Receding-Horizon Motion Planning of Quadrupedal Robot Locomotion



Oliwier Aleksander Melon

Mansfield College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

Declaration

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Oliwier Aleksander Melon, Mansfield College

Acknowledgements

I am deeply grateful to my co-supervisors, Prof. Maurice Fallon and Prof. Ioannis Havoutis, for giving me the unique opportunity, extensive resources and guidance to explore, learn and work on exciting research problems.

To my examiners, Dr Nicolas Mansard and Prof. Paul Goulart, thank you for your time, expertise and constructive feedback which helped me improve this thesis. I would also like to thank my ToS and CoS assessors, Prof. Nick Hawes and Prof. Steve Roberts, for their insightful advice which helped shape this work.

Heartfelt thanks to the ORI administrators, engineers, academics, researchers and students whom I had the pleasure of working with—especially to the DRS group and my fellow DPhil student, Yiduo Wang, for sharing the ups and downs of this journey with me. I was extremely fortunate to have collaborated with Dr Mathieu Geisert, Dr David Surovik, Dr Romeo Orsolino and Dr Wolfgang Merkt. By virtue of their encouragement, support and contributions—including countless hours spent together discussing, planning, implementing, testing and documenting the research—I was able to reach this point.

Mansfield College, its staff and students have been the source of wonderful memories. It was an honour to serve as MCR President in 2020–2021 and a pleasure working with many like-minded people over the years. Thank you to my rowing friends at MCBC for all the great times on and off the river.

I could not have completed these studies without my friends and family. I wish to thank my housemates for their company, especially Alexander Cliffe and Armands Strikis with whom I lived for a long time and share many good memories. I am so thankful to my partner, Ella Buys, for her unwavering love and invaluable support, and for being a world of difference during Covid and caring for me during the most testing periods. Special thanks to Alex and Ella for helping me review this thesis. Lastly, I am eternally grateful to my mother, Jowita Melon, for always doing everything she can to help me overcome any hurdles and achieve my goals.

Funding: My gratitude also goes to UKRI EPSRC, EU Horizon 2020 projects THING and MEMMO, and Moog Inc. for the provision of financial support which made this research possible.

Abstract

Quadrupedal robots are designed to offer efficient and robust mobility on uneven terrain. This thesis investigates combining numerical optimization and machine learning methods to achieve interpretable kinodynamic planning of natural and agile locomotion.

The proposed algorithm, called Receding-Horizon Experience-Controlled Adaptive Legged Locomotion (RHECALL), uses nonlinear programming (NLP) with learned initialization to produce long-horizon, high-fidelity, terrain-aware, whole-body trajectories. RHECALL has been implemented and validated on the ANYbotics ANYmal B and C quadrupeds on complex terrain.

The proposed optimal control problem formulation uses the single-rigid-body dynamics (SRBD) model and adopts a direct collocation transcription method which enables the discovery of aperiodic contact sequences. To generate reliable trajectories, we propose fast-to-compute analytical costs that leverage the discretization and terrain-dependent kinematic constraints.

To extend the formulation to receding-horizon planning, we propose a segmentation approach with asynchronous centre of mass (COM) and end-effector timings and a heuristic initialization scheme which reuses the previous solution. We integrate real-time 2.5D perception data for online foothold selection. Additionally, we demonstrate that a learned stability criterion can be incorporated into the planning framework.

To accelerate the convergence of the NLP solver to locally optimal solutions, we propose data-driven initialization schemes trained using supervised and unsupervised behaviour cloning. We demonstrate the computational advantage of the schemes and the ability to leverage latent space to reconstruct dynamic segments of plans which are several seconds long.

Finally, in order to apply RHECALL to quadrupeds with significant leg inertias, we derive the more accurate lump leg single-rigid-body dynamics (LL-SRBD) and centroidal dynamics (CD) models and their first-order partial derivatives. To facilitate intuitive usage of costs, constraints and initializations, we parameterize these models by Euclidean-space variables. We show the models have the ability to shape rotational inertia of the robot which offers potential to further improve agility.

Contents

List of Figures . . . . . **ix**

List of Tables . . . . . **x**

List of Abbreviations . . . . . **xi**

List of Symbols . . . . . **xv**

1 Introduction **1**
- 1.1 Context and motivation 1
- 1.2 Aims and scope 4
- 1.3 Outline 4
- 1.4 Contributions 7
- 1.5 Publications 9

2 Background **11**
- 2.1 Introduction 11
- 2.2 Overview of legged robots 12
  - 2.2.1 Modern robots and their notable predecessors 12
  - 2.2.2 Onboard sensors 16
  - 2.2.3 Challenges 17
  - 2.2.4 Assumptions 19
- 2.3 Modelling of legged robots 20
  - 2.3.1 Rigid-body dynamics model 21
  - 2.3.2 Centroidal dynamics model 23
  - 2.3.3 Single-rigid-body dynamics model 24
  - 2.3.4 Linear inverted pendulum 24
- 2.4 Motion planning 25
  - 2.4.1 Heuristics 25
  - 2.4.2 Sampling-based planning 26
  - 2.4.3 Trajectory optimization 27
  - 2.4.4 Offline planning 37
  - 2.4.5 Online replanning 39

2.5 Motion learning . . . 41
2.5.1 Reinforcement learning . . . 42
2.5.2 Locomotion using reinforcement learning . . . 43
2.5.3 Inverse reinforcement learning . . . 44
2.5.4 Data-driven optimal control . . . 44
2.6 Motion control . . . 47
2.6.1 Virtual-model control . . . 48
2.6.2 Inverse-dynamics control . . . 48
2.6.3 Whole-body control . . . 49
2.6.4 Low-level control . . . 49
2.7 Conclusion . . . 50

3 Costs and constraints for reliable trajectories . . . 52
3.1 Introduction . . . 53
3.1.1 Contributions . . . 53
3.2 Related works . . . 54
3.3 Methodology . . . 55
3.3.1 Problem formulation . . . 55
3.3.2 Analytical costs and kinematic constraints . . . 59
3.3.3 Whole-body control . . . 65
3.4 Results . . . 65
3.4.1 Validation in simulation . . . 66
3.4.2 Validation on hardware . . . 67
3.5 Discussion . . . 71
3.5.1 Computational challenges . . . 71
3.5.2 Continuous locomotion . . . 71
3.5.3 Perception . . . 71
3.6 Conclusion . . . 72

4 Receding-horizon adaptive trajectory optimization . . . 73
4.1 Introduction . . . 74
4.1.1 Contributions . . . 75
4.2 Related works . . . 76
4.2.1 Motion planning via model-predictive control . . . 76
4.3 Methodology . . . 77
4.3.1 Segmentation-based problem formulation . . . 77
4.3.2 Heuristic initialization . . . 79
4.3.3 State prediction due to the optimization delay . . . 80
4.3.4 Terrain perception . . . 82

4.3.5 Whole-body controller and robustness to uncertainty in perception . . . 82
4.3.6 Locomotion replanning using a dynamic stability criterion . 83
4.4 Results . . . 85
4.4.1 Heuristic initialization . . . 85
4.4.2 Learned stability margin . . . 87
4.5 Discussion . . . 88
4.6 Conclusion . . . 90

5 Data-driven solver initialization . . . 91
5.1 Introduction . . . 92
5.1.1 Contributions . . . 93
5.2 Related works . . . 93
5.2.1 Data-driven initialization . . . 93
5.2.2 Data-driven locomotion . . . 94
5.3 Methodology . . . 95
5.3.1 Learned initialization of an optimization solver . . . 96
5.3.2 Multimodal imitation learning . . . 99
5.4 Results . . . 103
5.4.1 Open-loop planning using LearnedInit . . . 103
5.4.2 Receding-horizon replanning using LMTR . . . 106
5.5 Discussion . . . 109
5.6 Conclusion . . . 111

6 Dynamics models for inertia shaping . . . 113
6.1 Introduction . . . 114
6.1.1 Contributions . . . 114
6.2 Related works . . . 114
6.3 Methodology . . . 116
6.3.1 Preliminaries . . . 116
6.3.2 Direct collocation . . . 120
6.3.3 Lump leg single rigid body dynamics . . . 121
6.3.4 Centroidal dynamics in Euclidean space . . . 126
6.4 Results . . . 134
6.5 Discussion . . . 137
6.6 Conclusion . . . 138

7 Conclusion . . . 140
7.1 Concluding summary . . . 140
7.2 Future work . . . 143

7.2.1 Leg inertia compensation . . . 143
7.2.2 Faster receding-horizon planning . . . 143
7.2.3 Improved learned initialization . . . 144
7.2.4 Reactive policy via inverse reinforcement learning . . . 144
7.2.5 Three-dimensional environment representation . . . 145

A Appendices . . . 146
A.1 Analytical costs and their partial derivatives . . . 147
A.2 Examples of superquadrics . . . 149
A.3 Terrain reconstruction for realistic simulation . . . 151
A.4 Computation time . . . 152
A.4.1 Discussion of the effects of TOWR's architecture . . . 152
A.4.2 Comparison of linear solvers . . . 153
A.4.3 Reordering of the Jacobian matrix . . . 153
A.4.4 Analysis of the timing statistics of IPOPT . . . 154
A.5 Analysis of LMTR results . . . 157
A.5.1 Analysis of mode selection . . . 157
A.5.2 Analysis of mode dimensionality . . . 157
A.6 Quaternion and its partial derivatives . . . 160

References . . . 162

List of Figures

1.1 Potential for quadrupeds (detecting nuclear radiation) 2
1.2 Biological inspiration (dog agility competition) 3
1.3 ANYmal B executing a long-horizon trajectory 7
1.4 The RHECALL framework being deployed on ANYmal C 8

2.1 Examples of humanoid robots 14
2.2 Examples of quadrupedal robots 15
2.3 Examples of onboard sensors used by legged robots 17
2.4 Diagrams of popular dynamics models 20
2.5 Types of transcription methods 32
2.6 An overview of optimal control methods 36
2.7 Robot model and phase-based parameterization in TOWR 39

3.1 Examples of optimized splines 58
3.2 Heightmap processing for foothold selection 62
3.3 Superquadrics as end-effector range-of-motion constraints 62
3.4 Terrain-dependent base pose constraint concept 63
3.5 Effects of analytical costs on optimized trajectories 66
3.6 Terrain-dependent base pose constraint tested in simulation 67
3.7 Snapshots of the execution of a dynamic trajectory 69
3.8 Terrain-dependent base pose constraint tested on hardware 70

4.1 ANYmal C using perception to cross a 20 cm step in real time . . . 75
4.2 The system for closed-loop replanning using Heuristic initialization 75
4.3 Segmentation-based direct collocation for RHC of legged robots . . 78
4.4 Receding-horizon Heuristic initialization 80
4.5 Prediction of the initial state 81
4.6 Analytical and learned stability margins 83
4.7 Replanning on uneven terrain using Heuristic 86
4.8 Pose tracking errors during a 5.5 min execution 86
4.9 The replanner using the learned stability margin on uneven terrain . 89

5.1 The system for open-loop planning using LearnedInit initializations 93

5.2 The system for closed-loop replanning using LMTR initialization . 93
5.3 Solver initialization using learned optimized solutions 96
5.4 The distinction between stationary and receding horizons 99
5.5 A diagram of bulk data generation for imitation learning 101
5.6 Diagrams of CVAE training and LMTR deployment 103
5.7 Optimization performance under different initialization schemes . . 104
5.8 Execution of trajectories initialized by LearnedInit in Gazebo 105
5.9 Execution performance under different initialization schemes 106
5.10 Snapshots of LMTR numerical trial on random steps 107
5.11 Cost and infeasibility during refinement of LMTR initializations . . 108
5.12 Mean cost and infeasibility during refinement 108
5.13 Online adaptation of LMTR initialization on flat ground 109
5.14 Online adaptation of LMTR initialization to uneven terrain 109

6.1 A 3D view of a function of third-order polynomials. 121
6.2 Key components of the lump leg single-rigid-body dynamics model . 122
6.3 The sparsity pattern of the Jacobian matrix of $\mathbf{q}$, $\mathbf{v}$ and $\dot{\mathbf{v}}$ 130
6.4 A comparison of equimomental ellipsoids of LL-SRBD and CD . . . 136

A.1 Examples of superquadrics 149
A.2 Superquadric and box intersection 150
A.3 Terrain reconstruction for more realistic simulation 151
A.4 The constraint Jacobian matrix sparsity patterns 154
A.5 Visualization of expert modes in the latent space 158
A.6 A comparison of a unimodal regressor, LMTR and TL 159

List of Tables

1.1 A chronological list of publications. 10

3.1 Optimization variables in TOWR 57
3.2 Computation time per solver iteration 70

6.1 A comparison of rotational inertia of LL-SRBD and CD 135
6.2 A comparison of rotational inertia of LL-SRBD and CD (%) 137

A.1 The effects of linear solvers on computation time 153
A.2 The effects of base-motion horizons on computation time 155

List of Abbreviations

AD	automatic differentiation
AIT*	adaptively informed trees
AMP	adversarial motion priors
ANN	artificial neural network
ARA*	anytime-repairing A*
BC	behaviour cloning
BDS	Bayesian diversity search
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BGMR	Bayesian Gaussian mixture regression
BIT*	batch-informed trees
BLAS	Basic Linear Algebra Subprograms
BRL	Biomimetic Robotics Lab
BVP	boundary-value problem
CD	centroidal dynamics
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
CMM	centroidal momentum matrix
CPG	central pattern generator
CPU	central processing unit
CRB	composite rigid body
CVAE	conditional variational autoencoder
CoM	centre of mass
CoP	centre of pressure
CoRL	Conference on Robot Learning
Croccodyl	Contact RObot COntrol by Differential DYnamic programming Library
DARPA	Defense Advanced Research Projects Agency
DDP	differential dynamic programming
DDTI	data-driven trajectory initialization
DPhil	Doctor of Philosophy
DP	dynamic programming
DRL	deep reinforcement learning

DoF	degrees of freedom
EIT*	effort-informed trees
ETH	Eidgenössische Technische Hochschule Zurich
EoM	equation of motion
FDDP	feasibility-driven DDP
FDM	finite-difference method
FD	finite difference
GNSS	Global Navigation Satellite System
GPMP	Gaussian process motion planning
GPR	Gaussian process regression
GPS	guided policy search
GPU	graphics processing unit
GP	Gaussian process
HAA	hip abduction/adduction
HFE	hip flexion/extension
HJB	Hamilton-Jacobi-Bellman
HSL	Harwell Subroutine Library
HZD	hybrid zero dynamics
ICP	Instantaneous Capture Point
ICRA	International Conference on Robotics and Automation
IEEE	Institute of Electrical and Electronics Engineers
IIT	Istituto Italiano di Tecnologia
IK	inverse kinematics
IL	imitation learning
IMU	inertial measurement unit
IOC	inverse optimal control
IPM	interior-point method
IPOPT	Interior Point Optimizer
IP	incremental projection
IRL	inverse reinforcement learning
IROS	International Conference on Intelligent Robots and Systems
KFE	knee flexion/extension
KKT	Karush-Kuhn-Tucker
KL	Kullback-Leibler
KNITRO	Nonlinear Interior point Trust Region Optimization
KOMO	K-Order Markov Optimization
LAPACK	Linear Algebra PACKage
LIP	linear inverted pendulum

LL-SRBD	lump leg single-rigid-body dynamics
LMTR	Latent-Mode Trajectory Regressor
LQG	linear-quadratic-Gaussian
LfD	learning from demonstration
LiDAR	Light Detection And Ranging
MDP	Markov Decision Process
MIP	mixed-integer programming
MIT	Massachusetts Institute of Technology
MLP	multi-layer perceptron
ML	machine learning
MPC	model-predictive control
MUMPS	MUltifrontal Massively Parallel Sparse direct Solver
MoE	Mixture of Experts
NLP	nonlinear programming
NN	neural network
OCP	optimal-control problem
OC	optimal control
ODE	Ordinary Differential Equation
ODRI	Open Dynamic Robot Initiative
PARDISO	PARallel DIrect SOlver
PDE	Partial Differential Equation
PID	proportional-derivative-integral
PMP	Pontryagin's Maximum Principle
PPO	Proximal Policy Optimization
PRM	Probabilistic Roadmap Method
PbD	programming by demonstration
QP	quadratic programming
RBD	rigid-body dynamics
RGB-D	red, green, blue and depth
RHC	receding-horizon control
RHECALL	Receding-Horizon Experience-Controlled Adaptive Legged Locomotion
RH	receding-horizon
RL	reinforcement learning
RMC	resolved-momentum control
RNEA	Recursive-Newton-Euler Algorithm
RNN	recurrent neural networks
RRG	Rapidly exploring Random Graph

