲインや，摩擦係数，関節 PD 制御で用いるゲイン等も含めた推定を行うという探索空間での違いがある．また目的はロボットの持つ本当の動力学パラメタを獲得することではなく，ロボットの運動をもっともよく近似することであるという目的の違いがある．

2.4.3.3　身体環境運動モデル空間での探索

身体環境モデルのパラメタと運動モデルのパラメタが相互作用し分離できないような問題，例えばバネを用いた受動歩行におけるバネと歩行動作の関係を扱うような問題では，両空間での同時探索が必要となる．図 2.10 のような流れで探索を行う．式（2.3）を示す．

$$\begin{aligned} Optimize_{\ x,b} \quad & f(s(x,b)) \\ s.t. \quad & x \in X, s \in S, b \in B \end{aligned} \tag{2.3}$$

2.5　各章における実体設計評価機構の使い方と意義

下図では実体設計評価機構が扱う二つの問題，身体運動モデルパラメタ生成と実体モデルパラメタ修正について，以降の章で実際に解いていく問題の概要（図 2.16），探索空間（図 2.17），探索方法（図 2.18）について具体的にまとめそれぞれの違いと意義についてまとめる．本論文で扱う支援例はケーススタディとしての側面が強いが，それぞれ異なる適用範囲を持ち支援システムの有効範囲を確認する意味で意義のある例題となっている．

三章とその他の違いは探索空間として全身の一姿勢のみを扱っている点である．接触の変化がないこのような問題では勾配法を用いて高速に探索することが可能である一方，可操作域可視化などの応用先も多い．そこに接触力シミュレーションと動力学計算を追加することで姿勢の物理的実現性まで考慮することを可能にするとともに新しい応用例として支持脚の力学まで考慮した可操作域の可視化を行うことで股関節間距離パラメタ生成を行う．ここで探索する身

解く問題の具体的内容

	身体運動モデルパラメタ生成	実体モデルパラメタ修正
三章	股関節間距離身体パラメタと片足立姿勢運動パラメタ生成	
四章	着座支持棒身体パラメタと着座運動パラメタ生成	着座支持棒摩擦身体パラメタと着座運動パラメタ修正
五、六章	着座支持棒身体パラメタと着座運動パラメタ生成	身体環境パラメタと歩行運動パラメタ修正

図 2.16. Search problem settings of following chapters.

探索するパラメタ

	身体運動モデルパラメタ生成	実体モデルパラメタ修正
三章	全身関節角度、股関節間距離パラメタ	
四章	全身関節角度列、接触状態列、接触高さパラメタ	滑り摩擦係数
五、六章	全身関節角度列生成器パラメタ、バネパラメタ	動力学モデルパラメタ、接触モデルパラメタ

図 2.17. Search spaces of following chapters.

探索の仕方

	身体運動モデルパラメタ生成	実体モデルパラメタ修正
三章	股関節間距離を変えて勾配法での全身関節角度探索	
四章	接触高さを変えて勾配法での姿勢列探索、接触グラフ探索	センサデータより測定
五、六章	進化計算で全身関節角度列生成器・バネパラメタ探索	進化計算で測定可能値から不能値推定し運動再生成

図 2.18. Search methods of following chapters.

体モデルパラメタはリンクの長さであり，ロボットの外形や可動域を決める重要なパラメタであり応用範囲は広い．

四章とその他の違いは，一つは摩擦を利用した動作は着座や立ち上がりといった足の接触に限らない動作で重要な役割を果たすものであるが，摩擦は実体と設計で誤差が生じやすいことが知られており行動実現は制御系のロバスト性やヒューリスティックパラメタに頼っていた．本支援システムでは摩擦を推定しながら動作実現でき新規性があり具体例として分かりやすい．また実体モデルパラメタ修正においてセンサから直接計測できる値を修正していくもっと

も簡単な例となっている点が五章，六章と異なり，オンラインでのパラメタ修正を実現している点も重要である．またここで探索する身体モデルのパラメタは接触点であり，配線等の都合により全身が環境に接触できるよう設計することが難しいロボットでは特定の接触点をあらかじめ設計に組み込むことがしばしば行われるため応用範囲は広い．

五章，六章では制御系パラメタ探索や衝突を伴う動的運動生成を扱うため，高速な勾配探索手法が適用できず進化計算手法を用いる．しかし進化計算手法は低速であり身体モデルのパラメタを網羅的に変えながら動作モデルパラメタを求めることはできないという制約がある．動作と身体モデルのパラメタを同時に探索することで小規模な問題なら扱えることがすでに示されているが [51]，本論文では不安定な動歩行を応用例として用いている点に新規性がある．さらに非剛体であり制御できない機械要素であるバネと受動軸を扱う問題であるため実体モデルパラメタ推定が必須であることと合わせて，身体行動創成支援システムの性能を具体的に示す評価として挑戦的な問題設定になっている．また解ける探索空間規模を増やすために重心フィードバックとステートマシンの簡単な構成ながら多様な運動を表現できることができると知られる SIMBICON [42] モデルを採用し運動モデルのパラメタとして百次元程度の規模の問題を扱うことで動歩行制御系パラメタの探索に成功している点が応用面で重要な発見である．ここで探索する身体モデルのパラメタは受動機械要素であり，制御が難しいためロボットの設計に用いられることは少ない．受動機械要素が扱えることを示すことそのものにも意義がある．

2.6　おわりに

本章ではロボット開発全体の流れを本論文で開発した等身大準受動脚型ロボットを具体例として見ていくなかで支援システムが扱う範囲と限界についてまずまとめた。本支援システムは人間により設計が行われる中で問題のモデルが定義された後に仕様から決定されない身体運動モデルのパラメタの生成をまず支援する。次にロボット完成後に、設計と異なる実体の身体環境モデルのパラメタを推定後に運動モデルのパラメタを再生成することで行動実現を支援する。以降の章ではこの実体設計評価機構に基づく支援例を示していく。

第 3 章

全身姿勢探索による片足支持に適した股関節間距離の決定

3.1 はじめに

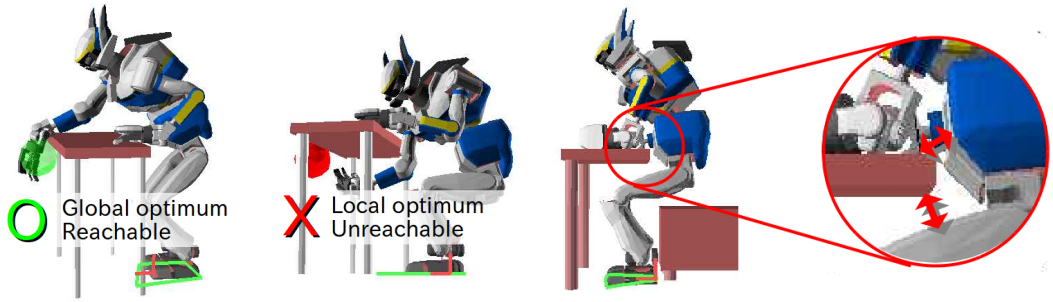
本章では全身自由度での姿勢探索手法とそれを用いた身体環境モデル空間での探索手法について扱い、身体モデルのパラメタ生成の例題として片足支持に適した股関節間距離の決定問題を解く。図 2.9 が解く問題のモデルと探索法をまとめたものである。支援システムにおけるロボット完成前の身体運動モデルのパラメタ生成において身体モデルのパラメタ変更と運動モデルのパラメタ生成を交互に繰り返す例を示す。本章で用いる姿勢探索手法は姿勢の探索のみで身体モデルのパラメタを評価できるような問題、例えば立ち姿勢に要求があるような場合に有効な手法であり、本章の高速化手法により網羅的に身体モデルのパラメタについて運動モデルのパラメタを探索し評価を可視化することで人間の設計における決断を助けるものである。

全身自由度を用いた動的運動の探索手法は近年多くの研究であつかわれている [15][57]。残された課題は計算速度である。[57] では全身三十三自由度の最適化計算に数時間から十時間程度を要しており、低速なスクリプト言語を用いていることを考慮しても数十分から一時間程度の計算が現状必要な規模の問題となっている。以下では関連した話題として全身姿勢最適化計算の高速化手法について論じる。まず全身姿勢探索について深く調べることが軌道最適化を高速化する手法に繋がっていくと考えているためである。全身自由度探索の計算時間を大きくしている問題は二つある。一つはロボット冗長な自由度を持つため複数の局所解が存在しそこにとらわれない手法が必要である点、もう一つは位置だけでなく関節負荷トルクまで考慮した際計算量が増大する点である。以下では前者について学習を用いた高速で初期値非依存な探索手法の提案を行う。後者については姿勢最適化における勾配の計算量について考察を行い、高速計算可能な近似勾配を提案する。

3.2　オートエンコーダを用いた冗長埋め込みによる高速な初期値非依存探索法

局所解に陥らないためには大域的な探索が必要となるが計算時間が必要となる．以下では学習を用いることで初期値に依存しない探索手法の高速化について述べる。なお、本章の内容は [20] にて発表した内容である。

ロボットの動作計画は，手先エンドエフェクタの位置姿勢空間といったタスクスペースでの探索から，全身の関節角度列空間といったコンフィギュレーションスペースでの探索 [16][34] へと発展しつつある．ランダムサンプリングアルゴリズムや非線形最適化といった探索手法を用いることによって，全身を用いた複雑な動作生成がオフラインでは可能になる一方で，オンラインでの探索には速度の面で未だ課題が残っていると言える．全身自由度を考慮し実時間で逐次的に目標動作を達成するための制御については，計算機の性能向上とアルゴリズムの改良により実現されつつあるが [58]，ロボットの動作計画では, エンドエフェクタの位置姿勢を保ち接触を維持するといった非線形の等式制約, 重心位置を支持領域内に収めるといった不等式制約を複数考慮する必要があり, 特に, 環境との衝突を回避といった, 非凸な制約を持つ問題（図 3.1） は複数の局所解を持つため, 山登り法といった局所探索手法 [19][18] では安定に解を求めることが難しいという課題がある．ランダムサンプリング手法 [59] を用いることで局所解を回避することが可能であるが, サンプリング空間の次元に対して指数関数的に問題が複雑化するため, サンプリング空間を狭める工夫 [7] が必要である．



1) Several local solutions due to the non-convexity of environment.　　2) Narrow feasible space for keeping contact and balancing

図 3.1. Our algorithm is fast and applicable to the problems which have several local solutions and narrow feasible space. (reprinted from [20])

本章で示す手法は局所解に陥らないためにランダムサンプリングアルゴリズムをベースとしつつ，探索を高速化するために探索空間の削減を行う．探索空間はロボットの全自由度空間（状態空間）ではなくタスク冗長空間，すなわちタスク空間に対する状態空間の冗長な自由度からなる空間を用いる．タスク冗長空間はオリジナルの探索空間である状態空間に比べて小さく，冗長空間での探索は状態空間での探索よりも容易であり高速化が見込める．このアイデア

実現のために問題となるのは以下にしてタスク冗長空間を計算するかという点である．一般にタスク冗長空間は状態空間に対して非線形の関係を持ち，陽に計算によって求めることはできない．3.2.1.1 章では，ニューラルネットワークと deep auto encoder を用いて計算するタスク冗長空間から状態空間への写像（タスク状態写像，TSM: Task State Map）のアイデアを示す．ニューラルネットワークの次元圧縮能力により，TSM は非常に多くのタスクと状態の組を学習でき，タスクから対応する状態の値を高速に計算することができる．本章の内容のもっとも重要な貢献はこの TSM のアイデアを示したことと，その性能について分析を行ったことであると考えている．加えて，多くの先行研究 [60][61] がタスクと状態の写像，例えば逆運動学問題について取り扱ってきているが，ほとんどは単腕 6 自由度を扱ったものであり，ヒューマノイドロボットの全身関節角度空間（本章では 28 自由度）についての学習は本研究が初めてである．本章では具体的なタスクとして全身自由度を状態空間とし，四肢先端の位置姿勢をタスクとした逆運動学問題を扱い，狭い隙間を歩いて脱出する問題（egress task） と複数局所解を持つテーブル裏へのリーチング問題を解く．

3.2.1 冗長埋め込みと探索空間削減

本章のアイデアは二つの構成要素を持つ．一つは，タスクとタスクの冗長空間からロボットの状態空間への写像を学習することである．以下ではこの写像を TSM (Task-State Map) と呼ぶ．図 3.2 は TSM の模式図を表す．二つめは，TSM を用いることによる探索空間の削減である．TSM を用いることで探索空間をロボットの状態空間からタスク冗長空間へと変更することができる．TSM は例えばエンドエフェクターの位置姿勢といったタスクと，タスク冗長空間を表すパラメタを引数にとり，ロボットの状態パラメタを返す．例えば七自由度マニピュレータであればエンドエフェクタの位置姿勢六自由度に対して一つの冗長なパラメタを持ちこれがタスク冗長空間を表す．タスクパラメタを固定し，タスク冗長空間のパラメタを変更しながら TSM を用いてロボットの状態パラメタを計算することで高速な探索が実現できる．

3.2.1.1 冗長埋め込み： タスク空間とタスク冗長空間からロボットの状態空間への写像

多関節ロボットの冗長な自由度を扱う研究は過去に多く存在したが，その冗長な自由度（以下ではタスク冗長空間）を如何にしてパラメタ化するかという問いに対する答えは明らかではなかった．本章では，タスク空間とタスク冗長空間からロボットの状態空間への写像（TSM: Task-State Map）をタスクと状態ベクトルの組からなる教師データセットを用いて学習する手法を提案する．学習にはオートエンコーダ [13, 11, 12] を用いる．オートエンコーダはニューラルネットワークの一種であり，入力層と出力層に同じ値を用いて学習することで，入出力層よりも小さな中間層に教師データの圧縮された表現を得ることができる．

図 3.3 では，青枠で囲まれた部分が入出力に等しい値を持つオートエンコーダ部，赤枠で囲まれた部分がタスクとその冗長空間からロボットの状態空間を計算する TSM 部である．図 3.3 は入力層を二種類持つことがオリジナルのオートエンコーダとは異なっている．二種類の入力層はそれぞれ図中左端のロボットの状態ベクトル入力層と，中間のタスクベクトル入力層

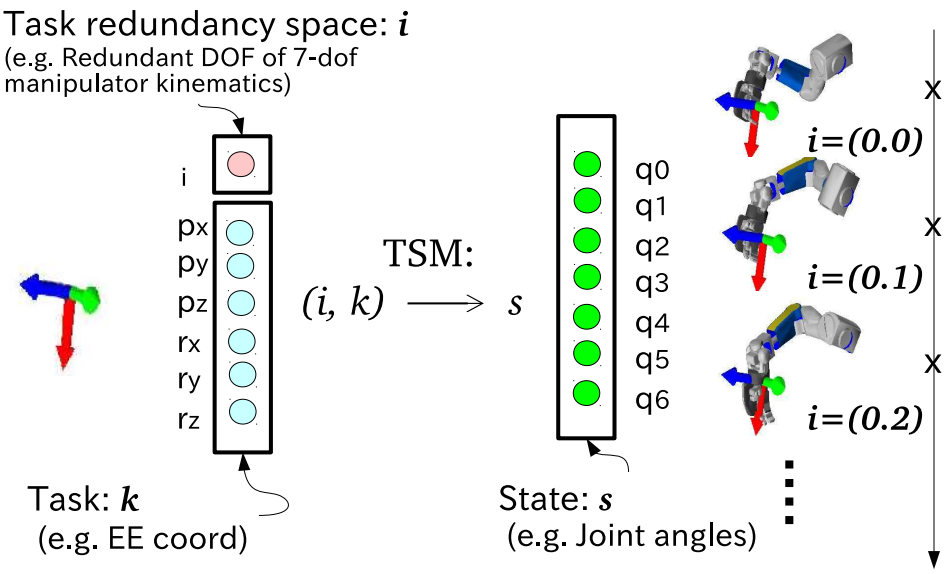


図 3.2. Redundancy embedding: Task-state map is a map from task space and task redundancy space towards state space (reprinted from [20]).

である．中間層は加えてオートエンコーダで圧縮される層を持ちこれがタスク冗長空間に対応する．タスク冗長空間パラメタはシグモイド関数を用いることで零より大きく一より小さな値に正規化される．したがって，このネットワークで入出力が学習されることで，赤枠部はタスクベクトルと零以上一以下の要素からなるタスク冗長空間ベクトルを入力としてロボットの状態ベクトルを返す TSM となる．このネットワークが学習可能であるかについては，図中赤枠のみでみれば通常のニューラルネットワークとまったく同じであるため明らかに可能である．図中赤丸のノードは学習中変化しつづけるが層が十分深ければ赤丸が乱数であれ何であれ学習は可能であるため（そのような学習に意味があるかは置いておいて）学習が可能か否かについては可能であることが分かる．また赤枠を除いた部分のネットワークについても層が十分深ければ通常のニューラルネットワークであるので仮に赤丸がどのような値になろうとも学習は可能である．

3.2.1.2　探索空間削減： 探索空間を状態空間からタスク冗長空間へ

本章では，コンフィギュレーションスペースでの探索問題を冗長空間での探索に変換する手法について述べる．一般に非線形最適化問題は以下の形式で表現できる．（図 3.1）：

$$
\begin{aligned}
& min_{\boldsymbol{s}} f(\boldsymbol{s}) \\
& s.t. \\
& \quad g(\boldsymbol{s}) = 0 \\
& \quad h(\boldsymbol{s}) < 0
\end{aligned}
\tag{3.1}
$$

$\boldsymbol{s}$ は探索空間であり，ロボットの関節角度列といったコンフィギュレーションスペースを表す．f, g, h はそれぞれ非線形最適化における評価関数，等式制約，不等式制約を与える式であ

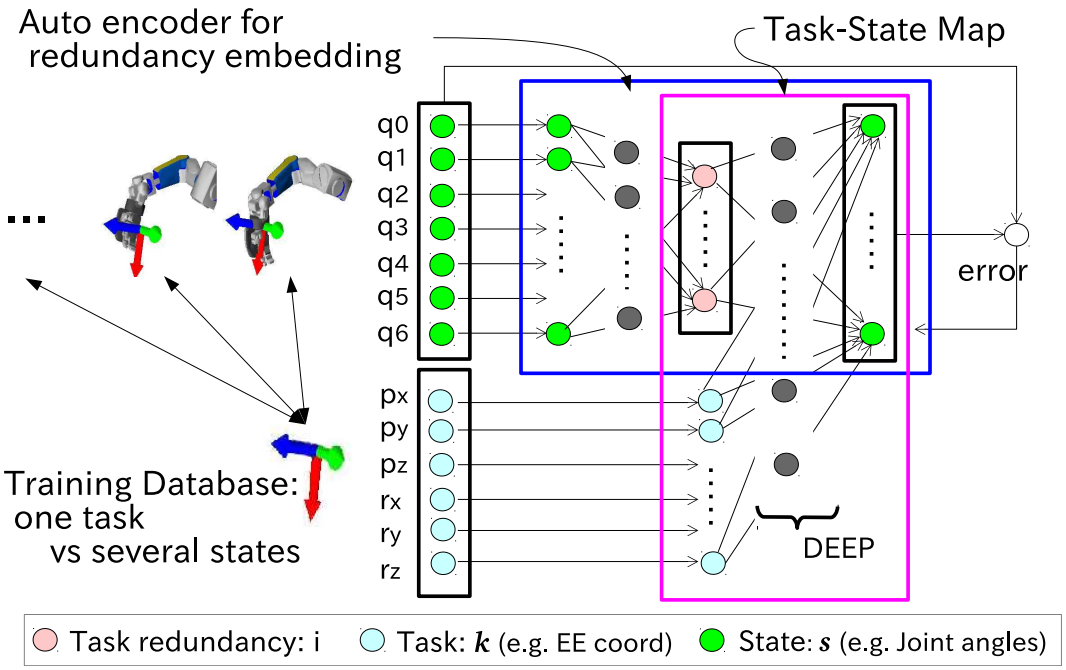


図 3.3. Deep-auto-encoder-based task-state map with two input layers: Robot's state input in left layer and task input in middle layer. Besides, middle layer also has task redundancy space parameters which is automatically embedded from training robot's state values. (reprinted from [20])

る．例えば，f として関節負荷トルクの大きさを用い，g に手先エンドエフェクタの目標位置姿勢の距離を用いれば，関節負荷トルクを最小化する逆運動学問題を表すことができる．

TSM はタスク空間のパラメタとタスク冗長空間のパラメタからロボットの状態空間のパラメタを返すものであると 3.2.1.1 章で説明した．したがって，TSM は式 3.1 の記号を用いて以下のように定義できる．

$$\boldsymbol{s} = TSM(\boldsymbol{i}, \boldsymbol{k}) \tag{3.2}$$

式（3.2）を用いると，タスク $\boldsymbol{k}$ を満足するロボット状態パラメタ $\boldsymbol{s}$ を探索する問題は，TSM の $\boldsymbol{k}$ を固定した状態で $\boldsymbol{i}$ を探索する問題に変換できる．

$$\begin{aligned} & min_{\boldsymbol{i}} f(\boldsymbol{s}) \\ s.t. \quad & \boldsymbol{s} = TSM(\boldsymbol{i}, \boldsymbol{k}) \\ & g(\boldsymbol{s}) = 0 \\ & h(\boldsymbol{s}) < 0 \end{aligned} \tag{3.3}$$

タスク冗長空間はオリジナルの探索空間よりも小さいため，式（3.3）の計算は式（3.1）の計算よりも高速に実現できる期待できる．

TSM はニューラルネットワークを用いているため教師データとの間に学習誤差を持つ．この誤差を吸収するため本研究では $\boldsymbol{s}$ まわりでの局所探索を組み合わせる．同様に局所探索を

組み合わせてサンプリング手法を用いることでタスク制約を満たす状態パラメタを高速にサンプリングする研究として [7] では乱数を初期値としてタスク制約を満たす解を勾配法で求めていた．本研究でも局所探索として勾配法を用いることは変わらないが，TSM ではサンプリングされる状態パラメタがタスク制約を満たす解のすぐそばに計算されるため局所探索は高速である．

3.2.1.2.1　データベースアプローチ

探索を高速化するための一つの簡単な方法としてはタスクに対する実現可能な解をあらかじめデータベースとして蓄えておくというものが考えられる．例えば，中川ら [15] は multi-dimensional B-spline お重み係数を用いてロボットの状態とタスク間の写像を表現し，その重みが実現可能な動作の条件を満たすように最適化することで高速化を実現した．しかしながら，このようなデータベースアプローチでは次元の呪いが課題として残る．タスク空間を等分してつくるデータベースはそれを表現するためのパラメタの数が指数関数的に増加するからである．具体的に計算してみると，三十自由度のロボットの全関節角度と四肢のエンドエフェクター位置姿勢六自由度（合計二十四自由度）の組み合わせをデータベース化することを考える．関節角度の空間を一次元につき五つのブロックに分け，それぞれの姿勢について順運動学を解くことで四肢のエンドエフェクター座標を計算すると，その組み合わせの数は $24 \times 5^{30} \approx 2 \times 10^{22}$ となる．関節角度を一次元あたり一バイトで表現したとしても必要なメモリは 10^{10} テラバイトとなり現状準備することは困難なサイズとなる．また，ロボットはタスクに対して冗長な自由度を持ちうるが，データベース上でいかにしてそのタスク冗長空間を表すパラメタを表現するかという課題も残されていた．本章の内容もまた一種のデータベースアプローチであるが，ニューラルネットワークを用いて次元圧縮することで，三十自由度程度の学習でも数メガバイトのメモリ量でデータベースを構築できることが以下では示される．また，オートエンコーダを用いることで冗長なタスク冗長空間を自動的に獲得するネットワークである．

3.2.1.2.2　Reachability Map

Reachability Map はロボットのエンドエフェクタが届く領域を表現するデータベースであり，リーチングタスクを成功できるロボットのルートリンク位置を決定するといった応用で用いられる [62]．さらに自己干渉といった他の制約を考慮できるような発展を遂げてきている [63]．本章の TSM はリーチ可能かではなくリーチできる解を蓄えるデータベースであるという点が Reachability Map とことなっている．したがって，解があるかどうかを調べたいといった問題（先のリーチングタスク可能なルートリンク配置問題など）では Reachability Map が効果的である．逆に，リーチングタスクの解を得たい場合には TSM がより有効である．

3.2.1.2.3 Model Reduction

高速化手法としては Model Reduction 手法もしばしば用いられる [64][23]. Model Reduction の目的は解の精度を保ちつつ探索空間を低次元化することである．対して TSM は探索空間を状態空間からタスク冗長空間へと低次元化する点は Model Reduction と類似しているが，解の精度は学習誤差のみに依存し学習誤差を下げれば下げるほど解の精度も上がっていくという特徴がある．以下の章では，狭い空間からの脱出タスクを取扱い，解空間が狭い問題についても TSM が適用可能であることを示す．

3.2.1.2.4 TSM の課題

TSM の課題の一つは学習段階で長い時間を要する点である．以下の実験では三十自由度値度の学習に二日程度かかっている．TSM はタスク毎に学習されるので，四肢を持つロボットの逆運動学問題をすべて学習するためにはそれらの全組み合わせの数学習を行わなければならない．その数は ${}_4C_1 +_4 C_2 +_4 C_3 +_4 C_4 = 15$ である．これらの学習は単純に並列化することができるがスレッド数の多いコンピュータが必要である．あるタスクで学習された TSM を他のタスクに再利用するためのアプローチとしては，式（3.3）いおけるタスクパラメタ k の一部を探索空間に含める方法が考えられる．左右の足の位置姿勢をタスクとする TSM を例にとってみると，左足を固定して右足を動かすタスクは，左足のタスクパラメタ $\boldsymbol{k}_l$ を固定した状態で，右足のタスクパラメタ $\boldsymbol{k}_r$ と冗長空間パラメタ $\boldsymbol{i}$ を探索することで解かれる．以下の章では，章 3.2.3.2 にて，リーチングする手の姿勢のみを追加の探索空間に含める例を扱う．タスクをどのように決定するかという方法論は TSM を用いた手法の性能を左右するトピックであるため，今後も研究の余地が残っていると考えている．

3.2.2 Task-State Map の学習実験： 単腕・双腕・全身のキネマティクス

本章では TSM のネットワーク構成と学習性能について調べる．ロボットのモデルとしては HRP2 [65] を改良した hrp2jsk [66] を用いた．

3.2.2.1 学習設定とアルゴリズム

それぞれの層の全ノードは隣合う層の全ノードと hyperbolic tangent 関数（$\frac{e^x - e^{-x}}{e^x + e^{-x}}$）を通して全接続される．誤差逆伝播するための誤差関数はネットワークの出力と教師データとの間の平均二乗誤差（MSE: Mean Square Error）を用いた．学習アルゴリズムとしては，オープンソースのディープラーニングフレームワークである Caffe [45] に実装された Nesterov's accelerated gradient [67] を用いた結果を以下では示すが，SGD を用いてもほぼ同様の結果が得られた．学習率は 1000 イテレーション毎に 0.977 倍した．これは 10000 イテレーションで $0.977^{10000/1000} \approx 0.1$ 倍になるよう決めた．TSM の学習では，教師データが状態空間の次元数に対して指数関数的に増えていくという難点があるため，本章では 100000 イテレーション毎に教師データを作り直すことで一度に準備する教師データの数を小さくする工夫を

行った．学習手続きを以下にまとめる．

1. 教師データ初期化: 全 N 関節の可動域を等しく n 等分する．n は全組み合わせの数 n^N が大きくなりすぎないよう 10^6 以下となるように決定した．n 等分した全組み合わせの姿勢に対して順運動学を解くことでエンドエフェクタ位置姿勢を計算し教師データとして用いた．
2. 学習率初期化：ヒューリスティックに決定した三つの学習率とそれぞれに対して三つのランダムに初期化されたネットワーク合計九つを生成する．九つのネットワークを 100000 イテレーション学習させもっとも誤算関数の小さくなったものを以下の手続きで用いる．
3. 学習: 100000 イテレーションの学習を行い, ネットワークを評価する．
4. ループ回数上限確認：全体のイテレーションが 40 x 100000 回を超えたら終了し，8) へ移動する．
5. 収束条件確認：評価値の減少が 20 回起こらなかった場合終了し，8) へ移動する．
6. 学習率更新：評価値が前の 100000 イテレーションよりも下がった場合は変化なし, 上がった場合は 1/1.1 倍, 変わらなかった場合は 1.1 倍として更新する．
7. 教師データ更新：ランダムに教師データを更新して, 3) にもどる．
8. 出力：もっとも評価の高かったネットワークを出力する．

学習ゲインの初期値三パターンは, 単腕の学習について,0.01 0.005 0.001, 双腕の学習について 0.005 0.001 0.0005, 全身の学習について 0.001 0.0005 0.0001 を発見的に用いた．ネットワークの評価には, ランダムに生成されたロボットの状態パラメタともっとも近いものをタスク冗長空間パラメタを百分割して探索し，さらにその距離を百回計算して平均した値を用いた．この値は表 3.1, 3.2, 3.3 では SCORE として表記されている．この評価値が 0 であれば, ネットワークはロボットのコンフィギュレーション空間を完全に表現したことがわかる．

3.2.2.2　ネットワークパラメタに対する Task-State Map 性能評価

ネットワーク構成を変化させたときに TSM の性能がどのようになるかを評価するために,TSM を図 3.4 のようにパラメタ化する．s, k, i はそれぞれ, ステート, タスク, タスクの冗長パラメタの次元であり, 本章では以下の値を用いた．

- 単腕:
  - $s = 9$: 七自由度の右腕と二自由度の腰関節
  - $k = 6$: 右腕エンドエフェクタの位置と姿勢
  - $i = 1, 2, 3$: 冗長な自由度の大きさは $s - k = 3$ と同じか小さい．
- 双腕:
  - $s = 16$: 七自由度の両腕と二自由度の腰関節
  - $k = 12$: 両腕エンドエフェクタの位置と姿勢
  - $i = 1, 2, 3$: 冗長な自由度の大きさは $s - k = 4$ と同じか小さい．

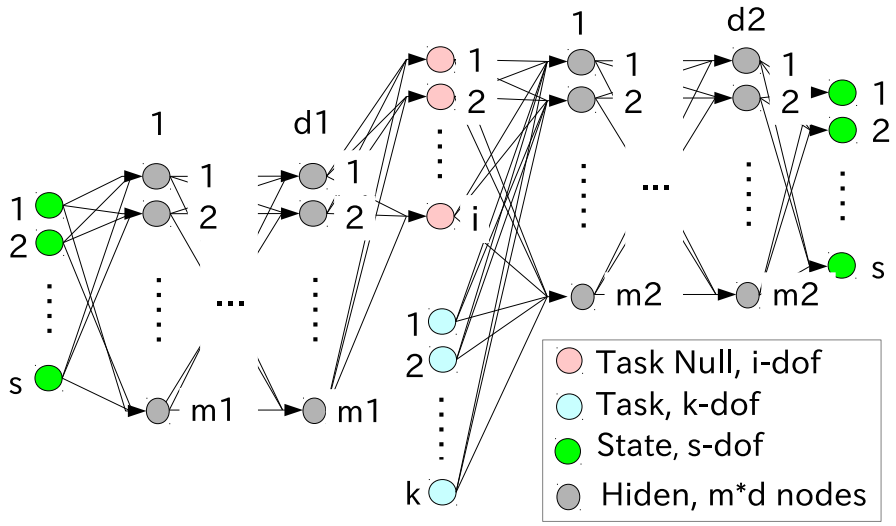


図 3.4. For testing the TSM network configuration, the sizes of nodes are parameterized. s, k, i are the degrees of freedoms of state, task and task redundancy space. The hidden layer of the left side has m1 × d1 nodes, and the right side has m2 × d2 nodes. (reprinted from [20])

- 全身:
  - $s = 28$: 七自由度の両腕と六自由度の両足，腰の二自由度
  - $k = 18$: 両腕と左足の位置姿勢六自由度，右足は全身多リンク系のルートリンクとする
  - $i = 1, 3, 5$: 冗長な自由度の大きさは $s - k = 10$ と同じか小さい.

i はヒューリスティックに $1, 2, 3$ あるいは 5 とした.

また, ネットワーク左側の隠れ層は, 各層のノード数が m1, 層の数が d1, ネットワーク右側は, ノード数 m2, 層数 d2 とした. $d1, d2, m1, m2$ とそれに対応する学習結果を表 3.1, 3.2, 3.3 にまとめる. いずれの学習も Intel Core i7-4790 CPU, 3.60GHz を用いた. すべての表は学習途中でもっとも小さかった SCORE をによって降順にソートされている.

表 3.1. TSM configurations and learning results ($s = 9, k = 6$)

m1	d1	i	d2	m2	TIME	SCORE ± SD
70	2	2	4	70	2.0h	0.15 ± 0.053rad
70	1	2	4	70	17.4h	0.16 ± 0.054rad
70	4	2	4	70	2.8h	0.17 ± 0.065rad
70	3	2	3	70	17.6h	0.17 ± 0.153rad
70	1	2	3	70	1.2h	0.18 ± 0.058rad
70	2	3	4	70	2.0h	0.19 ± 0.063rad
70	2	1	4	70	17.2h	0.19 ± 0.150rad
70	1	3	3	70	1.2h	0.20 ± 0.079rad
70	1	2	2	70	16.5h	0.21 ± 0.069rad
70	4	3	4	70	20.6h	0.21 ± 0.198rad
70	3	3	3	70	18.8h	0.21 ± 0.202rad
70	4	1	4	70	18.1h	0.22 ± 0.155rad
70	3	1	3	70	2.1h	0.23 ± 0.087rad
70	1	1	4	70	1.7h	0.23 ± 0.092rad
70	1	1	3	70	16.5h	0.23 ± 0.089rad
70	1	3	2	70	0.8h	0.23 ± 0.089rad
70	2	2	2	70	16.8h	0.24 ± 0.173rad
70	2	3	2	70	1.2h	0.24 ± 0.088rad
70	1	3	4	70	1.7h	0.25 ± 0.087rad
70	2	1	2	70	16.8h	0.25 ± 0.085rad
70	1	1	2	70	16.3h	0.26 ± 0.091rad

表 3.1 は単腕の学習結果を表す．全学習が二十時間以内に終わっていることが確認できる．上位五つのネットワークは同じ $\boldsymbol{i}$ を持ちその値は 2 であった．

表 3.2. TSM configurations and learning results ($s = 16, k = 12$)

m1	d1	i	d2	m2	TIME	SCORE ± SD
150	2	3	5	150	11.0h	0.21 ± 0.068rad
150	2	2	5	150	11.0h	0.22 ± 0.061rad
150	1	2	4	150	21.3h	0.23 ± 0.077rad
150	1	3	4	150	21.4h	0.23 ± 0.065rad
100	2	3	5	100	20.1h	0.25 ± 0.087rad
100	2	2	5	100	20.0h	0.26 ± 0.074rad
100	1	3	4	100	18.2h	0.26 ± 0.083rad
100	2	2	4	100	19.0h	0.26 ± 0.083rad
100	2	3	4	100	19.2h	0.27 ± 0.063rad
100	1	2	4	100	18.0h	0.27 ± 0.093rad
150	1	2	3	150	19.4h	0.27 ± 0.094rad
150	2	1	5	150	10.9h	0.28 ± 0.092rad
150	1	3	3	150	19.6h	0.29 ± 0.082rad
150	2	1	4	150	23.0h	0.29 ± 0.087rad
100	1	2	3	100	16.8h	0.30 ± 0.092rad
150	2	2	4	150	22.9h	0.30 ± 0.107rad
100	2	1	4	100	19.2h	0.31 ± 0.104rad
150	1	1	3	150	19.5h	0.31 ± 0.109rad
100	1	3	3	100	17.1h	0.33 ± 0.100rad
100	1	1	3	100	17.1h	0.34 ± 0.104rad
100	1	1	4	100	18.1h	0.36 ± 0.105rad
150	2	3	4	150	23.0h	0.39 ± 0.110rad
100	2	1	5	100	20.1h	0.40 ± 0.116rad
150	1	1	4	150	21.4h	0.44 ± 0.152rad

表 3.2 は双腕の学習結果を示す．全ての学習は一日以内に終了した．上位四つのネットワークでは同じ $m1, m2$ が用いられておりその値は 150 であった．また 150 は実験に用いた全 $m1, m2$ の値のなかで最大であったので，より大きな値を用いることで双腕での学習はより SCORE を下げられる可能性がある．

表 3.3. TSM configurations and learning results ($s = 28, k = 18$)

m1	d1	i	d2	m2	TIME	SCORE ± SD
150	4	3	8	150	40.3h	0.34 ± 0.061rad
150	3	3	8	150	38.5h	0.34 ± 0.076rad
200	3	3	6	200	44.3h	0.35 ± 0.088rad
200	3	3	8	200	50.0h	0.35 ± 0.071rad
200	4	3	8	200	52.8h	0.35 ± 0.069rad
150	3	3	6	150	34.9h	0.36 ± 0.069rad
200	2	3	4	200	35.3h	0.37 ± 0.066rad
150	2	3	4	150	28.1h	0.37 ± 0.102rad
150	4	5	8	150	40.4h	0.38 ± 0.073rad
200	3	1	8	200	50.0h	0.38 ± 0.090rad
200	2	1	4	200	35.1h	0.38 ± 0.080rad
200	4	5	8	200	52.9h	0.38 ± 0.083rad
200	4	1	8	200	53.0h	0.38 ± 0.081rad
150	3	5	8	150	38.6h	0.39 ± 0.081rad
150	2	1	4	150	28.2h	0.39 ± 0.091rad
150	4	1	8	150	40.4h	0.39 ± 0.073rad
200	3	5	8	200	50.0h	0.40 ± 0.075rad
150	3	1	8	150	38.7h	0.40 ± 0.074rad
200	3	1	6	200	44.5h	0.41 ± 0.081rad
200	3	5	6	200	44.3h	0.41 ± 0.099rad
150	3	5	6	150	34.9h	0.41 ± 0.077rad
150	3	1	6	150	34.9h	0.42 ± 0.097rad
200	2	5	4	200	35.2h	0.44 ± 0.098rad
150	2	5	4	150	28.1h	0.44 ± 0.074rad

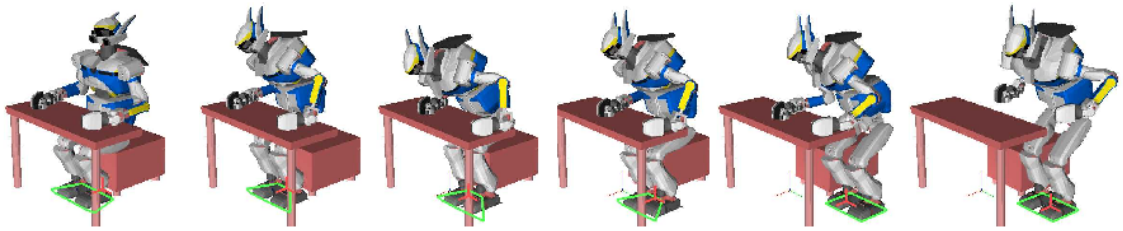
表 3.3 は全身自由度での学習結果を示す．学習には二三日かかっていることが分かる．上位八つのネットワークは全て $i = 3$ であった．$i = 3$ のネットワークは全てで八つなので，$i = 3$ が上位を独占していることが分かる．i が大きくなるにつれて TSM の学習能力は高くなっていくと考えられるが，ネットワークの評価として用いた SCORE は ランダムなロボットの状態パラメタに対して 100 回 i を変化させたときの最小距離であるので，i が大きすぎるネットワークでは十分に近い答えを見つけることができずに SCORE が大きくなってしまったためと考えられる．表中でもっとも小さな SCORE は 0.34 ラジアン（約 20 度）で，これは 180 度の可動域をもつ関節角度を三等分したデータベースの SCORE と同程度であるこのようなデータベースを作るには四バイト浮動小数で関節角度ベクトルを表現したとして

も $4 \times 3^{28} \approx 100 \times 10^{12}$ バイト（100TB）程度のメモリが必要であるが，本章で学習された TSM は 1.4 メガバイト程度であった．

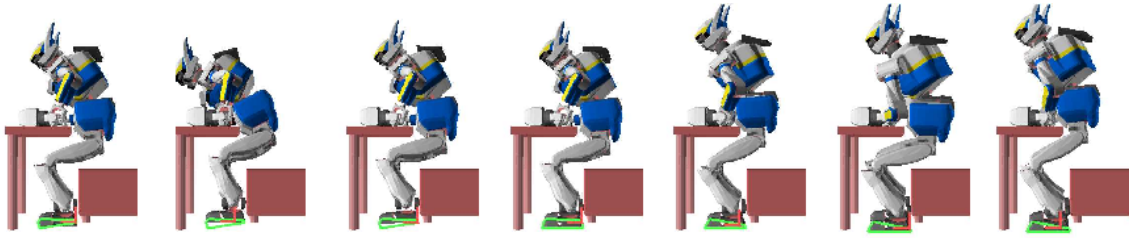
3.2.3 探索実験: 狭い隙間からの脱出と遮蔽物裏へのリーチング

本章では狭い空間からの脱出タスクと，遮蔽物の裏に手を伸ばすリーチングタスクを解くことで，TSM を用いた動作生成を実演し，残された課題について考察を行う．

3.2.3.1 狭い空間からの脱出タスク



1) Front view of the egress postures



2) Side view of the egress postures

図 3.5. The application result of the egress posture generation. Total 82 postures with given contact states without collision are calculated in 2.2 seconds. For balancing and avoiding collision, forward-bent kneeling postures are needed and these complex postures are stably calculated. (reprinted from [20])

本章では，TSM を用いた動作生成により狭い空間からの脱出タスクを解く．全接触位置姿勢はヒューリスティックにより与えられ，八十二の姿勢を全身関節角度の空間で学習した TSM を用いて計算する．

図 3.6 はアルゴリズムの流れを示す．まず，零以上一以下のランダムなタスク冗長空間パラメタ $\boldsymbol{i}$ と与えられたタスクパラメタ $\boldsymbol{k}$ を用いて TSM （ 3.2）を計算することでロボットの状態空間パラメタ $\boldsymbol{s}$ を得る．さらに，得られた $\boldsymbol{s}$ を初期値としてタスク制約（両手両足の位置姿勢）が十分満たされるようニュートン法を用いた局所探索を行う．局所探索は位置が 5mm 姿勢が 0.05radian（約 3 度）の誤差で目標値に近づくまで行った．ここまでの手続きには百

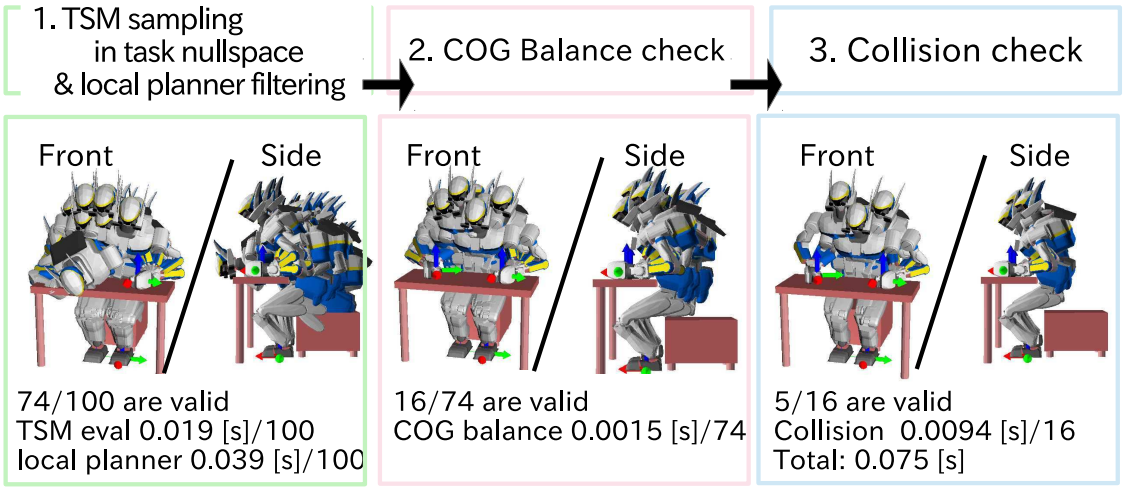


図 3.6. Sampling in task redundancy space and filtering for balancing and collision avoiding. 100 sampling took 0.075 seconds, and 5 feasible solutions are found by using Intel Core i7-3520M CPU 2.90GHz. (reprinted from [20])

点のサンプリングで 58 ミリ秒，一つの姿勢あたり 580 マイクロ秒かった．次に重心位置がロボットの支持多角形に含まれているかを確認した．支持多角形は両手両足の位置を地面の高さに投影したときにできる四角形を用いたがより正確な支持領域の計算としては一般化 ZMP がある [68]．最後に全ての衝突が起こりうるリンクペア（ここでは 196 組）について凸形状の衝突計算を行い，自己干渉のない姿勢のみを出力する．図 3.6 の例では百点のサンプリングで五つの解が 75 ミリ秒で得られた．

表 3.4. The computation times of egress posture generation

i) m1/m2=150/150

d1/d2	2/4	3/6	3/8	4/8
i=1	-	-	-	-
i=3	15.8 sec	5.5 sec	1.9 sec	2.2 sec
i=5	210.5 sec	82.1 sec	118.6 sec	97.7 sec

ii) m1/m2=200/200

d1/d2	2/4	3/6	3/8	4/8
i=1	-	-	-	-
i=3	44.3 sec	70.9 sec	21.9 sec	9.3 sec
i=5	342.7 sec	410.8 sec	880.0 sec	200.0 sec

以上の手続き 82 タスクについて解くことで脱出行動を生成した．表 3.4 は表 3.3 のそれぞれの学習済みネットワークを用いて解き比べたときの計算にかかった総時間を表している．

$i=1$ のネットワークで時間がかかれていないのは解が見つからなかったためであり，$i=3,5$ では全てのネットワークで答えが見つかっていることが分かる．表を縦に見ることで，i 以外の条件が同じで i だけが異なるネットワーク間の計算速度を比較することができるが，$i=3$ のネットワークでは $i=5$ のネットワークより十倍程度高速であることが見て取れる．このことは，i が TSM を用いたときの探索空間の大きさであり，i が大きくなることで探索空間の体積が指数関数的に大きくなることから計算時間の増大を招いたものと考えられる．また，二つの表の同じ要素を見ることで，ネットワークの幅 $m1, m2$ が異なるときの計算速度を比較することができるが，m1/m2=150/150 のほうが m1/m2=200/200 よりも数倍高速であることが見て取れる．このことは隣合う層が全結合するネットワークのパラメタ数がネットワーク幅の二乗に比例することから理解できる（$200^2/150^2 \approx 1.8$）．最速のパラメタは i=3, d1/d2=3/8，m1/m2=150/150 でありその計算時間は 1.9 秒であった．また計算速度上位二つのネットワークは表 3.3 の SCORE 上位二つのネットワークと同じであった．このことからネットワークの評価として用いた，ランダムな状態パラメタと百回の冗長空間サンプリングで見つかった最近傍状態パラメタの距離は，計算速度とネットワークの学習性能をともに考慮した指標として有効であることが示唆された．

図 3.5 は生成された脱出姿勢の一部である．ネットワークは 表 3.3 でもっとも SCORE の高かった i=3, d1/d2=4/8，m1/m2=150/150 を用いた．珪砂には Intel Core i7-3520M CPU 2.90GHz を用いて全体で 2.2 秒，一姿勢あたり 30 ミリ秒程度かかった．

3.2.3.2 遮蔽物裏へのリーチングタスク

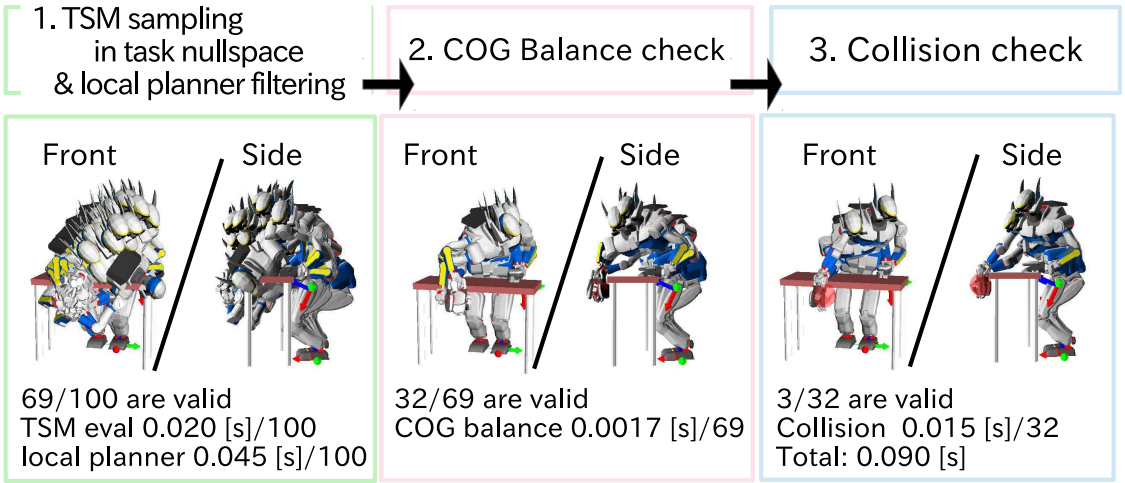


図 3.7. Sampling in task redundancy space and filtering for balancing and collision avoiding. 100 sampling took 0.090 seconds, and 3 feasible solutions are found. (reprinted from [20])

本章では TSM を用いた探索アルゴリズムの応用として複数の局所解を持つ問題を扱い，遮蔽物裏へのリーチングタスクを解く．図 3.7 は動作生成アルゴリズムの流れを示す．このアル

ゴリズムは前章の図 3.6 とほぼ同じものである．違いは，前章のタスクでは四肢の位置姿勢をすべてタスクとして扱ったが，このタスクではリーチングに用いる右手に関して位置のみをタスクとして解く．したがって，サンプリングを行う空間はタスク冗長空間と右手の姿勢タスク空間である．また前章で用いた局所探索アルゴリズムにおいても右手は位置の誤差のみを最小化することとした．図では三つの実現可能な姿勢が 90 ミリ秒で得られた．計算に用いたコンピュータは前章と同じで Intel Core i7-3520M CPU 2.90GHz，学習済みネットワークも同様に表 3.3 でもっとも SCORE の高かったものを用いている．

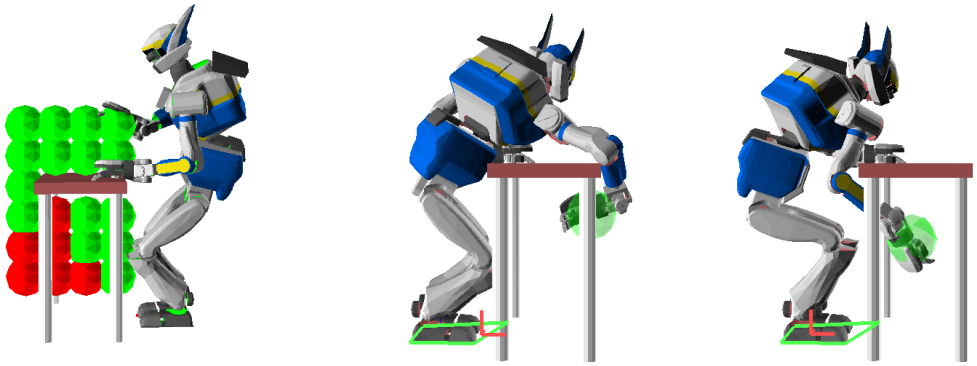


図 3.8. Reachability map of the right arm and the reaching postures are shown. Red spheres are in unreachable positions, and green spheres are in reachable positions. (reprinted from [20])

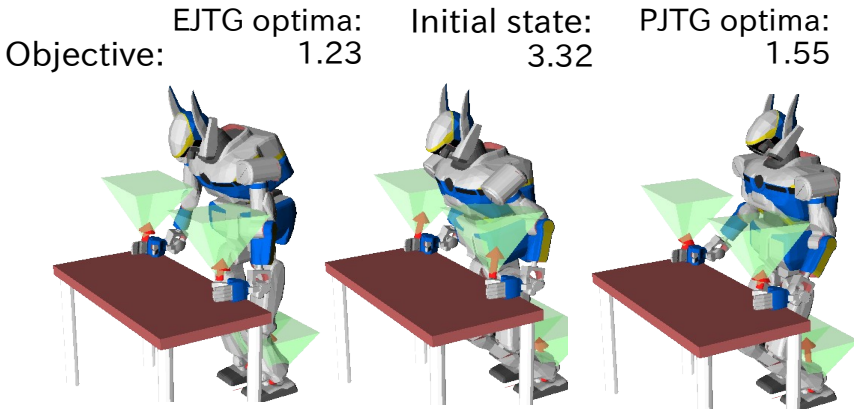
図 3.8 は生成されたリーチング姿勢を表す．中央の図は右端の図の局所解となっており逆もそうである．本アルゴリズムは初期姿勢に依存しないため，これらの解を安定して得ることができ，計算時間は平均して 50 ミリ秒程度であることが確認された．

3.2.4　学習を用いた初期値非依存な探索高速化手法についてのまとめ

本章では，TSM: Task State Map を用いてロボットのタスク冗長空間とタスク空間からロボットの状態空間への写像を学習し，タスク冗長空間での探索を行うことで高速なサンプリングが実現でき，狭い解空間しかない問題や局所解を含む問題についても高速に答えを求められることを確認した．TSM は単純に答えを蓄えておくデータベースアプローチに比べ大幅に小さなメモリ容量しか必要とせず，従来の干渉とバランスを考慮した逆運動学探索手法と比べ数倍から十倍程度高速に答えを得られることが確認された．姿勢に限らず運動軌道への拡張が今後の重要な課題である．

3.3 低計算コストな擬似関節負荷トルク勾配を用いた高速な低負荷姿勢探索法

力学を考慮しない逆運動学は，30 自由度程度のロボットで数ミリ秒で探索が可能となっている．しかし，力学を考慮する場合には関節負荷トルク（の勾配）の計算がボトルネックとなる．本章では低計算コストな関節負荷トルク勾配を提案し速度と探索性能を評価する。なお、本章のアイデアは [48] にて発表されている。



	Time [ms]	Iteration	Rate [%]
Total	12.48	100	100
Initialize	0.08	1	1
FK	1.26	100	10
ID	1.27	100	10
QP	1.46	100	11
EJTG	5.45	100	44
J#	2.62	100	21
else	0.34	-	3

	Time [ms]	Iteration	Rate [%]
Total	7.60	100	100
Initialize	0.08	1	1
FK	1.44	100	19
ID	1.34	100	18
QP	1.60	100	21
PJTG#	1.05	100	14
J#	1.68	100	22
else	0.41	-	5

図 3.9. The Exact Joint Torque Gradient (EJTG) is a bottleneck in the posture optimization process. In contrast, the Pseudo Joint Torque Gradient (PJTG) is not a bottleneck and accelerates the optimization process.

ロボットの動作生成では，軌道全体の探索も重要な問題であり多くの先行研 [69][16] [41] があるが，姿勢の探索はその基礎となる問題であり，軌道の中継点として用いられたり [31][48][35] と実用上も重要な問題である．本章では負荷最小姿勢の探索を計算オーダーから高速化する手法を提案する．ロボットの姿勢探索, 特に脚立登り [48][70] や崖登りなどといった高い負荷が問題となるような探索問題では, 接触リンクの位置姿勢や転倒回避等の制約を満たすことに加え関節負荷を実現可能な範囲に収める工夫が必要不可欠である．過去の研究 [71] では, 関節負荷トルクの勾配を用いて勾配法の要領で姿勢を探索する方法が提案させている．しかし, 関節負荷トルク勾配の計算量に関する考察はあまりなされて来なかった．関節負荷ト

ルク勾配（EJTG: Exact Joint Torque Gradient）を用いた方法には計算コストの問題がある．関節負荷トルクは全関節の自由度 N と同じだけの自由度を持つため, 勾配行列の大きさは N^2 となる. したがって, 最小でも $\mathcal{O}(N^2)$ の時間計算量が必要である．3.3.3.1.2 章では実際に $\mathcal{O}(N^2)$ の手続きが存在することを示す．EJTG の計算量が $\mathcal{O}(N^2)$ であることは，これは逆運動学におけるヤコビ行列の時間計算量が $\mathcal{O}(N)$ であることと比べて一桁大きく, 計算量のボトルネックとなりうるという問題を孕んでいる．

以上の問題を受けて，本研究では, 計算コストの問題について時間計算量 $\mathcal{O}(N)$ の擬似関節負荷トルク勾配（PJTG: Pseudo Joint Torque Gradient）を提案する．PJTG の特色は，関節負荷トルクを仮想的に接触反力, 反モーメントに近似しその勾配を関節負荷トルク勾配の近似として用いる方法である．PJTG は発見的に定義された手法であるためその適用範囲や探索性能に疑問が残るが，本章では関節負荷トルク勾配を用いるアルゴリズムとの比較実験を行い, 計算時間と解の最適性, 求解性の評価を行った．本章の目的は姿勢探索を高速化する手法を提案し，その性能について評価を行うことである．

姿勢探索アルゴリズムとしては，満たすべき制約を幾何的なものと力学的なものに分割し, それぞれを信頼性の高い探索器で繰り返し解く方法（IK-QP 繰り返し法）を紹介する．接触リンクの位置姿勢や干渉回避を含む探索問題は逆運動学 (IK) としてよく知られた問題であり, 多くの理論と信頼性の高い実装が存在する．また, 凸な線形不等式制約つきの二次表関数を持つ探索問題は二次計画問題 (QP) と呼ばれ, 高速安定な求解が可能である.IK と QP, 二つの安定な探索器を交互に解くこと探索の信頼性を高めることを意図した方法である．また, 脚立登り動作の探索例を通して探索性能の評価を行う．IK-QP 繰り返し法は [48] でも簡単な発表を行っている内容である．

3.3.1　接触のモデルと実現可能なロボットの状態

本章で扱う接触のモデルと, 実現可能なロボットの状態について記述する．

3.3.1.1　接触状態のモデルと滑り・転がりに関する不等式拘束

接触モデルを表現するために，[72][34][73] では複数の接触点を接触ごとに定義し，正の反力が摩擦の条件を満たしつつ発揮されるという条件を用いている．対して，[74][75][48] では接触力・モーメントを考え接触面長さから求められるモーメントの条件を用いて表現する．後者の方法は [76] で見られるような圧力中心が接触面内にあるというモデルと等価である．本章は後者を採用するが，アルゴリズムとして接触力の条件が線形であるという性質しか用いないため前者での代用も可能である．接触を保つための摩擦や回転を防ぐために, 六次元の接触反力 $\boldsymbol{F}$ が満たすべき条件を $\mathcal{F}$(式 3.4) で表現する．

$$-\left(\boldsymbol{M}_i^{-}\boldsymbol{F}_i+\boldsymbol{F}_i^0\right) \leq \boldsymbol{F}_i \leq \left(\boldsymbol{M}_i^{+}\boldsymbol{F}_i+\boldsymbol{F}_i^0\right) \tag{3.4}$$

$$\boldsymbol{M}_i^{\pm}=\begin{pmatrix} 0 & 0 & \mu_i^x & 0 & 0 & 0 \\ 0 & 0 & \mu_i^y & 0 & 0 & 0 \\ 0 & 0 & b_{\pm} & 0 & 0 & 0 \\ 0 & 0 & l_i^y & 0 & 0 & 0 \\ 0 & 0 & l_i^x & 0 & 0 & 0 \\ 0 & 0 & \mu_i^z & 0 & 0 & 0 \end{pmatrix}, \boldsymbol{F}_i^0=\begin{pmatrix} f_i^{x0} \\ f_i^{y0} \\ f_i^{z0} \\ n_i^{x0} \\ n_i^{y0} \\ n_i^{z0} \end{pmatrix}$$

式 3.4 は接触状態の満たすべき条件を接触面のローカル座標系で表現したもので $+z$ 方向を接触面の法線方向とする．F_i は 6 自由度の接触反力, $M_i{}^{\pm}$ は摩擦係数 (μ_x, μ_y) と接触面形状 (l_x, l_y) からなる行列, $\boldsymbol{F_{i0}}$ は面への垂直反力に依存しないロボットが発揮できる力であり把持力の表現等に用いる．$b_{\pm}$ は unilateral 拘束と bilateral 拘束を区別するための項であり, 面との一方向からの接触を表す際には $b_{-}=0, b_{+}=\inf$ になる．図 3.10 に式 3.4 で扱うことのできる接触例を示す．順番に点接触, 線接触, 面接触, 握り接触について述べる．

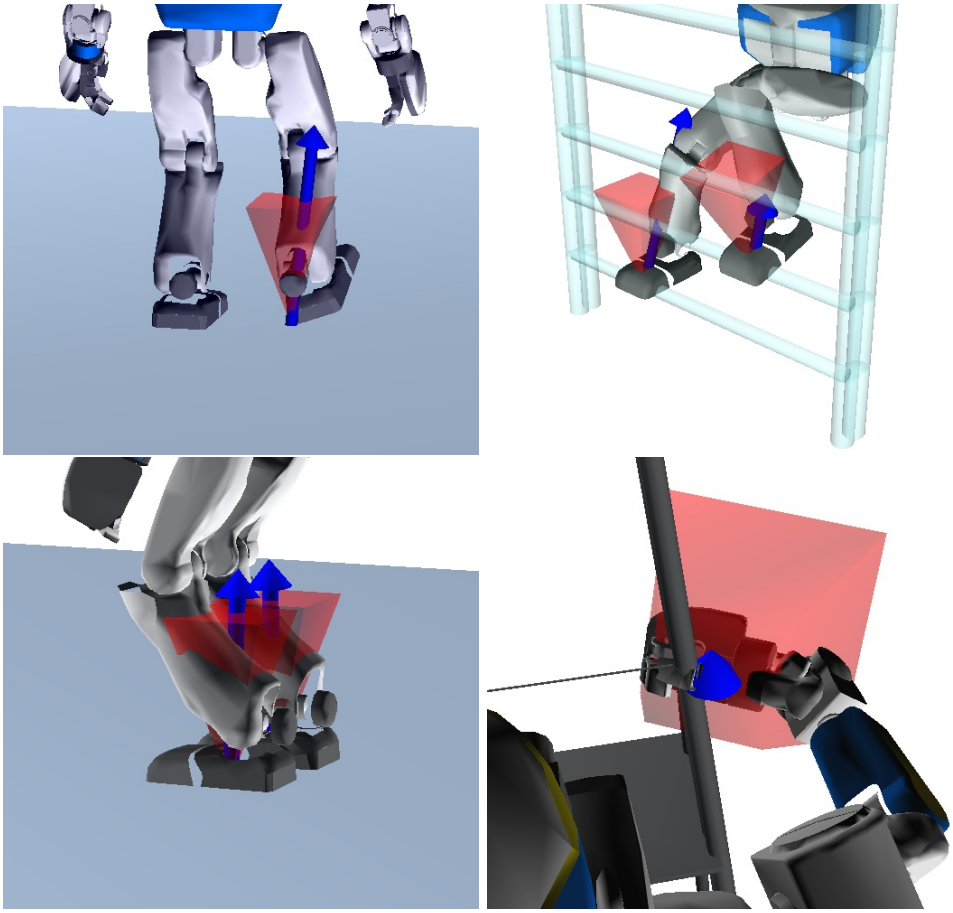


図 3.10. Examples that illustrate (clockwise from upper-left) point, line, plane, and grasp contacts.

3.3.1.1.1　点接触

図 3.10 左上に点接触の例を示す．点接触では, 式 3.4 にて $l_x = l_y = 0$ であり, 接触点まわりに回転モーメントを発揮できない．また幾何的な拘束としては接触点周りにフリー. すなわち位置の拘束あり, 姿勢の拘束なしである．

3.3.1.1.2　線接触

図 3.10 右上に線接触の例を示す．ここで，接触のローカル座標系において $+x$ が接触リンクのつま先方向を洗わし y 軸で線接触が起こっているとする．式 3.4 にて $l_x = 0$ であり, 接触線まわりに回転モーメントを発揮できない．また幾何的な拘束としては接触面周りにフリー. すなわち位置の拘束あり, 姿勢の拘束は x 軸周りにのみ存在する．

3.3.1.1.3　四角面接触

図 3.10 左下に面接触の例を示す．位置姿勢の六自由度が拘束され, 接触面での回転モーメントも三自由度を発揮できる．

3.3.1.1.4　握り接触

図 3.10 右下に握り接触の例をしめす．面接触と同様, 六自由度が拘束される．さらに接触面法線方向の力に依存しない力 ($\boldsymbol{F_0}$) を発揮できる．

3.3.1.1.5　凸面接触

凸な接触面であれば $M^{\pm}$ を変形させた $M^{\pm *}$ についても Eq.3.4 を追加で満たすことで表現ができる．非凸の場合は本章では扱わない．

3.3.1.2　実現可能なロボットの状態

実現可能とは, リンク系が接触拘束を満たすこと, リンク系と環境との干渉が起こらないこと, 運動方程式を満たすこと, 関節負荷トルクがモータの関節負荷トルク制約を満たすこと, 接触反力が滑りや回転を起こさない (起こす) こととする．以下にそれぞれについて述べる．

3.3.1.2.1　環境との接触を保つ

リンク系の運動学については [77] [78] が詳しい．また 4.3.1 章に数式を用いた解説を行った．リンク系はリンク間のジョイントのみが駆動しすべて剛体であるとする．各ジョイントの回転角度といったジョイントパラメタ $\boldsymbol{q}$ と接触リンクの位置姿勢 $\boldsymbol{x}$ の関係 ($Kinematics$) について以下が成り立つとする．

$$\boldsymbol{x} = Kinematics(\boldsymbol{q}) \tag{3.5}$$

この時, すべての接触リンク位置姿勢 $\boldsymbol{x}_i$ が, その目標位置姿勢 $\boldsymbol{x}_i^d$ に誤差 ϵ_i で一致する条件

を満たすこととする．

$$\forall i; \quad \|\boldsymbol{x}_i - \boldsymbol{x}_i^d\| \leq \epsilon_i \tag{3.6}$$

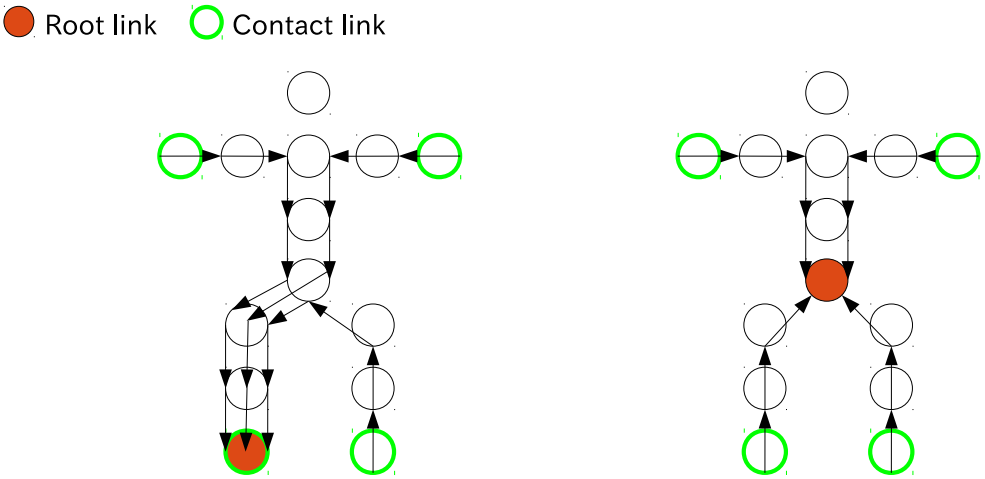
また, ジョイントパラメタとリンク位置姿勢の一回微分を関係付ける行列 J はヤコビ行列と呼ばれ，以下のように定義される．

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{3.7}$$

ヤコビ行列は逆運動学問題だけでなく，逆動力学問題（式 3.9)）や順動力学問題（4.3.1 章）でも用いられる．本章では勾配を用いた繰り返し計算によって逆運動学を解く．関節角度 $\boldsymbol{q}$ を以下の計算によって更新する．

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{x}} + (\boldsymbol{E} - \boldsymbol{J}^{\#}\boldsymbol{J})\boldsymbol{z} \tag{3.8}$$

ここで，$\boldsymbol{E}$ は三次元の単位行列，$\boldsymbol{z}$ は任意のベクトル，$\boldsymbol{J}^{\#}$ はヤコビ行列 $\boldsymbol{J}$ の擬似逆行列である．$\boldsymbol{z}$ は重心位置を動かしたり関節負荷トルクを減らしたりする項として用いることで $\boldsymbol{x}$ 以外の複数タスクを考慮することができる．しかし本章では，$\boldsymbol{z}$ として常に $\boldsymbol{0}$ を用い，関節負荷トルクを減らすための複数タスクを解く際には重み付き SR-Inverse [79] を用いている．具体的な式については続く 3.3.2.2 章で述べる．



(a) Root link = right foot　　(b) Root link = pelvis link

図 3.11. Kinematic chains of the robot, with each arrow connecting a child link to its parent link.

また，多リンク系であるロボットのリンクチェーンの定義としては図 3.11(a) を用いた．本章では接触を保つリンクが存在しない動作（例えばジャンプ）を扱わないため，接触していて動かないリンクをルートリンクにすることで制約を減らす工夫を行っている．

3.3.1.2.2　運動方程式を満たす

リンク系各ジョイントの発揮する関節負荷トルク $\boldsymbol{\tau}$ は k 個の接触反力 $\boldsymbol{F}_i$ と, 重力や慣性力といったその他 $l-k$ 個の力 $\boldsymbol{w}_i$, および力の加わる位置姿勢についてのヤコビ行列 J_i を用いて以下のように書ける．また，トルクは接触力による項とそれ以外（dynamics 関数）に分けることができることが知られている．より具体的な式の計算については 4.3.1 章で補足を行っている．

$$\boldsymbol{\tau} = \sum\nolimits_{i\in[0,l)} \boldsymbol{J}_i^T \boldsymbol{w}_i \tag{3.9}$$

$$= Dynamics^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) + \sum\nolimits_{i\in[0,k)} \boldsymbol{J}_i^T \boldsymbol{F}_i \tag{3.10}$$

$$\forall_{i\in[0,k)};\quad \boldsymbol{F}_i = \boldsymbol{w}_i$$

本章では, 静的な姿勢の探索を扱うので, この条件を各接触点の位置 x_i, y_i, z_i と接触反力 $\boldsymbol{F}$ を用いて定義される静的釣り合いの式で代用する（式 3.11）．

$$\sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\boldsymbol{F}_i + \boldsymbol{G}(\boldsymbol{c})\ [m\boldsymbol{g}^T,\ \boldsymbol{0}^T]^T = 0 \tag{3.11}$$

$$\boldsymbol{G}(\boldsymbol{x}_i) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -z_i & y_i & 1 & 0 & 0 \\ z_i & 0 & -x_i & 0 & 1 & 0 \\ -y_i & x_i & 0 & 0 & 0 & 1 \end{pmatrix}, \boldsymbol{x}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$$

ここで x_i, y_i，z_i は i 番目の接触リンクの三次元位置を表す．$\boldsymbol{F}_i$ はその点に加わっている力とモーメントであり，m はロボットの全質量，$\boldsymbol{g}$ は重力加速度である．式 3.11 が満たされれば式 3.10 も満たされる．これは図 3.11 でどちらのキネマティクスチェーンを用いても同様である．

3.3.1.2.3　関節負荷トルクがモータの関節負荷トルク制限を満たすこと

関節負荷トルクの制約について，以下の不等式を考慮する．

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \tag{3.12}$$

3.3.1.2.4　接触力が滑りや転がりを起こさないこと

全接触力をまとめたベクトル $\boldsymbol{F} = [\boldsymbol{F}_0^T \cdots \boldsymbol{F}_{k-1}^T]^T$. が式 3.4 を満たすよう計算する．

3.3.2　問題設定: 姿勢最適化問題の定式化とアルゴリズム

3.3.2.1　負荷指標と姿勢最適化

姿勢最適化の評価関数として, 以下にしめすベクトル（BRLV: Body Retention Load Vector [48]）を用いる．BRLV は全関節負荷トルクと接触力・モーメントを含み，それぞれの要素は

最大値で正規化される．

$$
{}^{\boldsymbol{BRL}}\boldsymbol{v}_i = \begin{bmatrix} \bar{\boldsymbol{\tau}}_i{}^T & \bar{\boldsymbol{F}}_i{}^T \end{bmatrix}^T \tag{3.13}
$$

$$
where, \begin{cases} \bar{\boldsymbol{\tau}}_i : & = & (\cdots \frac{\tau_{ij}}{\tau_{ij}^{max}} \cdots)^T \\ \bar{\boldsymbol{F}}_i : & = & (\cdots \frac{F_{ij}}{F_{ij}^{max}} \cdots)^T \end{cases}
$$

BRLV は図 3.11(b) で示すキネマティスチェーンを用いて各接触リンクとそこからルートリンクへと繋がるリンク系について定義する．i は各接触を区別し，τ_{ij} の j は関節，F_{ij} の j は接触力・モーメントの各要素を表す．DC モータで駆動される軸関節ではの最大負荷トルクは式 3.12 に示すように身体モデルパラメタ生成段階で決まる定数である．接触力・モーメントの最大値は関節の負荷を考えなければ摩擦によって決定される．本章では式 3.4 の右辺をヒューリスティックに決定した定数力 $\boldsymbol{F}_i$ を用いて計算した値を BRLV における最大値として扱う．定数力を用いる理由は，BRLV を力に対して線形化することで計算を高速化するためである．$\boldsymbol{F}_i^{max} = \boldsymbol{M}_i^{\pm}\boldsymbol{F}_i^{est} + \boldsymbol{F}_i^0$ とし手足の出力差を考慮し，腕の接触については $\boldsymbol{F}_i^{est} = [0\ 0\ 50\ 0\ 0\ 0]^T$ 脚については $\boldsymbol{F}_i^{est} = [0\ 0\ 500\ 0\ 0\ 0]^T$ を用いた．BRLV の大きさを最小化することでトルクと力の不等式制約からの余裕度を最大化することを意図している．

${}^{\boldsymbol{BRL}}\boldsymbol{v}_i$ の二乗和を最小化しつつ 3.3.1 章にまとめた条件を満たす姿勢を探索する問題を式 3.14 にまとめなおす．ここで探索空間は全関節の角度 $\boldsymbol{q}$ に加えて角接触の接触力・モーメント $\boldsymbol{F} = [\boldsymbol{F}_0 \cdots \boldsymbol{F}_{k-1}]$ である．

$$
\begin{aligned}
min_{\boldsymbol{q},\boldsymbol{F}} : & \sum\nolimits_i {}^{\boldsymbol{BRL}}\boldsymbol{v}_i^T {}^{\boldsymbol{BRL}}\boldsymbol{v}_i \\
s.t.\ \ & \forall i;\ \ \boldsymbol{x}_i = Kinematics(\boldsymbol{q}) \\
& \forall i;\ \ \|\boldsymbol{x}_i - \boldsymbol{x}_i^d\| \leq \epsilon_i \\
& \boldsymbol{\tau} = Dynamics^{-1}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}) + \sum\nolimits_i \boldsymbol{J}_i^T \boldsymbol{F}_i \\
& \forall i; -\left(\boldsymbol{M}_i^{-}\boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \leq \boldsymbol{F}_i \leq \left(\boldsymbol{M}_i^{+}\boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \\
& \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \\
& \sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\boldsymbol{F}_i + \boldsymbol{G}(\boldsymbol{c})\ [m\boldsymbol{g}^T,\ \boldsymbol{0}^T]^T = 0
\end{aligned} \tag{3.14}
$$

式 3.14 を解くために，IK-QP 繰り返し法は式 3.14 の制約を逆運動学で扱うことのできる幾何的制約と二次計画問題で扱うことのできる力学的制約の二種類に分割する．式 3.15 に式 3.14 を分割したものを示す．

$$
\begin{aligned}
min_{\boldsymbol{q}} : & \sum\nolimits_i {}^{\boldsymbol{BRL}}\boldsymbol{v}_i^T {}^{\boldsymbol{BRL}}\boldsymbol{v}_i \\
s.t.\ \ & \forall i;\ \ \boldsymbol{x}_i = Kinematics(\boldsymbol{q}) \\
& \forall i;\ \ \|\boldsymbol{x}_i - \boldsymbol{x}_i^d\| \leq \epsilon_i \\
& [\boldsymbol{\tau},\ \boldsymbol{F}] = QP(\boldsymbol{q})
\end{aligned} \tag{3.15}
$$

ここで関数 QP は以下のように定義する．二次計画問題により固定された姿勢で力学的な制約を考慮する同様の定式化は [48], [80], [75] にも見られる．二次計画問題のソルバとしては

[25] に実装された Goldfarb-Idnani 法 [24] を用いた.

$$
\begin{aligned}
QP(\boldsymbol{q}) = \arg\min_{\boldsymbol{\tau},\boldsymbol{F}} \quad : \quad & \sum\nolimits_i \boldsymbol{BRL}\boldsymbol{v}_i^T \boldsymbol{BRL}\boldsymbol{v}_i \\
s.t. \quad & \boldsymbol{\tau} = Dynamics^{-1}(\boldsymbol{q}, \boldsymbol{0}, \boldsymbol{0}) + \sum\nolimits_i \boldsymbol{J}_i^T \boldsymbol{F}_i \\
& \forall i; -\left(\boldsymbol{M}_i^- \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \leq \boldsymbol{F}_i \leq \left(\boldsymbol{M}_i^+ \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \\
& \boldsymbol{\tau}_{min} \leq \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \\
& \sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\boldsymbol{F}_i + \boldsymbol{G}(\boldsymbol{c})\ [m\boldsymbol{g}^T,\ \boldsymbol{0}^T]^T = 0
\end{aligned}
\tag{3.16}
$$

この分割のメリットは，逆運動学（以下 IK）と二次計画問題（以下 QP）は頻繁に扱われる問題であり，多くの安定なソルバが実装されていることにある．一般に，非線形最適化問題を逐次的に二次計画問題に近似することで繰り返し法を用いて解く手法は SQP （Sequential Quadratic Programming）として知られている [81]．IK-QP 繰り返し法と SQP の違いは，IK-QP 繰り返し法は一部の制約のみを線形として扱い逆運動学部分は非線形のまま解く点にある．QP 関数では，関節負荷トルク $\boldsymbol{\tau}$ と接触力・モーメント $\boldsymbol{F}$ の制約を考慮しているが，解のない問題にたいしてこの関数は失敗する可能性がある．例えば，ロボットの腕が自重を支えるのに十分強くない場合に逆立ちのような姿勢でこの問題を解いても関節負荷トルクの制約を満たす解を見つけることはできない．式 3.15 では評価関数に BRLV を用いることで不等式を soft constraints としても考慮しており，二次計画問題が解けない場合は不等式条件を用いずに評価関数と静的釣り合いの式のみを考慮した問題を解くことで値を更新した.

3.3.2.2　逆運動学と接触力最適化の繰り返し解法

本章では，3.15 を解くための IK-QP 繰り返し法について述べる．アルゴリズム 1 を示す．アルゴリズム 1 の各関数の解説を以下に示す.

3.3.2.2.1　“solveQP”

式 3.16 を解く.

3.3.2.2.2　“calcTorqueGradient”

この関数は関節負荷トルク勾配を計算し，関節負荷を減らすための関節速度を計算するのに用いる．具体的な計算手続きについては続く 3.3.3 章で述べる.

3.3.2.2.3　“calcForceGradient”

この関数は接触力・モーメントの勾配を計算し，それらを減少させるための重心速度を計算する．式 3.14 では，接触力・モーメントは探索空間に含まれるためその勾配は存在しないが，式 3.15 では力とモーメントが二次計画問題 3.16 の解として姿勢から一意に定まるため姿勢に対する勾配が存在する．この力とモーメントの勾配は厳密には関数 QP を微分して得られるものであるが，二次計画問題の解の微分を数値微分により求めるのは計算コストが高く，ま

Algorithm 1 IK-QP iterative method

Require: $\boldsymbol{q}$, $\boldsymbol{x}, \mathcal{F}, \mathcal{T}$

$\boldsymbol{q}$: Initial posture. $\boldsymbol{x}$: Target contact coordinates.

$\mathcal{F}$: Contact force constraints. $\mathcal{T}$: Joint torque limitations.

Variable: .

$\boldsymbol{F}$: Contact force. $\boldsymbol{\tau}$: Joint torque.

$\boldsymbol{F}_b$: The best $\boldsymbol{F}$. $\boldsymbol{\tau}_b$: The best $\boldsymbol{\tau}$. $\boldsymbol{q}_b$: The best $\boldsymbol{q}$.

$J_f, \boldsymbol{v}_f$: Contact force gradient matrix and desired velocity.

$J_\tau, \boldsymbol{v}_\tau$: Joint torque gradient matrix and desired velocity.