RRT	Rapidly exploring Random Tree
RSJ	Robotics Society of Japan
RIEMO	Riemannian Motion Optimization
SAC	Soft Actor-Critic
SBP	sampling-based planning
SD	symbolic differentiation
SEA	series elastic actuator
SH	stationary-horizon
SLIP	spring-loaded inverted pendulum
SLQ	sequential-linear-quadratic
SNOPT	Sparse Nonlinear Optimizer
SQP	sequential quadratic programming
SRBD	single-rigid-body dynamics
STOMP	Stochastic Trajectory Optimization for Motion Planning
SVM	support vector machine
TL	trajectory library
TOWR	Trajectory Optimization for Walking Robots
TO	trajectory optimization
TRAJOPT	Trajectory Optimization for Motion Planning
UAV	unmanned aerial vehicle
VMC	virtual-model control
WBC	whole-body control
ZMP	zero-moment point
***k*-NN**	*k*-nearest neighbors
ILQR	iterative linear-quadratic regulator
GRF	ground reaction force

List of Symbols

Abbreviations (used in subscripts)

Symbol	Description
a	actuated
ang	angular
b	base
c	current
der	derivative
dir	directional
dyn	dynamics
ee	end-effector
eq	equality (constraint)
err	error
ext	external
f	final
g	gravitational effects
h	heuristic
h	hip
H	horizon
in	inequality (constraint)
j	joint
lin	linear
max	maximum
min	minimum
ref	reference
st	stance
sw	swing
u	underactuated
0	initial

Scalars

Symbol	Description	Unit
a	anchor (kernel)	
c	centre	
c	contact state	
c	cycle factor (gait)	
ΔT	time duration	s
d	duty factor (gait)	
h	height (terrain)	m
k	image dimension	
k	scaling factor	
λ	eigenvalue	
l	length	m
m	stability margin	m
m	mass	kg
μ	friction coefficient	
n	number	
ω	weight (cost)	
ϕ	roll angle	rad
ψ	yaw angle	rad
q	joint angle	rad
Q	quaternion	
r	image resolution	m
ρ	density	$\mathrm{kg\,m^{-3}}$
σ	standard deviation	
t	time	s
θ	pitch angle	rad

Vectors

Symbol	Description	Unit
$\mathbf{a}$	lengths of the semi-axes of an ellipsoid	
$\mathbf{c}$	centre of pressure	
$\boldsymbol{\chi}$	state (learning)	
$\boldsymbol{\delta}$	linear position offset	m
$\mathbf{\Delta T}$	phase durations (stance and swing)	s
$\mathbf{e}$	unit vector or eigenvector	

Symbol	Description	Unit
$\mathbf{f}$	end-effector force	N
$\boldsymbol{\gamma}$	goal	
$\mathbf{g}$	gravity (acceleration) vector	$\mathrm{m\,s^{-2}}$
$\mathbf{h}$	centroidal (linear and angular) momenta	
$\hat{\mathbf{i}}$	unit vector in the direction of x axis	
$\hat{\mathbf{j}}$	unit vector in the direction of y axis	
$\mathbf{k}$	angular momentum	$\mathrm{kg\,m^2\,s^{-1}}$
$\hat{\mathbf{k}}$	unit vector in the direction of z axis	
$\mathbf{l}$	linear momentum	$\mathrm{kg\,m\,s^{-1}}$
$\boldsymbol{\lambda}$	input vector (MLP)	
$\boldsymbol{\mu}$	mean (average) values	
$\mathcal{N}$	nodes (optimization variables)	
$\mathbf{h}$	nonlinear effects (torque) vector	$\mathrm{N\,m}$
$\mathbf{n}$	normal vector	
$\mathbf{o}$	terrain obstacle (line/edge)	
$\boldsymbol{\omega}$	angular velocity	$\mathrm{rad\,s^{-1}}$
$\mathbf{1}$	vector of ones	
$\mathbf{p}$	end-effector linear position	m
$\boldsymbol{\phi}$	decoder network (LMTR) weights (parameters)	
$\mathbf{q}$	generalized coordinates	
$\mathbf{r}$	base linear position	m
$\boldsymbol{\rho}$	centroid to leg centre of mass linear position	m
$\boldsymbol{\sigma}$	standard deviation values	
$\boldsymbol{\tau}$	joint torque	$\mathrm{N\,m}$
$\boldsymbol{\tau}$	full trajectory	
$\boldsymbol{\theta}$	angular position (Euler angles)	rad
$\boldsymbol{\theta}$	neural network weights (parameters)	
$\mathbf{u}$	unit vector	
$\mathbf{u}$	control (input) trajectory	
$\mathbf{v}$	generalized velocities	
$\mathbf{w}$	optimization variables	
$\mathbf{x}$	state trajectory	
$\mathbf{x}$	task	
$\boldsymbol{\xi}$	instantaneous capture point	
$\mathbf{y}$	solution	
$\mathbf{z}$	mode	

Symbol	Description	Unit
$\mathbf{0}$	vector of zeros	

Matrices

Symbol	Description
$\mathbf{A}$	centroidal momentum matrix
$\mathbf{C}$	transformation matrix ($\dot{\boldsymbol{\theta}} \mapsto \boldsymbol{\omega}$)
$\mathbf{C}$	coriolis and centrifugal effects matrix
$\mathbf{G}$	Gaussian kernel matrix
$\mathbb{1}$	identity matrix
$\mathbf{I}$	image (heightmap) matrix
$\mathbf{I}$	inertia matrix
$\mathbf{J}$	Jacobian matrix
$\mathbf{M}$	mass matrix
$\mathbf{N}$	normals (and offsets) matrix
$\mathbf{R}$	rotation matrix
$\mathbf{S}$	selection matrix
$\mathbf{W}$	weight matrix
$\mathbb{0}$	matrix of zeros

Functions

Symbol	Description
A	a function approximator
d	distance metric (mode selection)
$\mathbf{f}$	a vector-valued function
J	a scalar-valued objective (cost) function
$\mathcal{N}$	normal distribution
O	big O notation (order of a function)
P	decoder network (CVAE)
$\boldsymbol{\pi}$	a policy
$\boldsymbol{\Gamma}$	a 3D third-order polynomial
Q	encoder network (CVAE)
T	a map

Spaces, sets, groups and algebra

Symbol	Description
$\mathcal{D}$	a dataset (of tasks and trajectories)
$\mathbb{H}$	the set of quaternions
$\mathfrak{J}$	joint or configuration space (planning)
$\mathbb{R}$	the set of real numbers
SE	Lie group, transformations
SO	Lie group, rotations
$\mathfrak{T}$	task or operational space (planning)
$\mathcal{X}$	task space (problem formulation)
$\mathcal{Y}$	solution space (problem formulation)
$\mathcal{Z}$	latent space (learning)

Frames

Symbol	Description
$\mathcal{B}$	base frame
$\mathcal{C}$	centre-of-mass frame
$\mathcal{L}$	local frame
$\mathcal{W}$	world frame

Mathematical operators

Symbol	Description
$\times$	cross product
δ	Kronecker delta
$\cdot$	dot product
$\times_n$	n-mode product (tensor-matrix multiplication)
$\bar{\times}_n$	n-mode product (tensor-vector multiplication)
$\odot$	Hadamard (element-wise) product
$\oplus$	tangent application
$\otimes$	outer product of tensors
T	transpose operator

1 Introduction

1.1 Context and motivation

Since the Industrial Revolution, machines and automation have continued to make a fundamental contribution towards the creation of economic growth and social value. Yet, while automated machines have gradually reduced the need for manual labour, many people are still engaged in dull, dirty, difficult, dangerous, dear (expensive) work[1]. Meanwhile, humans possess the dexterity, intellect, creativity and emotional capacity to engage in more interesting, fulfilling and safe activities.

The long-term vision of roboticists has been a utopian future in which machines are versatile and *autonomous*. Recently—thanks to remarkable progress in computer technology, the increase in processing power and the leveraging of mathematics—the development of advanced robots with superior perception, adaptability and mobility has become possible. An example of such innovation is shown in Fig. 1.1.

High-precision, fixed-base manipulators are now prevalent in the manufacturing industry where tasks are repetitive and whose environment does not

[1]Occasionally, undesirable jobs are denoted by the '3Ds' (dirty, dangerous and demeaning) neologism.



Figure 1.1: Detecting nuclear radiation at the Chernobyl nuclear power plant using a radiation sensor mounted on the Boston Dynamics Spot quadrupedal robot. The image was extracted from the video at https://youtu.be/PEgCZLvfgX8.

necessitate versatility because it is largely deterministic. Wheeled and flying robots were proposed for tasks requiring mobility; however, the former are generally ill-suited to uneven terrain, and the latter have limited operating time and load-carrying capacity. Alternatively, legged robots have been designed to operate in settings where limbs are advantageous, such as industrial facilities with stairs, natural environments with slopes and uneven terrain, or disaster sites with obstacles such as crevices in the ground surface or large debris. Recently, it has been recognized that the same principles that allow robots to walk could be applied to exoskeletons to provide paraplegic patients with increased mobility assistance.

While this morphology appears intuitively useful, novel solutions are required for a robot to move with the dexterity and poise of a human or an animal. One of the key requisites are locomotion algorithms—programs which produce complex motion plans for the whole body—that are robust to uncertainties in the models of the governing physics. In order for the robot to accurately and precisely execute a generated trajectory, the plan must not only respect the underlying dynamic and kinematic limits of the robot's mechanical system, it must

also leverage momentum so as to maintain stability while traversing challenging terrain. By walking at higher speeds, intelligently selecting footholds and adjusting timing of contact with the environment the robot can achieve natural, dynamic mobility.

To adapt to ever-changing conditions and sustain motion, the algorithm must replan while incorporating sensed information of the robot's surroundings and its state. Legged animals are able to make these kinds of rapid inferences about their circumstances; they subconsciously relate them to relevant prior experiences to leverage the properties of their bodies in a coordinated manner as shown in Fig. 1.2. Biological locomotion serves as inspiration to motion-generation research in robotics.



Figure 1.2: A Border Collie looking ahead and leveraging its body while rapidly slaloming between weave poles at the Masters Agility Championship at Westminster (photo by Reuters).

1.2 Aims and scope

This study aims to develop new methods for agile kinodynamic planning[2] for quadrupedal robots in the real world. The objective of this thesis is to propose a framework for online replanning of long-horizon, high-fidelity, terrain-aware, whole-body trajectories[3] on the robot under modelling and sensing uncertainties. The work will investigate how to leverage optimization-based methods whose convergence to a locally optimal solution can be accelerated by initializing the solver with an informed guess. To this end, a machine-learning-based approach will be explored. A pre-computed diverse dataset of solutions to locomotion problems will be generated. The dataset will be concisely approximated by a neural network which will function as the initializer of the solver during real-time deployment.

1.3 Outline

The work completed as part of this thesis is presented in the following chapters. At the beginning of each chapter, a reference is made to peer-reviewed publications from which material has been adopted.

Chapter 1: *Introduction*

This chapter gives the context and motivation for the development of quadrupedal robots and their motion-planning algorithms. General objectives and the scope of this thesis are presented. The structure of this document, summarizing the work completed as part of this study, is outlined. The resulting scientific contributions are highlighted, and the corresponding peer-reviewed publications are listed.

[2]Here, kinodynamic planning involves finding positions, velocities, accelerations and forces of the robot joints such that they satisfy kinematic (e.g., terrain, reachability, obstacle avoidance) and dynamic (dynamics, force limits) constraints.

[3]Whole-body trajectories refer to the interdependent motion plans for the robot's CoM and its end effectors (feet).

Chapter 2: *Background*

This chapter is composed of the introductory material, and offers a general yet extensive review of literature related to the topics explored in this study. Accurate mathematical models of the dynamics of legged robots are introduced. Popular methods for motion planning, specifically trajectory optimization (TO) and nonlinear programming (NLP), are discussed in depth. Techniques using machine learning for motion generation, and trajectory representation and reconstruction are referenced. Also, motion control is reviewed, especially the use of whole-body control (WBC) as the means by which complex trajectories are executed and stabilized.

Chapter 3: *Costs and constraints for reliable trajectories*

This chapter focuses on the formulation of optimization techniques for the generation of feasible, viable and reliable whole-body trajectories. We propose, and describe the development of, fast-to-compute analytical (integral) costs—for trajectory optimization based on direct collocation parameterized by polynomials of Hermite form—and the construction of kinematic constraints, dependent on the sensed terrain. The effectiveness of these elements is validated experimentally, in simulation and on hardware, and these results are thoroughly discussed. The chapter includes elements of [1], [2].

Chapter 4: *Receding-horizon adaptive trajectory optimization*

This chapter describes our adaptation of direct-collocation-based trajectory optimization for receding-horizon use. We explain how we achieve this via the *asynchronous* and segmentation-based formulation, and a heuristic that efficiently reuses the previous plan. We detail the testing and deployment, showing the application and utility of our method. We attempt to improve the computational

efficiency of the algorithm by exploiting the problem structure and by adapting the optimization solver. We propose and test replacing the simplified dynamics model with a learned *stability margin* constraint which accounts for torque limits. The chapter includes elements of [2], [3].

Chapter 5: *Data-driven solver initialization*

This chapter presents our attempts to accelerate trajectory optimization, through the initialization of primal variables of an interior-point method using a neural network. This method is trained in a supervised fashion (behaviour cloning), on a self-generated dataset of reliable expert motion on complex terrain. To enable replanning from dynamic states, we propose to segment optimal long-horizon expert trajectories, with static initial and final conditions, into shorter sections, with (a) dynamic initial and final conditions and (b) decoupled, *asynchronous* base and end-effector trajectories, leveraging a latent-space regressor to learn (unsupervised) and reconstruct the resulting multimodal classes. We demonstrate the execution of the reconstructed trajectories, on the real robot, with minimal or no refinement. The chapter includes elements of [1], [4], [2].

Chapter 6: *Dynamics models for inertia shaping*

This chapter outlines the extension of the proposed Receding-Horizon Experience-Controlled Adaptive Legged Locomotion (RHECALL) framework from an existing single-rigid-body dynamics (SRBD) model to lump leg single-rigid-body dynamics (LL-SRBD) and centroidal dynamics (CD) models. These models can account for the nonlinear effects of moving joints on the distribution of the inertia and the dynamics of the robot. This enables us to generate more dynamically accurate trajectories; they

reduce the risk of exceeding the intrinsic capabilities of the real physical system and are more likely to be successfully executed.

Chapter 7: *Conclusion*

This chapter concludes the thesis and discusses future work.

1.4 Contributions

This thesis summarizes the outcomes of the undertaken research on locomotion planning algorithms specifically designed for quadrupedal robots. During these studies, a framework, named Receding-Horizon Experience-Controlled Adaptive Legged Locomotion (RHECALL), which leverages numerical optimization and machine learning has been developed. The proposed methods have been tested using the ANYbotics ANYmal quadrupeds (ANYbotics [5], Hutter et al. [6], Hutter et al. [7]). Fig. 1.3 and Fig. 1.4 demonstrate the methods being successfully deployed on these systems.

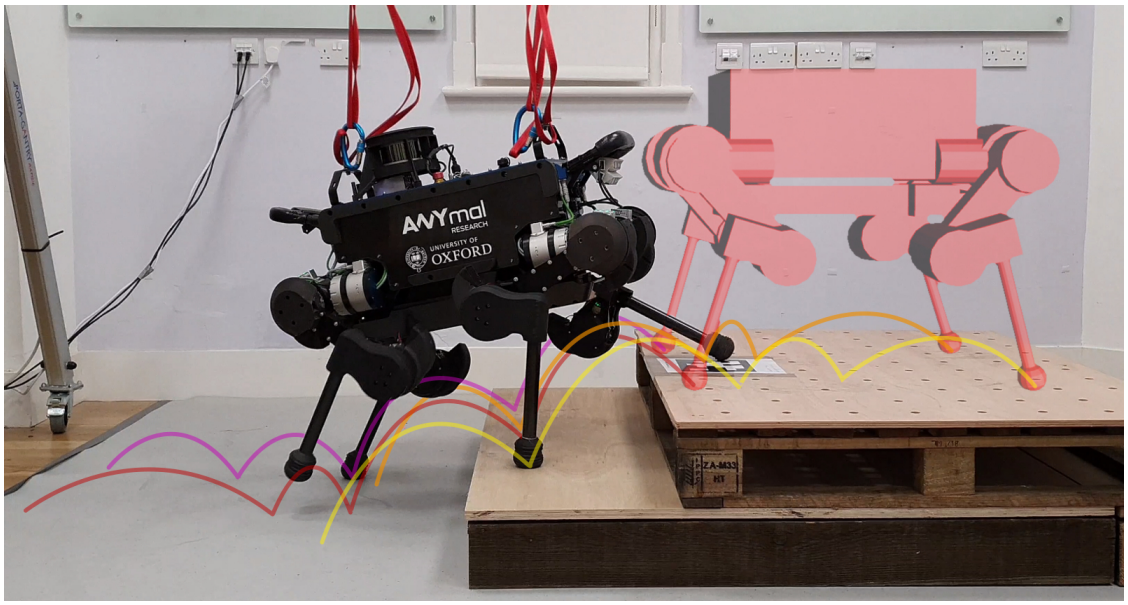


Figure 1.3: The ANYbotics ANYmal B quadruped executing a dynamic, long-horizon whole-body trajectory on complex terrain. The trajectory was quickly computed by a neural network and adapted by an optimization solver.

The main scientific contributions of the work undertaken as part of this Doctor of Philosophy (DPHIL) study are:

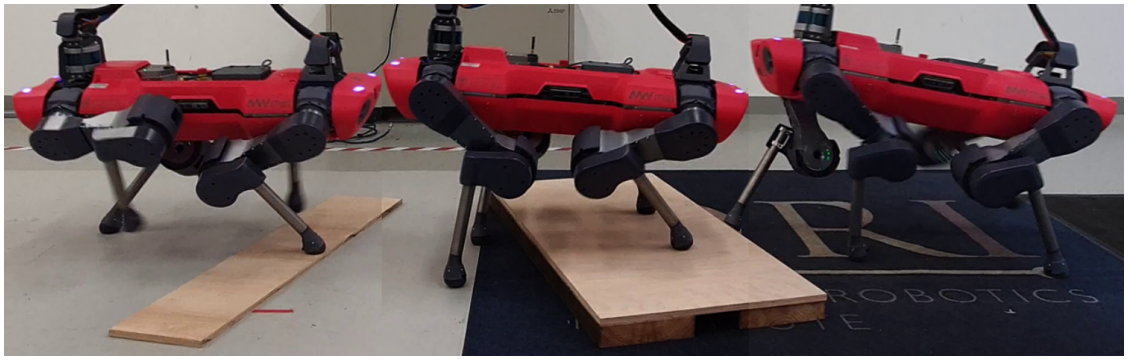


Figure 1.4: The main contribution of this work, the Receding-Horizon Experience-Controlled Adaptive Legged Locomotion (RHECALL) receding-horizon framework, being deployed on the ANYbotics ANYmal C quadruped.

- The development of fast-to-compute analytical costs and kinematic constraints, for trajectory optimization, to produce motion plans with *desirable* styles[4], which facilitate *reliable* execution of dynamic quadrupedal locomotion on complex terrain.

- The first algorithm for perceptive planning of dynamic, high-fidelity locomotion for quadrupedal robots in a *receding-horizon* fashion using a nonlinear trajectory optimization based on direct collocation, capable of finding base and end-effector trajectories, footholds and *asynchronous*[5], *aperiodic*[6] contact sequences.

- A framework for generating, processing, learning and reconstructing dynamic trajectories which enables: (a) the *acceleration* of optimization solvers through warm-starting, and (b) the provision of short-term guidance that *abstracts* temporally extended objectives[7].

- The implementation of the CD model, with analytical first-order derivatives, for a direct-collocation-based optimization problem parameterized by decision variables expressed in *Euclidean* space.

[4]We call the manner in which the robot moves a style.

[5]End-effector trajectories do not use the time step chosen for the discretization of the base motion splines.

[6]We do not confine the robot to any periodic gait (e.g., trot).

[7]The neural network (NN) provides a segment of a long-horizon plan it deems most fitting to a situation.

- Extensive, experimental validation of the proposed methods, on multiple state-of-the-art hardware platforms, demonstrating its viability in the presence of real-world challenges.

1.5 Publications

The work undertaken as part of the DPHIL studies, in collaboration with other researchers, resulted in the publication of the following peer-reviewed articles, which are summarized in Table 1.1, and accompanied by links to the respective papers 📄, videos 🎥, and presentations 👨‍🏫.

First-authored:

[1] **O. Melon**, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1410–1416
📄 https://ieeexplore.ieee.org/document/9196562
🎥 https://youtu.be/LKFDB_BOhl0
👨‍🏫 https://youtu.be/YWhtdNb83yw

[2] **O. Melon**, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811
📄 https://ieeexplore.ieee.org/document/9560794
🎥 https://youtu.be/Qqs5hhp3hHQ
👨‍🏫 https://youtu.be/bfc_TOU3AeA

Co-authored:

[4] D. Surovik, **O. Melon**, M. Geisert, M. Fallon, and I. Havoutis, "Learning an expert skill-space for replanning dynamic quadruped locomotion over obstacles," in *Proceedings of the Conference on Robot Learning*, PMLR, 2021
https://corlconf.github.io/corl2020/paper_346/
https://youtu.be/bHspVpSGAjk
https://youtu.be/w3v1BingoXQ

[3] R. Orsolino, S. Gangapurwala, **O. Melon**, M. Geisert, I. Havoutis, and M. Fallon, "Rapid Stability Margin Estimation for Contact-Rich Locomotion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 8485–8492
https://ieeexplore.ieee.org/document/9636474
https://youtu.be/E2cUo1-F9Xw
https://youtu.be/cg2Tme0t4QA

[8] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, **O. Melon**, I. Havoutis, and I. Posner, "First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5343–5350
https://ieeexplore.ieee.org/document/9340737
https://youtu.be/z1UopkfIPhI
https://ieeexplore.ieee.org/document/9340737/media

Table 1.1: A chronological list of publications resulting from the work conducted as part of the DPhil studies. To date, 2 first-author and 3 co-author papers, have been published in top-tier conferences.

Name	Ref.	Author	Date Pub.	Type	Organization	Links
ICRA20	[1]	1st	2020/05	Conf.	IEEE ICRA	
IROS20	[8]	6th	2020/11	Conf.	IEEE/RSJ IROS	
CoRL20	[4]	2nd	2020/11	Conf.	CoRL	
ICRA21	[2]	1st	2021/05	Conf.	IEEE ICRA	
IROS21	[3]	3rd	2021/11	Conf.	IEEE/RSJ IROS	

2
Background

2.1 Introduction

Legged robots have a very long and fascinating history, which is summarized in Sec. 2.2.1. These robots use sophisticated sensors listed in Sec. 2.2.2. The planning and execution of dynamic locomotion for these systems is an open-research problem due to many associated challenges discussed in Sec. 2.2.3. To make it practicable, a number of assumptions listed in Sec. 2.2.4 and various dynamics models described in Sec. 2.3 are devised. Using methods described in Sec. 2.4 locomotion trajectories can be planned while conforming to a set of environmental conditions and physical restrictions. These methods can benefit from learning techniques mentioned in Sec. 2.5. To realize the planned motion, a control system based on approaches outlined in Sec 2.6 is required to convert the high-level plan into low-level torque commands for the individual actuators that cause a robot to move. The chapter is concluded in Sec. 2.7 where the choice of particular methods employed in this thesis is explained.

2.2 Overview of legged robots

2.2.1 Modern robots and their notable predecessors

The timeline of walking machines by Hoggett [9] lists many ingenious and inspiring walking machines built throughout history. According to ancient Chinese records discussed by Yan [10], some of the first legged machines were the Wooden Horse Carriage of Lu Ban around 480 BC and the Wooden Ox and Gliding Horse of Zhu-ge Liang around 230 AC. In around 1495, Leonardo da Vinci, produced detailed drawings of a mechanical knight capable of basic upper-body motions [11]. In 1663, the Cart with Legs by Francis Potter was presented by Robert Hooke, a British polymath, to the Royal Society [12].

As documented by Raibert [13], significant milestones in the development of legged robots can be traced back as far as 1850 when Chebyshev, the famous Russian mathematician, designed his linkages that convert rotational motion to approximate linear motion and enabled the construction of the plantigrade machine capable of trot-like motion. During this period, Muybridge was tasked with finding out whether a trotting horse leaves the ground with all four feet at the same time which he proved with photographs—an analysis he then extended to over 40 mammals including humans ([14], [15]), which offered great value to research on legged locomotion. Soon, peculiar inventions appeared; Rygg [16] patented a human-powered mechanical horse and Wallace [17] patented a one-legged hopping tank.

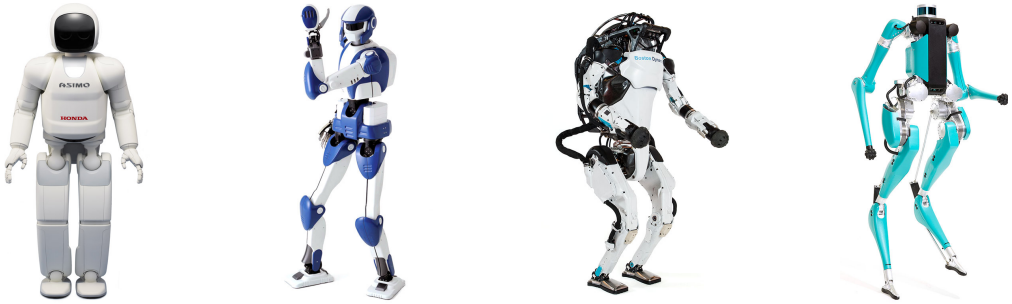
Between 1960 and 1970, the control of inverted pendulums, the basis of many walking robots, was developed [18]. In 1966, McGhee and Frank designed Phoney Pony, the first machine controlled by simple digital logic [19]. In 1968, General Electric's Walking Truck, a $3.3\,\mathrm{m}$ tall human-operated hydraulic quadruped weighing $1400\,\mathrm{kg}$, was shown climbing railway sleepers [20]. In the 1970s, the Masha hexapod by Gurfinkel et al. [21] used hybrid computer controls (tethered) to climb stairs; later, Gorinevsky and Shneider [22] used it to demonstrate force control for locomotion.

In the 1980s, Raibert [23] proposed and simulated a two-dimensional hopping system which served as the basis for a real one-legged hopping machine [24]. Later, Raibert et al. [25] generalized the ideas for a quadruped capable of running with many gaits via the concept of a 'virtual leg'. At the end of the decade interest in bipedal robots grew; Takanishi et al. [26] achieved dynamic walking on WL-10RD and Playter and Raibert [27] proposed an in-flight feedback control strategy to enable a biped to execute somersaults. Likely a successor of Space General's Moonwalker, a creation inspired by Tinsley's concept and Mirrison's prototype of the Iron Mule Train, in 1991, the octopod built by Todd [28] was shown walking in building sand and climbing stairs.

In 1998, Fujita and Kitano [29] developed MUTANT which became Sony's AIBO [30], an entertainment robot with sophisticated touch, hearing, sight, balancing and other capabilities. Honda developed its E- and P-series humanoids which led to the creation of ASIMO in 2000 [31], a robot with impressive environment-sensing and agile-locomotion capabilities. Around 2000, Defense Advanced Research Projects Agency (DARPA) funded Boston Dynamics, a spin-out from the Massachusetts Institute of Technology (MIT), to build a "rough-terrain walker". In 2005, BigDog arrived [32]. In 2011, the company developed a research-oriented LittleDog [33]. Focused on military applications, Boston Dynamics also created AlphaDog Proto, LS3 and Wildcat—a quadruped capable of running at $32\,\mathrm{km\,h^{-1}}$.

Groups at Eidgenössische Technische Hochschule Zurich (ETH) developed ALoF to aid research into perceptive quadrupedal locomotion in 2011 [34] and StarlETH for compliant locomotion via series elastic actuators (SEAs) in 2012 [35].

Researchers at Istituto Italiano di Tecnologia (IIT) focus on hydraulically actuated quadrupeds. HyQ was revealed in 2011 [36], and its improved version, HyQ2Max appeared in 2017 [37]. IIT, in a partnership with Moog Inc., built HyQReal, a $130\,\mathrm{kg}$, $90\,\mathrm{cm}$ tall, hydraulic robot [38]. Hydraulic actuation



(a) ASIMO (b) HRP-4 (c) Atlas (d) Digit

Figure 2.1: Examples of humanoid robots: (a) ASIMO demonstrated an impressive locomotion and manipulation skill set; (b) HRP-4 was a popular research platform; (c) Atlas navigated a parkour course; and (d) Digit carried packages in a warehouse.

yields high power density and speed, but the oil can be a safety and environmental hazard.

In the meantime, humanoids with many degrees of freedom (DOFs) were constructed including HRP family [39]; iCub [40] and COMAN [41] by IIT; Boston Dynamics' Atlas [42] whose first iteration was $188\,\mathrm{cm}$ tall, weighing $155\,\mathrm{kg}$ with 130 states and 28 actuated DOFs [43]; and PAL's TALOS [44].

While having lower energy density, electro-mechanical actuation requires simpler system architecture and is more precise as the actuators allow the motion of a joint to be finely controlled. The Biomimetic Robotics Lab (BRL) group at MIT is working on the design of and control algorithms for Cheetah quadrupeds [45], [46], [47] which are intended for high-speed and energy-efficient running. For instance, Park et al. [48] demonstrated Cheetah 2 bounding at $6\,\mathrm{m\,s^{-1}}$. Recently, the group began working on a humanoid designed for acrobatic behaviours [49]. Ghost Robotics [50] manufactures Vision 60, a quadruped constructed for unstructured urban and natural environments for defence, homeland and enterprise applications. Boston Dynamics [51] sells Spot, likely the most recognized quadruped known for its robustness to disturbances, and autonomous sensing and inspection capabilities. Similarly, ANYbotics [5], a spin-out from ETH, developed ANYmal [6], [7] for industrial inspection. The legs of this versatile machine are driven by twelve electric motors mounted at the joints to facilitate whole-body control techniques. The kinematic structure



(a) Phoney Pony **(b)** BigDog **(c)** Mini Cheetah **(d)** HyQReal

Figure 2.2: Examples of quadrupedal robots: (a) Phoney Pony was the first machine controlled by a digital logic [19]; (b) Boston Dynamics BigDog was a hydraulic military robot with *ankle* joints; (c) MIT Mini Cheetah was used to execute athletic motions like backflips; and (d) IIT HyQReal generated enough force to pull a small airplane.

of the robot is designed to achieve dynamic mobility to allow it to overcome obstacles and stairs. Unitree [52] built several customer- and industry-grade quadrupeds like Laikago and A1.

Open Dynamic Robot Initiative (ODRI) released hardware documentation, and open-sourced the software, of a torque-controlled modular robot architecture for the Solo quadrupedal and the Bolt bipedal robots discussed by Grimminger et al. [53]. Agility Robotics [54] built bipedal, ostrich-like Cassie and humanoid Digit. Wandercraft [55] invented Atlante, a self-balancing exoskeleton for paraplegic patients which automatically adapts its algorithm to the weight and size of the user.

To conserve time and effort in predominately flat and open environments, researchers have been investigating supplementing legged robots with wheels. Prominent examples of hybrid, wheeled-legged robots include Roller-Walker by Endo and Hirose [56], Boston Dynamics' Handle [51], Mars rovers with active suspension Sherpa by Cordes et al. [57] and MAMMOTH by Reid et al. [58] Momaro by Klamt and Behnke [59] Skaterbot by Geilinger et al. [60], Ascento by Klemm et al. [61], Centauro by Klamt et al. [62] or ANYmal on Wheels by Bjelonic et al. [63]. Jud et al. [64] adapted the Menzi Muck hydraulic excavator for autonomous operation which is the first such system to integrate leg kinematics and wheel odometry, in addition to inertial measurement unit (IMU) and

Global Navigation Satellite System (GNSS), into state estimation.