Procedure: IKQP

1: **repeat**
2: $[\boldsymbol{\tau}, \boldsymbol{F}] \leftarrow$ solveQP($\boldsymbol{q}$, $\mathcal{F}$)
3: $[J_f, \boldsymbol{v}_f] \leftarrow$ calcForceGradient($\boldsymbol{q}$, $\boldsymbol{F}$, $\mathcal{F}$)
4: $[J_\tau, \boldsymbol{v}_\tau] \leftarrow$ calcTorqueGradient($\boldsymbol{q}$, $\boldsymbol{F}$, $\boldsymbol{\tau}$, $\mathcal{T}$)
5: $\boldsymbol{q} \leftarrow$ inverseKinematics($\boldsymbol{x}$, $J_f, \boldsymbol{v}_f$, $J_\tau, \boldsymbol{v}_\tau$)
6: $[\boldsymbol{q}_b, \boldsymbol{\tau}_b, \boldsymbol{F}_b] \leftarrow$ updateBest($\boldsymbol{q}_b, \boldsymbol{\tau}_b, \boldsymbol{F}_b, \boldsymbol{q}, \boldsymbol{\tau}, \boldsymbol{F}$)
7: **until** loopLimitExceeded
8: **return** $[\boldsymbol{q}_b, \boldsymbol{\tau}_b, \boldsymbol{F}_b]$

た解析解も求められない．微分可能でない点がありうるなどの問題がある．本章では，この力とモーメントの勾配を求めるために式 3.11 を用いる．

$$\sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\boldsymbol{F}_i + \begin{pmatrix} m\boldsymbol{g} \\ \boldsymbol{c} \times m\boldsymbol{g} \end{pmatrix} = 0$$

$$\Rightarrow \sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\ \Delta\boldsymbol{F}_i - \begin{pmatrix} \boldsymbol{0} \\ m\boldsymbol{g}\times \end{pmatrix} \Delta\boldsymbol{c} = 0 \tag{3.17}$$

以上から，接触力・モーメントの勾配を以下のように定義する．

$$\Delta\boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ m\boldsymbol{g}\times \end{pmatrix}^{\#} \sum\nolimits_i \boldsymbol{G}(\boldsymbol{x}_i)\ \Delta\boldsymbol{F}_i \tag{3.18}$$

$$= \sum\nolimits_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i}\ \Delta\boldsymbol{F}_i \tag{3.19}$$

ここで， $\Delta\boldsymbol{F}_i$ は ${}^{\boldsymbol{BRL}}\boldsymbol{v}_i$ の力とモーメントに関する項であり．アルゴリズム 1 では，$\boldsymbol{v}_f$ は $\Delta\boldsymbol{c}$， $\boldsymbol{J}_f$ は重心ヤコビ行列．式 3.18 右辺の一番目の係数は 3×6 行列である．行列 $[\boldsymbol{0}; m\boldsymbol{g}\times]$

の $0,1,2,5$ 行は全要素が零であるので，$3,4$ 行について解くことで以下を得る．

$$\begin{pmatrix} \mathbf{0} \\ m\boldsymbol{g}\times \end{pmatrix}^{\#} = \begin{pmatrix} 0 & 0 & 0 & -1/mg & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/mg & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.20}$$

$$\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} = \frac{1}{mg} \begin{pmatrix} -z_i & 0 & x_i & 0 & -1 & 0 \\ 0 & -z_i & y_i & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.21}$$

このような接触力・モーメント勾配は擬似逆行列のとり方によって無数に定義することができ，上式は唯一の解ではないことに注意が必要である．上式がどのような意味を持つのかについて見てみる．まず，z_i 項は重心一を $\Delta \boldsymbol{F}_i$ の $0,1$ 番めの要素と反対方向へと動かす項である．このことは重心が力を受けている方向（押されている方向）に動くことで力を逃すような動きをすることを意味する．次に x_i, y_i の項は垂直反力に応じて接触位置から離れる向きに動く．これは重心が大きな力を受けている接触点から遠ざかることで他の接触に反力を分配する挙動を表している．最後に ± 1 の項を見てみると，接触モーメントを減らす向きに重心を動かす項であることが分かる．

3.3.2.2.4　“inverseKinematics”

この関数は逆運動学問題を解くことで接触リンクの位置姿勢を目標値 $\boldsymbol{x}$ に移動させる計算を行う．本章では，[79] に良く似た勾配法に基づく繰り返し最適化計算を実装した．これは，lisp の方言である euslisp [43][44] に実装された逆運動学ソルバの C++ 言語を用いた再実装である．euslisp での実装には [18] の衝突回避計算も実装されており，本章ではトルクの問題にフォーカスするため自己干渉まで扱わないが，本章で用いる手法は逆運動学計算において自己干渉を考慮することも可能であると考えている．逆運動学関数 $inverseKinematics$ の $\boldsymbol{x}$ 以外の引数は接触力と関節負荷トルクを減少させるために用いる．

$$\begin{aligned} \boldsymbol{q} + \Delta \boldsymbol{q} &= \boldsymbol{q} + \boldsymbol{J}_{all}^{\#} \boldsymbol{v}_{all} \\ \boldsymbol{J}_{all} = \begin{pmatrix} \boldsymbol{J} \\ \boldsymbol{J}_f \\ \boldsymbol{J}_\tau \end{pmatrix},\ \boldsymbol{v}_{all} &= \begin{pmatrix} \boldsymbol{x}^d - \boldsymbol{x} \\ \boldsymbol{v}_f \\ \boldsymbol{v}_\tau \end{pmatrix} \end{aligned} \tag{3.22}$$

接触位置姿勢を保つという拘束 $\boldsymbol{x}^d - \boldsymbol{x}$ は等式制約であるが，$\boldsymbol{v}_F$, $\boldsymbol{v}_\tau$ を減少させる拘束は不等式制約であるため区別が必要である．以下では重み付き擬似逆行列を用いて区別する．

$$\begin{aligned} \boldsymbol{J}_{all}^{\#} &= \left(\boldsymbol{J}_{all}^T \boldsymbol{W} \boldsymbol{J}_{all}\right)^{-1} \boldsymbol{J}_{all}^T \boldsymbol{W} \\ \boldsymbol{W} &= diag(\boldsymbol{E}, w_f \boldsymbol{E}, w_\tau \boldsymbol{E}) \end{aligned} \tag{3.23}$$

$\boldsymbol{J}_{all}^T \boldsymbol{W} \boldsymbol{J}_{all}$ の逆行列計算では SR Inverse [79] を用いた．ここで， $\boldsymbol{E}$ は単位行列であり幾何制約 $\boldsymbol{x}^d - \boldsymbol{x}$ の重みに用いる．さらに，w_f, w_τ は力とトルクに関する制約の主にである．以下の実験では，定数 0.01 を両方のゲインに用いている．この値は以下の実験で用いたのと別のテストケースについてマニュアルに調節した値である．PJTG のについては，EJTG と違

い行の次元が列の次元よりも小さいため擬似逆行列の定義が異なる．

$$\Delta \boldsymbol{q} = \boldsymbol{J}_{all}^T \left(\boldsymbol{J}_{all} \boldsymbol{J}_{all}^T \right)^{-1} \boldsymbol{W} \boldsymbol{v}_{all} \tag{3.24}$$

重みの大きさは同様に以下の実験で用いたのと別のテストケースについてマニュアルに調節を行い 5 とした．

3.3.2.2.5 "updateBest"

この関数は全ての制約が満たされかつ評価関数が最善である状態 $\boldsymbol{q}_b, \boldsymbol{\tau}_b, \boldsymbol{F}_b$ を蓄えておき，もしより最善な状態が見つかったらそれらを更新する手続きである．

3.3.3 関節負荷トルク勾配の定義と計算量

この章では, まず EJTG について説明を行う．これはアルゴリズム 1 の $calcTorqueGradient$ 関数の定義である．以下ではまず，EJTG の定義と計算量について 3.3.3.1 章で説明を行う．次に，高速計算可能な PJTG について 3.3.3.2 章で説明を行う．最後に，3.3.3.3，3.3.3.4 章で計算時間の比較と零点の比較を行う．

3.3.3.1 厳密な関節負荷トルク勾配: 定義と計算量

3.3.3.1.1 EJTG の定義

静的な問題では，全ての力とモーメントは接触によるものと重力によるものに分けられる．関節負荷トルク $\boldsymbol{\tau}$ は 式 3.9 から以下のように計算できる．

$$\boldsymbol{\tau} = \sum\nolimits_{i \in [0,N)} {}^g\boldsymbol{J}_i^T \begin{pmatrix} m_i \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} + \sum\nolimits_{i \in [0,k)} \boldsymbol{J}_i^T \boldsymbol{F}_i \tag{3.25}$$

ここで，${}^g\boldsymbol{J}_i^T$ は i 番めのリンクの重力が加わる位置（重心）のヤコビ行列，m_i は i 番めのリンクの質量，$\boldsymbol{g}$ は重力加速度ベクトルである．重力による関節負荷トルク項を以下のように変形しておく．

$$\boldsymbol{\tau}_{\boldsymbol{g}} = \left(\sum_{i=0}^{N-1} m_i {}^g\boldsymbol{J}_i^T \right) \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \tag{3.26}$$

これらを微分して EJTG の定義を得る．

$$\begin{aligned} \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}} = & \sum\nolimits_{i \in [0,N)} \frac{\partial\, {}^g\boldsymbol{J}_i^T}{\partial \boldsymbol{q}} \begin{pmatrix} m_i \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \\ & + \sum\nolimits_{i \in [0,k)} \frac{\partial \boldsymbol{J}_i^T}{\partial \boldsymbol{q}} \boldsymbol{F}_i \end{aligned} \tag{3.27}$$

さらに重力項の EJTG を以下のように変形しておく．

$$\frac{\partial \boldsymbol{\tau}_{\boldsymbol{g}}}{\partial \boldsymbol{q}} = \left(\sum\nolimits_{i \in [0,N)} m_i \frac{\partial\, {}^g\boldsymbol{J}_i^T}{\partial \boldsymbol{q}} \right) \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \tag{3.28}$$

[82], [71]，[34] にもこれら関節負荷トルク勾配についての解説がある．3.3.6.1 章でより具体的な計算式を示す．続く 3.3.3.1.2 章では計算量をみつもる．

3.3.3.1.2　EJTG の計算量

式 3.28 を式のまま計算すると $N \times N \times 6$ のテンソルである $\partial\, {}^g\boldsymbol{J}_i^T/\partial \boldsymbol{q}$ を N 回計算するのでその計算量は $\mathcal{O}(N^3)$ となってしまう．しかし，[83] で用いられるような重心ヤコビアンを $\mathcal{O}(N)$ で計算するアルゴリズムを拡張することで，EJTG の計算量を $\mathcal{O}(N^2)$ に減らすことが可能である．このとき，接触数 k が関節の自由度 N よりも十分小さければ，接触力も考慮した全体の EJTG 式 3.27 の計算量もまた $\mathcal{O}(N^2)$ となる．この計算手続きについて以下に示す．式 3.28 にて重力ベクトルのモーメント項は全て零であるので，ヤコビ行列についても位置の項のみを考えれば十分である．各リンクの重心位置の関節角度による微分は以下のようになる．$(q_{k_j}, j \in [0, l), o \prec k_0 \prec \cdots \prec k_{l-1} \prec i)$

$$\frac{\partial^l \boldsymbol{c}_i^o}{\partial q_{k_0} \cdots \partial q_{k_{l-1}}} = \boldsymbol{a}_{k_0}^o \times \cdots \boldsymbol{a}_{k_{l-1}}^o \times (\boldsymbol{c}_i^o - \boldsymbol{p}_{k_{l-1}}^o) \tag{3.29}$$

ここで，$\boldsymbol{c}_i^o$ は i 番めの関節の o 番めの関節からみた重心位置，$\boldsymbol{p}_i^o$ は i 番めの関節の o 番めの関節からみた位置，$\boldsymbol{a}_i^o$ は i 番めの関節の o 番めの関節からみた回転軸ベクトルである．さらに，$i \prec j$ は i 番めの関節が j 番めの関節の親のひとつ（ルートリンクに近いほうが親とする）であることを表す．この式 (3.29 の導出は 3.3.6.1 章でも補足する．

以上から，3.28 式の右辺は以下のように書き直せる．

$$\begin{aligned} &\sum_{i=k_{l-1}}^{N-1} m_i \frac{\partial^l \boldsymbol{c}_i^o}{\partial q_{k_0} \cdots \partial q_{k_{l-1}}} \\ &= \boldsymbol{a}_{k_0}^o \times \cdots \boldsymbol{a}_{k_{l-1}}^o \times \Big(\sum_{i=k_{l-1}}^{N-1} m_i \boldsymbol{c}_i^o - \sum_{i=k_{l-1}}^{N-1} m_i \boldsymbol{p}_{k_{l-1}}^o \Big) \end{aligned} \tag{3.30}$$

式 3.30 は再帰的に ${}^*\boldsymbol{c}_i^o = \sum_{i=0}^{N-1} m_i \boldsymbol{c}_i^o$ と ${}^*m_i = \sum_{i=k_{l-1}}^{N-1} m_i$ をあらかじめ計算しておくことで，$\mathcal{O}(N)$ で計算することが可能である．トルク勾配の計算 3.28 では，式 3.30 を $l = 2$ として計算すればよく，それには二つの外積の計算を $k_0, k_1 \in [0, N)$ として計算すればよい．全身関節ロボットでは $N \approx 30$ 程度であるため，計算オーダーだけでなく係数も重要になる．以下ではより詳細に計算量を見積もっていく．

図 3.12 に EJTG の重力に関する項を計算するために必要な掛け算の数を見積もる．この行列 $\frac{\partial \boldsymbol{\tau_g}}{\partial \boldsymbol{q}}$ は対称行列となるため下半分の要素を計算するのに必要な掛け算の数を数える．式 3.30 に示すように，それぞれの項の計算には二回の外積の掛け算と二回のベクトルの内積を計算する必要がある．したがって一要素あたり 18 回の掛け算が必要で，これを $N \times N$ 行列の下半分全てについて行うので，全体の計算では $9N(N-1)$ 回の掛け算が必要となることが分かる．

次に EJTG の接触力に関する項について掛け算の数を数え上げる．図 3.13 は EJTG 接触力項の各要素の計算式を示す．接触点は動かないため接触力の発揮される点はリンクに固定し

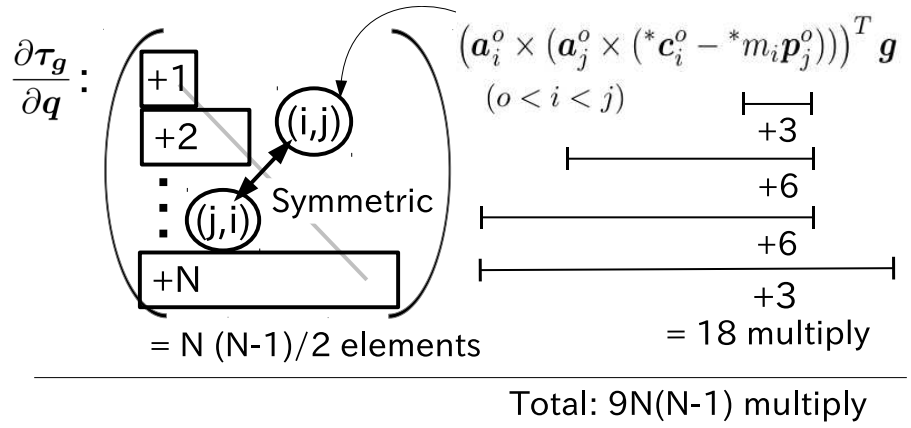


図 3.12. EJTG computational complexity analysis for the gravity force term. Because this term is a symmetric matrix, it is enough to compute the lower triangular part ($\sum_{i\in[0,N]} i = N(N-1)/2$ elements).

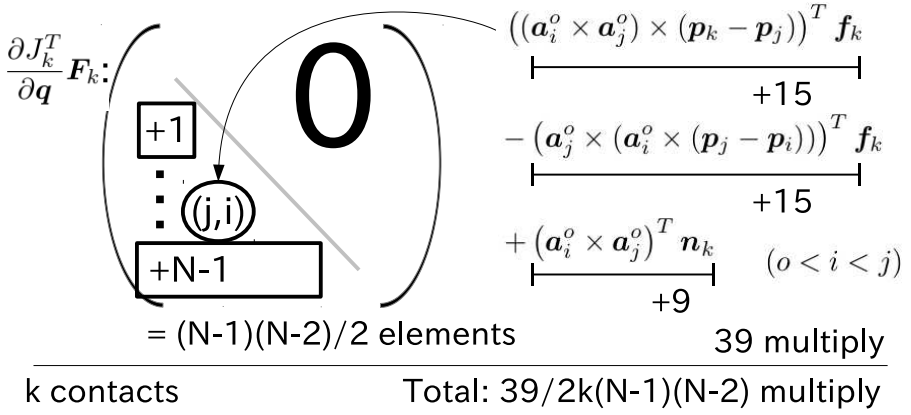


図 3.13. EJTG computational complexity analysis for the contact force and moment term. $\boldsymbol{f}_k$ is the 3-dof contact force of k-th contact, and $\boldsymbol{n}_k$ is the 3-dof contact moment. Because all elements of upper triangular part are zero, it is enough the lower triangular part excluding the diagonal elements ($\sum_{i\in[0,N)} i = (N-1)(N-2)/2$ elements).

ても環境に固定してもよい．本章では後者の立場をとる．重力はリンクに固定される位置に加わる力なので図 3.12 とは異なる式になることに注意してほしい．各要素の計算については 3.3.6.2 章で補足する．図から，掛け算の総数は $39/2k(N-1)(N-2)$ であることが分かる．重力項と合わせると全体の掛け算総数は $9N(N-1) + 39/2k(N-1)(N-2)$ である．また，ここまでの考察は掛け算のみを数え上げているが，実際には足し算やメモリ確保といった要素も計算時間に影響を与えているため正確にこの値が計算時間を表すわけではないことに注意してほしい．また以下の実験では，EJTG の SR Inverse を用いて関節角度の更新量を計算して

いる．1) as follows:

$$\Delta \boldsymbol{q} = \left(\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}} \right)^{\#} \Delta \boldsymbol{\tau} \tag{3.31}$$

ここで式 3.31 の計算量は $\mathcal{O}(N^3)$ である．しかし，EJTG の計算量の係数が $9 + 39/2k$ であることを 3.3.3.1.2 章でこれまで確認してきた．したがって， $N = 30$ かつ $k = 4$ の問題では計算量の主要な項は SR Inverse の計算に生じる $N \times N^2 = 30 \times N^2$ ではなく，$9 + 39/2k \times N^2 = 87 \times N^2$ である．

3.3.3.2　擬似関節負荷トルク勾配: 接触力近似による計算高速化

前 3.3.3.1.2 章で述べたように，EJTG の計算量は $\mathcal{O}(N^2)$ となる．一方，逆運動学で用いられるその他の計算はたかだか $\mathcal{O}(N)$ であり，EJTG は計算量のボトルネックとなりうる．そこで本章では，$\mathcal{O}(N)$ で計算可能な擬似関節負荷トルク勾配 PJTG を提案する．本章のアイデアは，関節負荷トルクを仮想的に接触力・モーメントとして扱うことで接触力・モーメントの勾配を負荷トルク勾配のかわりにもちいるというものである．仮想力・モーメントを ${}^{\tau}\boldsymbol{F}_i$ とすると，$\boldsymbol{\tau}_i = \hat{\boldsymbol{J}}_i{}^T {}^{\tau}\boldsymbol{F}_i$ の条件から以下を得る．

$$\forall i;\ {}^{\tau}\boldsymbol{F}_i = (\hat{\boldsymbol{J}}_i{}^T)^{\#} \boldsymbol{\tau}_i \tag{3.32}$$

ここで，$\boldsymbol{\tau}_i$ は全関節の負荷トルクをあらわし，図 3.11(b) のリンクチェーンを用いて各接触リンク i からルートリンクへと連なるリンク系についてそれぞれ定義する．${}^{\tau}\boldsymbol{F}_i$ は i 番めの接触リンクに加わる仮想力・モーメント，$\hat{\boldsymbol{J}}_i$ は i 番めの接触のヤコビ行列を接触リンクからルートリンクまでのリンク系で計算したものである．ここで，図 3.11(b) のキネマティクスチェーンを用いたのは接触を区別しないためである．図 3.11(a) のチェーンではルートリンクの接触に連なるリンク系は存在しないので仮想力・モーメントを計算できない．${}^{\tau}\boldsymbol{F}_i$ と接触力・モーメント勾配式 3.18 を用いて重心位置を以下のように更新する．

$$\Delta \boldsymbol{c} = \sum\nolimits_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} \Delta {}^{\tau}\boldsymbol{F}_i$$

また重心ヤコビアンを用いて以下が成り立つ．

$$\Delta \boldsymbol{c} = \boldsymbol{J}_c \Delta \boldsymbol{q} \tag{3.33}$$

ここで，$\boldsymbol{J}_c$ はロボットの全身リンクに対する重心を全関節角度で微分したもの（重心ヤコビアン）である．重心ヤコビアンを用いて PJTG を以下のように定義する．

$$\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}}^{'} = \left(\boldsymbol{J}_c^{\#} \sum\nolimits_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} (\hat{\boldsymbol{J}}_i{}^T)^{\#} \right)^{\#} \tag{3.34}$$

式 3.34 を用いて，関節負荷トルクを減少させるための関節角度の更新式は以下のようになる．

$$\Delta \boldsymbol{q} = \boldsymbol{J}_c^{\#} \sum\nolimits_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} (\hat{\boldsymbol{J}}_{\boldsymbol{i}}{}^T)^{\#} \Delta \boldsymbol{\tau}_i \tag{3.35}$$

これは，アルゴリズム 1 における *calcTorqueGradient* 関数の PJTG 計算である．ここで，$\boldsymbol{J}_\tau = \boldsymbol{J}_c$，$\boldsymbol{v}_\tau = \sum_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} (\hat{\boldsymbol{J}_i}^T)^\# \Delta\boldsymbol{\tau}_i$．次に PJTG の計算量を見積もる．図 3.14 に計算の概略を示す．

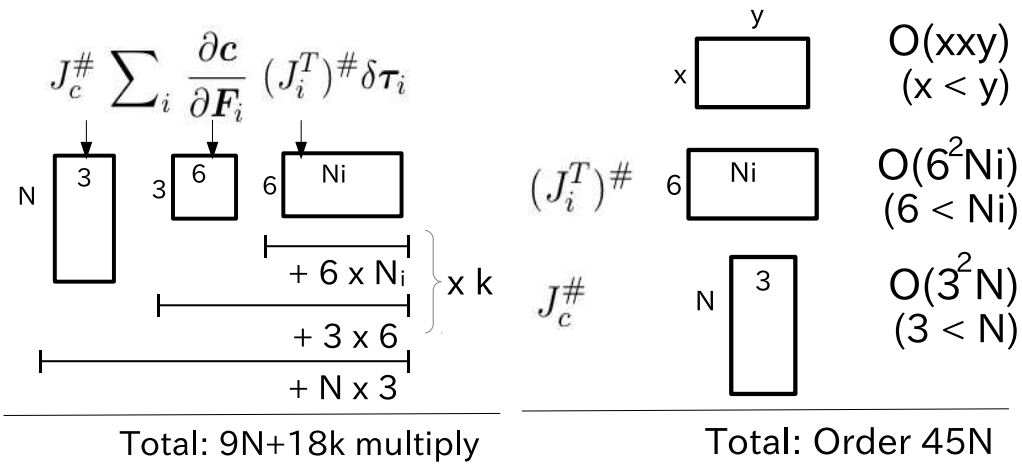
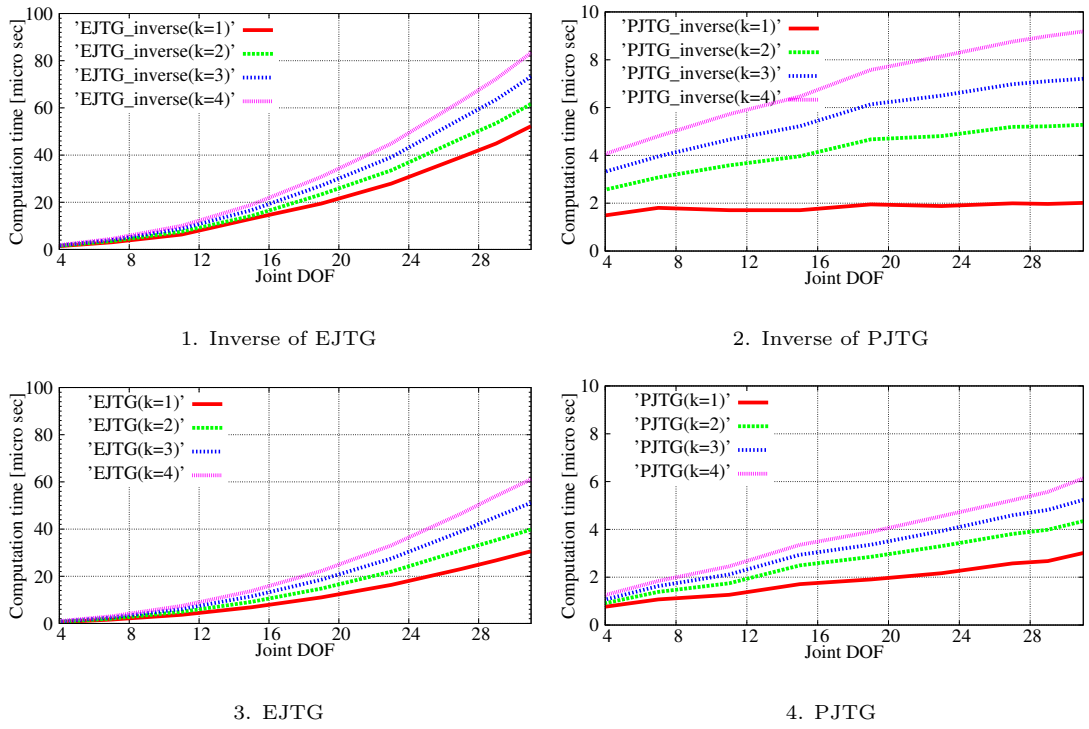


図 3.14. PJTG$^\#$ computational complexity analysis, with the left half showing the multiplications and the right half showing the inverse calculations.

はじめに各行列が計算できているとして，その掛け算に必要な計算量を見積もる．総関節数 N，全接触数 k，接触毎のヤコビ行列 $\boldsymbol{J}_i$ とその列数 N_i ($N \approx \sum_i N_i$)．一般に，$x \times y$ 行列と $y \times z$ 行列の掛け算には，$x \times y \times z$ 回の掛け算が必要である．したがって，式 3.35 の計算では，$N \times 6$, 3×6, $6 \times N_i$, $N_i \times 1$ の行列をかけるため，右から描けていくことで，$\sum_i (6 \times N_i \times 1 + 3 \times 6 \times 1) + N \times 3 \times 1 = 9N + 18k$ 回の掛け算が必要である．以上から掛け算の計算量が $\mathcal{O}(N)$ であることも分かる．次に行列を計算するための計算量について考察する．ヤコビ行列 $\boldsymbol{J}_i$ と重心ヤコビアン $\boldsymbol{J}_c$ は計算量が $\mathcal{O}(N)$ であることが知られている．ムーアペンローズの擬似逆行列では，$x \times y$ の行列に対して計算量が $x > y$ ならば $\mathcal{O}(xy^2)$, $y > x$ ならば $\mathcal{O}(x^2 y)$ であることが知られている．$\boldsymbol{J}_c$ は $3 \times N$ 行列，$\boldsymbol{J}_i$ は $6 \times N_i$ 行列であるので全体の計算量は $\mathcal{O}(3^2 N) + \sum_i \mathcal{O}(6^2 N_i) = \mathcal{O}(45N)$ となる．以上の結果から，PJTG の計算に必要な掛け算の数は $54N + 18k$ 程度となる．EJTG の計算量と比較するとオーダで N 倍軽量である．さらに，EJTG の計算量の係数は $9 + 39/2k$ であるので，$N = 30$ かつ $k = 4$ とすると，$(9 + 39/2 \times 4) \times 30 \times 30 = 25,650$ であるのに対して，PJTG は $54 \times 30 + 18 \times 4 = 1692$ である．掛け算のみを数え上げているので厳密な計算ではないが，PJTG が EJTG より十倍程度高速に計算できることが確認できた．また，実測により速度を比較した結果も以下に示す．



1. Inverse of EJTG　　2. Inverse of PJTG

3. EJTG　　4. PJTG

図 3.15. A comparison of computational times of the EJTG and PJTG with respect to the total number of joints. Here, k indicates the number of contacts, with: $k = 1$: arm only; $k = 2$: both arms; $k = 3$: both arms and right leg; and $k = 4$: both arms and both legs.

3.3.3.2.1　PJTG と EJTG の関係について

接触力・モーメントが静的釣り合いの式から計算できるとし，力の関節角度による微分が可能であるならば，EJTG には以下のように第三項が存在する．

$$\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}} = \left(\frac{\partial \boldsymbol{\tau}_g}{\partial \boldsymbol{q}} + \sum\nolimits_i \frac{\partial \boldsymbol{J}_i^T}{\partial \boldsymbol{q}} \boldsymbol{F}_i \right) + \left(\sum\nolimits_i \boldsymbol{J}_i^T \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{q}} \right) \tag{3.36}$$

右辺第一項，第二項はこれまで EJTG と呼んできたものである．式 3.11 から力の勾配が計算できるとすると，

$$\begin{aligned} \boldsymbol{F} &= -\boldsymbol{G}^{\#}[m\boldsymbol{g}^T, (\boldsymbol{c} \times m\boldsymbol{g})^T]^T \\ &= [\boldsymbol{F}_0^T, \cdots, \boldsymbol{F}_{k-1}^T]^T \\ \boldsymbol{G} &= [\boldsymbol{G}(x_0), \cdots, \boldsymbol{G}(x_{k-1})] \end{aligned} \tag{3.37}$$

$\boldsymbol{G}$ の擬似逆行列は複数考えられるので，3.37 は唯一の解ではないことに注意してほしい．

まとめると，式 3.36 右辺第三項は以下のようになる．

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}} = \boldsymbol{G}^{\#} \begin{pmatrix} \mathbf{0} \\ m\boldsymbol{g}\times \end{pmatrix} \boldsymbol{J}_c \tag{3.38}$$

$$\begin{aligned} \boldsymbol{J}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}} &= \sum\nolimits_i \boldsymbol{J}_i^T \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{q}} \\ &= \boldsymbol{J}^T \boldsymbol{G}^{\#} \begin{pmatrix} \mathbf{0} \\ m\boldsymbol{g}\times \end{pmatrix} \boldsymbol{J}_c \end{aligned} \tag{3.39}$$

$$\boldsymbol{J} = [\boldsymbol{J}_0, \cdots, \boldsymbol{J}_{k-1}]$$

式 3.39 の擬似逆行列を考えることで，式 (3.34 と似た式が得られる．

$$\begin{aligned} \left(\boldsymbol{J}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}\right)^{\#} &= \boldsymbol{J}_c^{\#} \begin{pmatrix} \mathbf{0} \\ m\boldsymbol{g}\times \end{pmatrix}^{\#} \boldsymbol{G} (\boldsymbol{J}^T)^{\#} \\ &\approx \boldsymbol{J}_c^{\#} \begin{pmatrix} \mathbf{0} \\ m\boldsymbol{g}\times \end{pmatrix}^{\#} \sum\nolimits_i \boldsymbol{G}_i (\hat{\boldsymbol{J}}_i{}^T)^{\#} \end{aligned} \tag{3.40}$$

$$= \boldsymbol{J}_c^{\#} \sum\nolimits_i \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{F}_i} (\hat{\boldsymbol{J}}_i{}^T)^{\#} = \left(\frac{\partial \boldsymbol{\tau}'}{\partial \boldsymbol{q}}\right)^{\#} \tag{3.41}$$

ここで，式 3.40 は，$\boldsymbol{J} = diag(\hat{\boldsymbol{J}}_0, \cdots, \hat{\boldsymbol{J}}_{k-1})$ が成り立つとき等式になる．図 3.11(b) で例えば腰関節がある場合には同じ関節が複数の接触リンク系に含まれるため近似になる．これらの結果から PJTG には三つの仮定が必要であることが分かる．まず，接触力・モーメントはロボットのコンフィギュレーションにより決定されるということ．次に，式 3.36 の右辺第一項，第二項を無視していること．最後に接触リンク系は互いに共通して含まれる関節が少ないということ．

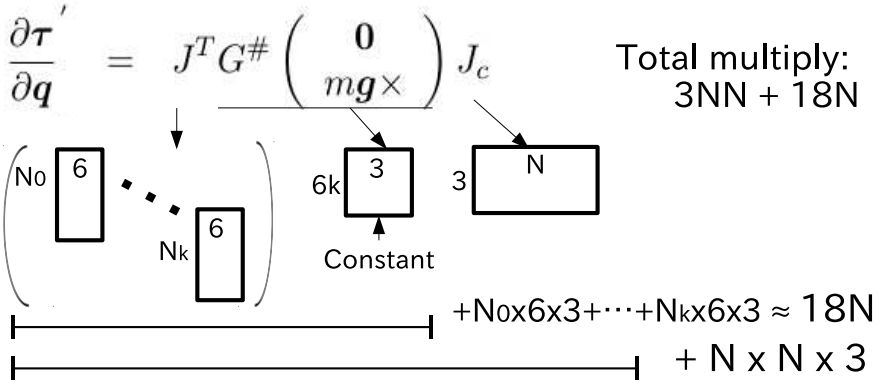


図 3.16. PJTG computational complexity analysis. Here, if $N = 30$ and $k = 4$, the total number of multiplications is 3240, which is approximately eight times smaller than that of the EJTG.

図 3.16 に式 3.39 の計算量を示す．3.3.3.1.2 章と同様に掛け算のみを数え上げると $N = 30$ かつ $k = 4$ の時には 3240 回の掛け算が必要であり，EJTG より 10 倍程度高速に計算できることが予想される．また，計算オーダーは $\mathcal{O}(N^2)$ である．

3.3.3.3　計算時間の測定

図 3.15 に計算時間を実際にコンピュータで測定した結果をしめす．ここで，k は全接触の数であり，$k = 1$ は右手のみ接触，$k = 2$ は両手，$k = 3$ は両手と右足，$k = 4$ は両手両足の接触を表す．PJTG は EJTG より $N = 30$ 付近でちょうど 10 倍程度高速に計算できていることが確認された．用いたコンピュータは Intel(R) Core(TM) i7-4770S CPU 3.10GHz で本章の実験では全て同様である．

3.3.3.4　関節負荷トルク勾配の零点について

PJTG が EJTG の代わりに使えるかどうかの確認として両勾配の零点について調べる．

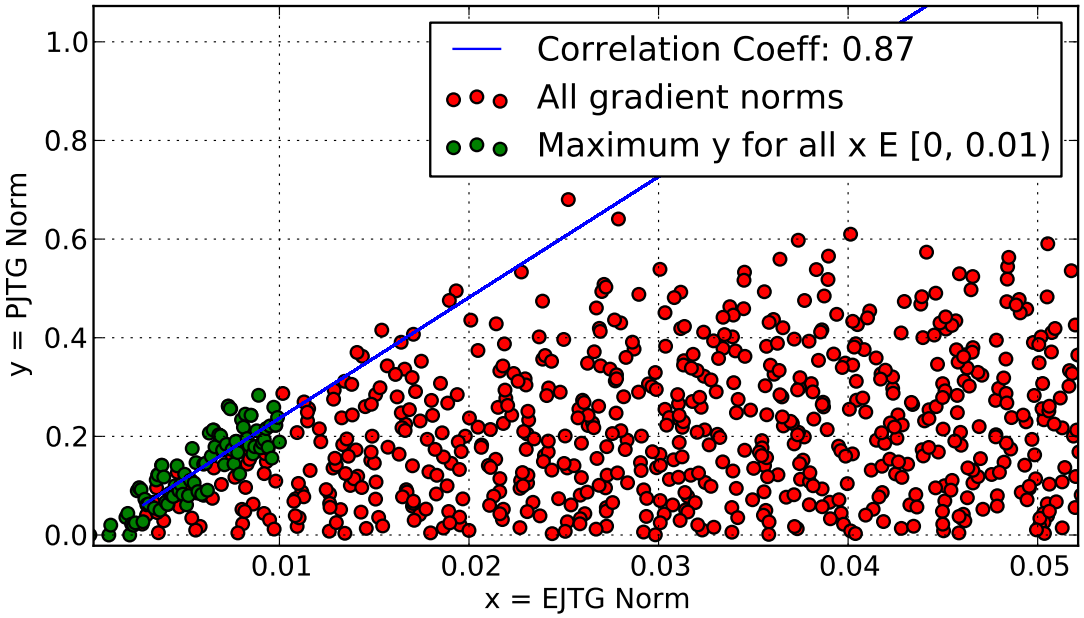


図 3.17. The relationship between EJTG and PJTG norms with random configurations. Blue line is the regression line of EJTG norms and the upper bounds of PJTG norms.

PJTG の零点は EJTG の零点に含まれることが確認された．図 3.17 は両勾配の大きさを二次元にプロットしたものである．図の生成には以下の手続きを用いた．

1. ランダムな姿勢と接触力・モーメントを生成する：全関節角度と接触力・モーメントをそれぞれの制約を満たすようにランダムに決める．関節角度は可動域を満たすように．接触力・モーメントは一つの接触については 3.11 を満たすように計算し残りは乱数により決定した．
2. トルクの二乗の勾配を計算する: 本章ではトルクの二乗を評価関数に含む．このとき収

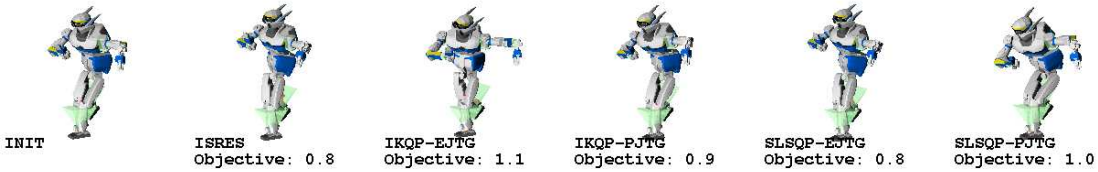


図 3.18. Generated feasible postures with both legs as contacts (i.e., $k = 2$ and $N = 12$)

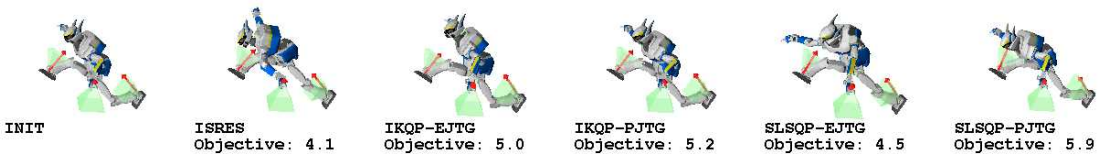


図 3.19. Generated feasible postures with both legs and a left arm as contacts (i.e., $k = 3$ and $N = 21$)

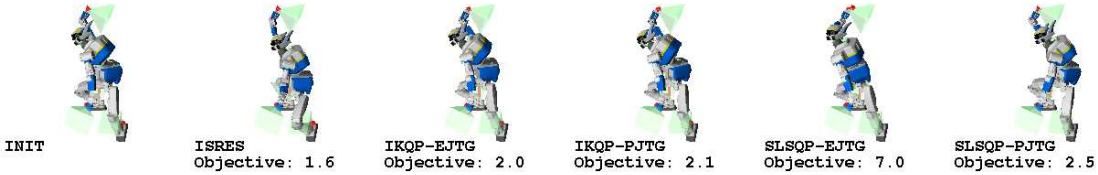


図 3.20. Generated feasible postures with both legs and both arms as contacts (i.e., $k = 4$ and $N = 28$)

束条件は以下のようになる．

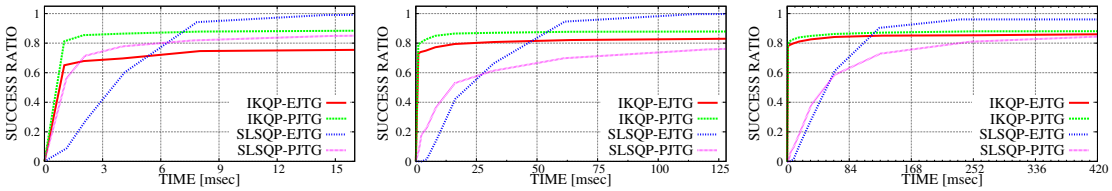
$$\|\frac{1}{2}\frac{\partial(\boldsymbol{\tau}^T\boldsymbol{\tau})}{\partial \boldsymbol{q}}\|^2 = \|\left(\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\tau}\|^2 = 0 \tag{3.42}$$

3. 勾配のノルムをプロットする: 式 3.42 から 100,000 点をプロットした．横軸に EJTG のノルム，縦軸に PJTG のノルムをプロットした．
4. PJTG ノルムと EJTG ノルム上側境界の相関: $x \in [0, 0.01)$ を 100 のブロックに分割する．それぞれのブロックの y で最大のものを集めて相関係数を計算する．

図より PJTG が零であっても EJTG が零とは限らないことが分かる．また，EJTG が零に近づくときの PJTG の最大値は EJTG と強い相関を持ち 0.87，EJTG が零に近づくことで PJTG も零に近づくことが示唆された．

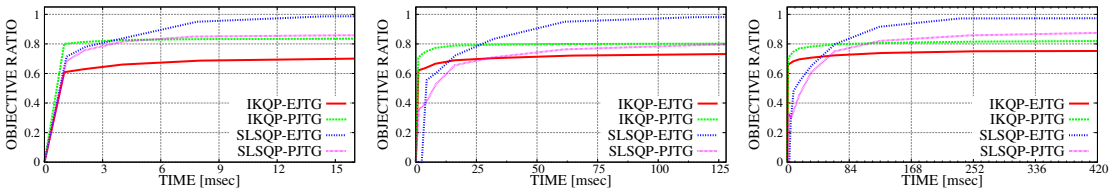
3.3.4 擬似負荷トルク勾配を用いた姿勢探索性能の評価実験

本章では，両勾配を用いたときの探索性能と計算時間について比較を行う．最適化アルゴリズムとして，IK-QP 繰り返し法と SLSQP [81] を用いる．SLSQP はオープンソースの非線形最適化ライブラリ NLopt [14] の実装を用いる．また正解データとして，同じく NLopt に実装された大域的最適化のための進化計算アルゴリズム ISRES [9][10] を用いて探索された値を用いる．比較する内容は，計算時間，評価関数の値，探索の成功率とする．ロボットのモデ



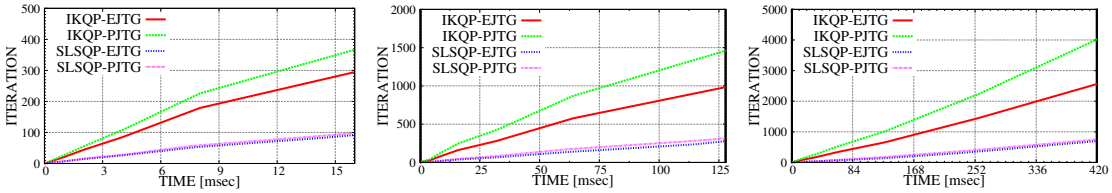
1. $k = 2, N = 12$　　2. $k = 3, N = 21$　　3. $k = 4, N = 28$

図 3.21. Total number of feasible solutions/that of global solver. The leftmost image shows both legs as contacts (i.e., $k = 2$), the center image shows both legs and left arm as contacts (i.e., $k = 3$), and the rightmost image shows both legs and arms as contacts (i.e., $k = 4$).



1. $k = 2, N = 12$　　2. $k = 3, N = 21$　　3. $k = 4, N = 28$

図 3.22. Average objective value of feasible solutions/that of global solver. The leftmost image shows both legs as contacts (i.e., $k = 2$), the center image shows both legs and left arm as contacts (i.e., $k = 3$), and the rightmost shows image both legs and arms contacts (i.e., $k = 4$).



1. $k = 2, N = 12$　　2. $k = 3, N = 21$　　3. $k = 4, N = 28$

図 3.23. Total iterations of all four algorithms, i.e., IKQP-PJTG, IKQP-EJTG, SLSQP-EJTG, and SLSQP-PJTG. The leftmost image shows both legs as contacts (i.e., $k = 2$), the center images shows both legs and a left arm as contacts (i.e., $k = 3$), and the rightmost image shows both legs and arms as contacts (i.e., $k = 4$).

ルとしては HRP-2 [84][65] を改良した HRP2JSK [66] を用いた.

3.3.4.1　実験に用いる姿勢探索アルゴリズム

3.3.4.1.1　IKQP-EJTG

アルゴリズム 1 + EJTG で 式 3.15 を解く.

3.3.4.1.2 IKQP-PJTG

アルゴリズム 1 + PJTG で式 3.15 を解く.

3.3.4.1.3 SLSQP-EJTG

SLSQP [81] + EJTG で式 3.14 を解く.

3.3.4.1.4 SLSQP-PJTG

SLSQP [81] + PJTG で式 3.14 を解く.

3.3.4.1.5 ISRES

ISRES [9][10] で 式 3.14 を解く.

3.3.4.2 進化計算アルゴリズムを用いたテストデータ生成

成功率や解の質を評価するために，接触数が $k = 2$, $k = 3$，$k = 4$ の三つの状態でそれぞれテストデータを生成した．テストデータは接触状態とそれを満たす最適解の組，千ペアからなる．接触状態はランダムに生成され，それぞれについて ISRES を用いて最適解をあらかじめ求めた．ランダムな突然変異により接触位置を厳密に満たすような解を見つけるのは時間がかかるので以下のヒューリスティックを用いた.

$$
\begin{aligned}
min_{\boldsymbol{q}_0,\boldsymbol{F}} : & \sum\nolimits_i \boldsymbol{BRL}\boldsymbol{v}_i^T \boldsymbol{BRL}\boldsymbol{v}_i \\
s.t. \quad & \boldsymbol{q} = localIK(\boldsymbol{q}_0, \boldsymbol{x}_0^d, \cdots, \boldsymbol{x}_{k-1}^d) \\
& [\boldsymbol{\tau},\ \boldsymbol{F}] = QP(\boldsymbol{q}) \\
& checkConstraint(\boldsymbol{q}, \boldsymbol{\tau}, \boldsymbol{F})
\end{aligned}
\tag{3.43}
$$

$localIK$ は, 初期姿勢 $\boldsymbol{q}_0$ を, 目標接触位置姿勢 $\boldsymbol{x}_0^d \cdots \boldsymbol{x}_{k-1}^d$ を満たす姿勢 $\boldsymbol{q}$ に補正する関数である．$localIK$ は Sec.3.3.2.2.4 の ”inverseKinematics” 関数を 10iteration 呼び出す実装を用いた．$checkConstraint$ は補正された姿勢 $\boldsymbol{q}$ が接触位置姿勢の拘束を満たすことの確認と, その姿勢での QP の解 $\boldsymbol{\tau}$ と $\boldsymbol{F}$ が摩擦やトルクの条件を満たすことの確認を行う関数である．ISRES の計算は一つの接触状態ごとに 30 分探索した．30 分という時間は前述した 4 つのアルゴリズムの出力が全て ISRES の解より評価値が低くなる値をヒューリスティックに決定した.

3.3.4.3 姿勢探索性能の評価実験の実験結果

図 3.18-3.20 は生成された姿勢の一部である．図 3.21-3.23 は，成功率，評価関数，総イテレーション数を時間を横軸にプロットしたものである．以下に詳しく各図を見ていく.

表 3.5. IKQP-PJTG ($k = 2, N = 12$)

	msec	% ± SD
Total	2.7	100.0 ± 0.0
FK	0.6	24.0 ± 2.7
ID	0.7	24.9 ± 2.3
QP	0.6	23.0 ± 5.5
PJTG$^{\#}$	0.3	11.8 ± 1.4
J$^{\#}$	0.3	10.6 ± 1.6
		81.0 Iterations

表 3.6. IKQP-EJTG ($k = 2, N = 12$)

	msec	% ± SD
Total	3.4	100.0 ± 0.0
FK	0.6	19.1 ± 1.7
ID	0.7	20.6 ± 2.1
QP	0.7	20.5 ± 4.8
EJTG	0.8	23.3 ± 2.1
J$^{\#}$	0.4	12.1 ± 1.9
		81.0 Iterations

表 3.7. SLSQP-PJTG ($k = 2, N = 12$)

	msec	% ± SD
Total	7.6	100.0 ± 0.0
FK	1.3	16.7 ± 6.5
ID	0.5	6.7 ± 2.6
PJTG	0.2	2.2 ± 0.8
SQP	5.7	74.5 ± 9.6
		58.8 Iterations

表 3.8. SLSQP-EJTG ($k = 2, N = 12$)

	msec	% ± SD
Total	9.6	100.0 ± 0.0
FK	0.7	7.4 ± 2.3
ID	0.3	2.9 ± 0.9
EJTG	0.7	7.3 ± 2.7
SQP	7.9	82.4 ± 5.0
		66.2 Iterations

3.3.4.3.1　成功率

Fig. 3.21 にそれぞれのアルゴリズムが出力した実現可能な解の数を,ISRES のそれで割った値をグラフにしたものを示す．ここで，$k = 2$ は両足が接触，$k = 3$ あ両足と左手が接触，$k = 4$ は両手両足が接触しているとする．正答率だけみると SLSQP-EJTG が最も高く, 接触数 $k = 2, 3, 4$ の全てで最終的には 100% 近い値となっている．$k = 4$ の時を見てみると，683 の解が ISRES により見つかっているが，SLSQP-EJTG では 658 （96%）が数百ミリ秒で見つかっている．これは二番目に正答率の高かった，IKQP-PJTG の 88%（601/683）と比べて 8% 程度高い．収束を見ると，IKQP のほうが SLSQP より早いことも分かる．全ての接触数で IKQP は数ミリ秒で正答率 80% である．IKQP-PJTG の四ミリ秒での正答率は四百ミリ秒時の正答率の 93%（0.82/0.88）である．一方，SLSQP-EJTG では 124 ミリ秒での正答率が四百ミリ秒時の 94%（0.90/0.96）となっている．勾配で比較すると，PJTG は IKQP と SLSQP のどちらを用いた場合でも正答率の立ち上がりが早い.

3.3.4.3.2　評価関数

図 3.22 はそれぞれのアルゴリズムの平均評価関数値を ISRES おそれで正規化したものをプロットしている．評価関数の値は探索に失敗した場合には極めて大きくなる可能性があるた

表 3.9. IKQP-PJTG ($k = 3, N = 21$)

	msec	% ± SD
Total	5.1	100.0 ± 0.0
FK	0.8	15.0 ± 3.2
ID	1.2	22.8 ± 3.0
QP	1.9	36.3 ± 9.4
PJTG$^{\#}$	0.5	10.0 ± 2.1
J$^{\#}$	0.6	11.8 ± 2.8
81.0 Iterations		

表 3.10. IKQP-EJTG ($k = 3, N = 21$)

	msec	% ± SD
Total	7.7	100.0 ± 0.0
FK	0.8	9.8 ± 1.4
ID	1.2	15.7 ± 2.3
QP	2.0	26.2 ± 7.6
EJTG	2.6	33.7 ± 4.6
J$^{\#}$	0.9	12.0 ± 2.3
81.0 Iterations		

表 3.11. SLSQP-PJTG ($k = 3, N = 21$)

	msec	% ± SD
Total	23.9	100.0 ± 0.0
FK	1.4	5.8 ± 2.4
ID	0.8	3.2 ± 1.4
PJTG	0.3	1.4 ± 0.5
SQP	21.4	89.6 ± 4.1
69.1 Iterations		

表 3.12. SLSQP-EJTG ($k = 3, N = 21$)

	msec	% ± SD
Total	32.9	100.0 ± 0.0
FK	0.9	2.7 ± 0.9
ID	0.5	1.4 ± 0.4
EJTG	2.8	8.6 ± 3.1
SQP	28.8	87.3 ± 4.2
75.9 Iterations		

め評価値の平均を計算する際には探索に成功しているもののみを用いている．全体の傾向としては図 3.21 に近い．最終的には SLSQP-EJTG がもっとも良い答えを買えしていることが分かる．IKQP-PJTG では 80% 程度の解を数ミリ秒で見つけている．

3.3.4.3.3 イテレーション数

図 3.23 は各アルゴリズムの総イテレーション数を表す．SLSQP-EJTG と SLSQP-PJTG のグラフはよく似ている．IKQP-EJTG は SLSQP よりイテレーションが多く, IKQP-PJTG は IKQP-EJTG と比べて 1.5 倍程度多い．このことは IKQP において関節負荷トルク勾配が計算時間のボトルネックとなっていることを意味する．大して SLSQP で差がないことは勾配計算以外の手続き，例えば二次計画問題がボトルネックになっていることが分かる．IKQP では接触力・モーメント空間の $6k$ 次元で二次計画問題を解くが，SLSQP では全関節角度空間と接触力・モーメント空間で二次計画問題を解いているためと予想される．

3.3.4.3.4 計算時間の構成要素

表 3.5-3.16 に計算時間の内訳を示す．いずれも 80 イテレーション付近でのデータであるが，NLopt 実装の SLSQP は勾配が零になった時点で探索をやめるため 80 よりやや短くなっている．IKQP は勾配の大きさにかかわらず 80 イテレーションの計算を行った．表の項目は

表 3.13. IKQP-PJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	9.0	100.0 ± 0.0
FK	0.9	9.7 ± 2.6
ID	1.8	19.4 ± 3.8
QP	4.3	47.2 ± 9.6
PJTG$^{\#}$	0.7	8.2 ± 2.2
J$^{\#}$	1.2	12.8 ± 3.4
		81.0 Iterations

表 3.14. IKQP-EJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	13.4	100.0 ± 0.0
FK	0.8	6.3 ± 1.1
ID	1.8	13.5 ± 2.9
QP	4.4	32.9 ± 8.6
EJTG	4.5	33.9 ± 5.9
J$^{\#}$	1.6	11.6 ± 2.5
		81.0 Iterations

表 3.15. SLSQP-PJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	53.6	100.0 ± 0.0
FK	1.4	2.5 ± 1.0
ID	0.8	1.5 ± 0.7
PJTG	0.5	0.9 ± 0.3
SQP	50.9	95.0 ± 1.8
		73.3 Iterations

表 3.16. SLSQP-EJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	68.2	100.0 ± 0.0
FK	1.0	1.4 ± 0.4
ID	0.6	0.8 ± 0.2
EJTG	4.9	7.2 ± 2.4
SQP	61.7	90.6 ± 3.0
		77.5 Iterations

以下である．順運動学 FK はヤコビアンや重心位置の計算も含む．逆運動学 ID は接触力・モーメントやトルクの制約チェックも含む．QP は式 3.15 を解くのにかかった時間．PJTG, EJTG は勾配の計算にかかる時間．SQP は SLSQP の計算で発生する時間でほぼ QP の計算である．また，SD は千のテストデータを用いた標準偏差を示す．IKQP-EJTG の表 3.6, 3.10, and 3.14) では EJTG がもっとも時間を食うことが分かる．対して，IKQP-PJTG の表 3.5, 3.9, 3.13 では PJTG がボトルネックとならないことで高速化されていることが分かる．

$k = 4, N = 28$ のときを見てみると,EJTG が全体にしめる計算時間は 33.9%, QP のそれは 32.9% である．したがって,PJTG を用いて EJTG の計算時間が QP にくらべて無視できる程度まで高速化できるとすると探索全体にかかる時間は $33,9\% \approx 4.54$[msec] 程度高速化されることが期待できる．実際,$k = 4, N = 28$ での IKQP-PJTG の計算時間内訳 3.13 を見てみると総計さん時間は 4.4[msec] 程度高速であり, 期待通りの結果となっている．また,QP の計算時間比率の SD が高めであることに注目して欲しい．これは QP が探索アルゴリズムであり, 実現可能解が存在しない問題については多くの探索を必要とするためである．

実際, 最終的に実現可能解が得られた接触状態のみでの平均をとってみると, Table 3.17,3.18 のようになった．QP の比率がさがり,EJTG の比率が上がっていることがわかる．より顕著な例として, Fig. 3.9 のように, 初期状態から実現可能な問題では, さらに QP の比率が下がることになる．つぎに SLSQP の表 3.7, 3.11, 3.15, 3.8, 3.12, 3.16 を見てみると, 計算時間のボ

表 3.17. IKQP-PJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	7.5	100.0 ± 0.0
FK	0.8	11.3 ± 2.2
ID	1.3	17.4 ± 3.3
QP	3.3	43.5 ± 10.5
PJTG$^{\#}$	0.7	9.6 ± 1.8
J$^{\#}$	1.1	14.9 ± 3.0
81.0 Iterations		

表 3.18. IKQP-EJTG ($k = 4, N = 28$)

	msec	% ± SD
Total	11.8	100.0 ± 0.0
FK	0.8	7.0 ± 0.9
ID	1.3	11.3 ± 1.4
QP	3.4	28.6 ± 8.4
EJTG	4.5	37.7 ± 4.7
J$^{\#}$	1.6	13.2 ± 2.1
81.0 Iterations		

トルネックは QP が 90% 以上を占めていることがわかる. これは IKQP が 接触反力空間でのみ QP を解いていたのに対し, SLSQP は接触反力と全関節角度の空間で QP を解いているためである. したがって,EJTG を高速化することによる探索アルゴリズム全体の高速化は IKQP ほどではなかった.

3.3.4.3.5 特異姿勢と成功率

多リンク系のマニピュレーション能力を表現する指標として, 可操作度 [85] がしばしば用いられる. Manipulability が小さな姿勢は特異姿勢とも呼ばれ, ヤコビ行列の逆行列が計算できなくなることにより, 勾配法に基づく手法を発散させる危険性がある. 本章では, 可操作度を用いて SLSQP と IKQP,PJTG と EJTG の探索性能を考察する.

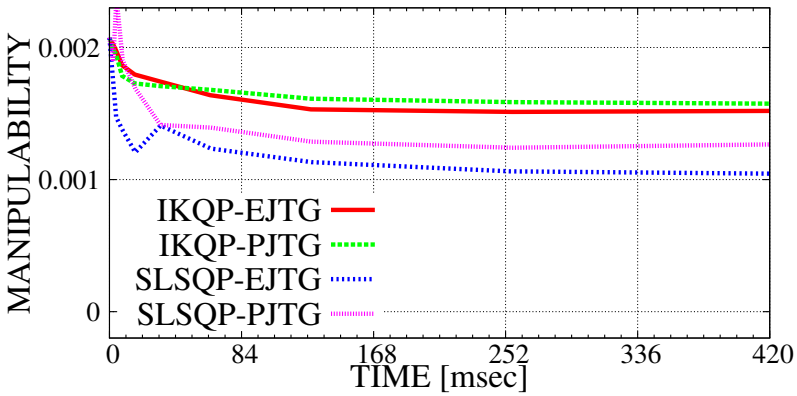
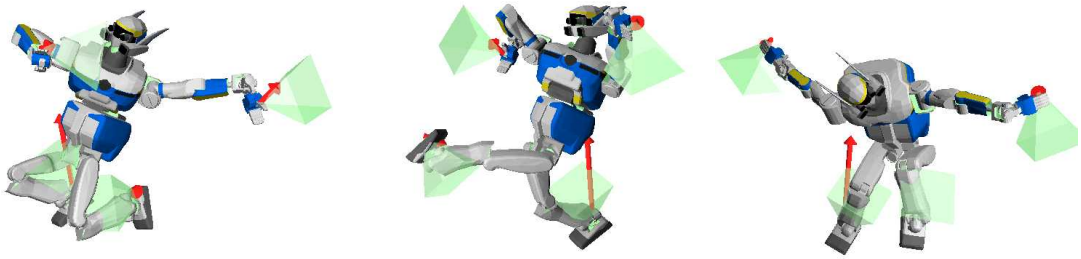


図 3.24. Average minimum manipulability of all limbs (i.e., $k = 4$ and $N = 28$) of feasible solutions. Here, the SLSQP has lower manipulability than that of the IKQP. Further, the PJTG has a slightly higher manipulability versus that of the EJTG.

$$\sigma_{min} = min\left(\sqrt{det(\hat{\boldsymbol{J}}_0\hat{\boldsymbol{J}}_0{}^T)}, \cdots, \sqrt{det(\hat{\boldsymbol{J}}_{k-1}\hat{\boldsymbol{J}}_{k-1}{}^T)}\right) \tag{3.44}$$

Fig. 3.24 に $k = 4, N = 28$ の条件で探索を行った時の, それぞれのアルゴリズムが出力する実現可能解の平均可操作度の遷移をグラフにしたものを示す.

$\hat{\boldsymbol{J}}_{i\in[0,k)}$ では接触リンクからルートリンクまでの末端側 6 自由度のみを用いて可操作度を計算している．これは平均をとったときに接触によって可操作度のオーダーに差が出ることを防ぐためである．最終的な平均可操作度を見ると,ISRES は 0.001015,SLSQP-EJTG は 0.001036 と ISRES にかなり近く,IKQP-PJTG では 0.001568 と 50% 程度大きかった. 全体では, ほぼ一貫して,SLSQP よりも IKQP のほうが 可操作度 は高く, ヤコビ行列の逆行列を計算しない SLSQP のほうが特異姿勢に近い解も出力できていることが確認できる. 実際,400 milli seconds 付近での IKQP-PJTG の出力解 602 のうち, 各接触四肢の 可操作度 最小値 が 10^{-4} を下回るものの数は 168 (28%) であったが, SLSQP-EJTG では 254/656 (39%) の違いが出た. さらに, SLSQP-EJTG では解くことができたが, IKQP-PJTG ではできなかった解は 78 あったが，そのうち 36（46%）は可操作度の最小値が 10^{-4} を下回った．いくつかの姿勢を下図 3.25 に示す.



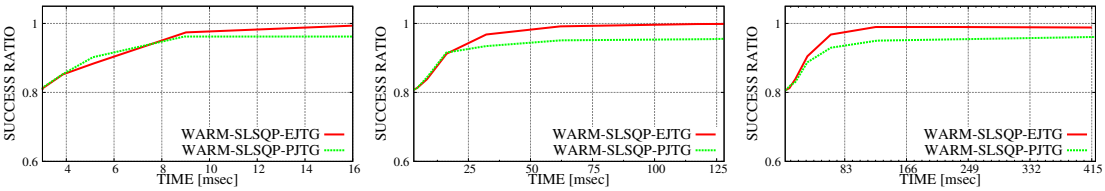
1. Left arm　　2. Left leg　　3. Both arm

図 3.25. Example postures with low manipulability, generated by the SLSQP-EJTG; the IKQP-PJTG solutions for these contact states were all infeasible.

この差は,SLSQP-EJTG が IKQP-PJTG より正答率が最終的には高くなることの理由が IKQP が特異姿勢の探索に不得手であるためという示唆している．また,PJTG と EJTG を比較してみると, いずれのアルゴリズムを用いた場合も EJTG の解のほうが可操作度が低くなっていることがわかる. したがって EJTG のほうが特異姿勢を得やすいことが示唆されている. このことは特異姿勢の探索に不得手な勾配法アルゴリズム IKQP において IKQP-PJTG が IKQP-EJTG の性能を上回る理由が特異姿勢にあり，特異姿勢を出力しやすい EJTG とヤコビ行列の逆行列を用いる IKQP 法の相性が悪いことを示唆している.

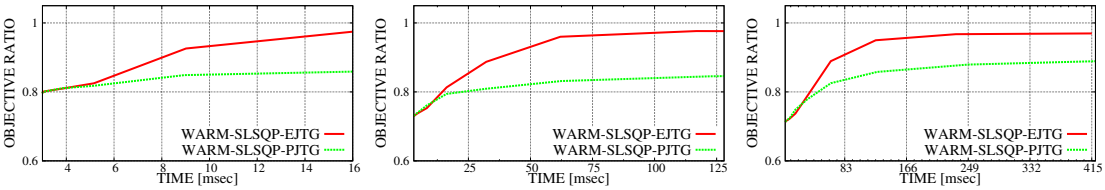
3.3.4.3.6　WARM-SLSQP: IKQP-PJTG を用いた SLSQP の高速化

SLSQP-EJTG は正答率は三十分の進化計算で得られた解にせまるものであることが図 3.21 から確認されたが,IKQP-PJTG にくらべ探索に時間がかかる点に改善の余地がある．本章では,IKQP-PJTG を用いて SLSQP-EJTG を warm start させることで性能が向上する



1. $k = 2, N = 12$　　2. $k = 3, N = 21$　　3. $k = 4, N = 28$

図 3.26. Total number of feasible solutions/that of global solver. Left image shows both legs contacts ($k = 2$), center shows both legs and left arm contacts ($k = 3$), and right shows both legs and arms contacts ($k = 4$).



1. $k = 2, N = 12$　　2. $k = 3, N = 21$　　3. $k = 4, N = 28$

図 3.27. Average objective value of feasible solutions/that of global solver. Left image shows both legs contacts ($k = 2$), center shows both legs and left arm contacts ($k = 3$), and right shows both legs and arms contacts ($k = 4$).

ことを検証する．図 3.26 は IKQP-PJTG で 三ミリ秒 探索したのち, それを初期値として SLSQP を用いた探索を行った時の正答率の遷移グラフである $k = 4, N = 28$ のときを見てみると, 正答率は 128 milli seconds 以降ほぼ横ばいで 99% となった．これは 図 3.21 における SLSQP-EJTG の正答率が 256 ミリ秒 以降横ばいで 96% であったのと比べて倍程度高速に, わずかに高い正答率が得られていることを意味している．図 3.27 は評価値の遷移であり, こちらも IKQP-EJTG より高速に収束していることが見て取れる．

3.3.5 擬似関節負荷トルク勾配についてのまとめ

ロボットの姿勢探索, 特に脚立登りや崖登りなどといった高い負荷が問題となるような探索問題では, 接触リンクの位置姿勢や転倒回避等の制約を満たすことに加え関節負荷を実現可能な範囲に収める工夫が必要不可欠である．過去の研究では, 関節負荷トルクの勾配を用いて勾配法により姿勢を探索する方法が提案させている．しかし関節負荷トルク勾配を用いた方法は計算コストに問題があるといえる．関節負荷トルクは全関節の自由度 N と同じだけの自由度を持つため, 勾配行列の大きさは N^2 となる. したがって, 最小でも $\mathcal{O}(N^2)$ の計算が必要となるためである．これは逆運動学におけるヤコビ行列の計算量が $\mathcal{O}(N)$ であることと比べて一桁大きく, 計算量のボトルネックとなりうる．本研究では時間計算量 $\mathcal{O}(N)$ で得られる擬似関節負荷トルク勾配を用いる方法を提案し, 関節負荷トルク勾配を用いた場合と比較して, 勾配の計算と姿勢探索全体の両方で高速であることを計算機実験により確認した．アルゴリズムと

して,SLSQP と IKQP を用いて探索の成功率と評価値の減少を比較したところ,IKQP では EJTG がボトルネックとなることが確認され, PJTG を用いることで各イテレーションにかかる時間を 30~40% 程度減少させられることが確認された. また, 正答率, 評価値ともに PJTG のほうがわずかに優れた結果となった．SLSQP と IKQP を比較すると,SLSQP-EJTG の性能が IKQP-PJTG を上回ったが, 探索速度では IKQP が高速だった．全四肢の接触でも IKQP-PJTG は 4[msec] で実現可能解の 82% を発見できたのに対し SLSQP-EJTG が 80% の解を発見するのに要した時間は 124[msec] 程度であった．また, 正答率の差について, 各アルゴリズムと勾配について探索により得られた姿勢の可操作度を比較したところ,EJTG は PJTG よりも低く,SLSQP は IKQP よりも低いことが確認された．このことは,SLSQP の性能が高い理由は SLSQP がヤコビ行列の逆を用いないために特異姿勢に強いためであり, IKQP-PJTG が IKQP-EJTG より性能で上回るのは EJTG が特異姿勢を含む可能性が高いためにヤコビ行列の逆を用いる探索手法と相性が悪いためであることが示唆された．PJTG はロボットに求められるタスクの複雑化，ロボットの大自由度化にともない今後ますます重要になっていくと考えている．

3.3.6　関節負荷トルク勾配計算法についての補足

3.3.6.1　１自由度回転ジョイント系の高速な同次変換行列勾配計算法

前章までで用いた計算について補足を行う．i 番めの関節の $i+1$ 番めから見た姿勢行列 $\boldsymbol{R}_{i+1}^{i}$ は以下のように計算できる．

$$\boldsymbol{R}_{i+1}^{i} = e^{[\boldsymbol{a}_i^i \times] q_i} = \sum_k \frac{q_i^k}{k!} [\boldsymbol{a}_i^i \times]^k \tag{3.45}$$

ここで，q_i は関節角度であり，$\boldsymbol{a}_i^i$ は回転軸ベクトルの i 番め関節座標での値である．次に, q_i により $i+1$ 番めの姿勢がどう回転するかは $\boldsymbol{R}_{i+1}^{i}$ 以下のように計算できる．

$$\frac{\partial \boldsymbol{R}_{i+1}^{i}}{\partial q_i} = [\boldsymbol{a}_i^i \times] e^{[\boldsymbol{a}_i^i \times] q_i} = [\boldsymbol{a}_i^i \times] \boldsymbol{R}_{i+1}^{i} \tag{3.46}$$

ここで，以下の計算が任意のベクトル $\boldsymbol{a}$, $\boldsymbol{b}$ と回転行列 $\boldsymbol{C}$ を用いて成り立つ.

$$\boldsymbol{C}(\boldsymbol{a} \times \boldsymbol{b}) = (\boldsymbol{C}\boldsymbol{a}) \times (\boldsymbol{C}\boldsymbol{b}) \tag{3.47}$$

以上から以下がなりたつ.

$$\begin{aligned} \boldsymbol{R}_i^{i-1} [\boldsymbol{a}_i^i \times] \boldsymbol{R}_{i+1}^{i} &= [(\boldsymbol{R}_i^{i-1} \boldsymbol{a}_i^i) \times] \boldsymbol{R}_i^{i-1} \boldsymbol{R}_{i+1}^{i} && (3.48) \\ &= [\boldsymbol{a}_i^{i-1} \times] \boldsymbol{R}_i^{i-1} \boldsymbol{R}_{i+1}^{i} && (3.49) \end{aligned}$$

さらに以下もなりたつ.

$$\begin{aligned}\frac{\partial \boldsymbol{R}_j^i}{\partial q_k} &= \prod_{l=i}^{j-1} \frac{\partial \boldsymbol{R}_{l+1}^l}{\partial q_k} \\ &= \boldsymbol{R}_{i+1}^i \cdots \frac{\partial \boldsymbol{R}_{k+1}^k}{\partial q_k} \cdots \boldsymbol{R}_j^{j-1} \\ &= [\boldsymbol{a}_k^i \times] \prod_{l=i}^{j-1} \boldsymbol{R}_{l+1}^l \\ &= [\boldsymbol{a}_k^i \times] \boldsymbol{R}_j^i \end{aligned} \tag{3.50}$$

同様に任意回微分も定義できる．$k_0 \cdots k_{n-1}$（$i \prec k_0 \prec \cdots \prec k_{n-1} \prec j$）による微分は,

$$\frac{\partial \boldsymbol{R}_j^i}{\partial q_{k_0} \cdots \partial q_{k_{n-1}}} = [\boldsymbol{a}_{k_0}^i \times] \cdots [\boldsymbol{a}_{k_{n-1}}^i \times] \boldsymbol{R}_j^i \tag{3.51}$$

と書ける．これらは各ジョイントの姿勢行列の値を順運動学計算を用いて予め持っておくことで n 回の行列の掛け算で求めることが可能である.

同様にジョイント i から見た子ジョイント j の相対位置 $\boldsymbol{p}_j^i$ は,

$$\boldsymbol{p}_j^i = \boldsymbol{p}_i^i + \sum_{k=i}^{j-1} \boldsymbol{R}_k^i \boldsymbol{p}_{k+1}^k \tag{3.52}$$

上式の関節回転角度 $\boldsymbol{q}_l$ による微分は, $\partial R_k^i / \partial q_l$ が $k \prec l$ のとき 0 であることから

$$\begin{aligned}\frac{\partial \boldsymbol{p}_j^i}{\partial q_l} &= [\boldsymbol{a}_l^i \times] \sum_{k=l}^{j-1} \boldsymbol{R}_k^i \boldsymbol{p}_{k+1}^k \\ &= \boldsymbol{a}_l^i \times (\boldsymbol{p}_j^i - \boldsymbol{p}_l^i) \end{aligned} \tag{3.53}$$

$\boldsymbol{p}_j^i$ の任意回微分は以下のようになる．ただし，$k_0 \cdots k_{n-1}$ $(i \prec k_0 \prec \cdots \prec k_{n-1} \prec j)$

$$\begin{aligned}\frac{\partial \boldsymbol{p}_j^i}{\partial \boldsymbol{q}_l} &= [\boldsymbol{a}_{k_0}^i \times] \cdots [\boldsymbol{a}_{k_{n-1}}^i \times] \sum_{k=k_{n-1}}^{j-1} \boldsymbol{R}_k^i \boldsymbol{p}_{k+1}^k \\ &= \boldsymbol{a}_{k_0}^i \times \cdots \boldsymbol{a}_{k_{n-1}}^i \times (\boldsymbol{p}_j^i - \boldsymbol{p}_{k_{n-1}}^i) \end{aligned} \tag{3.54}$$

これらは各関節の位置ベクトルを順運動学計算を用いて予め持っておくことで n 回の行列の掛け算で求めることが可能である.

3.3.6.2 空間に固定された点のヤコビ行列

式 3.53 で $\boldsymbol{p}_i^j$ が固定ベクトルであるとする．このとき式 3.53 の微分は式 3.54 と異なり以下のようになる.

$$\begin{aligned}&\frac{\partial \boldsymbol{a}_l^i \times (\boldsymbol{p}_j^i - \boldsymbol{p}_l^i)}{\partial q_s} \\ &= \frac{\partial \boldsymbol{a}_l^i}{\partial q_s} \times (\boldsymbol{p}_j^i - \boldsymbol{p}_l^i) - \boldsymbol{a}_l^i \times \left(\frac{\partial \boldsymbol{p}_j^i}{\partial q_s} - \frac{\partial \boldsymbol{p}_l^i}{\partial q_s} \right) \end{aligned} \tag{3.55}$$

ここで，$\boldsymbol{p}_i^j$ が定数ベクトルであることから，$\partial \boldsymbol{p}_i^j / \partial q_s$ は常に零である．$i \prec s \prec l \prec j$ とすると，式 3.51 と式 3.54 から以下が成り立つ．

$$\frac{\partial \boldsymbol{a}_l^i}{\partial q_s} = \boldsymbol{a}_s^i \times \boldsymbol{a}_l^i$$
$$\frac{\partial \boldsymbol{p}_l^i}{\partial q_s} = \boldsymbol{a}_s^i \times (\boldsymbol{p}_l^i - \boldsymbol{p}_s^i)$$

結果として，式 3.55 は以下のように計算できる．

$$\frac{\partial \boldsymbol{a}_l^i \times (\boldsymbol{p}_j^i - \boldsymbol{p}_l^i)}{\partial q_s}$$
$$= (\boldsymbol{a}_s^i \times \boldsymbol{a}_l^i) \times (\boldsymbol{p}_j^i - \boldsymbol{p}_l^i) - \boldsymbol{a}_l^i \times (\boldsymbol{a}_s^i \times (\boldsymbol{p}_l^i - \boldsymbol{p}_s^i)) \tag{3.56}$$

3.4　全身姿勢探索による片足支持に適した股関節間距離の決定

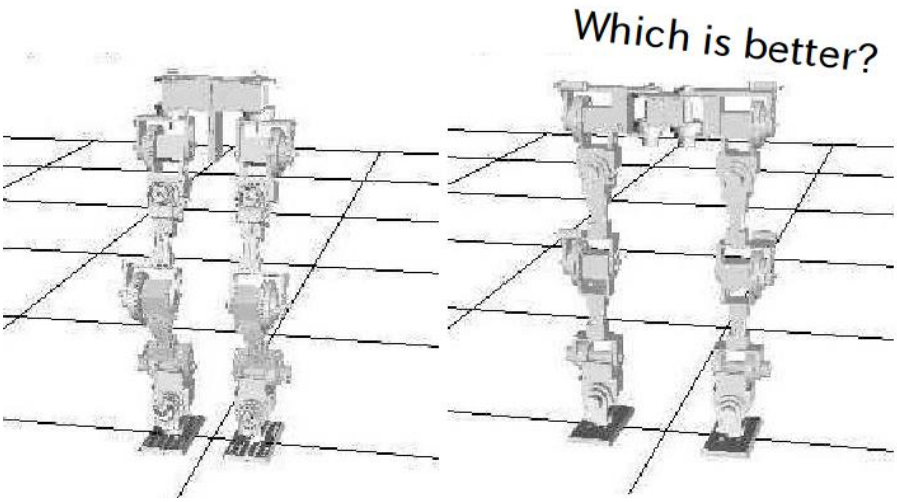
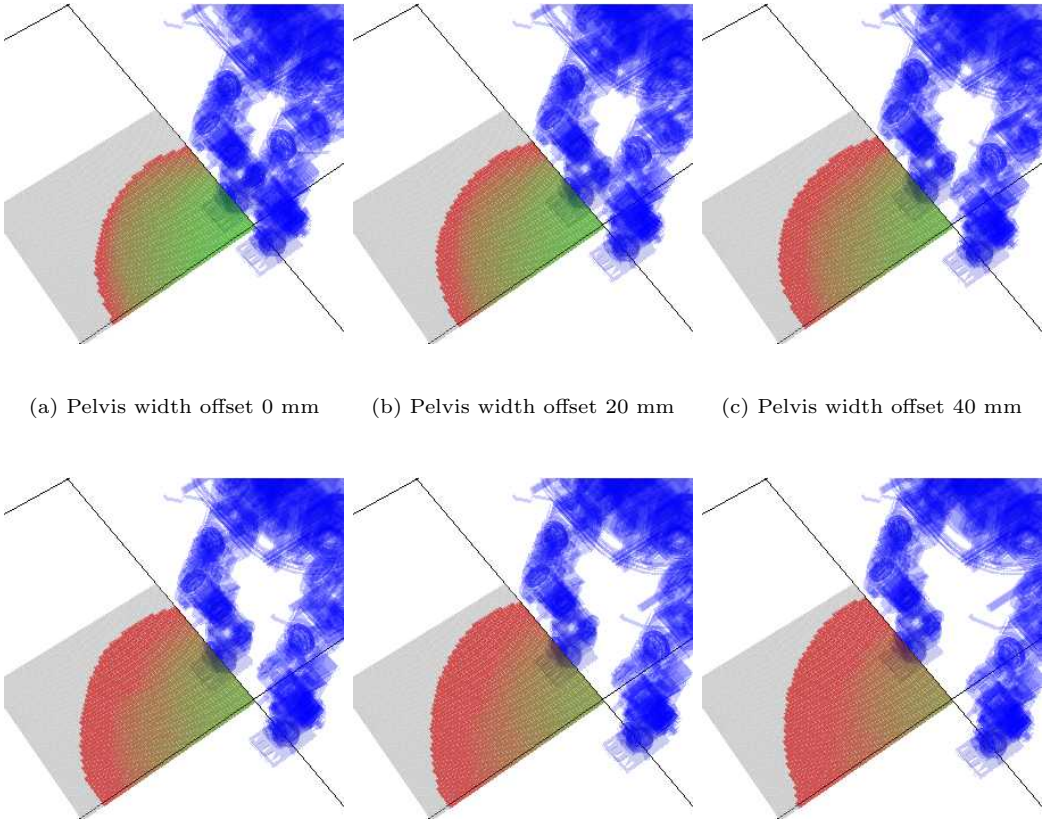


図 3.28. The distance between both legs largely change the walking motion and evaluation.

本章では高速計算可能な負荷トルク勾配の提案を行い，探索時間と探索成功率や評価値の関係を詳細に調べることで，与えられたモデルに対して高速確実に解を得られるソルバを実装した．これを身体環境モデルを変化させながら用いることで行動生成可能なモデルを獲得したり高評価な行動が可能となるようモデル探索したりといった応用が可能である．本章では，身体行動創成支援の例として片足支持に適した股関節間距離の決定問題を解く．本論文で開発した脚型ロボットは，身体設計段階で股関節間距離の異なる図 3.28 のような案を出し，省エネルギー歩行に適した左側の身体設計を採用している．省エネルギー歩行ではなるべく低負荷になるべく大きく前に足を出せることが必要である．前足の到達可能領域は固定された後ろ足を中心とする楕円形になると予想できる．したがって前に大きく足を出すためには後ろ足の正面付近に前足を置く必要がある．このとき広い股関節間距離は股関節のモーメントアームを大きくしてしまうため必要負荷トルクが大きくなり消費エネルギーも増大すると予想できる．



(a) Pelvis width offset 0 mm (b) Pelvis width offset 20 mm (c) Pelvis width offset 40 mm

(d) Pelvis width offset 60 mm (e) Pelvis width offset 80 mm (f) Pelvis width offset 100 mm

図 3.29. RMS (Root mean square) of joint torque of all joints are visualized with respect to pelvis width offset from 0mm to 100mm. Left foot is fixed in the origin, and right foot is placed in 600 mm × 500 mm square. Right leg is support leg and exterts all contact force to support whole body weight. Gray means there is no feasible solution, Red and green means there is some feasible solutions and the RMS of joint torque is visualized with RGB=(RMS, 1 - RMS, 0). Hense, color is green if RMS=0, color is red if RMS=1(100 Nm), and color is yellow if RMS=0.5. Long pelvis has large reachable reasion, but green area is small. In contrast, short pelvis has small reachable reasion, but green are is large.

以上の考察を検証するために図 3.29 に示すような探索実験を行った．股関節間距離を限界まで狭くした身体モデルパラメタから 0mm, 20mm, 40mm, 60mm, 80mm, 100mm と広げていき，左足を中心として右足を動かしながら右足を支持脚として関節負荷トルクを最小化した姿勢を探索しその最小化された関節負荷トルクを可視化している．可視化には関節負荷トルクを最大発揮トルクで割ったものの二乗平均平方根 (RMS: Root Mean Square) を用い，RGB = (RMS, 1-RMS, 0) に従い色をつけた．したがって最大負荷トルクの 50% を黄色としてそれより負荷が小さければ緑色，大きければ赤色となる．最大負荷トルクは 100 Nm である．色のついていない箇所では答えが一つも見つかっていない．図 3.29 を見てみると股関節間距離が広いほど色のついた実現可能領域は特に横側に広がっていることが確認できる．ま

た緑色の低負荷領域は狭まっており 100mm 広げたものではほとんどが赤色で中心付近にわずかに黄色領域を残すのみとなっている．

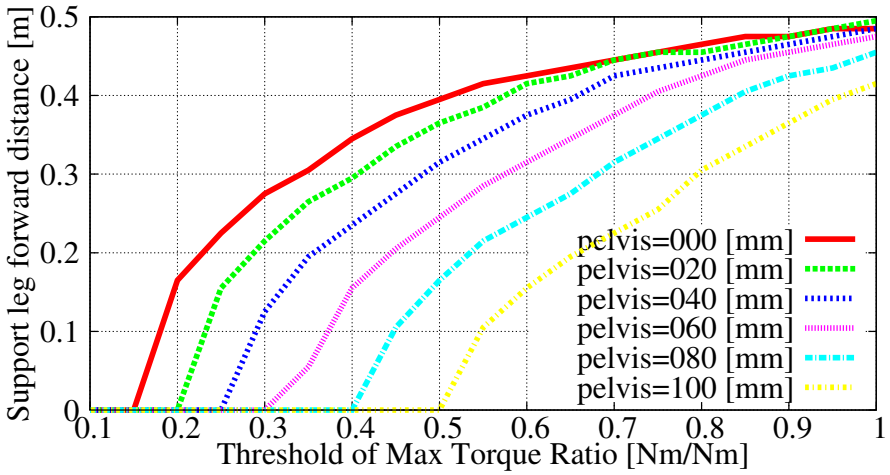


図 3.30. This graph shows the maximum support leg forwarding distance with respect to threshold of max joint torque. Vertical axis shows support leg forward distance, and horizontal axis shows threshold of max joint torque ratio. With threshold less than 0.5, short pelvis robot could move support leg forward largely.