2.2.2 Onboard sensors

Any robotic system which moves through space requires sensors to learn about its own state and its surroundings and uses this information for *state estimation*[1] and planning.

As discussed by Barfoot [65], the sensors can be categorized as *proprioceptive* and *exteroceptive* depending on whether they sense the *internal* state of the robot or the *external* state of the environment, respectively.

Proprioception refers to measuring kinematic (position, velocity and acceleration) and dynamic quantities (force, torque) by, e.g., IMUs (Fig. 2.3a) that measure the linear acceleration and angular velocity of a body, encoders and torque sensors (Fig. 2.3b) which measure the position of a motor and the torque it generates. These measurements are obtained at high frequency and with low latency which is essential for accurate state estimation and motion control, a topic discussed in Sec. 2.6.

Exteroception captures information about the absolute state of the environment (like its structure and features) by devices such as red, green, blue and depth (RGB-D) cameras (Fig. 2.3c) and Light Detection And Ranging (LIDAR) laser-based scanners (Fig. 2.3d). Due to the size of the data and the required preprocessing, the sensors provide low frequency and high latency measurements. These are used for planning of trajectories as discussed in Sec. 2.4. Notably the cameras' depth data can be used to construct a 2.5D elevation map (heightmap), i.e., a grid-like model of the terrain where each (x, y) point or cell contains a singular height value (Fankhauser and Hutter [66]). Meanwhile, LIDAR enables a full 3D reconstruction of the environment (Wang et al. [67]).

[1] State estimation is the process of using sensor measurements and analytical models to determine the internal and relative (to a chosen global frame) state of a system in an environment.



(a) IMU



(b) joint



(c) RGB-D camera



(d) LiDAR

Figure 2.3: Typical examples of sensors used by legged robots: (a) Epson G365 IMU; (b) Maxon modular integrated robot joint which consists of an electric actuator, with integrated encoders and a torque sensor; (c) Intel RealSense D435i RGB-D camera; and (d) Velodyne Puck (VLP-16) LiDAR.

2.2.3 Challenges

The real-world challenges associated with legged robots are the consequences of their physical construction (which includes their structure, components and sensors) and the difficulty of the tasks they are expected to solve in an autonomous fashion. Some of the most significant difficulties are:

Nonlinearity and nonconvexity: The models of the physics that govern the motion of rigid bodies, as well as the dynamics of the actuators, are inherently nonlinear. Typically, the combination of nonlinearities results in the optimization problem becoming nonconvex, i.e., there are multiple locally optimal solutions, and finding the global optimum is exceptionally difficult.

Degrees of freedom (DoFs): Legged robots can have dozens of DoFs whose motion must be controlled simultaneously. Each additional DoF becomes another input variable[2] of the dynamics function which increases the complexity of the problem. In the case of methods like dynamic programming (DP), where the increase is exponential, Bellman [68] referred to this challenge as the 'curse of dimensionality'.

Underactuation: Some of the DoFs of legged robots (e.g., those of the *base* link) are not directly actuated and are instead indirectly controlled via the

[2]The number of input variables of a function is known as its *dimensionality*.

motors located in the robot's limbs which generate force by interacting with the environment.[3]

Contacts: The interaction with the environment through contact is a discontinuous, time-dependent event which makes the motion-planning problem non-differentiable.[4] Considering discrete contacts will make the optimization problem *more nonconvex*[5].

Redundancy: As a result of the high number of DOFs and underactuation, there exists more than one way of achieving a given objective; defining the *style* in which it should be achieved is nontrivial.

Dimensionality: Because of all the aforementioned challenges, the number of controllable variables can be very large.

Sensing and state estimation: Modelling a real environment and localizing a robot in space, i.e., determining the *pose and twist*[6], is paramount to locomotion planning. Perception algorithms may produce inaccurate models or imprecise signals that *drift*—a behaviour caused by sensor deficiencies, atmospheric conditions, deformable terrain, unaccounted for mechanical compliance or contact slipping—negatively affecting motion-planning algorithms that directly (naively, without compensation) rely on them. Vigorous motions can result in the bending of limbs and motion blur inducing kinematic and visual odometric errors respectively (Wisth et al. [70]). Having minimal base motion helps with the acquisition of data by the onboard sensors, for localization and mapping purposes. It is a dual problem which inspires coupled research on robust perception and planning methods.

Problem formulation: There is no unique way in which a robot completes a desired task. While taking into account certain requirements—like

[3] Robots with an underactuated base are referred to as *floating-base* systems.
[4] Robots that make and break contact with the environment are referred to as *hybrid* systems.
[5] Hodge et al. [69] quantify nonconvexity using the *convexity coefficient*.
[6] Pose and twist are quantities which combine the linear and angular *positions* and *velocities*, respectively.

conserving energy, minimizing superfluous motion, achieving a particular style of motion, avoiding collisions with the environment or maintaining balance by being robust to perturbations—deciding how to represent the problem and how to formulate costs, constraints or rewards which facilitate the requirements can be nonintuitive.

Computational resources: Real physical systems execute algorithms on computers with limited computational resources which must complete various calculations and processes in real time.

2.2.4 Assumptions

In order to formulate a mathematical model of a robot and reduce the complexity of the defined problem, certain assumptions can be made:

Dynamics: The rigid-body dynamics (RBD) model considers bodies to be non-deformable while the single-rigid-body dynamics (SRBD) model further simplifies it by neglecting the effects of joint velocities on linear and angular momenta and assuming a non-changing rotational inertia[7]. The simplest models, like the linear inverted pendulum (LIP), assume that the angular velocity and acceleration of the base are zero. Many successful approaches approximate the dynamics with educated heuristics.

Contacts: Typically, only intentional contact between the robots' end effectors and the environment is considered; in the case of quadrupedal robots, the contacts are often considered to be rigid and concentrated at a point.

Phase durations: The timings of contact phases are generally predefined.

Terrain representation: Generally, planning algorithms assume the environment to be static. While separate, high-level navigation algorithms represent the environment in 3D, most locomotion planning algorithms avail of

[7]The SRBD model precomputes the rotational inertia matrix by combining the inertia of all links while in some nominal configuration.

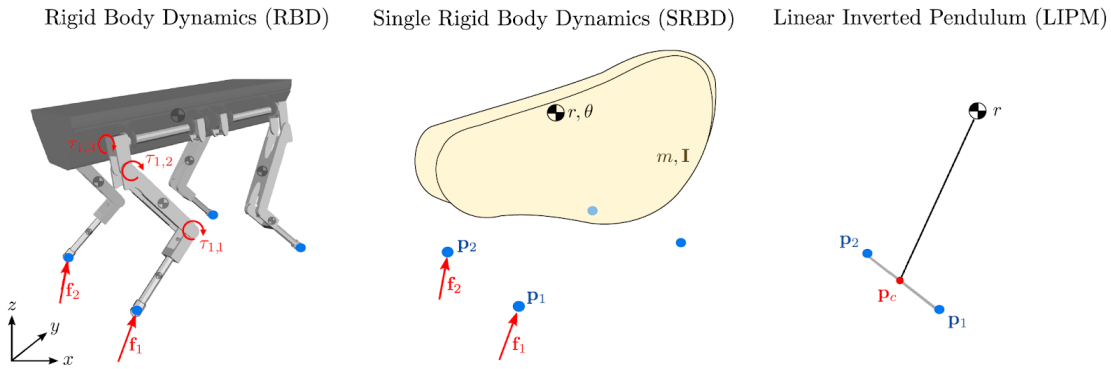


Figure 2.4: The diagrams of the three main dynamic models used for legged locomotion, listed in order of decreasing complexity, as presented by Winkler [71].

a 2.5D heightmap as a model of the traversable terrain. The algorithms that map the terrain for the purpose of locomotion generally ignore the semantics, i.e., the type of terrain and its dynamics.

2.3 Modelling of legged robots

A physical system we wish to understand and control must be represented and approximated by an appropriate mathematical model. The system is characterized by its state $u(x, y, z, t)$—a function of independent variables—at any point in space $\{x, y, z\}$ and time t. Partial differential equations (PDEs) can model the relationships between an unknown function, some independent variables and the partial derivatives of the function with respect to the variables.

In the case of robotics, the state variables are functions of time only (e.g., positions, velocities and accelerations) and can therefore be modelled as a system of Ordinary Differential Equations (ODEs) of the general, *explicit* form

$$\mathbf{x}^{(n)}(t) = \mathbf{f}(\mathbf{x}^{(n-1)}(t), \ldots, \mathbf{x}(t), \mathbf{u}^{(n-1)}(t), \ldots, \mathbf{u}(t), t), \tag{2.1}$$

where n is the highest order of the derivative, $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the input vector and $\mathbf{f}(\cdot)$ is a nonlinear, vector-valued function representing the dynamics of the robot. Mechatronic systems which have fewer actuators m than the number of generalized coordinates n, i.e., $n > m$, are considered *underactuated*. They have unactuated (passive) DoFs which are *distributed* throughout the

kinematic chain as a result of flexibility in the links or joints (e.g., series-elastic actuators), or *concentrated* at the root (base) link of the kinematic chain. The latter type connects the *floating-base* link via a virtual, rigid-body, 6-DOFs joint to inertial ground (Nenchev et al. [72]). If the state variables are appropriately redefined, the n^{th}-order dynamics described by Eq. 2.1 can be expressed as n first-order differential equations, using the state-space representation, of the well-known form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t). \tag{2.2}$$

Typically, when modelling dynamical systems in robotics, it is sufficient to consider first- and second-order time derivatives; we denote them using Newton's notation (e.g., $\dot{\mathbf{x}}, \ddot{\mathbf{x}}$) and omit the explicit reference to time t.

In the following subsections, a number of different mathematical models of dynamical robotic systems, known as equations of motion (EOM), will be introduced.

2.3.1 Rigid-body dynamics model

The dynamics of underactuated, rigid systems, such as floating-base walking robots, can be modelled using the rigid-body dynamics (RBD) model. It omits the nonlinearities associated with the actuators and assumes no structural deformations. In the case of legged platforms, the torso cannot be directly controlled, the base (b) is propelled using the actuators located in the limbs, at the joints (j). Such a floating-base system is modelled by the following matrix ODE

$$\mathbf{M}(\mathbf{q})\,\dot{\mathbf{v}} + \underbrace{\mathbf{C}(\mathbf{q},\mathbf{v})\,\mathbf{v} + \boldsymbol{\tau}_{\mathrm{g}}(\mathbf{q})}_{\mathbf{h}(\mathbf{q},\mathbf{v})} = \mathbf{S}^{\intercal}\,\boldsymbol{\tau} + \mathbf{J}^{\intercal}(\mathbf{q})\,\mathbf{f} \tag{2.3}$$

whose components are:

$\mathbf{q}$	$\in \mathbb{R}^{(3+n_{\boldsymbol{\theta}}+n_\mathrm{j})}$	generalized coordinates
$\mathbf{v}, \dot{\mathbf{v}}$	$\in \mathbb{R}^{(6+n_\mathrm{j})}$	generalized velocities and accelerations
$\mathbf{M}(\mathbf{q})$	$\in \mathbb{R}^{(6+n_\mathrm{j})\times(6+n_\mathrm{j})}$	mass matrix (orthogonal)
$\mathbf{C}(\mathbf{q},\mathbf{v})$	$\in \mathbb{R}^{(6+n_\mathrm{j})\times(6+n_\mathrm{j})}$	Coriolis and centrifugal effects
$\boldsymbol{\tau}_\mathrm{g}(\mathbf{q})$	$\in \mathbb{R}^{(6+n_\mathrm{j})}$	torque due to gravity
$\mathbf{h}(\mathbf{q},\mathbf{v})$	$\in \mathbb{R}^{(6+n_\mathrm{j})}$	torque due to the nonlinear effects
$\mathbf{S}$	$\in \mathbb{R}^{(6+n_\mathrm{j})\times(6+n_\mathrm{j})}$	selection matrix
$\boldsymbol{\tau}$	$\in \mathbb{R}^{n_\mathrm{j}}$	generalized torques
$\mathbf{J}(\mathbf{q})$	$\in \mathbb{R}^{3n_\mathrm{ee}\times(6+n_\mathrm{j})}$	(geometric) Jacobian of position of external forces
$\mathbf{f}$	$\in \mathbb{R}^{3n_\mathrm{ee}}$	external forces

The orientation of the floating base $\boldsymbol{\theta}$ can be parametrized using Euler angles or a quaternion, which results in $n_{\boldsymbol{\theta}} = 3$ or $n_{\boldsymbol{\theta}} = 4$, respectively. Therefore, the dimensions of the generalized coordinates $\mathbf{q} = \begin{bmatrix}\mathbf{q}_\mathrm{b}^\intercal & \mathbf{q}_\mathrm{j}^\intercal\end{bmatrix}^\intercal = \begin{bmatrix}{}^{\mathcal{W}}\mathbf{r}^\intercal & \boldsymbol{\theta}^\intercal & \mathbf{q}_\mathrm{j}^\intercal\end{bmatrix}^\intercal$, where ${}^{\mathcal{W}}\mathbf{r} \in \mathbb{R}^3$ is the base linear position (expressed in the world frame $\mathcal{W}$), are not necessarily equal to the dimensions of the generalized velocities $\mathbf{v} = [\mathbf{v}_\mathrm{b}^\intercal \ \mathbf{v}_\mathrm{j}^\intercal]^\intercal = \begin{bmatrix}{}^{\mathcal{B}}\dot{\mathbf{r}}^\intercal & {}^{\mathcal{B}}\boldsymbol{\omega}^\intercal & \mathbf{v}_\mathrm{j}^\intercal\end{bmatrix}^\intercal$. The latter include Euclidean-space linear ${}^{\mathcal{B}}\dot{\mathbf{r}}$ and angular ${}^{\mathcal{B}}\boldsymbol{\omega}$ velocities of the base (expressed in the base frame $\mathcal{B}$) as well as 'joint-space' velocities of the joints $\mathbf{v}_\mathrm{j}$, respectively. Splitting the tensors of Eq. 2.3 into unactuated (u) and actuated (a) components of the 'floating' base (b) and the joints (j)

$$\begin{aligned}\begin{bmatrix}\mathbf{M}_{\mathrm{u,b}} & \mathbf{M}_{\mathrm{u,j}}\\ \mathbf{M}_{\mathrm{a,b}} & \mathbf{M}_{\mathrm{a,j}}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{v}}_\mathrm{b}\\ \dot{\mathbf{v}}_\mathrm{j}\end{bmatrix} + \begin{bmatrix}\mathbf{C}_{\mathrm{u,b}} & \mathbf{C}_{\mathrm{u,j}}\\ \mathbf{C}_{\mathrm{a,b}} & \mathbf{C}_{\mathrm{a,j}}\end{bmatrix}\begin{bmatrix}\mathbf{v}_\mathrm{b}\\ \mathbf{v}_\mathrm{j}\end{bmatrix} + \begin{bmatrix}\boldsymbol{\tau}_{\mathrm{u,g}}\\ \boldsymbol{\tau}_{\mathrm{a,g}}\end{bmatrix}\\ = \begin{bmatrix}\mathbf{S}_{\mathrm{u,b}} & \mathbf{S}_{\mathrm{u,j}}\\ \mathbf{S}_{\mathrm{a,b}} & \mathbf{S}_{\mathrm{a,j}}\end{bmatrix}^\intercal\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\tau}\end{bmatrix} + \begin{bmatrix}\mathbf{J}_{\mathrm{u,b}} & \mathbf{J}_{\mathrm{u,j}}\\ \mathbf{J}_{\mathrm{a,b}} & \mathbf{J}_{\mathrm{a,j}}\end{bmatrix}^\intercal\begin{bmatrix}\mathbf{0}\\ \mathbf{f}\end{bmatrix}.\end{aligned} \tag{2.4}$$

Note that $\mathbf{M}_{\mathrm{u,j}} = \mathbf{M}_{\mathrm{a,b}}^\intercal$ is the coupling inertia matrix (Nenchev et al. [73]), which results in an inertial coupling between the passive (underactuated) and active (actuated) DOFs. More concisely, Eq. 2.4 can be represented using two matrix ODEs

$$\mathbf{M}_\mathrm{u}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{h}_\mathrm{u}(\mathbf{q},\mathbf{v}) = \mathbf{J}_\mathrm{u}^\intercal(\mathbf{q})\mathbf{f}, \tag{2.5a}$$

$$\mathbf{M}_\mathrm{a}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{h}_\mathrm{a}(\mathbf{q},\mathbf{v}) = \boldsymbol{\tau} + \mathbf{J}_\mathrm{a}^\intercal(\mathbf{q})\mathbf{f}. \tag{2.5b}$$

This detailed model can be used to generate complex trajectories for high-dimensional humanoids as shown by Buss et al. [74], Mombaur [75] and Lengagne et al. [76].

2.3.2 Centroidal dynamics model

The centroidal dynamics (CD) model was introduced by Orin et al. [77]. It represents the underactuated dynamics of the RBD model introduced in Sec. 2.3.1, shown in Eq. 2.5a. Budhiraja et al. [78] proposed a rigorous mathematical framework which enforces consensus, i.e., results in equivalent solutions for the two models.

CD considers the linear momentum $\mathbf{l} \in \mathbb{R}^3$ and angular momentum $\mathbf{k} \in \mathbb{R}^3$ of a body, which can be computed as follows

$$\mathbf{h} = \begin{bmatrix} \mathbf{l} \\ \mathbf{k} \end{bmatrix} = \mathbf{A}(\mathbf{q})\,\mathbf{v}, \tag{2.6}$$

where $\mathbf{A} \in \mathbb{R}^{6\times(6+n)}$ is the centroidal momentum matrix (CMM), a product of a Jacobian and an inertia matrix as shown by Orin and Goswami [79]. It is a function that maps the generalized velocities $\mathbf{v}$ to the aforementioned aggregate momenta $\mathbf{h}$ of the robot's centroid.[8] It can be differentiated with respect to time resulting in the following ODE, which is the CD model

$$\frac{\mathrm{d}(\mathbf{A}\mathbf{v})}{\mathrm{d}t} = \mathbf{A}(\mathbf{q})\dot{\mathbf{v}} + \dot{\mathbf{A}}(\mathbf{q})\mathbf{v} = \begin{bmatrix} m\,\mathbf{g} + \sum_{i=1}^{n_{\mathrm{ee}}} {}^{\mathcal{W}}\mathbf{f}_i \\ \sum_{i=1}^{n_{\mathrm{ee}}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r}(\mathbf{q}) - {}^{\mathcal{W}}\mathbf{p}_i(\mathbf{q}) \right) \right) \end{bmatrix}, \tag{2.7}$$

where the change of momentum of the CoM is expressed in terms of the sum of the external forces and the moments acting on the body as shown by Lee and Goswami [82]. The matrix $\mathbf{A} \in \mathbb{R}^{n_{\mathrm{b}}\times n}$ maps the momentum into the frame at the *centroid* ${}^{\mathcal{W}}\mathbf{r}$. Here, $\mathbf{g} \in \mathbb{R}^3$ is the acceleration due to gravity; $m \in \mathbb{R}$ is the total mass of the robot and ${}^{\mathcal{W}}\mathbf{p}_i \in \mathbb{R}^3$ is the position of the ith end-effector.

Dai et al. [83] and Kuindersma et al. [43] used this model with full kinematics for highly dynamic whole-body motion planning for the 68-dimensional version of the Atlas robot. Similarly, Herzog et al. [84] proposed iteratively optimizing for kinematic and momentum trajectories and tested the approach in simulation using the Sarcos humanoid robot on challenging terrain.

[8] Kajita et al. [80] proposed the resolved-momentum control (RMC)—an earlier approach in which matrices that map joint rates into aggregate momenta for multibody dynamic systems were presented. Wensing and Orin [81] showed how this and related quantities can be efficiently computed.

2.3.3 Single-rigid-body dynamics model

It is often assumed that quadrupeds have light legs; hence, the effect of joint velocities on the momentum can be neglected. During the execution of periodic gaits the positions of the joints only marginally deviate from the nominal configuration. As a result, the mass and inertia distributions of the legs can be considered constant and be combined with those of the base. As shown by Winkler [85], Eq. 2.7 can be simplified to give the following linear and angular Newton-Euler equations that represent the single-rigid-body dynamics (SRBD) model

$$m\,{}^{\mathcal{W}}\ddot{\mathbf{r}} = m\,\mathbf{g} + \sum_{i=1}^{n_{\text{ee}}} {}^{\mathcal{W}}\mathbf{f}_i, \tag{2.8a}$$

$${}^{\mathcal{W}}\mathbf{I}(\boldsymbol{\theta})\,{}^{\mathcal{W}}\dot{\boldsymbol{\omega}} + {}^{\mathcal{W}}\boldsymbol{\omega} \times {}^{\mathcal{W}}\mathbf{I}(\boldsymbol{\theta})\,{}^{\mathcal{W}}\boldsymbol{\omega} = \sum_{i=1}^{n_{\text{ee}}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right)\right). \tag{2.8b}$$

The SRBD model is defined in terms of the following quantities and their time derivatives: CoM position ${}^{\mathcal{W}}\mathbf{r}(t) \in \mathbb{R}^3$ and orientation $\boldsymbol{\theta}(t) \in \mathbb{R}^3$, its angular velocity ${}^{\mathcal{W}}\boldsymbol{\omega}(t) \in \mathbb{R}^3$, and the full-body inertia ${}^{\mathcal{W}}\mathbf{I} \in \mathbb{R}^{3\times3}$. This Euclidean-space representation is no longer dependent on the joint configuration $\mathbf{q}_\text{j}$ and allows for a more intuitive definition of constraints. Winkler et al. [86] used it to execute motion on ANYmal and Bledt et al. [87] deployed on the Cheetah robot.

2.3.4 Linear inverted pendulum

One of the simplest dynamics models, first used to study bipedal walking, is the LIP by Kajita et al. [88]

$$m\,{}^{\mathcal{W}}\ddot{\mathbf{r}}_{xy} = \sum_{i=1}^{n_{\text{ee}}} {}^{\mathcal{W}}\mathbf{f}_{i,xy} = \frac{mg}{h}\left({}^{\mathcal{W}}\mathbf{r}_{xy} - {}^{\mathcal{W}}\mathbf{c}_{xy}\right). \tag{2.9}$$

It enables motion in the 2D xy-plane under the assumption that the CoM height h and contact positions are constant, and angular velocity and acceleration are zero. This simple model uses the centre of pressure (CoP) ${}^{\mathcal{W}}\mathbf{c}_{xy} \in \mathbb{R}^2$ and is only suited for coplanar motion (e.g., flat ground) as it does not allow for complex dynamical details to be captured. Extensions of this model which enable more dynamic motions such as running are discussed in Sec. 2.4.1.

2.4 Motion planning

Motion planning involves solving a specific task by finding a path or a trajectory[9], represented by some function of parameters $\mathbf{w}$, between a given initial and some desired final point, which can be defined explicitly or implicitly. These plans can be expressed for individual joints or the end effectors, in *joint*[10] $\mathfrak{J} \subset \mathbb{R}^n$ or *task*[11] $\mathfrak{T} \subset SE(3)$ spaces, respectively. Simple plans involve point-to-point motions; however, motion planning for hybrid systems—that are subjected to many constraints due to interaction with the environment—is a challenging problem, as discussed in Sec. 2.2.3.

2.4.1 Heuristics

In the past, trajectories were designed heuristically, i.e., using pragmatic strategies and templated models. Full and Koditschek [90] hypothesized that complex nonlinear motion can be expressed by a simple model (fewer variables and parameters) while exhibiting a targeted behaviour. In his book, Raibert [91] discusses studies that investigate the principles which enable physical systems to balance themselves. Blickhan [92] proposed using the spring-loaded inverted pendulum (SLIP) model for running and hopping. Kajita et al. [93] used a simple class of trajectories known as potential energy conserving orbit to control the robot's centre of gravity, and formed control laws for the footholds. Westervelt et al. [94] developed the hybrid zero dynamics (HZD) theory which, rather than creating an open-loop trajectory, introduces virtual constraints to reduce the dimensionality of the problem which results in a closed-loop system with an exponentially stable orbit. Many approaches use the CoP or zero-moment point (ZMP) criterion proposed by Kajita et al. [95]. It does not allow for the computation of dynamic motions as it only describes the lateral accelerations of the CoM. It does not consider individual contact forces and assumes the heights

[9] As explained by Siciliano et al. [89], a trajectory is a geometric path on which a timing law is specified, using velocities and/or accelerations at each point.

[10] Joint or *configuration* space involves joint angles and their time derivatives.

[11] Task or *operational* space involves Cartesian positions and their time derivatives.

of the CoM and the footholds to be constant. Caron et al. [96] generalized the ZMP metric to account for frictional constraints and multiple noncoplanar contacts. Dai and Tedrake [97] used limit cycles. Ijspeert [98] reviewed biologically inspired controllers based on central pattern generators (CPGs).

Approaches that leverage simple heuristics in place of more expressive models of the dynamics do not generalize well to arbitrary, demanding situations or diverse tasks[12]. Adaptability to terrain—through optimal foothold selection, force adjustment or base orientation—is typically required. As the problem becomes more complex, defining heuristics becomes increasingly more challenging. As a consequence, other methods for legged locomotion planning have been studied.

2.4.2 Sampling-based planning

One way to simply and efficiently generate trajectories is to use sampling-based planning (SBP) algorithms. Points in the state space are sampled at random or using a strategy. The points are then connected using a graph which enables the path between the start and goal configurations to be found. This is in contrast to search algorithms, e.g., Dijkstra [99] or A* [100], which discretizing the workspace, a process that does not scale well.

In 1987, Donald [101] used SBP to plan the motion of a robot with 6 DoF. In the review, Elbanhawi and Simic [102] note the two most popular algorithms for generating feasible plans: Rapidly exploring Random Tree (RRT) by LaValle [103] and Probabilistic Roadmap Method (PRM) by Kavraki et al. [104]. The solutions obtained using these algorithms are suboptimal. The algorithms lack *completeness*, i.e., they provide no guarantee to find a solution (if one exists); they ensure only *probabilistic* completeness. To improve the quality of the solution, post-processing (e.g., smoothing) and variants of RRT were proposed; LaValle and Kuffner [105] extended RRT to iteratively minimize a cost function for kinodynamic planning. A faster bidirectional extension, leveraging two

[12] A *task* defines a motion-planning problem.

RRTs, known as RRT-Connect was introduced by Kuffner and LaValle [106]. Kingston et al. [107] wrote a review of *constrained* SBP methods that allow for realistic motion planning directly on the constraint manifold. However, when using these methods, no theoretical guarantees about optimality can be made—as proven by Karaman and Frazzoli [108]—who proposed new, asymptotically optimal[13] variants PRM*, Rapidly exploring Random Graph (RRG) and RRT*. Karaman and Frazzoli [109] applied RRT* to a non-economic dynamical system. Gammell et al. [110] proposed to unify graph- and sampling-based planning using batch-informed trees (BIT*). Strub and Gammell [111] presented adaptively informed trees (AIT*) and effort-informed trees (EIT*) which outperformed these earlier methods.

While SBP methods have been used to plan dynamic legged locomotion, they either make strong assumptions or require significant offline precomputation. The approach of Shkolnik et al. [112], which used Boston Dynamics LittleDog, ignored the pitch, vertical position and velocities resulting in a reduction of the sampling dimension which significantly improved search efficiency. Using the same robot, Kalakrishnan et al. [113] achieved a bounding gait on rough terrain by employing the anytime-repairing A* (ARA*) algorithm to search over a precomputed terrain-specific reward grid sampled from a motion primitive library [114]. Additionally, the Dijkstra [99] algorithm was used to plan the path of the body to facilitate these footholds. Winkler et al. [115] used ARA* to search over a tree of possible actions and selecting locally optimal footholds allowing the HyQ quadruped to step over obstacles, gaps and stepping stones.

2.4.3 Trajectory optimization

All TO methods leverage mathematical optimization to find a trajectory. The explicit planning for longer time horizons enables the smooth traversal of difficult terrain. While such trajectories can produce better system performance,

[13] An asymptotically optimal method returns a solution whose cost converges almost surely to the optimum.

they are more difficult to compute, which generally rules out the online use of global methods.

For planning of collision-free motion of a floating-base system or a manipulator, a number of simpler optimization-based methods have been devised. They ignore the dynamics of the system and optimize a cost function to produce a set of configurations. The trajectories are smooth with velocity and acceleration information obtained implicitly, i.e., using finite-difference method (FDM).

Covariant Hamiltonian Optimization for Motion Planning (CHOMP) is a seminal framework by Ratliff et al. [116] which uses a precomputed signed distance field and leverages gradient descent to find trajectories which avoid collisions with the environment; however, it will struggle when initialized with an infeasible trajectory. Stochastic, gradient-free methods such as Stochastic Trajectory Optimization for Motion Planning (STOMP) by Kalakrishnan et al. [117] can optimize nondifferentiable constraints by sampling noisy trajectories. Both methods use fine discretization which is slow relative to methods like Trajectory Optimization for Motion Planning (TRAJOPT) by Schulman et al. [118] which uses fewer states, formulates the problem as a sequential quadratic programming (SQP) problem and employs continuous-time collision checking. It can also incorporate velocity and torque constraints. Toussaint [119] used the augmented Lagrangian method in their framework K-Order Markov Optimization (KOMO).

Some approaches use probabilistic inference and parameterize the trajectories with very few states. Gaussian processes (GPs) can be used to represent trajectories in continuous time with very few states; these include Gaussian process motion planning (GPMP) by Mukadam et al. [120] GPMP2 by Dong et al. [121]. The latter formulates the problem as probabilistic inference exploiting the sparsity of the underlying linear system. Ratliff et al. [122] introduced Riemannian Motion Optimization (RIEMO) which uses Riemannian geometry. RIEMO maps a point to a smooth manifold where obstacles can be represented more naturally such that a geodesic, or a minimum cost path, can be determined more easily.

2.4.3.1 Optimal-control problem

Betts [123] reviews a number of numerical methods for TO, which are used for *optimal control*—or more precisely, to find a control for dynamical systems. Originally used to plan the ascent paths for rockets, this class of local mathematical techniques is now being applied to legged locomotion. TO approaches have been used for short to medium scale motion planning in recent research (Diehl et al. [124], Posa et al. [125]).

A typical optimal-control problem (OCP) is defined by decision variables which are infinite-dimensional vector functions. It involves an objective function of the form

$$\underset{t_0,t_\mathrm{f},\mathbf{x}(t),\mathbf{u}(t)}{\text{minimize}} \quad \underbrace{M\left(t_0,t_\mathrm{f},\mathbf{x}(t_0),\mathbf{x}(t_\mathrm{f})\right)}_{\text{boundary term}} + \underbrace{\int_{t_0}^{t_\mathrm{f}} L\left(t,\mathbf{x}(t),\mathbf{u}(t)\right)\,dt}_{\text{integral term}}. \tag{2.10}$$

The function is in Bolza form as it consists of the Mayer term $M(\cdot)$ and the Lagrange term $L(\cdot)$ (Betts [123]). This continuous-time problem is subject to system dynamics; path constraints; boundary constraints; limits or bounds on the state $\mathbf{x}$ or control $\mathbf{u}$; on the initial $\mathbf{x}(t_0)$ and final $\mathbf{x}(t_\mathrm{f})$ states; and time t (Kelly [126]).

2.4.3.2 Numerical methods for optimal control

Betts [123] emphasized that all numerical methods used to solve OCPs utilize Newton-based iteration to adjust a finite set of variables and avail of implicit[14] Runge-Kutta schemes. As discussed by Diehl et al. [124], there exist three major categories of such methods:

Dynamic programming (DP) It is a global method developed by Bellman [128]. It involves solving a Hamilton-Jacobi-Bellman (HJB)[15] first-order nonlinear PDE which gives a *necessary and sufficient condition* for optimality of a control function $\mathbf{u}$ with respect to a loss function J. The method uses the *principle*

[14] TO is a boundary-value problem (BVP) and not an initial value problem (Betts [123]). For BVP, any scheme becomes effectively, implicit (Ascher et al. [127]).

[15] HJB is a generalization of the Hamilton-Jacobi theory.

of optimality[16]. Solving the HJB equation results in a global, optimal policy $\mathbf{u}^*(t) = \boldsymbol{\pi}^*(\mathbf{x}(t), t)$, a closed-loop solution that provides the optimal control for every point in the state space. The approach is computationally intensive and is primarily intended for low-dimensional problems without constraints.