図 3.30 は RMS の閾値を横軸にとり，その閾値を満たす範囲でもっとも足が前にだせる距離を縦軸にとったグラフである．股関節間距離が狭いほど曲線が上側に移動し低負荷で遠くに足がだせていることが確認できる．以上の結果から本論文で開発したロボットでは股関節間距離を限界まで狭くした身体設計になっている．

3.5　おわりに

本章では、全身自由度を用いた姿勢探索手法には計算時間の問題が以前残っているという考えから、計算速度の問題を解決するために動作生成の基礎となる逆運動学問題高速化を取り上げ，局所解を高速に脱出するための探索手法，関節負荷トルク上限といった力学的制約も考慮した低負荷姿勢探索手法について提案し、それを用いて片足支持に適した股関節間距離を探索した。

第 4 章

接触遷移行動探索による着座支持棒長さの決定

4.1 はじめに

本章では接触遷移行動探索手法とそれを用いた身体環境モデル空間での探索手法について扱い、身体モデルのパラメタ生成の例題として着座支持棒長さの決定問題を解く。また実体の身体環境モデルのパラメタ修正の例題としてセンサ情報から測定可能な摩擦係数の修正と運動モデルのパラメタ再生成により行動実現を行う。図 2.11 が解く問題のモデルと探索法をまとめたものである。支援システムにおけるロボット完成前の身体運動モデルパラメタ生成において身体モデルのパラメタ変更と運動モデルのパラメタ生成を交互に繰り返す例、ロボット完成後の実体モデルパラメタ修正において推定する実体身体モデルのパラメタがセンサにより測定できる場合の例を示す。本章で提案する接触遷移方式の探索機能も備える行動生成手法は接触を保ちながら動作するような一連の行動生成を可能とするものである。例えば立ち上がりや着座といった接触を保って移動する動作では、行動の過程で全身が複雑に動き続けるため少数の姿勢の確認だけでは行動が実現可能であるかを判断することができないという難しさがあり、本支援システムがそのような問題を扱えることを示す意義がある。また摩擦は滑り動作において設計と実体の間でずれが大きくなりやすいことが知られており、摩擦を扱えることを示す点にも意義がある。

探索空間をロボットの全身関節角度空間にしたとしても、探索の条件によってはその自由度が制限されることになる。特に接触状態は離散的に変化し、また現実の人間環境が一般に滑らかでない要素を含む。歩行では一歩目二歩目はあっても 1.5 歩目はないし、人工物の多くはエッジのある形状である。このことから動作生成手法では、接触をあらかじめ与えられたものとして扱ったり、その探索を分けて行うといった工夫がしばしば行われる。しかし接触を別に考えて探索しても全ての動作が探索できるわけではないことを [86] では示しており、接触の仕方に滑りや転がりといった選択肢を与える方法を提案している。以下の章ではまずその手法について述べる。

接触状態に仮定を置かず探索を行うもう一つの方法としては、接触を分けて考えることな

く、汎用の動力学シミュレーションを用いてシミュレーションの出力する全結果のなかで動作を探索することでロボットの持つ自由度を損なわない行動生成が可能である。以下の章では、次に本研究で用いる動力学シミュレーションの構成についてまとめ、シミュレーションを用いた歩行動作生成を行うことで接触状態に滑りや転がりが含まれていることを見る。

4.2　複数の接触遷移方式探索機能を有する行動生成法

本 4.2 章の内容は [36] に掲載された内容であり、図と文章で重複がある。

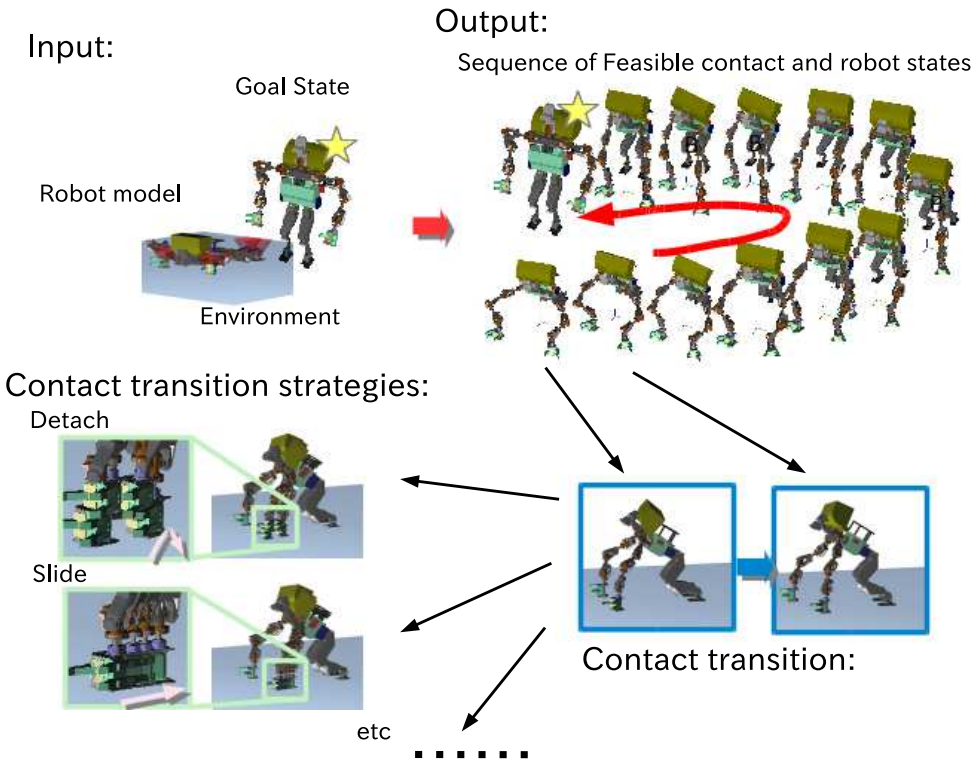


図 4.1. Our goal is to generate a sequence of feasible robot motions and contact states with given robot model, enviroment model, and goal state. Our planner is novel in that it switches multiple contact transition strategies according to task objective. (reprinted from [36])

本章では接触遷移を含むモデルベース行動生成問題を扱い，複数の接触遷移方式を計算量の増加を抑えて組み合わせる手法を提案する．接触遷移方式として具体的には，歩行に代表される Detach 接触遷移と，Slide 接触遷移の統合を扱い，動力学シミュレータを用いた立ち上がり実験，実等身大ヒューマノイドロボットを用いた滑り着座行動実験を行う．本手法の入力は，ロボットの，初期姿勢，初期接触状態，終了条件，接触可能なリンクおよびその摩擦係数，接触面形状，環境モデルと接触可能リンクが環境中で接触した時の位置姿勢の候補（後述 4.2.2.1.1 章）とする．また出力は，初期状態から終了状態に至るまでの接触遷移を含む実現可

能な動作列である（図 4.1）．接触状態を切り替える方法として過去の研究では，接触を一旦取り除いてからリンクを移動させ再度接触を行う Detach 接触遷移が主に扱われてきたが，その他にも，接触を保ったまま連続的に変化させる滑り（Slide 遷移）や，転がり（Rotate 遷移）といった複数の方式が存在する．これらを適切に切り替えながら行動を生成することができれば，行動の評価値改善と求解可能範囲の拡大が期待できる．例えば椅子に座る行動を生成する問題では，まず椅子のへりに臀部リンクを接触させてから滑らせることで深く座るといった方法が有効である．

先行研究と比較した本章の特徴を以下に述べる．人型でない多関節ロボットを用い一般の崖登り問題を解いた例として [27][28] の研究では ‘contact-before-motion’ という方法が提唱されている．離散的な問題である接触状態（contact）の探索と，連続的な問題である動作（motion）の探索を分けて行うことで高速に問題を解くことを意図した方法である．‘contact-before-motion’ と同様に，接触と動作の探索を分けて解く考え方を人型のロボットに適用し，関節負荷や幾何拘束といった非線形探索問題を解くことにより一般性の高い全身行動生成を実現した例としては Hauser[29][72] らの研究や，Escande[31][32]，Bouyamane[33][34] らの研究が挙げられ，脚立登りや車への乗り込みと言った複雑な接触変化を必要とする動作を生成することが可能となっている．‘contact-before-motion’ の他にも，接触平面との距離を評価関数に組み込み数値最適化手法を用いることで動作と接触の探索を同時に行う方法 [41] や，その距離関数を任意の環境モデルに対し滑らかに近似することで一般性を高める研究もある [40] が，非凸な環境，例えば脚立登り行動 [48][70] [75] では，足場と手すりに囲まれた何もない空間が多くの局所解を生むことになるため，接触状態の探索を分ける方法が有効である．

‘contact-before-motion’ をそのまま用いて滑り等複数の接触状態遷移方式を考慮した探索を行うには，動作の探索において相補性条件を用いて Detach 遷移と Slide 遷移を切り替える方法が考えられるが，それには二つの問題がある．一つ目の問題は，動作探索が多リンク系の運動学や力学を考慮した非線形最適化問題であり，行動生成全体における計算時間のボトルネックとなるため，相補性条件を追加するといった複雑化は望ましくないという点である．また，Detach/Slide 遷移に限らない一般の遷移を対象とする場合には，現実的な時間で求解が可能な遷移方式数の規模についての調査も必要である．もう一つの問題は，ロボットの摩擦係数や制御系の構成によっては，Detach 遷移と その他で優先順位をつけて解く要求が生じうるが，そのための定式化は明らかではないという点である．また，仮に評価関数の重み等を用いて Detach 遷移を優先的に探索することができたとしても，接触を伴う動作探索では，探索結果に接触力や関節負荷トルクといった様々な項が非線形に関わるため，直感的な操作は困難であると予想される．本章では，接触状態選択と動作探索という二つの問題を分けて解くという点は ‘contact-before-motion’ と同様であるが，それに加えて，Detach 遷移に限らない複数の接触遷移方式の切り替えを探索する問題を第三の問題として分割して解く点が異なる．本章では ‘planner-before-motions’ と名付ける新たなフレームワークを提案する．‘contact-before-motion’ が動作の探索の前に接触を探索する方法であるのに対し，‘planner-before-motions’ は ‘contact-before-motion’ に基づき接触と動作を生成する動作生成器（planner）をまず選択し，その出力動作を組み合わせて全体の動作列（motions）をつく

る．本手法は，個々の動作探索を複雑化することなく，任意の動作生成器を組み合わせることができ，動作生成器の優先度を関節負荷トルクや接触力といった条件と分離して，単一のパラメタを調整することによって直感的に変更できるという点で優れている．動作探索の選択子が増えることによる計算時間増大の問題については，探索アルゴリズムの評価関数に工夫を行い高速化をはかった．最後に，複数の接触遷移方式を切り替えることで探索の幅を広げることができ，実現可能行動の拡充と，探索される行動の評価値向上が起こることを，動力学シミュレーションと実等身大ヒューマノイドを用いた実験で検証する．

4.2.1　複数の接触遷移方式探索機能を有する行動生成手法の構成

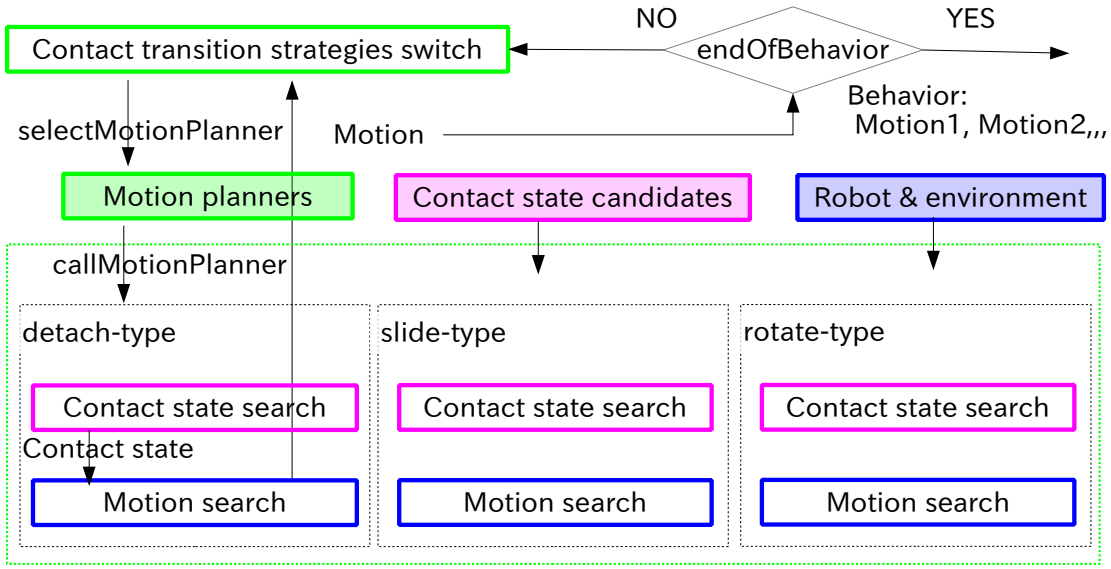


図 4.2. Overview of ‘planner-before-motions’ framework which uses given candidates of contact states and models of robot and environment, and which selects motion planner and generates motion while ‘endOfBehavior’ function returns false. (reprinted from [36])

‘planner-before-motions’ は，環境やロボットリンク系のモデル，接触状態の候補，目標状態が与えられた状態でその目標状態を達成するための接触と動作の列を生成する探索器である．本章では，‘動作’ という単語を接触状態の遷移を達成するためのロボット状態の列という意味で用いる．‘行動’ という単語を目標状態を達成するための ‘動作’ の列という意味として用いる．図 4.2 に ‘planner-before-motions’ の模式図を示す．‘Motion Planners’ で示す構成要素は，接触状態の探索（‘Contact state search’）と動作の探索（‘Motion search’）を行う動作生成器を複数もつ．これらの動作生成器は ‘contact-before-motion’ に基づくものである．我々のフレームワークは，‘Motion Planners’ が複数の動作生成器を持ち，それぞれを評価関数に基づき切り替えながら動作を生成し，目標状態が達成されたところで探索を終了，行動を出力する．‘planner-before-motions’ の手続き（Algorithm 2）は [32] と同様に最良優先探索

をベースとしているが，探索幅制限を持ち，isGoodEnough 関数が真を返す場合は深さ優先探索のように即座に次の探索に移行するという特徴がある．また，接触状態の評価には，探索時点の情報のみを用いる山登り法を用いている（後述 4.2.2.1.2 章）．したがって，ダイクストラ法といった最適解を探索する方法と異なり行動全体の評価に対し最適であることは保証しない．また，単純な最良優先探索を用いる場合よりも評価値が劣ったものになる可能性がある．一方で，十分優良な解がある場合はより高速に実行可能解を探索することが期待できる．このような実装にした理由は，実機での実験を考えた場合には，過度に最適化された解を求めることよりも，実現が十分可能な範囲で行動を高速に生成することが重要と判断したためである．実機で探索された行動を実行する場合，その行動が複雑な接触遷移を含むほど正確な実行は困難になることが予想される．そのような場合には高速な再探索が複数回必要である．また，摩擦係数といった計測が困難な未知のパラメタを推定しながら動作を行う状況でも，パラメタを更新しながらの再探索が必要になる（4.2.4 章）．以上の理由により，本章では計算速度に重きをおいた実装を行った．

各アルゴリズムの概要を以下に述べる．Algorithm 2 は ‘planner-before-motions’ の擬似コードである．はじめに，‘planner-before-motions’ は selectMotionPlanner 関数を用いて動作生成器を選択する．selectMotionPlanner 関数は評価関数に基づき動作生成器を選択する手続きである．また，selectMotionPlanner 関数は以前探索したことのある動作生成器と接触状態の組みを $\mathcal{B}$ に蓄えておき無限ループを防ぐ．次に，callMotionPlanner 関数が接触状態の候補から次の接触状態を選択し動作を探索する．callMotionPlanner 関数で行われる手続きについては続く 4.2.2 章でより詳しい説明を行う．最後に，isGoodEnoughMotion 関数が生成された動作を評価し，評価値が低ければ，もう一度動作生成器の選択を行う．高ければ次の行動の探索に移る．これらの手続きは loopExceeded 関数が真を返すまで続く．loopExceeded 関数は計算時間や展開されたノードの数が上限を超えないか，接触状態候補や動作生成器のなかに未探索のものが残っているかをチェックする手続きである．

Algorithm 3 は，callMotionPlanner 関数の手続きを表す擬似コードである．callMotionPlanner 関数は ‘contact-before-motion’ の考えに基づき，評価関数に応じて次の接触状態を選択し（selectContactState 関数，Algorithm 4），動作を生成する（generateMotion 関数，Algorithm 5）．各動作生成器は未探索の接触状態の候補（$\mathcal{C}$）を持ち，探索済みの接触状態候補 $\boldsymbol{c}^{'}$ は，removeContactState 関数により削除される．

genWholebodyBehaviorTmp 関数は，isGoodEnoughMotion 関数が偽を返した動作を蓄えておく値 $\mathcal{S}_{tmp}$ の中から実現可能な動作を評価の高い順にソートしてから選択し次の動作の探索へと移行するための関数である．

4.2.2 複数の接触遷移方式探索機能を有する行動生成手法の実装

4.2.2.1 接触状態選択について

‘contact-before-motion’ では，まず接触状態を選択しそれを満たすような動作を探索するため，最終的に出力される動作の探索性能・計算速度は，実現可能な動作を生成しうる接触状

Algorithm 2 Pseudo code of behavior generator involving contact transition strategies switching (reprinted from [36].)

Require: $\mathcal{M}$, $\mathcal{S}$, $\mathcal{S}_{tmp}$, s, $\mathcal{B}$
  $\mathcal{M}$: set of motion planners.
  $\mathcal{S}$: set of successful robot and contact states.
  $\mathcal{S}_{tmp}$: tmp set of robot and contact states.
  s: current robot and contact state.
  $\mathcal{B}$: visited set of motion planners and contact states.
Variable: .
  m: selected motion planner.
  $s^{'}$: robot and contact state.
  $\mathcal{S}^{'}$, $\mathcal{S}^{''}$: set of robot and contact states.
Procedure: genWholebodyBehavior
  if endOfBehavior($\mathcal{S}$) **then**
    return $\mathcal{S}$
  else if loopExceeded($\mathcal{S}_{tmp}$, $\mathcal{M}$) **then**
    return genWholebodyBehaviorTmp($\mathcal{M}$,$\mathcal{S}$,$\mathcal{S}_{tmp}$, $\mathcal{B}$)
  else
    $m \leftarrow$ selectMotionPlanner($\mathcal{M}$,s, $\mathcal{B}$)
    ($s^{'}$, $\mathcal{S}^{'}$) $\leftarrow$ callMotionPlanner(m, s)
    if isGoodEnoughMotion($\mathcal{S}^{'}$) **then**
      $\mathcal{S}^{''} \leftarrow$ genWholebodyBehavior($\mathcal{M}$,[$\mathcal{S}^{'}\mathcal{S}$],$\emptyset$,$s^{'}$, $\mathcal{B}$)
      if endOfBehavior($\mathcal{S}^{''}$) **then**
        return $\mathcal{S}^{''}$
      end if
    else
      push(($s^{'}$, $\mathcal{S}^{'}$), $\mathcal{S}_{tmp}$)
    end if
    return genWholebodyBehavior($\mathcal{M}$,$\mathcal{S}$,$\mathcal{S}_{tmp}$,s, $\mathcal{B}$)
  end if

態の選択を，動作の探索なしに如何に精度よく行えるかにかかっている．この問題に対して [32] では，目標状態へと近づくように接触を選択することに加え，接触状態候補ごとに姿勢探索問題を解くことで，実現可能な解が一つも存在しないような接触状態を探索から取り除く工夫がなされている．本章も目標に近づくほど高い評価を与える評価関数を用いる点は同じであるが，高速化のために接触候補ごとの動作探索は行わず，代わりに，ロボットの負荷を減少させるような接触を選択するための項を追加で考慮する．以下では，まず，具体的な接触状態選択のための評価関数について述べる．次に，接触状態の探索アルゴリズムと計算時間の評価を

Algorithm 3 Pseudo code of callMotionPlanner. This function is based on the 'contact-before-motion' approach (reprinted from [36])

Require: m, s

m: selected motion planner.

s: current robot and contact state.

Variable: .

$\boldsymbol{c}^{'}$: set of selected contact state.

$s^{'}$: generated robot and contact state.

$\mathcal{S}^{'}$: set of robot and contact states.

$\mathcal{C}$: All candidates of contact states.

Procedure: callMotionPlanner

$\mathcal{C} \leftarrow$ getCotnactState(m)

$\boldsymbol{c}^{'} \leftarrow$ selectContactState(m, s, $\mathcal{C}$)

removeCotnactState(m, $\boldsymbol{c}^{'}$)

$(s^{'}, \mathcal{S}^{'}) \leftarrow$ generateMotion(m, s, $\boldsymbol{c}^{'}$)

return $(s^{'}, \mathcal{S}^{'})$

行う．

4.2.2.1.1 接触リンク，目標位置姿勢，接触反力にかかる制約からなる接触状態の表現

本章では，接触状態 $c = (l, \boldsymbol{p}_l, \boldsymbol{r}_l, \boldsymbol{n}_l, \mathcal{F}_l)$ を，接触を行うロボットのリンクごとに一意に決まる識別子 l，接触面上でリンクがとるべき位置 $\boldsymbol{p}_l$，姿勢 $\boldsymbol{r}_l$，接触面の法線ベクトル $\boldsymbol{n}_l$，接触を保つための摩擦や回転を防ぐために接触毎に定義される六次元の接触反力 $\boldsymbol{F}_l$ が満たすべき条件 $\mathcal{F}_l$ で表現する．$\mathcal{F}_l$ としては，摩擦係数と接触面の辺の長さを用いた一次不等式を用いた．リンクの識別子 l は，接触状態の評価においては，一つのリンクばかりが選択され続けることを防ぐ目的で用いられる．動作探索においては位置姿勢を保つ接触リンクを識別する目的で用いる．

4.2.2.1.2 負荷指標を減少させ目標状態に近づくことを目的とする接触状態選択の評価関数

List4.1. Considered conditions for contact state selection

a) 目標状態へと近づく接触を選ぶ項 （h_0）

b) 接触の移動量をなるべく抑える項 （h_1）

c) 負荷を低減させる反力を得られる接触を選ぶ項 （h_2）

d) 大きな負荷を受けている接触を移動させない項 （h_3）

e) 移動リンクごとにバイアスをかける項 （k）

接触選択の評価関数では，List4.1 に示す項を式（4.1）の形式で考慮する．

$$\begin{aligned} \underset{c=(l,\boldsymbol{p},\boldsymbol{r},\boldsymbol{n},\mathcal{F})\in\mathcal{C}}{\arg\max} \quad &: H \\ H &= k\cdot\exp(\alpha h_0-\beta h_1+\gamma(h_2-h_3)) \end{aligned} \tag{4.1}$$

これは，Algorithm 4（後述）における，getHighestEvaluatedContactState 関数である．getHighestEvaluatedContactState 関数は，与えられた接触状態候補 $\mathcal{C}$ の中から評価関数の値 H が最大となるものを出力する．k, h_0, h_1, h_2, h_3 は評価対象である接触状態 c に依存する値，α, β, γ は正のゲインである．後の実験では α, β, γ を発見的方法により決定している．終了状態が初期状態の近くにある場合には，α を大きく，β を小さくすることで greedy な探索を行うことができる．逆に α を小さく，β を大きくすることで，終了状態にそれほど近づかない接触候補も広く探索することができる．以下に，それぞれの項について述べる．評価関数 H では，目標状態に近づくような探索を行うことに加え，接触反力や関節負荷トルクといった負荷指標を低減することを目指す（List4.1.c，4.1.d）．そのための本章のアプローチは，接触反力とその滑り・転がり制約との距離と，関節負荷トルクとその出力制約との距離を，それぞれ接触反力に近似し（負荷指標ベクトル），負荷指標ベクトル方向に力を発揮できる面を選択することで負荷を低減する．例えば滑りそうな接触がある場合には，その滑りと逆方向に負荷指標ベクトルが向くため，滑りと反対方向に反力を発揮できる面が選択される．立ち姿勢にて脚が大きな負荷トルクを要する場合には，その接触反力としての表現が鉛直上向きのベクトルとなることで体を支える接触が選択される．同時刻の各リンクの接触をそれぞれ下付き数字を用いて区別することとし，もしリンク l がすでに接触している場合には，その接触の下付き数字として 0 を用いて区別することとする．i 番目のリンク接触について，接触反力反モーメントベクトル $\boldsymbol{w}_i$ の各要素を，$\mathcal{F}_i$ を満たす実現可能な最大値ベクトル $\boldsymbol{w}_i^{max}$ の各要素で割ったベクトルを $\boldsymbol{w}_i/\boldsymbol{w}_i^{max}$ と定義する．接触リンクからベースリンクまでの各ジョイントの関節負荷トルクベクトル $\boldsymbol{\tau}_i$ の各要素を最大トルクベクトル $\boldsymbol{\tau}_i^{max}$ の各要素で割ったベクトルを $\boldsymbol{\tau}_i/\boldsymbol{\tau}_i^{max}$ と定義する．接触リンクからベースリンクまでの各ジョイントのヤコビ行列 $\boldsymbol{J}_i$ を用いて，負荷指標ベクトルを以下のように定義する．

$$\boldsymbol{F}_i^{BRLV} = \boldsymbol{w}_i/\boldsymbol{w}_i^{max} + \boldsymbol{J}_i^{\#}\boldsymbol{\tau}_i/\boldsymbol{\tau}_i^{max} \tag{4.2}$$

$\boldsymbol{F}_i^{BRLV}$ 方向に反力を発揮できる接触は，接触面の法線と負荷指標ベクトルの内積を最大化するようなものを探索することで選択することができる．面の位置ベクトル $\boldsymbol{p}$，法線ベクトル $\boldsymbol{n}$，各接触点の位置ベクトル $\boldsymbol{p}_i$，単位行列 $\boldsymbol{E}$ を用いて以下を最大化する．

$$h(\boldsymbol{p},\boldsymbol{n}) = \begin{pmatrix} \boldsymbol{n} \\ \boldsymbol{p}\times\boldsymbol{n} \end{pmatrix}^T \sum\nolimits_i \begin{pmatrix} \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{p}_i\times & \boldsymbol{E} \end{pmatrix} \boldsymbol{F}_i^{BRLV} \tag{4.3}$$

以上を用いて，h_2, h_3 を次のように定義する．

$$h_2 = h(\boldsymbol{p},\boldsymbol{n}) \tag{4.4}$$

$$h_3 = \begin{cases} h(\boldsymbol{p}_0,\boldsymbol{n}_0) & \text{hasContact}(l) \\ 0 & \text{else} \end{cases} \tag{4.5}$$

h_3 は移動するリンクがもしも接触していた場合に，その接触を移動させることによって負荷指標ベクトル方向に発揮できる力がどの程度減少するかを表し，大きな力を発揮しているリンクの接触を移動させないための項である．したがって，そのリンクが現在接触していない場合は h_3 は 0 である．$\boldsymbol{p}_0$ はリンク l がすでに接触している場合の接触リンク位置，$\boldsymbol{n}_0$ はその接触面法線ベクトルである．

h_0 は目標状態に近づく接触状態を選択するための項であり，目標方向が決まっているような問題，例えば登り動作を生成するような場合には目標方向ベクトルとして鉛直上向きの単位ベクトルを用い，目標方向ベクトルと，接触リンクの現在の位置から評価対象である接触状態の位置の相対ベクトルの内積を h_0 として用いる．すなわち，目標方向ベクトル $\boldsymbol{d}$，移動リンクの現在の位置 $\boldsymbol{p}_0$，評価対象接触状態 c の位置 $\boldsymbol{p}$ を用いて，

$$h_0 = \boldsymbol{d}^T(\boldsymbol{p} - \boldsymbol{p}_0) \tag{4.6}$$

を計算する．特定の位置に移動することを目標とする場合には，評価対象である接触状態の位置と目標位置の距離の -1 倍を h_0 として用いる．すなわち，評価対象接触状態 c の位置 $\boldsymbol{p}$，目標状態の位置 $\boldsymbol{p}_g$ を用いて

$$h_0 = -\|\boldsymbol{p}_g - \boldsymbol{p}\| \tag{4.7}$$

を計算する．h_1 は遠すぎる接触状態を選択しないための項であり，現在の接触リンクの位置姿勢と目標のそれの距離を用いた．k はリンクと動作生成器毎に選択をバイアスするための項である．直近で選択された接触状態列の中に，評価対象の移動リンクが多く含まれているほど小さな値とすることで，特定のリンクばかりが移動することを防ぐ．また，以下の実験では，Detach 遷移を優先的に探索するために，Detach 遷移の k を Slide 遷移のそれの 1.1 倍にして用いる．そのため，もし一つの接触状態候補への遷移が Detach 遷移と Slide 遷移の両方で可能な場合，常に Detach 遷移が先に選択されることになる．また，Detach 遷移の選択のされ易さを調整するためには，この k を単純に大きくすればよく直感的である．なお，exp を用いている理由は評価値を正の値に補正するためである．

4.2.2.2 接触状態選択の探索アルゴリズム

接触状態の選択（selectContactState 関数）には，動作生成器ごとに定義する評価関数に基づき全ての接触状態を虱潰しに探索し最大の評価を与える接触状態を出力する手続きを用いた（Algorithm 4）．removeUnReachableContact 関数は，動作生成器ごとに実現不能な接触状態を取り除くために用いる．Detach 遷移では接触するリンクから体幹までの総リンク長さよりも遠くにある接触状態候補と，接触候補の位置がすでに接触しているリンクの位置と干渉するものを取り除く．Slide 遷移ではそれらに加えて，現在の接触面と目標のそれが同一面にない場合を取り除く．その後，接触遷移が可能な接触状態候補 $\mathcal{C}'$ の中から，最大の評価を与える接触状態 c が getHighestEvaluatedContactState 関数により選択され，現在の接触状態 $\boldsymbol{c}$ のなかから同じ接触リンクを持つものとスワップ（swapContactWithSameLink 関数）する．なお，この手続きは各接触リンクの接触状態リストの中から一つの接触状態のみを切り替

えるものである．以下の実験では接触が一つずつしか変化しない動作のみを扱うが，複数の接触状態変化を扱う場合は Algorithm 4 に追加の拡張が必要である．

Algorithm 4 Pseudo code of selectContactState (reprinted from [36]).

Require: m, s, $\mathcal{C}$

m: selected motion planner.

s: current robot and contact state.

$\mathcal{C}$: All candidates of contact states.

Variable: .

$\mathcal{C}'$: reachable candidates of contact states.

$\boldsymbol{c}$: contact states.

c: contact state with the highest evaluation.

Procedure: selectContactState

$\mathcal{C}' \leftarrow$ removeUnReachableContact(m, s, $\mathcal{C}$)

$c \leftarrow$ getHighestEvaluatedContactState(m, s, $\mathcal{C}'$)

$\boldsymbol{c} \leftarrow$ getContactStates(s)

$\boldsymbol{c} \leftarrow$ swapContactWithSameLink($\boldsymbol{c}$, c)

return $\boldsymbol{c}$

虱潰し探索で十分である理由は，接触状態の評価関数として前章で述べた評価関数（H）を用いたため，動作探索の計算コストに比べ十分無視できるからである．実際，euslisp[43] 処理系と，計算機として Intel Core i7-3520M CPU 2.90GHz を用いて 1000 の接触状態を評価する実験を行ったところ，計算に要した時間は 0.169 秒であった．これはのちに示す動作生成器の計算にかかる時間が数秒から数十秒であることと比べて無視できる．

4.2.2.3　動作生成について

力や関節負荷トルクの条件を考慮しつつ動作を生成する方法については多くの先行研究 [16] [69] がある．本章では，Detach 遷移と Slide 遷移動作を探索するアルゴリズムとして，静的な条件で四姿勢の経由点を探索し，それぞれをスプライン関数で接続するアルゴリズムを用いた．本章のアルゴリズムには以下の課題が残されている．

- 静的な経由姿勢を通過するため動的に接触状態を切り替えるような動作は探索できない．
- 経由姿勢間の接続時間は固定値（５秒）を用いている．
- 経由点を四姿勢に限定しているため Detach 遷移の途中で Slide 遷移に切り替えるような行動を生成できない．

本章の趣旨は複数の接触遷移方式を統合することにあり，動作生成アルゴリズムを本章の内容に限定するものではないが，後に示す着座行動，立ち上がり行動生成など，多くの応用が可

能であることが本章では確認された．より動的な動作生成器への拡張法としては，B-spline 関数を用い動作の軌道関数を最適化する手法 [16]，経由姿勢の接続時間を自動的に探索する手法としては [69] がある．

4.2.2.3.1 接触反力と関節負荷の条件を考慮した実現可能な動作の表現

ロボットの全身行動を実現可能なロボットの状態の列として表現する．実現可能とは，リンク系が接触拘束を満たすこと，リンク系と環境との干渉が起こらないこと，運動方程式を満たすこと，関節負荷トルクがモータの関節負荷トルク制約を満たすこと，接触反力が滑りや回転を起こさない（起こす） こととした（List4.2）．

List4.2. Conditions of feasible robot's state

a) 接触リンクが位置姿勢を満たす．
b) ロボットリンクと環境が接触しない．
c) 多リンク系の運動方程式を満たす．
d) 関節負荷トルクの上限を満たす．
e) 接触反力が滑り・転がりを起こさない（起こす）．

4.2.2.3.2 四段階接触状態遷移に基づく動作生成法

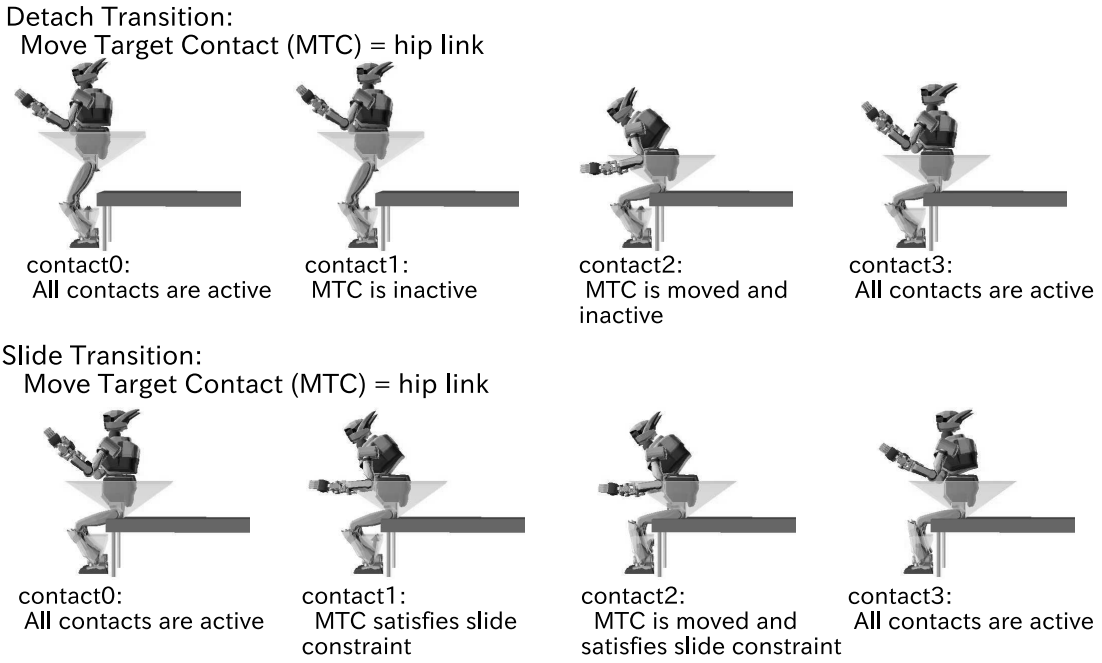


図 4.3. Detach/Slide transitions with 4 contact states.(reprinted from [36])

現在の接触状態から目標の接触状態へと遷移するための動作を，静歩行のように四段

階に接触状態を遷移させることで生成する（図 4.3）．四つの段階をそれぞれ，contact0, contact1，contact2，contact3 と呼ぶことにする．静歩行では，重心が両足中央に位置する両足接地（contact0），支持脚に重心を移動（contact1），重心を支持脚に保ったまま遊脚を移動（contact2），重心を両足中央に移動（contact4）となる．Detach 接触遷移では，contact0 が初期姿勢で全接触が保たれる．contact1 では移動する接触の位置姿勢のみ保ち接触力を 0 とする，contact2 は 接触を移動させて位置姿勢を保ったまま接触力を 0 とする，contact3 は再び全接触が保たれる．Slide 接触遷移でも同様の手続きを用いたが，contact1，contact2 で移動する接触の反力を 0 とはせずに次章の滑り遷移時の摩擦制約，式（4.19）を満たすように探索を行った．

Algorithm 5 に四段階接触状態遷移に基づく動作生成アルゴリズムを示す．現在の接触状態 $\boldsymbol{c}_0$ から目標の接触状態 $\boldsymbol{c}$ へと移動する動作を，四つの静的なロボット状態 s_0, s_1, s_2, s_3 を探索し，それを動的に接続することで生成する手続きである．それぞれの静的なロボット状態は robotStateOptimize 関数を用いて List4.2 の条件を満たすように探索される．探索アルゴリズムについては勾配法に基づく最適化手続き [48] [35] を用いた．robotStateOptimize 関数の第三引数は，位置姿勢を保つが力を発揮できない接触状態であり，現在の接触状態 $\boldsymbol{c}_0$ と目標接触状態 $\boldsymbol{c}$ の差集合（setDifference 関数）を用いる．探索された静的なロボット状態とそれらをつないで作られた状態軌道は，逆運動学を解くことで環境との干渉を回避するよう補正される（generateTrajectory 関数）．その際，Slide 遷移するリンクの位置は接触面上を直線で移動するよう計算し，Detach 遷移については接触面から 80 mm の高さを通過する山なりのスプライン軌道にそって計算した．最後に constraintsSatisfiedTrajectory 関数により，速度加速度を考慮した List4.2 の条件を満たすような接触反力の軌道を探索し，以上すべての探索に成功した動作軌道を generateMotion 関数の出力とする．

4.2.2.3.3　constraintsSatisfiedTrajectory 関数

前章で用いた constraintsSatisfiedTrajectory 関数について補足する．関節角度の軌道が与えられた時，接触力・モーメント軌道を摩擦やトルク上限を考慮しつつ最適化する問題は二次計画問題として定式化できることをしめす．動作軌道を $q(t) = \sum_{j=0}^{M-1} p_j b_j(t)$ のように有限個の基底関数の重み付けで表現し，パラメタ p_j の空間で探索する手法がしばしば用いられる．基底関数として B スプラインを用いた先行例としては [16] がある．定義域 $x \in [x_{min}, x_{max}]$ の B スプライン関数 $b_{j,n}$ の定義を以下にしめす．N は B スプライン多項式の最大次数，M は $M > N$ を満たす B スプライン基底関数の自由度である．また小文字 n, j は $n \leq N$，$j \leq M$

Algorithm 5 Pseudo code of generateMotion (reprinted from [36]).

Require: m, s_0, $\boldsymbol{c}$

m: selected motion planner.

s_0: current robot and contact state.

$\boldsymbol{c}$: target contact states.

Variable: .

$\boldsymbol{c}_0$: current contact states.

$\boldsymbol{c}_{mv}$: contact states without contact wrench.

s_1, s_2, s_3: robot states and contact states in contact1-3.

$\mathcal{S}$: robot states trajectory.

Procedure: generateMotion

$\boldsymbol{c}_0 \leftarrow$ getContactStates(s_0)
$\boldsymbol{c}_{mv} \leftarrow$ setDifference($\boldsymbol{c}_0$, $\boldsymbol{c}$)
$s_1 \leftarrow$ robotStateOptimize(m, $\boldsymbol{c}_0$, $\boldsymbol{c}_{mv}$)
if not constraintsSatisfied(s_1) **then**
  return $\emptyset$
end if
$\boldsymbol{c}_{mv} \leftarrow$ setDifference($\boldsymbol{c}$, $\boldsymbol{c}_0$)
$s_2 \leftarrow$ robotStateOptimize(m, $\boldsymbol{c}$, $\boldsymbol{c}_{mv}$)
if not constraintsSatisfied(s_2) **then**
  return $\emptyset$
end if
$s_3 \leftarrow$ robotStateOptimize(m, $\boldsymbol{c}$, $\emptyset$)
if not constraintsSatisfied(s_3) **then**
  return $\emptyset$
end if
$\mathcal{S} \leftarrow$ generateTrajectory(s_3, s_2, s_1, s_0)
if not constraintsSatisfiedTrajectory($\mathcal{S}$) **then**
  return $\emptyset$
end if
return (s_3, $\mathcal{S}$)

を満たす.

$$b_{j,0}(x) = \begin{cases} 1 & (x_j \leq x < x_{j+n+1}) \\ 0 & (otherwise) \end{cases} \tag{4.8}$$

$$b_{j,n}(x) = \frac{1}{Nh}\{(x - x_j)\ b_{j,n-1}(x) + (x_{j+n+1} - x)\ b_{j+1,n-1}(x)\} \tag{4.9}$$

$$h = \frac{x_{max} - x_{min}}{M - N} \tag{4.10}$$

$$x_j = hj + \frac{Mx_{min} - Nx_{max}}{M - N} \tag{4.11}$$

B スプライン基底関数は以下のような凸包性を持つ.

$$\forall x; \sum_{j=0}^{M-1} b_j(x) = 1 \tag{4.12}$$

$$\forall x; 0 \leq b_j(x) \leq 1 \tag{4.13}$$

したがって，B スプライン基底関数で表現される軌道 $q(t) = \sum_{j=0}^{M-1} p_j b_j(t)$ は以下を満たす.

$$\forall j \quad p_j \leq a \Rightarrow \forall t \quad q(t) \leq a \tag{4.14}$$

式（4.14）より，軌道全体にかかる等式・不等式拘束を有限個のパラメタに対する拘束として表現できる.

摩擦・転倒のモデルを式（4.15）のように，各接触力・モーメント F_i の一次の拘束として与える [48].

$$-\left(M_i^- \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \ \leq \ \boldsymbol{F}_i \ \leq \ \left(M_i^+ \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \tag{4.15}$$

$M_i^{\pm}$ は摩擦係数や接触面の形状に関わる行列，F_{i0} は接触力に依存しない力で把持力等の表現に用いる．B スプライン基底関数にて F_i の軌道を扱えば，軌道全体にかかる式（4.15）の拘束は，基底ベクトルの大きさ個の線形不等式で表現できる．対して，関節負荷にかかる拘束や，リンク系の加速度運動と接触力の関係は二次以上の式になるため簡単には表せない．ここでは，軌道上の離散化された数点でこれら非線形の拘束を満たす軌道を考える．与えられた $k \in [0, K)$ 点と，その点でのリンク系の運動 $\boldsymbol{q}_k$ と反力 F_k を用いて.

$$\begin{pmatrix} \mathbf{0} \\ \boldsymbol{\tau}_k \end{pmatrix} = Dyn^{-1}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) + J^T(\boldsymbol{q}_k)\boldsymbol{F}_k \tag{4.16}$$

$$\boldsymbol{\tau}_{min} < \boldsymbol{\tau}_k < \boldsymbol{\tau}_{max} \tag{4.17}$$

ルートリンクはトルクを発揮できないので，式（4.16）左辺のベクトルは上から 6 つの要素が全て 0 である．$\boldsymbol{q}_k$ とその微分が固定すると，式 4.16 は線形の不等式になる．したがって，与えられたリンク系の運動に対して，摩擦・転倒・関節負荷の拘束を満たす接触力の軌道の満た

すべき拘束はすべて線形の等式・不等式で表される．したがって，

$$
\begin{aligned}
& min \sum_i \boldsymbol{F}_i W_{f_k} \boldsymbol{F}_i + \boldsymbol{w}_{f_k}^T \boldsymbol{F}_i \\
& \forall_i; -\left(M_i^- \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \ \leq \ \boldsymbol{F}_i \ \leq \ \left(M_i^+ \boldsymbol{F}_i + \boldsymbol{F}_{i0}\right) \\
& \forall_k; \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{\tau}_k \end{pmatrix} = Dyn^{-1}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k) + J^T(\boldsymbol{q_k}) \boldsymbol{F_k} \\
& \forall_k; \boldsymbol{\tau}_{min} < \boldsymbol{\tau}_k < \boldsymbol{\tau}_{max}
\end{aligned}
\tag{4.18}
$$

は二次計画問題として解くことができる．なお，以上では軌道の遷移時間を定数として扱っているが，二分探索により最大定数倍の計算時間で遷移時間を探索により決定することも可能である．

4.2.2.3.4 滑り，転がり接触状態遷移について

滑りについては，contact1，contact2 において，List4.2 の条件に式（4.19）を追加して動作を探索した．$\boldsymbol{d}_{slide}$ は滑り方向を示す単位ベクトル，F_z は接触反力の法線方向成分，μ は摩擦係数である．

$$
[{\boldsymbol{d}_{slide}}^T \ \boldsymbol{0}^T \ ] \boldsymbol{F} = -F_z \mu \tag{4.19}
$$

転がりについては式（4.20）を追加した．$\boldsymbol{d}_{rotate}$ は回転軸方向を示す単位ベクトル，F_z は接触反力の法線方向成分，ξ は接触点から回転の起こる接触面上の稜線までの距離である．

$$
[\boldsymbol{0}^T \ {\boldsymbol{d}_{rotate}}^T \ ] \boldsymbol{F} = -F_z \xi \tag{4.20}
$$

マニピュレーションの研究では，滑りや転がりを利用した操作形態をグラスプレスマニピュレーションとして分類しており [87]，転がりを利用した操作として pivoting [88] や tumbling [89] がある．ヒューマノイドロボットの足裏滑り動作については [90] にて詳細に調べられている．

4.2.3 複数の接触遷移方式探索機能を用いた行動生成実験

本章にて提案した ‘planner-before-motions’ を用いて，二つの行動生成実験を行った．Detach 遷移と Slide 遷移の切り替えの有る無しで，探索される行動の評価値改善と求解可能な問題範囲の拡大が起こる例を確認する．

4.2.3.1 立ち上がり行動生成実験

立ち上がり行動を生成した（図 4.5）．ロボットのモデルには，大出力ヒューマノイドロボット STARO [91] を用いた．図 4.4 に接触状態候補と接触リンク候補を矢印で示す．接触リンクの候補は両手両足．接触状態の候補は面の上に格子状等間隔に並べるものとした．摩擦係数は 0.5，接触面の形状には 0.05m × 0.05m の正方形を用いた．動作生成手法には Detach 遷移と Slide 遷移を探索するものを用いた．接触状態候補の評価パラメタは，$\alpha = 1.0$，$\beta = 1.0$，

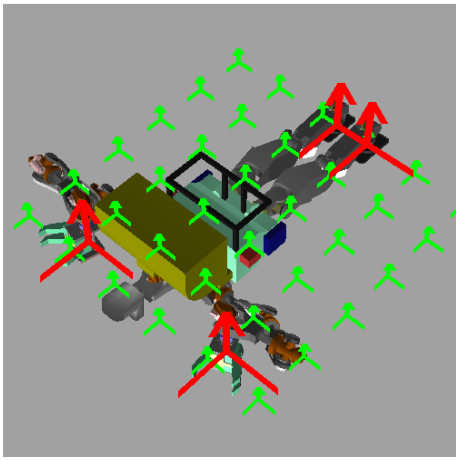


図 4.4. Green arrows show the all contact candidates and red arrows show the contact link candidates for standing up behavior.(reprinted from [36])

$\gamma = 10.0$．h_0 には式（4.7）を用い，目標位置 $\boldsymbol{p}_g$ には，その時々のロボット重心位置を用いた．探索の終了条件（endOfBehavior 関数）は両脚のみでバランスできることとした．動作生成器の選択 selectMotionPlanner 関数には，接触状態選択と同じ評価関数を用いた．すなわちそれぞれの動作生成器に次の接触状態を一つ選択させ，選択された接触状態の評価値が最も高い動作生成器を選択した．

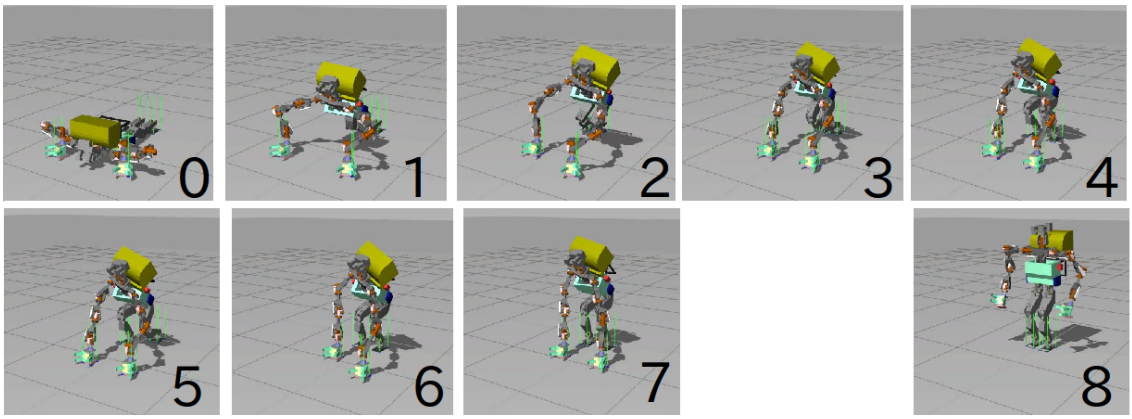


図 4.5. Standing up behavior achieved in gazebo dynamic simulator world.（reprinted from [36])

図 4.5 に出力結果を動力学シミュレータ gazebo [38] で実行したときの様子を示す．それぞれの画像は初期姿勢 0，終了姿勢 8 と，探索された接触状態ごとの姿勢を表している．遷移方式は $1 \rightarrow 2$ が右手の Slide 遷移，$2 \rightarrow 3$ が左手の Slide 遷移，$3 \rightarrow 4$ が右足の Slide 遷移，$4 \rightarrow 5$ が左足の Detach 遷移，$5 \rightarrow 6$ が右手の Slide 遷移，$6 \rightarrow 7$ が左手の Slide 遷移であった．行動生成には 105.6 秒を要し，破綻なく立ち上がり行動が実行されることが確認された．

同様の実験を Slide 遷移なしで行った場合と比較した（Table1）．各関節負荷トルクをそれぞれの最大負荷トルクで割ったベクトル $\boldsymbol{\tau}/\boldsymbol{\tau}^{max}$ を確認したところ，立ち上がり行動全体での $\boldsymbol{\tau}/\boldsymbol{\tau}^{max}$ 各要素の最大値は，Slide 遷移ありのものが 0.895，なしのものが 0.946 と僅かに

Slide 遷移ありのもののほうが低負荷であった．また，終了条件を満たすまでに探索された接触状態数，計算時間を比較すると，Slide 遷移なしの場合が 111 接触状態を 170.2 秒で探索，ありの場合が 74 接触状態を 105.6 秒で探索と Slide 遷移ありのほうが効率よく解を発見できていることが確認された．接触遷移方式の切り替えにより解の評価値が改善される例が示された．

Table1 Comparison between w/ slide and w/o slide

	Time	Output	Searched	$\boldsymbol{\tau}/\boldsymbol{\tau}^{max}$
w/ slide	105.6 s	8 contacts	74 contacts	0.895
w/o slide	170.2 s	9 contacts	111 contacts	0.946

4.2.3.2 シートへの座り行動生成

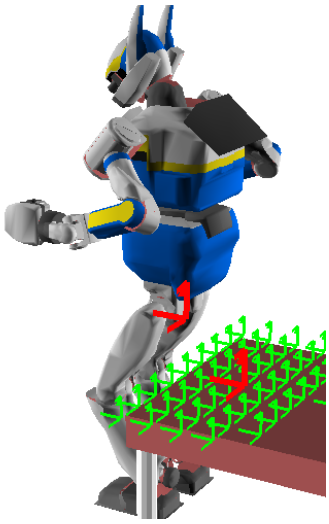


図 4.6. Contact state candidates and contact link candidates are shown. Green arrows show all contact state candidates. Red row shows goal contact state and hip contact link. (reprinted from [36])

椅子への座り込み行動を生成した（図 4.7）．図 4.6 に，接触リンクと接触状態の候補を示す．終了姿勢（図 4.7.5）は重心が足裏面の構成する多角形からはみ出した姿勢であり，臀部の接触なしに初期姿勢（図 4.7.0）から静的に遷移させることは不可能である．接触リンクには両脚と臀部リンクを用いたが両脚の接触状態候補には最初の接触のみを与えた．ロボットの臀部リンクの接触関するパラメタは，摩擦係数が並進・回転ともに 0.5，リンクの接触面の大きさは 0.1m の正方形であるとした．接触状態選択の評価関数では $\alpha = 1.0$，$\beta = 0.7$，$\gamma = 100.0$ のパラメタを用いた．h_0 には式（4.7）を用いた．接触状態の候補は，椅子モデルの上面を 0.1m × 0.1m の正方形で格子状に分割した各交点を与えた．動作生成器には，一リンクの移動のみを伴う Detach 遷移と Slide 遷移を探索するものを用いた．計算に 2.82 秒を要した．

図 4.7 に出力結果を動力学シミュレータ gazebo [38] で実行したときの様子を示す．それぞれの画像は 0 が初期姿勢，$1, 2, 3$ および $4, 5$ がそれぞれ探索された二つの接触状態を満たす姿勢である．図 4.7.0-1 の Detach 遷移と，図 4.7.3-4 の Slide 遷移がともに破綻なく実行され

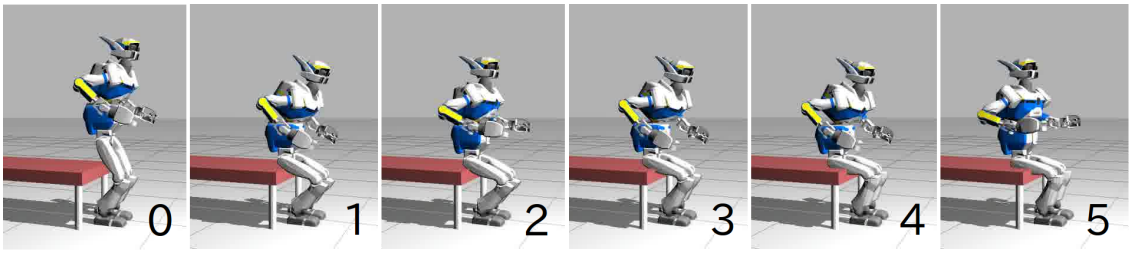


図 4.7. Seating motions achieved in gazebo world. Because we used soft seat model, robot's hip is slightly buried into seat model. (reprinted from [36])

ることが確認された.

また，Detach 遷移のみの場合と Detach/Slide 遷移を切り替える場合とで，探索の成功する目標接触位置の範囲が増減することを確認するため，全ての接触状態候補に対して，それを目標接触状態とする問題を探索した．図 4.8 にその結果を示す．灰色の四角が，両足のリンクを固定したままで臀部リンクを十分近づけることのできる接触状態候補を示し，黒が Slide 遷移ありで探索に成功した接触状態，白が Slide 遷移なしで探索に成功した接触状態である．切り替え有りの場合がなしの場合よりも倍程度多くの目標接触状態を達成できることが確認された．接触遷移方式の切り替えにより実行可能性が広がる例が示された.

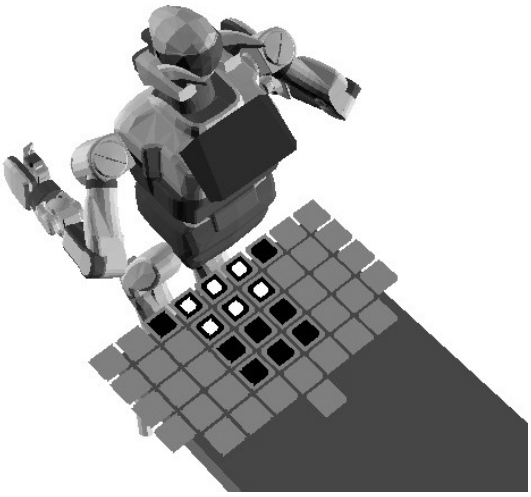


図 4.8. Reachable map of hip link. Gray tiles are all contact candidates. Black tiles are reachable contacts with Detach/Slide switching. White tiles are reachable contacts without switching. (reprinted from [36])

4.2.4　実等身大ヒューマノイドにおける滑り接触遷移を用いた摩擦パラメタ推定と着座行動実験

滑りを含む行動を実ロボットが行う際の問題点として摩擦係数の推定が困難であるという点があげられる．実際，以下にしめす実験では，着座行動時に臀部リンクを滑らせる際の摩擦力を，両足裏の力センサにより測定したが，推定された摩擦係数は多い時で 30% 程度振動しており正確な予測は困難であった．しかしながら，実際にロボットが動く中で，滑りが可能か否

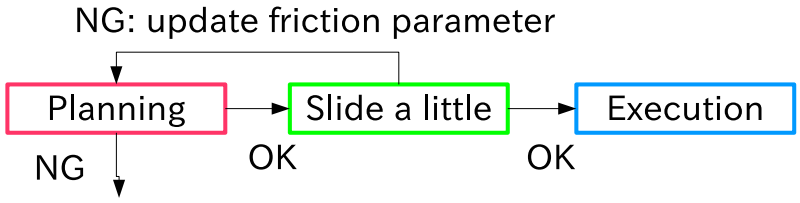


図 4.9. Experiment flow: plan motion, slide a little, and execute motion if the slide is possible. (reprinted from [36])

かという二値の判別ならば可能である．図 4.9 に，本章で行った摩擦係数推定実験のフロー図を示す．図 4.9 では，まずはじめに小さな摩擦係数の予測値（ここでは 0.08）を用いて動作を探索する．その後，実ロボットが滑り動作をわずかに実行し，摩擦力を計測，摩擦パラメタの推定を行う．計測された摩擦係数が動作生成に用いた摩擦係数の予測値よりも大きい場合には，動作の実行を停止し，倍の摩擦係数を用いて再度動作生成を行う．そうでなければ動作を実行する．

図 4.10 に実験に用いた椅子の寸法を示す．変形しない高さ 430mm の台の上に，厚さ 60mm のスポンジが乗せられている．スポンジはロボットが座ると台に比べて厚さが無視できる柔らかい材質のものを用いた．したがって動作計画で用いた椅子のモデルは高さを 430mm として扱った．スポンジを用いた理由は，ロボットの太ももリンクを保護し硬い椅子と直接接触し摩耗することを防ぐためである．硬い椅子を用い同様の実験をする場合には，太ももリンクの外装を工夫する，臀部リンクのみが接触するような脚の身体設計を行う等，ロボット自体への変更が必要である．

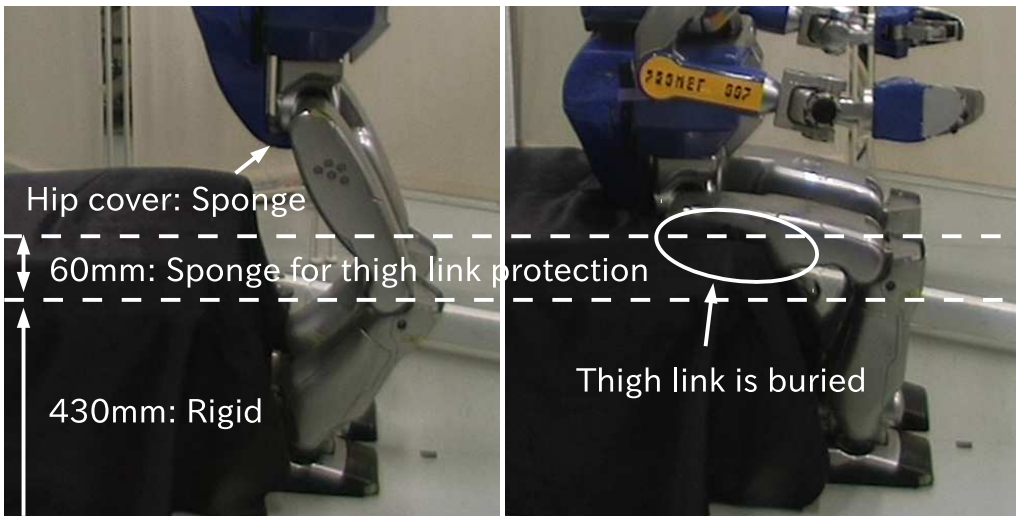


図 4.10. Chair conditions: 430mm rigid chair and 60mm soft sponge. Sponge is soft enough for ignoring the thickness. (reprinted from [36])

図 4.11 に，実等身大ヒューマノイドロボット HRP-2 [65] を用いた実験の様子を示す．グ

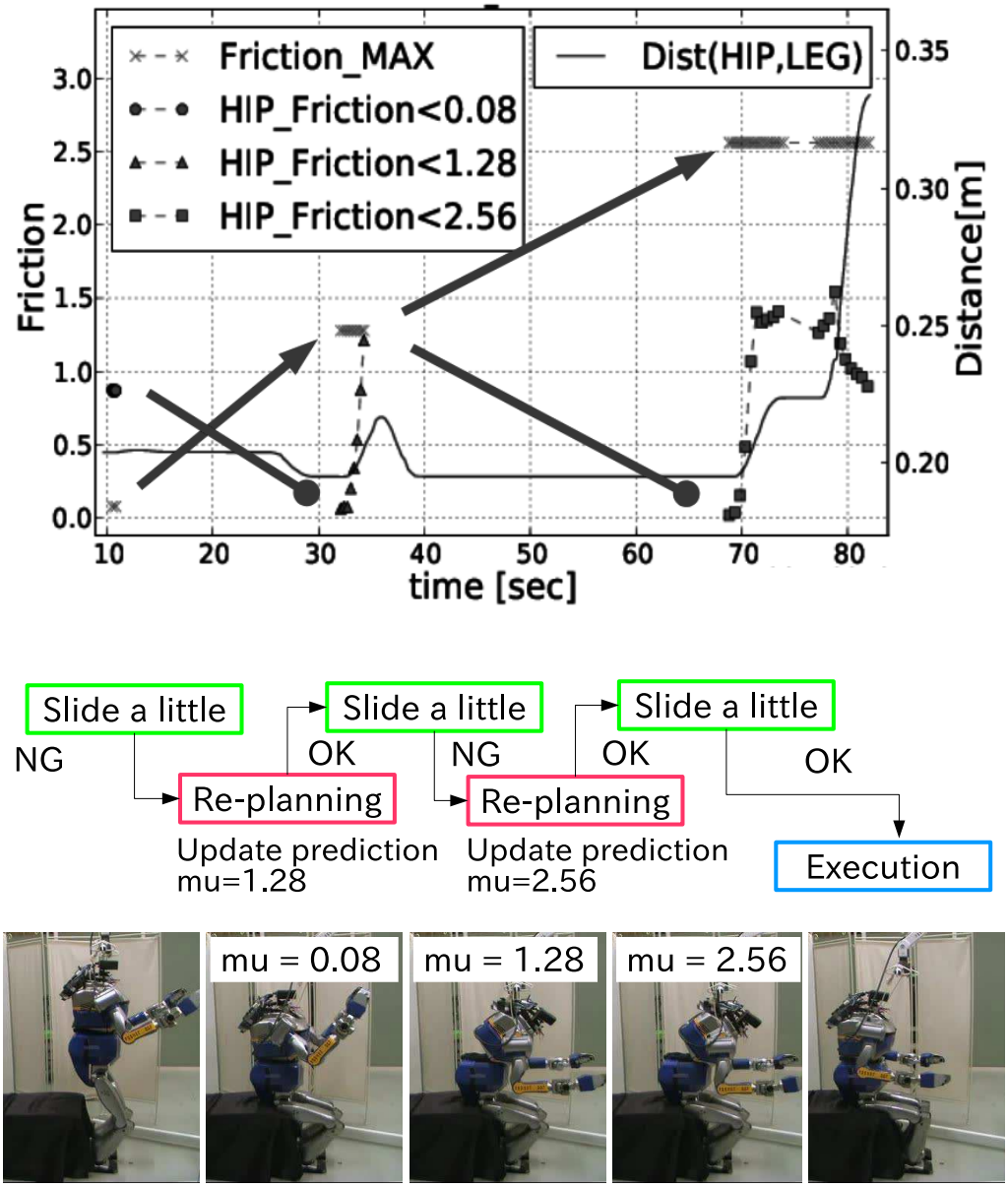


図 4.11. Experimental result of friction coefficient prediction and slide motion execution. (reprinted from [36])

ラフは，測定された力センサの値から推定される摩擦係数の値を縦軸に，時間を横軸にとったものである．摩擦係数の測定には以下の式を用いた．

$$\begin{aligned} F_h^z &= mg - (F_r^z + F_l^z) \\ F_h^x &= -(F_r^x + F_l^x) \\ \mu_h &= \frac{F_h^x}{F_h^z} \end{aligned} \tag{4.21}$$

両足首に設置された 6 軸力センサの出力のうち，鉛直上向成分力 F_r^z, F_l^z と滑り方向成分力 F_r^x, F_l^x，ロボットの全荷重 m および重力加速度 g を用いて，ロボット臀部リンク力の鉛直上

向成分 F_h^z，滑り方向成分 F_h^x，摩擦係数 μ_h は式（4.21）のようになる．またグラフ中，点線で示す値は左側縦軸の摩擦係数を表し，バツ印は動作計画で用いた摩擦係数，丸印，三角印，四角印はセンサにより推定された摩擦係数の予測値を示す．実線で示す値は右側縦軸の距離を表し，ここでは HRP-2 の各軸のエンコーダより測定された姿勢における，両足中心からの臀部リンクまでの水平距離を示す．動作計画で用いる摩擦係数は 0.08 を予測の初期値として，予想値を上回る摩擦力が測定されるごとに倍々で更新をした．図 4.11 では，最初の摩擦係数の計測値が 0.8 程度であったため，二回目の摩擦係数予測値は 0.8 以上でかつ 0.08×2 の累乗倍である 1.28，三回目ではその倍の 2.56 が用いられた．実験の結果，摩擦係数が 1.28 以上 2.56 未満であることが自動的に推定され，滑り着座行動が実現された．その際，両足裏中心を原点とする臀部リンク位置は 144.6 mm 滑り移動した．動作計画時の臀部リンクの滑り距離は 149.6 mm であった．また 図 4.9 では，動作計画が複数回呼ばれるため，動作計画にかかる計算時間が課題となりうるが，この実験では，目標状態が初期状態のすぐ近くにあるため，その探索にかかる時間は数秒から十数秒程度であり，障害とはならなかった．

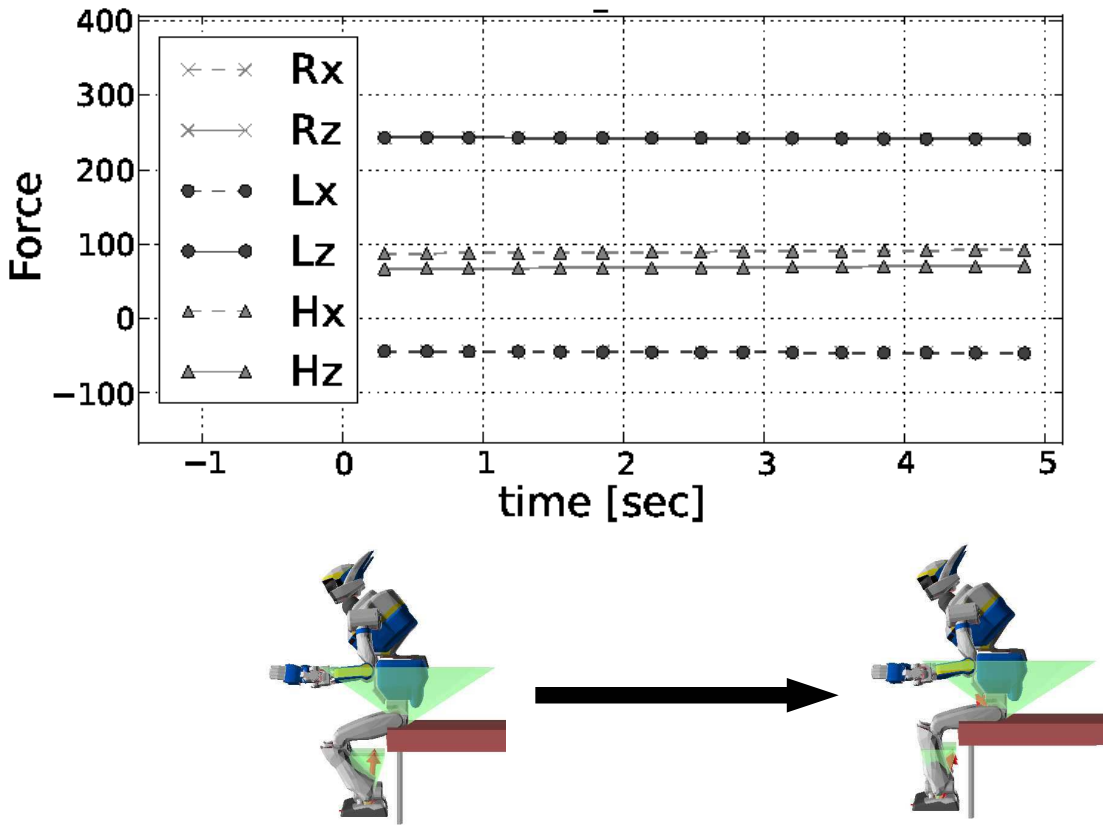


図 4.12. Planned force transitions. Rx=Forward right leg force, Rz=Vertical right leg force, Lx=Forward left leg force, Lz=Vertical left leg force, Hx=Forward hip force, and Hz=Vertical hip force. Because robot's model is symmetrical, both leg force is completaly the same and the right leg force is hidden behind the left force. (reprinted from [36])

図 4.12 に動作計画で得られた滑り動作時の接触反力グラフを示す．式（4.18）を B スプラインのパラメタ $N = 3$, $M = 20$, $K = 15$ でといたものである．図中，Rx,Rz はそれぞれ右足接触力の滑り方向，垂直方向を示す．Lx,Lz は左足についての力であり，Hx,Hz は臀部リンクのものである．また HRP-2 のモデルは左右対称であり滑り動作も左右の区別がないものであるため，計画された反力は左右の両足でまったく同じものとなっている．結果，図中の右足反力のグラフは左足反力のグラフの背後に隠れてしまっている．滑り動作は 5 秒のほぼ静的な動作であるため，計画された反力はほぼ直線となっている．臀部リンクの反力 Hx,Hz を見てみると，$Hx = \mu Hz$ が厳密に満たされていることがわかる．これは B スプラインの性質，式（4.14）によるものである．計算 0.2 秒程度要した．摩擦係数は $\mu = 1.3$ を用いた．

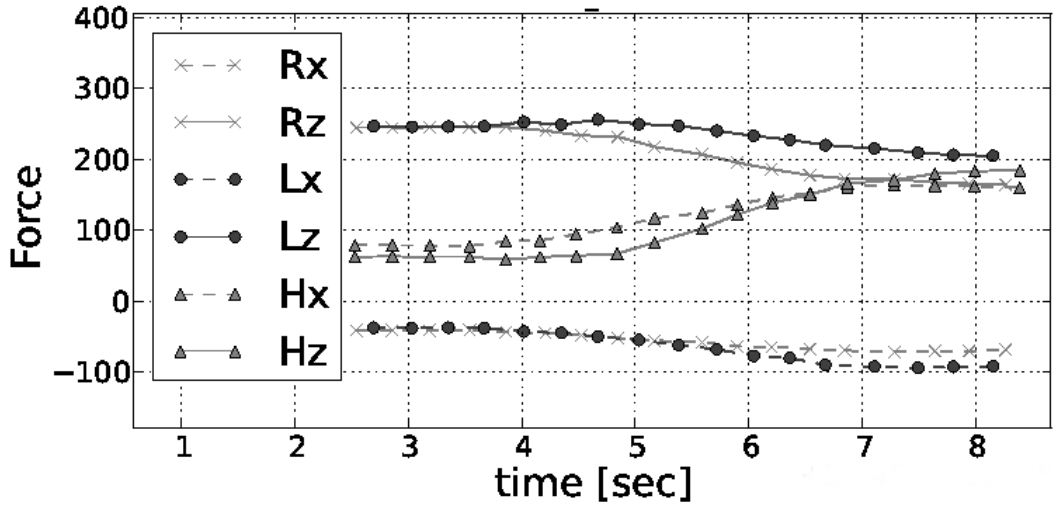


図 4.13. Actual force transitions.(reprinted from [36])

次に，図 4.13 に測定された反力遷移のグラフを示す．初期値は計画値と近い値であることが分かるが，動作終了時の値は臀部リンクの反力では 100 N 程度異なっていることがわかる．終了時の HRP-2 の関節角度，ルートリンクの姿勢は計画値とほぼ等しかったことから，この差は両脚と臀部リンクの間に生じた内力であると考えられる．接触箇所の柔らかい部分，臀部リンクのカバーと椅子表面のスポンジが弾性変形することで力が蓄えられた状態であると予想される．このように，実現可能な接触反力の計画値が得られていても実際の運動結果が計画通りになるとは限らない．蓄えられた内力は解放される際に振動を引き起こしロボットの姿勢や接触状態を目標値と異なるものに変えてしまうという危険性もある．今後の課題としては，バランスや滑り反力を考慮した実時間制御 [92] が必要である．

4.2.5　複数の接触遷移方式探索機能を有する行動生成法についてのまとめ

本章では，環境やロボットのモデルは既知として，目標状態へと至る実現可能な接触状態とロボット状態の列を生成する問題を解くためのアプローチとして，‘planner-before-

motions’ と名付ける新しい手法を提案した．‘planner-before-motions’ は，同様の問題を解く既存手法である ‘contact-before-motion’ の考えを拡張させたものということができる．‘contact-before-motion’ では接触状態の探索と動作の生成という二つの問題を解いていたが，‘planner-before-motions’ は三つの問題を解く．‘contact-before-motion’ に基づき接触と動作を生成する動作生成器（planner）を複数持ち，それらを評価関数によって切り替えることによって，動作生成器，接触状態，動作の三つの要素を探索するのである．‘planner-before-motions’ は，動作探索の複雑化を招くこと無く，滑りや転がりといった複数の接触遷移を含む行動生成を行うことができ，接触遷移方式ごとに評価のバイアスを単一のパラメタを変更することで直感的に変更できる点で優れている．また，接触遷移方式を切り替えることで，探索される行動の質向上と実行可能性の拡充が起こる例が，滑りの有る無しで行動生成問題を解き比べることで確認された．複数の接触遷移方式を探索することによる計算コストの増加については，計算コストの高い動作探索を行うこと無く，ロボットの負荷を考慮した接触選択を可能にする評価関数を用いる工夫により高速化をはかった．滑りや最大出力についての危険度を接触反力として表現し，その方向に大きな力を得ることのできる接触を選択することで，実現可能な動作を生成しうる接触を高速に評価する手法を用いた．これは動作探索の結果を接触状態のみから予測する方法であり，動作探索を行う方法に比べ精度では劣ると予想される．しかし実際のロボットの動作では，行動の正確無比な実行は困難であり，十分に優良な解を高速に再探索できることが，最も重要であると考える．また，高速な再探索により，摩擦係数が未知の環境であっても滑り着座行動が可能になることが，実等身大ヒューマノイドロボットを用いた実験で示された．実験で得られた知見として，滑り動作時にロボットと環境の接触面間に内力が蓄積するという課題が見られた．また予想されたことではあるが，滑り動作時の接触反力は制御なしには計画値に従わないことが確認された．接触力の制御については押し付け倣い作業など多くの先行研究があり，今後それらの手法を用いるなどして，滑り接触反力の制御が可能になるかについての調査研究が課題として残されている．本章の成果は，任意の動作生成手法を組み合わせて全体の行動を生成するアルゴリズムにより，滑りや歩行といった複数の動作形式を統一的に扱い行動生成を行う手法を構築し，ロボットの接触を伴う行動生成問題の一般性を向上させたことである．

4.3 動力学シミュレーションを用いた接触行動同時探索法

多リンク系の動力学、運動学についてはすでに多くの優れた解説 [77] [78] が存在する。以下では本論文で用いた動力学シミュレータの実装に合わせて用いた数式と簡単な導出をまずまとめる。以下の動力学シミュレータで用いた数式はすべて既知のものであるが、のちに示すモデル誤差最小化やモデル・制御パラメタも含んだ最適化問題が現実的な時間で解けているのは、この簡易動力学シミュレータが高速であるためという理由が大きいと考え、その構成を明示しておくことは有益であるため本文中に載せる。シミュレータの構成についての説明の後、実装されたシミュレーションを用いて平らな地面の上での動歩行行動生成実験を行う。

4.3.1　高速計算可能な簡易動力学シミュレータ構成

4.3.1.1　動力学シミュレータの全体構成

質点と無限に続く平面がそれぞれ一つだけある世界を考える．質点の運動をシミュレーションするためには，以下の三つの計算が必要である．

1. 床と質点が衝突しているかを確認し，衝突しているなら反力を計算する．
2. 質点が受けている力からニュートンの運動方程式を解くことで質点の加速度を決定する．
3. 質点の加速度を積分し新しい位置を計算する．

多リンク系のシミュレーションでもこの流れは変わらない．2. は順動力学計算と呼ばれ，接触力とロボットの発揮するトルクからロボットの各関節の加速度を計算する手続きである．3. は順運動学と呼ばれ，関節の位置・速度・加速度を用いて全リンクの三次元空間での位置および姿勢とその速度・加速度を計算する．以下では逆順に，3. 順運動学，2. （逆動力学 $\rightarrow$）順動力学，1. 接触力計算の順に説明を行う．

4.3.1.2　順運動学

本章で用いる記号を以下の表 4.3 にまとめる．下付き添字は二番目が値の示す対象関節の ID，一番目が値を表現する座標系を持つ関節の ID とし，一番目の座標系 ID は世界座標系については省略することとする．

Table4.3. All parameters used in this forward-kinematics section

$\boldsymbol{p}_i$	i-th joint position in world frame
R_i	i-th joint orientation matrix in world frame
$\boldsymbol{v}_i$	i-th joint velocity in world frame
$\boldsymbol{w}_i$	i-th joint angular velocity in world frame
$\dot{\boldsymbol{v}}_i$	i-th joint acceleration in world frame
$\dot{\boldsymbol{w}}_i$	i-th joint angular acceleration in world frame
$\boldsymbol{p}_{ij}$	j-th joint position in i-th joint frame
R_{ij}	j-th joint orientation matrix in i-th joint frame
$\boldsymbol{a}_i$	i-th joint axis vector in world frame
$\boldsymbol{a}_{ii}$	i-th joint axis vector in i-th joint frame

本章で考える多リンク系の模式図 4.14 を以下にしめす．

Algorithm 6 Pseudo code of dynamics simulation

Require: i, Δt, t_{max}

i: root link id.

Δt: simulation time step.

t_{max}: maximum simulation duraction.

Variable: t

t: current time sec.

Procedure: DynamicsSimulation(i, Δt, t_{max})

1: $t = 0$
2: initSim(i)
3: **while** $t \leq t_{max}$ **do**
4: procSim(i, Δt)
5: $t = t + \Delta t$
6: **end while**

Procedure: initSim(i)

7: InitializeAllVariables()
8: ForwardKinematics(i)

Procedure: procSim(i, Δt)

9: CalcForce(i, Δt)
10: ForwardDyanmics(i)
11: UpdateEuler(i, Δt)
12: ForwardKinematics(i)

4.3.1.2.1 再帰的な位置・速度・加速度の計算について

一番目の関節の位置姿勢は図 4.14 に示す幾何学的な関係から以下の式（4.22）のように計算される．

$$\begin{aligned} \boldsymbol{p}_1 &= \boldsymbol{p}_0 + R_0 \boldsymbol{p}_{01} \\ R_1 &= R_0 R_{01} e^{\boldsymbol{a}_{11} q_1} \end{aligned} \tag{4.22}$$

ここで，速度と位置の微分，角速度と姿勢行列の微分には以下の式（4.23）関係がなりたつ．

$$\begin{aligned} \dot{\boldsymbol{p}}_i &= \boldsymbol{v}_i \\ \dot{R}_i &= \boldsymbol{w}_i \times R_i \end{aligned} \tag{4.23}$$

ただしベクトルの行列との外積は以下式（4.24）のように定義する．

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}, \boldsymbol{x} \times R = \begin{pmatrix} 0 & -x_2 & x_1 \\ x_2 & 0 & -x_0 \\ -x_1 & x_0 & 0 \end{pmatrix} R \tag{4.24}$$

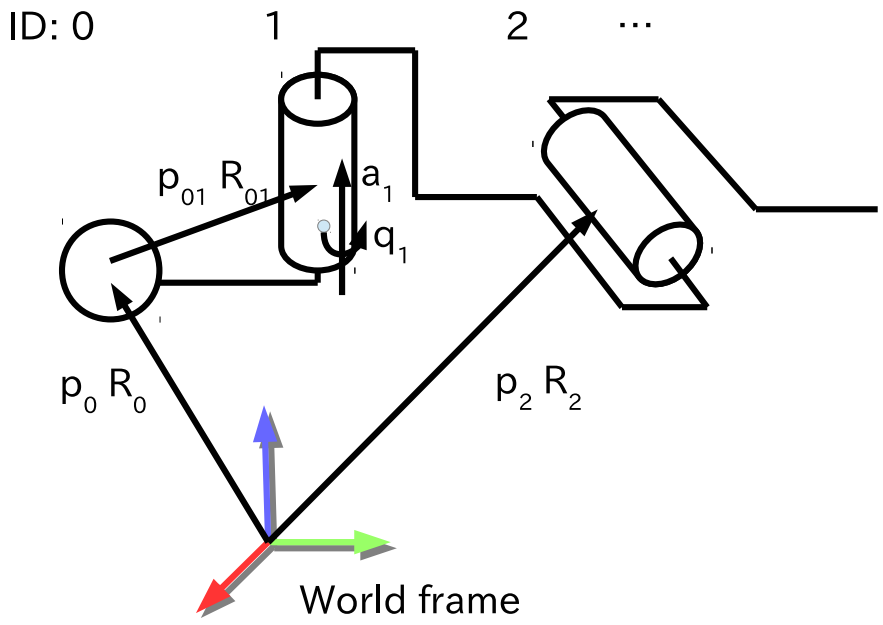


図 4.14. Link tree: $\boldsymbol{p}_i$ is i-th joint position, R_i is i-th joint orientation, and $\boldsymbol{p}_{ij}, R_{ij}$ are relative j-th value seen from i-th joint.

速度，角速度は式（4.22）を微分することによって親関節の値を使って再帰的に定義できる．

$$\begin{aligned}
\dot{\boldsymbol{p}}_1 &= \dot{\boldsymbol{p}}_0 + \dot{R}_0 \boldsymbol{p}_{01} \\
&= \boldsymbol{v}_0 + \boldsymbol{w}_0 \times R_0 \boldsymbol{p}_{01} \\
&= \boldsymbol{v}_1 \\
\dot{R}_1 &= \dot{R}_0 R_{01} e^{\boldsymbol{a}_{11} q_1} + R_0 R_{01} \dot{e}^{\boldsymbol{a}_{11} q_1} \\
&= \boldsymbol{w}_0 \times R_0 R_{01} e^{\boldsymbol{a}_{11} q_1} + \boldsymbol{a}_1 \dot{q}_1 \times R_0 R_{01} e^{\boldsymbol{a}_{11} q_1} \\
&= (\boldsymbol{w}_0 + \boldsymbol{a}_1 \dot{q}_1) \times R_1 \\
&= \boldsymbol{w}_1 \times R_1
\end{aligned} \tag{4.25}$$

ただし，式（4.25）では以下の式（4.26）が成り立つことを利用している．

$$\begin{aligned}
R_0 R_{01} \dot{e}^{\boldsymbol{a}_{11} q_1} &= R_0 R_{01} \left(\boldsymbol{a}_{11} \dot{q}_1 \times e^{\boldsymbol{a}_{11} q_1} \right) \\
&= \left(R_0 R_{01} \boldsymbol{a}_{11} \dot{q}_1 \right) \times \left(R_0 R_{01} e^{\boldsymbol{a}_{11} q_1} \right) \\
&= \boldsymbol{a}_1 \dot{q}_1 \times R_0 R_{01} e^{\boldsymbol{a}_{11} q_1}
\end{aligned} \tag{4.26}$$

これは一般に，任意の回転行列 R と任意の三次元ベクトル $\boldsymbol{a}, \boldsymbol{b}$ を用いて以下の式（4.27）が成り立つためである．すなわち外積を計算するときに，すべてのベクトルを等しく回転させてから外積を計算することと，外積を計算してから回転させることが等しい．

$$R \left(\boldsymbol{a} \times \boldsymbol{b} \right) = \left(R \boldsymbol{a} \right) \times \left(R \boldsymbol{b} \right) \tag{4.27}$$

次に加速度，各加速度の計算について考える．加速度，角加速度も式（4.25）を微分するこ

とによって親関節の値を使って再帰的に定義できる．

$$\begin{aligned}\dot{\boldsymbol{v}}_1 &= \dot{\boldsymbol{v}}_0 + \dot{\boldsymbol{w}}_0 \times R_0\boldsymbol{p}_{01} + \boldsymbol{w}_0 \times (\boldsymbol{w}_0 \times R_0\boldsymbol{p}_{01}) \\ \dot{\boldsymbol{w}}_1 &= \dot{\boldsymbol{w}}_0 + \boldsymbol{w}_0 \times \boldsymbol{a}_1\dot{q}_1 + \boldsymbol{a}_1\ddot{q}_1\end{aligned} \tag{4.28}$$

ただし

$$\dot{\boldsymbol{a}}_1 = \dot{R}_0\boldsymbol{a}_{11} = \boldsymbol{w}_0 \times R_0\boldsymbol{a}_{11} = \boldsymbol{w}_0 \times \boldsymbol{a}_1 \tag{4.29}$$

である．

4.3.1.2.2 空間速度を用いた計算について

空間速度を用いることで，順運動学以外の逆運動学や逆動力学でも再利用可能なヤコビアンとその微分を用いて順運動学計算を行うことができる．本論文で用いた動力学シミュレーションでは本章に示す計算で順運動学が計算されている．まず空間速度の定義を以下式（4.30）にしめす．

$$\begin{aligned}\boldsymbol{u}_i &= \boldsymbol{v}_i - \boldsymbol{w}_i \times \boldsymbol{p}_i \\ \dot{\boldsymbol{u}}_i &= \dot{\boldsymbol{v}}_i - \dot{\boldsymbol{w}}_i \times \boldsymbol{p}_i - \boldsymbol{w}_i \times \dot{\boldsymbol{p}}_i \\ &= \dot{\boldsymbol{v}}_i - \dot{\boldsymbol{w}}_i \times \boldsymbol{p}_i - \boldsymbol{w}_i \times (\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{p}_i)\end{aligned} \tag{4.30}$$

次に式（4.25） を空間速度を用いて変換する．

$$\begin{aligned}\boldsymbol{u}_1 &= \boldsymbol{v}_1 - \boldsymbol{w}_1 \times \boldsymbol{p}_1 \\ &= \boldsymbol{v}_0 + \boldsymbol{w}_0 \times R_0\boldsymbol{p}_{01} - (\boldsymbol{w}_0 + \boldsymbol{a}_1\dot{q}_1) \times (\boldsymbol{p}_0 + R_0\boldsymbol{p}_{01}) \\ &= \boldsymbol{v}_0 - \boldsymbol{w}_0 \times \boldsymbol{p}_0 - \boldsymbol{a}_1\dot{q}_1 \times (\boldsymbol{p}_0 + R_0\boldsymbol{p}_{01}) \\ &= \boldsymbol{u}_0 + (\boldsymbol{p}_1 \times \boldsymbol{a}_1)\,\dot{q}_1 \\ \boldsymbol{w}_1 &= \boldsymbol{w}_0 + \boldsymbol{a}_1\dot{q}_1\end{aligned} \tag{4.31}$$

ここで，以下の変数式（4.32）を定義する．

$$\begin{aligned}\Delta\boldsymbol{u}_i &= \boldsymbol{p}_i \times \boldsymbol{a}_i \\ \Delta\boldsymbol{w}_i &= \boldsymbol{a}_i\end{aligned} \tag{4.32}$$

式（4.31）より，空間速度を用いた速度・加速度は以下式（4.33）ように定義できる．

$$\boldsymbol{u}_1 = \boldsymbol{u}_0 + \Delta\boldsymbol{u}_i\dot{q}_1 \tag{4.33}$$

$$\boldsymbol{w}_1 = \boldsymbol{w}_0 + \Delta\boldsymbol{w}_i\dot{q}_1$$

$$\dot{\boldsymbol{u}}_1 = \dot{\boldsymbol{u}}_0 + \Delta\dot{\boldsymbol{u}}_i\dot{q}_1 + \Delta\boldsymbol{u}_i\ddot{q}_1 \tag{4.34}$$

$$\dot{\boldsymbol{w}}_1 = \dot{\boldsymbol{w}}_0 + \Delta\dot{\boldsymbol{w}}_i\dot{q}_1 + \Delta\boldsymbol{w}_i\ddot{q}_1$$

式（4.32）の微分は以下の式（4.35）ように計算できる．

$$
\begin{aligned}
\Delta\dot{\boldsymbol{u}}_i &= \dot{\boldsymbol{p}}_i \times \boldsymbol{a}_i + \boldsymbol{p}_i \times \dot{\boldsymbol{a}}_i \\
&= (\boldsymbol{v}_j + \boldsymbol{w}_j \times (\boldsymbol{p}_i - \boldsymbol{p}_j)) \times \boldsymbol{a}_i + \boldsymbol{p}_i \times (\boldsymbol{w}_j \times \boldsymbol{a}_i) \\
&= (\boldsymbol{u}_j + \boldsymbol{w}_j \times \boldsymbol{p}_i) \times \boldsymbol{a}_i + \boldsymbol{p}_i \times (\boldsymbol{w}_j \times \boldsymbol{a}_i) \\
&= \boldsymbol{u}_j \times \boldsymbol{a}_i - \boldsymbol{a}_i \times (\boldsymbol{w}_j \times \boldsymbol{p}_i) - \boldsymbol{p}_i \times (\boldsymbol{a}_i \times \boldsymbol{w}_j) \\
&= \boldsymbol{u}_j \times \boldsymbol{a}_i + \boldsymbol{w}_j \times (\boldsymbol{p}_i \times \boldsymbol{a}_i) \\
&= \boldsymbol{u}_j \times \Delta\boldsymbol{w}_i + \boldsymbol{w}_j \times \Delta\boldsymbol{u}_i \\
\Delta\dot{\boldsymbol{w}}_i &= \dot{\boldsymbol{a}}_i \\
&= \boldsymbol{w}_j \times \boldsymbol{a}_i \\
&= \boldsymbol{w}_j \times \Delta\boldsymbol{w}_i
\end{aligned}
\tag{4.35}
$$

ただし，式（4.35）では以下のヤコビ恒等式（4.36）が成り立つことを利用している．

$$
\boldsymbol{a} \times (\boldsymbol{w} \times \boldsymbol{p}) + \boldsymbol{p} \times (\boldsymbol{a} \times \boldsymbol{w}) + \boldsymbol{w} \times (\boldsymbol{p} \times \boldsymbol{a}) = 0 \tag{4.36}
$$

最後に，以上をまとめたアルゴリズムを Algorithm. 7 に示す．手続き ForwardKinematics はルートリンクを引数とし各リンクのキネマティクス情報を計算しながら子リンクの ForwardKinematics を呼び出す再帰手続きである．

4.3.1.3　逆動力学

本章で用いる記号を表 4.4 にまとめる．

Table4.4. All parameters used in this inverse-dynamics section

P_i	i-th joint momentum in world frame
L_i	i-th joint angular momentum in world frame
m_i	i-th joint weight
I_i	i-th joint inertia tensor in world frame
$\boldsymbol{c}_i$	i-th joint centroid in world frame
$\dot{\boldsymbol{c}}_i$	i-th joint centroid velocity in world frame
$\ddot{\boldsymbol{c}}_i$	i-th joint centroid acceleration in world frame
$\boldsymbol{w}_i$	i-th joint angular velocity in world frame
$\dot{\boldsymbol{w}}_i$	i-th joint angular acceleration in world
$\boldsymbol{f}_i$	i-th joint inner force in world frame
$\boldsymbol{n}_i$	i-th joint inner moment in world frame
τ_i	i-th joint torque

Algorithm 7 Pseudo code of forward kinematics

Require: i

i: joint id.

Procedure: ForwardKinematics(i)

updateFK(i)
for all k in childrens(i) **do**
  mainFK(k, i)
end for

Procedure: updateFK(i)

$\boldsymbol{a}_i = R_i \boldsymbol{a}_{i,i}$
$\boldsymbol{c}_i = R_i \boldsymbol{c}_{i,i}$
$I_i = R_i I_{i,i} R_i^T + m_i [\boldsymbol{c}\times][\boldsymbol{c}\times]^T$

Procedure: mainFK(i, j)

$\boldsymbol{p}_i = \boldsymbol{p}_j + R_j \boldsymbol{p}_{j,i}$
$R_i = R_j R_{j,i} e^{\boldsymbol{a}_{i,i} q_i}$
updateFK(i)
$\Delta \boldsymbol{u}_i = \boldsymbol{p}_i \times \boldsymbol{a}_i$
$\Delta \boldsymbol{w}_i = \boldsymbol{a}_i$
$\Delta \dot{\boldsymbol{u}}_i = \boldsymbol{u}_j \times \Delta \boldsymbol{w}_i + \boldsymbol{w}_j \times \Delta \boldsymbol{u}_i$
$\Delta \dot{\boldsymbol{w}}_i = \boldsymbol{w}_j \times \Delta \boldsymbol{w}_i$
$\boldsymbol{u}_i = \boldsymbol{u}_j + \Delta \boldsymbol{u}_i \dot{q}_i$
$\boldsymbol{w}_i = \boldsymbol{w}_j + \Delta \boldsymbol{w}_i \dot{q}_i$
$\dot{\boldsymbol{u}}_i = \dot{\boldsymbol{u}}_j + \Delta \dot{\boldsymbol{u}}_i \dot{q}_i + \Delta \boldsymbol{u}_i \ddot{q}_i$
$\dot{\boldsymbol{w}}_i = \dot{\boldsymbol{w}}_j + \Delta \dot{\boldsymbol{w}}_i \dot{q}_i + \Delta \boldsymbol{w}_i \ddot{q}_i$
for all k in childrens(i) **do**
  mainFK(k, i)
end for

4.3.1.3.1 重心等関節上の点の速度・加速度について

以下の式（4.37）のようになる．

$$
\begin{aligned}
\boldsymbol{c}_i &= \boldsymbol{p}_i + R_i \boldsymbol{c}_{ii} \\
\dot{\boldsymbol{c}}_i &= \boldsymbol{v}_i + \boldsymbol{w}_i \times R_i \boldsymbol{c}_{ii} \\
&= \boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{p}_i + \boldsymbol{w}_i \times R_i \boldsymbol{c}_{ii} \\
&= \boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{c}_i \\
\ddot{\boldsymbol{c}}_i &= \dot{\boldsymbol{u}}_i + \dot{\boldsymbol{w}}_i \times \boldsymbol{c}_i + \boldsymbol{w}_i \times \dot{\boldsymbol{c}}_i \\
&= \dot{\boldsymbol{u}}_i + \dot{\boldsymbol{w}}_i \times \boldsymbol{c}_i + \boldsymbol{w}_i \times (\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{c}_i)
\end{aligned}
\tag{4.37}
$$

4.3.1.3.2 運動量について

運動量 P_i，角運動量 L_i，重心位置 $\boldsymbol{c}_i$，質量 m_i，関節上の点 j の世界座標系での位置 $\boldsymbol{r}_j$，世界座標系で見た慣性テンソル I_i の関係は以下式（4.38）のように計算できる．

$$
\begin{aligned}
\boldsymbol{P}_i &= m_i \dot{\boldsymbol{c}}_i \\
&= m_i \boldsymbol{u}_i + m_i \boldsymbol{w}_i \times \boldsymbol{c}_i \\
\boldsymbol{L}_i &= m_i \sum\nolimits_j \boldsymbol{r}_j \times (\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{r}_j) \\
&= m_i \left(\sum\nolimits_j \boldsymbol{r}_j \right) \times \boldsymbol{u}_i + m_i \sum\nolimits_j \boldsymbol{r}_j \times (\boldsymbol{w}_i \times \boldsymbol{r}_j) \\
&= m_i \left(\sum\nolimits_j \boldsymbol{r}_j \right) \times \boldsymbol{u}_i + m_i \sum\nolimits_j [\boldsymbol{r}_j \times] [\boldsymbol{r}_i \times]^T \boldsymbol{w}_i \\
&= m_i \boldsymbol{c}_i \times \boldsymbol{u}_i + I_i \boldsymbol{w}_i
\end{aligned}
\tag{4.38}
$$

運動量・角運動量は微分することで力・モーメントになる．

$$
\begin{aligned}
\dot{\boldsymbol{P}}_i &= m_i \dot{\boldsymbol{u}}_i + m_i \dot{\boldsymbol{w}}_i \times \boldsymbol{c}_i + m_i \boldsymbol{w}_i \times \dot{\boldsymbol{c}}_i \\
&= m_i \dot{\boldsymbol{u}}_i + m_i \dot{\boldsymbol{w}}_i \times \boldsymbol{c}_i + \boldsymbol{w}_i \times \boldsymbol{P}_i \\
\dot{\boldsymbol{L}}_i &= m_i \dot{\boldsymbol{c}}_i \times \boldsymbol{u}_i + m_i \boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i \dot{\boldsymbol{w}}_i + \dot{I}_i \boldsymbol{w}_i \\
&= m_i \boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i \dot{\boldsymbol{w}}_i + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{w}_i \times \boldsymbol{L}_i
\end{aligned}
\tag{4.39}
$$

ただし，以下の式（4.40）が成り立つことを用いた．まずヤコビ恒等式を用いて

$$
\begin{aligned}
&\sum\nolimits_j \boldsymbol{r}_j \times (\boldsymbol{w} \times (\boldsymbol{w} \times \boldsymbol{r}_j)) \\
=& \sum\nolimits_j - (\boldsymbol{w} \times \boldsymbol{r}_j) \times (\boldsymbol{r}_j \times \boldsymbol{w}) - \boldsymbol{w} \times ((\boldsymbol{w} \times \boldsymbol{r}_j) \times \boldsymbol{r}_j) \\
=& \sum\nolimits_j \boldsymbol{w} \times (\boldsymbol{r}_j \times (\boldsymbol{w} \times \boldsymbol{r}_j)) \\
=& \boldsymbol{w} \times I \boldsymbol{w}
\end{aligned}
\tag{4.40}
$$

次に慣性テンソルの微分について式（4.41）

$$
\begin{aligned}
&\dot{I}_i \boldsymbol{w}_i \\
=& m_i \sum\nolimits_j \dot{\boldsymbol{r}}_j \times (\boldsymbol{w}_i \times \boldsymbol{r}_j) + \boldsymbol{r}_j \times (\boldsymbol{w}_i \times \dot{\boldsymbol{r}}_j) \\
=& m_i \sum\nolimits_j (\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{r}_j) \times (\boldsymbol{w}_i \times \boldsymbol{r}_j) \\
&+ \boldsymbol{r}_j \times (\boldsymbol{w}_i \times (\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{r}_j)) \\
=& m_i \sum\nolimits_j \boldsymbol{u}_i \times (\boldsymbol{w}_i \times \boldsymbol{r}_j) + \boldsymbol{r}_j \times (\boldsymbol{w}_i \times \boldsymbol{u}_i) \\
&+ \boldsymbol{r}_j \times (\boldsymbol{w}_i \times (\boldsymbol{w}_i \times \boldsymbol{r}_j)) \\
=& m_i \boldsymbol{u}_i \times (\boldsymbol{w}_i \times \boldsymbol{c}_i) + m_i \boldsymbol{c}_i \times (\boldsymbol{w}_i \times \boldsymbol{u}_i) + \boldsymbol{w}_i \times I_i \boldsymbol{w}_i \\
=& \boldsymbol{w}_i \times \boldsymbol{L}_i - m_i \left(-\boldsymbol{u}_i \times (\boldsymbol{w}_i \times \boldsymbol{c}_i) + \boldsymbol{c}_i \times (\boldsymbol{u}_i \times \boldsymbol{w}_i) + \boldsymbol{w}_i \times (\boldsymbol{c}_i \times \boldsymbol{u}_i) \right) \\
=& \boldsymbol{w}_i \times \boldsymbol{L}_i + 2 m_i \boldsymbol{u}_u \times (\boldsymbol{w}_i \times \boldsymbol{c}_i)
\end{aligned}
\tag{4.41}
$$

最後に以下式（4.42）が成り立つ．

$$\begin{aligned} & m_i\dot{\boldsymbol{c}}_i \times \boldsymbol{u}_i + \dot{I}_i\boldsymbol{w}_i \\ =\ & m_i\left(\boldsymbol{u}_i + \boldsymbol{w}_i \times \boldsymbol{c}_i\right) \times \boldsymbol{u}_i + \dot{I}_i\boldsymbol{w}_i \\ =\ & -m_i\boldsymbol{u}_i \times \left(\boldsymbol{w}_i \times \boldsymbol{c}_i\right) + \dot{I}_i\boldsymbol{w}_i \\ =\ & \boldsymbol{w}_i \times \boldsymbol{L}_i + m_i\boldsymbol{u}_i \times \left(\boldsymbol{w}_i \times \boldsymbol{c}_i\right) \\ =\ & \boldsymbol{w}_i \times \boldsymbol{L}_i + \boldsymbol{u}_i \times \boldsymbol{P}_i \end{aligned} \tag{4.42}$$

4.3.1.3.3 運動方程式の再帰的な表現について

運動方程式を以下の式（4.43）にしめす．

$$\begin{aligned} \dot{\boldsymbol{P}}_i &= \boldsymbol{F}_i \\ \dot{\boldsymbol{L}}_i &= \boldsymbol{N}_i \end{aligned} \tag{4.43}$$

ここで，$\boldsymbol{F}_i, \boldsymbol{N}_i$ は i 番目の関節に加わる全力・モーメントを表しており，多リンク系では親関節との内力 $\boldsymbol{f}_i, \boldsymbol{n}_i$，全子関節内力の反力 $\sum_k -\boldsymbol{f}_k, \sum_k -\boldsymbol{n}_k$ ，重力 $m_i\boldsymbol{g}, \boldsymbol{c}_i \times m_i\boldsymbol{g}$ と外力 $\boldsymbol{f}_{ei}, \boldsymbol{n}_{ei}$ の和に相当する．すなわち式（4.39）を用いて，

$$\begin{aligned} \dot{\boldsymbol{P}}_i &= \boldsymbol{f}_i - \sum_k \boldsymbol{f}_k + m_i\boldsymbol{g} + \boldsymbol{f}_{ei} \\ &= m_i\dot{\boldsymbol{u}}_i + m_i\dot{\boldsymbol{w}}_i \times \boldsymbol{c}_1 + \boldsymbol{w}_i \times \boldsymbol{P}_i \\ \dot{\boldsymbol{L}}_i &= \boldsymbol{n}_i - \sum_k \boldsymbol{n}_k + \boldsymbol{c}_i \times m_i\boldsymbol{g} + \boldsymbol{n}_{ei} \\ &= m_i\boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i\dot{\boldsymbol{w}}_i + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{w}_i \times \boldsymbol{L}_i \end{aligned} \tag{4.44}$$

また関節負荷トルクは以下式（4.45）のように計算できる．

$$\tau_i = \Delta\boldsymbol{u}_i^T\boldsymbol{f}_i + \Delta\boldsymbol{w}_i^T\boldsymbol{n}_i \tag{4.45}$$

末端リンクでは，子リンクの力・モーメントは零と考えられるので，式（4.44）は末端リンクから遡って計算することで全リンクの力・モーメントを計算でき，それらを用いてトルクも計算できる．以上をまとめたアルゴリズムを Algorithm. 8 に示す．手続き InverseDynamics はルートリンクを引数とし子リンクをたどっていって末端リンクが見つかるとそこから逆順にダイナミクス情報を計算しながらルートリンクへと戻る再帰手続きである．

4.3.1.4 順動力学

本章で用いる記号を表 4.5 にまとめる．

順運動学は，リンクの運動が子リンクの影響を受けないことを利用して，親リンクから順に計算することで $\mathcal{O}(N)$ の手続きで求めることができた．逆動力学は，リンクの力の釣り合いが子リンクと親リンクとの内力の影響を受けるが末端リンクは親リンクのみとの内力を考えればよいことを利用して，末端リンクから親リンクとの内力を計算していくことで $\mathcal{O}(N)$ の手続きで求めることができた．しかし，順動力学ではリンクの速度・加速度が未知であり関節間

Algorithm 8 Pseudo code of inverse dynamics

Require: i

i: joint id.

Procedure: InverseDynamics(i)

1: **for all** k in childrens(i) **do**
2:　InverseDynamics(k)
3: **end for**
4: $\boldsymbol{P}_i = m_i \boldsymbol{u}_i + m_i \boldsymbol{w}_i \times \boldsymbol{c}_i$
5: $\boldsymbol{L}_i = m_i \boldsymbol{c}_i \times \boldsymbol{u}_i + I_i \boldsymbol{w_i}$
6: $\dot{\boldsymbol{P}}_i = m_i \dot{\boldsymbol{u}}_i + m_i \dot{\boldsymbol{w}}_i \times \boldsymbol{c}_1 + \boldsymbol{w}_i \times \boldsymbol{P}_i$
7: $\dot{\boldsymbol{L}}_i = m_i \boldsymbol{c}_i \times \dot{\boldsymbol{u}}_i + I_i \dot{\boldsymbol{w}}_i + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{w}_i \times \boldsymbol{L}_i$
8: $\boldsymbol{f}_i = \dot{\boldsymbol{P}}_i + \sum_k \boldsymbol{f}_k - m_i \boldsymbol{g} - \boldsymbol{f}_{ei}$
9: $\boldsymbol{n}_i = \dot{\boldsymbol{L}}_i + \sum_k \boldsymbol{n}_k - \boldsymbol{c}_i \times \boldsymbol{m}_i \boldsymbol{g} - \boldsymbol{n}_{ei}$
10: $\tau_i = \Delta \boldsymbol{u}_i^T \boldsymbol{f}_i + \Delta \boldsymbol{w}_i^T \boldsymbol{n}_i$

Table4.5. All parameters used in this forward-dynamics section

P_i	i-th joint momentum in world frame
L_i	i-th joint angular momentum in world frame
m_i	i-th joint weight
I_i	i-th joint inertia tensor in world frame
$\boldsymbol{c}_i$	i-th joint centroid in world frame
$\boldsymbol{w}_i$	i-th joint angular velocity in world frame
$\dot{\boldsymbol{w}}_i$	i-th joint angular acceleration in world
$\boldsymbol{f}_i$	i-th joint inner force in world frame
$\boldsymbol{n}_i$	i-th joint inner moment in world frame
τ_i	i-th joint torque
$\boldsymbol{y}_i$	i-th joint inner force and moment in world frame
$\ddot{\boldsymbol{x}}_i$	i-th joint linear/angular acceleration in world frame

内力も未知であるために，親リンクから考えていく場合には関節間内力が計算できず，末端リンクから考える場合には速度・加速度が分からないという問題が起こる．したがって，全リンクの力の釣り合いの式を連立させた問題を解く必要があり計算量は $\mathcal{O}(N^3)$ になってしまう．

ところが Featherstone[93] らは，親リンクとそれに連なる子リンクすべての釣り合いを考えることで $\mathcal{O}(N)$ の手続きが可能となることを発見した（図 4.15）．それによると親リンク i と全子リンク k，それらをまとめたリンク全体の運動方程式は以下式（4.46）の形式で子リン

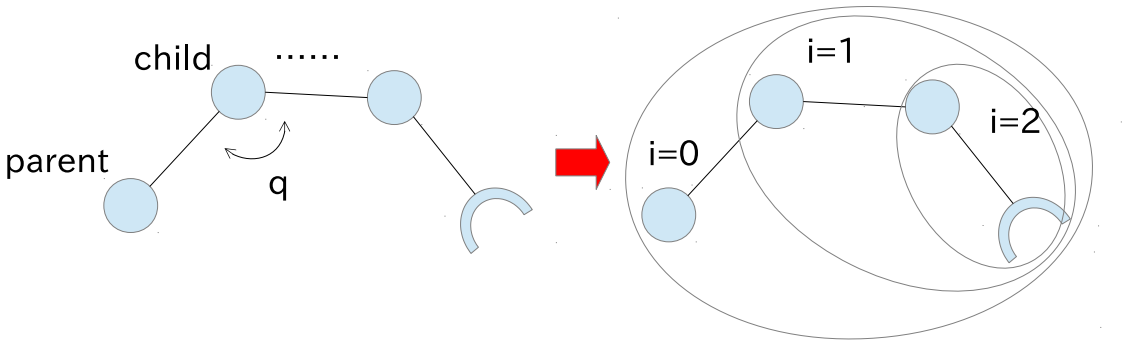


図 4.15. The idea of articulated body algorithm. Left is tree structure, and right is articulated body.

クとの内力を排除した形式で表現しなおせる．

$$\boldsymbol{y}_i = M_i^A \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i^A \tag{4.46}$$

また以下の記号式（4.47）を新たに導入する．

$$\begin{aligned} \boldsymbol{y}_i - \sum_k \boldsymbol{y}_k &= M_i\ \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i \\ \ddot{\boldsymbol{x}}_k &= \ddot{\boldsymbol{x}}_i + \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + \dot{\boldsymbol{s}}_k \ddot{q}_k \\ \tau_i &= \dot{\boldsymbol{s}}_i{}^T \boldsymbol{y}_i \end{aligned}$$

式（4.47）と前章までに求めた式の対応は以下のようになる．まず式（4.34）から，式（4.47）が成り立つ．

$$\begin{aligned} \ddot{\boldsymbol{x}}_i &= \begin{pmatrix} \dot{\boldsymbol{u}}_i \\ \dot{\boldsymbol{w}}_i \end{pmatrix} \\ \dot{\boldsymbol{s}}_i &= \begin{pmatrix} \Delta \boldsymbol{u}_i \\ \Delta \boldsymbol{w}_i \end{pmatrix} \\ \ddot{\boldsymbol{s}}_i &= \begin{pmatrix} \Delta \dot{\boldsymbol{u}}_i \\ \Delta \dot{\boldsymbol{w}}_i \end{pmatrix} \end{aligned} \tag{4.47}$$

つぎに式（4.44）から，式（4.48）が成り立つ．

$$\begin{aligned} \boldsymbol{y}_i &= \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{n}_i \end{pmatrix} \\ M_i &= \begin{pmatrix} m_i E & -m_i[\boldsymbol{c}_i\times] \\ m_i[\boldsymbol{c}_i\times] & I_i \end{pmatrix} \\ \boldsymbol{z}_i &= \begin{pmatrix} -m_i \boldsymbol{g} - \boldsymbol{f}_{ei} + \boldsymbol{w}_i \times \boldsymbol{P}_i \\ -m_i \boldsymbol{c}_i \times \boldsymbol{g} - \boldsymbol{n}_{ei} + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{w}_i \times \boldsymbol{L}_i \end{pmatrix} \end{aligned} \tag{4.48}$$

以上を用いて関節負荷トルク τ と関節角加速度 $\ddot{q}$ の関係を求める．τ について式（4.49）の

ように変換する．

$$\begin{aligned}\tau_k &= \dot{\boldsymbol{s}}_k{}^T \boldsymbol{y}_k \\ &= \dot{\boldsymbol{s}}_k{}^T M_k^A \ddot{\boldsymbol{x}}_k + \dot{\boldsymbol{s}}_k{}^T \boldsymbol{z}_k^A \\ &= \dot{\boldsymbol{s}}_k{}^T M_k^A \ddot{\boldsymbol{x}}_i + \dot{\boldsymbol{s}}_k{}^T M_k^A \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + \dot{\boldsymbol{s}}_k{}^T I_k^A \dot{\boldsymbol{s}}_k \ddot{q}_k + \dot{\boldsymbol{s}}_k{}^T \boldsymbol{z}_k^A\end{aligned} \tag{4.49}$$

これを $\ddot{q}_k$ について解くと式（4.50）のようになる．

$$\begin{aligned}\ddot{q}_k &= \left(\tau_k - \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{x}}_i - \eta_k\right) \iota_k \\ \eta_k &= \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{s}}_k \dot{q}_k + \dot{\boldsymbol{s}}_k{}^T \boldsymbol{z}_k^A \\ \boldsymbol{\zeta}_k^T &= \dot{\boldsymbol{s}}_k{}^T M_k^A \\ \iota_k &= \left(\boldsymbol{\zeta}_k^T \dot{\boldsymbol{s}}_k\right)^{-1}\end{aligned} \tag{4.50}$$

以上から $M_i^A, \boldsymbol{z}_i^A$ の再帰的な表現である式（4.51）が得られる．

$$\begin{aligned}\boldsymbol{y}_i &= M_i^A \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i^A \\ &= \boldsymbol{y}_i - \sum_k \boldsymbol{y}_k + \sum_k \boldsymbol{y}_k \\ &= M_i \ddot{\boldsymbol{x}}_i + \sum_k M_k^A \ddot{\boldsymbol{x}}_k + \boldsymbol{z}_i + \sum_k \boldsymbol{z}_k^A \\ &= M_i \ddot{\boldsymbol{x}}_i + \sum_k M_k^A \left(\ddot{\boldsymbol{x}}_i + \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + \dot{\boldsymbol{s}}_k \ddot{q}_k\right) + \boldsymbol{z}_i + \sum_k \boldsymbol{z}_k^A \\ &= \left(M_i + \sum_k M_k^A\right) \ddot{\boldsymbol{x}}_i + \boldsymbol{z}_i + \sum_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k\right) \\ &\quad + \sum_k M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \boldsymbol{\zeta}_k^T \ddot{\boldsymbol{x}}_i - \eta_k\right) \iota_k \\ &= \left(M_i + \sum_k \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right)\right) \ddot{\boldsymbol{x}}_1 \\ &\quad + \boldsymbol{z}_i + \sum_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \eta_k\right) \iota_k\right)\end{aligned} \tag{4.51}$$

まとめると

$$\begin{aligned}M_i^A &= M_i + \sum_k \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right) \\ \boldsymbol{z}_i^A &= \boldsymbol{z}_i + \sum_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \eta_k\right) \iota_k\right)\end{aligned} \tag{4.52}$$

末端リンクでは，$M_i^A = M_i, \boldsymbol{z}_i^A = \boldsymbol{z}_i$ と考えられるので，式（4.52）は末端リンクから遡って計算することで全リンクの $M_i^A, \boldsymbol{z}_i^A$ を計算することができ，関節角加速度も計算できる．以上をまとめたアルゴリズムを Algorithm. 9 に示す．手続き ForwardDynamics はルートリンクを引数とし子リンクをたどっていって末端リンクが見つかるとそこから逆順に順動力学計算に必要なダイナミクス情報を計算しながらルートリンクへと戻り，今度はルートリンクから末端リンクに向けて関節角加速度と空間加速度を計算する．計算では重心位置等を用いるので，順動力学計算の前に順運動学計算を行っておく必要もある．

Algorithm 9 Pseudo code of forward dynamics

Require: i

i: joint id.

Procedure: ForwardDynamics(i)

updateFD(i)

$\ddot{\boldsymbol{x}}_i = -\left(M_i^A\right)^{-1} \boldsymbol{x}_i^A$

for all k in childrens(i) **do**

  mainFD(k, i)

end for

Procedure: updateFD(i)

for all k in childrens(i) **do**

  updateFD(k)

end for

$\boldsymbol{P}_i = m_i \boldsymbol{u}_i + m_i \boldsymbol{w}_i \times \boldsymbol{c}_i$

$\boldsymbol{L}_i = m_i \boldsymbol{c}_i \times \boldsymbol{u}_i + I_i \boldsymbol{w_i}$

$$M_i = \begin{pmatrix} m_i E & -m_i[\boldsymbol{c}_i\times] \\ m_i[\boldsymbol{c}_i\times] & I_i \end{pmatrix}$$

$$\boldsymbol{z}_i = \begin{pmatrix} -m_i \boldsymbol{g} - \boldsymbol{f}_{ei} + \boldsymbol{w}_i \times \boldsymbol{P}_i \\ -m_i \boldsymbol{c}_i \times \boldsymbol{g} - \boldsymbol{n}_{ei} + \boldsymbol{u}_i \times \boldsymbol{P}_i + \boldsymbol{w}_i \times \boldsymbol{L}_i \end{pmatrix}$$

$$\dot{\boldsymbol{s}}_i = \begin{pmatrix} \Delta \boldsymbol{u}_i \\ \Delta \boldsymbol{w}_i \end{pmatrix}$$

$$\ddot{\boldsymbol{s}}_i = \begin{pmatrix} \Delta \dot{\boldsymbol{u}}_i \\ \Delta \dot{\boldsymbol{w}}_i \end{pmatrix}$$

$\boldsymbol{\zeta}_i^T = \dot{\boldsymbol{s}}_i{}^T M_i^A$

$\iota_i = \left(\boldsymbol{\zeta}_i^T \dot{\boldsymbol{s}}_i\right)^{-1}$

$M_i^A = M_i + \sum_k \left(M_k^A - M_k^A \dot{\boldsymbol{s}}_k \boldsymbol{\zeta}_k^T \iota_k\right)$

$\boldsymbol{z}_i^A = \boldsymbol{z}_i + \sum_k \left(\boldsymbol{z}_k^A + M_k^A \ddot{\boldsymbol{s}}_k \dot{\boldsymbol{q}}_k + M_k^A \dot{\boldsymbol{s}}_k \left(\tau_k - \eta_k\right) \iota_k\right)$

$\eta_i = \boldsymbol{\zeta}_i^T \ddot{\boldsymbol{s}}_i \dot{q}_i + \dot{\boldsymbol{s}}_i{}^T \boldsymbol{z}_i^A$

Procedure: mainFD(i, j)

$\ddot{q}_i = \left(\tau_i - \boldsymbol{\zeta}_i^T \ddot{\boldsymbol{x}}_j - \eta_i\right) \iota_i$

$\ddot{\boldsymbol{x}}_i = \ddot{\boldsymbol{x}}_j + \ddot{\boldsymbol{s}}_i \dot{\boldsymbol{q}}_i + \dot{\boldsymbol{s}}_i \ddot{q}_i$

for all k in childrens(i) **do**

  mainFD(k, i)

end for

順動力学計算により得られた加速度情報は，積分することで速度と位置の計算に用いられる．Taylor 展開に置ける一次近似，すなわち単純に加速度に時間をかけた値だけ速度が変化するという計算法は Euler 法として知られている．ルートリンクの速度を積分し位置を更新する際には，空間速度から速度を計算する必要があることに注意が必要である．

Algorithm 10 Pseudo code of euler method

Require: i, Δt

i: joint id.

Δt: simulation time step.

Procedure: UpdateEuler(i, Δt)

$\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i \Delta t$

$\boldsymbol{v}_i = \boldsymbol{v}_i + \dot{\boldsymbol{v}}_i \Delta t$

$R_i = e^{\boldsymbol{w}_i \Delta t} R_i$

$\boldsymbol{w}_i = \boldsymbol{w}_i + \dot{\boldsymbol{w}}_i \Delta t$

for all k in childrens(i) **do**

　procEuler(k, Δt)

end for

Procedure: procEuler(i, Δt)

$q_i = q_i + \dot{q}_i \Delta t$

$\dot{q}_i = \dot{q}_i + \ddot{q}_i \Delta t$

for all k in childrens(i) **do**

　procEuler(k, Δt)

end for

4.3.1.5　接触判定と接触力計算

本章で用いる記号を表 4.6 にまとめる．

メッシュデータで表現される凸な剛体同士の衝突計算は，両剛体内を動く二つの点の距離を最小化するという二次計画問題として定式化でき厳密に解くことができることが知られている [94][95]．また衝突時の接触力計算も摩擦や地面とのめり込みといった不等式条件を考慮する必要があるため探索問題として解く方法がしばしばとられる．しかしながら，ロボットのシミュレーションでは，シミュレーションに含まれる膨大なパラメタを実機に合わせる必要があり，過度にモデルに厳密な計算を行うよりは，実機の挙動を含みうる単純なモデルを用いて高速に計算できることがより重要であると考える．本論文では，接触リンク凸形状の各頂点のみと無限に広がる平面の接触を考え，接触時の接触力は環境へのめり込み量を用いた spring dumper モデルを用いる．頂点のみの衝突を考えるため接触リンクより小さな平面を持つ物体との衝突を扱うことはできない．

今，i 番目の関節リンクの凸形状各頂点の j 番目の位置を $\boldsymbol{p}_{vj}$，関節との相対位置を $\boldsymbol{p}_{vij}$ と

Table4.6. All parameters used in this forward-dynamics section

$\boldsymbol{v}_{vj}$	j-th mesh velocity in world frame
$\boldsymbol{p}_{vj}$	j-th mesh vertice in world frame
$\boldsymbol{p}_{vj0}$	previous j-th mesh vertice in world frame
$\boldsymbol{p}_{vj00}$	collided j-th mesh vertice in world frame
b_{wj}	distance between w-th wall and j-th vertice
$\boldsymbol{n}_w$	unit normal vector of w-th wall in world frame
$\boldsymbol{f}_{vj}$	contact force of j-th vertice in world frame
$\boldsymbol{k}_j$	spring gain of j-th vertice force calculation
$\boldsymbol{d}_j$	dumper gain of j-th vertice force calculation
$\boldsymbol{f}_{fj}$	friction force of j-th vertice in world frame
$\boldsymbol{k}_{fj}$	spring gain of j-th vertice friction calculation
$\boldsymbol{d}_{fj}$	dumper gain of j-th vertice friction calculation
μ_j	friction coefficient of j-th vertice
$\boldsymbol{b}_{fwj}$	friction direction of j-th vertice in world frame

呼ぶことにすると，

$$\boldsymbol{p}_{vj} = \boldsymbol{p}_i + R_i \boldsymbol{p}_{vij} \tag{4.53}$$

無限に広がる面 w 上のと一点 $\boldsymbol{p}_w$ と法線単位ベクトル $\boldsymbol{n}_w$ が与えられたとき，点 j と面 w の距離 b_{wj} は

$$b_{wj} = \boldsymbol{n}_w^T \left(\boldsymbol{p}_{vj} - \boldsymbol{p}_w \right) \tag{4.54}$$

b_{wj} が負であるとき衝突が起こっている．このときの接触力を spring dumper モデルで計算する．バネ定数 k_j とダンパ係数 d_j，点 j の速度ベクトル $\boldsymbol{v}_{vj}$ を用いて

$$\begin{aligned} \boldsymbol{f}_{vj} &= ||\boldsymbol{f}_{vj}|| \boldsymbol{n}_w \\ ||\boldsymbol{f}_{vj}|| &= max\left(0, -\left(k_j b_{wj} + d_j \boldsymbol{n}_w^T \boldsymbol{v}_{vj}\right)\right) \end{aligned} \tag{4.55}$$

この力 $\boldsymbol{f}_{vj}$ は点と面の接触により点が面に押し返される力を表しており，常に面の法線ベクトルと同じ向きに働く力であることに注意が必要である．速度 $\boldsymbol{v}_{vj}$ は単位時間 Δt 前の点 j の位置 $\boldsymbol{p}_{vj0}$ を用いて

$$\boldsymbol{v}_{vj} = \frac{\left(\boldsymbol{p}_{vj} - \boldsymbol{p}_{vj0}\right)}{\Delta t} \tag{4.56}$$

次に摩擦力を考慮する．単位時間前の距離 b_{wj0} が負で衝突が起こっており，一番最初に衝突の起こった点 j の位置 $\boldsymbol{p}_{vj00}$，摩擦力に関する spring dumper 係数を k_{fj}，d_{fj} と定義する

とき摩擦力を以下のように計算する．

$$\begin{aligned} \boldsymbol{f}_{fj} &= -\left(k_{fj}||\boldsymbol{b}_{fwj}|| + d_{fj}\boldsymbol{n}_{fwj}^T\boldsymbol{v}_{vj}\right)\boldsymbol{n}_{fwj} \\ \boldsymbol{n}_{fwj} &= \frac{\boldsymbol{b}_{fwj}}{||\boldsymbol{b}_{fwj}||} \\ \boldsymbol{b}_{fwj} &= \boldsymbol{b}_{fwj}^{\perp} - \boldsymbol{n}_w^T\boldsymbol{b}_{fwj}^{\perp}\boldsymbol{n}_w \\ \boldsymbol{b}_{fwj}^{\perp} &= \boldsymbol{p}_{vj} - \boldsymbol{p}_{vj00} \end{aligned} \tag{4.57}$$

さらに摩擦係数 μ_j を越える力が加わったとき，摩擦力の原点 $\boldsymbol{p}_{vj00}$ を更新し，摩擦力の大きさを摩擦係数に合わせる．

$$\begin{aligned} \boldsymbol{p}_{vj00} &\leftarrow \boldsymbol{p}_{vj} \\ \boldsymbol{f}_{fj} &\leftarrow \frac{\mu_j||\boldsymbol{f}_{vj}||}{||\boldsymbol{f}_{fj}||}\boldsymbol{f}_{\boldsymbol{fj}} \end{aligned} \tag{4.58}$$

以上の手続きを擬似コードで書くと，

4.3.1.6　センサシミュレーション

本章で用いる記号を表 4.7 にまとめる．

Table4.7. All parameters used in this forward-dynamics section

$\boldsymbol{p}_s$	sensor position in world frame
R_s	sensor orientation in world frame
$\boldsymbol{p}_{ps}$	sensor position in parent link frame
R_{ps}	sensor orientation in parent link frame
$\boldsymbol{u}$	parent link spacial velocity in world frame
$\boldsymbol{v}$	parent link velocity in world frame
$\boldsymbol{w}$	parent link angular velocity in world frame
$\boldsymbol{p}$	parent link position in world frame
R	parent link orientation in world frame
$\boldsymbol{e}_x$	$[0; 0; 1]$
$\boldsymbol{g}$	gravity $= [0; 0; 9.80665]$
$\boldsymbol{s}_m$	magnetic sensor output in sensor frame
$\boldsymbol{s}_g$	gyro sensor output in sensor frame
$\boldsymbol{s}_a$	acceleration sensor output in sensor frame

センサの位置姿勢は親リンクの位置姿勢と定数である相対位置姿勢を用いることで以下式（4.59）のように計算できる．

$$\begin{aligned} \boldsymbol{p}_s &= \boldsymbol{p} + R\boldsymbol{p}_{ps} \\ R_s &= RR_{ps} \end{aligned} \tag{4.59}$$

Algorithm 11 Pseudo code of contact force calculation

Require: i, Δt

i: joint id.

Δt: simulation time step.

Procedure: CalcForce(i, Δt)

procCF(i, Δt)
$\boldsymbol{f}_i = \sum_j \boldsymbol{f}_{vj}$
$\boldsymbol{n}_i = \sum_j \boldsymbol{p}_{vj} \times \boldsymbol{f}_{vj}$
for all k in childrens(i) **do**
  CalcForce(k, Δt)
end for

Procedure: procCF(i, Δt)

for all j in vertices(i) **do**
  $\boldsymbol{p}_{vj0} = \boldsymbol{p}_{vj}$
  $\boldsymbol{p}_{vj} = \boldsymbol{p}_i + R_i \boldsymbol{p}_{vij}$
  $\boldsymbol{v}_{vj} = \frac{(\boldsymbol{p}_{vj} - \boldsymbol{p}_{vj0})}{\Delta t}$
  $b_{wj0} = b_{wj}$
  $b_{wj} = \boldsymbol{n}_w^T (\boldsymbol{p}_{vj} - \boldsymbol{p}_w)$
  if $b_{wj} \leq 0$ **then**
    $||\boldsymbol{f}_{vj}|| = max\left(0, -\left(k_j b_{wj} + d_j \boldsymbol{n}_w^T \boldsymbol{v}_{vj}\right)\right)$
    $\boldsymbol{f}_{vj} = ||\boldsymbol{f}_{vj}|| \boldsymbol{n}_w$
    if $b_{wj0} \leq 0$ **then**
      $\boldsymbol{b}_{fwj}^{\perp} = \boldsymbol{p}_{vj} - \boldsymbol{p}_{vj00}$
      $\boldsymbol{b}_{fwj} = \boldsymbol{b}_{fwj}^{\perp} - \boldsymbol{n}_w^T \boldsymbol{b}_{fwj}^{\perp} \boldsymbol{n}_w$
      if $\epsilon \leq ||\boldsymbol{b}_{fwj}||$ **then**
        $\boldsymbol{n}_{fwj} = \frac{\boldsymbol{b}_{fwj}}{||\boldsymbol{b}_{fwj}||}$
        $\boldsymbol{f}_{fj} = -\left(k_{fj} ||\boldsymbol{b}_{fwj}|| + d_{fj} \boldsymbol{n}_{fwj}^T \boldsymbol{v}_{vj}\right) \boldsymbol{n}_{fwj}$
        if $||\boldsymbol{f}_{vj}|| \mu_j \leq ||\boldsymbol{f}_{fj}||$ **then**
          $\boldsymbol{p}_{vj00} = \boldsymbol{p}_{vj}$
          $\boldsymbol{f}_{fj} = \frac{\mu_j ||\boldsymbol{f}_{vj}||}{||\boldsymbol{f}_{fj}||} \boldsymbol{f}_{\boldsymbol{fj}}$
        end if
        $\boldsymbol{f}_{vj} = \boldsymbol{f}_{vj} + \boldsymbol{f}_{fj}$
      end if
    else
      $\boldsymbol{p}_{vj00} = \boldsymbol{p}_{vj}$
    end if
  end if
end for

姿勢センサから見た，磁気センサ，ジャイロセンサ，加速度センサ，の出力はワールド座標系での出力をセンサ座標系での値に変換すればよい．姿勢センサの値は位置に依存しない値であるため単純に姿勢行列の逆をかければよく以下の式（4.60）で求めることができる．

$$\begin{aligned} \boldsymbol{s}_m &= R_s^T \boldsymbol{e}_x \\ \boldsymbol{s}_g &= R_s^T \boldsymbol{w} \\ \boldsymbol{s}_a &= R_s^T \left(\ddot{\boldsymbol{p}}_s + \boldsymbol{g} \right) \end{aligned} \tag{4.60}$$

加速度センサの値は，空間速度，空間加速度を用いて以下式（4.61）のように計算できる．

$$\begin{aligned} \dot{\boldsymbol{p}}_s &= \boldsymbol{v} + \boldsymbol{w} \times (\boldsymbol{p}_s - \boldsymbol{p}) \\ &= \boldsymbol{u} + \boldsymbol{w} \times \boldsymbol{p} + \boldsymbol{w} \times (\boldsymbol{p}_s - \boldsymbol{p}) \\ &= \boldsymbol{u} + \boldsymbol{w} \times \boldsymbol{p}_s \\ \ddot{\boldsymbol{p}}_s &= \dot{\boldsymbol{u}} + \dot{\boldsymbol{w}} \times \boldsymbol{p}_s + \boldsymbol{w} \times \dot{\boldsymbol{p}}_s \\ &= \dot{\boldsymbol{u}} + \dot{\boldsymbol{w}} \times \boldsymbol{p}_s + \boldsymbol{w} \times \boldsymbol{u} + \boldsymbol{w} \times (\boldsymbol{w} \times \boldsymbol{p}_s) \end{aligned} \tag{4.61}$$

4.3.1.7　速度評価実験

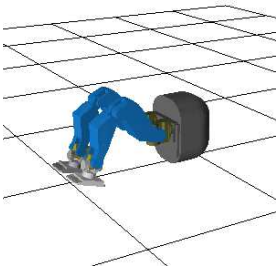


図 4.16. CHIDORI.

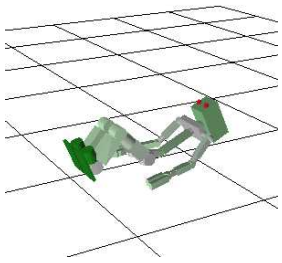


図 4.17. SampleRobot.

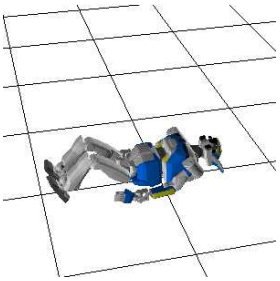


図 4.18. HRP2JSK.

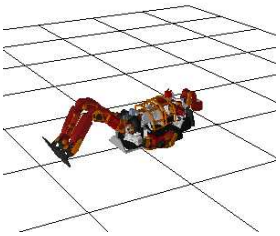


図 4.19. JAXON.

Intel(R) Core(TM) i7-4770S CPU 3.10GHz を用いて HRP2JSK [66]，JAXON [96]，千鳥 [97] ，SampleRobot [98] を高さ 100mm から落下させ地面に崩れ落ちるシミュレーションを行うことで速度を測定した（図 4.16，4.17，4.18，4.19）．速度の比較と関節自由度をまとめた結果を表 4.8 に示す．

Table4.8. Falling down simulation for CHIDORI, Samplerobot, HRP2JSK, STARO, and JAXON to compare computation time.

CHIDORI	SampleRobot	HRP2JSK	JAXON
12 dof	29 dof	32 dof	33 dof
59 kHz	35 kHz	22 kHz	22 kHz

脚型ロボット CHIDORI が 十二自由度である他は全身の関節角度を含むため三十自由度自由度前後であり，自由度が多いほど計算速度が遅くなっていることが見て取れる．これは順動

力学計算がオーダー N の計算量を持つこと，接触力計算が接触リンク数に応じて計算量が変化しその最悪計算量がオーダー N であることから妥当な結果であると言える．動力学シミュレータの計算速度比較は [99] に詳しい．それによると二十五自由度の人型ロボットが脱力して寝そべるシミュレーションでは従来のシミュレータを用いて $10 \approx 15$ kHz 程度での計算が可能であることが示されている．本章のシミュレータは計算オーダ N であるため，計算時間が $\alpha N + \beta$ で表せるとすると計算周期は $1/(\alpha N + \beta)$ であるので表 4.8 の最小自由度のロボット CHIDORI と最大自由度のロボット JAXON を用いて以下式（4.62）が成り立つ．

$$\begin{aligned} 12\alpha + \beta &= \frac{1}{59} \\ 33\alpha + \beta &= \frac{1}{22} \end{aligned} \tag{4.62}$$

式（4.62） を解くことで α と β が計算でき，$N = 25$ のときの計算時間が予測できる．

$$25\alpha + \beta \approx \frac{1}{28.9} \tag{4.63}$$

$$\begin{aligned} \alpha &= \frac{1}{33-12}\left(\frac{1}{22} - \frac{1}{59}\right) \\ &= 0.00136 \\ \beta &= \frac{1}{59} - \frac{12}{33-12}\left(\frac{1}{22} - \frac{1}{59}\right) \\ &= 0.00066 \end{aligned} \tag{4.64}$$

従来シミュレータより二倍から三倍程度高速であることが分かる．この比較は本章のシミュレータが干渉計算をリンクのバウンディングボックス等少数の接触点についてしか考慮していない点で不公平である．[99] では多自由度リンクによる把持シミュレーションも可能な接触力計算を行っているが本章の手法でそれを行うにはリンクの接触点数を増やす必要がありその場合の速度比較はまた違った結論に成りうる．しかし歩行シミュレーションなど接触が比較的単純な状況をシミュレーションする場合においては本章のシミュレーションは高速であることが示されたと考える．

本章で紹介した動力学シミュレータは高速であるが，さらなる高速化も可能であると考えている．以下にそのためのアイデアをまとめる．

- 十二自由度程度の脚型ロボットでは，順動力学計算を Featherstone らの方法ではなく運動方程式を書き下し吐き出し法などを用いて計算するほうが高速である可能性がある．しかし全身の含まれるロボットでは自由度が三十以上となることが多く今後の拡張を見越して前者の方法を用いている．
- 空間速度を用いず通常の速度を用いたほうが順運動学は高速化されると試算している．その場合は順動力学計算で空間速度を用いている都合，全体の計算量を見直す必要があり，より深い考察が必要である．
- リンクの回転姿勢を行列で表現しているが，これをクオータニオンにすることで姿勢の

計算に関しては高速化が見込める．位置の計算も踏まえてどちらが高速であるかはより深い考察が必要である．

4.3.2　動歩行探索のための軽量な歩行モデル

4.3.2.1　既存の動歩行モデルのまとめ

動歩行生成のアルゴリズムは数多く存在するが，離散化や多項式近似を用いてコンフィギュレーション空間を有限のパラメタ空間で近似し最適化問題を解く手法（以下最適化アプローチ，direct collocation とも言う）と，なんらかの歩行モデルを与えそのパラメタを動力学シミュレーション等を用いて決定する方法（以下モデル学習アプローチ，direct shooting とも言う）にその多くが大別できる．これらとはやや異なるが，受動歩行の研究では歩行を周期的な物理現象として捉え，ある状態から同じパラメタを持つ別の状態への遷移を繰り返すというモデル（リミットサイクル）を用いて歩行を生成するものもある．それらについて以下に簡単なまとめを行う．

4.3.2.1.1　最適化アプローチ

重心軌道 [46][47] や全身軌道 [100] を離散化や多項式近似により表現し最適化手法を用いて軌道を決定する．これらの手法は高速であり，特に重心軌道のみを対象とした最適化計算はリアルタイムな歩行生成に適しているため広く用いられている．一方で転倒が起こらないような条件として Zero Moment Point （以下 ZMP）が足裏支持領域に含まれるという条件を用いるためあらかじめ足平をどこに配置していくかという問題（フットステッププランニング）を解いておく必要がある．また踵から接地し踵を軸に足平を転がして... といった具合に人間は歩行を行うが，このような動作を行うには，最適化計算においてあらかじめそのような制約をかすことで実現が可能であるが，足を転がすのが良いのか，それともベタ足でいくのかといった探索までも含めた解き方は難しい．

4.3.2.1.2　モデル学習アプローチ

歩行動作を行うモデルをあらじめ定義しておき，そのパラメタを動力学シミュレーションや人間のチューニングにより決定することで歩行を行う．モデルの表現を簡潔にしすぎると単純な動作しか生成できないが，逆に複雑にしてしまうと次元の呪いにより探索に膨大な時間を要してしまうという問題がある．一方で最適化手法，特に勾配を用いた局所探索アルゴリズムでは解くことの難しい不連続なモデルや条件を考慮することが可能であり，if-then ルールで記述された制御則や関節負荷トルク制限といった非線形不等式も扱うことが容易にできる．さらに足平の着地位置，着地方法などに仮定を置かないためそれらも探索により決定することが可能である．歩行モデルの例を二つ以下に示す．

4.3.2.1.3 CPG モデル

生物の脊髄には周期パターンを生成する神経系（CPG: Central Pattern Generator）が存在し，歩行といった周期運動に用いられていることが知られている [101]．CPG を用いて歩行行動を実現した研究としては [102] [103] [104] がある．

4.3.2.1.4 SIMBICON モデル

SIMBICON （Simple Biped Locomotion Control [42]）は静的な目標姿勢を重心位置速度に応じたフィードバックとともに遷移するステートマシーンで歩行を表現するモデルであり，歩行に限らず走行やスキップ，バク宙といった多様な行動を動力学シミュレーションと人間のチューニングにより表現できることが示されている．またロボットの歩行にも応用されはじめている [105]．

4.3.2.1.5 リミットサイクル アプローチ

足が点接触であるようなモデルを考える．運動方程式は力によって加速度を決定し，力は足（点）と地面の非弾性衝突及び重力との釣り合いによって一意に定まる．したがってある時刻におけるロボットの接触状態，各関節の角度と角速度が与えられることで，力が計算でき加速度が計算できそれを積分することで速度・位置が計算できるといった具合にその後の歩行を一意に計算することができる．この接触状態・関節角度・関節角速度を選択し動力学シミュレーションを行い同じ接触状態・関節角度・関節角速度が再び現れるよう（リミットサイクル）探すことで歩行軌道を生成する手法が存在する [106]．また安定な歩行を行うためには，得られたリミットサイクルが安定であることを確認する必要がある．

4.3.2.1.6 探索性能と計算時間からみた歩行生成手法の比較

計算時間を度外視すればモデル学習アプローチのほうが最適化アプローチよりも高い性能をもった歩行生成が行えると考えられる．前述のとおり最適化アプローチは足を転がすといった複雑な接触遷移を表現することや， if-then ルールが含まれるような制御則を含めることが困難である．[36] のような方法でフットステッププランニングに加えて接触遷移方式を探索することも可能であるが，微分不能な制御則等の考慮は，SQP といった高速な局所探索アルゴリズムの利用を難しくし，最適化アプローチの高速性を損なうものである．

モデル学習アプローチを用い制御パラメタまでも含む探索を行う際の課題は，問題の規模を解ける範囲まで小さくすることであり，そのために必要なものはなるべく軽量なモデルである．本章では歩行モデルとして CPG を用いるものと SIMBICON を用いるものについて述べたが，モデルパラメタの数で比較すると差はなくどちらでも問題ない．以下ではモデルパラメタが人間にも理解しやすい形式であること，フィードバックがセンサデータに対する一次の項であり二次以降への拡張が直感的に可能であること，イベント駆動式ステートマシンであり歩行からの停止や回転といった拡張が容易であるという理由から SIMBICON を用いるが

学習アルゴリズムは進化計算に基づくものであるため CPG を用いても同様に歩行を生成できる．

4.3.2.2　ステートマシン型歩行モデル

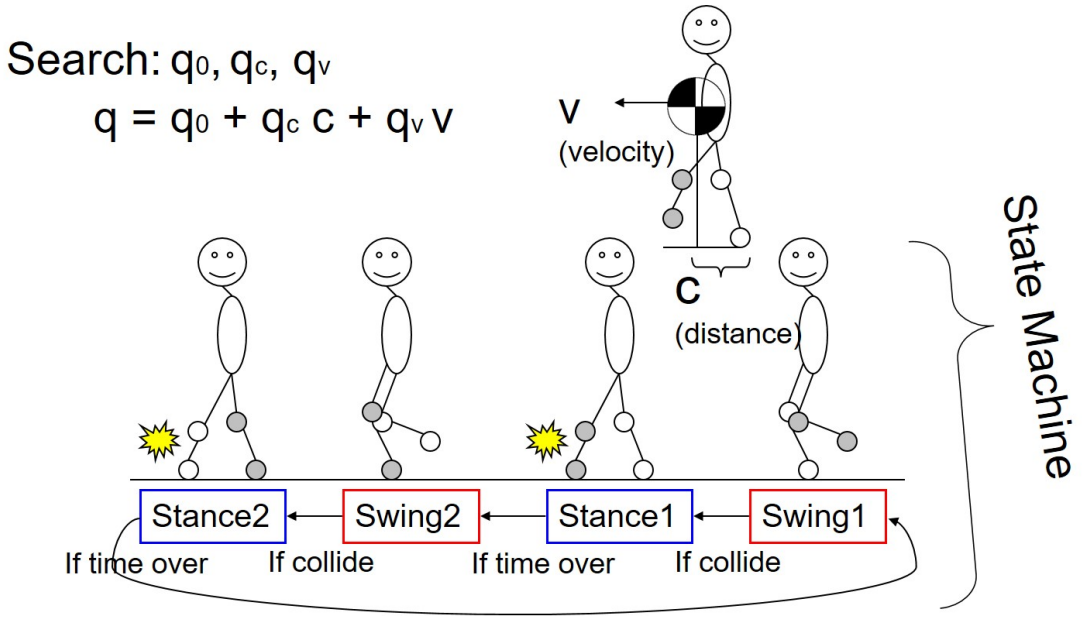


図 4.20. SIMBICON [42] is a simple walk control model which has an definite number of states and control joints to balance based on a linear model of COG position/velocity.

図 4.20 に 歩行モデル SIMBICON [42] の模式図を示す．SIMBICON は有限の姿勢をステートとする有限要素機械であり，ステート間の移動において重心の位置速度を用いて線形にバランス制御を行う．ステートの遷移には接触判定と時間を用い，遊脚の力センサの出力が閾値を越えるか，制限時間内にそれが起こらなかった場合に次のステートへと遷移する．フィードバックを式で表すと，重心の支持脚からの距離を $\boldsymbol{c}$，重心の速度を $\boldsymbol{v}$ として，ステートが持つ目標関節角度 $\boldsymbol{q}_0$，フィードバックゲイン $\boldsymbol{q}_c, \boldsymbol{q}_v$ を用いて式（4.65）のように計算する．

$$\boldsymbol{q} = \boldsymbol{q}_0 + \boldsymbol{q}_c \boldsymbol{c} + \boldsymbol{q}_v \boldsymbol{v} \tag{4.65}$$

重心位置速度 c, v は並進二自由度を用いる．したがってフィードバックゲイン $\boldsymbol{q}_c, \boldsymbol{q}_v$ は関節自由度数 $\times 2$ の行列である．人間のチューニングにより，$\boldsymbol{q}_0, \boldsymbol{q}_c, \boldsymbol{q}_v$ を決定することで動力学シミュレーション上で様々な歩行を行えることが示されている．以上が元論文の実装についての概要であるが，以下ではさらに本研究で用いた拡張について述べる．

4.3.2.3　歩き初めと歩き終わりを考慮したステート遷移

本研究で用いたステートマシンを図 4.21 に示す．歩き初めと歩き終わりを表現するために，全部で十のステートを用いた．図左端の三つが歩きはじめを表現するためのステート，右端の

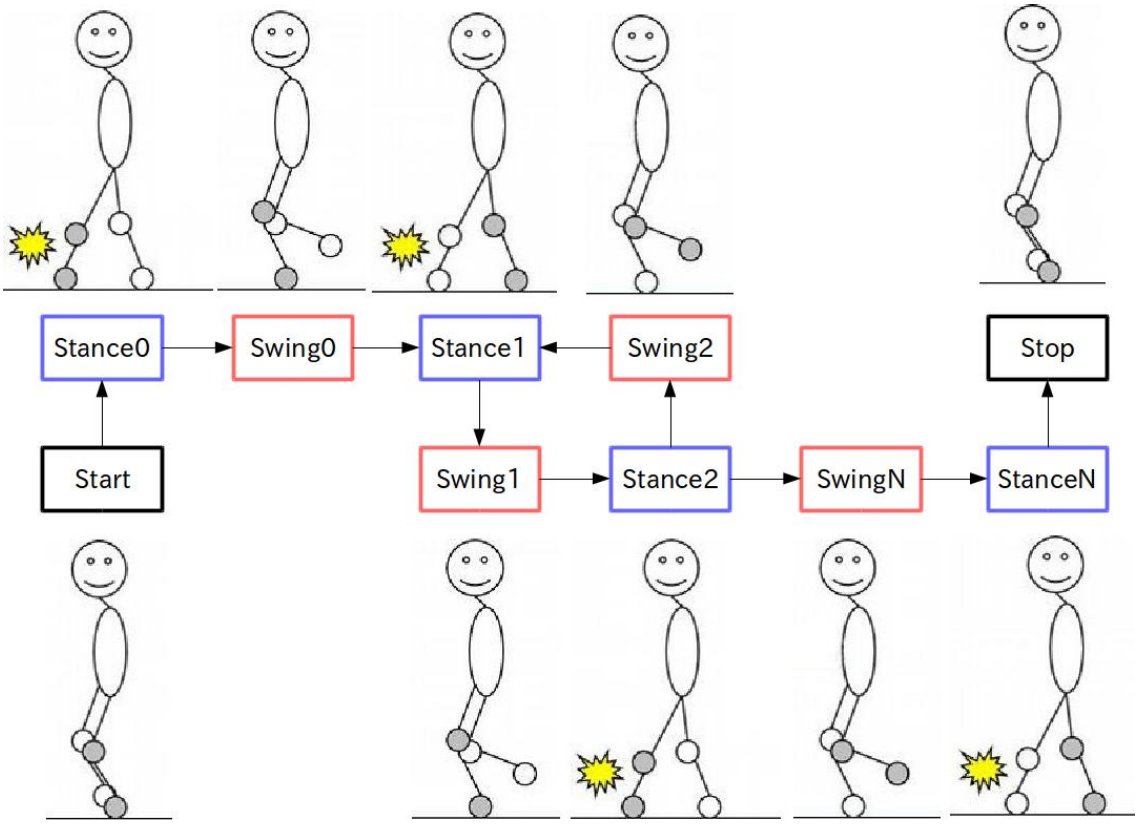


図 4.21. To consider start and stop sequence of walking, total 10 states are used; 3 for start walking, 3 for stop walking, and 4 states for walking loop.

三つが歩き終わりを表現するためのステート，そして中央の四つのステートが輪を描いて遷移することで歩行中のステートを表現する．また歩行のはじめとおわりの姿勢は与えられるものとし，さらに中央の歩行ループでは左右対称の動作に限定することでステート数を減らして用いる．すなわち歩き始めに二つ，歩行ループに二つ，歩き終わりに二つの計六つのステートで歩行を表現した．

4.3.2.4 位置制御のための線形補間

SIMBICON では式（4.65）で計算された目標関節角度に対して PD 制御で関節駆動トルクを決定しロボットを動作させるが，本研究ではハイゲインの電流 PD 制御により DC モータを位置制御するため，目標関節角度の補間が必要である．補間方法は単純な線形補間により $\boldsymbol{q}_0$ を補間する．あるステートに入ってから数えて時刻 t における関節角度 $\boldsymbol{q}(t)$ を，重心位置速度 $\boldsymbol{c}(t), \boldsymbol{v}(t)$，ステート遷移目標時間 Δt を用いて以下の式（4.66）で補間し位置制御の制御目標値として用いる．

$$\begin{aligned} \boldsymbol{q}(t) &= \boldsymbol{q}_0(t) + \boldsymbol{q}_c \boldsymbol{c}(t) + \boldsymbol{q}_v \boldsymbol{v}(t) \\ \boldsymbol{q}_0(t) &= (\boldsymbol{q}_0 - \boldsymbol{q}(0)) \frac{t}{\Delta t} + \boldsymbol{q}(0) \end{aligned} \tag{4.66}$$

4.3.2.5　ヨー回転のフードバック補償

足平のすべりによって体の向きが回転してしまっても，磁気センサやジャイロセンサを用いて向きが測定できていればそれを補償することができる．体幹リンクの進行方向正面から見た回転角度を $r(t)$，その速度を $w(t)$ として，新たなフィードバックゲイン $\boldsymbol{q}_r, \boldsymbol{q}_w$ を導入することで以下の 式（4.67）のように目標関節角度を更新する．

$$\begin{aligned} \boldsymbol{q}(t) &= \boldsymbol{q}_0(t) + \boldsymbol{q}_c \boldsymbol{c}(t) + \boldsymbol{q}_v \boldsymbol{v}(t) + \boldsymbol{q}_r r(t) + \boldsymbol{q}_w w(t) \\ \boldsymbol{q}_0(t) &= (\boldsymbol{q}_0 - \boldsymbol{q}(0)) \frac{t}{\Delta t} + \boldsymbol{q}(0) \end{aligned} \tag{4.67}$$

4.3.2.6　足平と体幹の回転補正

SIMBICON ではシミュレーション上で厳密なトルク制御ができることを利用して，股関節のトルクを体幹が回転しないように与えることで姿勢の維持を行っているが実際のトルク制御ロボットではモデル誤差を含むためこの方針は実現困難である．また位置制御ロボットではそもそもこの方針は使えない．そこで，ヤコビ行列と最急降下法を用いて体幹とさらに足平の回転を抑圧する項を加える．式（4.66）の目標値を用いて順運動学計算により体幹と足平の姿勢を計算する．体幹の水平面に対する傾きを $\boldsymbol{r}_r$ とし，支持脚関節の水平面に直行しない軸を持った二関節を含むヤコビ行列を J_r として関節角度 $\boldsymbol{q}(t)$ を以下のように補正する．

$$\boldsymbol{q}(t)^{'} = \boldsymbol{q}(t) + \alpha_r J_r^T(t) \boldsymbol{r}_r(t) \tag{4.68}$$

α_r は回転補正を行う重みゲインでありこれものちの探索に含める．足平の回転補正についても同様に行う．足の水平面に対する傾きを $\boldsymbol{r}_f$ とし，脚関節の水平面に直行しない軸を持った二関節を含むヤコビ行列を J_f として関節角度 $\boldsymbol{q}(t)$ を以下のように補正する．

$$\boldsymbol{q}(t)^{'} = \boldsymbol{q}(t) + \alpha_f J_f^T(t) \boldsymbol{r}_f(t) \tag{4.69}$$

4.3.2.7　受動軸の振りぬき速度を変更可能な遊脚軌道

膝関節の受動的な振る舞いを考えるときに，脚が線形に補間されるモデルでは足を振り抜くような動作が表現できないと考えられる．股関節の加減速により脚全体の慣性力を変化させることで膝下のリンクが放り投げられるような項をヒューリスティックに加える．

$$q_0(t) = \begin{cases} (q_1 - q(0)) \frac{t}{t_1} + q(0) & (t < t_1) \\ (q_0 - q_1) \frac{t - t_1}{\Delta t - t_1} + q_1 & (t \geq t_1) \end{cases} \tag{4.70}$$

図 4.22 に示すような折れ線にそって股関節ピッチ軸の補間を行う．折れ線の中点（q_1）を目標姿勢（q_0）から離れた値にすることで，一度ためて加速しながら素早く振りぬくことができる．逆に近い値を用いれば直線に近い遅い遊脚移動を表現できる．折れ線の中点 q_1 とその時刻 t_1 も探索に含める．

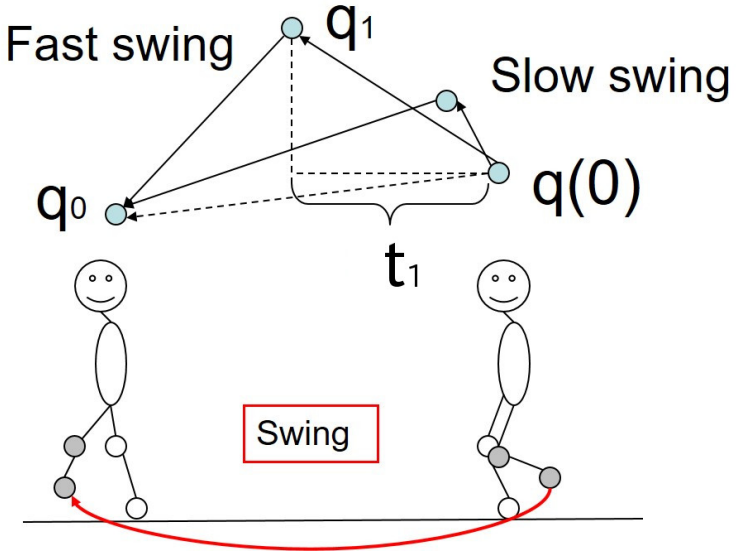


図 4.22. Polyline interpolation model to control swing velocity of passive joint.

4.3.2.8 重心位置計算に置ける姿勢推定アルゴリズム

動力学シミュレーションではロボットの姿勢を直接取得することができるが，実ロボットで測定できるのは加速度センサ，ジャイロセンサ，磁気センサなどであり，そこから姿勢推定を行う必要がある．動作の探索時と実ロボットの動作時では，この姿勢推定アルゴリズムを等しくしておく必要がある．以下では，Madgwick らの開発した最急降下法に基づく姿勢推定アルゴリズムと，そのオープンソース実装を用いる [107]．このアルゴリズムは加速度センサ，ジャイロセンサ，磁気センサの九次元ベクトルから姿勢をクオータニオン表現で計算するものであるが，磁気センサがない場合も同じソフトウェアで姿勢が推定できるよう実装してあるため，磁気センサがないロボットでも同様に用いることが可能である．

4.3.2.9 フィードバックゲインの関節軸方向による次元削減

これまでの拡張を全て用いた時の探索空間のパラメタ数を一度見積もってみる．$\boldsymbol{q}_0$ は全関節角度次元 N であり，$\boldsymbol{q}_c, \boldsymbol{q}_v$ はそれを水平面の二次元分でそれぞれ $2N$，$\boldsymbol{q}_r, \boldsymbol{q}_w$ はヨーまわりの一次元分でそれぞれ N，さらに股関節は膝下の振り抜き速度を調節するための q_1, t_1 が両足分でそれぞれ 2，体幹と足平の平面二次元 d ねお回転を抑制する α_r, α_f は両足を考慮すれば $2\times3=6$, 最後にステート間遷移時間 Δt の 1, 合わせて $N+2N+2N+N+N+2+2+6+1=7N+11$ 次元となる．さらにこれが全ステート分必要であることを考え，歩き始め 2 ステート，歩行中が $4/2=2$ ステート（左右対称な動作であるので半分），歩き終わりに 2 ステートとするとさらに $2+2+2=6$ 倍で $42N+66$ 次元のパラメタとなる．$N=12$ 自由度の脚型ロボットでも 570 次元程度の探索が必要である．この規模であっても一から三週間程度進化計算を行えば良い答えが得られることが確認できているが，実用上は長くとも数日以内に答

えが確認できることが望ましい．それが可能な規模は筆者の環境でおおよそ 100 次元程度である．

SIMBICON では探索空間を削減するための工夫として $\boldsymbol{q}_c, \boldsymbol{q}_v$ において前方向の転倒を防ぐ項では，遊脚股関節ピッチのみを考慮し，横方向についてはロール軸のみを考慮するといったヒューリスティックを用いている．全身の動力学を考えるとピッチ軸以外が前方向の転倒に影響を与えないということは誤りであるが主要項はピッチ軸であると考えられるため良い近似である．本研究でも同様の近似を用いることで探索空間の次元を制限する．まず前方向の転倒については遊脚の股関節ピッチ膝関節ピッチ，支持脚の足首ピッチの三軸を用いる．つぎに横方向の転倒については遊脚の股関節ロールと支持脚の足首ロールを用いる．最後に回転方向については支持脚股関節のヨー軸を用いる．以上は脚の関節構成が股関節にロールピッチヨー軸，膝関節にピッチ軸，足首関節にロールピッチ軸を少なくとも持つロボットに限った方法であり，これ以外の関節構造のロボットについては再度考察が必要であるが，これまで開発された多くのロボットが同様の関節構造を持つため応用範囲は広いと考えられる．

以上に加えて，回転補正ゲイン α_r, α_f，ステート遷移時間 Δt，フードバックゲイン $\boldsymbol{q}_c, \boldsymbol{q}_v, \boldsymbol{q}_r, \boldsymbol{q}_w$ をそれぞれ全ステート共通に用いるとする．股関節ピッチの振り抜きパラメタ q_1, t_1 は歩行ステート中のみ共通で用いるとする．再度探索空間のパラメタ数を見積もってみる．目標姿勢 $\boldsymbol{q}_0$ は歩行動作の大枠を決定するため全 N 自由度を用い，6 ステートで $6N$ 次元．$\boldsymbol{q}_c, \boldsymbol{q}_v, \boldsymbol{q}_r, \boldsymbol{q}_w$ は遊脚股関節膝関節，支持脚足首関節の合計 5 自由度を速度と位置で 2 つづつで 10 次元．回転補正ゲイン 6，ステート遷移時間 1，股関節ピッチの振り抜きパラメタ 2 を合わせて $6N + 10 + 6 + 1 + 2 = 6N + 19$．12 自由度の脚型ロボットでは 91 次元となる．

4.3.3　動力学シミュレーションを用いた動歩行行動制御探索法

もう一度探索空間を表にまとめ確認する．

表 4.9 における $\mathcal{X}$ について探索を行いパラメタを決定することでさまざまな動歩行行動が生成できる．以下では探索における評価関数と条件について述べる．動力学シミュレーションを用いた探索では，現実に起こりうるすべての現象について探索を行うことができるため一般性の高い行動が生成できる反面，適切に条件を絞って探索を行わなければ意図したのとまったく異なる行動が生成されうるという難しさがある．

4.3.3.1　人間によるパラメタの決定と歩行実験

SIMBICON では，人間が動力学シミュレーションの結果を見ながらパラメタを決定することによって多様な歩行が生成できることが示されている．これは各ステートの持つパラメタが人間にとって分かりやすい意味を持つためである．各ステートの目標関節角度 $\boldsymbol{q}_0$ は姿勢を表し，ステートごとにそれを並べたときには紙芝居のように歩行動作の特徴点を並べたものが現れる．またフィードバックゲイン $\boldsymbol{q}_c, \boldsymbol{q}_v$ は，シミュレーションの挙動を見つつ，前のめりに倒れるのならばもっと足を前に出すように，横に倒れるようならばそちら側へ足を伸ばすように

Table4.9. Search space contents and their defrees of freedom

	Description	DoF
$\boldsymbol{q}_0^i$	Target joint angles of i-th state	$6 \times N$
$\boldsymbol{q}_c$	Feedback matrix for horizontal centroid	4
$\boldsymbol{q}_v$	Feedback matrix for horizontal centroid velocity	4
$\boldsymbol{q}_r$	Feedback vector for z-axis rotation	1
$\boldsymbol{q}_w$	Feedback vector for z-axis rotation velocity	1
α_r	Feedback matrix to reduce horizontal rotation of pelvis	2
α_f	Feedback matrix to reduce horizontal rotation of feed	4
Δt	State transition time	1
t_1	Polyline endpoint time of crotch interpolation	1
q_1	Polyline endpoint of crotch interpolation	1
$\mathcal{X}$	All parameters	$6N + 19$

と直感的に決めることができる．本章では，前章までに述べた歩行モデルで歩行行動が生成できることを確認するために，人間によるパラメタの決定を試みる．

まず，各ステートの持つ目標関節位置 $\boldsymbol{q}_0$ を 図 4.23 に示す．Start/Stop 姿勢は膝を伸ばした姿勢から，足先が 30cm 上に移動するよう逆運動学を解いて生成した姿勢である．Stance0 はさらにそこから重心位置を左足に向かって 65% 移動させたもの，Swing0 は重心を 80% 左足に移動させたのち右足を $60mm$ 上側に移動するよう逆運動学を解いたものである．Stance1 は Start 姿勢と同じもの．Swing1 は Start 姿勢から重心を 70% 右足に移動させたのち左足を $60mm$ 上側に移動するよう逆運動学を解いたものである．Stance2, Swing2 はそれぞれ Stance1, Swing1 を左右反転させた姿勢．SwingN, StanceN はそれぞれ Swing0, Stance0 と同じものを用いた．

4.3.3.1.1 シミュレーション実験

以上の姿勢を用いてシミュレーション上でパラメタを人力で調整した．図 4.24 にシミュレーション上で行われた足踏み動作を示す．足踏みは五秒を過ぎたところで停止ステートへ遷移するよう実装した．

図 4.25 は歩行中に推定された姿勢センサのクオータニオンを表している．前述の Madgwick らの手法 [107] によりなめらかに姿勢が推定できていることが分かる．また，図 4.26 は歩行中のステート遷移について表したものである．0 から 9 までのステートがそれぞれ，Start, Stance0, Swing0, Stance1, Swing1, Stance2, Swing2, SwingN, StanceN, Stop に対応している．0,1,2,3 で一歩目，4,5 で二歩目，6,3 で三歩目，その後 4,5,6,3 を二回繰り返し（四五六七歩目），4,5 で八歩，飛んで 7,8,9 で九歩目で停止していることが分かる．最後に図 4.27 に各時刻における関節角度目標値と，シミュレーションされた関節位置を重ねてプロッ

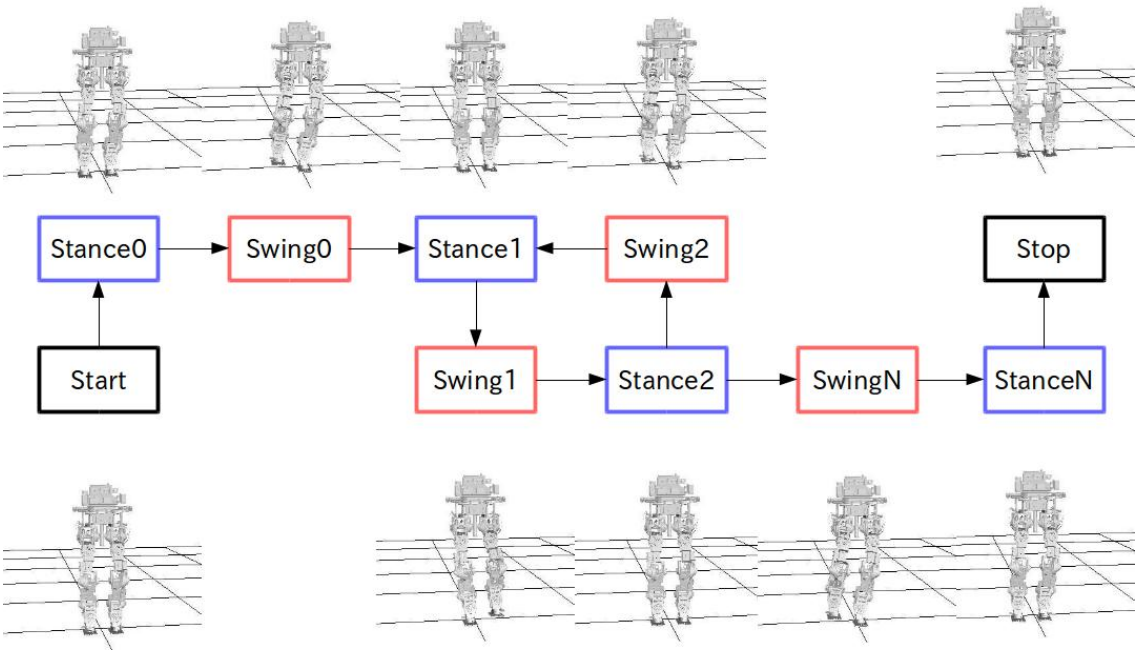


図 4.23. Stepping motion is heuristically generated. Start/Stop postures are generated by moving both legs 30 mm upward. Stance0 posture is by moving CoG 65 % leftside. Swing0 is by moving CoG 80 % leftside and moving right foot 60 mm upward. Stance1 is the same as Start posture. Swing1 is by moving CoG 70 % rightside and moving left foot 60 mm upward. Stance2 is the horizontal flip posture of Stance1. Swing2 is the horizontal flip posture of Swing1. SwingN is the same as Swing0. StanceN is the same as Stance0.

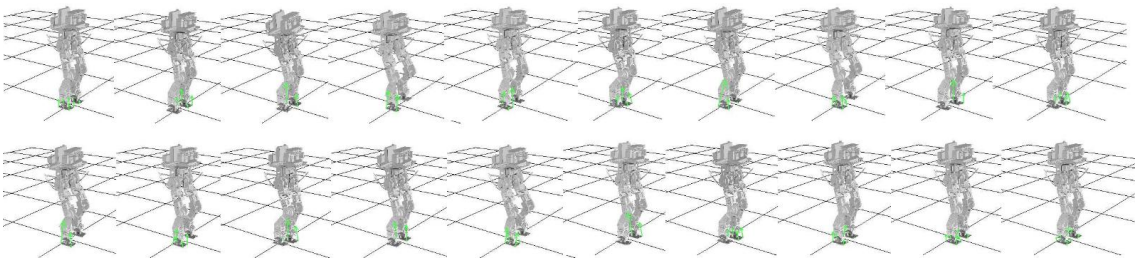


図 4.24. Snapshots of stepping motion in simulation world.