Indirect methods These are local methods which "optimize then discretize", i.e., they construct the *necessary conditions*[17], which are discretized to produce a two-point BVP. The problem is solved numerically, by computing the gradient of the objective function, to locate the roots of the necessary conditions (Betts [129]). While being accurate, indirect formulation creates practical difficulties. They are nonintuitive to implement due to the requirement to compute the quantities that are not physical, i.e., the Hamiltonian and its partial derivatives with respect to state variables. The methods are particularly difficult to initialize as they rely on being warm-started close to a solution due to relatively small region of convergence. They can also be numerically unstable. When the problem involves algebraic path inequalities, the sequence of constrained and unconstrained subarcs must be guessed. However, robust and efficient variants, based on differential dynamic programming (DDP) (Mayne [130]), have been recently proposed and will be discussed in Sec. 2.4.3.4.

Direct Methods These are local methods which "discretize then optimize", i.e., they transcribe the TO problem into a nonlinear programming (NLP) problem before solving it. Unlike indirect methods, they do not require an analytic expression for the necessary conditions, or for the adjoint variables to be initialized. Instead, they directly optimize the objective function by adjusting state and control variables. The variables are *parameterized* either *sequentially* (control only) or *simultaneously* (state and control). Relative to indirect methods, they

[16] Bellman's principle of optimality is a property of a problem with an optimal solution which can be represented by a number of subproblems with optimal subsolutions.

[17] Necessary conditions are referred to as the Euler-Lagrange equations and include state equations, boundary (transversality) conditions, adjoint equations and control equations. The control equations are based on Pontryagin's Maximum Principle (PMP) (Betts [129]).

are easy to use. Derivatives can be approximated using the finite-difference method (FDM)[18] requiring minimal knowledge of optimal control theory. The constrained-arc sequence does not need to be guessed. Direct methods are more robust since they require a rough initial guess only for state and control variables. They converge better thanks to globalization strategies that improve a merit function.

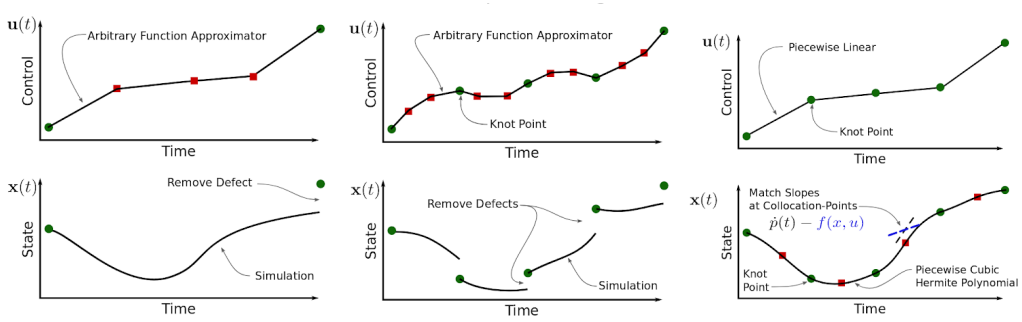
Direct methods are a popular choice for robotics (Kelly [126]). Collocation-based methods were used with a range of torque-controlled walking robots. Boston Dynamics indicated that the company uses a direct method on their state-of-the-art humanoid Atlas and quadruped Spot (Robotics Today [131]).

2.4.3.3 Transcription methods

Both, indirect and direct methods require the infinite-dimensional TO described in Sec. 2.4.3.1 to be *transcribed*[19] into a *parameter optimization* problem. The following transcription methods can be used:

Shooting methods They transcribe and optimize control variables and obtain state variables through simulation via integration (Runge-Kutta schemes); hence, they always implicitly satisfy the dynamics but are poorly conditioned (a small change in the initial controls can greatly change the state trajectory). There are two main categories of these methods. Single shooting (shown in Fig. 2.5a) arguably the simplest technique which is limited to basic problems or those with excellent initialization. It uses a single segment to represent the entire trajectory and a single constraint, known as a defect constraint, is applied on the final state. Multiple shooting (shown in Fig. 2.5b) extends this technique by using multiple segments and defect constraints. The former results in a small, dense program which is hard to solve. The latter produces a large but sparse nonlinear program which can be efficiently simulated in parallel. Multiple shooting methods enable

[18] FDM is slow due to function evaluations and prone to rounding errors.
[19] Transcription can also be referred to as *discretization* of the continuous TO problem.



(a) Single Shooting **(b)** Multiple Shooting **(c)** Collocation

Figure 2.5: Transcription methods used for trajectory optimization (Kelly [132]).

more robust convergence. They are particularly applicable to problems with few path constraints with simple control laws, e.g., space flight.

Collocation This method approximates continuous functions with individual or a collection (splines) of polynomials (shown in Fig. 2.5c), joined at *knot points*, defined by a finite set of coefficients. Collocation methods can be divided into H- and P-methods, based on the means of achieving convergence. The former increases the number of low-order polynomials while the latter increases their order. Direct collocation is a category whose types differ depending on the order of the polynomials. Direct transcription represents controls as constants and states as linear functions which results in defects that need to be removed. Meanwhile, Hermite-Simpson collocation represents the state using cubic polynomials of Hermite form and satisfies the dynamics using Simpson's quadrature. Orthogonal collocation uses (potentially different) high-order polynomials for the spline. Pseudospectral discretization (spectral collocation) uses basis functions in the time domain. The methods enforce the dynamics constraint explicitly via the derivative form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ by matching the slope of the polynomial at chosen *collocation points* or the integral form $\mathbf{x} = \int \mathbf{f}(\mathbf{x}, \mathbf{u})\, dt$ and quadrature rules. Collocation methods are applicable to problems where both state and control must be accurately determined, e.g., robot manipulators.

In the context of hybrid systems that interact with the environment via discrete contacts, there are several merits of direct collocation relative to mul-

tiple shooting. Collocation makes optimizing state variables possible; this enables the discovery of feasible and optimal contact points. Thanks to the polynomial-based parameterization, where the duration of each polynomial can be independently modified, the durations of individual phases can be freely adjusted. In contrast, shooting methods only permit the discretization interval to be changed which has the effect of adjusting the durations of all phases at once. Shooting methods also have a relatively narrow feasible region which makes them difficult to initialize as opposed to collocation-based methods which can be warm-started using expert demonstrations or precomputed trajectories to avoid local minima.

2.4.3.4 Other approaches

Other notable TO approaches that leverage the aforementioned methods and are applicable to walking robots include:

Differential dynamic programming (DDP) Mayne [130] introduced a second-order, indirect method for continuous time, unconstrained problems whose more complete derivation by Jacobson and Mayne [133] resulted in the DDP single-shooting optimal-control algorithm. It locally approximates the dynamics and the cost-to-go as quadratic—which results in quadratic convergence. It uses two stages: a backward pass to generate a sequence of control inputs and a forward pass to compute the state trajectory. Using just the first-order derivatives of the dynamics, Weiwei Li and Emanuel Todorov [134] showed that DDP can be reduced to a much more efficient iterative linear-quadratic regulator (iLQR). This work was extended to a stochastic system via linear-quadratic-Gaussian (LQG) by Todorov and Li [135].

Lin and Arora [136] handled constraints on the states by solving quadratic programming (QP) problems. Tassa et al. [137] proposed Box-DDP which includes box inequality constraints on the controls. Xie et al. [138] proposed C-DDP which accommodates nonlinear inequality constraints on both the states

and the controls. Giftthaler and Buchli [139] introduced a projection method. Howell et al. [140] combined iLQR with an augmented Lagrangian method, by Plancher et al. [141], with an active-set projection method. Farshidian et al. [142] proposed a constrained, parallelized framework which extends a state-of-the-art sequential-linear-quadratic (SLQ) programming algorithm from Sideris and Bobrow [143] to efficiently plan open-loop locomotion trajectories in an MPC fashion.

Giftthaler et al. [144] generalized iLQR by leveraging the multiple shooting method which improved its convergence rate and allowed it to handle infeasible initialization. Building upon it, Mastalli et al. [145] introduced the Contact RObot COntrol by Differential DYnamic programming Library (CROCCODYL) framework and its solver feasibility-driven DDP (FDDP) enabling efficient generation of highly dynamic motions. Mastalli et al. [146] proposed direct-indirect hybridization to combine [137] and [144] resulting in the Box-FDDP algorithm.

Most DDP-based approaches ignore second-order derivatives of the dynamics which are tensorial and expensive to compute; Nganga and Wensing [147] leveraged reverse-mode automatic differentiation (AD) and a modified Recursive-Newton-Euler Algorithm (RNEA) algorithm to compute this information while matching the speed of classical iLQR.

Some approaches investigate accelerating DDP-based model-predictive control (MPC) using a graphics processing unit (GPU) (Plancher and Kuindersma [148], Plancher and Kuindersma [149]). Recently, Plancher et al. [150] computed the necessary gradients of RBD 3 times faster than using an optimized central processing unit (CPU) implementation (Carpentier et al. [151]).

Multiphase methods These methods cater to problems that involve a mixed sequence of continuous and discrete phases, such as a walking robot that makes and breaks contact with the environment. Patterson and Rao [152] developed dedicated software for solving multiple-phase optimal control problems.

Through-contact methods Such methods are specific to hybrid dynamical systems. They treat contact forces as optimization variables and employ complementarity constraints at each grid point. They are discussed by Mordatch et al. [153], Posa and Tedrake [154] and Posa et al. [155].

2.4.3.5 Nonlinear programming

As mentioned above, direct methods result in a nonlinear programming (NLP) problem which consists of discretized algebraic equations of finite dimensions whose decision variables are real numbers. An NLP problem is structured as follows:

$$\underset{\mathbf{w} \in \mathbb{R}^n}{\text{minimize}} \quad J(\mathbf{w}), \tag{2.11a}$$

$$\text{subject to} \quad \mathbf{f}_{\text{eq}}(\mathbf{w}) = \mathbf{0}, \tag{2.11b}$$

$$\mathbf{f}_{\text{in}}(\mathbf{w}) \leq \mathbf{0}, \tag{2.11c}$$

where $\mathbf{w} = (\mathbf{x}(t), \mathbf{u}(t))$ is a vector of decision (optimization) variables; $J(\cdot)$ is a scalar-valued objective function (performance index) being minimized[20]; $\mathbf{f}_{\text{eq}}(\cdot)$ and $\mathbf{f}_{\text{in}}(\cdot)$ are vector-valued functions representing equality and inequality constraints, respectively.

Once the TO problem is transcribed, the NLP process is initialized by solving a simplified version of the full problem. This can be done by ignoring the dynamics, omitting the most restrictive constraints or using a grid which can be used for a heuristic between the initial and final states. Then, the trajectory spline, which is made up of polynomials, is recreated from a set of discrete optimal solutions using interpolation.

Methods for solving NLP problems Nonlinear constrained optimization problems can be efficiently solved using algorithms based on the Newton

[20]When the objective function is concave (a reward), rather than convex (a penalizing cost), the function is *maximized* instead.

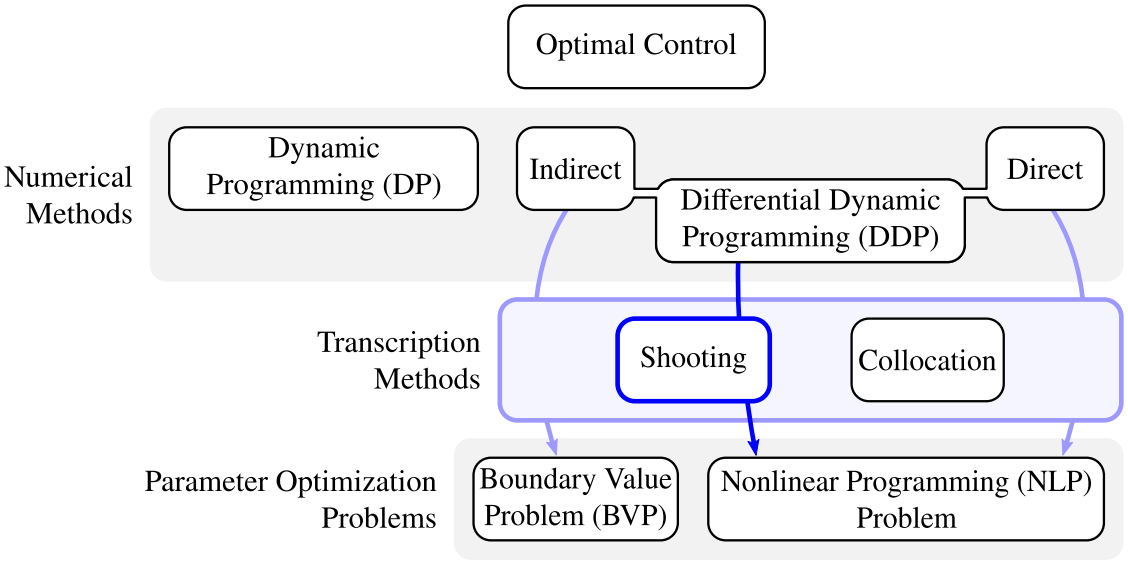


Figure 2.6: An overview of methods for optimal control for legged robot locomotion.

method such as active-set SQP (Nocedal and Wright [156]) or primal-dual interior-point method (IPM)[21] (Mehrotra [157]).

SQP iteratively reformulates the problem as a QP subproblem, i.e., it locally approximates the objective function as a quadratic function and the constraints as affine functions. The process can be computationally intensive. If the active sets of constraints do not change then SQP can be easily initialized with a near-optimal guess to accelerate its convergence.

IPM replaces inequality constraints in Eq. 2.11c by equality constraint and new slack variables[22] which are integrated into the objective function using a logarithmic barrier term. The algorithm's strategy is to solve a sequence of barrier problems. IPM requires the initial values of the variables to be strictly within their bounds, i.e., the 'interior', or it will enforce it by modifying the point according to settable options.

The methods need the first- and second-order partial derivatives[23] of the objective function and the constraints, though the latter is typically efficiently

[21] IPM is also known as a 'barrier' method.
[22] After the replacement, slack variables are the only inequality constraints.
[23] The first- and second-order partial derivatives of a function are known as the Jacobian matrix and Hessian tensor, respectively.

approximated using the limited-memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm.

Termination occurs when Karush-Kuhn-Tucker (KKT) first-order optimality conditions are satisfied to chosen tolerances.

NLP solvers The SQP method is employed by Gill et al. [158] in Sparse Nonlinear Optimizer (SNOPT) commercial solver and the IPM is used by Wächter and Biegler [159] in Interior Point Optimizer (IPOPT) open-source solver, respectively. The Nonlinear Interior point Trust Region Optimization (KNITRO) commercial solver by Byrd et al. [160] offers IPM[24] and active-set methods.

Mittelmann [161] benchmarked and compared various NLP solvers for a variety of problems; the results suggest KNITRO is typically slightly faster than IPOPT while both tend to outperform SNOPT.

Noteworthy, IPM requires a sparse symmetric indefinite linear solver, e.g., MUltifrontal Massively Parallel Sparse direct Solver (MUMPS) by Amestoy et al. [162], PARallel DIrect SOlver (PARDISO) by Schenk et al. [163], or Harwell Subroutine Library (HSL) MA27 and MA57 by Duff [164]. MA57 was recommended for small problems by Tasseff et al. [165] and by Ferrolho et al. [166] for manipulation and locomotion robotics problems. Linear solvers use efficient linear algebra subroutines, e.g., Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK).

2.4.4 Offline planning

Unlike online replanning discussed in Sec. 2.4.5, offline planning is not generally restricted by a tight time budget. It therefore enables the formulated problem to use more accurate dynamics models, or to incorporate costs and constraints.