トしたものを示す.

4.3.3.1.2　実ロボット実験

次に同じパラメタを，動力学シミュレーションではなく実ロボットで実行した際の様子を図 4.28 に示す．歩行の停止ステートへの遷移も同様に五秒すぎた後最初の Stance2 にから遷移する実装を用いた．図のように転倒することなくその場足踏みが実現された．

シミューション実験と同様にいくつかのセンサデータをグラフに表示する．図 4.29 は歩行

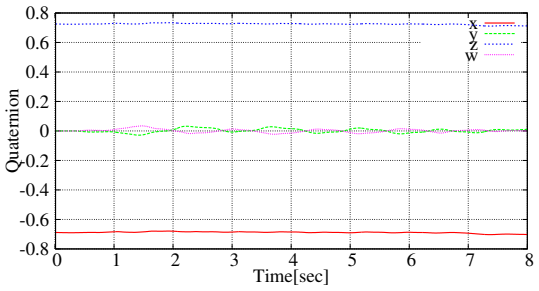


図 4.25.

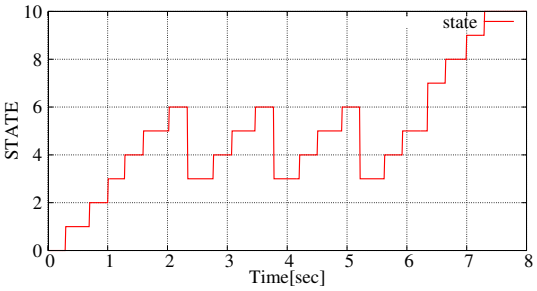
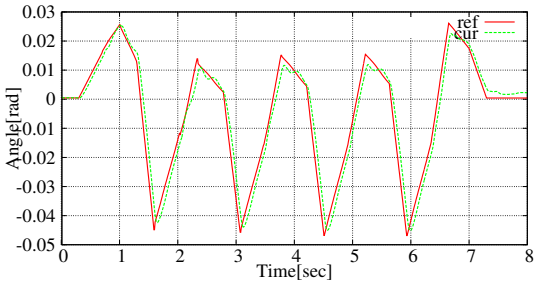


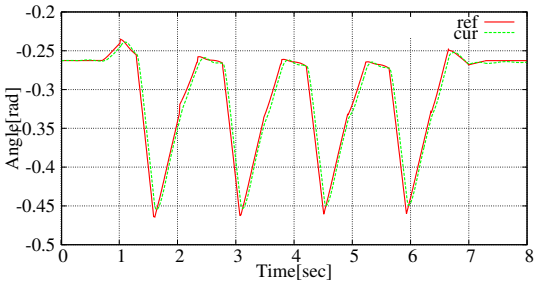
図 4.26.

4.25. Transition of quaternion while stepping. These values are calculated by using an open source implementation of [107].

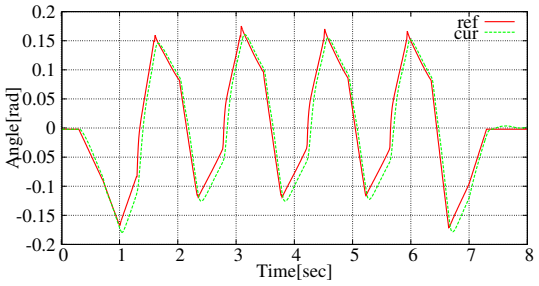
4.26 State transition while stepping. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.



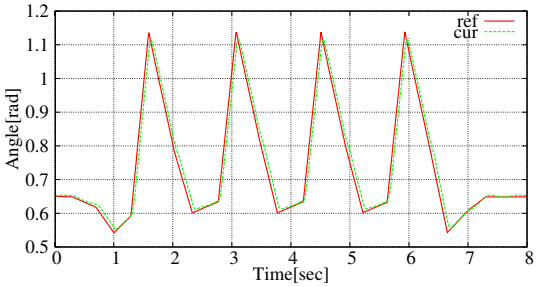
ID 1



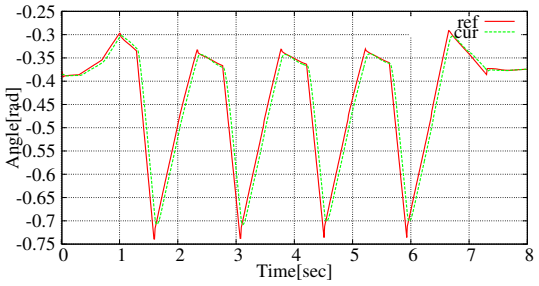
ID 2



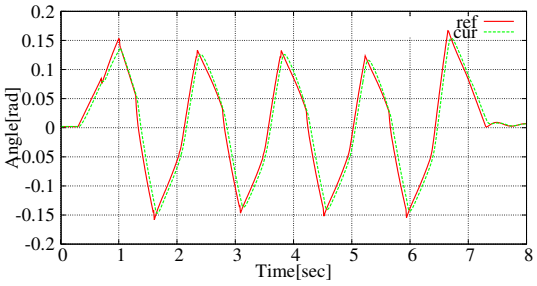
ID 3



ID 4



ID 5



ID 6

図 4.27. Joint angle targets and simulated values for left six joints while stepping motion. Because the stepping motion is symmetric, right joints move similar to left joints.

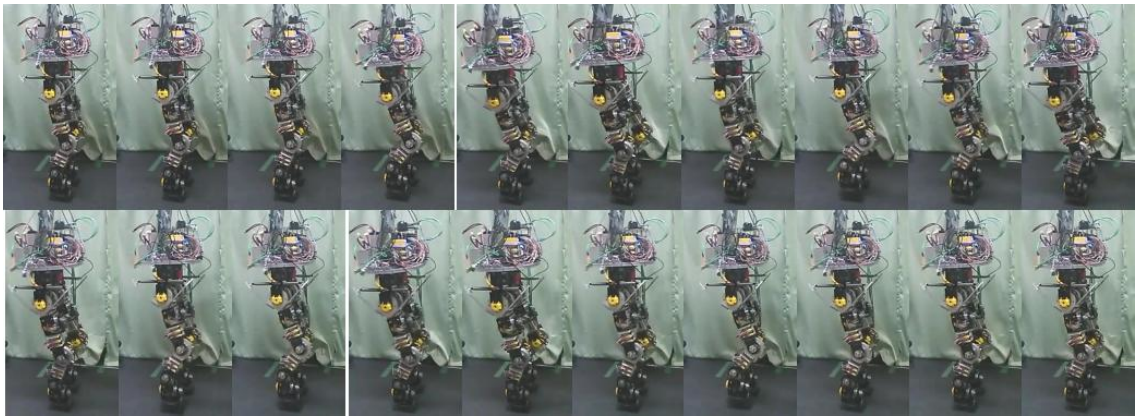


図 4.28. Snapshots of stepping motion in real world.

中に推定された姿勢センサのクオータニオンを表している．前述の Madgwick らの手法 [107] によりなめらかに姿勢が推定できていることが分かる．また，図 4.30 は歩行中のステート遷移について表したものである．0 から 9 までのステートがそれぞれ，Start, Stance0, Swing0, Stance1, Swing1, Stance2, Swing2, SwingN, StanceN, Stop に対応しているのも同様である．0,1,2,3 で一歩目，4,5 で二歩目，6,3 で三歩目，その後 4,5,6,3 を一回だけ繰り返し（四五歩目），4,5 で六歩，飛んで 7,8,9 で七歩目で停止していることが分かる．これはシミュレーション実験での歩数よりも二歩少ない結果となっている．制御の終了ステートへの遷移が始まる時間は等しく五秒であるため，実機のほうがシミュレーションよりも一歩にかかる時間が少し長くなっていると考えられる．また，ステートの遷移は接触力の変化で起こるため，シミュレーションよりも遅く接触力変化が検知されているということでもある．力センサのノイズや遅延をモデル化することでこの差は吸収できると考えられるが，もっと単純に接触力変化検出のスレショルドを実機とシミュレーションで違う値にすることでも対応できる（今回はともに 40 Nm）．モデル誤差を吸収する試みについては別で述べる．

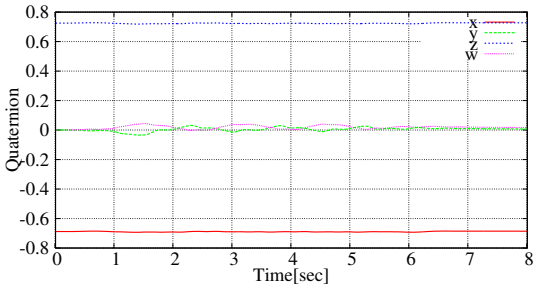
最後に関節角度のエンコーダによる測定値と目標値を重ねたグラフを図 4.31 に示す．前半四秒程度まではシミュレーションとよく似た見た目のグラフが得られているが，五秒から先では大きくことなっており，モデルの誤差を吸収する必要がある．以上から，本章で実装したモデルを用いることで安定な歩行動作・制御を生成することが可能であり，また実ロボットを用いても実現できるものであることが確認できた．

4.3.3.2　省エネルギー歩行のための評価関数

CoT: Cost of Transport とは単位移動距離単位質量に対する全消費エネルギーであり移動効率を表す指標として用いられる [100] [108]．式で表すと式（4.71）のようになる．

$$c_{et} = \frac{E}{mgd} \tag{4.71}$$

E は総消費エネルギーであり電源電圧と電流を測定することで計算できる．m はロボット



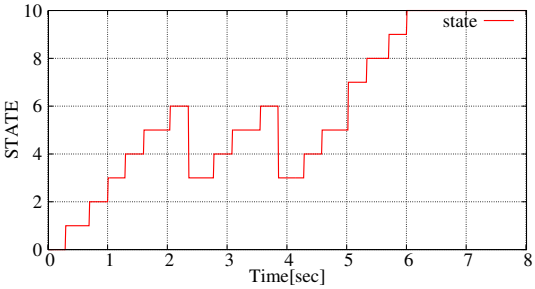
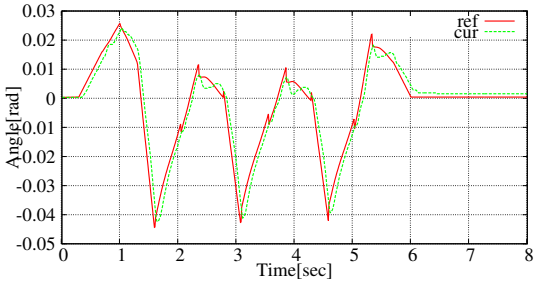


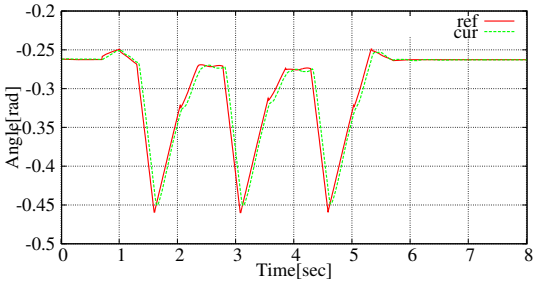
図 4.29. **図 4.30.**

4.29. Transition of quaternion while stepping. These values are calculated by using an open source implementation of [107].

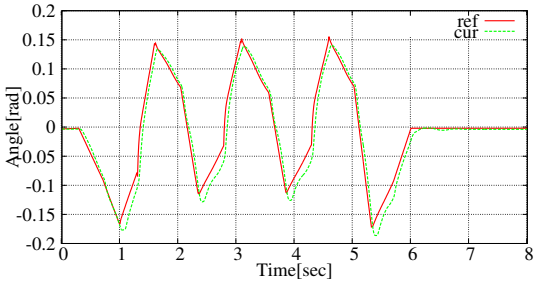
4.30. State transition while stepping. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.



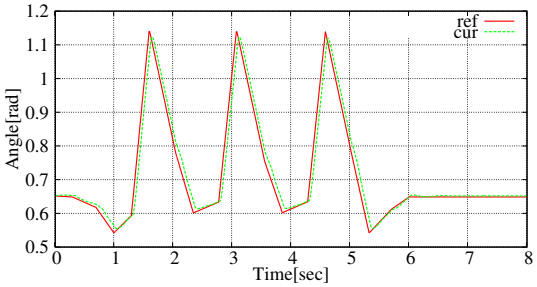
ID 1



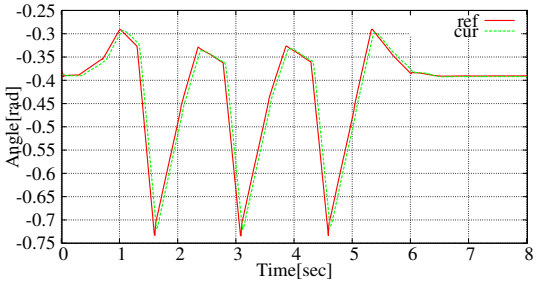
ID 2



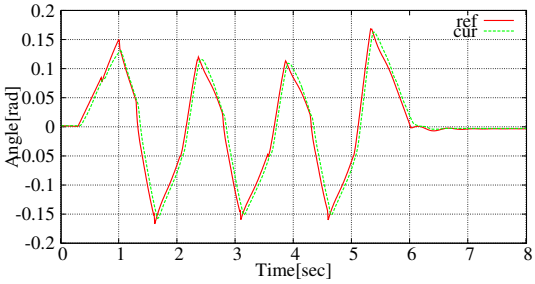
ID 3



ID 4



ID 5



ID 6

図 4.31. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

の総質量，g は重力加速度，d は並進移動量である．等身大ヒューマノイドロボットでの CoT 最小値は [100] で報告されている 1.33 であるが同チームの別の報告（[108]，同年に同じ国際会議 ICRA で同時発表）では 1.6 と幅がある．原因としては CoT がコンピュータの消費電力やモータドライバのエネルギーまで含む複雑なモデルを持っているために様々な要因で値が変わりうるためと考えられる．例えば気温や湿度は電装系の電気抵抗を増加させ消費エネルギーを増加させる．地面の傾きが前にくだっていれば移動量 d が増え，逆なら減るといったことも考えられる．したがって別のロボット・別の環境と比較し優劣をつけるためには環境の測定やセンサ類の測定誤差まで考慮する必要があり本研究ではオーダの比較程度にとどめることとする．しかし同じロボット同じ環境で比較実験をする場合にはロボットの移動効率を直接的に表現でき有用であるため本研究でも CoT を評価関数に用いる．

実ロボットで CoT を計測することは，大本の電源電圧・電流を測定して掛け合わせるだけで容易に可能である．一方シミュレーションで CoT を評価関数に用いるには，コンピュータやモータドライバの消費エネルギ，電装系の熱損失，ギアやベアリングの摩擦による機械損失，多リンク系の運動エネルギーを計算する必要がある．本研究では式（4.72）のモデルを用いる．

$$
\begin{aligned}
E &= \alpha_e \int dt \left(dE_f + dE_t + dE_0 \right) \\
dE_k &= \sum\nolimits_{i \in [0,N)} max(\tau_i \omega_i, \alpha_r \tau_i \omega_i) \\
&= \sum\nolimits_{i \in [0,N)} max(I_i V_i, \alpha_r I_i V_i) \\
dE_t &= \boldsymbol{I}^T \boldsymbol{I} R \\
dE_0 &= C
\end{aligned}
\tag{4.72}
$$

ただし，関節ごとのトルク，関節角速度，電流，電圧を以下のベクトルで表現する．

$$
\boldsymbol{\tau} = \begin{pmatrix} \tau_0 \\ \vdots \\ \tau_{N-1} \end{pmatrix}, \boldsymbol{\omega} = \begin{pmatrix} \omega_0 \\ \vdots \\ \omega_{N-1} \end{pmatrix}, \boldsymbol{I} = \begin{pmatrix} I_0 \\ \vdots \\ I_{N-1} \end{pmatrix}, \boldsymbol{V} = \begin{pmatrix} V_0 \\ \vdots \\ V_{N-1} \end{pmatrix} \tag{4.73}
$$

またシミュレーションされた関節トルク，角速度から電流，電圧を計算する際は以下の式を用いた．

$$
\begin{aligned}
\boldsymbol{I} &= A_\tau^{-1} \boldsymbol{\tau} \\
\boldsymbol{V} &= A_\tau \boldsymbol{\omega}
\end{aligned}
\tag{4.74}
$$

多リンク系の運動エネルギー，摩擦損失，環境の衝突によるエネルギー損失は摩擦トルクを含まないアクチュエータの発揮するトルク $\boldsymbol{\tau}$ と関節の角速度 $\boldsymbol{\omega}$ の積により計算できる（エネルギー保存則）．また $\boldsymbol{\tau}^T \boldsymbol{\omega}$ が負である時，電源に逆電流が流れることで回生エネルギーを生じる．[109] のモータドライバはコンデンサに蓄えられた逆電流を再利用することができるが全てが充電されるわけでないため，回生エネルギーの α_r 倍がマイナスのエネルギーとして積分されるよう $\sum_{i \in [0,N)} max(I_i V_i, \alpha_r I_i V_i)$ を項 dE_k として計算する単純なモデルを用いる．

$dE_t = \boldsymbol{I}^T \boldsymbol{I} R$ の項はモータの熱損失，$dE_0 = C$ はコンピュータやモータドライバの総消費電力であり定数として扱う．またトルクと電流，角速度と電圧について比例関係がなりたつとした．A_r はトルク定数と呼ばれ，単位電流量あたりのモータの出力トルクを表す．MAXON 製のモータ [110] では，型番からトルク定数を調べることができる．多リンク系の消費する全エネルギーが関節負荷トルクと関節角速度の積 $\boldsymbol{\tau}^T \boldsymbol{\omega}$ で計算でき，それがモータの電流電圧から計算される消費エネルギー $\boldsymbol{I}^T \boldsymbol{V}$ と等しいとすると，トルク定数の逆数は電圧と関節角速度の比例係数を表していることが分かる．

以上により CoT がシミュレーションできたとして，CoT は歩行に成功した動作に対してしか意味を持たないことに注意が必要である．並進移動量 d を動力学シミュレーション上でのロボットの重心座標を用いて本研究では計算するが，正面に全力で倒れこむような動作は比較的省エネルギーで並進移動量 d を大きくすることができるため CoT の良い動作として出力されてしまう恐れがある．このような歩行の条件を満たさないものを探索中に単純に弾き，悪い評価値を与えてしまうと，こんどは大半のパラメタが同じ悪い評価値を持つことになってしまい探索がうまくいかない．そこで本研究では，歩行に失敗している間は歩行に成功していた時間を増やすように探索し，歩行が一定時間以上成功した場合のみ CoT を減らすよう探索する手法をとる．具体的には以下を最大化する．

$$Maximize \begin{cases} t_{max} + \frac{1}{CoT} & (t > t_{max}) \\ t & (t \leq t_{max}) \end{cases} \tag{4.75}$$

t はシミュレーションが停止した時刻である．

4.3.3.3 実現可能な歩行生成のための条件

歩行に失敗した場合はシミュレーションを停止させ，失敗までにかかった時間を最大化するようにパラメタを変化させていくことで歩行動作を生成する．以下に歩行の失敗条件として本研究でも用いたものを示す．

4.3.3.3.1 自己リンク間干渉の確認

全関節リンク間の衝突を計算し，衝突が見つかった場合は失敗とする．衝突計算には PQP [94] を用いた．またこの計算はコストが高いためシミュレーション時間で百ミリ秒おきに行なった．

4.3.3.3.2 環境と接してはならないリンクの衝突確認

歩行中に足首から下以外は地面と衝突しないとし，他のリンクが環境と衝突し力を受けたら失敗とする．また外装に被われていないリンクが力を受けてロボットが破損することを防ぐ意図もある．

4.3.3.3.3 転倒しないことの確認

歩行中に体幹リンクの高さが地面から 80 センチを下回まわった場合に転倒とし失敗とする．

4.3.3.3.4　関節速度上限の確認

シミュレーションされた関節速度の大きさが 3.14 rad/s を越えた場合失敗とする．

4.3.3.3.5　関節負荷トルク上限の確認

シミュレーションされた関節負荷トルクの大きさが 90 Nm を越えた場合失敗とする．これはモータドライバの流せる電流量により決まる値である．

4.3.3.3.6　接触力モーメント上限の確認

シミュレーションされた足裏力センサの接触モーメントの大きさが 50 Nm を越えた場合失敗とする．これは力センサの破壊モーメントにより決まる値である．

4.3.3.3.7　シミュレーション制限時間の確認

あらかじめ決められた制限時間以上に歩行ができた場合は歩行成功であり，CoT を計算し評価する．

4.3.3.3.8　立ち止まっていないことの確認

転倒するまでの時間が評価関数に用いられているため，まったく動かず立ち尽くすような結果がしばしば得られる．それを防ぐために，両足平の地面からの高さを監視し，シミュレーション時間で 1000 ミリ秒以上の間高さが 5 mm 以下でありつづけた場合に失敗とする．

4.3.3.3.9　ジャンプしていないことの確認

スキップや走行のような動作が得られたら興味深い結果であるが，歩行を素早く見つけるためにここでは弾くことにする．両足平の地面からの高さを監視し，シミュレーション時間で 500 ミリ秒以上の間高さが 5 mm 以上でありつづけた場合に失敗とする．

4.3.3.4　動力学シミュレーション上での進化計算による動歩行行動制御探索実験

非線形最適化ライブラリ NLopt [14] に実装された大域的最適化のための進化計算アルゴリズム ISRES [9][10] を用いた．ISRES は不等式制約を満たさない解の優劣を考慮した方法であるが，本章の実験では不等式を用いないため評価値の劣った解についての突然変異と評価値の優れた解についてのネルダー・ミード法 [111] を単純に組み合わせた手続きとなっている．

図 4.32 に探索された SIMBICON モデルを探索したときの，評価関数 式（4.75）の遷移を六日間に渡ってプロットしたグラフを示す．探索を五スレッドを独立して行った．歩行は十五秒後に停止ステートに遷移し二十秒間転倒しないものを探索した．したがってグラフ縦軸で $1/20 = 0.05$ を下回るスレッドは転倒しない解を発見しているということになる．結果四スレッドが一日程度，二日間では全スレッドが転倒しない解を見つけられていることが図より確認できる．図 4.33 は SIMBICON モデルを探索したときの，CoT の遷移を六日間に渡って

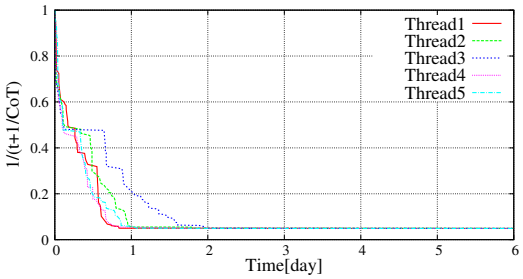


図 4.32. The transition of objective function $(\frac{1}{t+\frac{1}{CoT}})$ while searching walking motion.

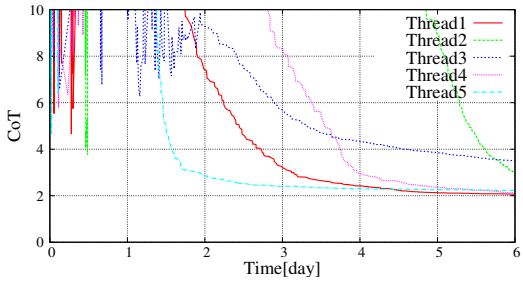


図 4.33. The transition of CoT: Cost of Transport while searching walking motion.

プロットしたものである．一番速く収束しているスレッドでは二日時点で CoT = 3 を下回る程度．六日時点で CoT = 2 をわずかに上回る程度の解が得られていることが確認できる．また三スレッドは六日時点でほぼ等しい $CoT \approx 2$ に近づいているが二スレッドはそれよりも高い CoT になっている．このことからシングルスレッドの探索は十分良い解を得るのに十分ではないことが分かる．

図 4.35 に得られた解のなかでもっとも評価値の高かったものについてシミュレーション中の様子を示す．図 4.34 は各スレートにおける目標姿勢 q_0 である．20 秒間のシミュレーションでの CoT は 2.05. 100 秒の歩行では CoT = 1.77 となった．消費エネルギーをプロットしてみると図 4.36 のようになった．歩き初めの $Time \approx 1$ と歩き終わりの $Time \approx 4$ 周辺で全身を加減速させるために消費エネルギーが一時的に大きくなっていることが分かる．歩き初めと終わりは大きな消費エネルギーを要するが移動距離はかせげないため，間の歩行期間を長くすることで CoT が小さくなると考えられる．

また歩行時の脚を拡大してみると図 4.37，4.38， 4.39 のようになった．時系列順に右から並べている．足が地面から浮いた状態からまず踵が接地し，踵まわりに足を転がすようにして足全体が地面を捉える状態へと遷移している．フットステップに条件を与えない探索方法でこのような足運びが解として得られたことは興味深い．

4.3.3.5 歩行の安定性について

本章の手法で得られた歩行動作は歩き始めから歩き終わりまでをまとめて探索しているが，歩行を停止させずに長距離を移動するといった使いかたをする場合には安定性の問題がある．シミュレーションで保証された安定に歩行ができる時間は，歩き終わる直前までであり，その後歩き続けられるかどうかを保証しない．その一秒後に転ぶ可能性があるということである．実際，探索された二十秒転ばない歩行動作をシミュレーション上で百秒歩行させたところ転ばずに最後まで歩くことのできた歩行は半分程度であった．

受動歩行の研究ではリミットサイクルという考えを用いて歩行の安定性を解析する．ある歩行状態からシミュレーションをはじめて同じ状態を見つかるときそれをリミットサイクルと呼ぶ．リミットサイクルを微分することで，初期状態にノイズが入ったときに一リミットサイク

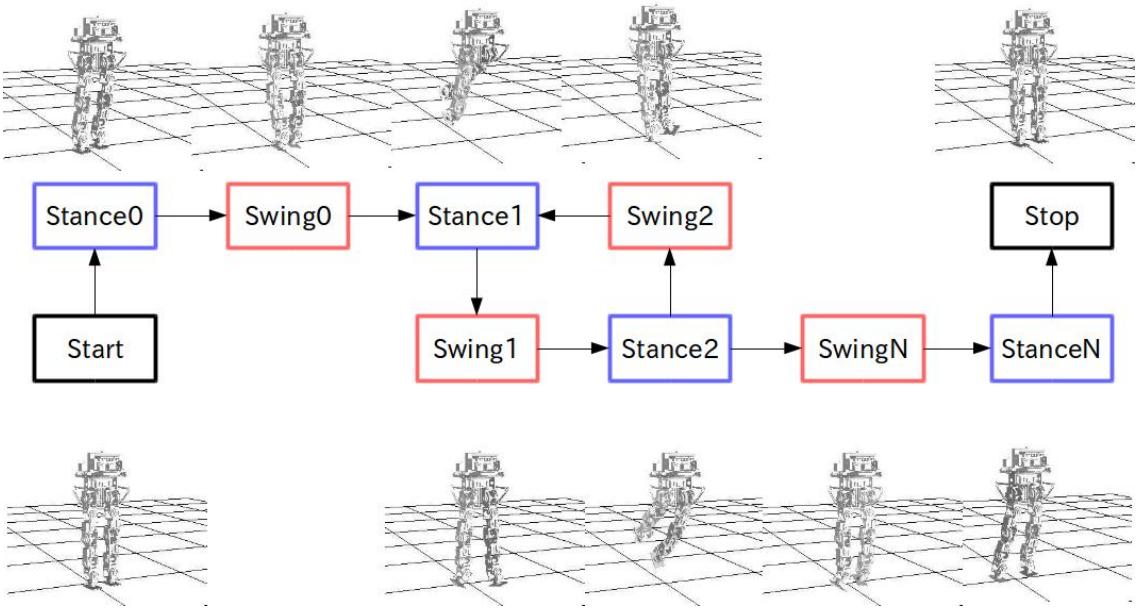


図 4.34. Walking motion is optimally generated. Start/Stop postures are given; all joint angles are zeros. The other postures are optimized to minimize CoT: Cost of Transport.

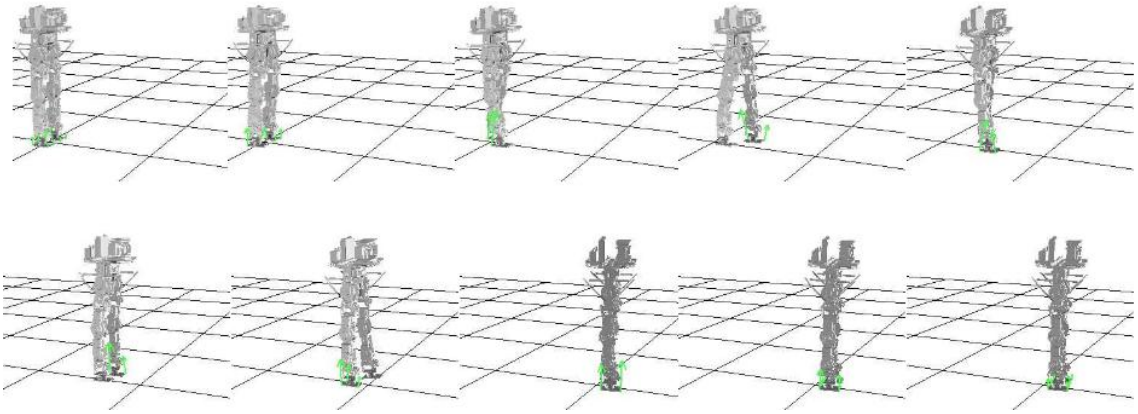


図 4.35. Snapshots of optimal walking motion in simulation world.

ル後の状態の変化量を計算することができ，これが無限回積算されたときに零に収束するならば安定であるという考え方である．本章で用いた歩行探索アルゴリズムでは，ハードウェアリミットなど歩行を実ロボットで実現するために満たさなければならない条件を確認し，満たしていないものを失敗とすることで実現可能な歩行動作を生成した．同様にリミットサイクルの安定条件を探索に加えることは可能である．その場合は歩行サイクルのみをあらかじめ最適化しておき，歩き始めと歩き終わりを別の探索問題として解くほうが効率が良いと予想している．

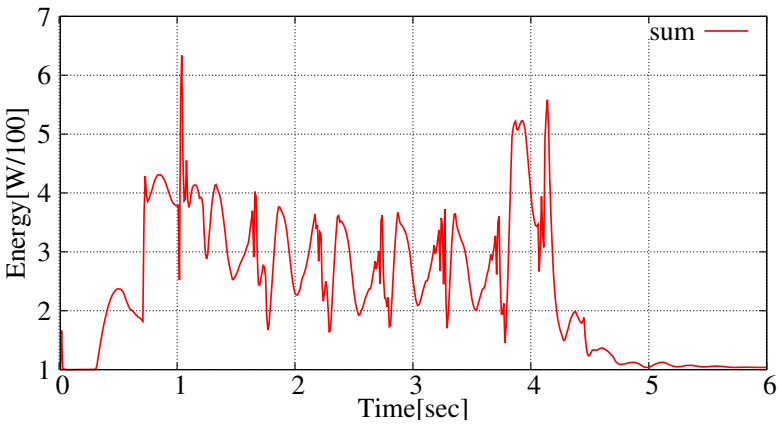


図 4.36. Energy consumption while optimized walking. The peak of energy consumptios are found in starting and stopping phase of walking.

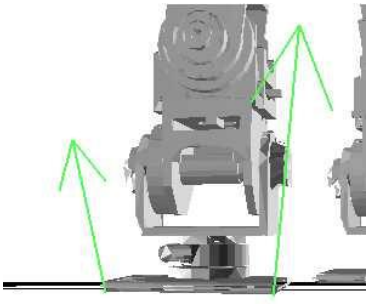


図 4.37. Foot on.

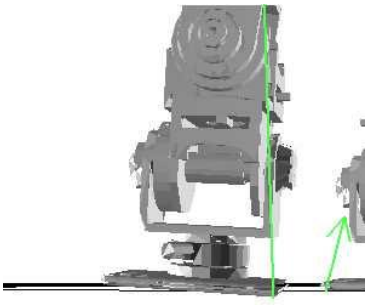


図 4.38. Heel touch.

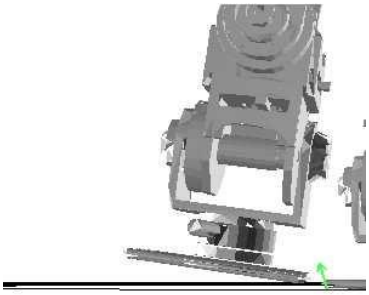


図 4.39. Foot off.

本研究では簡便な方法として，探索中に評価値が更新される度にその解をログファイルに書き出しておき，探索が終了したのちにファイルを下から順に確認していって最初に見つかった歩行予定時間より長く歩行できる動作を用いている．本章の実験で得られたログファイルでは上位百の答えのうち百秒以上歩ける答えは 46% であった．さらに一万秒歩行できる解は 16% であり，そのような解で一番目に見つかったのはログファイル末尾から数えて 31 番目であった．CoT でみると，二十秒を歩行する解は CoT = 2.05 であったのに対し，一万秒歩ける解は二十秒で CoT = 2.12 と 5% 以下の差であった．同様の方法でリミットサイクルの安定性を確認することで安定な歩行を見つけることも可能である．

4.4　接触遷移行動探索による着座支持棒長さの決定

本章では接触遷移方式の探索機能も統合した行動生成アルゴリズムを提案し，着座行動を含むすべりやころがりを利用する行動生成を達成した．これを身体環境モデルを変化させながら用いることで行動生成可能なモデルを獲得したり高評価な行動が可能となるようモデル探索し

たりといった応用が可能である．脚のみに限らない接触を伴う行動生成では関節負荷トルクをはじめ複雑な非線形制約を考慮する必要があり人間による運動モデルパラメタ生成は困難である．本章では着座行動をとりあげ，環境との干渉が起こらない着座行動生成が可能な体幹支持棒の長さを探索により決定することで，身体行動創成支援の例を確認する．

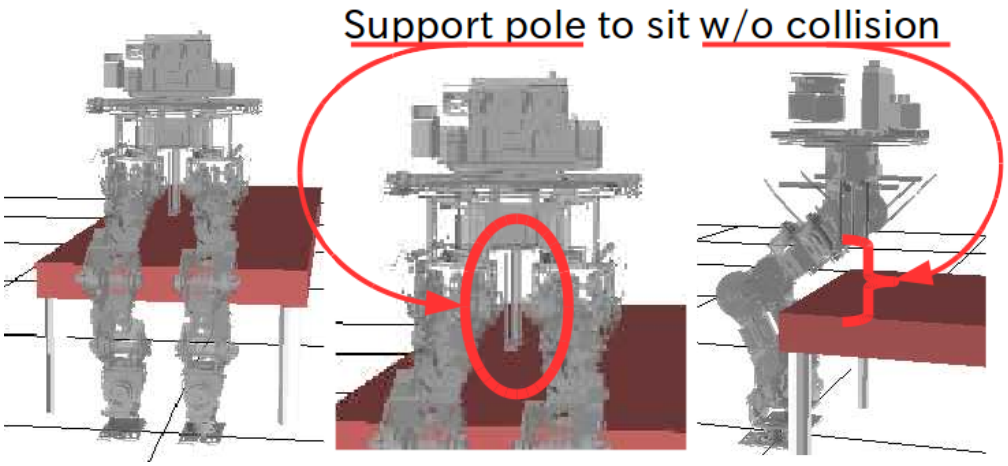


図 4.40. Front/side view of sitting motion.

図 4.40 に本章で考える問題の模式図を示す．脚型ロボットの体幹リンク下には着座時に体幹を支持するための棒が取り付けられており，この棒の長さを適切に設定することで椅子との干渉の起こらない着座行動が可能である．

Table4.10. Support pole height v.s. sitting feasibility

height [mm]	260	270	280	290	300	310	320	330	340	350
Feasible?	no	no	yes	yes	yes	yes	yes	yes	no	no

Table 4.10 に支持棒高さを買えながらの探索により探索により着座が可能か否かを確認した結果をまとめる．支持棒高さは 200mm から 400mm までを 10mm 刻みに確認し，実現可能であった 270mm から 330mm を中心に十の高さについてのみテーブルには示している．高さが低すぎると太股の干渉や運動学的に答えがでないなどの理由により実現可能な解が得られなかった．逆に高すぎると足の長さが足りず深く座ることができないために解が得られなかった．

以上の結果を元に，着座可能な一番短い支持棒をロボットの身体設計に組み込み，開発した脚型ロボットを用いて実際に着座行動実験を行った様子を図 4.41 にしめす．HRP-2 を用いた実験と同様に摩擦係数の推定を行いながら実験を行い，一度目の試しすべり動作で摩擦係数が計画値 0.2 よりも小さいことが確認できたため，そのまま着座行動が実行されている．すべり動作は 15mm 行われ両脚のすべりは起こらなかった．実験で用いた脚型ロボットは脚部の配線がむき出しであり，着座時の干渉が起こると危険である．実験では十分に干渉のマージンをとって探索した行動を用いているため安全に実験が達成できている．

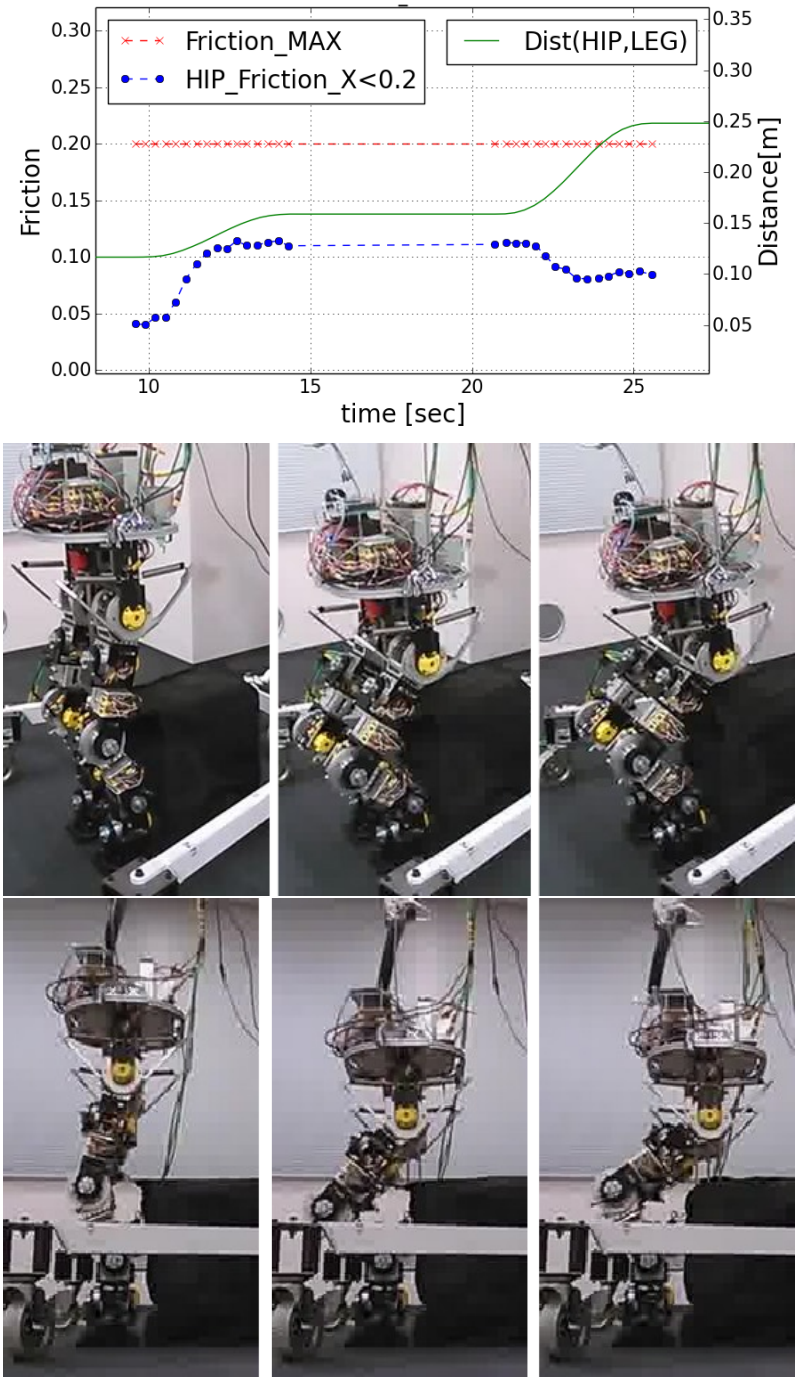


図 4.41. Front/side view of sitting motion.

4.5 おわりに

本章では接触遷移を伴う行動生成手法として最適化手法とグラフ探索を組み合わせたもの、動力学シミュレーションと進化計算を組み合わせたものそれぞれについて実装と実験を行っ

た。前者については接触遷移方法として単一のものに限らず，滑りや転がりも拡張考慮した行動生成手法を扱い，実ロボットを用いた滑り着座行動実験を示した。後者についてはさらに制御パラメタの探索も含めた動歩行生成実験を行った。さらにそれを用いて着座行動を可能とするような体幹支持棒の長さを探索した。

第 5 章

運動結果距離最小化による歩行行動の忠実再現

5.1 はじめに

本章では実ロボットの運動結果の一部がセンサ情報から得られたとして、センサ情報から計算できない身体環境モデルのパラメタを推定し、歩行行動パラメタを再生成することで歩行行動を計画通り忠実に実現する実験を行う。図 2.12 が解く問題のモデルと探索法をまとめたものである。本章の内容は支援システムにおけるロボット完成後の実体の身体環境モデルのパラメタ推定においてセンサから測定できない値の推定を行う手法であり、適用範囲は広い。

以下ではまず、本研究で開発した脚型ロボットの身体構成、ソフトウェア構成についてまとめ。つぎにその身体情報を用いての行動生成実験、実体再現実験を行っていく。

5.2 受動軸を備える脚型ロボットの身体構成・ソフトウェア構成

5.2.1 ハードウェア構成

5.2.1.1 概要

図 5.1 に完成した ML1 の全身写真を示す．足裏から体幹リンク股関節付け根までの高さは 918 mm，全体重は 49.847 kg の等身大サイズの脚型ロボットである．関節自由度は十二自由度，関節配置は片足につき根元から順にヨー軸，ピッチ軸，ロール軸，ピッチ軸，ピッチ軸，ロール軸の六自由度となっている．コンピュータは Intel NUC [112]．センサは両脚先端に六軸力センサ（ワコーテック株式会社の WEF-6A500-10[113]），体幹リンクに十次元姿勢センサ（VN-100 [114]），モータはすべて同じで ブラシレス DC モータ maxon EC-4pole 30 200W [110]，全関節にはアブソリュートエンコーダ Heidenhain ECI 1118 [115] と，減速比 160 倍のハーモニックギア shd 20 [116]，無励磁動作ブレーキ BXR-LE [117] を配置している．以下，1 番から 8 番まで番号をふった箇所について述べる．

1 番の図 5.2 は ML1 の体幹リンク上側について拡大して表示したものである．体幹には

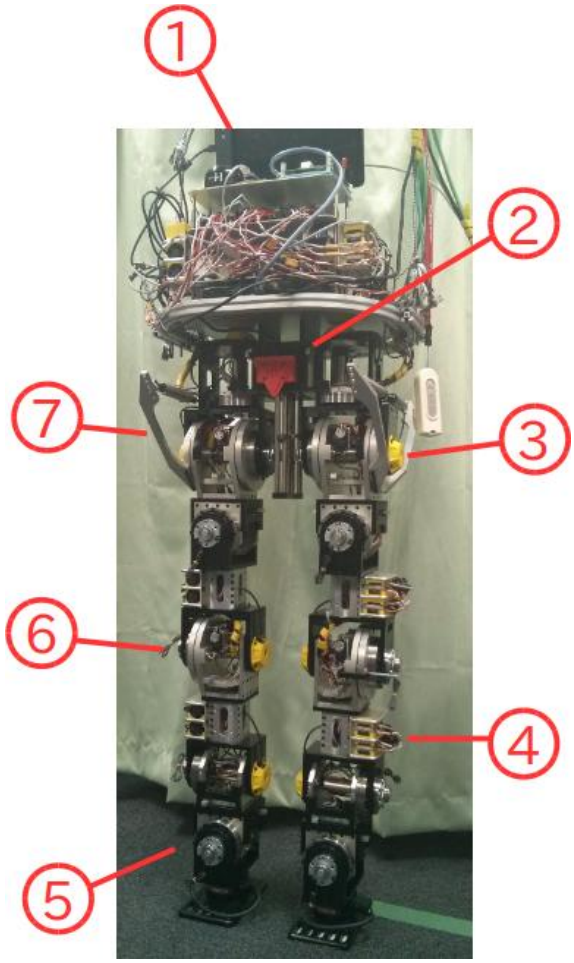


図 5.1. Full-length photograph of ML1. Each part is described in the following pictures.

バッテリー（図左），モータドライバや電流センサとの通信基板，異常警告用スピーカー（図真中），12・24・60 ボルトの電源ターミナル端子台（図右）等が搭載されている．図には写っていないが，その他にコンピュータ [112]，電圧変換を行う DCDC コンバータ，手動電源スイッチ，無線電源スイッチもここに配置されている．バッテリについては [118] に用いられているものと同様のものを用いている．

2 番の図 5.3 は体幹リンク下側を拡大したものである．体幹リンク前側中央には加速度・角速度・磁気・圧力を測定可能な十次元姿勢センサ（VN-100 [114]）が配置されている．また体幹リンク下面には着座時に太股関節が地面と接触することを防ぐための支持棒をつけている（図右）．

3 番の図 5.4 は股関節ピッチ軸に配置されたアブソリュートエンコーダ [115] を示している．それぞれのアブソリュートエンコーダは黄色いカバーで保護されている．図中黒い板のように見える領域は脚根元のヨー軸に固定されており，ここにアブソリュートエンコーダ本体を固定している．図中銀色に見える円盤は股関節ピッチ軸により駆動されるリンクであり，こちらにアブソリュートエンコーダ回転軸を固定することでリンク間の相対回転を計測している．

図 5.2. Zoom picture of upper side of body link. Left image shows battery. Middle image shows communication boards between sensors, motor drivers, and computer. Right image shows electric terminal block including 12 voltage, 24 voltage, and 60 voltage.

図 5.3. Zoom picture of lower side of body link. 10-dof imu sensor (vn-100 [114]) is placed on the middle of front plane. Bottom plane has five poles to sustain body link when sitting.

4 番の図 5.5 はモータドライバを示す．[109] の超小型大出力モータドライバを用いている．モータドライバは二関節分を一セットとし，体幹に一セット，膝関節上に一セット，膝関節下に一セットを両脚分で合計六セット十二個配置されている．これはモータの数の同じである．余談であるが，図中モータドライバを保護するように折れ曲がった三本の銀棒は丸棒取手として売られている [119] ものを応用している．

図 5.6 は ML1 の足先二軸を示す．足平には軽量化のため五本の溝が掘られているが，同時に五本の指を模している．足平と足先先端関節（ロール軸）の間には六軸力センサが配置されている．力センサとしてはワコーテック株式会社の WEF-6A500-10[113] を用いた．

図 5.7 は膝関節を拡大したものである．膝関節により駆動される銀色のリンクは二つの独立して回転する円盤を持っている．これらを図中央に見えるラジコンサーボで水平ピンを抜き差しすることで動力の伝達を切り替える．このクラッチ機構については続く章で解説する．

図 5.8 は股関節ピッチ軸を拡大したものである．この軸にも膝関節と同様のクラッチ機構が搭載されている．またリンクからは角のようなものが伸びそれがバネによって股関節ヨー軸につながっていることが分かる．角が固定されている円盤はピッチ軸と独立して回転するリンクであり，クラッチピンが挿入されることでバネが力を伝達し歩行を助ける機構になっている．

図 5.4.

図 5.5.

図 5.6.

図 5.7.

5.4. Absolute Encoder (Heidenhain ECI 1118 [115]) is placed on each joint. The encoders are protected with yellow cover.
5.5. A pair of motor drivers. Each pair is placed in body link, upper side of knee joint, and lower side of knee joint. Therefore, total number of motor drivers is 12 (6 pair), and this is the same number as maxon motor.
5.6. Knee joint has clutch mechanism. Knee link has two independent rotor links and these links are connected or disconnected by horizontal pin driven by small servo motor.
5.7. Righ leg end effector. Foot link is cut of five grooves to lighten, and these grooves mimic toe fingers. 6-axis force sensor is placed on the foot.

図 5.8. Crotch pitch joint has clutch machine. Clutch link has three independent rotor links, and these links are connected or disconnected by horizontal pin driven by small servo motor. One rotor has hone which is connected to crotch yaw link through spring. This spring supports walking motion.

このクラッチ機構及びバネについても続く章で解説する．なお，角が中ほどでへこんでいるのはピッチ軸を図手前に向かって持ち上げたときに緩衝しないようピッチ軸の可動域を阻害しないためである．

5.2.1.2　既存ロボット JAXON との比較

開発した脚型ロボット ML1 を先行研究で開発された等身大ヒューマノイドロボット JAXON の右足と並べたものを図 5.9 に示す．赤い足が JAXON のものである．足裏から脚

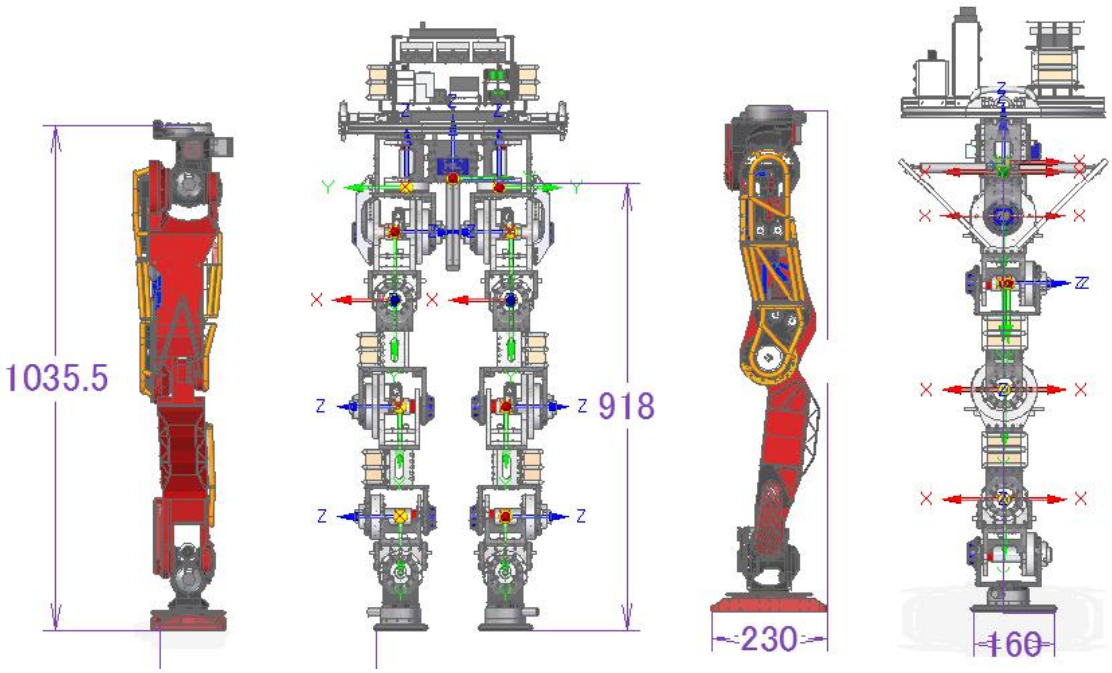


図 5.9. Red leg is right leg of JAXON[1], and gray leg is developed leg. Height from foot to root joint is 918 mm, which is similar to JAXON but a little shorter. Foot length is 160mm and is shorter than JAXON.

根元関節までの距離を比較すると JAXON が 1035.5 mm であり，ML1 は 918 mm とやや短いが同程度の長さである．足平の長さも JAXON 230 mm に大して 160 mm と小さいが，これは用いている力センサの定格モーメントが小さいため力センサの破壊を防ぐために大きなモーメントが発生しないような短い足平を用いているためである．ML1 は JAXON の備えるアブソリュートエンコーダに加え，クラッチとブレーキを備えるがリンクをほとんど大きくすることなく開発できていることが分かる．

5.2.1.3 関節配置

ML1 の関節配置を 図 5.10 に示す．赤い円筒が膝を伸ばして立ったときに回転軸が正面を向く関節（ロール軸）．緑は回転軸が横を向く関節（ピッチ軸），青は縦を向く関節（ヨー軸）である．関節自由度は片足につき六自由度の計十二自由度．関節可動域は全軸共通でプラスマイナス九十度，零度のときに膝が伸びて立つ姿勢になる．また各関節を区別するための識別番号を振っている．体幹リンクを 0 番として，左足根元から順に 1 – 6 番，右足根元から同様に 7 – 12 番とする．また関節間をつなぐリンクは番号の小さいほう（体幹に近い親関節）の番号を用いて識別することにする．

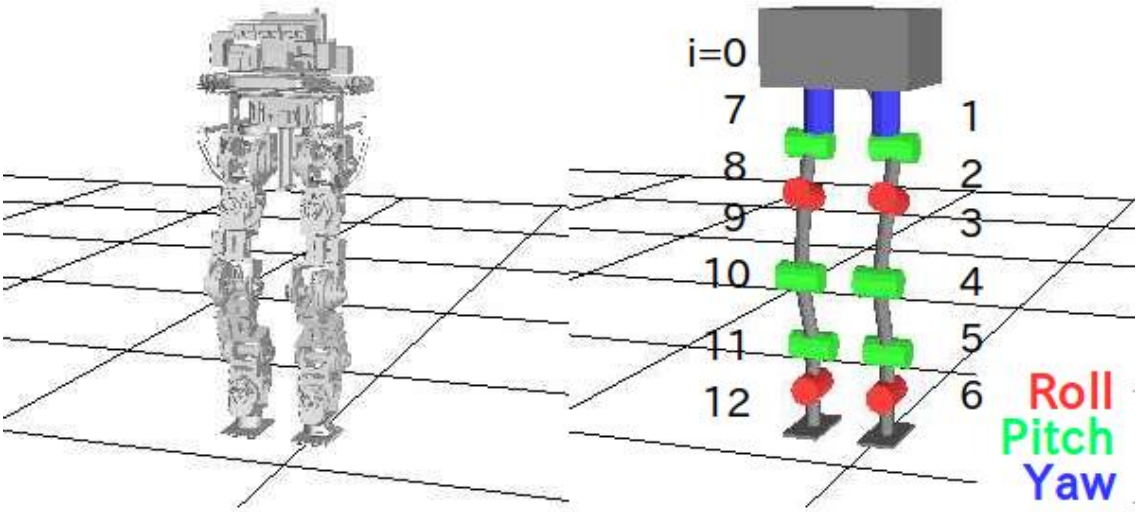


図 5.10. Joint configuration. Red is roll, green is pitch, and blue is yaw joints. Each joint has unique joint id i. Root link (BODY) is zero, and joint near to 0-th joint has smaller id. Left joints have smaller joint id than right joints.

5.2.1.4　電装系

図 5.11 に ML1 の電装系を表す模式図を示す．大本の電源は 12 ボルトと 60 ボルトである．これらは電流センサ，電圧センサが取り付けられ消費電力が計測できるようになっている．12 ボルトはモータドライバの駆動に用いることに加えコンピュータ電源 19 ボルトとクラッチサーボ電源 6 ボルトを DCDC コンバータにより生成している．60 ボルトはモータの入力電源である．また 60 ボルトから DCDC コンバータにより作られた 24 ボルトが無線リレースイッチを駆動し，いつでも遠隔からモータ入力電源 60 ボルトを導通・非導通へと切り替えられるような安全装置となっている．

5.2.1.5　リンクの質量・重心・慣性テンソル

表 5.1 に各リンクの質量，重心，慣性テンソル，関節間相対位置姿勢についてまとめたものを示す．値のオーダが見やすいように単位を変えているので，重心位置と関節間相対位置ではミリメートル単位だが慣性テンソルではメートル単位を用いている．表から分かる通り，関節の原点姿勢はすべて等しく単位行列となるようにモデルを作っている．そのため R_{ji} は常に単位行列である．

5.2.2　制御フロー

ML1 と制御コンピュータ間の通信と制御フローについて，図 5.12 に概略をしめす．このシステムは [118] に示される単一スレッド方式の制御フローをベースとしつつ，続く章に示す制

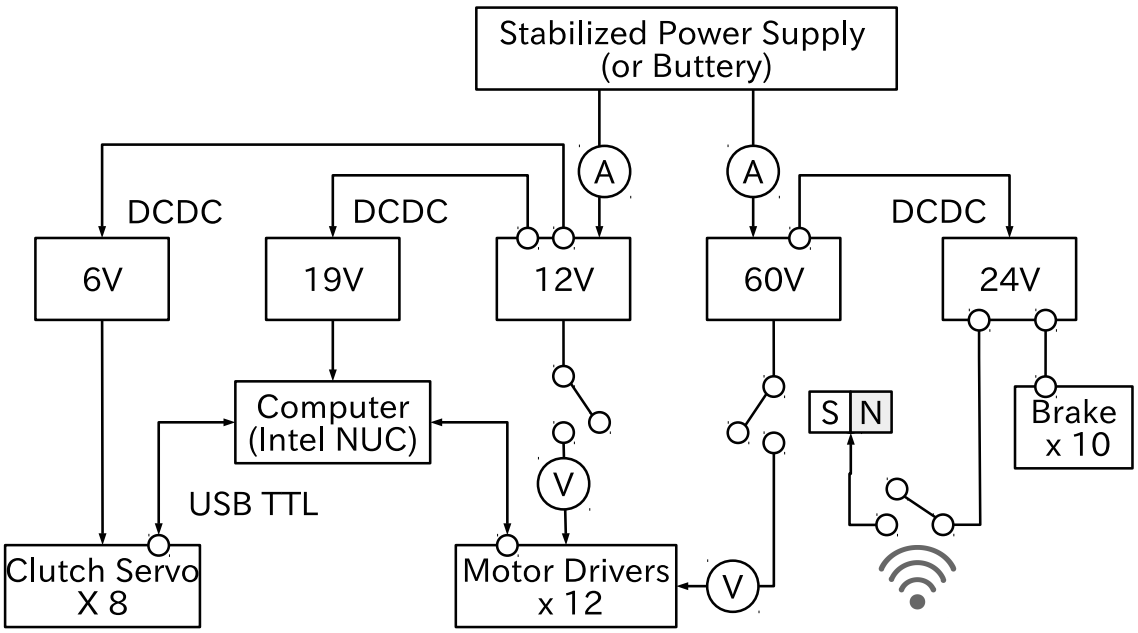


図 5.11. Electric system has two original power source: 12V and 60V direct current. DC-to-DC converter changes the 12V to 19 and 6 V, and 60 V to 24V. 60V is the motor power source and its flow is switched by DC power relay which is switched by wireless switch. 12V is the power source of logical circuit of motor dirvers, 19V is of computer, and 6V is of clutch servo motors.

御（図中右端 Euslisp）を別スレッドで統合したものになっている．図中左端に示すロボットはコンピュータ上の共有メモリと 1000 Hz でのリアルタイム通信を行いセンサデータの送信とモータドライバの位置制御目標値・位置制御ゲインの受信を行う．共有メモリは OpenRTM [120] を用いた制御フロー hrpsys [121] から読み込まれいくつかの RT コンポーネントを経由して位置制御目標値を計算，共有メモリへの書き込みを行う．センサデータとしてはモータのエンコーダ角位置，温度，電流電圧，リンクに取り付けられたアブソリュートエンコーダの角位置，姿勢センサの磁気センサ，加速度センサ，ジャイロセンサ，六軸力センサの値がある．hrpsys の RT コンポーネントとしては，hrpsys の歩行制御で用いるものを中心にいくつか図に示している．以下に簡単な説明をまとめる．

- SequencePlayer: 目標位置に向かって現在の関節位置から線形に補間する．図中 Euslisp で示す制御フローは SequencePlayer に位置目標値を渡すことでロボットを制御する．
- ForwardKinematics: 順運動学を計算する．
- KalmanFilter: Extended Kalman Filter を用いて加速度センサとジャイロセンサを用いて姿勢を計算する．
- AutoBalancer: 静的安定性を保つよう関節角度目標値を修正する．hrpsys を用いた歩行時には AutoBalancer が歩行軌道を生成する．Euslisp 制御フローが歩行軌道を生成

Table5.1. Robot link properties

	i	j	m_i	c_i	I_i			p_{ji}	R_{ji}		
NAME	ID	PID	[g]	[mm]	$[gm^2]$			[mm]			
BODY	0	-1	16917.0	−0.74 −3.12 133.10	160.033 −4.496 2.294	−4.496 162.546 −14.954	2.294 −14.954 204.884	-	-		
LLEG-LINK0	1	0	2068.0	0.02 28.59 −40.19	14.871 0.003 −0.001	0.003 13.080 −5.884	−0.001 −5.884 10.564	0.00 95.00 −19.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
LLEG-LINK1	2	1	4082.0	5.48 −10.73 −51.25	22.358 −0.239 −4.272	−0.239 21.994 2.239	−4.272 2.239 9.393	0.00 24.50 −91.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
LLEG-LINK2	3	2	2182.0	3.23 13.66 −108.15	12.184 −0.027 0.396	−0.027 12.095 −6.232	0.396 −6.232 6.049	8.50 −0.00 −139.50	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
LLEG-LINK3	4	3	3560.0	0.01 24.96 −71.06	26.143 −0.005 −0.007	−0.005 25.012 −9.904	−0.007 −9.904 7.178	−6.00 −10.50 −219.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
LLEG-LINK4	5	4	2956.0	7.55 7.55 −57.00	11.074 0.168 −3.827	0.168 11.074 −1.269	−3.827 −1.269 5.977	−0.00 −1.50 −225.50	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
LLEG-LINK5	6	5	1616.0	10.68 0.61 −58.47	4.398 0.029 −1.252	0.029 6.324 −0.135	−1.252 −0.135 3.809	8.50 −0.00 −114.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK0	7	0	2068.0	−0.02 −28.59 −40.19	14.871 0.003 0.001	0.003 13.080 5.884	0.001 5.884 10.564	0.00 −95.00 −19.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK1	8	7	4082.0	5.46 10.73 −51.25	22.358 0.241 −4.262	0.241 21.995 −2.244	−4.262 −2.244 9.394	0.00 −24.50 −91.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK2	9	8	2184.0	3.23 −13.64 −108.15	12.161 0.015 0.387	0.015 12.071 6.227	0.387 6.227 6.049	8.50 −0.00 −139.50	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK3	10	9	3560.0	−0.01 −24.96 −71.06	26.143 0.007 0.007	0.007 25.012 9.904	0.007 9.904 7.178	−6.00 10.50 −219.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK4	11	10	2956.0	7.55 −7.55 −57.00	11.074 −0.168 −3.823	−0.168 11.074 1.265	−3.823 1.265 5.977	0.00 1.50 −225.50	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000
RLEG-LINK5	12	11	1616.0	10.32 −0.26 −58.47	4.399 0.014 −1.169	0.014 6.336 0.065	−1.169 0.065 3.821	8.50 −0.00 −114.00	1.000 0.000 0.000	0.000 1.000 0.000	−0.000 0.000 1.000

するときには AutoBalancer は何もしない.

- Stabilizer: 力センサの値を用いて目標 ZMP を満たすように関節角度をリアルタイムに修正する．Euslisp 制御フローがバランス制御を行っているときには Stabilizer は何もしない.
- CollisionDetector: ロボットのリンク間衝突計算を行い，自己衝突が起こるような目標

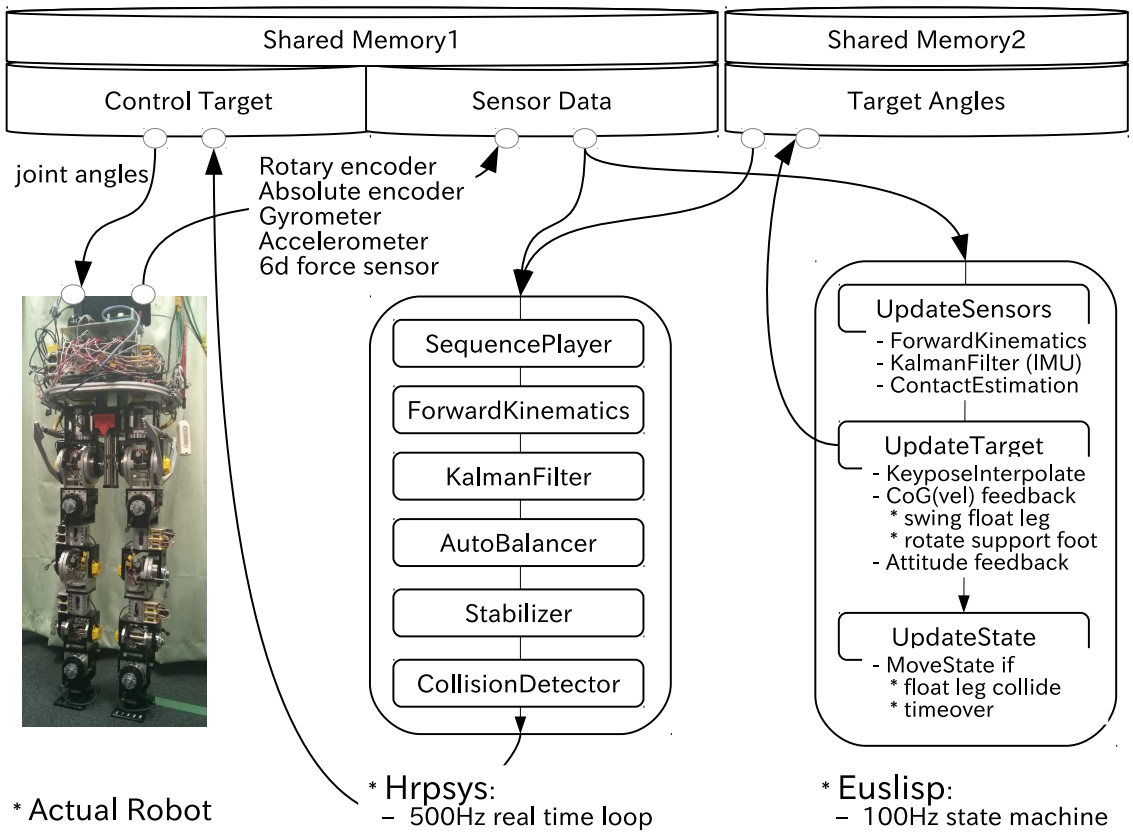


図 5.12. There exist two real time control flows. First one is hrpsys loop which uses OpenRTM, interpolates target joint angles in 500Hz, and check self collision. Second one is state machine loop which calculates target joint angles to interpolate in Hrpsys with 100Hz. These loops communicate with each other and actual robot through shared memory. The shared memory is separated into two sections. First one is used to copy in both directions: from motor drivers to computer, and from computer to motor drivers in 1000Hz. Second one is used to communicate state machine and hrpsys.

関節角度が与えられたときにはそれを弾く．

図中 Euslisp で示す制御フローが歩行軌道，バランス制御を行っているときには hrpsys は目標関節角度の補間と自己衝突計算しか行わない．Euslisp 制御フローは共有メモリからセンサデータを読み込み目標関節角度を計算し，ロボットと hrpsys がやりとりを行う共有メモリ 1 とは別の共有メモリ 2 にその目標関節角度を書き込む．hrpsys の SequencePlayer は，毎周期共有メモリ 2 を読み位置目標値が更新されたことを示すフラグ立っている場合は位置目標値を読み込み補間して 共有メモリ 1 に位置目標値を書き込むことで実ロボットを動作させる．Euslisp 制御フローと hrpsys は独立の Kalman filter により姿勢を推定するが，これはのちに示すように Euslisp 制御フローが動力学シミュレーション上のロボットで行動を最適化するため hrpsys に依存しない実装を用いる必要があったためである．

5.2.3　オープンソース制御系を用いた動歩行実験と消費エネルギー量

図 5.13. Snapshots of dynamic walking motion by using hrpsys which is contained in ML1 control flow. The walking control of hrpsys is general, and does not depend on any robot specific parameter. Therefore, it is easy to test walking. In this figure, ML1 walk 0.5 [m] in about 30 seconds. The motion is very slow because the balance control of hrpsys uses 6-axis contact force sensors of both legs, but ML1's force sensor could measure less than 10 [Nm]. As the result, fast walking fails to balance due to the inertial force of ML1's self weight.

図 5.12 に示すように ML1 は hrpsys による制御ループと Euslisp で示す制御ループの二つを有する．hrpsys に実装された動歩行制御はロボットモデルを用いた重心軌道最適化と，

その重心軌道を満たすような全身軌道生成（逆運動学）により歩行動作を生成する．したがってロボットのモデルが与えられさえすれば，それ以外にロボット固有のパラメタはほとんど存在せず，フットステップの足幅を逆運動学が失敗しない程度にしたり，重心位置のモデル誤差を吸収するよう数ミリのオフセットを与えてやるだけで迅速に歩行行動をテストすることが可能である．

図 5.13 は hrpsys の動歩行を ML1 で行った際の様子を示している．0.5 m の歩行に 30 秒程度を要し，再現性を持って前歩き後ろ歩きが実現できた．0.5 m / 30 s は極めて低速な歩行動作であるが，これは hrpsys のバランス制御が足裏の六軸力センサのモーメント出力を用いて計算された ZMP を用いたものであるため，ML1 に搭載された定格 10 Nm のセンサではバランスできる領域が狭く（約 5cm），高速な歩行では ML1 の自重によって生じる慣性力を打ち消すことが難しいことが実験により確認され，最大限早く歩けるパラメタをヒューリスティックに決定した上での速度である．

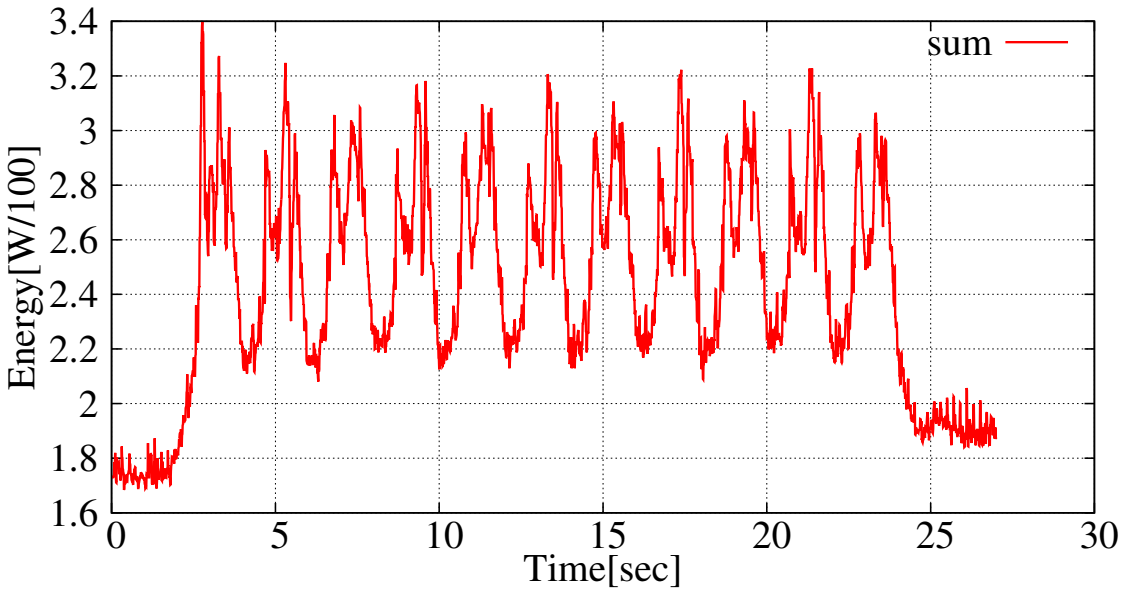


図 5.14. Energy comsumption while walking by using hrpsys. Total 6568.4 J for 0.5 m walking, therefore the CoT: Cost of Transport is $6568.4J/(49.8kg\times9.8m/s.5m) \approx 26.3$.

図 5.14 は歩行中の消費エネルギーを時間についてプロットしたグラフである．縦軸のエネルギーについては数百ワットのオーダーであるため見やすいよう百分の一した値を表示している．歩行は 11 歩行われ，図を見ると 11 回分のピーク消費エネルギーが確認できる．片足支持期中にピークを経て両脚支持期に移行するなかで消費エネルギーが減少し，再度片足支持

期に移るなかで増加に転じている．これを時間について積分したものが歩行に要した総エネルギーであり，それを移動距離（と重力の積）で割ったものは CoT: Cost of Transport と呼ばれ移動効率を評価する指標としてしばしば用いられる．図より CoT を計算すると，総消費エネルギーが 6568.4 J，ロボットの体重が 49.8 kg，重力加速度が 9.8 m/s^2 で，移動距離が 0.5 m であったので $6568.4/(49.8 \times 9.8 \times 0.5) \approx 26.9$ となる．これは [100] で示される他の最新のロボットと比べて数倍から十倍程度効率の効率の悪い歩行である．効率の悪い理由としては，前述の通り定格の小さい力センサにより ZMP を計測しバランス制御を行っているため歩行速度を高めることが難しい点が一つあげられる．移動量は CoT に逆比例するため，歩行速度を倍にできれば単純に CoT も半分になるためである．移動速度の他にもう一点，膝を曲げた歩行であり主に膝関節に高負荷がかかるためにモーターとモータードライバの消費する電流量が高くなり，結果として消費エネルギーが増えている点が課題としてあげられる．膝を伸ばし接触力を関節モータではなくロボットの骨格で受けることでモータが位置を保持するのに必要なエネルギーを小さくできるが，膝を伸ばしすぎるとバランス制御で用いる逆運動学計算において特異点の問題が生じることが知られており，完全に膝を伸ばした歩行を hrpsys の実装を用いて行うことは難しい．

5.3　動力学演算を用いた実体の身体環境モデルのパラメタ修正

5.3.1　実体モデルパラメタ修正に基づく行動の忠実再現法

5.3.1.1　探索空間：実体の身体環境モデルのパラメタについて

Table5.2. Contact parameters

Name	Unit	Description	Range	Dim
f_{thr}	N	Threashold to check if contact or not	$\pm 15N$	1
k_c	N/m	P gain of PD calculation of contact vertical force	$\pm 10\%$	1
d_c	Ns/m	D gain of PD calculation of contact vertical force	$\pm 10\%$	1
k_f	N/m	P gain of PD calculation of contact horizontal force	$\pm 10\%$	1
d_f	Ns/m	D gain of PD calculation of contact horizontal force	$\pm 10\%$	1
μ_f	-	Friction coefficience	± 0.4	1
Total				6

Table5.3. i-th joint parameters

Name	Unit	Description	Range	Dim
m_i	kg	weight of i-th joint	$\pm 1kg$	1
c_i	m	Centroid seen from i-th joint	$\pm 0.1m$	3
I_i	kgm^2	Inertia matrix around c_i	$\pm 0.01kgm^2$	9(6)
$\boldsymbol{p}_{ji}$	m	i-th joint position seen from j-th joint	$\pm 0.001m$	3
r_{ji}	deg	i-th joint orientation seen from j-th joint	$\pm 0.1deg$	3
k_i	Nm/rad	P gain of PD position control of i-th joint	$\pm 10\%$	1
d_i	Nms/rad	D gain of PD position control of i-th joint	$\pm 10\%$	1
Total				18

推定する身体と環境のパラメタについて表にまとめる．表 5.2 は環境とロボットの接触力計算に用いるパラメタである．表 5.3 はロボットの各関節ごとに定義されるパラメタである．12 関節 13 リンクのロボットでは $13 \times 18 + 6 = 240$ 次元の空間で探索を行う．

5.3.1.2 評価関数：再現度評価のための運動結果距離関数について

前章で述べた探索空間を，動力学演算器の出力する運動結果の差が小さくなるように探索することで実体の身体環境モデルのパラメタを推定することができる．運動結果は物理演算の出力と実世界の両方で同様に取得できる値を用いなければ差を計算できない．以下では，各関節の各位置センサの出力する関節角度度とその目標値，姿勢推定の結果得られるクオータニオンをシミュレーション時間で百 Hz でログをとり，シミュレーション間でそのログの差の二乗和平均を再現度関数として，これを最小化する．時間に対する関節角度ベクトルの配列をインデックス $i \in (0, M]$ を用いて $\boldsymbol{q}_{enc}(i)$，その目標値を $\boldsymbol{q}_{ref}(i)$ クオータニオンを $\boldsymbol{q}_{rot}(i)$ と表すことにし，シミュレーションでは $'$ をつけて $\boldsymbol{q}'_{enc}(i)$，$\boldsymbol{q}'_{ref}(i)$，$\boldsymbol{q}'_{rot}(i)$ のように区別する．ロボットの自由度を N，クオータニオンは 4 次元であることからそれぞれの次元数は N, N, 4 となり再現度 κ を式（5.1）から計算する．

$$\kappa = \frac{\sum_i (\boldsymbol{q}_i - \boldsymbol{q}'_i)^T (\boldsymbol{q}_i - \boldsymbol{q}'_i)}{M\,(N + N + 4)} \tag{5.1}$$

$$\boldsymbol{q}_i = \boldsymbol{q}_{enc}(i) + \boldsymbol{q}_{ref}(i) + \boldsymbol{q}_{rot}(i)$$
$$\boldsymbol{q}'_i = \boldsymbol{q}'_{enc}(i) + \boldsymbol{q}'_{ref}(i) + \boldsymbol{q}'_{rot}(i)$$

また，複数の行動について再現度を計算する場合は，それぞれの再現度関数の返す値のなかで最大のものを再現度として用い最小化した．行動 $j \in [0, L)$ についての再現度 κ_j を用いて，式（5.2）を計算する．

$$\kappa = max\,(\kappa_0 \cdots \kappa_{L-1}) \tag{5.2}$$

加えて転倒したかどうかという情報も考慮する．シミュレーションと実ロボットでの運動結果が転倒の意味で異なっている場合，例えば実ロボットは転倒するような運動モデルのパラメタがシミュレーションでは転倒しなかった場合は，κ におおきな値（実際は 100 を用いた）を入れることでそのような実体身体環境モデルパラメタを生成しないよう工夫した．

5.3.2 実体モデルパラメタ修正に基づく行動の忠実再現実験

本章では，前章に述べた身体パラメタ推定と歩行行動の忠実再現法を用いて，実際に脚型ロボット ML1 を用いた歩行行動の忠実再現実験を行う．実験は，運動モデルパラメタ生成（Generation Phase），動作実行（Execution Phase），実体身体環境モデルパラメタ推定（Identification Phase）を繰り返すことで行う．本章ではの実験データが比較的多いため別章 5.5 章にまとめて示すことにする．

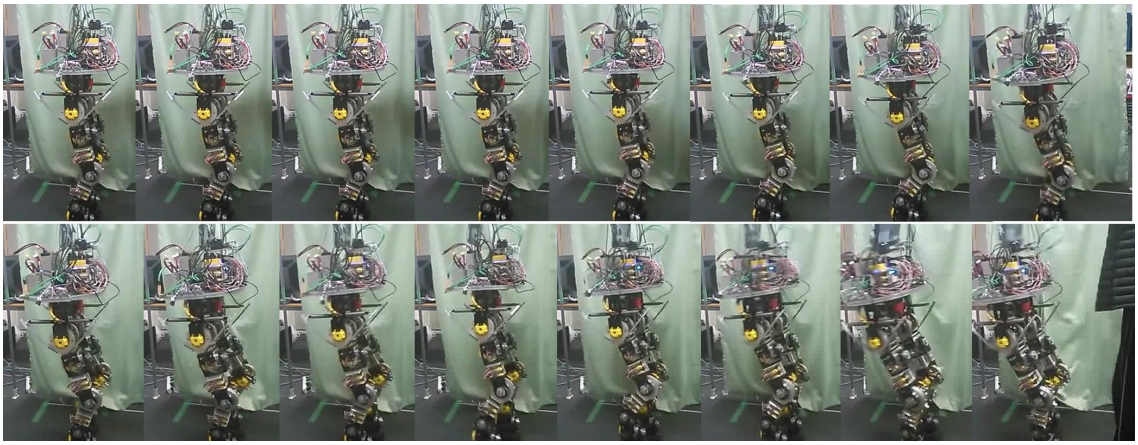


図 5.15. Snapshots of walking motion in real world, but failed.

5.3.2.1　発見的方法による実体の身体環境モデルのパラメタ推定と行動再生成実験

5.3.2.1.1　Generation Phase1

まず初めに人間がヒューリスティックに作成した歩行動作について身体パラメタ推定を行う．各ステートの持つ目標関節位置 $\boldsymbol{q}_0$ を 5.5.1 章図 5.25 に示す．Start/Stop 姿勢は膝を伸ばした姿勢から，足先が 30cm 上に移動するよう逆運動学を解いて生成した姿勢である．Stance0 はさらにそこから重心位置を左足に向かって 65% 移動させたもの，Swing0 は重心を 80% 左足に移動させたのち右足を 60mm 上側に 120mm 前側に移動するよう逆運動学を解いたものである．Stance1 は Start 姿勢と同じもの．Swing1 は Start 姿勢から重心を 70% 右足に移動させたのち左足を 60mm 上側に 120mm 前側に移動するよう逆運動学を解いたものである．Stance2, Swing2 はそれぞれ Stance1, Swing1 を左右反転させた姿勢．SwingN, StanceN はそれぞれ Swing0, Stance0 と同じものを用いた．

5.5.2.1 章図 5.31 は歩行中に加速度センサ・ジャイロセンサ・磁気センサシミュレーションの値を用いて推定された三次元回転姿勢のクオータニオン表現である．また，図 5.32 は歩行中のステート遷移について表したものである．0 から 9 までのステートがそれぞれ， Start, Stance0, Swing0, Stance1, Swing1, Stance2, Swing2, SwingN, StanceN, Stop に対応している．九歩目で停止していることが分かる．5.5.2.1 章図 5.33 に各時刻における関節角度目標値と，シミュレーションされた関節位置を重ねてプロットしたものを示す．

5.3.2.1.2　Execution Phase 1

次に同じパラメタを，動力学シミュレーションではなく実ロボットで実行した際の様子を図 5.15 に示す．歩行の停止ステートへの遷移も同様に五秒すぎた後最初の Stance2 から遷移する実装を用いた．結果，図のように転倒した．

シミューション実験と同様にいくつかのセンサデータをグラフに表示する．グラフは転倒しハンガーでロボットがつられる直前までの 2.5 秒間のみについて表示している．5.5.2.2 章図

5.34 は歩行中に推定された姿勢センサのクオータニオンを表している．また，図 5.35 は歩行中のステート遷移について表したものである．0 から 9 までのステートがそれぞれ， Start, Stance0, Swing0, Stance1, Swing1, Stance2, Swing2, SwingN, StanceN, Stop に対応しているのも同様である．最後に関節角度のエンコーダによる測定値と目標値を重ねたグラフを 5.5.2.2 章図 5.36 に示す．

5.3.2.1.3 Identification Phase 1

前章までで得られた実機とシミュレーションのログデータを用いてロボットの実体の身体環境モデルのパラメタを推定する．5.5 章図 5.67 に探索時の再現度式（5.2） を縦軸に，探索にかかった時間を横軸にとったグラフを示す．実機とシミュレーションのログには，5.3.2.1.1 章でつくった歩行動作と，4.3.3.1 章でつくったその場足踏み動作の二つを用いた．二日間の探索で 0.00005.9 rad^2 （0.44 度） まで差を縮めることができた．また得られた実体の身体環境モデルのパラメタは，5.5 章表 5.6 に示している．CAD 図面等ロボット設計段階での実体の身体環境モデルのパラメタは 5.5 章表 5.5 に示している．得られた実体の身体環境モデルのパラメタから重心位置を計算し，どの程度 CAD 値から離れているかを計算したところ，後ろに 0.20 mm，左に 1.67 mm，下に 55.70 ずれている結果となった．重心が下側に移動しているということは足が CAD 値よりも重くなっているということである．転倒した理由を考察すると，実ロボットで歩行した際は足を前に出す加速度運動による慣性力がシミュレーションよりも大きくなり後ろに倒れてしまったのだと予想できる．

実機の挙動がシミュレーションで再現されることを確認するため，再度前章の実験で失敗した歩行動作をシミュレーション上で実行する．5.5.2.3 章に示される図が 5.5.2.2 章の図を元に推定された実体の身体環境モデルのパラメタを用いてシミュレーションされた歩行動作である．良く特徴を捉えていることが確認できる．

5.3.2.1.4 Generation Phase 2

得られた実体の身体環境モデルのパラメタを用いて動作を再生成する．本章ではもっとも簡単な方法として，重心位置をオフセットする手法を用いる．後ろに倒れることが分かっているので，前に重心を移動させれば転倒しないと考えられる．シミュレーション上で重心位置を 5 mm 刻みでオフセットさせ動作を確認した．結果を以下の表 5.4 に示す．

Table5.4. Centroid offset v.s. CoT

c_{offset} [mm]	0	5	10	15	20	25	30	35	40	45
CoT	err	err	19.8	15.7	14.0	13.2	12.7	12.7	err	err

上行が重心位置のオフセット値であり下行が転倒した場合は err，転倒しなかった場合は CoT を計算した．10 mm 以上オフセットさせることで歩行できるが，40mm 以上では逆に前向きに転倒してしまうことが確認された．CoT をみてみるとオフセット値が大きいほど下がるという結果になった．これは重心が前にずれいている場合，バランス制御により足がより前

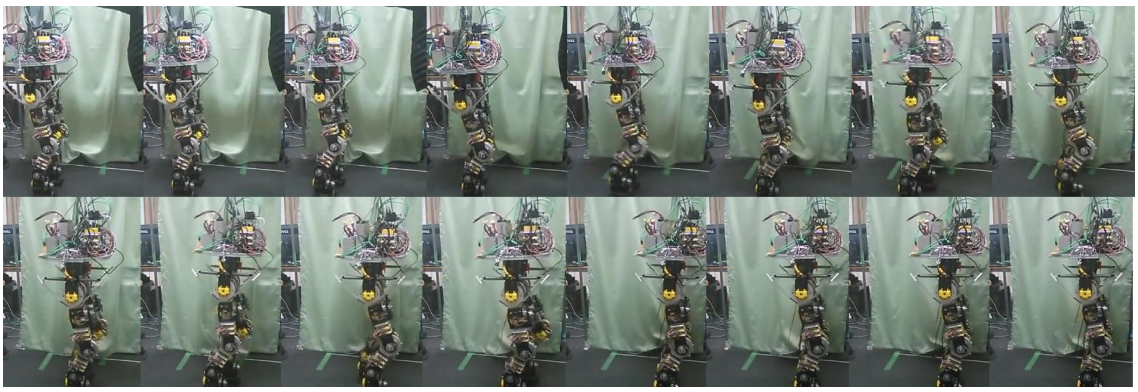


図 5.16. Snapshots of walking motion in real world.

へと伸びることによって歩行の速度が大きくなり，結果として CoT が小さくなっていると考えられる．逆に 10 mm 付近の動作では，後ろに倒れそうになるのを足が後ろにでることで防いでいる状態であり前への移動量が小さいため CoT は大きい．歩行の安定性という面では歩行可能なオフセット値の中央値付近が安定と考えられるため，以下では 20 mm のオフセットを用いた動作を実ロボットで試していく．20 mm オフセットでのシミュレーションでの動作のログは 5.5.2.4 章の各図に示される．

5.3.2.1.5　Execution Phase 2

前章で生成した重心位置を 20mm オフセットさせた歩行動作を実ロボットで実行した．その際の様子を図 5.16 に示す．歩行の停止ステートへの遷移も同様に五秒すぎた後最初の Stance2 から遷移する実装を用いた．図のように安定して歩行できることが確認された．

5.5.2.5 章の各図に運動結果のグラフを示す．図 5.43 は歩行中に推定された姿勢センサのクオータニオンを表している．図 5.44 は歩行中のステート遷移について表したものである．関節角度のエンコーダによる測定値と目標値を重ねたグラフを図 5.45 に示す．このときの再現度は 0.0016 rad^2 （2.28 度）であった．再現度 2.28 度の行動が実現された．

最後に歩行中の消費エネルギーについてグラフに示す．縦軸のエネルギーについては数百ワットのオーダーであるため見やすいよう百分の一した値を表示している．hrpsys を用いた歩行では $CoT \approx 26.9$ であったが，この実験で用いた歩行動作では約 10.5 となり，2.5 倍程度エネルギー効率が改善された．

5.3.2.1.6　Identification Phase 2

最後に，重心位置補正後の運動結果ログを用いて実体の身体環境モデルのパラメタを再度推定する．5.5 章図 5.68 に探索時の再現度式（5.2） を縦軸に，探索にかかった時間を横軸にとったグラフを示す．実機とシミュレーションのログには，5.3.2.1.1 章でつくった歩行動作と，4.3.3.1 章でつくったその場足踏み動作も用い，追加で前章の重心位置補正後の歩行動作のログも用いた．二日間の探索で 0.00025 rad^2（0.91 度）まで誤差を減らすことができた．得

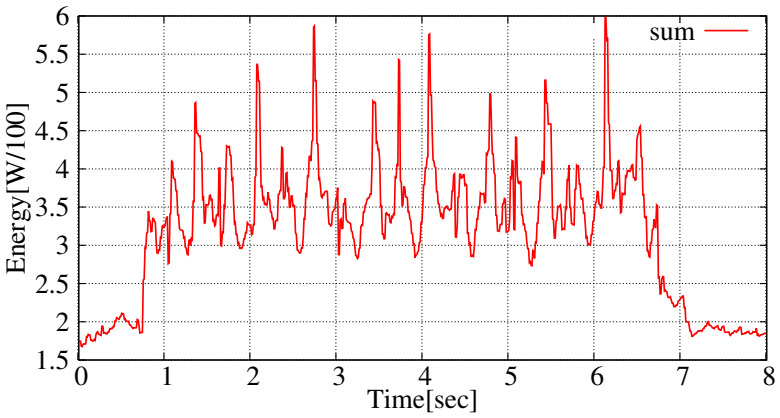


図 5.17. Energy comsumption while walking by using hrpsys. Total 2553.8 J for 0.5 m walking, therefore the CoT: Cost of Transport is $2553.8J/(49.8kg \times 9.8m/s \times 0.5m) \approx 10.5$, which is about 2.5 times lower than hrpsys walking.

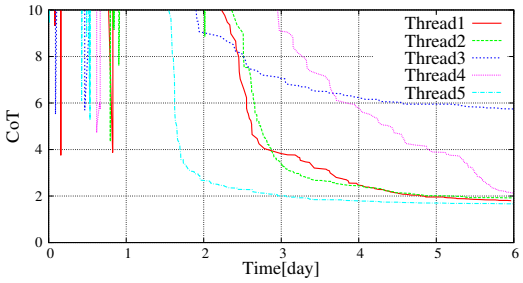


図 5.18. The transition of CoT: Cost of Transport while searching walking motion.

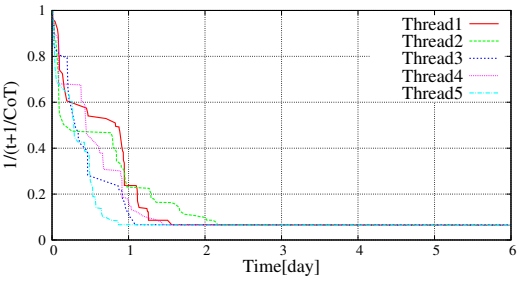


図 5.19. The transition of objective function $(\frac{1}{t+\frac{1}{CoT}})$ while searching walking motion.

られた実体の身体環境モデルのパラメタは，5.5 章表 5.7 に示している．

5.3.2.2 実体の身体環境モデルのパラメタ推定と行動再生成実験

5.3.2.2.1 Generation Phase1

最適化された歩行実現を試みる．各ステートの持つ目標関節位置 $\boldsymbol{q}_0$ を 5.5.1 章図 5.27 に示す．Start/Stop 姿勢は全関節角度が零であるように与え，それ以外の姿勢は探索により初期値なしで決定された．シミュレーション上での歩行中の様子は図 5.28 に示される．また歩行探索時の評価関数，Cost of Transport (CoT) の遷移を図 5.19 と図 5.18 に示す．探索は五スレッド独立に行い一番よいものを用いた．評価関数は転倒までにかかった時間 t と CoT を用いて $\frac{1}{t+\frac{1}{CoT}}$ のように計算し，10 秒すぎたところで停止ステートに以降をはじめ 15 秒までシミュレーションを行った．したがって評価値が $\frac{1}{15} = 0.066...$ を下回っているスレッドでは転倒しない歩行動作が得られていることになり，図 5.19 をみると二日程度で全スレッドが歩行動作の生成には成功していることがわかる．一方 CoT のグラフ図 5.18 をみてみると

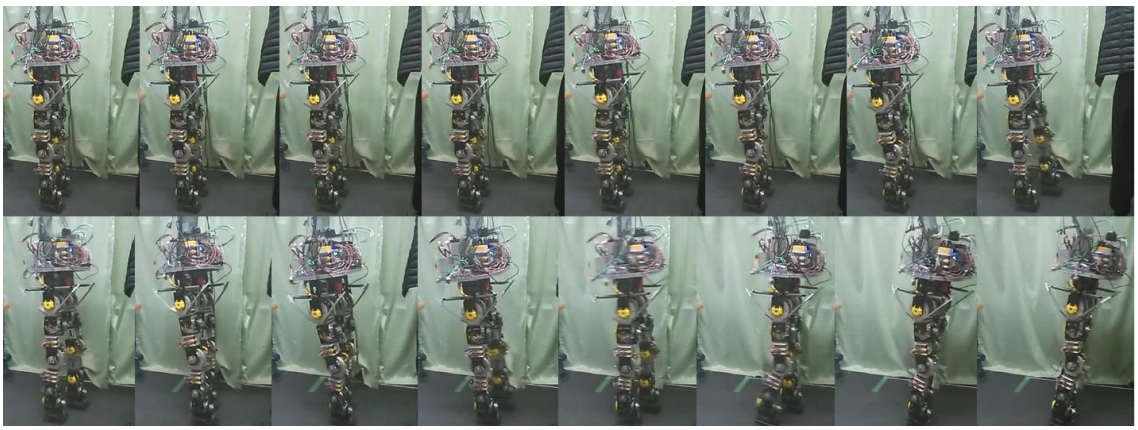


図 5.20. Snapshots of optimal walking motion in real world, but failed.

六日間の探索で四スレッドはほぼ等しい値 $CoT \approx 2.0$ 程度になっていることが分かるが一スレッドは比較的大きな値 $CoT \approx 6.0$ 程度となっており六日で必ずしも全スレッドが同様の答えを得られるわけではないことが確認された．なお，ここで計算された CoT は歩きはじめと歩き終わりを含んでいるが，止まらずに百秒程度歩きつづけるともっとも CoT の良い動作では 1.3 ≈ 1.4 程度になることが確認されている．

5.3.2.2.2　Execution Phase1

最適化された歩行を実ロボットで実行した際の様子を図 5.20 に示す．歩行の停止ステートへの遷移は五秒すぎた後最初の Stance2 から遷移する実装を用いたがその前に図のように転倒することが確認された．

5.5.2.8 章の各図に運動結果のグラフを示す．図 5.52 は歩行中に推定された姿勢センサのクオータニオンを表している．また，図 5.53 は歩行中のステート遷移について表したものである．最後に関節角度のエンコーダによる測定値と目標値を重ねたグラフを図 5.54 に示す．

5.3.2.2.3　Identification Phase1

前章の転倒したログを追加してロボットの実体の身体環境モデルのパラメタを推定する．5.5 章図 5.69 に探索時の再現度式（5.2） を縦軸に，探索にかかった時間を横軸にとったグラフを示す．実機とシミュレーションの運動結果ログとしては，5.3.2.1.1 章でつくった歩行動作と，4.3.3.1 章でつくったその場足踏み動作，5.3.2.1.4 章の重心位置補正 20mm の歩行動作と，前章の最適化された歩行動作を用いた．二日間の探索で 0.00057 rad^2（1.36 度）の再現度が得られることが確認された．探索の結果である実体の身体環境モデルのパラメタは 5.5 章表 5.8 に示している．

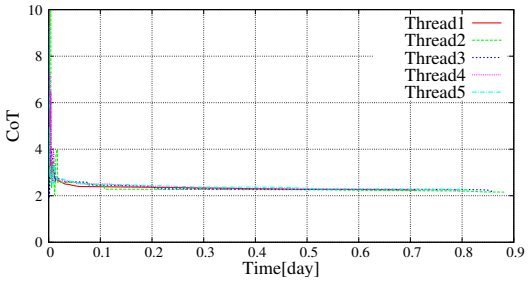


図 5.21. The transition of CoT: Cost of Transport while searching walking motion.

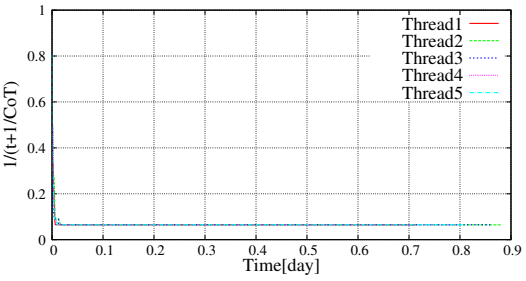


図 5.22. The transition of objective function $(\frac{1}{t+\frac{1}{CoT}})$ while searching walking motion.

5.3.2.2.4 Generation Phase2

再度歩行動作を最適化した．各ステートの持つ目標関節位置 $\boldsymbol{q}_0$ を 5.5.1 章図 5.29 に示す．Start/Stop 姿勢は全関節角度が零であるように与え，それ以外の姿勢は探索により一つ前の失敗した歩行を初期値として用いることで決定された．歩行中の様子は図 5.28 に示される．また歩行探索時の評価関数，Cost of Transport (CoT) の遷移を図 5.22 と図 5.21 に示す．探索は五スレッド独立に行い一番よいものを用いた．初期値として一つ前の実体の身体環境モデルのパラメタを用いて最適化されたものを用いたため探索は一日程度で全スレッドがほぼ変化しなくなっていることが確認できる．

5.3.2.2.5 Execution Phase2

実ロボットで実行した際の様子を図 5.23 に示す．歩行の停止ステートへの遷移は五秒すぎた後最初の Stance2 から遷移する実装を用いた．図のように安定して歩行できることが確認された．

実ロボットでの運動結果ログは 5.5.2.11 章の各図に示される．図 5.61 は歩行中に推定された姿勢センサのクオータニオンを表している．図 5.62 は歩行中のステート遷移について表したものである．関節角度のエンコーダによる測定値と目標値を重ねたグラフを図 5.63 に示す．

最後に歩行中の消費エネルギーについてグラフに示す．縦軸のエネルギーについては数百ワットのオーダーであるため見やすいよう百分の一した値を表示している．hrpsys を用いた歩行では $CoT \approx 26.9$ であったが，この実験で用いた歩行動作では約 4.6 となり，5 倍程度エネルギー効率が改善された．

5.3.2.2.6 Identification Phase2

最後に，これまで得られたすべての運動結果ログを用いて実体の身体環境モデルのパラメタを再度推定した．5.5 章図 5.70 に探索時の再現度式（5.2） を縦軸に，探索にかかった時間を横軸にとったグラフを示す．二日間の探索で 0.00068 rad^2（1.49 度）まで誤差を減らすことができた．得られた実体の身体環境モデルのパラメタは，5.5 章表 5.9 に示している．

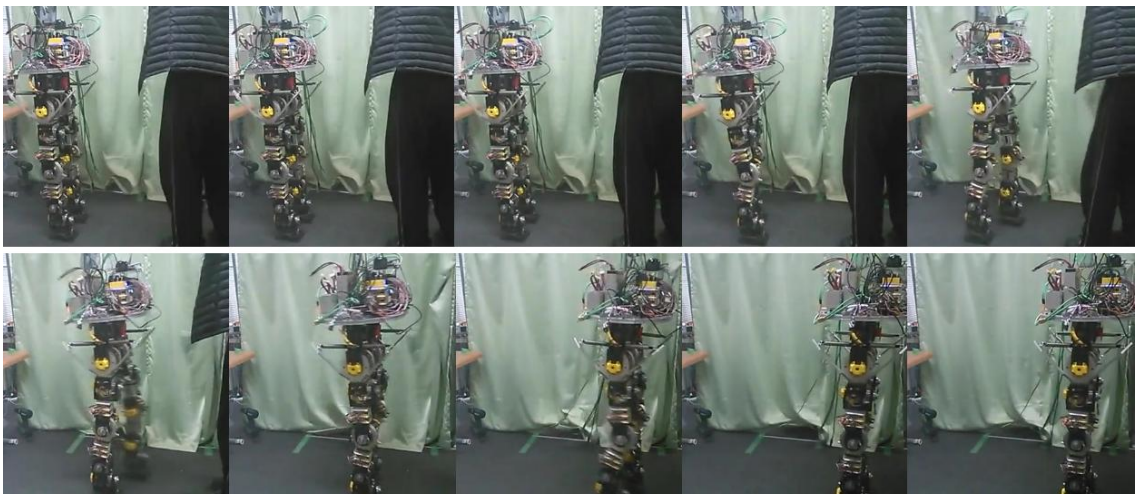


図 5.23. Snapshots of optimal walking motion in real world.

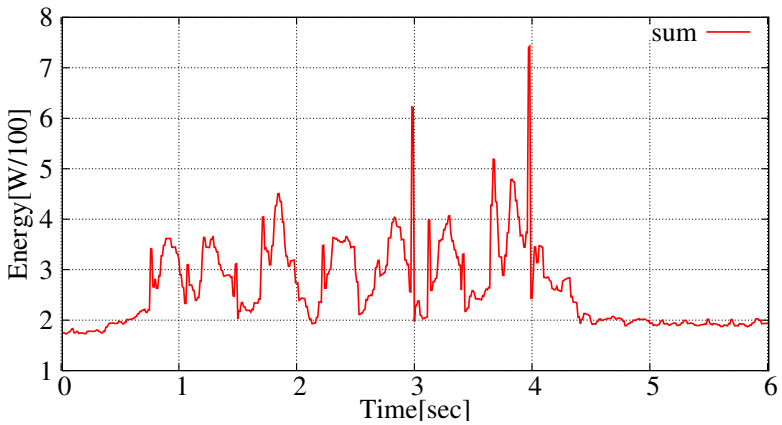


図 5.24. Energy comsumption while walking by using hrpsys. Total 1581.8 J for 0.7 m walking, therefore the CoT: Cost of Transport is $1581.8J/(49.8kg \times 9.8m/s \times 0.7m) \approx 4.6$, which is about 5 times lower than hrpsys walking.

5.4　おわりに

本章では前章で実装した動力学シミュレーションと進化計算を組み合わせた探索手法によって歩行行動探索と、実ロボットの身体モデルパラメタ推定を複数回行うことで動力学演算の実世界に対する忠実度を評価しながら歩行行動実現を達成した。また、本研究で開発した脚型ロボットの身体構成、電装系構成、制御ソフトウェア構成についてもここでまとめた。

5.5 補足：動力学演算を用いた実体の身体環境モデルのパラメタ修正実験ログ

5.5.1 歩行動作の各ステート姿勢と歩行の様子

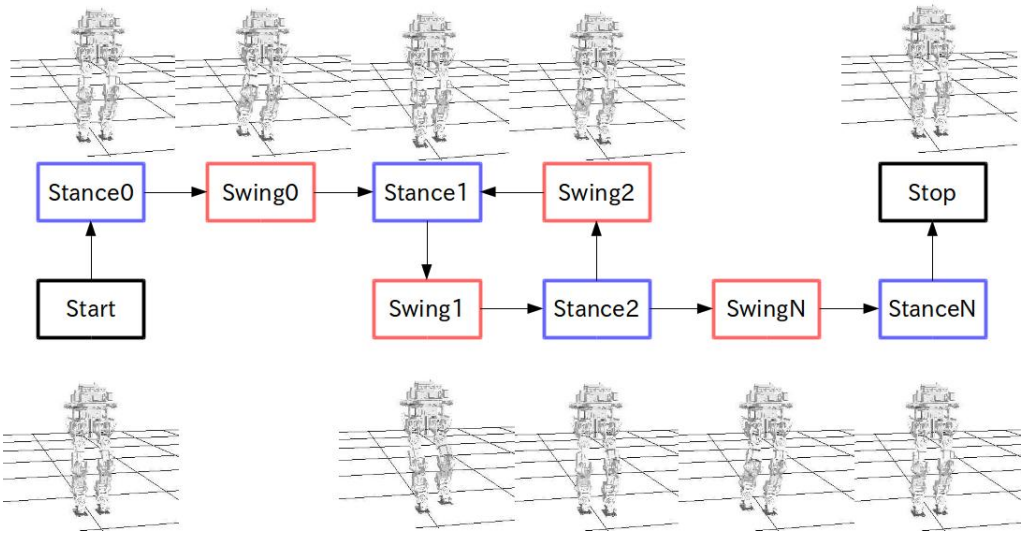


図 5.25. Walking motion is heuristically generated. Start/Stop postures are generated by moving both legs 30 mm upward. Stance0 posture is by moving CoG 65 % leftside. Swing0 is by moving CoG 80 % leftside and moving right foot 60 mm upward and 120 mm forward. Stance1 is the same as Start posture. Swing1 is by moving CoG 70 % rightside and moving left foot 60 mm upward and 120 mm forward. Stance2 is the horizontal flip posture of Stance1. Swing2 is the horizontal flip posture of Swing1. SwingN is the same as Swing0. StanceN is the same as Stance0.

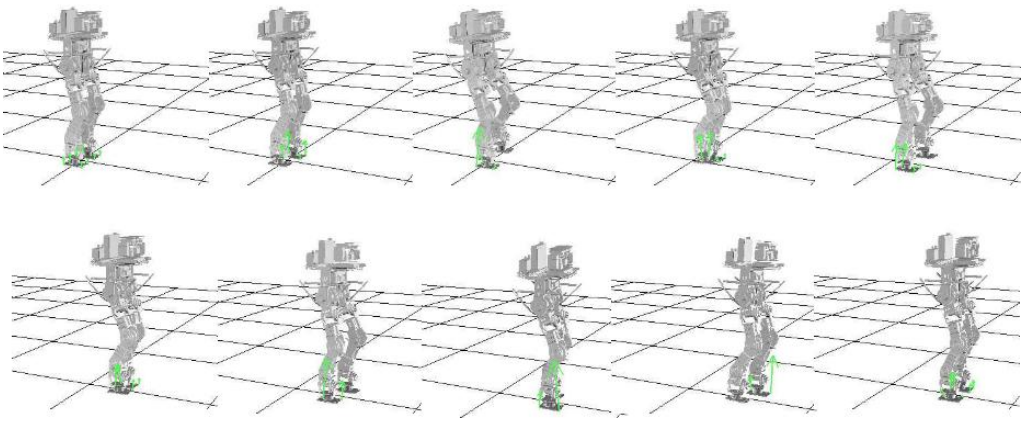


図 5.26. Snapshots of walking motion in simulation world.

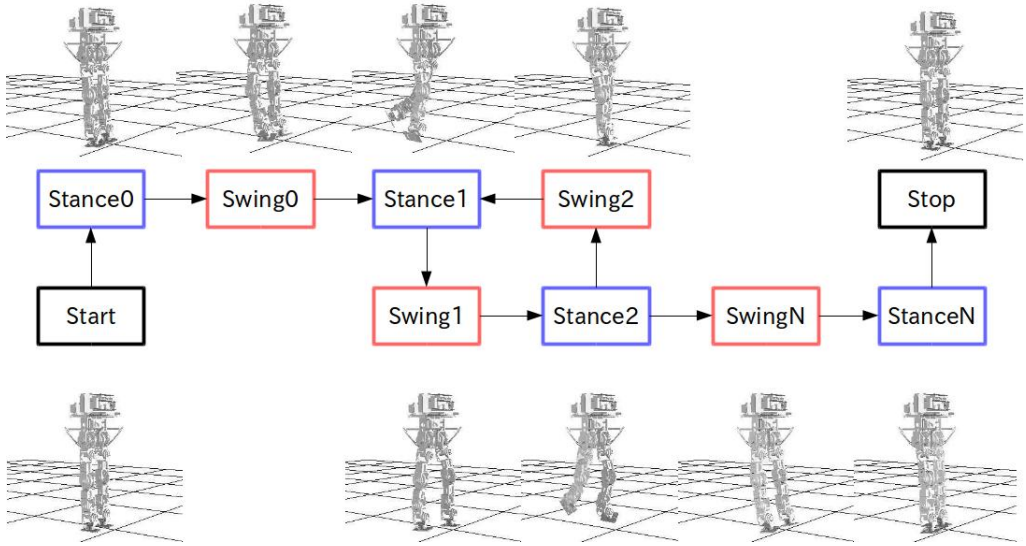


図 5.27. Walking motion is optimized. Start/Stop postures are given and all joint angles are zeros. The other postures are generated by optimization procedure without initial guess.

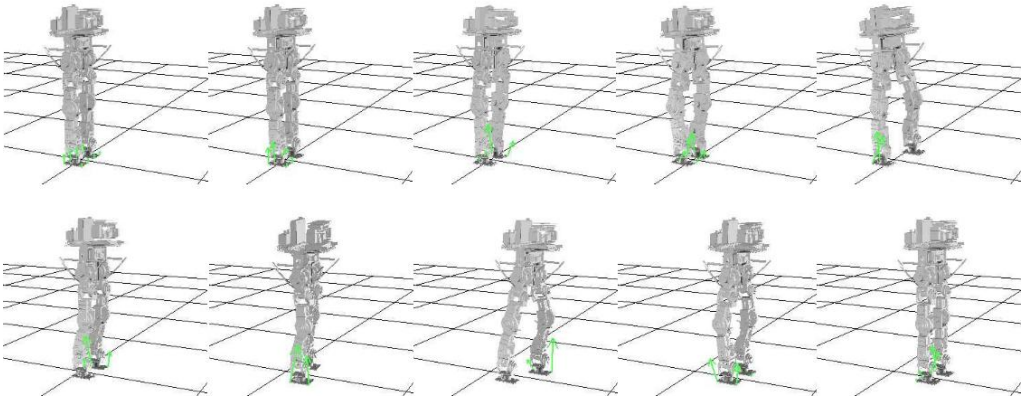


図 5.28. Snapshots of walking motion in simulation world.

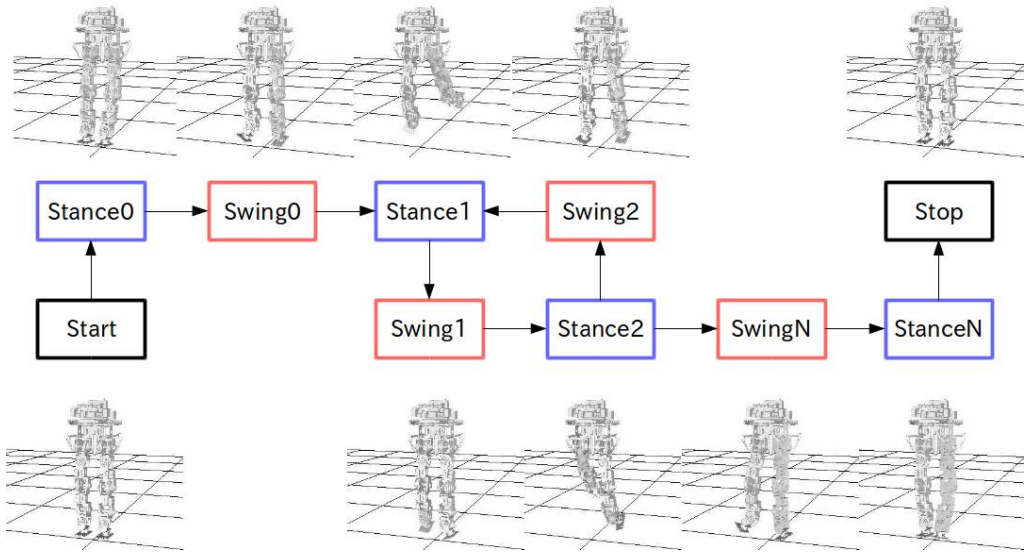


図 5.29. Walking motion is optimized. Start/Stop postures are given and all joint angles are zeros. The other postures are generated by optimization procedure without initial guess.

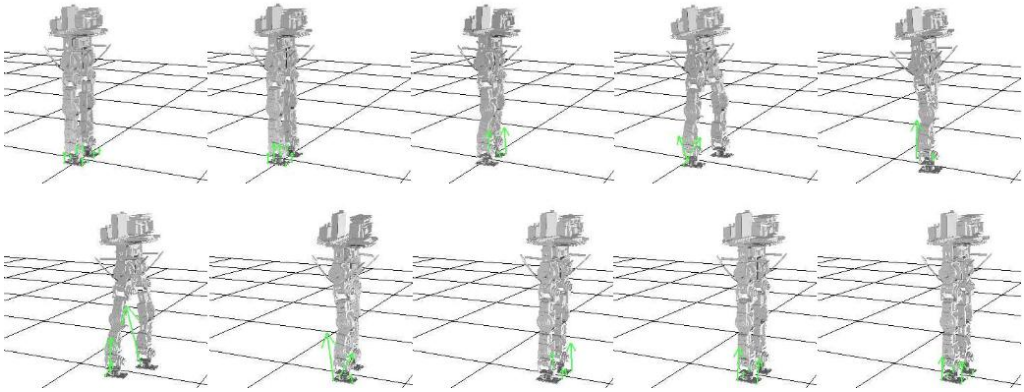


図 5.30. Snapshots of walking motion in simulation world.

5.5.2 実行された運動結果ログ

5.5.2.1　シミュレーションでの膝曲げ歩行

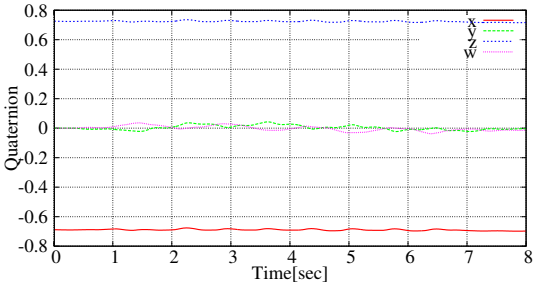


図 5.31. IMU Quaternion

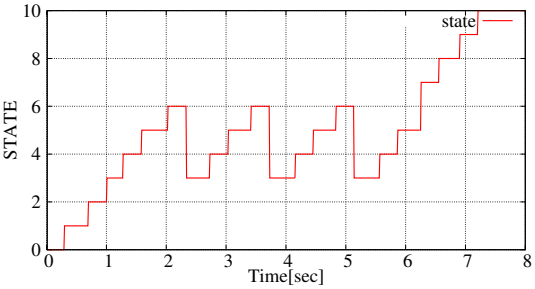
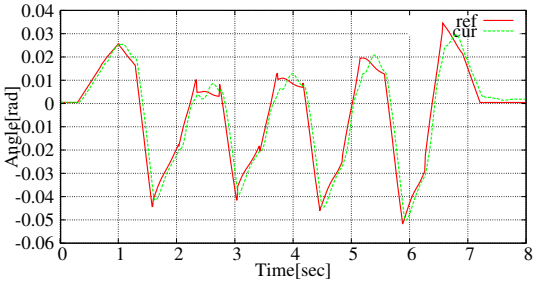
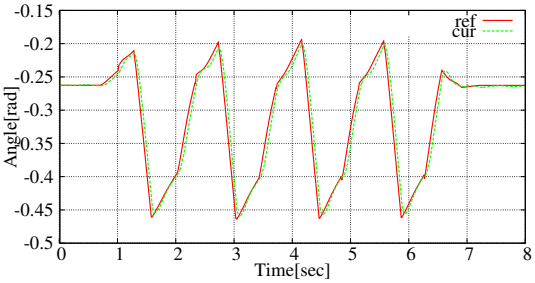


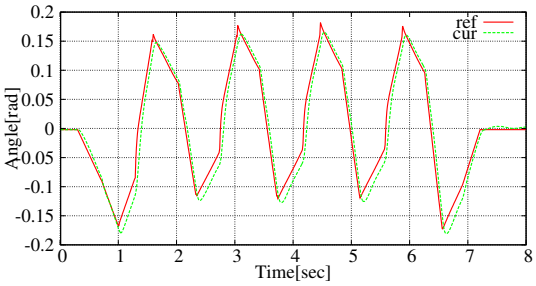
図 5.32. SIMBICON states



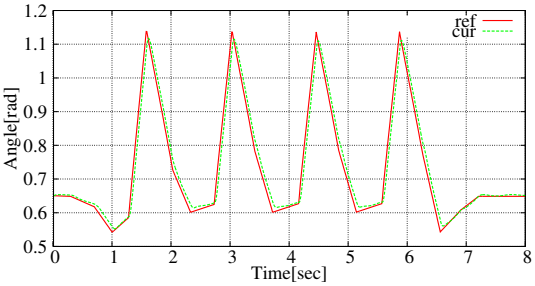
ID 1



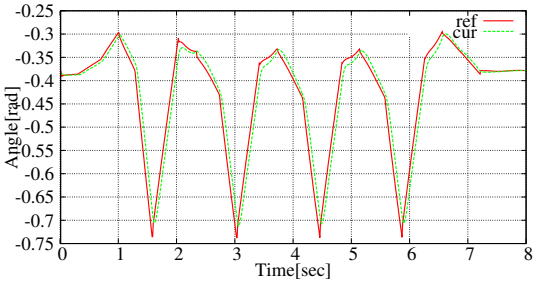
ID 2



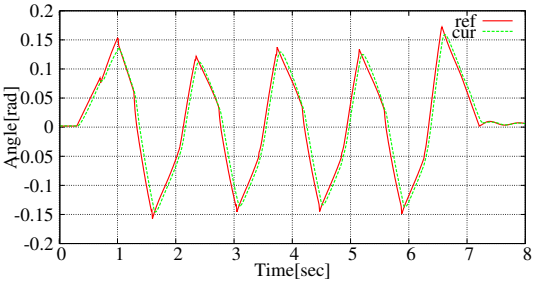
ID 3



ID 4



ID 5



ID 6

図 5.33. Joint angles

5.31. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.32. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.33. Joint angle targets and simulated values for left six joints while walking motion. Because the walking motion is symmetric, right joints move similar to left joints.

5.5.2.2 実ロボットでの膝曲げ歩行

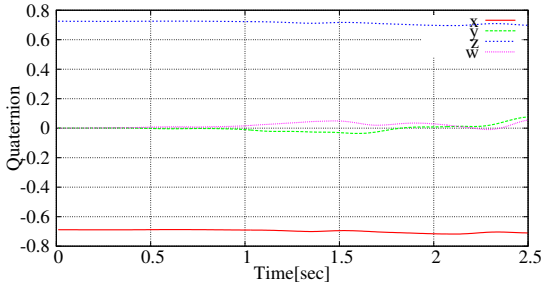


図 5.34. IMU Quaternion

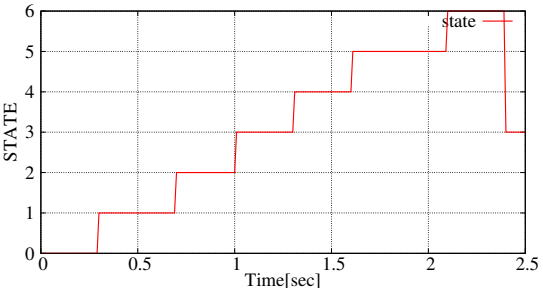
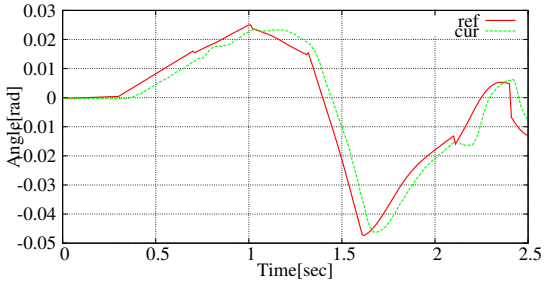
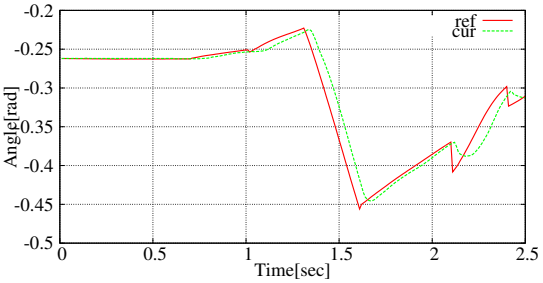


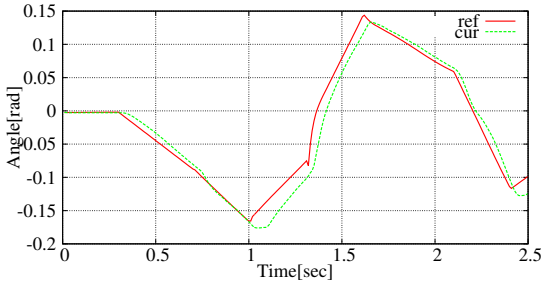
図 5.35. SIMBICON State



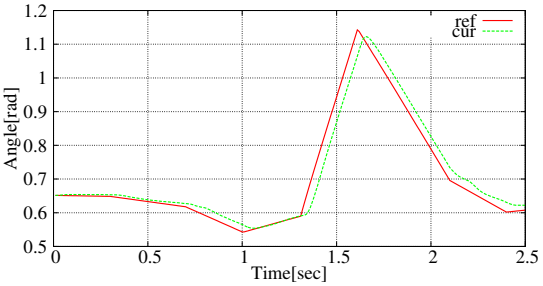
ID 1



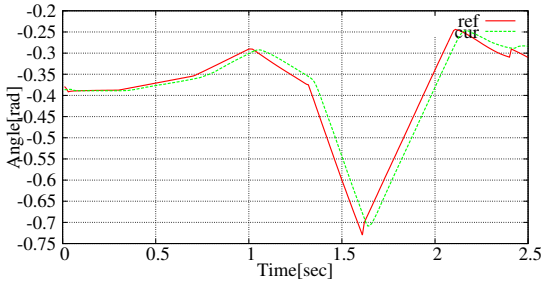
ID 2



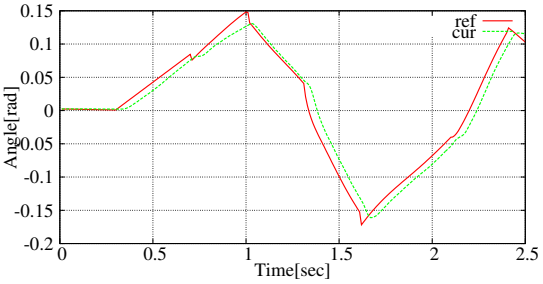
ID 3



ID 4



ID 5



ID 6

図 5.36. Joint angles

5.34. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.35. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.36. Joint angle targets and measured values for left six joints while walking motion of actual robot.

5.5.2.3　実体の身体環境モデルのパラメタ推定後のシミュレーションでの膝曲げ歩行

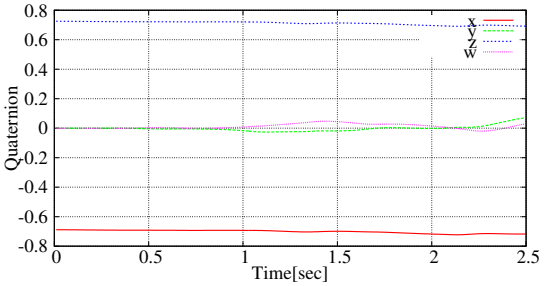


図 5.37. IMU Quaternion

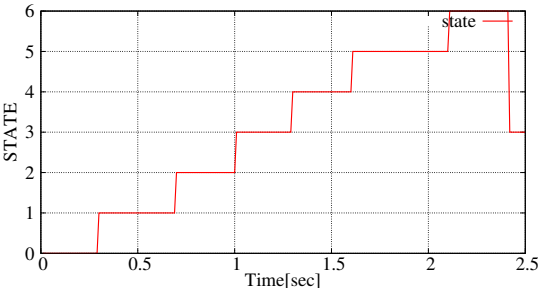
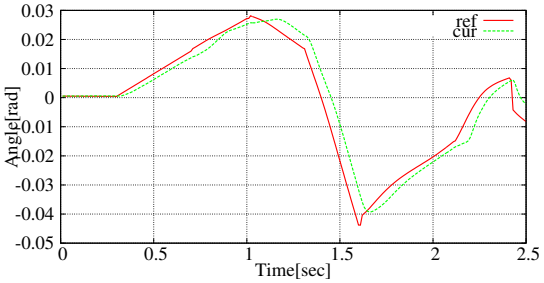
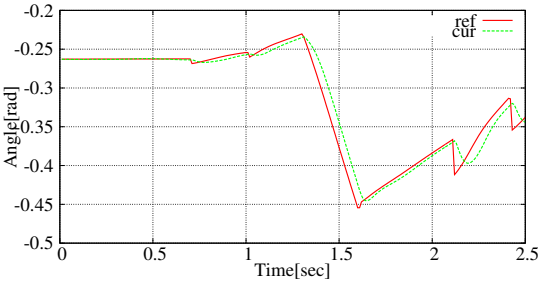


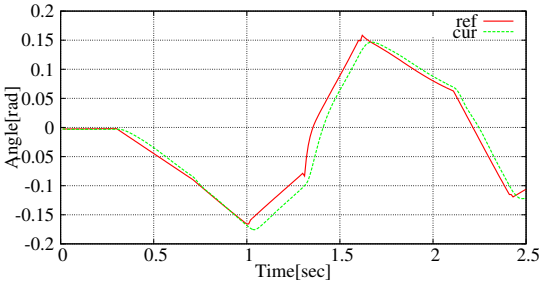
図 5.38. SIMBICON State



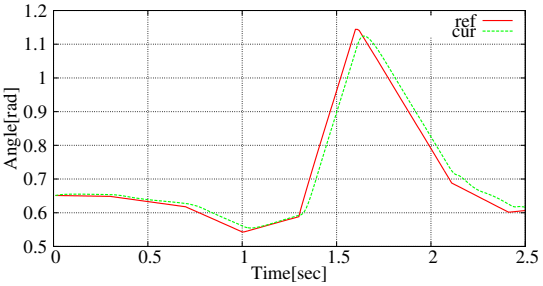
ID 1



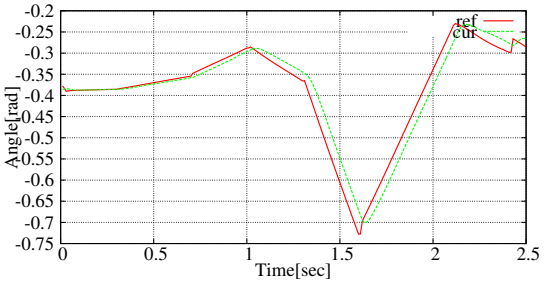
ID 2



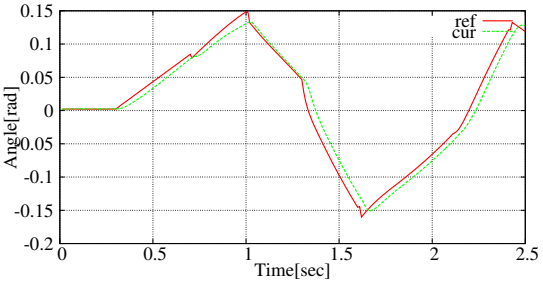
ID 3



ID 4



ID 5



ID 6

図 5.39. Joint angles

5.37. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.38. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.39. Joint angle targets and measured values for left six joints while walking motion of actual robot.

5.5.2.4 シミュレーションでの膝曲げ歩行重心補正 20 mm

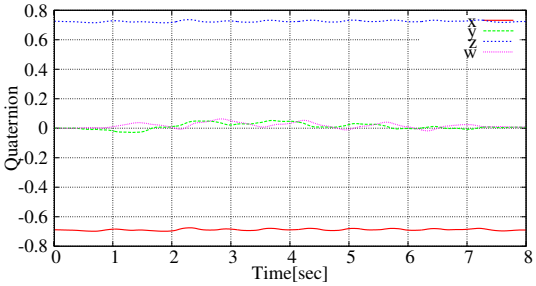


図 5.40. IMU Quaternion

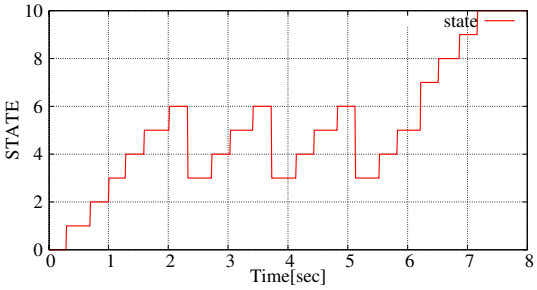
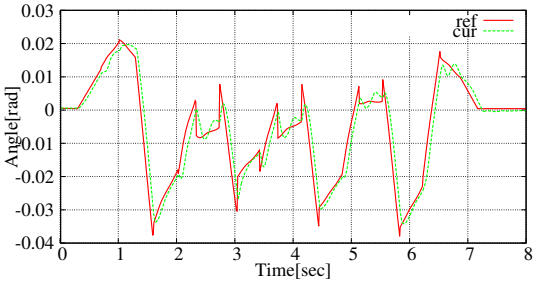
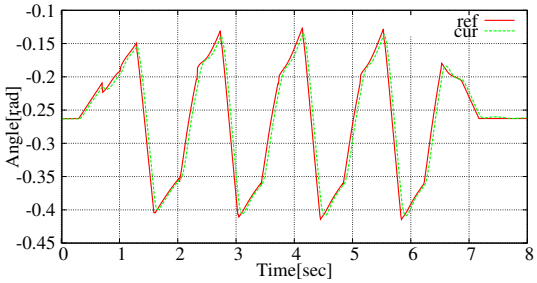


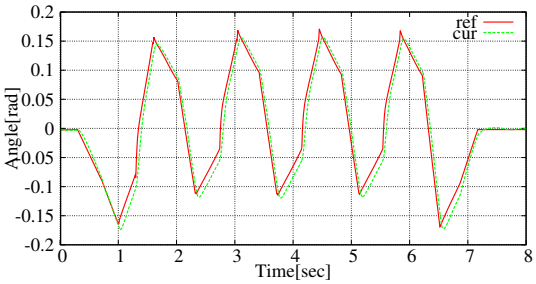
図 5.41. SIMBICON State



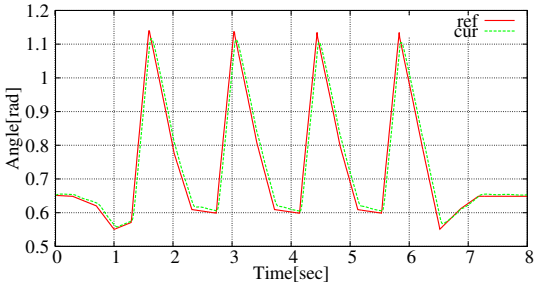
ID 1



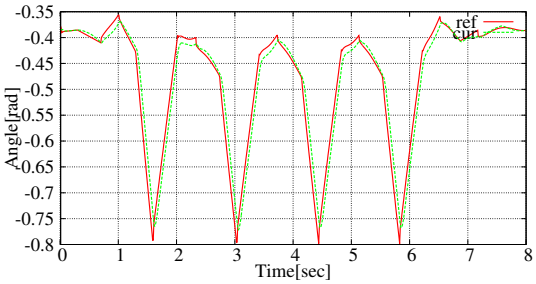
ID 2



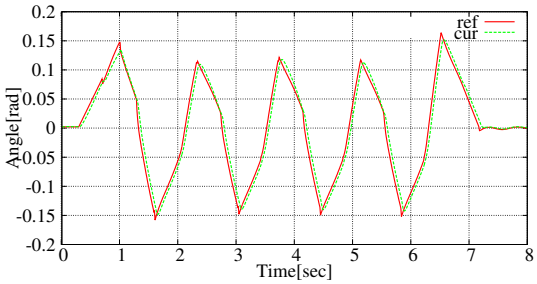
ID 3



ID 4



ID 5



ID 6

図 5.42. Joint angle

5.40. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.41. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.42. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.5　実ロボットでの膝曲げ歩行重心補正 20 mm

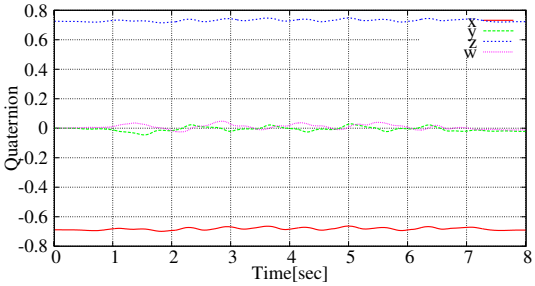


図 5.43. IMU Quaternion

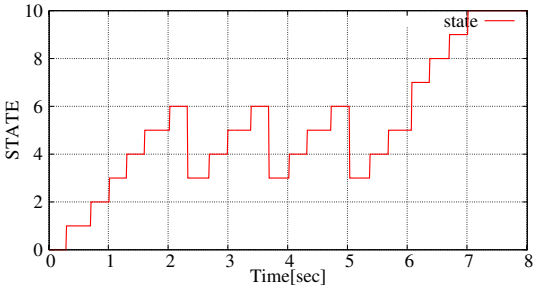
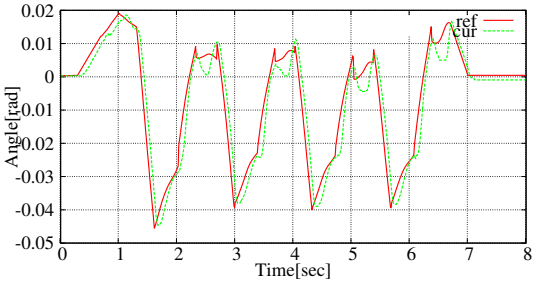
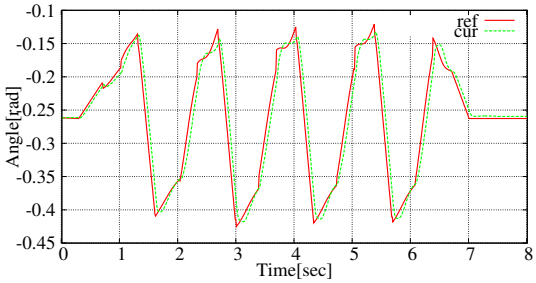


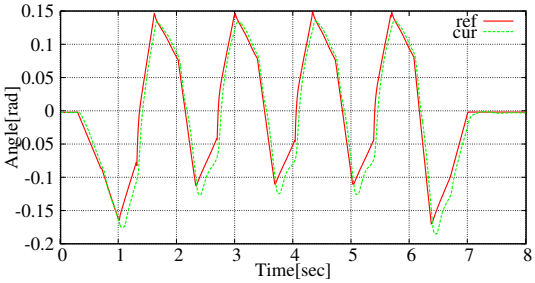
図 5.44. SIMBICON State



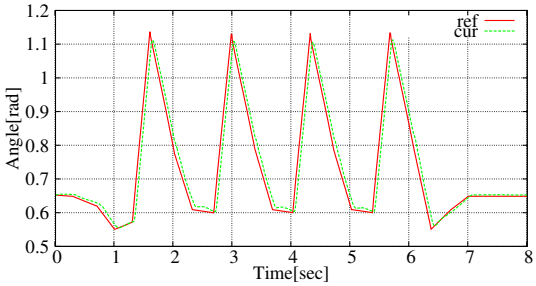
ID 1



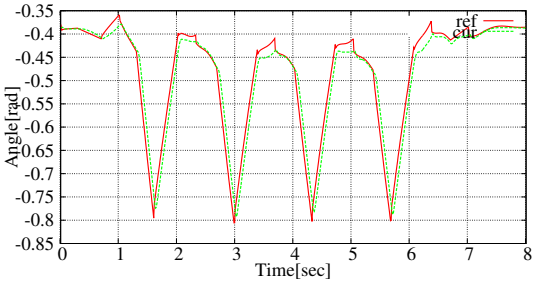
ID 2



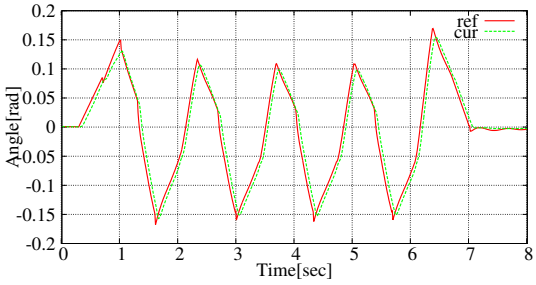
ID 3



ID 4



ID 5



ID 6

図 5.45. Joint angle

5.43. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.44. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.45. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.6 実体の身体環境モデルのパラメタ推定後のシミュレーションでの膝曲げ歩行重心補正 20 mm

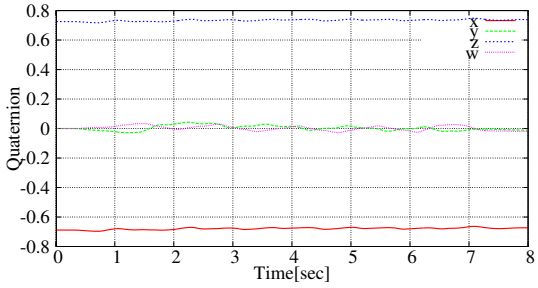


図 5.46. IMU Quaternion

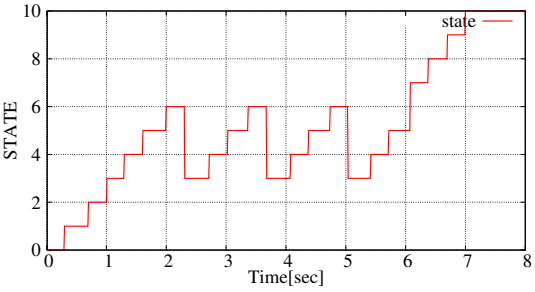
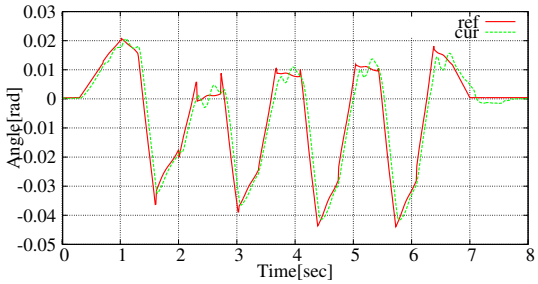
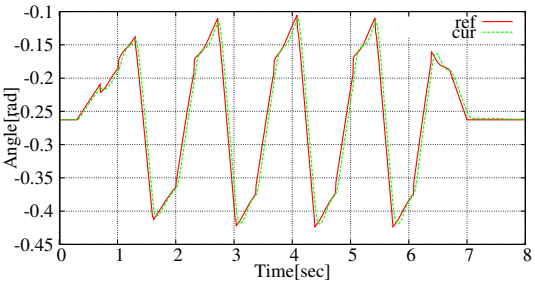


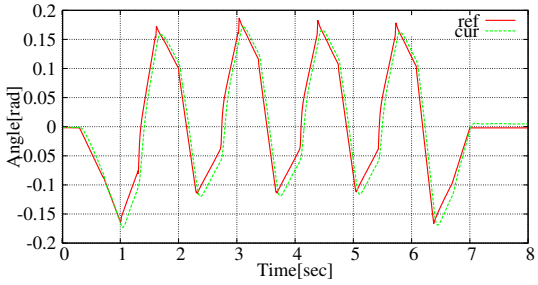
図 5.47. SIMBICON State



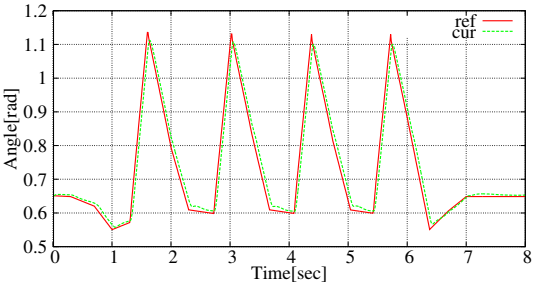
ID 1



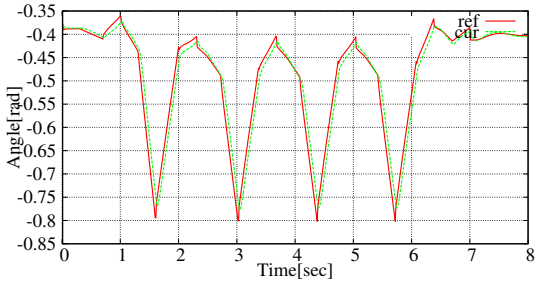
ID 2



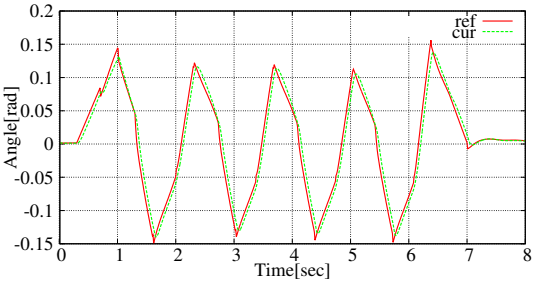
ID 3



ID 4



ID 5



ID 6

図 5.48. Joint angle

5.46. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.47. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.48. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.7　シミュレーションでの最適化された膝伸ばし歩行

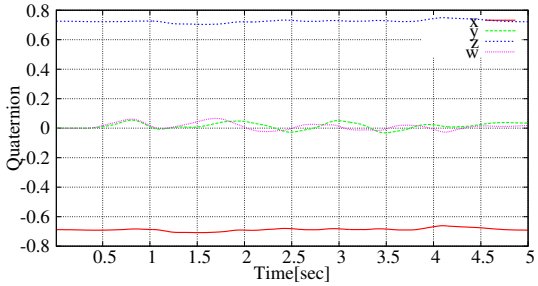


図 5.49. IMU Quaternion

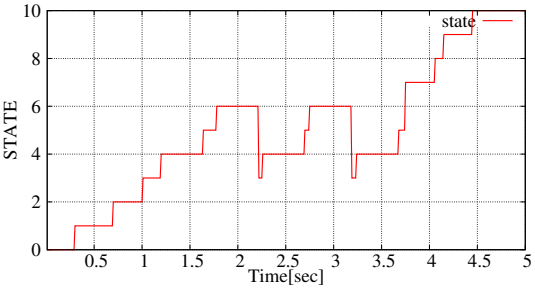
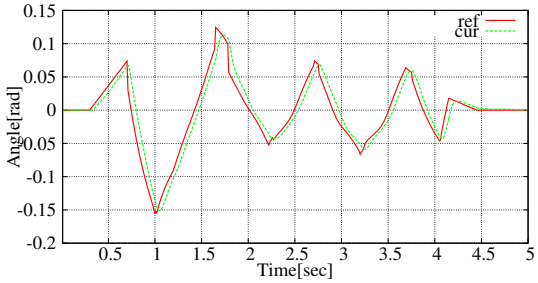
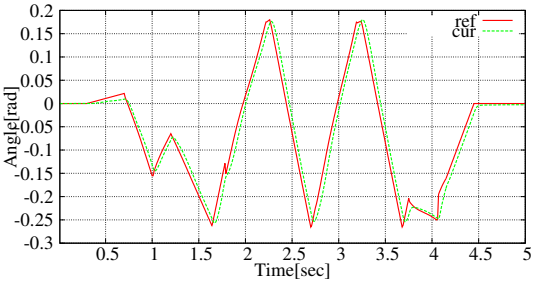


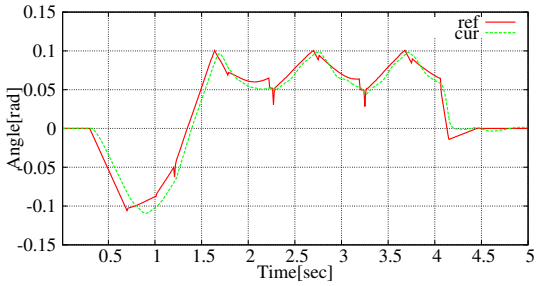
図 5.50. SIMBICON State



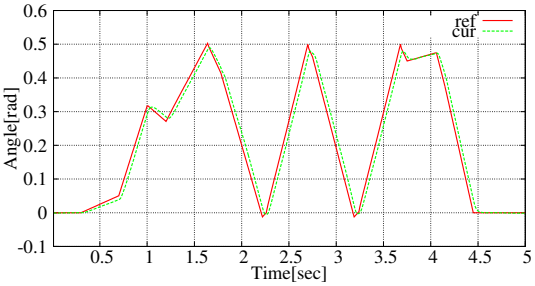
ID 1



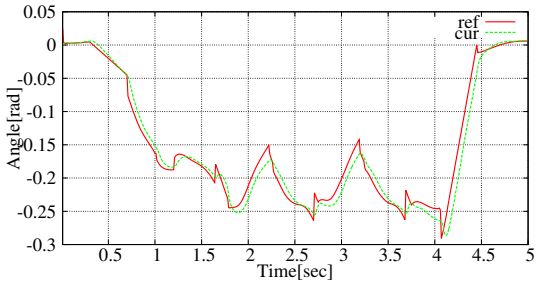
ID 2



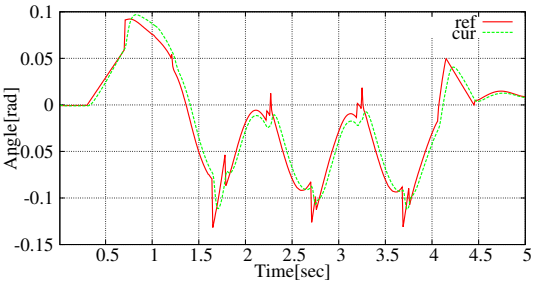
ID 3



ID 4



ID 5



ID 6

図 5.51. Joint angles

5.49. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.50. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.51. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.8 実ロボットでの最適化された膝伸ばし歩行

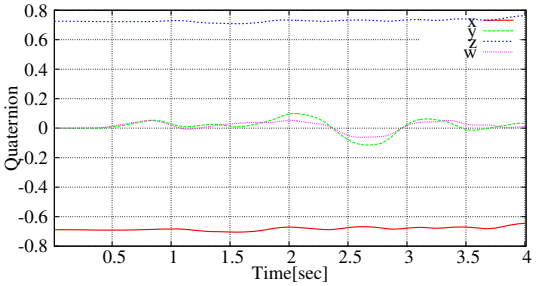


図 5.52. IMU Quaternion

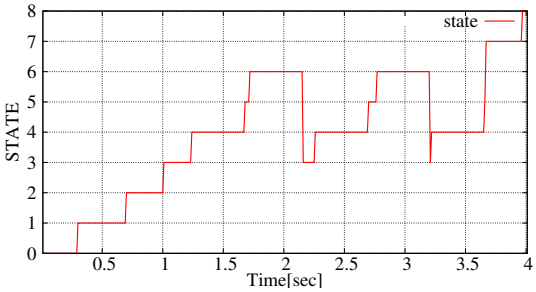
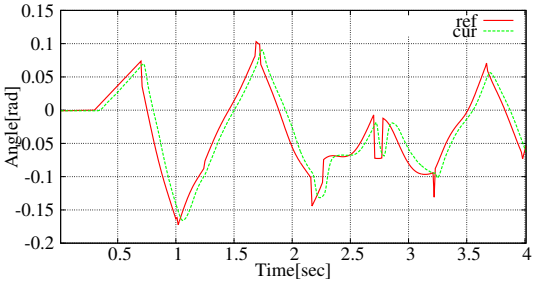
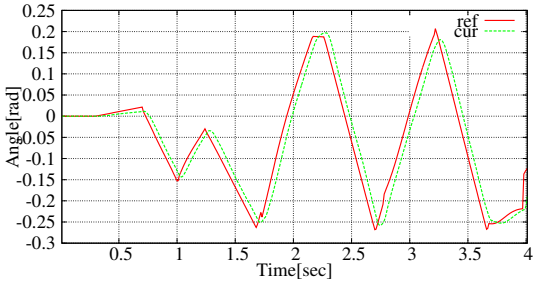


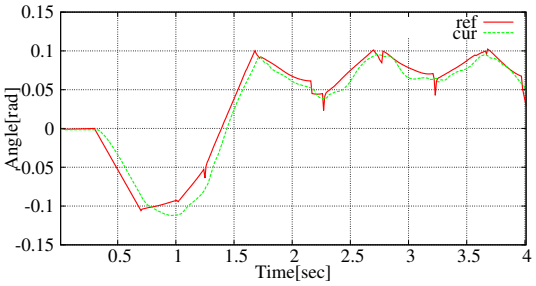
図 5.53. SIMBICON State



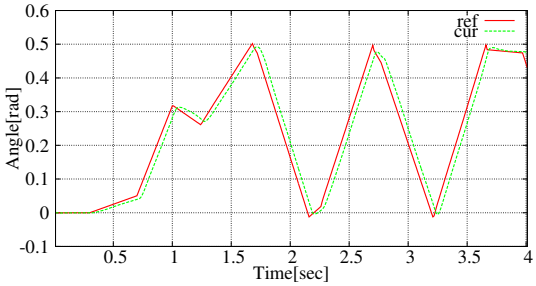
ID 1



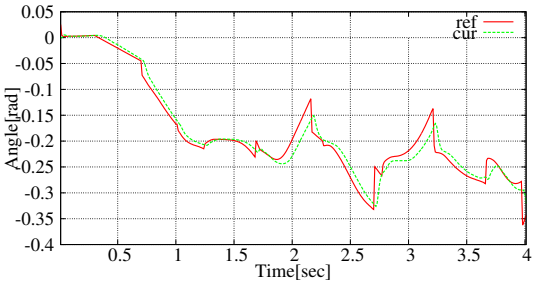
ID 2



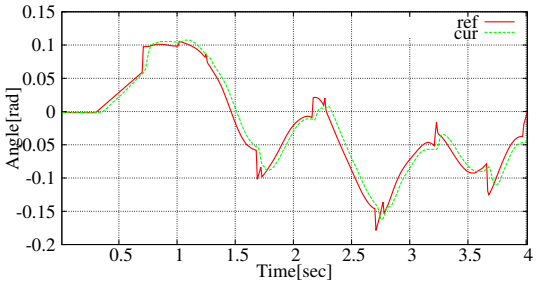
ID 3



ID 4



ID 5



ID 6

図 5.54. Joint angles

5.52. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.53. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.54. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.9　実体の身体環境モデルのパラメタ推定後のシミュレーションでの最適化された膝伸ばし歩行

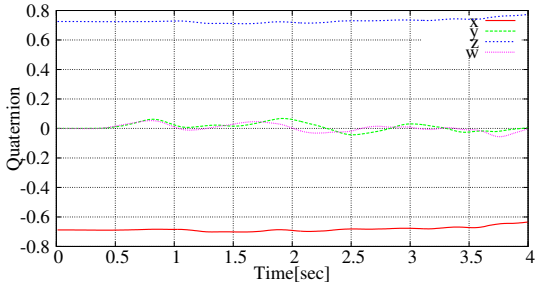


図 5.55. IMU Quaternion

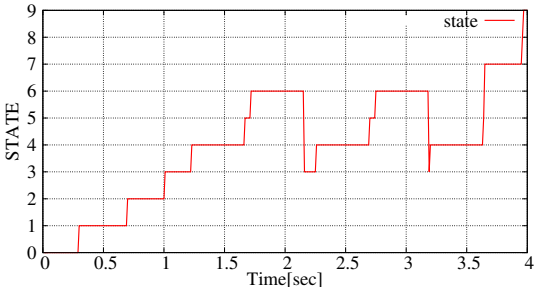
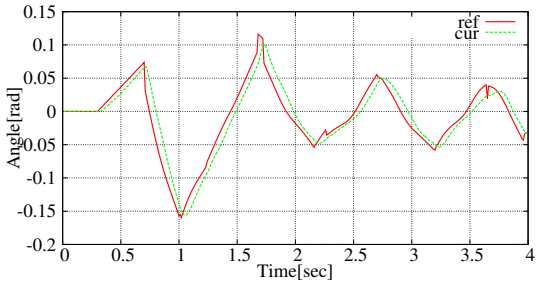
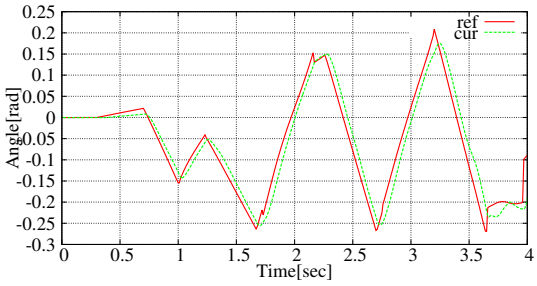


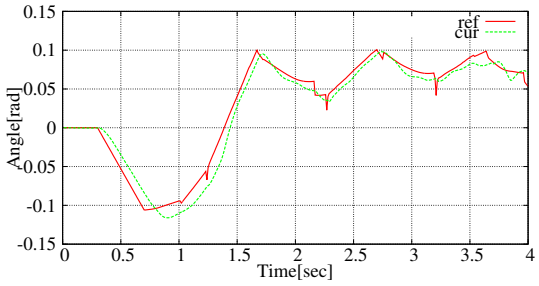
図 5.56. SIMBICON State



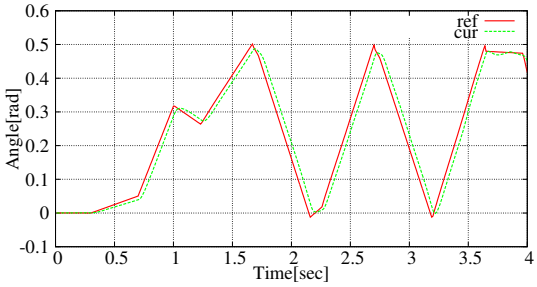
ID 1



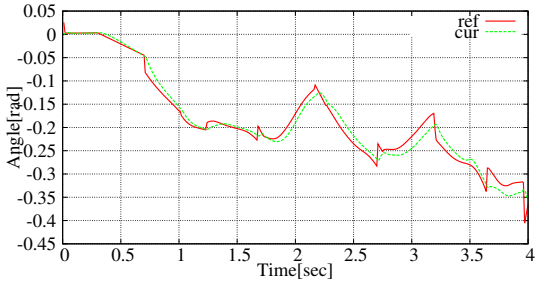
ID 2



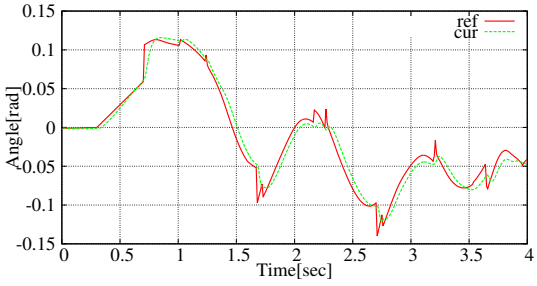
ID 3



ID 4



ID 5



ID 6

図 5.57. Joint angles

5.55. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.56. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.57. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.10 シミュレーションでの再度最適化された膝伸ばし歩行

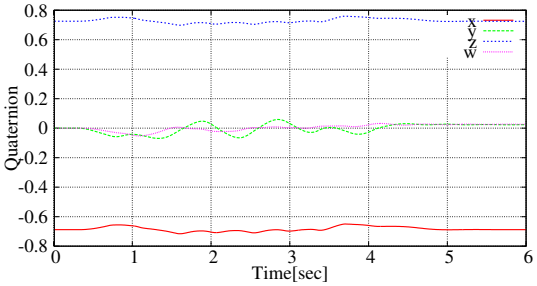


図 5.58. IMU Quaternion

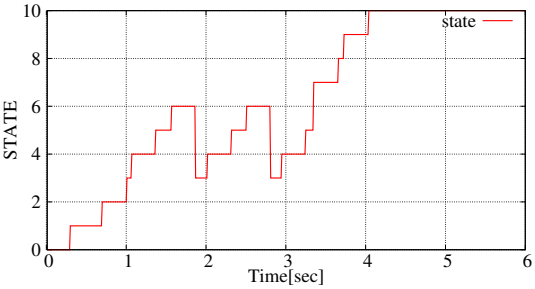
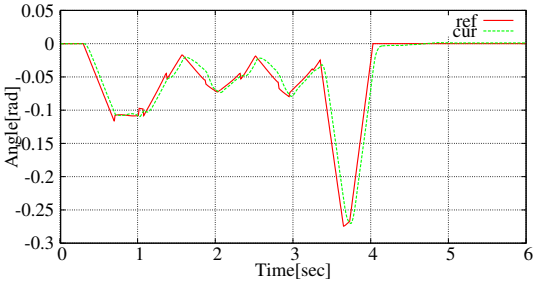
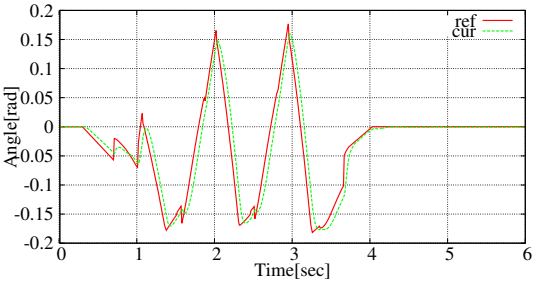


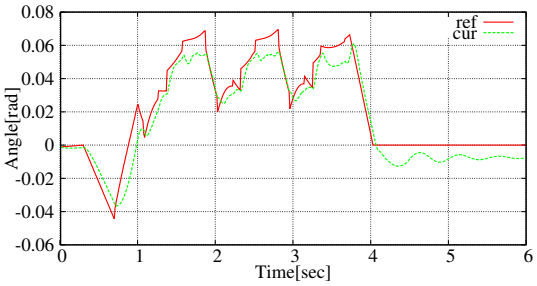
図 5.59. SIMBICON State



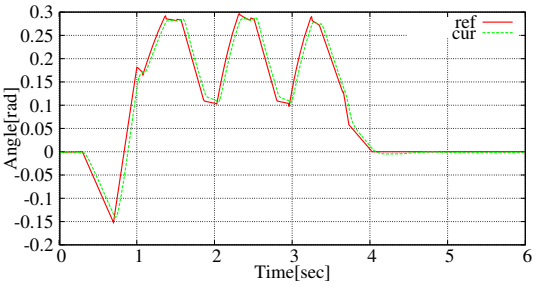
ID 1



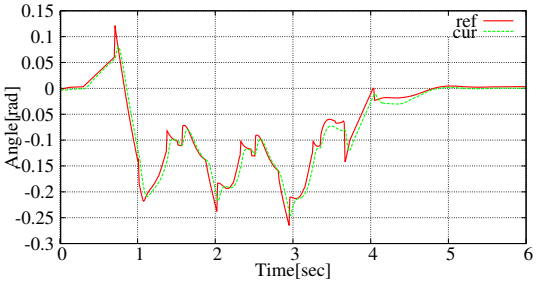
ID 2



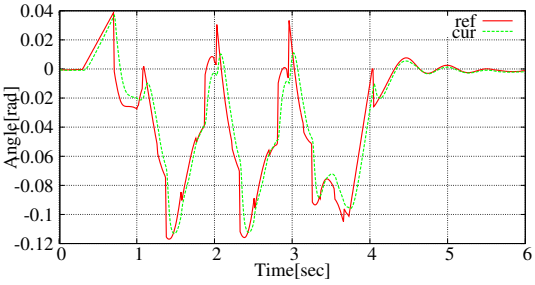
ID 3



ID 4



ID 5



ID 6

図 5.60. Joint angle

5.58. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.59. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.60. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.11　実ロボットでの再度最適化された膝伸ばし歩行

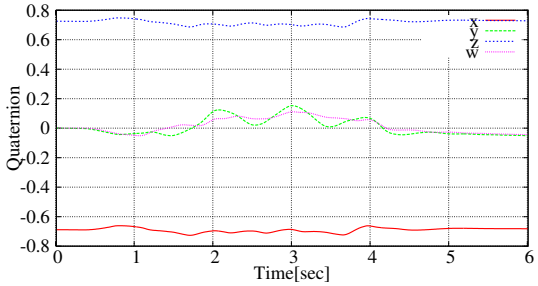


図 5.61. IMU Quaternion

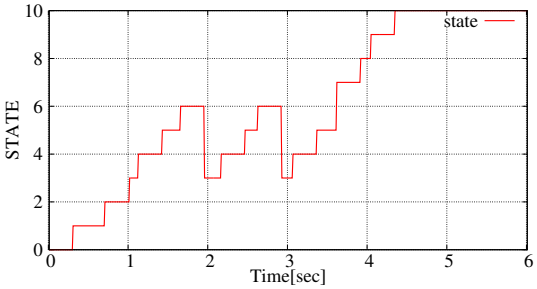
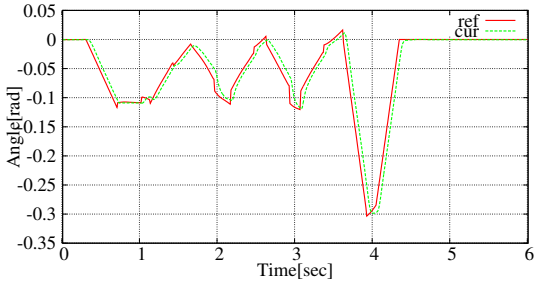
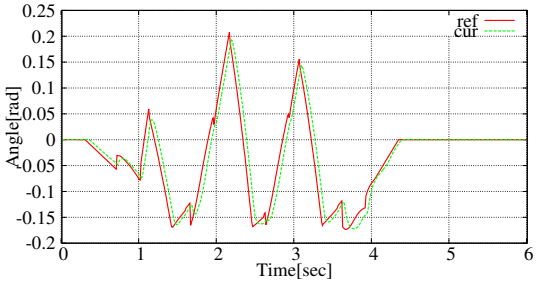


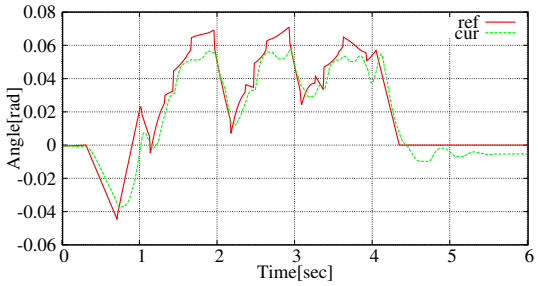
図 5.62. SIMBICON State



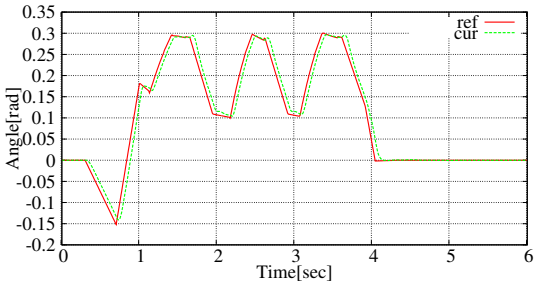
ID 1



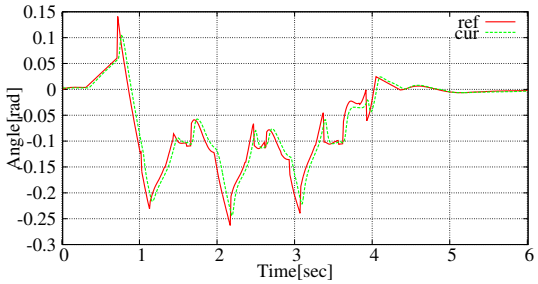
ID 2



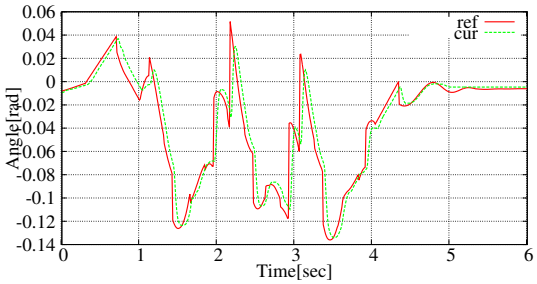
ID 3



ID 4



ID 5



ID 6

図 5.63. Joint angle

5.61. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.62. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.63. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.2.12 実体の身体環境モデルのパラメタ推定後のシミュレーションでの再度最適化された膝伸ばし歩行

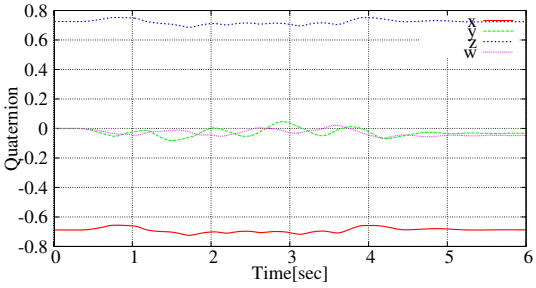


図 5.64. IMU Quaternion

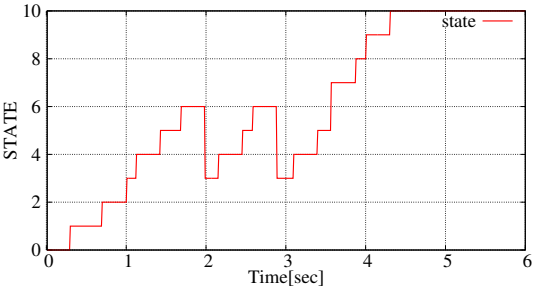
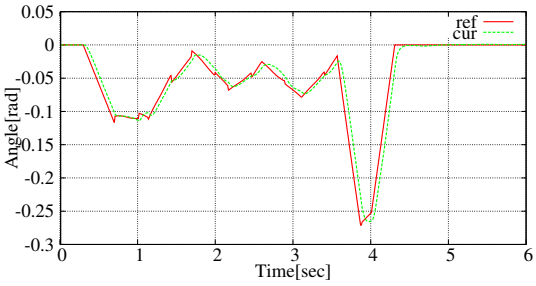
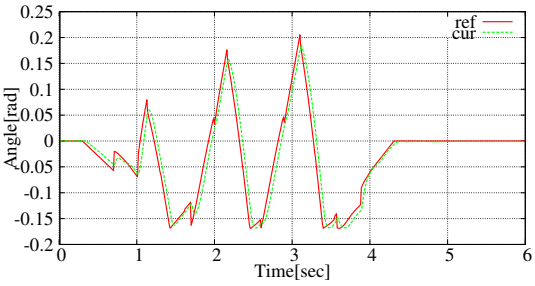


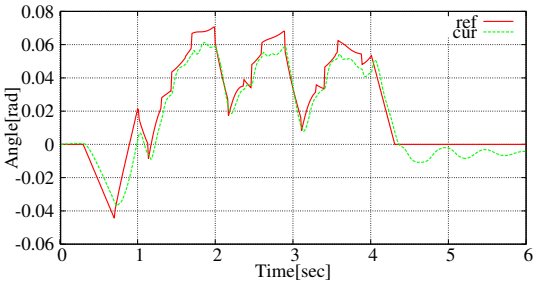
図 5.65. SIMBICON State



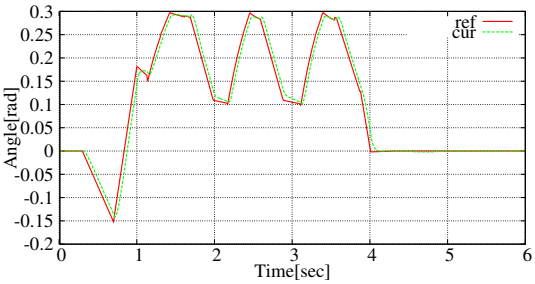
ID 1



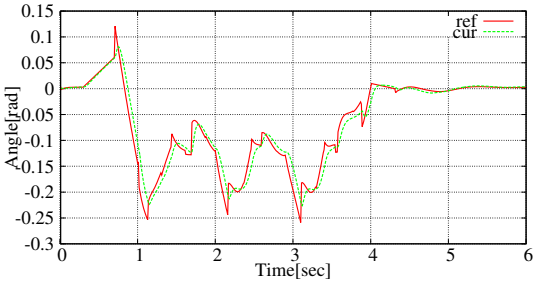
ID 2



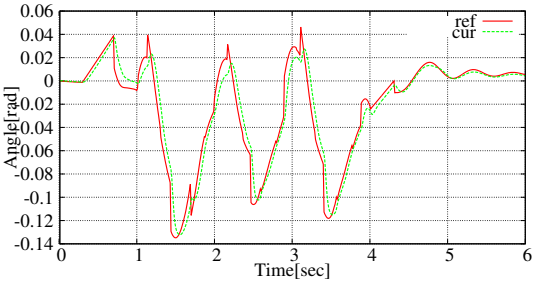
ID 3



ID 4



ID 5



ID 6

図 5.66. Joint angle

5.64. Transition of quaternion while walking. These values are calculated by using an open source implementation of [107].