Nevertheless, in order to find a solution to the problem and do it in a reasonable time, most successful approaches attempt to reduce its complexity by

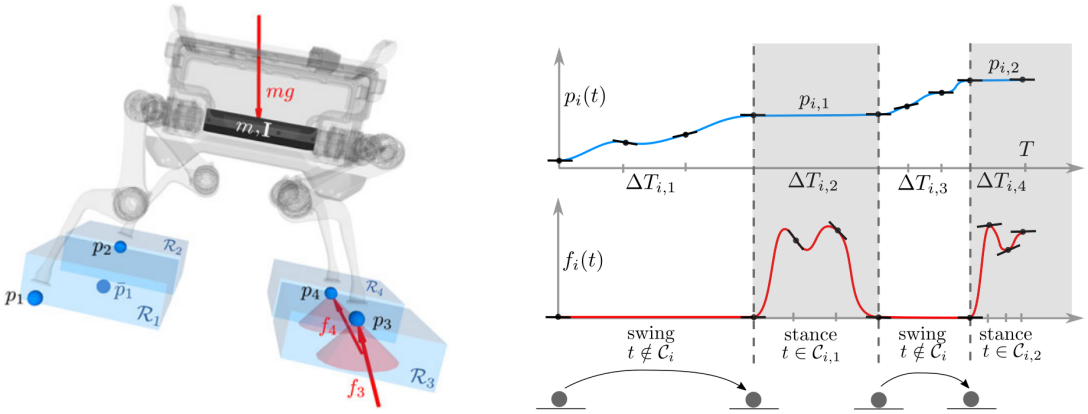
[24] KNITRO offers direct and projected conjugate gradient variants of IPM.

breaking the problem down into simpler subproblems. For instance, they develop means of selecting 'good' footholds or predefine the gait. A key characteristic of legged locomotion optimization problems are discontinuities due to contact transitions of the end-effectors. Two approaches have been used to solve this problem. The first one reformulates contact as a smooth rather than a discontinuous state (Todorov [167], Erez and Todorov [168], Mordatch et al. [153]). This transforms the discontinuous and minima-prone problem into a continuous problem that can be solved with homotopy methods[25]. The second approach defines the problem as a succession of phases separated by the contact state transition of each end-effector.

In their pioneering work, Winkler et al. [86] used direct collocation to formulate a feasibility problem and proposed a novel phase-based parameterization to find phase durations hence enabling the discovery of arbitrary contact sequences. Additionally, the positions of the footsteps were included in the set of decision variables. Using the SRBD model, this nonconvex, high-dimensional problem could be solved quickly. These attributes of this formulation are shown in Fig. 2.7. The proposed method has been open-sourced as part of the Trajectory Optimization for Walking Robots (TOWR) library. While convenient, in its native form, TOWR produces unrealizable trajectories and is missing features needed for simulation or real-world deployment[26]. Medeiros et al. [169] built upon the framework by integrating wheel constraints and including a controller which enabled a real hybrid wheel-legged version of ANYmal B to traverse steep obstacles. Jelavic et al. [170] also used TOWR for a walking excavator. Later, to find global solutions in a more general manner, Jelavic et al. [171] initialized the optimization-based planner using RRT*. Chignoli et al. [49] developed their own planning and control framework for agile quadrupedal motion where a classifier was used to 'warm-start' the optimization. Similarly, Herzog [172]

[25] Homotopy methods solve a problem using homotopy—a continuous transformation from the easy problem to the original (complicated) problem.

[26] **Melon** et al. [1] were the first work to address these issues and combine it with a learning-based initialization scheme.



(a) Robot representation. **(b)** Phase-based parameterization.

Figure 2.7: TOWR [86] uses (a) an SRBD model with box end-effector range-of-motion constraints $\mathcal{R}_i$ for the ith foot and (b) phase-based (swing or stance) parameterization of the foot's motion $p_i(t)$ and force $f_i(t)$ splines which are made up of cubic polynomials. It can create arbitrary gaits by optimizing the phase durations ΔT_i.

combined the generation and execution of the whole-body trajectories into a single framework.

2.4.5 Online replanning

The iterative computation of the desired trajectories, while availing of the sensed or estimated state of the robot in an environment, is known as motion *replanning*. In literature, this approach is often referred to as model-predictive control (MPC), or equivalently receding-horizon control (RHC), when it is achieved using an optimization-based technique (Mayne et al. [173]). While TO is typically thought to be *offline* and slow (e.g., <10 Hz), MPC is *online* and fast (e.g., 10 Hz to 100 Hz). The offline generation of trajectories is very limiting. The computed plan can quickly become infeasible due to tracking errors, inaccurate terrain models or a poor-performing state estimator. Controlling systems in an MPC-fashion gives them reactive properties. They can adapt to emerging terrain or dynamic environment. They become robust to internal and external disturbances, such as unmodelled system dynamics and slippage or perturbations, respectively.

In most cases, the timing and position of the contacts are computed externally, and the optimization only solves for the centroidal motion of robot (Carpentier et al. [174], Mastalli et al. [175], Bellicoso et al. [176]). Ponton et al. [177] relaxed the nonlinear problem and employed mixed-integer programming (MIP) to discover foothold locations, enabling a humanoid robot to recover from external pushes. Focchi et al. [178] used a state machine to generate static walking behaviours on significant slopes. Bellicoso et al. [179] used the LIP model to determine the reference footholds which allowed them to track a desired base velocity. This method has been extended by Bellicoso et al. [176] to update foothold locations online, also using the ZMP criterion.

The majority of currently deployed legged robots already use MPC but, in order to achieve high replanning rates, short time horizon and simplified dynamics models are typically used. Winkler et al. [115] used a 'cart-table' LIP model with the ZMP criterion to plan dynamic motions on terrain for HyQ. Caron and Pham [180] built a framework which optimized all timing variables online thanks to the simpler LIP model. Bellicoso et al. [176] constructed an MPC framework based on the ZMP criterion. Di Carlo et al. [181] simplified the SRBD model to make the problem convex (and hence easier to solve) enabling online replanning on the Cheetah quadruped and demonstrating robust execution of a variety of gaits with a prediction horizon of $0.5\,\mathrm{s}$ computed at $50\,\mathrm{Hz}$. Dai et al. [83] used SLIP with full kinematic and collision models to generate motion plans for an older version of Atlas, a 34-DOFs humanoid robot, using the SNOPT solver. Di Carlo et al. [181] simplified the dynamics of the Cheetah quadruped to make the problem convex and hence easier to solve. Bledt and Kim [182] implemented a regularized predictive controller which simultaneously optimized footstep placements and ground reaction forces.

In 2015, the first real-time execution of a whole-body MPC on a complex dynamic humanoid was demonstrated by Koenemann et al. [183]. Building upon their earlier work, Bellicoso et al. [176] used ZMP constraints (Bellicoso

et al. [179]) with their whole-body control (WBC) (Bellicoso et al. [184]) to execute various gaits in an MPC fashion on a real quadruped. Neunert et al. [185] deployed a nonlinear MPC framework, which used the RBD model, on a real quadruped; it replanned at up to $190\,\mathrm{Hz}$ with a time horizon of $0.5\,\mathrm{s}$. Farshidian et al. [142] developed a DDP-based FASTSLQ planner whose computational complexity with respect to the optimization time horizon is linear $O(n)$, in contrast to the cubic-time complexity $O(n^3)$ of the original sequential quadratic programming (SQP) solver (Sideris and Bobrow [143])—this feature makes it feasible for MPC to be applied to high-dimensional systems. Grandia et al. [186] improved it by integrating frequency-dependent costs (Grandia et al. [187]) and friction cone constraints.

Rather than relying on a WBC to track trajectories produced by a planning algorithm which uses a simplified dynamics model, Mastalli et al. [188] used the full RBD model and their BOX-FDDP solver to plan dynamic trajectories *and* control them directly using the ILQR's Riccati gains.

It is clear that significant progress has been made in the area of online planning with the help of model-based techniques; however, many exciting challenges still exist which can benefit from the use of alternative approaches. Future research needs to focus on long-horizon planning which will enable solving multimodal, temporally extended tasks. In the next section, learning-based methods will be discussed.

2.5 Motion learning

Planning complex, multimodal, long-horizon locomotion strategies from scratch in real time, in the presence of real-world uncertainties, is a high-dimensional and nonconvex problem whose solving demands large computational resources.

To offset the computationally heavy burden, some approaches precompute trajectories—via sampling-based planning (SBP), trajectory optimization (TO), or motion-capture systems—and store them in trajectory libraries (TLS) or using function approximators such as NNs. Machine learning (ML) techniques such

as imitation learning (IL)[27] and reinforcement learning (RL) can be applied to approximate datasets or models.

IL involves reproducing or 'mimicking' behaviours demonstrated by an expert[28] via behaviour cloning (BC), i.e., learning a mapping between states and actions. The learning can happen in either a supervised or an unsupervised fashion.

RL involves exploring and interacting with the environment to maximize a cumulative reward. The approach generalizes well but is typically inefficient. Its variant, inverse reinforcement learning (IRL), can be used for IL.

In Sec. 2.5.1, methods that employ RL and IRL are discussed. In Sec. 2.5.4, data-driven methods for optimal control are reviewed; in particular, the concept of a data-driven initialization of an optimization solver is discussed.

2.5.1 Reinforcement learning

As discussed by Sutton et al. [191], RL is a major subfield of machine learning (ML) with a long history. It involves an experiential framework within which trial-and-error methods or model-based optimization are used to discover and learn a policy which controls the actions of an agent interacting with the environment in a way that maximizes future rewards. RL tries to learn and solve a discrete-time Markov Decision Process (MDP).

Model-free RL offers the ability to automatically discover complex behaviours that are hard to manually define. It has been demonstrated that deep reinforcement learning (DRL) generalizes well to a diverse range of problems like games (Mnih et al. [192], Silver et al. [193]), finance (Deng et al. [194]) and robotics (Schulman et al. [195]). Since on-policy methods like Proximal Policy Optimization (PPO) (Schulman et al. [196]), great progress has been made in

[27] IL is also known as learning from demonstration (LfD) (Schaal [189]) or programming by demonstration (PbD) (Billard et al. [190]).

[28] The demonstrated behaviours are sometimes called 'past (or previous) experiences'.

improving RL's sample efficiency[29]. Faster off-policy methods like Soft Actor-Critic (SAC) (Haarnoja et al. [197]) also exist.

To solve complicated tasks involving dynamic skills, curriculum-based learning was used by Xie et al. [198]. Yin et al. [199] used a sample-efficient Bayesian diversity search (BDS) algorithm to let a simulated humanoid discover high jump strategies, such as the 'Fosbury flop', without prior examples and minimal reward tuning.

Tobin et al. [200] proposed a technique called *domain randomization* as means of bridging the 'reality gap' and facilitating 'sim-to-real' transfer of complex motions to real platforms. Valassakis et al. [201] has showed that rather than randomizing dynamics' parameters, it is sufficient to inject random forces into the simulation.

2.5.2 Locomotion using reinforcement learning

In their seminal work, Hwangbo et al. [202] showed the viability of DRL-based controllers for quadrupeds and the importance of a network which approximates the dynamics of the actuators resulting in the learning of reactive and robust locomotion policies.

Lee et al. [203] used the ANYmal C quadruped to demonstrate traversal of arbitrary terrain using only proprioception. Similarly, Siekmann et al. [204] showed the Cassie biped robustly ascend and descend stairs using only proprioceptive feedback. Rudin et al. [205] accelerated PPO-based DRL, using a parallelized approach on a GPU, enabling such blind locomotion policies to be trained in minutes.

Tsounis et al. [206] used a hierarchical approach to plan different gaits and footholds using exteroceptive measurements. Gangapurwala et al. [207] combined an RL-based foothold planner and corrective feed-forward joint torque policies with a ZMP-based motion planner by Bellicoso et al. [176], which relies on an optimization-based whole-body controller (Bellicoso et al. [184]), to

[29]Sample efficiency refers to the amount of experience—including states, actions and rewards—that is required to optimize a policy.

enable perceptive and reactive traversal of uneven terrain on ANYmal B and C quadrupeds. Miki et al. [208] showed impressive perceptive locomotion on highly irregular and uncertain terrain where proprioception signals can be invalid; the framework is robust thanks to a 'belief state encoder' which leverage exteroceptive data. However, it is still limited to short-horizon planning.

2.5.3 Inverse reinforcement learning

Kalman [209] introduced the idea of inverse optimal control (IOC), also known as inverse reinforcement learning (IRL) (Ng and Russell [210]), which can be used for IL. These approaches work by inferring the objective or reward functions from expert demonstrations. Peng et al. [211] developed methods to enable simulated robots to mimic capture data of athletic motions like running, doing a cartwheel, dancing and doing flips. Brakel et al. [212] used IRL to imitate optimized, long-horizon trajectories produced by TOWR (Winkler et al. [86]). Another interesting direction involves adversarial imitation learning; Ho and Ermon [213] derived a model-free imitation learning method which trains a policy directly from data and achieves expert-level performance, although requiring many samples. Adversarial motion priors (AMP) method, introduced by Peng et al. [214], was used by Vollenweider et al. [215] to enable the hybrid wheeled-legged version of ANYmal C quadruped to transition to and operate in humanoid configuration.

2.5.4 Data-driven optimal control

When using a model-based optimal control (OC) approach, precomputed trajectories can be used to *initialize* or 'warm-start' a solver within the convergence region of a local optimum. Such an 'initial guess' can be quickly refined into a high-quality, feasible and locally optimal solution.

2.5.4.1 Trajectory libraries

A classic strategy for reducing the online computational effort is to use trajectory libraries (TLs) which store precomputed trajectories that can be retrieved based on the similarity of their associated tasks to tasks encountered online.

To accelerate motion planning based on the A* planner, Stolle and Atkeson [216] used TLs which were converted into a policy via a nearest-neighbour search in the state-space to retrieve the closest trajectory segment. In a similar fashion, Stolle et al. [217] worked on legged locomotion on uneven terrain where they extracted feasible footsteps from long trajectories which could later be used to recreate full motion plans. Jetchev and Toussaint [218] used a 'situation descriptor' and a 'trajectory prediction' (Jetchev and Toussaint [219]) to obtain task-space trajectories in new situations.

The main advantage of using a TL is its ability to represent trajectories of different lengths. However, the approach is limited and may not provide optimal solutions as it is heavily reliant on a lookup metric. Because it does not permit interpolating between the stored trajectories, it does not generalize to unfamiliar tasks. It is also inefficient due to an unsustainable memory footprint which also affects how fast a solution can be retrieved.

2.5.4.2 Trajectory regression

An alternative strategy is to fit a regression model to the relationship between tasks and trajectories. Its benefits include not requiring an explicit lookup metric and reduced memory requirements.

Dragan et al. [220] predicted only low-dimensional attributes of trajectories that put them in the basin of attraction of a low-cost minimum.

Levine [221] proposed to use deep and recurrent neural networkss (RNNs) as policies that approximate optimal controllers; this has been achieved by using TO to 'guide' the policy search.

Bency et al. [222] demonstrated that the time complexity of a NN-based approach (RNN) for a manipulation tasks is much better (linear) than using sampling based planners A* (exponential) and RRT (polynomial).

Forte et al. [223] generalized entire expert trajectories using Gaussian process regression (GPR), maintaining their original quality with significant data reduction.

2.5.4.3 Multimodality

Legged locomotion data has a *multimodal distribution*[30]. To represent such data, some approaches used Mixture of Experts (MoE) networks, i.e., a network for each mode. In the field of computer animation, Zhang et al. [224] and Starke et al. [225], used multiple regressors to represent the variety of motions exhibited by dogs and humans. Hong et al. [226] used MPC to refine reconstructed soccer motions in order to ensure that they are physically correct.

In the field of robotics, Lembono et al. [227] used an 'ensemble' or a mix of function approximators—including k-nearest neighbors (k-NN), GPR and Bayesian Gaussian mixture regression (BGMR)[31]—greatly improving success rates. Carius et al. [228] learned—in a sample-efficient way relative to RL—to directly minimize the control Hamiltonian by mimicking multimodal MPC demonstrations of various quadrupedal gaits. Tang and Hauser [229] also used MoE networks to deal with discontinuities in the mapping from tasks to optimal solutions due to distinct drone trajectories belonging to discrete homotopy classes. Merkt et al. [230] used persistent homology to understand the structure of the solution space which enabled a dataset of quadcopter trajectories to be automatically clustered.

To remove the need for multiple networks and discrete categories of actions, conditional variational autoencoders (CVAEs) can be used to express continu-

[30] Quadrupedal locomotion involves making discrete choices (e.g., deciding which foot to lift first or which side to pass an obstacle on) and can be achieved in many distinct styles (e.g., using different gaits) which makes it *multimodal*.