5.65. State transition while walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

5.66. Joint angle targets and measured values for left six joints while stepping motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

5.5.3　実体の身体環境モデルのパラメタ推定計算時の距離関数グラフ

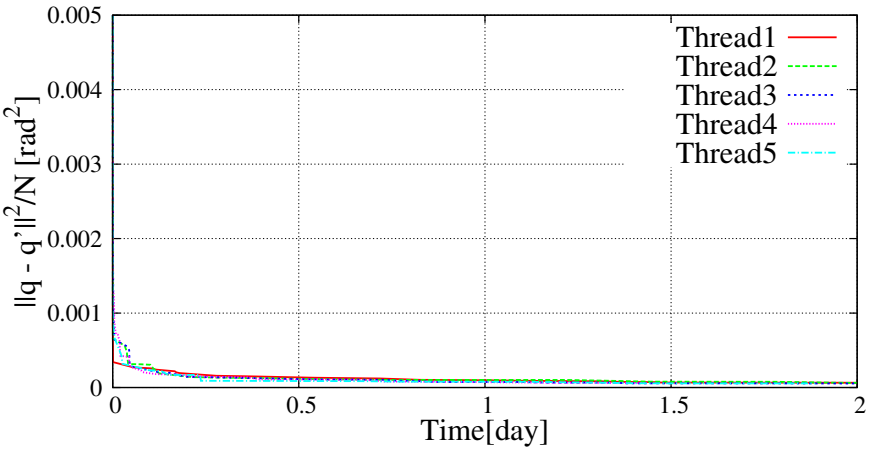


図 5.67. By minimizing the distance of log data between actual robot and its simulation, body and environment parameters are predicted. Vertical axis shows the distance and horizontal axis shows day to search. Log data contains heuristic stepping motion and walking motion.

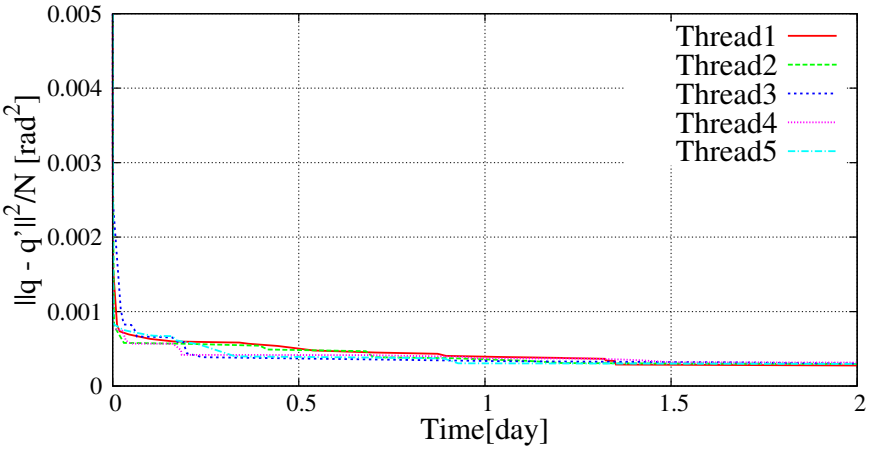


図 5.68. By minimizing the distance of log data between actual robot and its simulation, body and environment parameters are predicted. Vertical axis shows the distance and horizontal axis shows day to search. Log data contains heuristic stepping motion, walking motion, and regenerated walking motion.

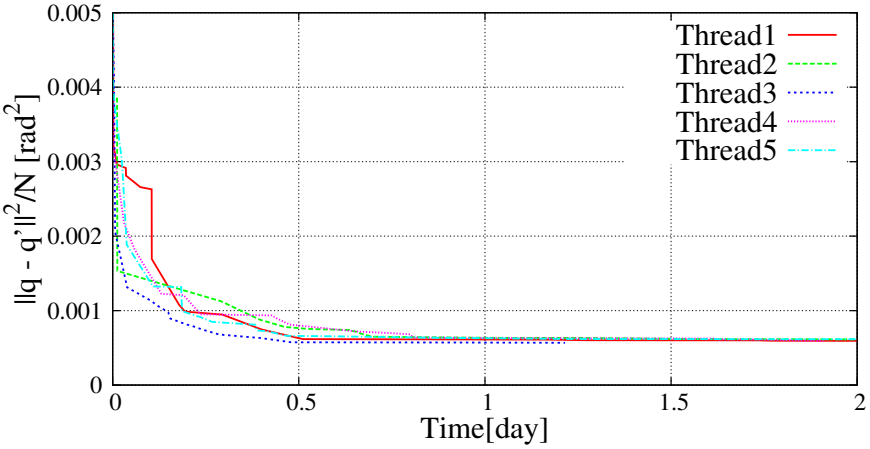


図 5.69. By minimizing the distance of log data between actual robot and its simulation, body and environment parameters are predicted. Vertical axis shows the distance and horizontal axis shows day to search. Log data contains heuristic stepping motion, walking motion, regenerated walking motion, and optimal walking motion.

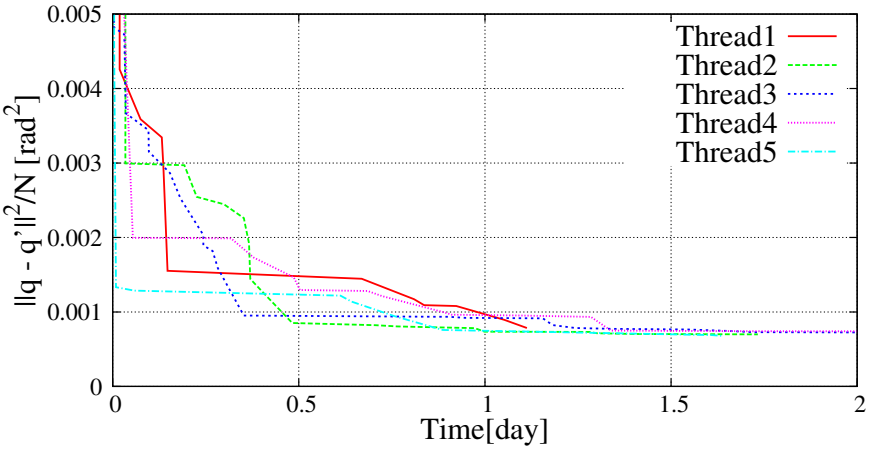


図 5.70. By minimizing the distance of log data between actual robot and its simulation, body and environment parameters are predicted. Vertical axis shows the distance and horizontal axis shows day to search. Log data contains heuristic stepping motion, walking motion, regenerated walking motion, optimal walking motion, and re-optimized walking motion.

5.5.4 推定された実体の身体環境モデルのパラメタ

Table5.5. Robot contact and link properties from CAD data

f_{thr}	k_c	d_c	k_f	d_f	mu_f
N	N/m	Ns/m	N/m	Ns/m	-
30.0	70000.0	700.0	50000.0	500.0	0.800

i	m_i	c_i	I_i	p_{ji}	R_{ji}	k_i	d_i
ID	[g]	[mm]	$[gm^2]$	[mm]	-	[Nm/rad]	[Nms/rad]
0	16917.0	−0.74 −3.12 133.10	160.033 −4.496 2.294 −4.496 162.546 −14.954 2.294 −14.954 204.884	-	-	-	-
1	2068.0	0.02 28.59 −40.19	14.871 0.003 −0.001 0.003 13.080 −5.884 −0.001 −5.884 10.564	0.00 95.00 −19.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5
2	4082.0	5.48 −10.73 −51.25	22.358 −0.239 −4.272 −0.239 21.994 2.239 −4.272 2.239 9.393	0.00 24.50 −91.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
3	2182.0	3.23 13.66 −108.15	12.184 −0.027 0.396 −0.027 12.095 −6.232 0.396 −6.232 6.049	8.50 −0.00 −139.50	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
4	3560.0	0.01 24.96 −71.06	26.143 −0.005 −0.007 −0.005 25.012 −9.904 −0.007 −9.904 7.178	−6.00 −10.50 −219.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
5	2956.0	7.55 7.55 −57.00	11.074 0.168 −3.827 0.168 11.074 −1.269 −3.827 −1.269 5.977	−0.00 −1.50 −225.50	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5
6	1616.0	10.68 0.61 −58.47	4.398 0.029 −1.252 0.029 6.324 −0.135 −1.252 −0.135 3.809	8.50 −0.00 −114.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5
7	2068.0	−0.02 −28.59 −40.19	14.871 0.003 0.001 0.003 13.080 5.884 0.001 5.884 10.564	0.00 −95.00 −19.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5
8	4082.0	5.46 10.73 −51.25	22.358 0.241 −4.262 0.241 21.995 −2.244 −4.262 −2.244 9.394	0.00 −24.50 −91.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
9	2184.0	3.23 −13.64 −108.15	12.161 0.015 0.387 0.015 12.071 6.227 0.387 6.227 6.049	8.50 −0.00 −139.50	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
10	3560.0	−0.01 −24.96 −71.06	26.143 0.007 0.007 0.007 25.012 9.904 0.007 9.904 7.178	−6.00 10.50 −219.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	7000.0	105.0
11	2956.0	7.55 −7.55 −57.00	11.074 −0.168 −3.823 −0.168 11.074 1.265 −3.823 1.265 5.977	0.00 1.50 −225.50	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5
12	1616.0	10.32 −0.26 −58.47	4.399 0.014 −1.169 0.014 6.336 0.065 −1.169 0.065 3.821	8.50 −0.00 −114.00	1.000 0.000 −0.000 0.000 1.000 0.000 0.000 0.000 1.000	3500.0	52.5

Table5.6. Robot contact and link properties identified with heuristic walk motion and its fall log

f_{thr}	k_c	d_c	k_f	d_f	mu_f
N	N/m	Ns/m	N/m	Ns/m	-
30.5	76412.0	643.6	48518.3	506.2	0.500

i	m_i	c_i	I_i	p_{ji}	R_{ji}	k_i	d_i
ID	[g]	[mm]	$[gm^2]$	[mm]	-	[Nm/rad]	[Nms/rad]
0	15384.9	9.60 1.85 63.79	150.747 4.048 7.524 4.048 161.203 −22.408 7.524 −22.408 196.809	-	-	-	-
1	1901.49	6.88 65.33 −38.39	16.614 −8.283 9.023 −8.283 4.549 4.078 9.023 4.078 9.512	−0.47 95.42 −19.14	1.000 −0.002 0.002 0.002 1.000 0.001 −0.002 −0.001 1.000	3260.6	53.7
2	4237.33	−79.49 11.73 42.54	28.644 2.244 −2.240 2.244 22.352 −1.795 −2.240 −1.795 15.945	−0.72 23.74 −90.18	1.000 −0.000 0.000 0.000 1.000 0.001 −0.000 −0.001 1.000	6648.4	107.8
3	1715.55	−8.50 −20.76 −115.77	10.692 3.125 7.681 3.125 11.231 −15.767 7.681 −15.767 7.026	8.76 0.06 −139.66	1.000 −0.000 −0.001 0.000 1.000 0.000 0.001 −0.000 1.000	6963.9	103.0
4	3052.79	43.31 −39.24 −107.86	24.418 −6.729 1.435 −6.729 16.563 −18.099 1.435 −18.099 13.003	−6.35 −10.52 −218.55	1.000 0.002 0.001 −0.002 1.000 0.001 −0.001 −0.001 1.000	6865.9	114.9
5	2217.31	79.01 36.60 27.38	20.031 3.068 5.934 3.068 11.752 0.222 5.934 0.222 0.790	−0.98 −2.36 −224.57	1.000 0.000 0.000 −0.000 1.000 0.001 −0.000 −0.001 1.000	3829.4	52.3
6	1047.98	19.04 −3.93 −5.62	9.580 1.964 3.419 1.964 11.536 6.998 3.419 6.998 −4.969	9.08 −0.27 −113.58	1.000 0.002 −0.002 −0.002 1.000 0.001 0.002 −0.001 1.000	3792.6	50.0
7	1968.51	6.33 −72.86 −36.25	16.610 8.276 9.017 8.276 4.543 −4.078 9.017 −4.078 9.510	−0.65 −95.37 −19.06	1.000 0.002 0.001 −0.002 1.000 −0.002 −0.001 0.002 1.000	3290.6	54.1
8	4318.98	−69.58 −8.42 42.11	28.641 −2.247 −2.238 −2.247 22.347 1.795 −2.238 1.795 15.942	−0.59 −24.21 −89.79	1.000 0.001 0.000 −0.001 1.000 −0.001 −0.000 0.001 1.000	6688.8	107.4
9	1752.65	−13.43 13.90 −108.90	10.689 −3.118 7.686 −3.118 11.241 15.777 7.686 15.777 7.024	9.01 0.38 −139.43	1.000 0.000 −0.001 −0.000 1.000 0.000 0.001 −0.000 1.000	6984.8	102.2
10	3002.01	36.32 47.57 −100.97	24.414 6.722 1.431 6.722 16.560 18.096 1.431 18.096 12.994	−5.90 10.46 −218.89	1.000 −0.001 0.002 0.001 1.000 −0.001 −0.002 0.001 1.000	6805.8	114.8
11	2123.57	79.74 −30.52 19.92	20.028 −3.065 5.929 −3.065 11.759 −0.232 5.929 −0.232 0.796	−0.76 2.06 −224.18	1.000 −0.001 0.001 0.001 1.000 −0.001 −0.001 0.001 1.000	3850.0	52.4
12	1105.06	16.38 −2.72 0.41	9.579 −1.972 3.421 −1.972 11.528 −6.997 3.421 −6.997 −4.964	8.92 −0.00 −113.32	1.000 −0.001 −0.002 0.001 1.000 −0.001 0.002 0.001 1.000	3790.6	50.0

Table5.7. Robot contact and link properties identified with heuristic walking motion, its fall log, regenerated motion, and its success log

f_{thr}	k_c	d_c	k_f	d_f	mu_f
N	N/m	Ns/m	N/m	Ns/m	-
18.6	75326.0	699.1	47245.7	468.5	0.545

i	m_i	c_i	I_i	p_{ji}	R_{ji}	k_i	d_i
ID	[g]	[mm]	[gm^2]	[mm]	-	[Nm/rad]	[Nms/rad]
0	14735.2	0.61 2.94 188.60	167.340 −2.210 4.161 −2.210 160.836 −16.852 4.161 −16.852 200.502	-	-	-	-
1	2612.33	−9.80 −32.49 −2.32	19.476 1.153 −5.267 1.153 19.645 0.333 −5.267 0.333 19.150	−0.21 95.77 −19.11	1.000 0.001 −0.001 −0.001 1.000 −0.001 0.001 0.001 1.000	3441.9	47.8
2	4920.91	−62.21 2.46 38.83	30.499 1.853 −1.089 1.853 30.817 −2.155 −1.089 −2.155 17.354	0.07 25.05 −91.37	1.000 0.000 0.001 −0.000 1.000 −0.001 −0.001 0.001 1.000	6983.2	98.6
3	2363.17	22.14 −64.64 −182.73	12.155 −6.145 0.883 −6.145 21.388 −0.243 0.883 −0.243 0.583	7.70 0.31 −140.22	1.000 −0.001 −0.001 0.001 1.000 −0.000 0.001 0.000 1.000	6503.0	110.8
4	3209.39	−30.82 −45.33 −109.35	32.743 −8.179 5.199 −8.179 26.936 −1.618 5.199 −1.618 11.485	−6.23 −11.09 −219.94	1.000 0.000 −0.001 −0.000 1.000 −0.001 0.001 0.001 1.000	6380.2	100.0
5	2969.69	3.38 −39.55 −38.10	7.560 −3.784 −4.622 −3.784 17.675 −6.755 −4.622 −6.755 11.204	−0.07 −1.41 −225.11	1.000 −0.000 −0.002 0.000 1.000 −0.002 0.002 0.002 1.000	3672.3	48.9
6	829.167	83.04 −55.83 −9.91	4.365 2.034 6.682 2.034 12.664 −7.273 6.682 −7.273 1.751	9.37 0.20 −114.29	1.000 0.001 −0.001 −0.001 1.000 0.000 0.001 −0.000 1.000	3747.3	54.9
7	2526.64	−19.59 35.56 6.40	19.472 −1.145 −5.274 −1.145 19.646 −0.341 −5.274 −0.341 19.156	0.05 −96.04 −18.92	1.000 −0.002 −0.000 0.002 1.000 0.001 0.000 −0.001 1.000	3419.1	47.6
8	4927.24	−61.82 −0.40 30.18	30.500 −1.855 −1.089 −1.855 30.810 2.151 −1.089 2.151 17.347	−0.00 −25.37 −91.36	1.000 0.000 0.001 −0.000 1.000 0.001 −0.001 −0.001 1.000	7011.0	99.5
9	2355.74	25.61 61.16 −185.32	12.146 6.139 0.874 6.139 21.384 0.247 0.874 0.247 0.586	7.53 −0.59 −140.45	1.000 0.001 −0.002 −0.001 1.000 0.001 0.002 −0.001 1.000	6538.1	111.0
10	3243.67	−31.72 45.65 −118.40	32.736 8.174 5.194 8.174 26.926 1.608 5.194 1.608 11.479	−6.54 11.36 −219.90	1.000 −0.001 −0.001 0.001 1.000 0.002 0.001 −0.002 1.000	6323.8	100.6
11	3016.26	−6.25 33.30 −39.00	7.555 3.783 −4.629 3.783 17.668 6.761 −4.629 6.761 11.201	0.21 1.54 −225.35	1.000 0.000 −0.002 −0.000 1.000 0.002 0.002 −0.002 1.000	3699.4	49.0
12	762.432	90.19 55.50 −14.80	4.361 −2.032 6.691 −2.032 12.669 7.274 6.691 7.274 1.760	9.27 −0.17 −114.62	1.000 −0.001 −0.002 0.001 1.000 −0.000 0.002 0.000 1.000	3737.9	55.2

Table5.8. Robot contact and link properties identified with optimal walking motion and its fall log

f_{thr}	k_c	d_c	k_f	d_f	mu_f
N	N/m	Ns/m	N/m	Ns/m	-
31.4	67047.4	676.2	51168.0	496.4	0.752

i	m_i	c_i	I_i	p_{ji}	R_{ji}	k_i	d_i
ID	[g]	[mm]	[gm^2]	[mm]	-	[Nm/rad]	[Nms/rad]
0	19428.4	−8.47 1.38 105.85	152.632 −5.061 −4.701 −5.061 158.807 −11.761 −4.701 −11.761 212.788	-	-	-	-
1	2469.0	52.75 125.66 −74.97	6.164 −1.284 −6.908 −1.284 17.182 1.689 −6.908 1.689 3.181	−0.97 94.91 −18.62	1.000 0.000 0.000 −0.000 1.000 −0.002 −0.000 0.002 1.000	3357.7	48.5
2	4526.94	−49.73 34.36 34.01	23.332 1.968 −0.426 1.968 15.752 1.032 −0.426 1.032 14.740	0.20 24.54 −90.90	1.000 0.000 −0.001 −0.000 1.000 0.001 0.001 −0.001 1.000	6361.2	102.9
3	1406.59	31.80 −63.51 −57.86	9.642 −8.219 −6.729 −8.219 3.467 −14.163 −6.729 −14.163 7.154	7.75 −0.65 −138.75	1.000 0.002 0.000 −0.002 1.000 −0.001 −0.000 0.001 1.000	7671.4	105.5
4	4302.44	45.70 −58.74 −133.22	30.992 9.725 −0.460 9.725 27.265 −1.394 −0.460 −1.394 16.031	−6.54 −10.48 −218.71	1.000 0.000 −0.001 −0.000 1.000 −0.001 0.001 0.001 1.000	6602.3	108.6
5	2415.86	29.06 84.23 −143.18	2.479 −8.436 3.209 −8.436 12.351 0.837 3.209 0.837 13.719	−0.33 −2.41 −225.08	1.000 −0.001 −0.000 0.001 1.000 0.002 0.000 −0.002 1.000	3156.8	53.8
6	2218.68	−73.33 81.61 29.85	8.822 1.777 6.995 1.777 5.646 −1.979 6.995 −1.979 11.347	8.04 −0.01 −114.88	1.000 −0.001 −0.001 0.001 1.000 0.001 0.001 −0.001 1.000	3751.2	48.9
7	2503.82	60.18 −119.72 −75.85	6.164 1.277 −6.903 1.277 17.191 −1.680 −6.903 −1.680 3.180	−0.62 −95.10 −18.90	1.000 −0.001 −0.000 0.001 1.000 0.002 0.000 −0.002 1.000	3369.6	48.8
8	4598.33	−44.91 −32.80 41.11	23.339 −1.970 −0.420 −1.970 15.749 −1.042 −0.420 −1.042 14.735	0.62 −24.94 −90.50	1.000 −0.001 0.000 0.001 1.000 −0.000 −0.000 0.000 1.000	6366.6	103.5
9	1496.84	35.49 65.01 −63.98	9.636 8.226 −6.728 8.226 3.471 14.156 −6.728 14.156 7.151	8.18 0.62 −139.15	1.000 −0.002 0.000 0.002 1.000 0.001 −0.000 −0.001 1.000	7639.9	105.5
10	4313.65	41.04 51.56 −127.88	30.986 −9.715 −0.453 −9.715 27.271 1.400 −0.453 1.400 16.039	−6.68 10.42 −218.38	1.000 −0.000 −0.000 0.000 1.000 −0.000 0.000 0.000 1.000	6555.7	108.8
11	2423.19	33.75 −86.88 −146.35	2.469 8.429 3.219 8.429 12.348 −0.835 3.219 −0.835 13.716	−0.37 2.12 −224.87	1.000 0.001 −0.001 −0.001 1.000 −0.002 0.001 0.002 1.000	3148.0	53.8
12	2126.95	−69.05 −81.84 24.41	8.822 −1.769 6.998 −1.769 5.652 1.971 6.998 1.971 11.347	8.40 −0.34 −114.52	1.000 0.000 −0.000 −0.000 1.000 −0.001 0.000 0.001 1.000	3719.9	49.0

Table5.9. Robot contact and link properties identified with optimal walking motion, its fall log, re-optimized walking motion, and its success log

f_{thr}	k_c	d_c	k_f	d_f	mu_f
N	N/m	Ns/m	N/m	Ns/m	-
40.9	64046.4	717.0	53422.0	495.3	0.602

i	m_i	c_i	I_i	p_{ji}	R_{ji}	k_i	d_i
ID	[g]	[mm]	$[gm^2]$	[mm]	-	[Nm/rad]	[Nms/rad]
0	18283.2	20.40 −8.31 95.96	150.868 −2.047 −3.154 −2.047 164.750 −8.432 −3.154 −8.432 198.556	-	-	-	-
1	2782.74	−91.74 65.14 −119.53	15.434 −9.836 −2.144 −9.836 8.944 0.322 −2.144 0.322 2.907	−0.50 94.30 −18.59	1.000 0.000 0.001 −0.000 1.000 0.000 −0.001 −0.000 1.000	3409.5	54.7
2	4492.07	24.72 32.04 29.17	14.307 −3.260 −0.278 −3.260 25.725 1.448 −0.278 1.448 14.071	0.72 24.16 −91.21	1.000 −0.001 −0.001 0.001 1.000 0.001 0.001 −0.001 1.000	7021.2	110.7
3	1389.22	−53.15 11.61 −28.24	21.737 3.126 8.126 3.126 4.547 0.640 8.126 0.640 14.408	9.31 −0.61 −139.57	1.000 0.000 −0.001 −0.000 1.000 −0.002 0.001 0.002 1.000	7018.3	107.8
4	3092.79	2.50 −0.39 −110.69	30.095 6.130 −8.879 6.130 22.019 −10.087 −8.879 −10.087 9.227	−5.11 −9.68 −218.24	1.000 −0.001 −0.001 0.001 1.000 −0.000 0.001 0.000 1.000	7562.2	112.4
5	2492.97	23.85 14.45 −137.53	14.314 6.414 −4.731 6.414 10.448 3.827 −4.731 3.827 4.370	0.70 −0.78 −225.87	1.000 −0.001 −0.001 0.001 1.000 0.000 0.001 −0.000 1.000	3153.7	57.0
6	1640.6	41.49 11.67 −99.28	10.503 −1.440 2.656 −1.440 8.211 0.543 2.656 0.543 −2.379	7.67 0.73 −114.16	1.000 −0.000 0.001 0.000 1.000 −0.002 −0.001 0.002 1.000	3481.8	53.0
7	2746.34	−92.69 −69.45 −119.04	15.433 9.827 −2.149 9.827 8.940 −0.322 −2.149 −0.322 2.900	−0.35 −94.37 −18.82	1.000 −0.001 0.001 0.001 1.000 −0.000 −0.001 0.000 1.000	3421.4	54.2
8	4395.53	21.54 −41.61 33.94	14.305 3.265 −0.269 3.265 25.722 −1.444 −0.269 −1.444 14.072	0.92 −24.50 −91.08	1.000 0.001 −0.001 −0.001 1.000 −0.001 0.001 0.001 1.000	7045.6	110.6
9	1373.24	−60.37 −20.27 −24.93	21.736 −3.117 8.133 −3.117 4.545 −0.636 8.133 −0.636 14.399	9.73 0.64 −139.54	1.000 −0.000 −0.001 0.000 1.000 0.002 0.001 −0.002 1.000	7064.9	107.6
10	3125.59	−4.89 −0.36 −115.79	30.093 −6.132 −8.878 −6.132 22.025 10.083 −8.878 10.083 9.233	−5.14 9.69 −218.38	1.000 −0.000 −0.002 0.000 1.000 0.001 0.002 −0.001 1.000	7535.4	112.3
11	2444.48	26.92 −13.14 −137.97	14.323 −6.415 −4.723 −6.415 10.441 −3.820 −4.723 −3.820 4.372	0.29 0.46 −225.59	1.000 0.000 −0.002 −0.000 1.000 0.000 0.002 −0.000 1.000	3146.5	56.6
12	1590.89	37.82 −3.79 −100.17	10.507 1.450 2.660 1.450 8.206 −0.539 2.660 −0.539 −2.375	7.44 −0.56 −114.65	1.000 0.000 0.001 −0.000 1.000 0.002 −0.001 −0.002 1.000	3462.7	52.8

第 6 章

身体運動モデルのパラメタ同時探索による準受動歩行支持バネ係数決定と行動生成制御

6.1 はじめに

身体環境モデル空間と運動モデル空間の同時探索手法について扱い，例題として準受動歩行動作に用いる体幹支持バネ係数の決定と，バネを用いた準受動歩行行動生成制御パラメタの同時探索実験を行う．図 2.10 が解く問題のモデルと探索法をまとめたものである．支援システムにおけるロボット完成前の身体運動モデルパラメタ生成において二つの空間を同時に探索する例と，ロボット完成後の実体モデルパラメタ修正において推定する実体身体モデルのパラメタがセンサにより測定できない場合の例を示す．本章では準受動歩行に必要なバネ係数と歩行運動モデルパラメタの探索支援を行った．この問題の意義は動的で不安定な運動，設計通りに実体化が難しいバネという非剛体の実体身体モデルのパラメタ推定にも本支援システムが有効であることを示したことである．受動軸として股関節を用いる準受動歩行は歩き始めに膝から下を使って上半身の姿勢を一度崩し，バネの力を蓄えてから歩き始めるという歩行開始時，歩行時，歩行終了時でことなる体の使い方が必要である．さらにバネの力で振動する上半身は直接制御できず動力学に従い運動するため，正確な上半身の制御には全身の動力学モデル，接触モデルのパラメタ推定が必要な例となっている．

以下では，身体モデル空間に歩行のための機構を有する脚型ロボットのバネを用いたクラッチ機構について述べる．またバネとクラッチを用いることで受動歩行行動が生成できることを確認し，身体モデル空間のバネ係数を歩行行動制御と同時に最適化することで省エネルギー歩行行動を行う．さらに環境モデル空間に含まれるような地面の傾きや柔らかさ，未知の外力といったものに対しても探索を行うことでロバストな行動生成ができることをみる．

6.2　省エネルギー歩行のための拘束解放可変機構

6.2.1　受動軸の有無からみた動歩行研究と受動歩行

アクチュエータを一切，あるいはほとんど用いずに，機構の物理特性と位置エネルギーといったロボットのアクチュエータ自身が発揮しないエネルギーを用いて歩行するロボット（受動歩行器）については，これまで多くの研究がなされてきている．受動歩行は，ロボット自身のエネルギを用いないため省エネルギーであり，受動関節は予期せぬ外力に対して不要な内力を発揮しないため，歩行動作を安全に行う上で望ましい性質を持つと言える．また BlueBiped [122] のような美しい受動歩行は見るものを魅了する力を持っている．

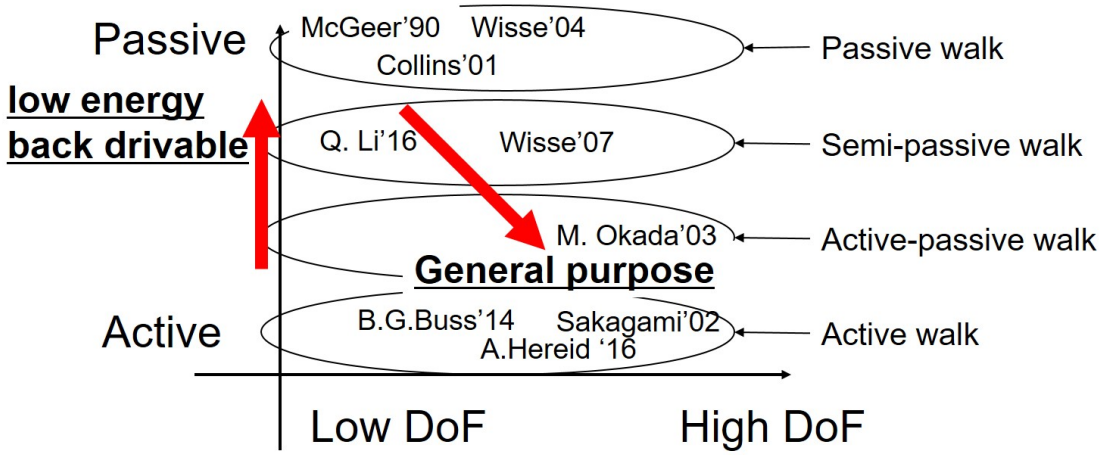


図 6.1. Several researches on walking motion are placed in two directions: passive or active, and low or high degree of freedom. Passive motion is low-energy and back-drivable but not general-purpose, and vice versa.

本章で目指すものを確認するため，歩行研究を受動軸の有無と関節自由度という点から考察する（図 6.1）．世界初の受動歩行ロボットは 1990 年に McGeer [123] が開発したものと言われる．McGeer の受動歩行器は股関節と膝関節からなる三本の足を持ち，外側二本の足を同期させることで横倒れを考える必要なく二次元のモデルで安定して歩行することができた．その約十年後の 2001 年に，Collins ら [124] は二本足の三次元モデルで受動歩行することに成功した．ここまでの受動歩行ロボットは股関節より上のリンクの重心が股関節よりも下にあったために上半身が回転して転倒する心配のないものであったが，2004 年に Wisse ら [106] が上半身を持つロボットでも二本足で受動歩行を可能にするモデルを考案した．Wisse らのモデルでは両脚の中心に上半身が常に位置するようなリンクを組むことで安定なリミットサイクルを探索できたという内容であった．また上半身を両足中央に厳密に配置せずともバネ等を用いて中央付近で振動させることでも歩行できる可能性が示唆された．以上は図 6.1 一番上の楕円に示す全関節が受動軸であるような受動歩行器（Passive walk）に関する研究であるが，同時期

に全関節が駆動軸（図 6.1 一番下の楕円，Active walk）であるような等身大ヒューマノイドロボット ASIMO [125] が発表されていることはあまりにも有名である．それ以前にも多数の全軸駆動型ロボットが存在し，最近でも MARLO [105] や DURUS [100] といった新しいロボットが次々と開発され，屋外不整地での安定歩行や省エネルギー歩行を競い合っている．

全軸受動ロボットと全軸能動ロボットを比較してみると，全軸受動ロボットではロボット内部から供給するエネルギーがなくとも歩くという強みがある一方で，歩行以外への応用はほとんど明らかにされてこなかった．対して全軸能動ロボットでは歩行に限らず多くの応用が示されてきた一方で省エネルギーの問題は以前として残っている．そこで省エネルギー性と汎用性を兼ね備えるロボットを作ろうという試みが図 6.1 上から二番目の楕円に示す準受動歩行（Semi-passive walk）である．準受動歩行は関節の一部が受動軸であり一部が能動軸であるようなロボットで，最近だと [126]，Wisse らの研究 [127] では等身大サイズのロボットでも実現されてきている．

さらに準受動歩行では受動軸は受動軸のまま能動軸は能動軸のままであり，互いに入れ替わることはないが，クラッチ機構を用いることで切り替えが可能である．backlash clutch [128] と呼ばれる遊びのあるかみ合わせ機構を用いることで受動軸と能動軸を切り替えながら歩行を行うというアイデアが示されている．その他にも直動軸を用いて受動リンクの可動域を制御することで受動軸を作り出すロボット [129] や，制御により擬似的に受動軸のような振る舞いを作り出す研究 [130] もある．図 6.1 下から二番目の楕円に示す領域を能動受動歩行（Active-passive walk）と本研究では呼ぶことにする．

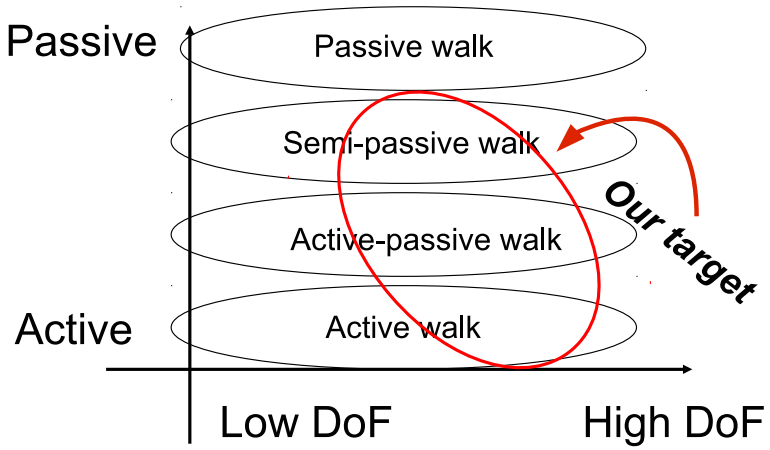


図 6.2. Low-energy-cost and general-purpose robot use both of active walk and passive walk.

汎用性と省エネルギー性を併せ持つロボットを開発したいという視点で図 6.1 を俯瞰してみると，従来研究では，一部に受動軸を用いることで省エネルギー化を試みる研究はあるものの代わりに汎用性を犠牲にしてしまっていると考える．汎用性と省エネルギー性を併せ持つロボットに必要なのは，受動歩行を実現しつつ全軸を能動的に駆動できるような機構であり，図

6.2 に示すように Active walk から (semi-)passive walk までを一台で実現できるロボットであり，これが本章で目指すものである．

6.2.2　受動歩行を能動歩行ロボットへ応用するための課題とクラッチ機構

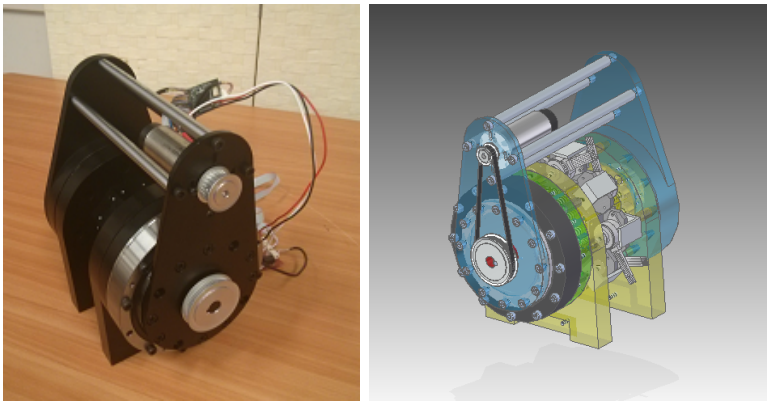


図 6.3. We design and develop a enough small brack/clutch mechanism mounted in robot's joints.

関節の能動・受動を切り替える機構としてはクラッチ機構 [131] が有名であるが，等身大ロボットの自重支えるための出力を実現するクラッチ機構は大型大重量化，大消費電力化してしまうという問題がある．例えば，クラッチ機構のなかでも比較的小型大出力として知られるツースクラッチ [131] を例に取ると，最新の等身大ヒューマノイドロボット JAXON [1] が発揮可能な出力 300Nm を実現する機構は，直径 134mm × 83mm 程度の樽型となり，その消費エネルギーは 56.6W 程度である [131]．直径 134mm × 83mm はおよそ JAXON の膝リンクの大きさと同じであり，これを膝に組み込むことは JAXON の膝体積を二倍にすることを意味する．また消費エネルギー 56.6W は JAXON の各軸モータが 定格 200W であること，モータが常に 200W を発揮するわけではないことを考えると，これも無視できない大きさであると言える．なお，300Nm は，120Kg の体重と 50cm の太もも長さを持つ人間がスクワットする際の，膝にかかる最大負荷と同じ程度であり，およそ人間と同程度の出力ということになる．クラッチ機構を小型化する試みとしては，backlash clutch [128] と呼ばれる遊びのあるかみ合わせ機構を用いるものがあるが，常に最大一方向にしか力を発揮できない機構であり，位置制御が困難であること，外乱に対して制御できない方向を持つことなど，その制御法については明らかになっているとは言いがたい．本研究の目的は，実等身大ヒューマノイドにも適用可能な出力と体積を持つクラッチ機構で，従来の位置制御を始めとする制御手法がそのまま適用可能な能動軸状態を作り出せる機構を開発することである（図 6.3）．

6.2.3 アクチュエータと角位置センサを有しソフトウェアにより受動軸を追従制御する水平ピンクラッチ機構

本章では，クラッチ機構の小型省電力化のためのアイデアについて述べる．

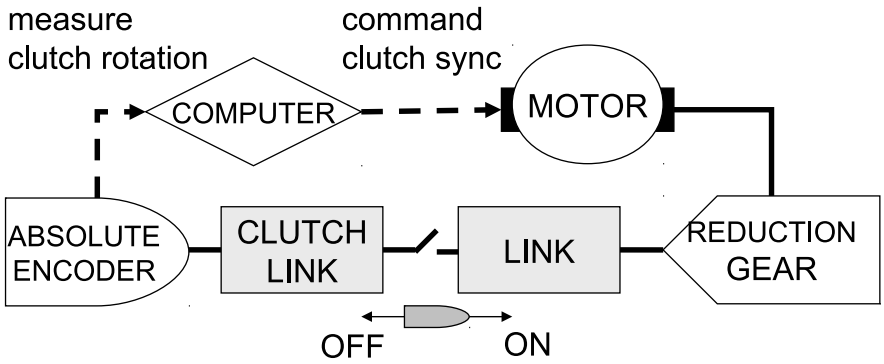


図 6.4. System flow of synchronized brake/clutch.

本章で述べるクラッチ機構の特徴は，機構の出力側（クラッチリンク）に角位置センサを有し，その出力にたいして入力側（リンク）をソフトウェアにより追従制御する点にある（図6.4）．

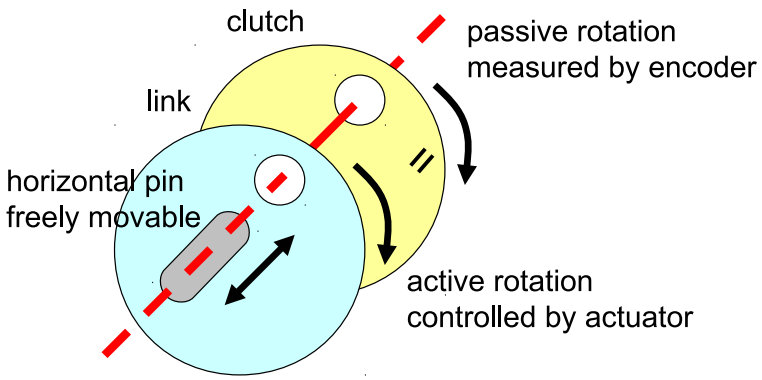


図 6.5. Both links have some holes for pin movement, and each hole has a relatively constant position.

また，両リンクにはピンの通過するための穴が開いており，それぞれの穴の相対位置関係は常に一定となるように制御される．したがってピンは自身の慣性力と摩擦力のみを受けながらピン穴内を自由に移動することができる．ピンの位置を両リンクにまたがって配置することで，トルクの伝達が行われる．いずれかの穴からピンが離れることでトルクは遮断される．従来のクラッチ機構では，両リンク間に電磁力等を用いてリンク間に摩擦力を生じさせることで

トルクの伝達量を制御していたが，本機構は摩擦力を用いず，ピンが機構的にトルクの伝達を行うため省エネルギー化が期待できる．実際，以降の章で示す一軸試験機では，ピンの位置を 5W の小型サーボモータで制御できることが確認できている．また，ロボットの各軸には本機構に必要となるアクチュエータや制御用コンピュータ，角位置センサがすでに搭載されているため，それらを流用することでクラッチ機構の小型化が実現できる．

6.2.4　一軸試験機の開発

本章では，開発した一軸試験機の，体積，出力強度，制御に求められる精度について述べる．

6.2.4.1　体積について

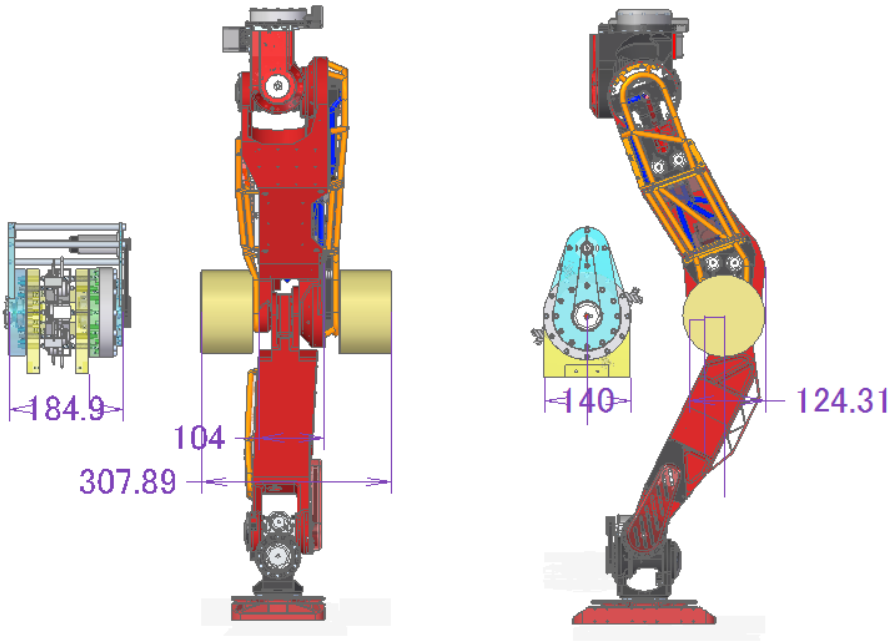


図 6.6. Size comparison with JAXON [1]. Small enough for real life-sized humanoid robot. JAXON has 32DOF, 127Kg, 200W motors, from 200 to 679Nm, volume of knee link is $150mm^3$.

図 6.6 に，一軸試験機と，等身大ヒューマノイド JAXON の脚部を並べて示す．JAXON の膝から伸びる黄色い円筒は，既存のクラッチ機構で最小のものとほぼ等しい体積を持つモデルであり実際には存在しない．図の右側，脚部を横から見た図を見ると，一軸試験機の幅が 140mm，JAXON の膝が 124 mm でありほぼ等しい大きさが実現できていることがわかる．図の左側，脚部を後ろから見た図を見ると，一軸試験機が幅 185 mm 程度で JAXON の幅 104mm に対して 1.8 倍程度大型化していることがわかる．ただし，一軸試験機では，直動アクチュエータの配置スペースを広くとっている都合，30 から 40mm 程度幅を狭くすることが可能である．したがって実際には 1.5 倍程度の大型化で済むと予想される．対して，既存の

ブレーキ・クラッチを表す黄色い円筒を見てみると，膝幅 307 mm 程度と，3 倍程度の大型化が必要と予想される．

6.2.4.2 強度について

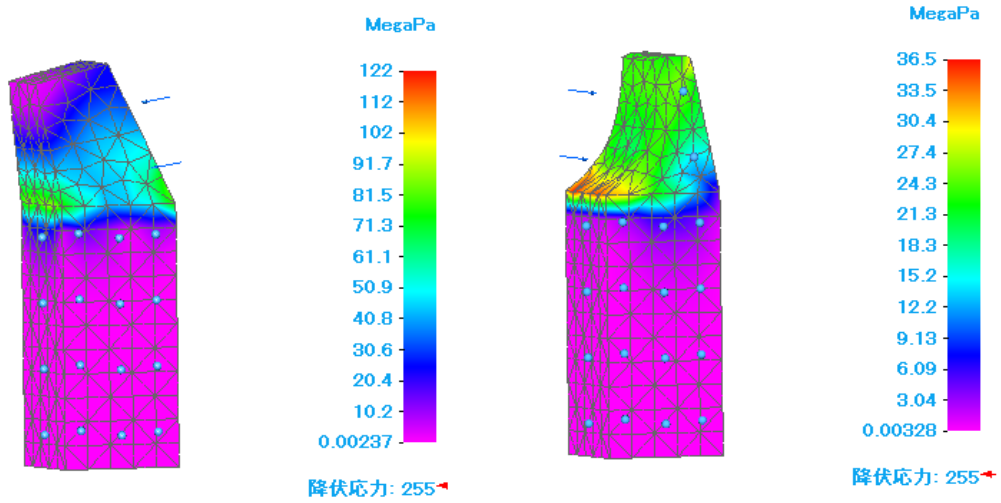


図 6.7. Simulated stress with finite element method (FEM).

図 6.7 に，有限要素法解析を用いて，300Nm のトルクが加わった時のピンの応力分布をシミュレーションした結果を示す．右の図は，ピンがリンクピン穴にぴったりと接触しているとした場合の結果，左の図は隙間があるとした場合の結果であり，例えばピン挿入時にリンク間相対位置が急激に変化するような場合には左の図に示す状況が起こりうる．右の図では，最大応力が 34.5MegaPa 程度であり，これは SCM440 の許容応力が 850MegaPa 程度であるのと比べて数十分の一である．左の図では最大応力 122MegaPa 程度であり，SCM440 の許容応力に比べて 7 倍程度の余裕がある．

6.2.4.3 ピンの傾斜について

ピンが機構的にトルクの伝達を行うためには，ピンの傾斜が受ける水平反力が摩擦力よりも小さくなる必要がある（図 6.8）．

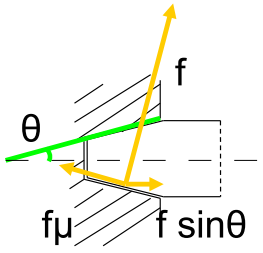


図 6.8. The condition of not sliding is $f\mu cos\theta \leq f sin\theta \Leftrightarrow \mu \leq tan\theta$. Because metal friction is about $\mu \geq 0.2$, if $\theta \geq tan^{-1}(0.2) \approx 11.3\ degrees$, $\leq tan\theta$ is satisfied.

このとき，ピンの位置に許される誤差の量は，ピンのリンク回転軸に対する半径方向位置か

ら図 6.9 のように計算できる．

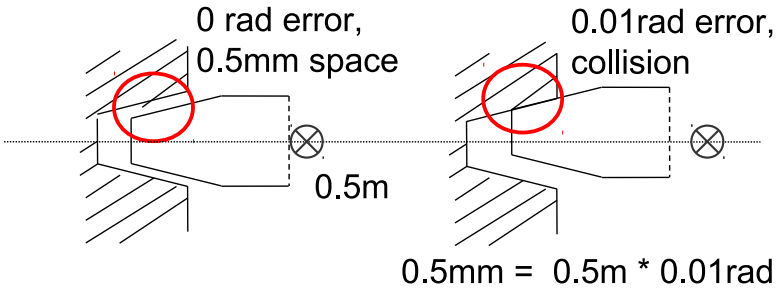


図 6.9. $tan^{-1}(0.2)$ slope and 0.5 m radius has about 0.01 *radians* error tolerance.

6.2.5　PD 制御を用いた同期制御評価予備実験

ここまで紹介してきた同期式水平ピンクラッチ機構の制御において一つの課題は同期制御の精度である．図 6.10 に同期精度評価のための実験装置を示す．受動リンク（以下 link2）は約五十度の高さ（軸真下を零度とする）から落とされる．アクチュエータにより駆動される link1 は link2 の位置をセンシングしながらそれを PD 制御により追いかける．

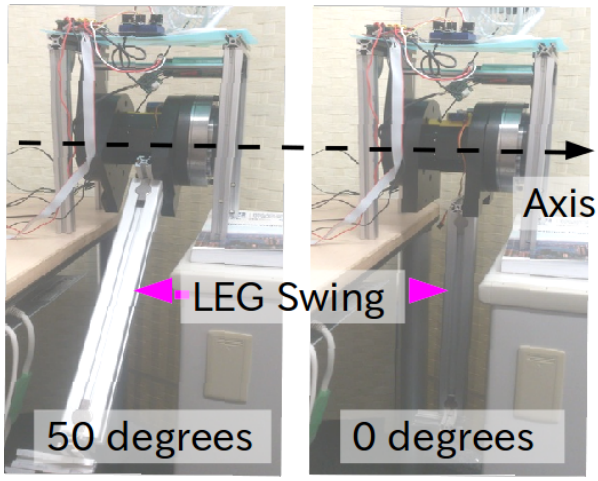


図 6.10. Synchronous control experiment: leg freely swings like pendulum.

図 6.11 い同期制御の結果を示す．図 6.11 の上側はセンシングされた位置を表わし link1 に取り付けられた ENCoder と link2 に取り付けられた ABSolute encoder の出力をそれぞれプロットしたものである．下側の図は，両位置の差分を表わし，最大で二度程度のずれが検出された．

以下ではこの二度の差を減らすための工夫を行っていく．

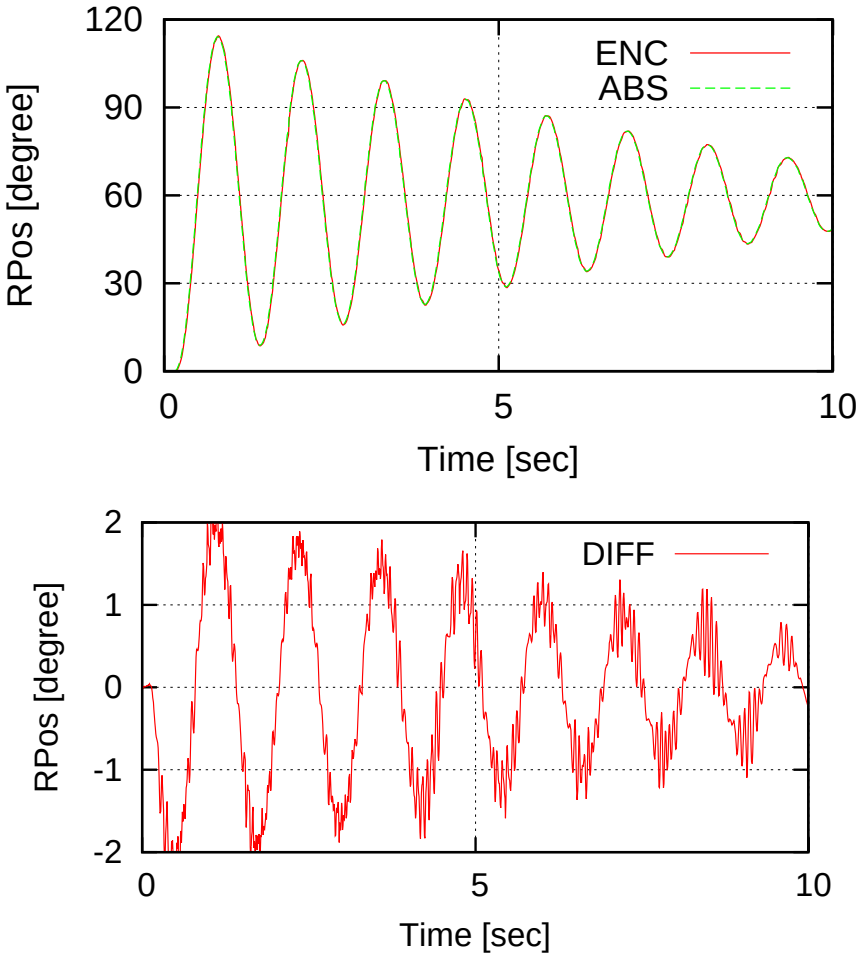


図 6.11. Experimental result (w/o EKF prediction): The target position of link1 is the current link2 position. The maximum error is over 2 degrees.

6.2.6 同期精度向上のためのカルマンフィルタを用いたリンク位置の予測

link1 が link2 の今の位置を次の目標値として PD 制御を行うかぎり時間遅れが生じることは避けられない．逆に考えると，link2 の次の制御周期における位置が分かればそれを目標に PD 制御を行うことで誤差を減らせる可能性がある．次の制御周期におけるリンク位置の予測に Extended Kalman Filter （以下 EKF）を用いることを考える [132].

6.2.6.1　動作モデル

慣性行列 I，質量 m，重力加速度 g，リンク位置 q ，リンク位置原点 q^0 を用いて受動振り子の運動方程式は以下のように表現できる．

$$\begin{aligned} I\ddot{q} &= -mgsin(q+q^0) \\ \Rightarrow \ddot{q} &= -\frac{mg}{I}sin(q+q^0) \end{aligned} \tag{6.1}$$

この振り子はロボットの関節を模したものであるので，関節位置の原点は動くものとして扱えるよう q^0 を導入した．また，この運動方程式ではリンクそのものの速度加速度運動を考慮していない．関節が高速に動作するロボットに適用するには運動方程式に改良が必要と予想される．

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \begin{pmatrix} \dot{q}_{k+1} \\ q_{k+1} \\ q^0_{k+1} \end{pmatrix} \\ &= \begin{pmatrix} \dot{q}_k - \Delta t\frac{mg}{I}sin(q_k+q^0_k) \\ q_k + \Delta t\dot{q}_k \\ q^0_k \end{pmatrix} + \boldsymbol{w}_k \end{aligned} \tag{6.2}$$

式 6.2 に EKF で用いる運動モデルを示す．k 番めの状態は，速度 $\dot{q}_k$，位置 q_k，そして位置原点 q^0_k により定義する．Δt は制御周期であり，10^{-3} 秒を以下では共通して用いることとする．$\boldsymbol{w}_k$ は零平均正規分布を仮定し，その分散 Σ_{w_k} としては $diagonal(0.0017\cdots0.0017)$ を用いた．

6.2.6.2　センサモデル

ロータリーエンコーダは link2 に取り付けら，その測定値は link2 の位置 q を表す．零平均正規分布 v_k と k 番めのセンサ状態 z_k は以下のように表現される．

$$z_k = q_k + v_k \tag{6.3}$$

$$= C\boldsymbol{x}_k + v_k \tag{6.4}$$

$$C = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \tag{6.5}$$

標準偏差 Σ_{v_k} は 0.0017 を用いた．

6.2.6.3　拡張カルマンフィルタ

EKF の計算は以下の手続きにより行われる．

6.2.6.3.1 Prediction

$$\begin{aligned}\boldsymbol{x}_{k+1|k} &= \begin{pmatrix} \dot{q}_{k|k} - \Delta t \frac{mg}{I} sin(q_{k|k} + q^0_{k|k}) \\ q_{k|k} + \Delta t \dot{q}_{k|k} \\ q^0_{k|k} \end{pmatrix} \\ &\approx \frac{\delta \boldsymbol{x}_{k+1|k}}{\delta \boldsymbol{x}_{k|k}} \boldsymbol{x}_{k|k}\end{aligned} \tag{6.6}$$

$$\frac{\delta \boldsymbol{x}_{k+1|k}}{\delta \boldsymbol{x}_{k|k}} = \begin{pmatrix} 1 & -\Delta t \frac{mg}{I} c & -\Delta t \frac{mg}{I} c \\ \Delta t & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (c = cos(q_{k|k} + q^0_{k|k})) \tag{6.7}$$

$$\Sigma_{x_{k+1|k}} = \left(\frac{\delta \boldsymbol{x}_{k+1|k}}{\delta \boldsymbol{x}_{k|k}} \right) \Sigma_{x_{k|k}} \left(\frac{\delta \boldsymbol{x}_{k+1|k}}{\delta \boldsymbol{x}_{k|k}} \right)^T + \Sigma_{w_k} \tag{6.8}$$

6.2.6.3.2 Kalman gain calculation

$$K_k = \Sigma_{x_{k+1|k}} C^T \left(C + \Sigma_{\boldsymbol{x}_{k+1|k}} C^T + \Sigma_{v_k} \right)^{-1} \tag{6.9}$$

6.2.6.3.3 Correction

$$\boldsymbol{x}_{k+1|k+1} = \boldsymbol{x}_{k+1|k} + K_k (z_k - C \boldsymbol{x}_{k+1|k}) \tag{6.10}$$

$$\Sigma_{\boldsymbol{x}_{k+1|k+1}} = (E - K_k C) \Sigma_{\boldsymbol{x}_{k+1|k}} \tag{6.11}$$

6.2.6.4 追従制御実験の結果

図 6.12 は EKF の prediction を用いた同期制御の結果を示す．上図は link1 の位置（ENCoder）と link2 の位置（ABSolute encoder）をプロットしたものであり，link1 と link2 がほぼ同期して振り子運動していることが見て取れる．下図はこれら位置の差分を表す．

図 6.11 と比較すると，EKF を用いたもののほうが誤差が小さくなっていることが分かる．図 6.11 では誤差の振動は最大で二度を越える程度の振幅を持ち，十秒時点でも誤差は一度を越えている．対して図 6.12 では，振幅は最大で 1.5 度程度であり，その後は一度を下回る誤差となっている．

図 6.13 は EKF の計算で求められる各パラメタであり，SENSOR が z_k，PREDICT が $q_{k|k-1}$，CORRECT が $q_{k|k}$，そして CENTER が $q^0_{k|k}$ である．一秒以降，位置原点

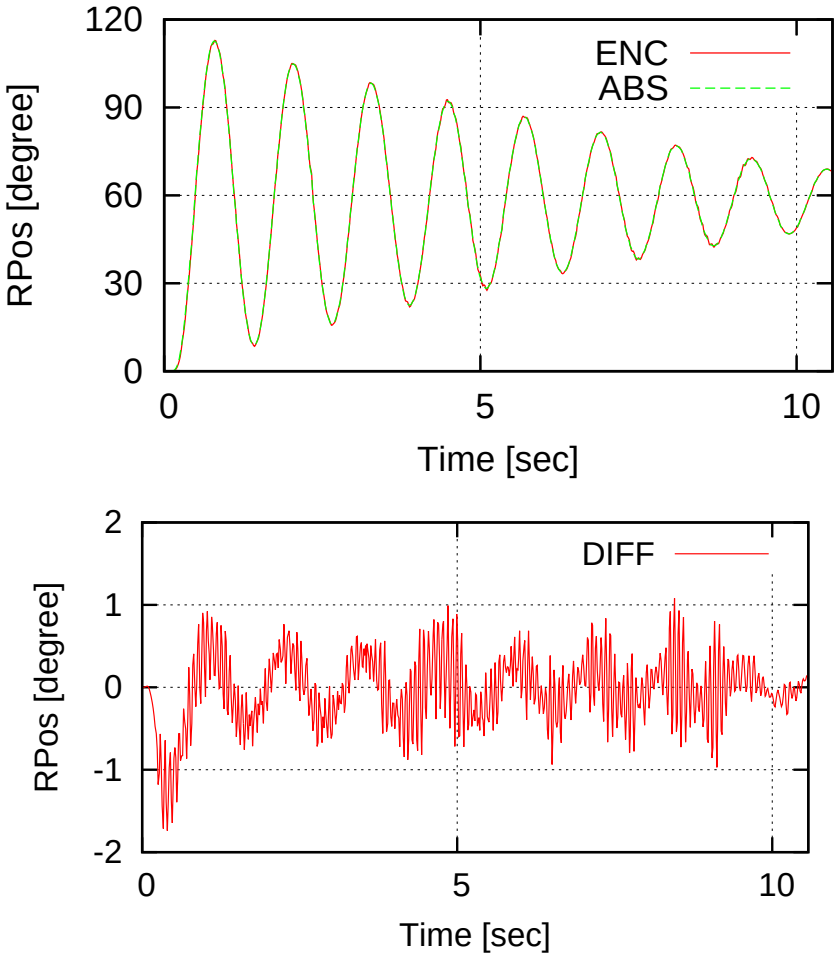


図 6.12. Experimental result (w/ EKF prediction): The target position of link1 is the predicted link2 position in the next timestep. The synchronous error is almost under 1.0 degrees.

CENTER=$q^0_{k|k}$ は八十度を中心に小さく振動しており，これは振り子運動のほぼ中央であることが分かる．$q^0_{k|k}$ を直接測定するセンサは実験に用いていないにも関わらず振動の中心が正しく予測できていることは興味深い結果である．

図 6.14 は link2 位置の prediction と correction の差を表す．DIFF $= q_{k|k-1} - q_{k|k}$ である．一秒以降，DIFF は 0.1 度程度であり正確な prediction が実現されていることが分かる．

6.2.7　等身大脚型ロボットへのクラッチ機構適用に向けた展望

本章では，等身大ヒューマノイドに搭載可能な強度と体積を持つクラッチ機構の開発，及び制御性能評価を行った．シミュレーションにより，機構が 300Nm の入出力を制御できる強度を持つこと，その体積が既存の小型大出力クラッチ機構を用いるのに比べ半分程度の体積となり，実等身大ヒューマノイド JAXON の膝機構として 1.5 倍程度の大型化で済むことを確認

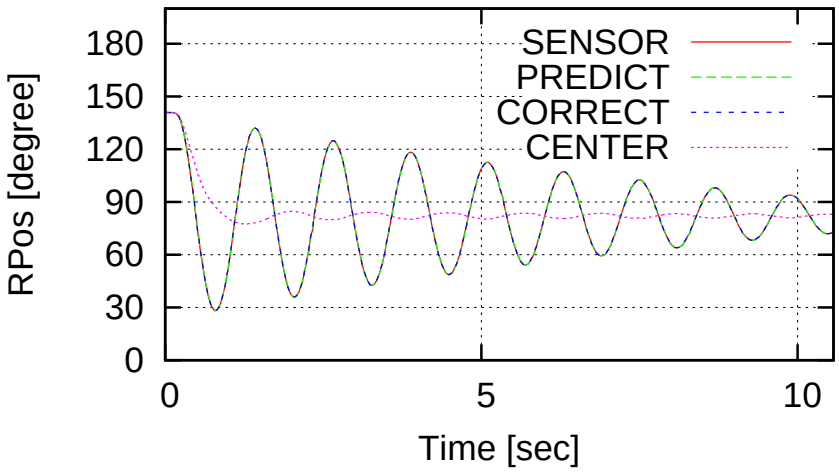


図 6.13. Transition of EKF parameters: SENSOR=z_k, PREDICT=$q_{k|k-1}$, CORRECT=$q_{k|k}$, CENTER=$q^0_{k|k}$.

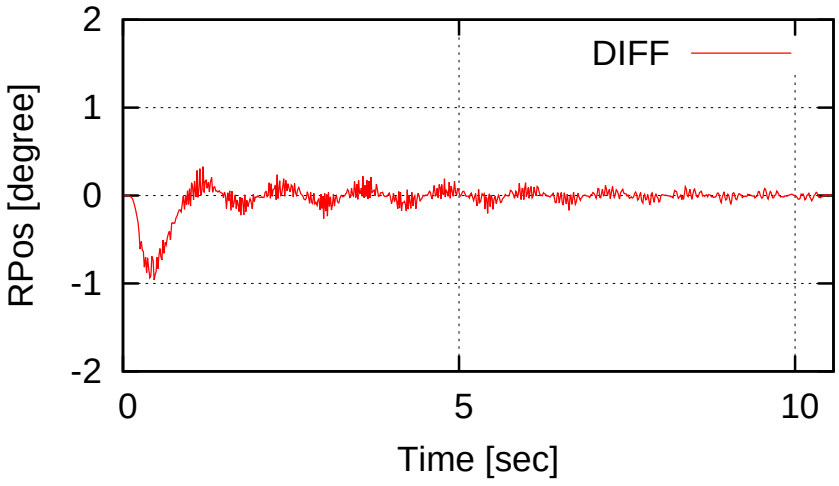


図 6.14. Distance between the predicted link2 position and the corrected one ($= q_{k|k-1} - q_{k|k}$).

した．さらに同期制御の精度を評価するために受動リンク（link2）を駆動リンク（link1）で追従制御する実験を行った．精度向上のため，link2 の位置を Extended Kalman Filter（EKF）を用いて link2 の一制御周期後位置を予測する手法を調査し精度向上が確認された．以降では本章のクラッチ機構を実際のロボットに搭載していくが，クラッチピンの傾斜は前章の実験で得られた図 6.11 が最大でも 2.2 度程度の誤差で追従できていたことからそれよりも大きく 2.5 度となるように角度を決めればよい．これは前章の実験で用いた脚はアルミフレームをつなげただけのものであり慣性モーメントが小さく振り子運動の周期も早いものであったが，実際のロボットの脚は金属骨格にモータや力センサ類を取り付けたものであり慣性モーメントは

大きい．したがって，図 6.11 よりも振動は遅くその分追従精度も高いと予想される．

6.2.8　ML1 への拘束解放可変機構の適用

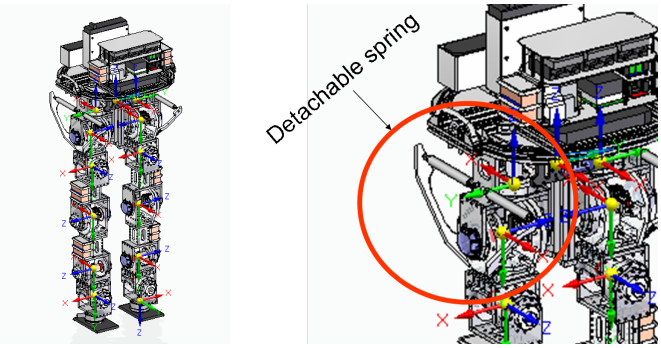


図 6.15. Left image shows ML1 whole-body cad date. Right image shows enlarged view of detachable spring mechanism. The spring connect and disconnect to upper body link to switch passive walk and active walk.

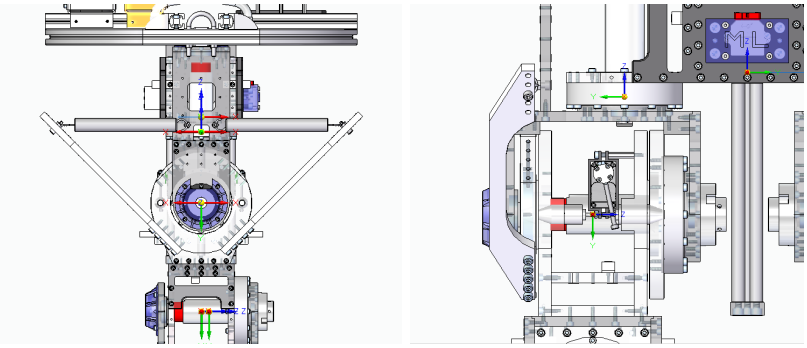


図 6.16. Left image shows the side of detachable spring mechanism, and right image shows the front view of the mechanism.

本研究で開発した脚型ロボット ML1 には股関節ピッチ軸にバネ機構が搭載されている．図 6.15 にその機構の外観をしめす．左図は脚型ロボット全体の CAD データを図として表示したものであり，右図はバネ機構を中心に拡大したものである．図 6.16 はさらにバネ機構を左図では横から，右図では正面から拡大して示したものである．バネはロボットの前後に一本ずつ伸び，足がまっすぐと地面にむけて伸びている姿勢で釣り合いがとれるようになっていることが分かる．またバネが固定された角のような棒は中央が凹んでいることが分かるが，これは関節の可動域を阻害しないためである

上半身のあるロボットであってもバネ機構により体幹リンクを両脚の中央付近にコントロールしてやることによって受動歩行が可能になることが，過去の研究で示されている [133]．しかしながら，能動歩行と受動歩行を一台で両方行うことを考えた時，そのバネ機構が問題となる可能性がある．図 6.17 にその問題について示す．一つ目の問題は，バネが関節と内力を生じる駆動を妨げることである．バネの釣り合いの位置から離れるほどこの力は大きくなり，関

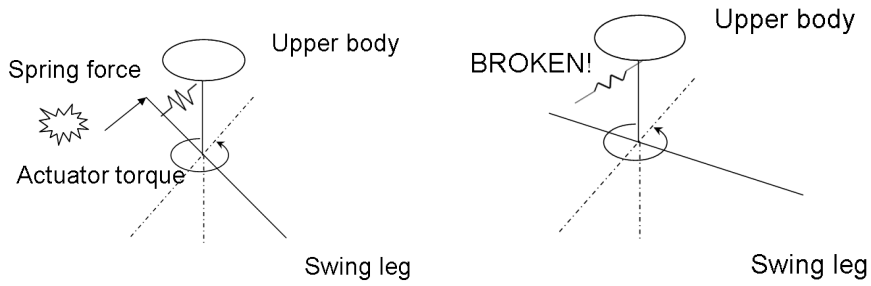


図 6.17. Spring mechanism for passive walk has two problems to disturb active walk and the other active motions. First problem is inner force which somtimes enlarges the joint torque and energy consumption. Second problem is corruption of spring as the result of high-load motion.

節の負荷を増大させることが考えられる．二つ目の問題は，バネの伸びに上限が存在することであり，この上限が関節の可動域を阻害したり，上限を越えた運動が起こることでバネの破損が起こりうることである．

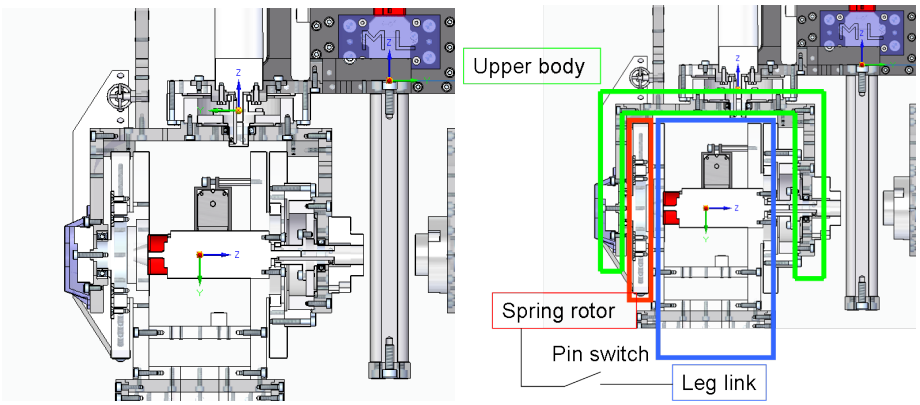


図 6.18. The sectional view of detachable spring mechanism. The mechanism has three independent links; The first one is upper body link (green link), the second one is spring rotor (red rotor), and the third one is leg link (blue link). These links independently rotate if the clutch pin is detached and vice versa.

以上の問題を解決するため，本研究では図 6.18 に示すような機構を用いてバネの力の伝達の遮断・接続を制御する方法を用いる．左図はバネ機構の断面図であり，右図はそれぞれのリンクを役割ごとに色付きの枠線で囲んで示している．緑のリンクは上半身リンクに固定されている．青のリンクは脚リンクである．赤の円盤は青リンクの回転軸と同じ回転軸を持ちつつ独立して回転でき，バネを介して上半身リンクにも繋がっている．円盤と脚リンクにはクラッチピンが配置されおり，これが円盤に刺さることでバネの力が上半身リンクと脚リンク間で伝達される．逆にクラッチピンが離れるとバネは自由に回転できる円盤のみに力を及ぼすことになるため上半身リンクと円盤は互いに固定されたリンクのように振る舞う．もちろん円盤は慣性

力でわずかに振動するがバネが体幹を支えるほど強いため振動は小さい．またクラッチピンはラジコンサーボ [134] で駆動される．

6.3　拘束解放可変機構を用いた歩行行動探索法と実験

クラッチ機構により受動軸と能動軸の切り替えが可能であるようなロボットで，受動軸を効率的に用いることで歩行に要する消費エネルギーを低減させることが可能であるかを検証する．クラッチを用いた歩行は二種類考えられる．一つはクラッチの拘束と解放を繰り返すことで，足を放り投げるようにして歩く方法（能動受動歩行）．もう一つはクラッチを解放したままにしておき他の関節の運動と受動軸の動力学に基づき歩行する方法（準受動歩行）である．以下ではまず能動受動歩行について軽く触れ，上半身のあるロボットのための準受動歩行モデルと，それに基づく行動探索法について実験と考察を行い，最後に実ロボットでの準受動歩行実験を行う．

6.3.1　クラッチの拘束解放を切り替えながら歩行する能動受動歩行

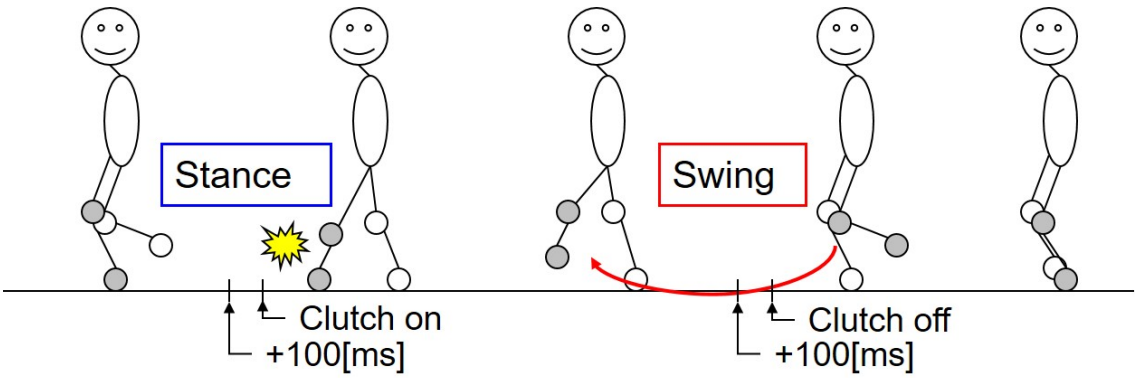


図 6.19. Search clutch on/off timing, and consider on/off delay.

膝関節にクラッチ機構を有し，それが百ミリ秒の遅延をもって拘束と解放を切り替えられるとする．この遅延はクラッチ機構の性質により決まるものである．遊脚ステート中のある時刻 t_{off} にクラッチが解放状態になり百ミリ秒の遅延を経て遊脚が受動軸になり，遊脚接地後の時刻 t_{on} にクラッチが拘束状態になり百ミリ秒の遅延後に能動軸になるというモデルで t_{on}, t_{off} を前述の探索空間 $\mathcal{X}$ に含めて探索を行った．結果，t_{off} が遊脚から接地までにかかる時間よりも大きな値になることでクラッチを用いない歩行が生成されることが確認された．生成された歩行について t_{on} だけをさらに直線探索したが，やはり CoT は下がらなかった．以上から本章で用いたモデルではクラッチの拘束解放を適切に行うことによって省エネルギー歩行は実現できないと考えるが，エネルギー以外の側面で応用できないかを今後研究したいと考えている．

受動軸の切り替えを歩行に応用することが可能であることを確認するために，CoT 最小化

だけではなく受動軸状態での移動距離を最大化するよう歩行動作を生成する問題をとき歩行実験を行った．歩行の様子を図 6.20 に示す．

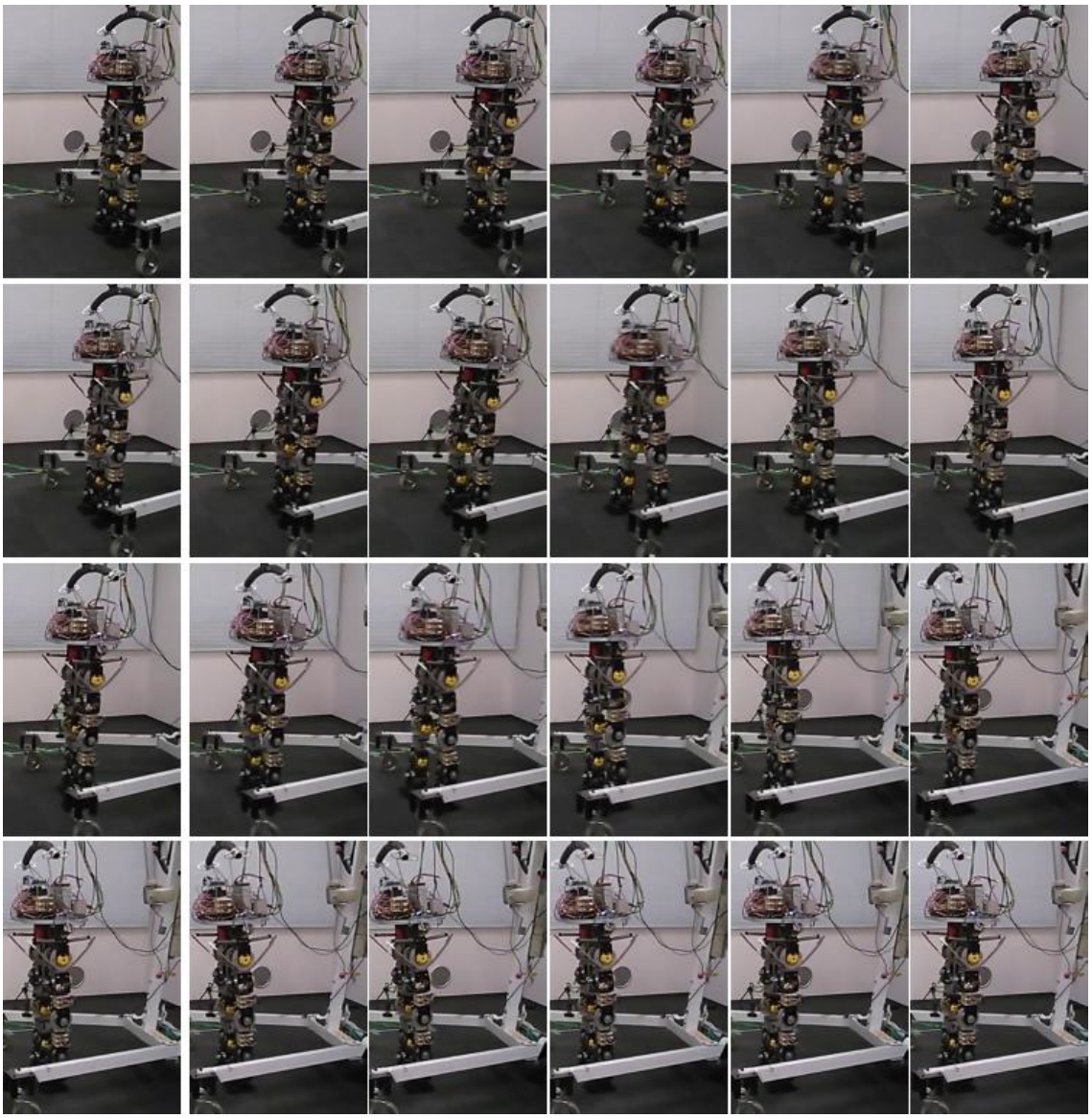


図 6.20. Snapshots of an active-passive walking motion in real world.

クラッチピンを駆動するラジコンサーボの指令値とエンコーダの測定値を図 6.21, 6.22 に示す．図中 state で示す線は +20 がクラッチピンを差した状態（能動軸），-20 が抜いた状態（受動軸），+10 が能動軸を受動軸にするためにラジコンサーボへと指令値が書き込まれている状態，-10 が受動軸を能動軸にするために指令値が書き込まれている状態を表す．-20, -10 の状態では受動軸にたいして追従して DC モータが制御される．図中 enc1, enc2 で示す線はラジコンサーボに搭載されたエンコーダの測定値でありそれぞれ原点がことなるためグラフの形は数度ずれて表示されている．すべてのラジコンサーボは $-10 \sim -8$ 度程度でちょうど

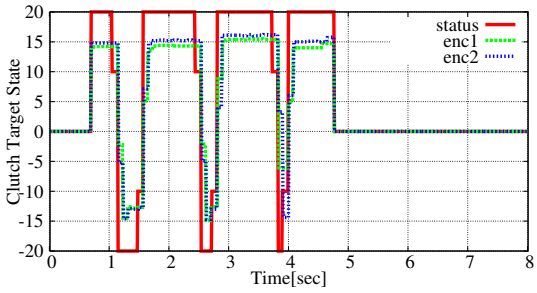


図 6.21. Right knee clutch status and servo encoder

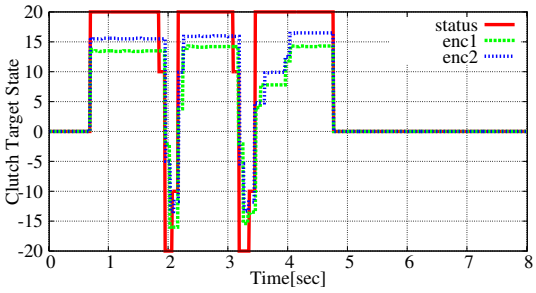


図 6.22. Left knee clutch status and servo encoder

ロータから抜けた状態になるが，それより 5 度余裕をもって −15 ～ −13 度程度を指令値として用いている．図より -10 度より小さなエンコーダ値の瞬間が存在することが確認でき，受動軸と能動軸の切り替えを伴う歩行が実現できていることが確認できる．

図 6.23，6.24，6.25 はそれぞれ実世界上での能動受動歩行中の姿勢センサクオータニオン，SIMBICON のステート，左足各関節の目標角度と測定角度である．ID=4 の関節角度グラフが受動軸と能動軸の切り替えを行っている左足膝関節のものであるが，受動軸となっている 2 秒付近，3 秒を少し過ぎたところで指令値と測定値が綺麗に重なっていることが確認できる．これは受動軸にたいして DC モータが追従制御されているためである．以上の実験により開発したクラッチ機構が歩行へも応用可能であることが示された．

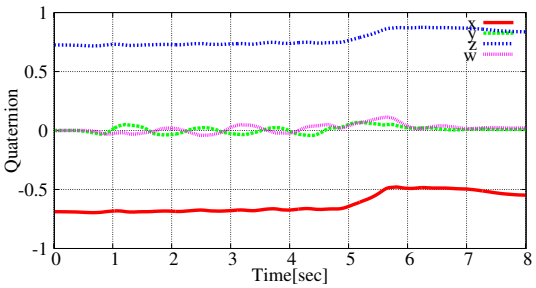


図 6.23. IMU quaternion

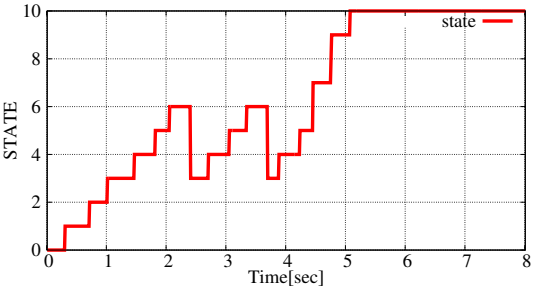
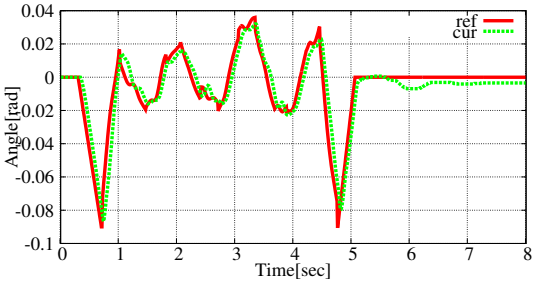
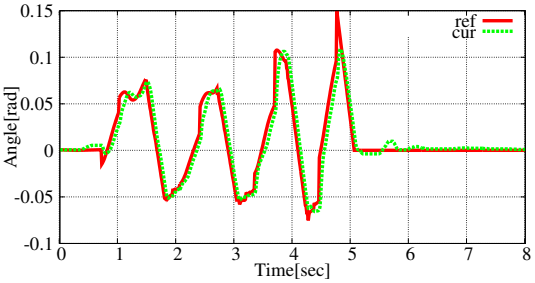


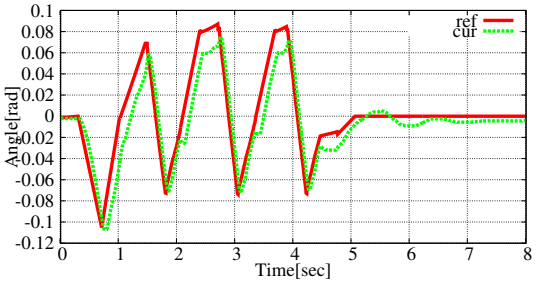
図 6.24. SIMBICON states



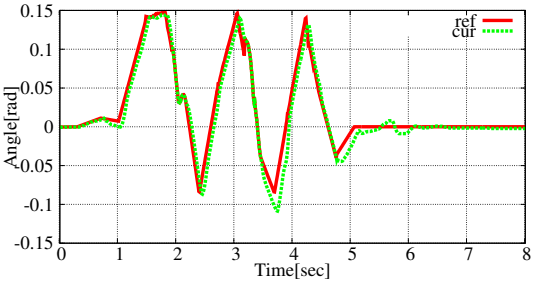
ID 1



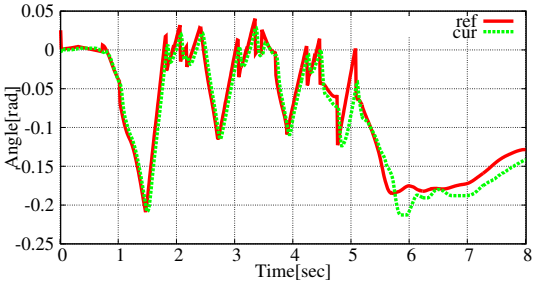
ID 2



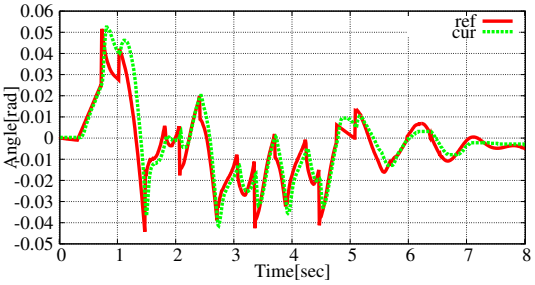
ID 3



ID 4



ID 5



ID 6

図 6.25. Joint angles

6.23. Transition of quaternion while active-passive walking motion in real world. These values are calculated by using an open source implementation of [107].

6.24. State transition while active-passive walking motion in real world. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

6.25. Joint angle targets and measured values for left six joints while active-passive walking motion in real world. Because the motion is symmetric, right joints move similar to left joints.

6.3.2　体幹リンクを有するロボットの受動歩行モデル

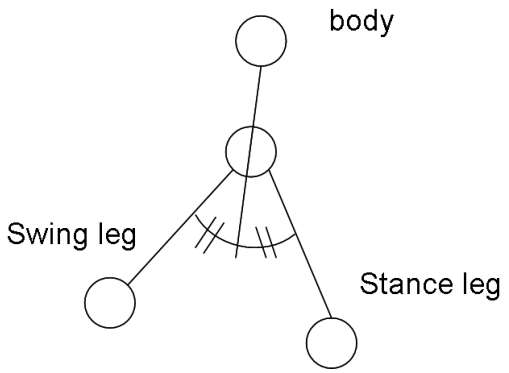


図 6.26. Passive walk with upper body needs upper body constratins such that the upper body is always placed in the middle of both legs.

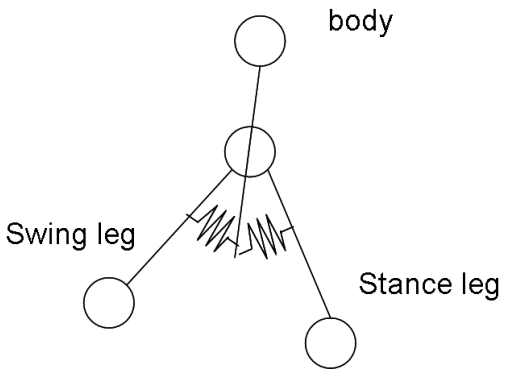


図 6.27. Passive walk with upper body will be possible if the upper body is placed around the middle of both legs by using spring.

上半身のあるモデルでは，脚の関節が受動軸になることで上半身が受動軸を回転軸として転倒してしまうという問題がある．[106] の研究では，この問題を解決するために体幹のリンクが両足の中心に常に位置するよう動くというモデルを提案している．またこのモデルを用いることで実際に安定な受動歩行動作が生成できることが示されている．図 6.26 にその模式図を示す．また同じ研究 [106] では，体幹を厳密に両足の中央に位置させずとも，ばね等を用いて中心付近に制御してやれば受動歩行できる可能性が示唆されており，[133] では実際に小型の受動歩行ロボットを作成し歩行が実現されている．

図 6.27 にその模式図を示す．体幹を両脚の中央に厳密に位置させるモデルを実ロボットで実現するためには，両脚を剛体のリンクで接続しそのリンク中央を体幹に固定するといった機構が必要であるが，バネを用いるモデルでは脚と体幹をバネで接続するだけでよく容易に機構を実現できるため，こちらのモデルを用いることとする．

以下，このモデルを用いて歩行行動制御が探索できるかを検証する．

6.3.3　バネとクラッチを用いた準受動歩行行動探索実験

6.3.3.1　運動モデル空間と身体環境モデル空間からなる探索空間について

4.3.3 章にて，SIMBICON を歩行モデルとする行動生成器の探索空間と探索条件，評価関数についてまとめた．本章での探索に置いても探索条件と評価関数は変わらない．探索空間についてもおおよそ同じであるが，本章の探索では身体環境モデル空間に属する体幹をささえるバネ機構のパラメタも探索空間に含める．バネの配置される位置姿勢は固定としてバネ係数のみを探索空間に含める．また探索の初期状態は全関節角度が零であるという点は同じである

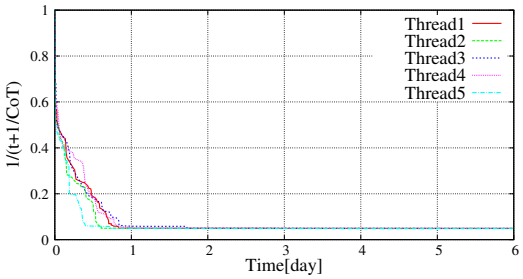


図 6.28. Transition of objective function ($\frac{1}{t+\frac{1}{CoT}}$) while searching semi-passive walking motion.

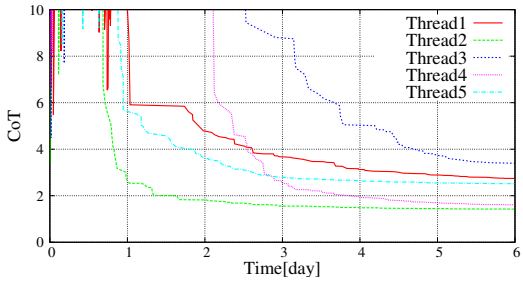


図 6.29. Transition of CoT: Cost of Transport while searching semi-passive walking motion.

が，バネとクラッチ機構を有する両股関節ピッチ軸はトルクを発揮せず関節の摩擦とバネのみで運動するようシミュレーションを行った．

6.3.3.2 準受動歩行行動の探索実験

図 6.28 に SIMBICON モデルとバネ係数を探索したときの，評価関数 式（4.75）の遷移を三日間に渡ってプロットしたグラフを示す．探索を五スレッドを独立して行った．歩行は十五秒後に停止ステートに遷移し二十秒間転倒しないものを探索した．したがってグラフ縦軸で $1/20 = 0.05$ を下回るスレッドは転倒しない解を発見しているということになる．図をみると全スレッドが一日程度で転倒しない解を見つけられていることが確認できる．また図 6.29 は CoT の遷移を同期間に渡ってプロットしたものである．一番速く収束しているスレッドでは二日時点で CoT $= 2$ を下回る程度．三日時点で CoT $= 1.5$ 程度の解が得られていることが確認できる．これは 4.3.3 章で示したバネをとクラッチを用いない最適歩行探索の六日目の結果（CoT ≈ 2.0）よりも 25% 程度エネルギー効率の良い答えである．得られたバネ係数は 4455.8 N/m であった．

歩行開始から三秒後に停止ステートへと遷移を開始し六秒間転倒しない動作のセンサシミュレーションを図に示す．図 6.32 は歩行中に推定された姿勢センサのクオータニオンを表す．二秒前から四秒後までの歩行中のクオータニオンが少し初期値よりオフセットしていることが確認できる．生成された準受動歩行では体幹をややのけぞらせることでバネの力で足を前にだすような行動が生成させているためである．

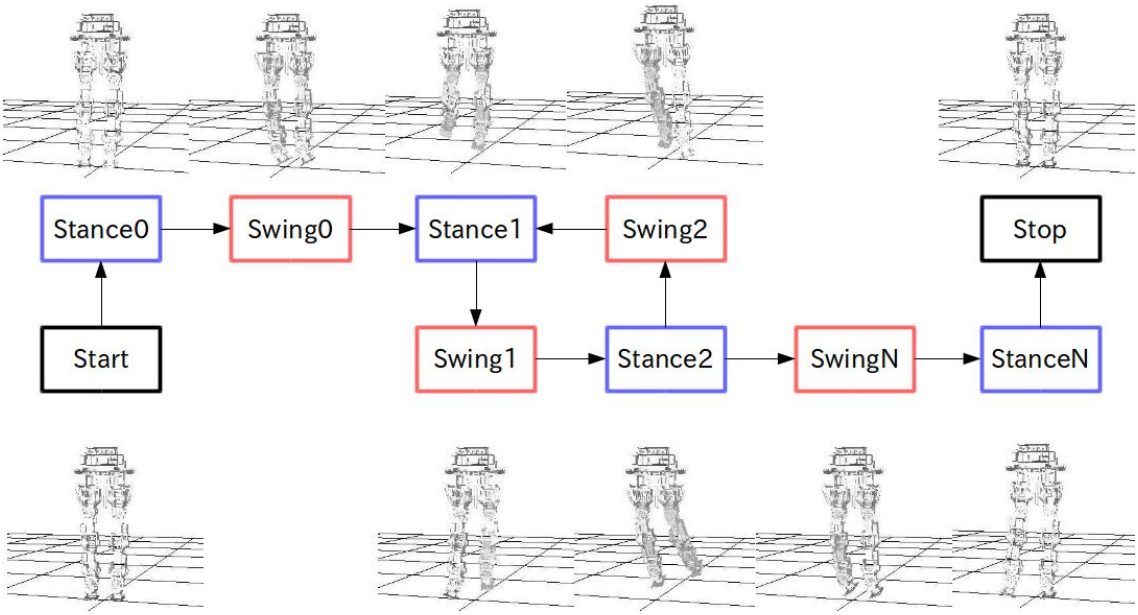


図 6.30. Semi-passive walking motion is optimally generated. Start/Stop postures are given; all joint angles are zeros. The other postures are optimized to minimize CoT: Cost of Transport.

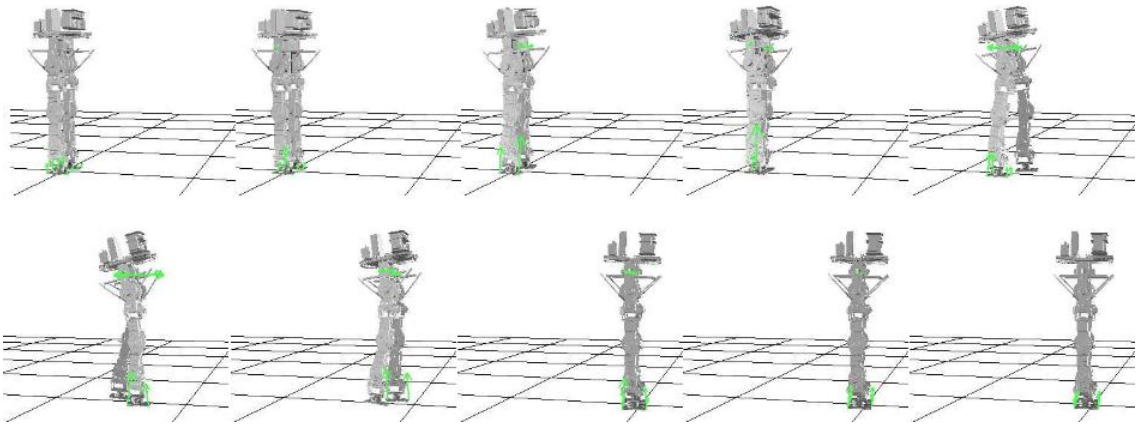


図 6.31. Snapshots of optimal semi-passive walking motion in simulation world.

図 6.33 は歩行中のステート遷移について表したものである．0 から 9 までのステートがそれぞれ， Start, Stance0, Swing0, Stance1, Swing1, Stance2, Swing2, SwingN, StanceN, Stop に対応している．六歩目で停止していることが分かる．最後に図 4.27 に各時刻における関節角度目標値と，シミュレーションされた関節位置を重ねてプロットしたものを示す．関節 ID = 2 のグラフで目標値が常に零なのは受動軸の目標角度として零を用いたためで意味はない．目標値と関係なく実測された関節位置はなめらかに振動していておりバネの力のみで運動する受動軸がシミュレーションできていることが確認できる．

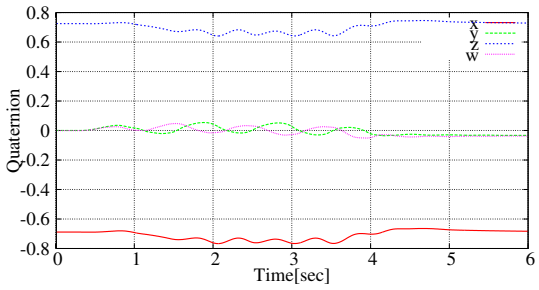


図 6.32. IMU quaternion

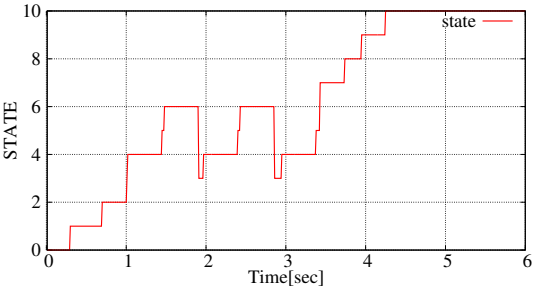
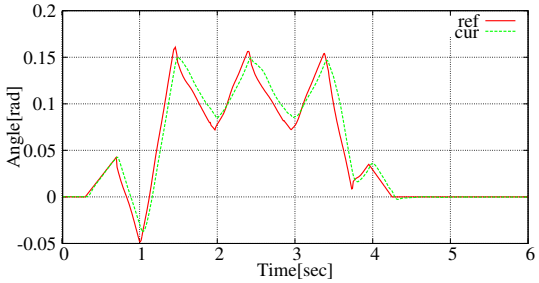
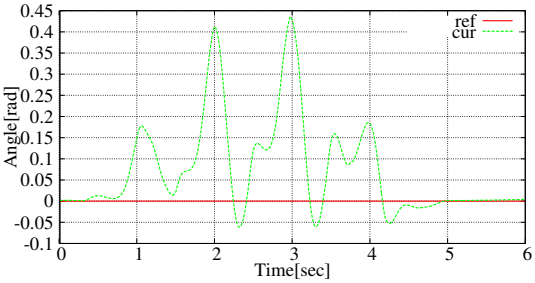


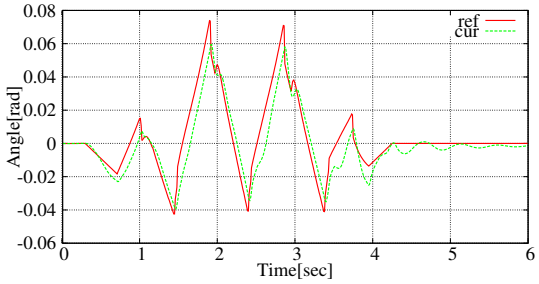
図 6.33. SIMBICON states



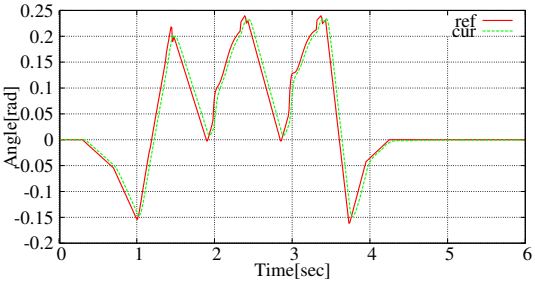
ID 1



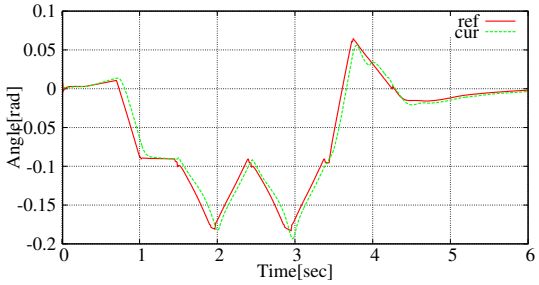
ID 2



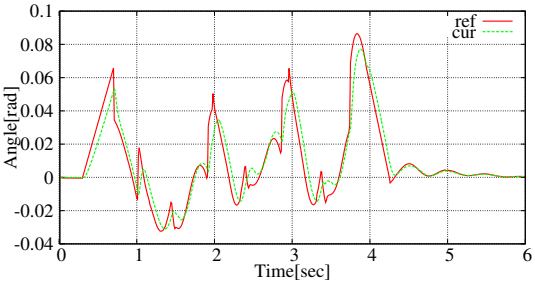
ID 3



ID 4



ID 5



ID 6

図 6.34. Joint angles

6.32. Transition of quaternion while semi-passive walking. These values are calculated by using an open source implementation of [107].

6.33. State transition while semi-passive walking. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

6.34. Joint angle targets and measured values for left six joints while semi-passive walking motion of actual robot. Because the stepping motion is symmetric, right joints move similar to left joints.

6.3.4　実ロボットによるバネとクラッチを用いた準受動歩行行動実験

準受動歩行行動が初期値を用いない探索により決定され，それを実ロボットで行った様子が図 6.35 である．一度モデル推定を行い歩行動作を修正しているため前章のシミュレーションとまったく同じ動作ではない．歩行は開始時刻から三秒を過ぎた時点から停止ステートへと遷移し六秒間で動作を終了した．転倒せずに再現性を持って数歩の歩行と停止が可能であることが確認され，開発したロボットの機構で準受動歩行が可能であることが示された．

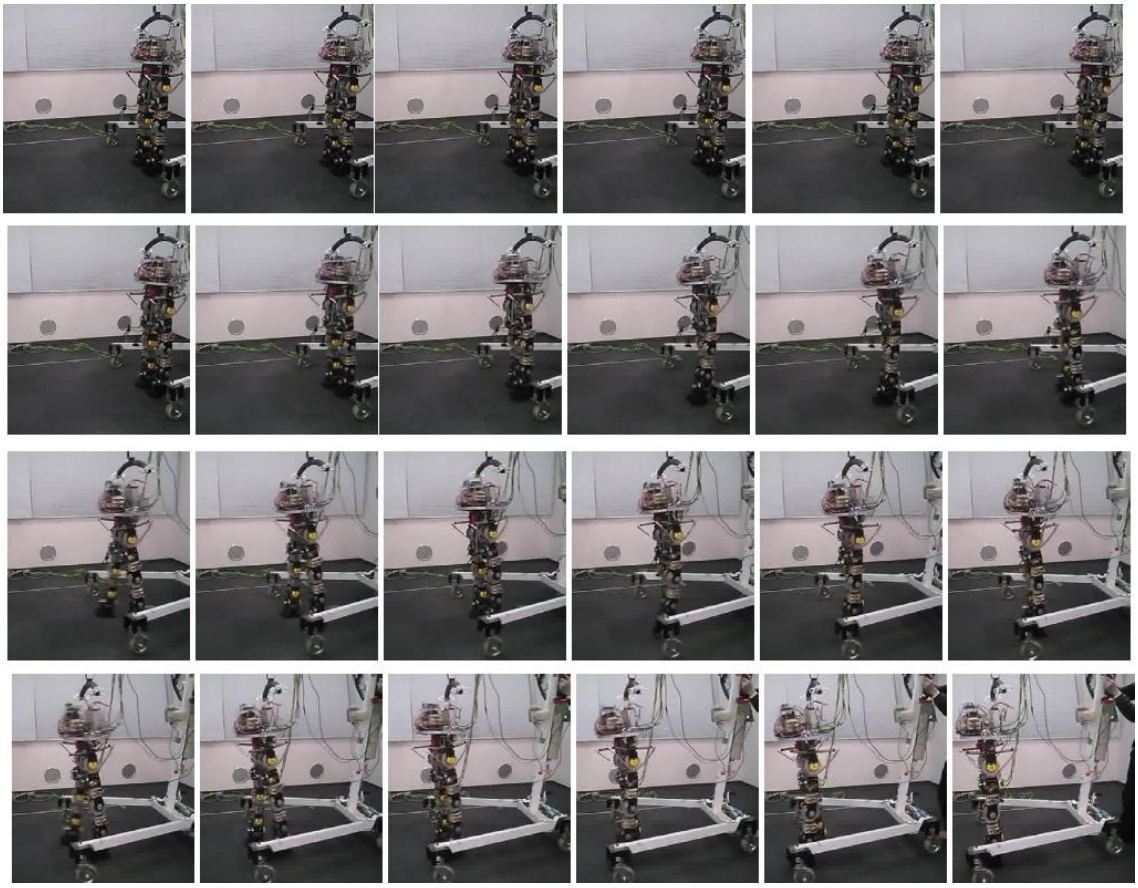


図 6.35. Snapshots of a semi-passive walking motion in real world.

図 6.36，6.37，6.38 はそれぞれ実世界上での準受動歩行中の姿勢センサクオータニオン，SIMBICON のステート，左足各関節の目標角度と測定角度である．受動軸である ID=2 の股関節ピッチ軸では指令値は常に零であるが，なめらかに振動していることが確認できる．

最後に歩行中の消費エネルギーを示す．前章で示した動歩行時の消費エネルギーと比べてピークのエネルギー量が半分程度まで小さくなっていることが確認できる．六秒間で 1 m の距離を 1350 J で移動し，CoT は約 2.7 であった．

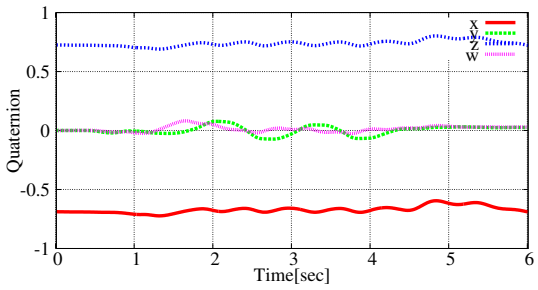


図 6.36. IMU quaternion

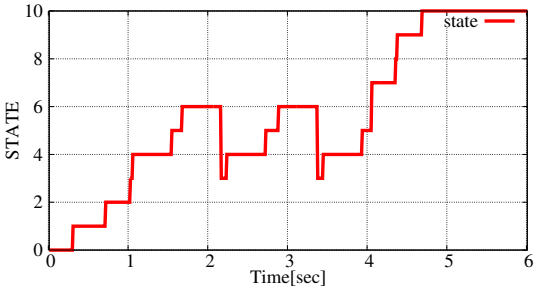
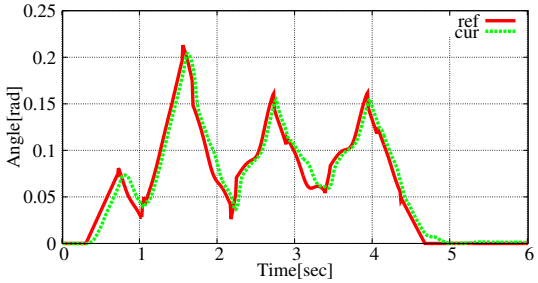
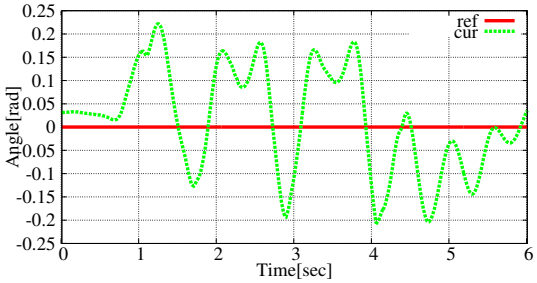


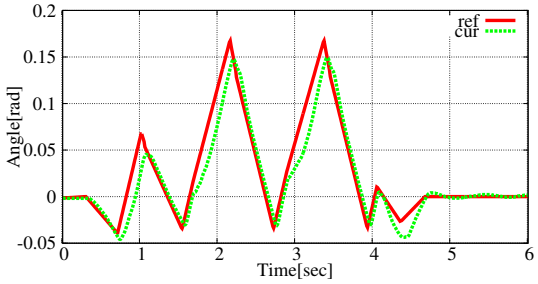
図 6.37. SIMBICON states



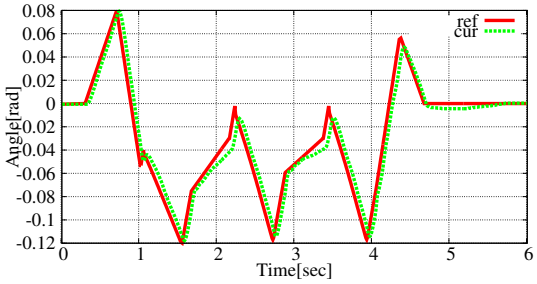
ID 1



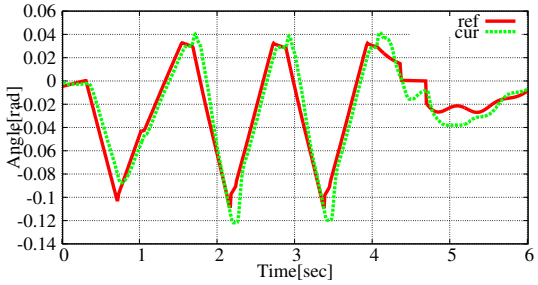
ID 2



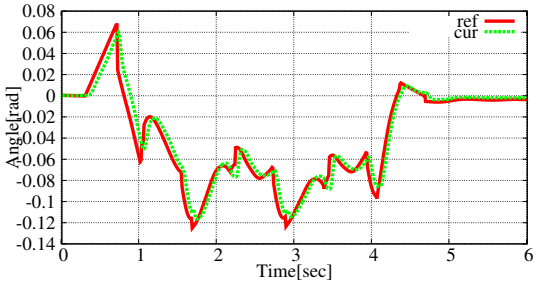
ID 3



ID 4



ID 5



ID 6

図 6.38. Joint angles

6.36. Transition of quaternion while semi-passive walking motion in real world. These values are calculated by using an open source implementation of [107].

6.37. State transition while semi-passive walking motion in real world. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

6.38. Joint angle targets and measured values for left six joints while semi-passive walking motion in real world. Because the motion is symmetric, right joints move similar to left joints.

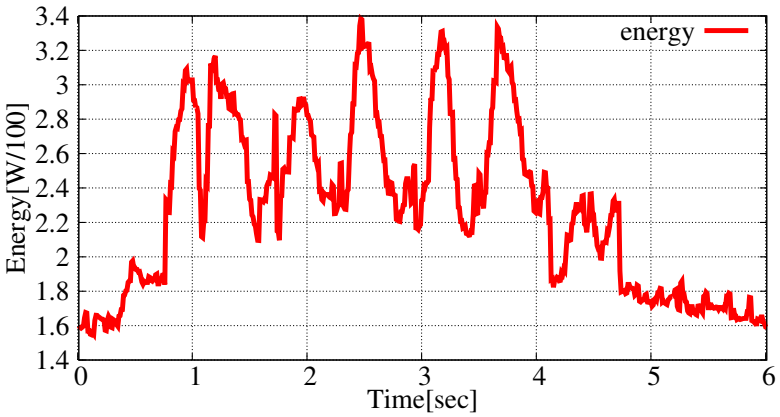


図 6.39. Energy consumption while optimized walking. The peak of energy consumptios are found in starting and stopping phase of walking.

6.3.5　実ロボットによる手動ブレーキを用いた低自由度歩行行動実験

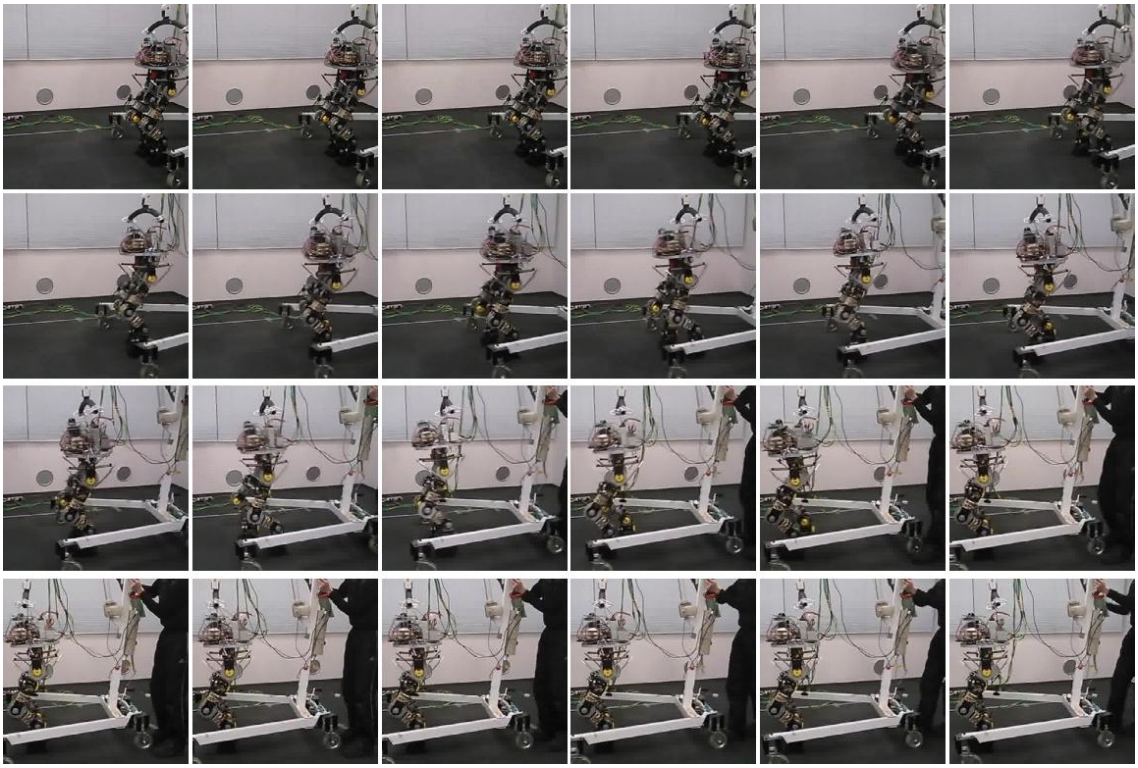


図 6.40. Snapshots of a brake walking motion in real world.

図 6.40 に両膝関節をブレーキにより固定した状態歩行を行った様子を示す．歩行は開始時刻から三秒を過ぎた時点から停止ステートへと遷移し六秒間で動作を終了した．

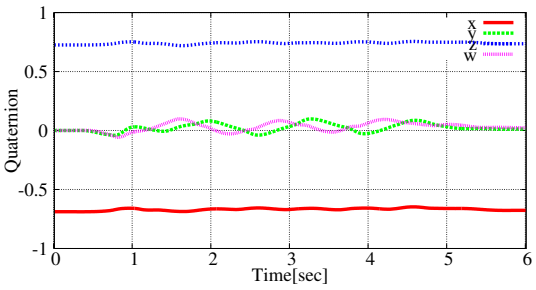


図 6.41. IMU quaternion

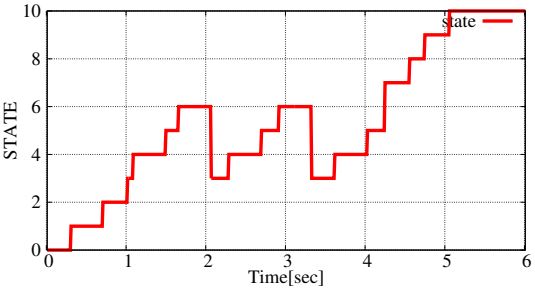
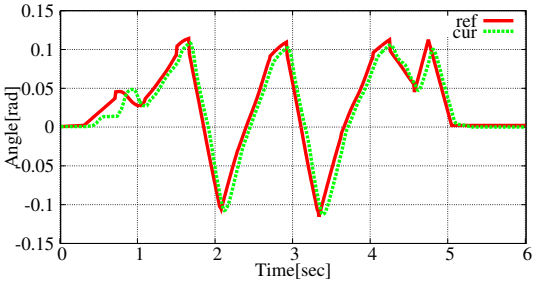
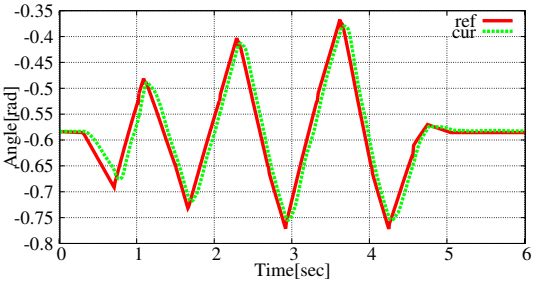


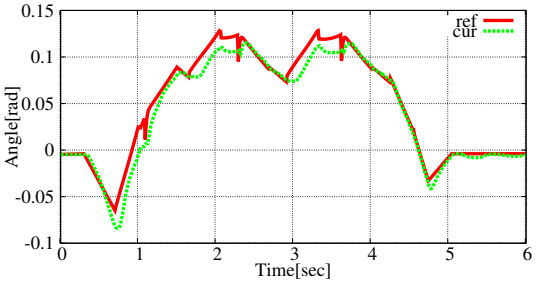
図 6.42. SIMBICON states



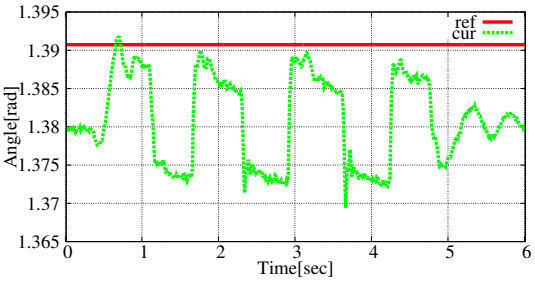
ID 1



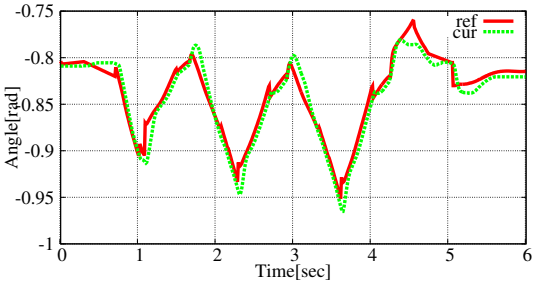
ID 2



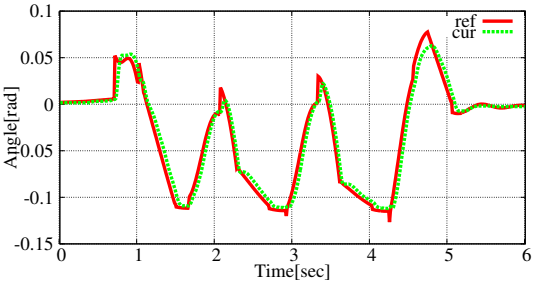
ID 3



ID 4



ID 5



ID 6

図 6.43. Joint angles

6.41. Transition of quaternion while brake walking motion in real world. These values are calculated by using an open source implementation of [107].

6.42. State transition while brake walking motion in real world. The state ids are 0 for Start State, 1 for Stance0, 2 for Swing0, 3 for Stance1, 4 for Swing1, 5 for Stance2, 6 for Swing2, 7 for SwingN, 8 for StanceN, and 9 for Stop.

6.43. Joint angle targets and measured values for left six joints while brake walking motion in real world. Because the motion is symmetric, right joints move similar to left joints.

図 6.41，6.42，6.43 はそれぞれ実世界上での歩行中の姿勢センサクオータニオン，SIMBICON のステート，左足各関節の目標角度と測定角度である．ID=4 の膝関節ピッチ軸がブレーキにより固定された軸であり，わずかに振動しているがその振幅は小さくほとんど動いていないことが確認できる．両膝を固定しての動作であるため，片足の自由度は 5 であり両足合わせて 10 自由度．これは体幹を固定して脚先端の位置姿勢を自由に制御することのできない低自由度のロボットであり，重心質点モデル等のモデルリダクションと逆運動学を用いた歩行生成手法では制御できないが，本論文で用いた全身自由度を用いた制御手法であれば可能である．

6.4　環境モデル空間を変化させながら行動生成器探索するロバスト歩行生成実験

環境モデル空間を行動生成器空間と同時に探索できるシステムでは，地面の傾きや柔らかさ，未知の外力といった環境の変化に対しての挙動も探索に含めることができる．以下では未知の外力にたいしてロバストにバランスをとるようなその場足踏み動作生成と実ロボットを用いた実験を行う．

6.4.1　未知外力の加わる環境でのロバスト歩行探索法

前 4.3.3 章にして示した省エネルギー歩行探索法の評価関数を変えることで，未知外力の加わる環境でのロバスト歩行探索と歩行実験を行った．

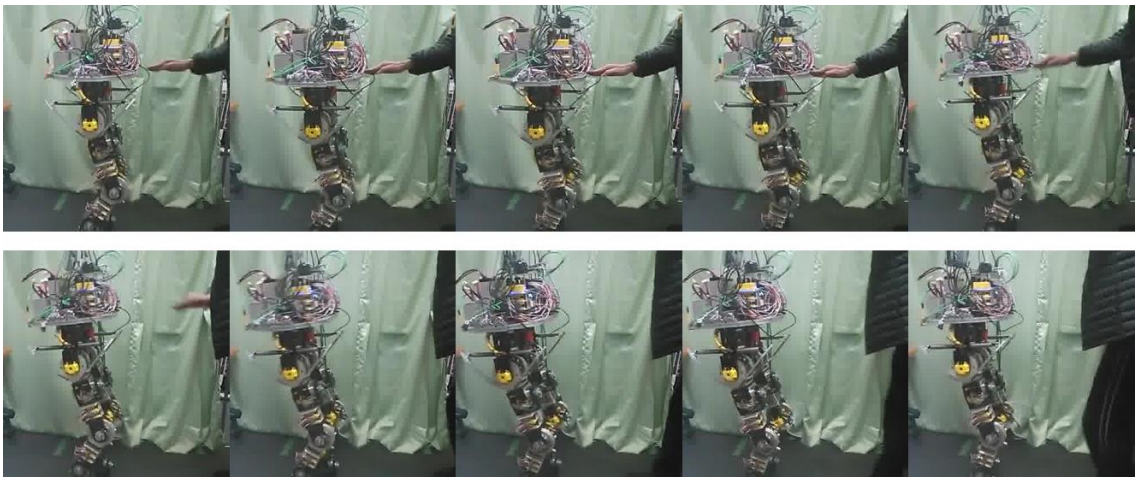


図 6.44. Snapshots of push recovery motion in real world.

具体的にはその場足踏み動作中にロボットに徐々に大きくなる外力を加えながらシミュレーションを行い転倒までにかかった時間を最大化することでより大きな外力に耐えられるような行動生成器を探索する．具体的に用いた手続きを以下にまとめる．

1. ロボットに三秒おきに四方向から順番に加わる力の大きさ f を零に初期化する．
2. 評価値 y を零に初期化する．
3. 歩行シミュレーションを五十秒行い転倒までにかかった時間 t を計算する．
4. t を y に加算し t が五十秒未満なら 6) へ移動する．
5. f の大きさを 30 Nm 大きくし，3) に戻る．
6. 評価値を出力する．

探索の結果力積 90 Nm × 0.1 s までの外力に耐えられる足踏み動作が生成された．図 6.44 に人間が足踏みをしているロボットを後ろに押したときの様子を示す．体幹が五度程度傾く程度に押しているが数歩後ろに下がることで体幹姿勢を回復していることが分かる．

6.5 おわりに

本章では前章で実装した動力学シミュレーションと進化計算を組み合わせた探索手法によって脚型ロボットの歩行補助機構のパラメタを最適化しつつ歩行行動も歩行行動生成する手法について実装と実験を行い，実脚型ロボットで歩行実現を行った．また本研究で開発した脚型ロボットのバネとクラッチを用いた歩行補助機構についてもまとめた．さらに環境の変化についてロバストな行動を探索する例として未知の外力が加わる状況で転倒しないように制御する行動生成器を探索する手法と実験を行った．

第 7 章

結論

7.1 成果と結論

人間により決定される探索の制約（仕様）が与えられた上でのロボット開発において、仕様から設計の身体運動モデルのパラメタが自明に決定されない問題や、実体の身体環境モデルのパラメタが設計通りに製造できない問題をそれぞれ支援する手法として、人間がモデルを与えそれに有効な探索手法を選択することで身体環境運動モデルのパラメタを探索することで自動化する身体行動創成支援システムを構築した。支援例として身体運動モデルのパラメタ生成、行動実現ともに難しい受動機械要素をもつ等身大脚型歩行ロボットを開発し短期間に行動実現を達成することで本支援システムが新しいロボットを創成する力を持つことを示した。本支援システムは受動軸やバネ、摩擦といった実体の身体環境モデルのパラメタが設計通りになることの難しい機械要素をもつロボットの行動実現、動作モデルのパラメタが身体モデルのパラメタと相互作用し仕様から設計に自明に変換できないような身体モデルのパラメタ生成を探索により自動化支援することでロボット開発を効率化しその自由度を広げるものである。

7.2 まとめと概要

- 背景と目的: 本論文ではロボット開発において、人間が手作業で行うことが難しい作業が存在するという背景のもと、そのような作業を探索問題として捉え、探索のための各種ツールとそれらを適切に使い分ける機構（実体設計評価機構）として、ロボット完成前の設計段階では目的となる運動について適する設計の身体モデルのパラメタを探索する問題、ロボット完成後の行動実現段階では実機の身体環境モデルのパラメタを推定し運動モデルのパラメタを再生する問題をそれぞれ解くことで、身体と行動の両方にたいしてその創成を支援する身体行動創成支援システムを構成することを目的とする。このような探索を扱うということは、運動モデルのパラメタだけを探索する従来研究から身体モデルのパラメタも探索するような問題へと拡張された探索問題を扱うということであり、シミュレータや非線形最適化手法といった様々なツールを有するとともに、それらを適切に組み合わせるようなシステムが必要となるものである。本論文の各章はその

ような異なるツールを用いた異なる問題設定を解くことで支援システムを評価する実験内容になっている。

- 前提と限界: 本研究ではロボット開発を四段階に分けて考えた。一段階目としてロボットには技術的な制約のない要求が発生する。二段階目として要求を人間が具体的な条件や制約として書き直す。三段階目として仕様を満たすように実体の作り方動かし方を一意に決める機械図面や制御器まで落とし込む。最後の四段階目として実際にロボットの完成品を用いて行動実現を試みていく。この全四段階において一段階目と二段階目は人間が行う工程であり支援システムでは扱わないとした。また三段階目と四段階目においても問題のモデルは与えられそのパラメタを探索する問題を扱った。この制約には二つの理由がある。一つは現実的に解ける規模の問題に絞るためという技術的な理由。もう一つは完成するロボットに人間が制約を加えることを可能にし、人間が望まないロボットが生成されることを防ぐことである。モデルも探索するような問題を考えることは可能である。例えばモデルが有限個であれば探索にかかる時間の制約を除けばすべてについて探索することは可能であるし、モデルの表現範囲を広げていけば評価関数である要求以外なしにロボットを生成することも小規模であれば可能である。しかしそのようにして生成されたロボットは人間の予想を裏切るものになる可能性がある。エネルギー効率を高めたいという要求の他に、車輪ではなく脚型ロボットに限らせるというような制約を与えることが支援システムの対象をモデルが与えられる問題に限る一番の理由である。
- 支援システムの特色: そのような限界のなかで支援システムは二つの特色を持っている。一つ目はロボット完成前後をともに支援する機構である。ロボット完成前の設計段階での身体運動モデルのパラメタ生成では仕様から決めるために探索が必要となるような設計の身体運動モデルのパラメタ生成を自動化することで支援する。ロボット完成後の行動実現においては完成した実体であるロボットの身体や、環境、運動の各パラメタである実体モデルパラメタを修正することで行動実現を支援する。二つ目は各機構における有効な探索アルゴリズムの提示である。本論文では接触の切り替わらない姿勢や姿勢列の探索に有効な勾配法アルゴリズムについて高速化手法を提案し三章にまとめた。四章では接触の切り替わる問題について接触状態遷移のグラフ探索と姿勢の勾配法探索を組み合わせる手法、動力学演算を利用した二手法についてまとめた。前者は接触を保ちながら移動する立ち上がり動作や着座動作といった問題に有効である。後者は制御系のパラメタや動的な接触切り替えといった問題に有効である。また実体モデルパラメタ修正手法について、センサデータから直接実体モデルパラメタが測定できる場合について四章、できない場合に測定可能値が一致するように実体モデルパラメタを探索する手法について五章、六章でまとめた。
- 適用例の意義：本論文で取り上げた支援例は、ケーススタディの側面を持つ一方で、先に述べた異なる特色を用いた例としてそれぞれ応用先が異なり意義のある問題になっている。三章ではリンク長を姿勢探索により探索した。ここでは股関節間距離を支援システムが決定してしまうわけではなく、各身体モデルのパラメタについて網羅的に運動モ

デルのパラメタを生成し可視化することで人間による身体モデルのパラメタ決定を支援する例となっている。このような探索が可能な理由は姿勢探索について有効であり高速な勾配法探索手法を支援システムが持つためであり、姿勢の探索など小規模な問題に限れば身体モデルのパラメタの影響を人間が一目で把握する支援が可能であることを示している。また探索した身体モデルのパラメタはリンクの距離でありロボットの外形や可動域を決める重要なパラメタであり応用例は広い。四章では接触点を滑りを用いて接触状態が切り替わる着座行動が可能となるように探索するとともに、滑り摩擦係数をセンサ値から計測し行動再生成を行うことでオンラインでの着座行動を実現した。この問題の一つ目の意義は環境との接触を保ち切り替えながら運動する一連の動作列すべてが実現可能であることを確認する支援例となっている点である。着座や立ち上がりでは全身自由度が複雑に運動するため少数の姿勢評価では行動が実現可能であるかを確認することができない。このような支援が可能な理由は支援システムが接触遷移を保ちながら運動する動作モデルのパラメタ生成に有効な接触グラフ探索と勾配法による姿勢探索法を持つためである。また実体モデルパラメタ推定としてもっとも単純な例としてセンサから実体身体モデルのパラメタが測定できる問題を扱うことで摩擦係数という、一般に設計時に実体化時の値を予測することが難しい実体身体モデルのパラメタ推定と運動モデルのパラメタ生成がオンラインで可能となることを示している。支援システムが滑りを伴う行動実現にも有効であることを示したことに意義がある。また探索した身体モデルのパラメタは接触点であり、多くのロボットが配線等の都合により全身での環境接触を扱えず、少数の接触してもよいリンクをもつ仕様になっているため応用例は広い。五章・六章では準受動歩行に必要なバネ係数と歩行運動モデルのパラメタ探索支援を行った。この問題の意義は動的で不安定な運動、設計通りに実体化が難しいバネという非剛体の実体モデルパラメタ推定にも支援システムが有効であることを示したことである。受動軸として股関節を用いる準受動歩行は歩き始めに膝から下を使って上半身の姿勢を一度崩し、バネの力を蓄えてから歩き始めるという歩行開始時、歩行時、歩行終了時でことなる体の使い方が必要である。さらにバネの力で振動する上半身は直接制御できず動力学に従い運動するため、正確な上半身の制御には全身の動力学モデル、接触モデルの推定が必要な例となっている。このような例が示せたことで制御不能なバネや受動軸といった受動機械要素をロボットの動的な運動に適用したことそのものに意義がある。

発表文献と研究活動

学術雑誌（筆頭）————————

(1) 野田 晋太朗, 野沢 峻一, 垣内 洋平, 岡田 慧, 稲葉 雅幸: 複数の接触遷移方式を統合する全身行動計画法とヒューマノイドの滑り接触遷移行動への応用, 日本ロボット学会誌, Vol.35, No.5, pp.393-402, 2017.

国際会議（筆頭）————————

(2) Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Redundancy Embedding for Search Space Reduction using Deep Auto-Encoder: Application to Collision-Free Posture Generation, in Proceedings of The 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016), pp.3698-3705, 2016.

(3) Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Contact involving whole-body behavior generation based on contact transition strategies switching, in Proceedings of The 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015), pp.2787-2794, 2015.

(4) Shintaro Noda, Masaki Murooka, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Generating Whole-body Motion Keep Away From Joint Torque, Contact Force, Contact Moment Limitations enabling Steep Climbing with a Real Humanoid Robot, in Proceedings of The 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), 2014.

(5) Shintaro Noda, Masaki Murooka, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Online maintaining behavior of high-load and unstable postures based on whole-body load balancing strategy with thermal prediction, in Proceedings of The 2014 IEEE International Conference on Automation Science and Engineering (CASE 2014), 2014.

受賞（筆頭）————————

(6) Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Redundancy Embedding for Search Space Reduction using Deep Auto-Encoder: Application to Collision-Free Posture Generation, IEEE Robotics and Automation Society Japan Chapter Young Award, IROS2016, 2016.10.

学術雑誌（共著）————————

(7) 小島 邦生, 唐澤 達史, 上月 豊隆, 黒岩 英則, 柚木崎 創, 岩石 智志, 石川 達矢, 小山 遼, 野田 晋太朗, 植田 亮平, 菅井 文仁, 野沢 峻一, 垣内洋平, 岡田 慧, 稲葉 雅幸: 高速大出力ヒューマノイドの研究用プラットフォーム JAXON の開発, 日本ロボット学会誌, Vol.34, No.7, pp.458-467, 2016.

(8) 室岡雅樹, 野田晋太朗, 野沢峻一, 垣内洋平, 岡田慧, 稲葉雅幸: 等身大ヒューマノイドにおける物体状態・操作力オンライン推定制御法に基づく大型重量物ピボット運搬行動の実現, 日本ロボット学会誌,Vol.32, No.7, 2013.

国際会議（共著）————————

(9) Shunichi Nozawa, Masaki Murooka, Shintaro Noda, Kunio Kojima, Yuta Kojio, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Unified Humanoid Manipulation of an Object of Unknown Mass Properties and Friction based on Online Constraint Estimation, in Proceedings of the 2017 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2017), pp.249–256, 2017.

(10) Moju Zhao, Koji Kawasaki, Xiangyu Chen, Shintaro Noda, Kei Okada, Masayuki Inaba: Whole-body aerial manipulation by transformable multirotor with two-dimensional multilinks, in Proceedings of The 2017 IEEE International Conference on Robotics and Automation (ICRA 2017), pp.5175–5182, 2017.

(11) Shunichi Nozawa, Shintaro Noda, Masaki Murooka, Kei Okada, Masayuki Inaba: Online Estimation of Object-Environment Constraints for Planning of Humanoid Motion on a Movable Object, in Proceedings of The 2017 IEEE International Conference on Robotics and Automation (ICRA 2017), pp.1291–1298, 2017.

(12) Ryo Terasawa, Shintaro Noda, Kunio Kojima, Ryo Koyama, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Achievement of Dynamic Tennis Swing Motion by Offline Motion Planning and Online Trajectory Modification Based on Optimization with a Humanoid Robot, in Proceedings of the 2016 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2016), pp.1094–1100, 2016.

(13) Kunio Kojima, Tatsuhi Karasawa, Toyotaka Kozuki, Eisoku Kuroiwa, Sou Yukizaki, Satoshi Iwaishi, Tatsuya Ishikawa, Ryo Koyama, Shintaro Noda, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Development of Life-Sized High-Power Humanoid Robot JAXON for Real-World Use, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), pp.838–843, 2015.

(14) Shunichi Nozawa, Eisoku Kuroiwa, Kunio Kojima, Ryohei Ueda, Masaki Murooka, Shintaro Noda, Iori Kumagai, Yu Ohara, Yohei Kakiuchi, Kei Okadad, Masayuki Inaba: Multi-Layered Real-Time Controllers for Humanoid's Manipulation and Locomotion Tasks with Emergency Stop, in Proceedings of the 2015 IEEE-RAS In-

ternational Conference on Humanoid Robots (Humanoids 2015), pp.381–388, 2015.

(15) Iori Kumagai , Ryo Terasawa , Shintaro Noda, Ryohei Ueda, Shunichi Nozawa , Yohei Kakiuchi , Kei Okada , Masayuki Inaba: Achievement of Recognition Guided Teleoperation Driving System for Humanoid Robots with Vehicle Path Estimation, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), pp.670–675, 2015.

(16) Yohei Kakiuchi, Kunio Kojima, Eisoku Kuroiwa, Masaki Murooka, Shintaro Noda, Iori Kumagai, Ryohei Ueda, Fumihito Sugai, Shunichi Nozawa, Kei Okada, Masayuki Inaba: Development of Humanoid Robot System for Disaster Response Through Team NEDO-JSK's Approach to DARPA Robotics Challenge Finals, in Proceedings of the 2015 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015), pp.805–810, 2015.

(17) Iori Kumagai, Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Whole Body Joint Load Reduction Control for High-Load Tasks of Humanoid Robot Through Adapting Joint Torque Limitation based on Online Joint Temperature Estimation, in Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014), pp.463-468, 2014.

(18) Masaki Murooka, Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Manipulation Strategy Decision and Execution Based on Strategy Proving Operation for Carrying Large and Heavy Objects, in Proceedings of The 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), 2014.

(19) Masaki Murooka, Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Manipulation Strategy Learning for Carrying Large Objects based on Mapping from Object Physical Property to Object Manipulation Action in Virtual Environment, in Proceedings of The 2014 IEEE International Conference on Automation Science and Engineering (CASE 2014), 2014.

(20) Shunichi Nozawa, Masaki Murooka, Shintaro Noda, Kei Okada, Masayuki Inaba: Description and Execution of Humanoid's Object Manipulation based on Object-environment-robot Contact States, in Proceedings of The 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013), pp. 2608-2615, 2013.

(21) Shunichi Nozawa, Masaki Murooka, Shintaro Noda, Yohei Kakiuchi,Kei Okada, Masayuki Inaba: Description and Execution of Manipulation and Locomotion by a Humanoid Robot Using Cascaded Contact States Graph, in Proceedings of the 2013 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2013), pp.492-498, 2013.

受賞（共著）——————

(22) Ryo Terasawa, Shintaro Noda, Kunio Kojima, Ryo Koyama, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Achievement of Dynamic Tennis Swing Motion by Offline Motion Planning and Online Trajectory Modification Based on Optimization with a Humanoid Robot, Mike Stilman Award, IEEE Humanoids2016, 2016.11.

(23) Masaki Murooka, Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, Masayuki Inaba: Manipulation Strategy Decision and Execution Based on Strategy Proving Operation for Carrying Large and Heavy Objects, Best Conference Paper Award, IEEE ICRA2014, 2014.6.

参考文献

[1] K. Kojima, T. Karasawa, T. Kozuki, E. Kuroiwa, S. Yukizaki, S. Iwaishi, T. Ishikawa, R. Koyama, S. Noda, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Development of life-sized high-power humanoid robot jaxon for real-world use. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pp. 838–843, Nov 2015.

[2] Defense Advanced Research Projects Agency. `http://www.theroboticschallenge.org/`. (last visited Jan 2015).

[3] Thomas Moulard, Florent Lamiraux, Karim Bouyarmane, and Eiichi Yoshida. RobOptim: an Optimization Framework for Robotics. In *Robomec*, p. 4p., Tsukuba, Japan, May 2013. http://www.jsme.or.jp/rmd/robomec2013/english/index.html.

[4] Russ Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2016.

[5] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, Vol. 19, No. 4, pp. 72–82, December 2012. `http://ompl.kavrakilab.org`.

[6] S. Chitta, I. Sucan, and S. Cousins. Moveit! *IEEE Robotics Automation Magazine*, Vol. 19, No. 1, pp. 18–19, March 2012.

[7] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-P. Laumond. Whole-body task planning for a humanoid robot: a way to integrate collision avoidance. In *2009 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 355–360, Dec 2009.

[8] Matthew Wall. Galib-a c++ genetic algorithms library, version 2.4, 1995.

[9] T. P. Runarsson and Xin Yao. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 35, No. 2, pp. 233–243, May 2005.

[10] T. P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp. 284–294, Sep 2000.

[11] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a

mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, Vol. 36, No. 4, pp. 193–202, 1980.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, Nov 1998.

[13] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313, No. 5786, pp. 504–507, 2006.

[14] Steven G. Johnson. The nlopt nonlinear-optimization package. `http://ab-initio.mit.edu`.

[15] 中川裕太. 評価関数と制約条件で表現される目標タスク空間からロボット状態空間への写像の解析と応用. Master's thesis, 東京大学大学院情報理工学系研究科知能機械情報学専攻, 2014.

[16] Sébastien Lengagne, Joris Vaillant, Eiichi Yoshida, and Abderrahmane Kheddar. Generation of Whole-body Optimal Dynamic Multi-Contact Motions. *The International Journal of Robotics Research*, Vol. 32, No. 9-10, pp. 1104–1119, 2013.

[17] Ryo Terasawa, Shintaro Noda, Kunio Kojima, Ryo Koyama, Fumihito Sugai, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Achievement of dynamic tennis swing motion by offline motion planning and online trajectory modification based on optimization with a humanoid robot. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 1094–1100, Nov 2016.

[18] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick. Real-time collision avoidance with whole body motion control for humanoid robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2053–2058, Oct 2007.

[19] A. Escande, S. Miossec, and A. Kheddar. Continuous gradient proximity distance for humanoids free-collision optimized-postures. In *2007 7th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 188–195, Nov 2007.

[20] S. Noda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Redundancy embedding for search space reduction using deep auto-encoder: Application to collision-free posture generation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3698–3705, Oct 2016.

[21] Wataru Takano, Hirotaka Imagawa, and Yoshihiko Nakamura. Spatio-temporal structure of human motion primitives and its application to motion prediction. *Robotics and Autonomous Systems*, Vol. 75, Part B, pp. 288 – 296, 2016.

[22] Katsu Yamane, Yoshifumi Yamaguchi, and Yoshihiko Nakamura. Human motion database with a binary tree and node transition graphs. *Autonomous Robots*, Vol. 30, No. 1, pp. 87–98, 2010.

[23] U. Nagarajan and K. Yamane. Automatic task-specific model reduction for humanoid robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2578–2585, Nov 2013.

[24] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, Vol. 27, No. 1, pp. 1–33, 1983.

[25] G. Guennebaud and L. D. Gaspero. eiquadprog.hpp. `http://www.diegm.uniud.it/digaspero/index.php/software`, `https://www.cs.cmu.edu/~bstephe1/eiquadprog.hpp`. (visited Aug 2016).

[26] S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2589–2594, May 2014.

[27] Timothy Wolfe Bretl. *Multi-step Motion Planning: Application to Free-climbing Robots*. PhD thesis, Stanford University, Stanford, CA, USA, 2005.

[28] Tim Bretl, Jean Claude Latombe, and Stephen Rock. Toward autonomous free-climbing robots. *Springer Tracts in Advanced Robotics*, Vol. 15, pp. 6–15, 2005.

[29] Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. Motion planning for legged robots on varied terrain. *The International Journal of Robotics Research*, Vol. 27, No. 11-12, pp. 1325–1349, 2008.

[30] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, Vol. 30, No. 6, pp. 678–698, 2011.

[31] Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, Vol. 61, No. 5, pp. 428 – 442, 2013.

[32] A. Escande, A. Kheddar, and S. Miossec. Planning support contact-points for humanoid robots and experiments on hrp-2. In *International Conference on Intelligent Robots and Systems, 2006 IEEE/RSJ*, pp. 2974–2979, Oct.

[33] K. Bouyarmane, J. Vaillant, F. Keith, and A. Kheddar. Exploring humanoid robot locomotion capabilities in virtual disaster response scenarios. In *Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 1099 – 1126, 2012.

[34] Karim Bouyarmane and Abderrahmane Kheddar. Humanoid robot locomotion and manipulation step planning. *Advanced Robotics*, pp. 1099–1126, 2012.

[35] Shintaro Noda, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Contact involving whole-body behavior generation based on contact transition strategies switching. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, pp. 2787–2794, Oct 2015.

[36] 野田晋太朗, 野沢峻一, 垣内洋平, 岡田慧, 稲葉雅幸. 複数の接触遷移方式を統合する全身行動計画法とヒューマノイドの滑り接触遷移行動への応用. 日本ロボット学会誌, Vol. 35, No. 5, pp. 393–402, 2017.

[37] Russell Smith. ODE: Open Dynamics Library. `http://www.ode.org/`. (last visited Jan 2015).

[38] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, Vol. 3, pp. 2149–2154 vol.3, Sept 2004.

[39] Gazebo. `http://gazebosim.org`. (last visited Jan 2015).

[40] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graph*, Vol. 31, No. 4, pp. 43:1–43:8, jul 2012.

[41] M. Al Borno, M. de Lasa, and A. Hertzmann. Trajectory optimization for full-body movements with complex contacts. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 19, No. 8, pp. 1405–1414, Aug 2013.

[42] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. In *ACM Transactions on Graphics (TOG)*, Vol. 26, p. 105. ACM, 2007.

[43] Toshihiro Matsui and Masayuki Inaba. Euslisp: an object-based implementation of lisp. *Journal of Information Processing*, Vol. 13, No. 3, pp. 327–338, 1990.

[44] euslisp/jskeus. `https://github.com/euslisp/jskeus`. (last visited Nov 2015).

[45] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, Vol. abs/1408.5093, , 2014.

[46] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 2, pp. 1620–1626 vol.2, Sept 2003.

[47] Siyuan Feng, X Xinjilefu, Weiwei Huang, and Christopher G Atkeson. 3d walking based on online optimization. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 21–27. IEEE, 2013.

[48] Shintaro Noda, Masaki Murooka, Shunichi Nozawa, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Generating whole-body motion keep away from joint torque, contact force, contact moment limitations enabling steep climbing with a real humanoid robot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1775–1781, 2014.

[49] M. Murooka, S. Noda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Manipulation strategy decision and execution based on strategy proving operation for carrying

large and heavy objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3425–3432, May 2014.

[50] J. Tan, Z. Xie, B. Boots, and C. K. Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2729–2736, Oct 2016.

[51] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 15–22. ACM, 1994.

[52] M. Azad, V. Ortenzi, H. C. Lin, E. Rueckert, and M. Mistry. Model estimation and control of compliant contact normal force. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 442–447, Nov 2016.

[53] Kenneth Hunt and Erskine Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics*, 1975.

[54] 鮎澤光, 中村仁彦ほか. ベースリンクの運動方程式を利用した脚型ロボットの最小力学パラメータの同定. 日本ロボット学会誌, Vol. 27, No. 9, pp. 1066–1077, 2009.

[55] 鮎澤光, 中村仁彦ほか. ベースリンクの非駆動性を利用した浮遊リンク系の力学同定法. 日本ロボット学会誌, Vol. 28, No. 8, pp. 1004–1013, 2010.

[56] Ko Ayusawa, Gentiane Venture, and Yoshihiko Nakamura. Identifiability and identification of inertial parameters using the underactuated base-link dynamics for legged multibody systems. *The International Journal of Robotics Research*, Vol. 33, No. 3, pp. 446–468, 2014.

[57] 寺澤良. 最適化計算に基づく等身大ヒューマノイドのダイナミックな全身高速動作生成に関する研究. Master's thesis, 東京大学大学院情報理工学系研究科知能機械情報学専攻, 2017.

[58] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 2014.

[59] Sachin Chitta, Ioan Alexandru Sucan, and Steve Cousins. Moveit! [ROS topics]. *IEEE Robotics Automation Magazine*, Vol. 19, No. 1, pp. 18–19, 2012.

[60] Raşit Köker, Tarık Çakar, and Yavuz Sari. A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. *Engineering with Computers*, Vol. 30, No. 4, pp. 641–649, 2014.

[61] Mihai Duguleana, Florin Grigore Barbuceanu, Ahmed Teirelbar, and Gheorghe Mogan. Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning. *Robotics and Computer-Integrated Manufacturing*, Vol. 28, No. 2, pp. 132 – 146, 2012.

[62] F. Zacharias, W. Sepp, C. Borst, and G. Hirzinger. Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In *2009 International Conference on Humanoid Robots (Humanoids)*, pp. 55–61, Dec 2009.

[63] Nikolaus Vahrenkamp and Tamim Asfour. Representing the robot's workspace through constrained manipulability analysis. *Autonomous Robots*, Vol. 38, No. 1, pp. 17–30, 2015.

[64] K. Yamane. Systematic derivation of simplified dynamics for humanoid robots. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 28–35, Nov 2012.

[65] Kazuhiko AKACHI. Takakatsu ISOZUMI. Masaru HIRATA. Sigehiko OHTA. Masakazu ISHIZAKI. Development of the humanoid robot, hrp-2 (in japanese). *Kawada technical report*, Vol. 23, pp. 20–25, 2004.

[66] K. Okada, T. Ogura, A. Haneda, J. Fujimoto, F. Gravot, and M. Inaba. Humanoid motion generation system on HRP2-JSK for daily life environment. In *Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation (ICMA)*, Vol. 4, pp. 1772–1777, July 2005.

[67] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, Vol. 27 (2), pp. 372–376, 1983.

[68] 原田研介, 梶田秀司, 金広文男, 藤原清司, 金子健二, 横井一仁, 比留川博久. ヒューマノイドロボットの脚腕協調における zmp 解析. 日本ロボット学会誌, Vol. 22, No. 1, pp. 28–36, 2004.

[69] K. Hauser. Fast dynamic optimization of robot paths under actuator limits and frictional contact. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2990–2996, May 2014.

[70] Joris Vaillant, Abderrahmane Kheddar, Herve Audren, Francois Keith, Stanislas Brossette, Kenji Kaneko, Mitsuharu Morisawa, Eiichi Yoshida, and Fumio Kanehiro. Vertical ladder climbing by the hrp-2 humanoid robot. In *Proceedings of the 2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 463–468, 2014.

[71] K. Bouyarmane and A. Kheddar. Static multi-contact inverse problem for multiple humanoid robots and manipulated objects. In *Proceedings of the 2010 International Conference on Humanoid Robots (Humanoids 2010)*, pp. 8–13, Dec 2010.

[72] Kris Hauser, Tim Bretl, and Jean-Claude Latombe. Non-gaited humanoid locomotion planning. In *Proceedings of the 2005 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2005)*, pp. 7–12, Dec 2005.

[73] C. Ott, M.A. Roa, and G. Hirzinger. Posture and balance control for biped robots based on contact force optimization. In *Proceedings of the 2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 26–33, Oct 2011.

[74] K. Nagasaka, Y. Kawanami, S. Shimizu, T. Kito, T. Tsuboi, A. Miyamoto, T. Fukushima, and H. Shimomura. Whole-body cooperative force control for a

two-armed and two-wheeled mobile robot using generalized inverse dynamics and idealized joint units. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3377–3383, May 2010.

[75] Masao Kanazawa, Shunichi Nozawa, Yohei Kakiuchi, Yoshiki Kanemoto, Mitsuhide Kuroda, Kei Okada, Masayuki Inaba, and Takahide Yoshiike. Robust vertical ladder climbing and transitioning between ladder and catwalk for humanoid robots. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, pp. 2202–2009, Oct 2015.

[76] L. Sentis, Jaeheung Park, and O. Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on Robotics*, Vol. 26, No. 3, pp. 483–501, June 2010.

[77] 梶田秀司. ヒューマノイドロボット. オーム社, 2005.

[78] 内山勝, 中村仁彦. 岩波講座ロボット学 2 ロボットモーション. 岩波書店, 2004.

[79] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, Vol. 108, No. 3, pp. 163–171, 1986.

[80] Jingru Luo, Yajia Zhang, K. Hauser, H.A. Park, M. Paldhe, C.S.G. Lee, M. Grey, M. Stilman, Jun Ho Oh, Jungho Lee, Inhyeok Kim, and P. Oh. Robust ladder-climbing with a humanoid robot with application to the darpa robotics challenge. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2792–2798, May 2014.

[81] Dieter Kraft. Algorithm 733; TOMP - fortran modules for optimal control calculations. *ACM Trans. Math. Softw.*, Vol. 20, No. 3, pp. 262–281, 1994.

[82] A. Hourtash. The kinematic hessian and higher derivatives. In *Proceedings of the 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 2005)*, pp. 169–174, June 2005.

[83] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Resolved momentum control: humanoid motion planning based on the linear and angular momentum. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Vol. 2, pp. 1644–1650 vol.2, Oct 2003.

[84] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi. Design of prototype humanoid robotics platform for hrp. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 3, pp. 2431–2436 vol.3, 2002.

[85] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, Vol. 4, No. 2, pp. 3–9, 1985.

[86] 野田晋太朗. ヒューマノイドにおける接触状態探索機能を備えた全身行動生成に関する

研究. Master's thesis, 東京大学大学院情報理工学系研究科創造情報学専攻, 2015.

[87] 相山康道, 稲葉雅幸, 井上博允. グラスプレス・マニピュレーションの研究: 操作形態の分類とピボット操作の実現. 日本ロボット学会誌, Vol. 14, No. 1, pp. 114–121, 1996.

[88] 室岡雅樹, 野田晋太朗, 野沢峻一, 垣内洋平, 岡田慧, 稲葉雅幸. 等身大ヒューマノイドにおける物体状態・操作力オンライン推定制御法に基づく大型重量物ピボット運搬行動の実現. 日本ロボット学会誌, Vol. 32, No. 7, pp. 595–602, 2014.

[89] 沢崎直之, 井上博允. 多指ハンドによる物体の転がし操作. 日本ロボット学会誌, Vol. 9, No. 5, pp. 560–571, 1991.

[90] 三浦郁奈子, 中岡慎一郎, 金広文男, 原田研介, 金子健二, 横井一仁, 梶田秀司. 足裏の滑りを利用した 2 足歩行ロボットの方向転換：滑り現象のモデル化と回転角の予測. 日本ロボット学会誌, Vol. 28, No. 10, pp. 1232–1242, dec 2010.

[91] Yoshito Ito. Takuya Nakaoka. Junichi Urata. Kazuya Kobayashi. Shunich Nozawa. Yuto Nakanishi. Kei Okada and Masayuki Inaba. Design and development of high-output life-size humanoid with water-cooled motor and motor driver. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, pp. 3433–3438, 2014.

[92] Kunio Kojima, Shunichi Nozawa, Kei Okada, and Masayuki Inaba. Shuffle motion for humanoid robot by sole load distribution and foot force control. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, 2015.

[93] Roy Featherstone. *Rigid Body Dynamics Algorithms.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[94] Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *ACM SIGGRAPH*, 1996.

[95] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. In *IEEE International Conference on Robotics and Automation (ICRA). IEEE*, 1999.

[96] 小島邦生, 唐澤達史, 上月豊隆, 黒岩英則, 柚木崎創, 岩石智志, 石川達矢, 小山遼, 野田晋太朗, 植田亮平, 菅井文仁, 野沢峻一, 垣内洋平, 岡田慧, 稲葉雅幸. 高速大出力ヒューマノイドの研究用プラットフォーム jaxon の開発. 日本ロボット学会誌, Vol. 34, No. 7, pp. 458–467, 2016.

[97] 唐澤達史. 脚の内外旋による転倒防止動作を利用したヒューマノイドの外乱対応歩行制御に関する研究. Master's thesis, 東京大学大学院情報理工学系研究科知能機械情報学専攻, 2016.

[98] SampleRobot. `https://github.com/fkanehiro/hrpsys-base/tree/master/sample/SampleRobot`. (last visited October 2017).

[99] T. Erez, Y. Tassa, and E. Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE International*

Conference on Robotics and Automation (ICRA), pp. 4397–4404, May 2015.

[100] Ayonga Hereid, Eric A Cousineau, Christian M Hubicki, and Aaron D Ames. 3d dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *IEEE International Conference on Robotics and Automation (ICRA). IEEE*, 2016.

[101] Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological cybernetics*, Vol. 52, No. 6, pp. 367–376, 1985.

[102] Gen Endo, Jun Nakanishi, Jun Morimoto, and G. Cheng. Experimental studies of a neural oscillator for biped locomotion with qrio. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 596–602, April 2005.

[103] L. Righetti and Auke Jan Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1585–1590, May 2006.

[104] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot. *The International Journal of Robotics Research*, Vol. 27, No. 2, pp. 213–228, 2008.

[105] B. G. Buss, A. Ramezani, K. Akbari Hamed, B. A. Griffin, K. S. Galloway, and J. W. Grizzle. Preliminary walking experiments with underactuated 3d bipedal robot marlo. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2529–2536, Sept 2014.

[106] Martijn Wisse, Arend L Schwab, and Frans CT van der Helm. Passive dynamic walking model with upper body. *Robotica*, Vol. 22, No. 06, pp. 681–688, 2004.

[107] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, June 2011.

[108] J. Reher, E. A. Cousineau, A. Hereid, C. M. Hubicki, and A. D. Ames. Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1794–1801, May 2016.

[109] 菅井文仁, 小島邦生, 野沢峻一, 垣内洋平, 岡田慧, 稲葉雅幸. 高速大出力ヒューマノイドのための超小型大出力モータドライバの開発. 第 34 回日本ロボット学会学術講演会講演論文集, pp. 1X2–01, Sept 2016.

[110] マクソンモータ. ブラシレス dc モータ maxon ec-4pole 30 200w. `https://www.maxonjapan.co.jp/maxon/view/content/ec-4pole-motors`. (last visited September 2017).

[111] J. A. Nelder and R. Mead. A simplex method for function minimization. *The*

Computer Journal, Vol. 7, No. 4, pp. 308–313, 1965.

[112] Intel Corporation. ミニ pc インテル nuc キット nuc6i7kyk. https://www.intel.co.jp/content/www/jp/ja/nuc/nuc-kit-nuc6i7kyk-features-configurations.html. (last visited September 2017).

[113] 株式会社ワコーテック. 静電容量型 6 軸力覚センサ (500n 品). http://www.wacoh-tech.com/products/dynpick/200n_500n_rcdb.html. (last visited September 2017).

[114] VectorNav Technologies. Vn-100 imu/ahrs. https://www.vectornav.com/products/vn-100. (last visited September 2017).

[115] ハイデンハイン株式会社. Heidenhain eci 1118 servo drive rotary encoder. https://www.heidenhain.co.jp/fileadmin/pdb/media/img/349529-J9.pdf. (last visited September 2017).

[116] 株式会社ハーモニックドライブシステムズ. Shd シリーズ 簡易ユニットタイプ. https://www.hds.co.jp/products/lineup/hd/01sr18_shd_2sh/. (last visited September 2017).

[117] 三木プーリ株式会社. 無励磁動作ブレーキ bxr le シリーズ. http://www.mikipulley.co.jp/JP/Products/ElectoromagneticClutchesAndBrakes/SpringActuatedTypeBrakes/BXR-LE/index.html. (last visited September 2017).

[118] Y. Kakiuchi, K. Kojima, E. Kuroiwa, S. Noda, M. Murooka, I. Kumagai, R. Ueda, F. Sugai, S. Nozawa, K. Okada, and M. Inaba. Development of humanoid robot system for disaster response through team nedo-jsk's approach to darpa robotics challenge finals. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 805–810, Nov 2015.

[119] 株式会社ミスミグループ. Misumi-vona ｜ミスミの総合 web カタログ. https://jp.misumi-ec.com. (last visited September 2017).

[120] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and Woo-Keun Yoon. Rt-middleware: distributed component middleware for rt (robot technology). In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3933–3938, Aug 2005.

[121] hrpsys base. Basic rt components and utilities to control robots using openrtm. https://github.com/fkanehiro/hrpsys-base. (last visited September 2017).

[122] Y. Ikemata, A. Sano, K. Yasuhara, and H. Fujimoto. Dynamic effects of arc feet on the leg motion of passive walker. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 2755–2760, May 2009.

[123] T. McGeer. Passive dynamic walking. *The international journal of robotics research*, Vol. 9, No. 2, pp. 62–82, 1990.

[124] Steven H Collins, Martijn Wisse, and Andy Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of*

Robotics Research, Vol. 20, No. 7, pp. 607–615, 2001.

[125] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent asimo: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Vol. 3, pp. 2478–2483. IEEE, 2002.

[126] Q. Li, G. Liu, J. Tang, and J. Zhang. A simple 2d straight-leg passive dynamic walking model without foot-scuffing problem. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5155–5161, Oct 2016.

[127] Martijn Wisse, Guillaume Keliksdal, Jan Van Frankenhyyzen, and Brian Moyer. Passive-based walking robot. *IEEE Robotics & Automation Magazine*, Vol. 14, No. 2, pp. 52–62, 2007.

[128] F. Zonfrilli, D. Wollherr, and Y. Nakamura. Walking control of the humanoid ut-theta. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pp. 698–704, July 2005.

[129] K. Koganezawa and O. Matsumoto. Active/passive hybrid walking by the biped robot tokai robo-habilis 1. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2461–2466 vol.3, 2002.

[130] Q. Huang, T. Hase, and K. Ono. Passive/active unified dynamic walking for biped locomotion. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 964–971, Dec 2007.

[131] MIKI PULLEY. Electromagnetic tooth clutches. `http://www.mikipulley.co.jp/data/pdf/jp/cb_ts_ct.pdf`. (last visited April 2016).

[132] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.

[133] 池俣吉人, 佐野明人, 加藤良樹. 股関節バネ-ダンパ機構を用いて上体を付加した受動歩行の解析. 日本機械学会論文集C編, Vol. 78, No. 796, pp. 3959–3969, 2012.

[134] 双葉電子工業株式会社. コマンド方式サーボ rs304md. `http://www.futaba.co.jp/robot/command_type_servos/rs304md`. (last visited September 2017).

謝辞

本論文は筆者が東京大学情報理工学系研究科創造情報学専攻博士課程に在学中、情報システム工学研究室 (JSK) において、稲葉雅幸教授の御指導のもとで執筆したものです.

ご指導を頂いた稲葉雅幸教授はじめ日頃多くの知識と示唆を与えてくださった情報システム工学研究室の皆様に深くお礼申し上げます。また中村仁彦教授、國吉康夫教授、千葉滋教授、岡田慧准教授、中山英樹講師には本論文をまとめるにあたり非常に有意義な多くのご意見を頂きましたこと感謝致します。最後に、私に関わってくださった全ての方々に最大限の感謝を捧げ謝辞とさせていただきます。

2017 年 12 月 8 日 野田晋太朗

付録： 開発したロボットの組み立て図