[31] Unlike GPR, BGMR handles discontinuous and multimodal problems.

ous modal variation as shown by Sohn et al. [231]. This approach was used for manipulation tasks by Noseworthy et al. [232] and Lynch et al. [233].

2.5.4.4 Initialization of optimal control solvers

Tassa et al. [234] used a library of local-control trajectories to extend DDP to receding-horizon control (RHC).

Berenson et al. [235] applied a data-driven trajectory initialization scheme in the domain of manipulation to speed up the computation of smooth paths while avoiding obstacles. Mansard et al. [236] used a NN to 'warm-start' non-linear MPC with a state-control trajectory approximation of motion plans for a unmanned aerial vehicle (UAV) generated using PRM allowing the solver to converge in 2 to 5 iterations; this improved upon a 'cold-start' $60\,\%$ failure rate, despite a budget of 50 iterations. Merkt et al. [237] compared various optimization solvers and initialization strategies in terms of success rates, convergence time for a high-dimensional pick-and-place problem using a 38-DoF NASA Valkyrie humanoid [238] in a complex environment. Lembono et al. [227] achieved faster and more frequent convergence when they warm-started a TO method TrajOpt using a learned initialization instead of a linearly interpolated guess. Lembono et al. [239] generated a 'memory of motion', i.e., a database of dynamic, collision-free locomotion, using the HPP Loco3D planner by Tonneau et al. [240] for the Crocoddyl solver by Mastalli et al. [145].

Bledt and Kim [241] developed a framework which automatically determines intuitive heuristics that serve as an initialization for an optimization-based planner capable of producing very agile and high-fidelity locomotion on real platforms.

2.6 Motion control

In the previous sections, we have discussed how motion planning (Sec. 2.4) can be accelerated using machine learning (Sec. 2.5). When feasible and optimal

whole-body trajectories are generated, they must typically[32] be converted by a high-level to joint-level position, velocity and force/torque setpoints using control methods discussed in Sec. 2.6.1, Sec. 2.6.2 and Sec. 2.6.3. All approaches require low-level controllers which ensure that the actuators track the joint-level setpoints as discussed in Sec. 2.6.4.

Proportional-derivative-integral (PID) control has been successfully used for precise but stiff motion control. For walking robots in particular, compliant interaction with the environment is essential due to inevitable sensing discrepancies and uncertainties as discussed by Escande et al. [242] and Neuhaus et al. [243]. Thanks to the series elastic actuators (SEAs), the torque at each joint can be directly controlled.

2.6.1 Virtual-model control

Pratt et al. [244] describes virtual-model control (VMC), an intuitive, straightforward, fast-to-compute approach with compliant characteristics. It computes the contact forces using the centroidal momentum model and converts them into joint torques. The swing motion is tracked using a separate PD controller. It has been used by Boaventura et al. [245] and Gehring et al. [246] on hydraulic and electric quadrupeds, respectively. Chen et al. [247] discusses the numerous shortcomings of the method that result in low control performance including response lag, force/torque distribution when in a 2-point contact, and disturbance caused by end-effector phase change.

2.6.2 Inverse-dynamics control

Mistry et al. [248] introduced the floating-base inverse-dynamics approach[33] which allows the robot to track the desired base trajectory while allowing some *active compliance* in the legs. It has been used by Kalakrishnan et al. [113] to

[32] As discussed in Sec. 2.4.5, Mastalli et al. [188] proposed a DDP-based approach which generates trajectories and Riccati gains that enable direct execution.

[33] Righetti et al. [249] showed that the operation-space control by Sentis and Khatib [250], [251] was equivalent to the independently developed inverse-dynamics control by Mistry et al. [248]

design a controller which enabled the LittleDog quadruped to traverse uneven terrain. Buchli et al. [252] built upon this method by adding a predictive force controller.

Alternatively, it is common to optimize for the ground-reaction forces to control the wrench on the base as shown by Focchi et al. [178]. Similarly, Farshidian et al. [253] tracked the planned contact forces directly.

2.6.3 Whole-body control

The state-of-the-art approaches from Bellicoso et al. [184], Fahmi et al. [254] and Xin et al. [255] all use whole-body, hierarchical[34] optimization involving quadratic programming and the RBD model. The approach generates joint accelerations, joint torques and contact forces. Mistry et al. [248] and Mansard [257] proposed methods of reducing the dimensionality of the formulated problem.

The controller by Bellicoso et al. [184] enables the tracking of Cartesian base and feet positions, velocities and accelerations at $400\,\mathrm{Hz}$ by solving a cascade of QP problems. The problems of higher-priority constraints which ensure that the rigid-body dynamics defined in Eq. 2.3 are fulfilled, the forces stay within the friction pyramids, the feet remain stationary during contact, and the torque limits are respected. The lower-priority costs are applied to the tracking of the base position and orientation, as well as swing motions. The hierarchy is strictly enforced and computationally efficient as solutions to successive lower-priority tasks are searched for only in the null space of higher-priority tasks.

2.6.4 Low-level control

Each actuator is directly controlled by a simple PID controller which tracks the joint position, velocity and acceleration (or torque) setpoints at $2.5\,\mathrm{kHz}$ to $10\,\mathrm{kHz}$. Its gains are generally dependent on a logic known as a contact-state machine. Lower gains are used for stance legs to ensure some compliance and higher gains are used for the swinging legs to enable precise tracking.

[34] In hierarchical optimization, the satisfaction of certain tasks or constraints can be weighted or strictly prioritized (Hutter et al. [256]).

2.7 Conclusion

This chapter gave a thorough summary of the major components associated with locomotion planning including an overview of legged robotics and the posed challenges (Sec. 2.2), system modelling (Sec. 2.3), planning methods (Sec. 2.4), learning techniques (Sec. 2.5) and control algorithms (Sec. 2.6).

Until recently, legged locomotion focused on blind, short-horizon trajectories, based primarily on simple, low-dimensional, LIP dynamics models, ZMP criteria and heuristic foot-placement strategies. In 2018, Winkler et al. [86] proposed to formulate a simplified feasibility problem using TO based on direct collocation. The formative idea did not impose a fixed gait sequence or footholds. Instead, a nonlinear solver iteratively determined what set of parameters and decisions would yield long-horizon dynamic motion on uneven terrain. While promising, the developed framework TOWR was not intended for online use and the *viability*[35] of the generated trajectories were not sufficiently tested in experiment. The approach relied on certain assumptions and simplified models which deviate from the real world.

Reflecting on the literature reviewed in this chapter, we leverage many of the aforementioned classical and state-of-the-art approaches. We chose the approach of Winkler et al. [86] to optimize end-effector phase durations independently of base motion. We transcribe our problem into a sparse NLP program using direct collocation and solve it using an IPM method via the IPOPT solver (Wächter and Biegler [159]). Collocation allows us to initialize both state and control decision variables and obtain fast local convergence (Diehl et al. [124]). We parameterize the variables with the values and derivatives of cubic polynomials (in Hermite form). This results in a set number of optimization variables enabling receding-horizon deployment and behaviour cloning for learning-based initialization. The trajectories that satisfy the dynamics to third-order enable accurate and precise tracking by the adopted whole-body controller

[35] In the context of this thesis, *viable* means executable by a real hardware system.

(Bellicoso et al. [184]). The controller lets us use simplified models during planning and, to a degree, makes the execution robust to sensing uncertainties and external disturbances. In order to accelerate the optimization we use IL. Specifically, we create a dataset of optimal motions, which we efficiently represent using a neural network and use it to 'warm-start' the solver.

3

Costs and constraints for reliable trajectories

This chapter includes elements of the following publications:

[1] **O. Melon**, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1410–1416

[2] **O. Melon**, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811

This chapter describes the use of trajectory optimization based on direct collocation and the SRBD model for the planning of long-horizon whole-body trajectories for quadrupedal robots. It outlines the development of analytical costs which exploit the structure of the problem and kinematic constraints that depend on the sensed terrain. The reliability of the resulting trajectories is validated in realistic simulators and on hardware using an adapted whole-body controller. While focused primarily on open-loop planning, some references are made to the receding-horizon variant of the framework presented in Chapter 4.

3.1 Introduction

Legged robots have begun to be applied in a range of application domains where advanced mobility is required. The key benefit of such machines is their flexibility in operating on uneven terrain. This goal implies increased complexity as legged robots are articulated systems with high-dimensional kinematics and dynamics that continually change their contact with the environment. Other hurdles arise due to real-world sensing, actuation limits, state estimation and perturbations. As a result, legged robots need an all-round motion planning approach to efficiently and robustly perform their tasks.

The generation of dynamic motions for these platforms is an open research problem with recent advances focusing on optimization-based approaches. Longer trajectories can produce better system performance, but are more difficult to compute due to inherent nonlinearities, ruling out the online use of global methods. Gradient descent is less demanding but can get stuck in poor local minima due to the strong nonconvexity of the problem. Thus, local optimization is appealing only if the constraints, costs, and the initial guess of the nonlinear program can all be specified effectively.

As shown in Fig. 1.3, by enhancing the formulation of the optimization problem with costs and constraints, we demonstrate a quadrupedal robot robustly executing long-horizon trajectories on uneven terrain.

3.1.1 Contributions

- Extension of the TOWR legged robot motion optimization framework (Winkler et al. [86]) with analytical costs to create dynamic trajectories that can be reliably tracked by a controller.

- The development of terrain-dependent range-of-motion constraints for the pose of the robot's base.

- The use of superquadrics as end-effector range-of-motion constraints which, relative to a simple box constraint, better approximate the real kinematic limits of a quadruped.

- Experiments in which the real ANYmal B and C robots trot or dynamically walk across flat or uneven terrain while using only onboard sensors for state estimation, verifying the validity of our approach and its suitability for real hardware.

3.2 Related works

Trajectory optimization (TO) approaches have been used for short to medium scale motion planning in recent research (Diehl et al. [124], Posa et al. [125]). They usually employ direct methods to transcribe a continuous, infinite-dimensional problem as a discrete, parametrized NLP (Betts [129]).

Legged locomotion optimization problems are subject to discontinuities due to contact transitions of the end-effectors. Two approaches have been used to recast it as a continuous problem. The first reformulates contact as a smooth rather than a discontinuous state (Todorov [167], Erez and Todorov [168], Mordatch et al. [153]). This transforms the discontinuous and minima-prone problem into a continuous problem that can be solved with homotopic methods. While it is easier to solve, it requires careful tuning and produces less realistic motions.

The second approach defines the problem as a succession of phases separated by the contact state transition of each end-effector. In most cases, the timing and position of the contacts are computed externally, and the optimization only solves for the centroidal motion of a robot (Ponton et al. [177], Carpentier et al. [174], Mastalli et al. [175], Bellicoso et al. [176]). In Winkler et al. [86], the contact state of each leg is considered separately, which theoretically allows it to generate different gaits when optimizing the duration of each phase. Moreover, the positions of the footsteps are included in the set of optimized variables. Unlike

the previously mentioned approaches, this formulation allows a highly nonconvex shape of the problem to be solved by recasting it as a feasibility problem.

In order to create more practical trajectories, it is typical to incorporate some measure of performance, or a cost, into the formulation of the optimization problem. Flash and Hogan [258] minimized the integral of the rate of change of acceleration of a hand. Uno et al. [259] minimized the integrated torque of a multijoint arm. Harris and Wolpert [260] minimized summed positional variance across repeated eye and arm movements. Similar costs were used by Kajita et al. [95], Ponton et al. [177] and Carpentier et al. [174] for legged locomotion. Grandia et al. [187] penalized high-frequency components in the control input to produce smoother motion.

The process of designing sophisticated cost functions, to achieve complex behaviours, requires a good understanding of the dynamics of the physical system. Meanwhile, the addition of cost terms increases the overall computation time; in Chapter 5, a potential solution to these problems is proposed which uses ML-based techniques to provide a near-optimal initial guess to the solver.

3.3 Methodology

In this section we present our motion planning and execution approach. We first review the TOWR trajectory optimization package, which uses a simplified dynamics model of the robot to plan motions for its legs and CoM between initial and final configurations. We present improvements and adaptions that help create trajectories which are more suitable for the real robot. We then overview the tracking whole-body controller that we use to compute the joint torques necessary to execute these generated motions and provide an illustrative evaluation.

3.3.1 Problem formulation

The paper extends the work of Winkler et al. [86] and their open-source library TOWR. The software is capable of producing highly dynamic trajectories for

a range of walking robots (hoppers, bipeds and quadrupeds) by discretizing the continuous locomotion problem into a numerically solvable NLP problem using a collocation-based transcription method. In this case, trajectories are constructed as splines of N cubic[1] polynomials, where each 1D polynomial and its first- and second-order time derivatives

$$x_i(t) = \; a + \; bt + \; ct^2 + dt^3, \tag{3.1a}$$

$$\dot{x}_i(t) = \; b + 2ct + 3dt^2, \tag{3.1b}$$

$$\ddot{x}_i(t) = 2c + 6dt, \tag{3.1c}$$

are defined by the coefficients (a, b, c, d). However, when in Hermite form, each 1D cubic polynomial $x_i(t; \Delta T, x_0, x_1, \dot{x}_0, \dot{x}_1)$ is parameterized by 5 decision variables, i.e., the duration of the polynomial (ΔT), and the values (x_0, x_1) and the derivatives $(\dot{x}_0, \dot{x}_1)$ at its start and end points called *nodes*, as follows:

$$a = \; x_0, \tag{3.2a}$$

$$b = \; \dot{x}_0, \tag{3.2b}$$

$$c = \; -\Delta T^{-2}\Big[3\,(x_0 - x_1) + \Delta T(2\dot{x}_0 + \dot{x}_1)\Big], \tag{3.2c}$$

$$d = \quad \Delta T^{-3}\Big[2\,(x_0 - x_1) + \Delta T(\;\dot{x}_0 + \dot{x}_1)\Big]. \tag{3.2d}$$

The approach uses the SRBD model described by Eq. 2.8, which assumes the system's mass is concentrated at a point and all of its inertias are combined into a single, unchanging inertia (computed for some nominal configuration). The model requires the values and derivatives[2] of the base linear position ${}^{\mathcal{W}}\mathbf{r}(t) \in \mathbb{R}^3$, orientation[3] $\boldsymbol{\theta}(t) \in \mathbb{R}^3$, end-effector linear positions ${}^{\mathcal{W}}\mathbf{p}(t) \in \mathbb{R}^3$ and ground reaction forces (GRFs) ${}^{\mathcal{W}}\mathbf{f}(t) \in \mathbb{R}^3$. Yet, because of the parameterization, only the vari-

[1] Having 4 DoFs, cubic polynomials demonstrate C^2 (parametric) continuity, enabling them to be continuous, smooth and have the same curvature at the meeting points. The cubic spline gives the 'best' trajectory between two points as it minimizes the average acceleration $\int |\ddot{x}(t)|^2 dt$, an affine function in t given in Eq. 3.1c.

[2] The second-order derivatives, i.e., accelerations, are not directly optimized; instead, they are computed using Eq. 3.1c.

[3] The angular information is parameterized using Euler angles $\boldsymbol{\theta}(t)$ and rates $\dot{\boldsymbol{\theta}}(t)$ which are then used to compute angular velocity ${}^{\mathcal{W}}\boldsymbol{\omega}(t) \in \mathbb{R}^3$ and acceleration ${}^{\mathcal{W}}\dot{\boldsymbol{\omega}}(t) \in \mathbb{R}^3$ as shown in Winkler et al. [86].

type	category	variables w		description
base	base-motion	${}^{w}\mathbf{r}$	${}^{w}\dot{\mathbf{r}}$	linear position and velocity
		$\boldsymbol{\theta}$	$\dot{\boldsymbol{\theta}}$	orientation and its derivative
ee	foot-motion	${}^{w}\mathbf{p}$	${}^{w}\dot{\mathbf{p}}$	linear positions and velocities
	forces	${}^{w}\mathbf{f}$	${}^{w}\dot{\mathbf{f}}$	forces and derivatives
	durations	$\mathbf{\Delta T}$		phase durations

Table 3.1: The optimization variables used in TOWR. These sets of variables include the values and derivatives at the start and end points of the polynomials that make up the trajectory splines.

ables $\mathbf{w}$ listed in Table 3.1 are optimized; their number can be computed using

$$\begin{aligned} \#\mathbf{w} = (3+3+3+3)n_{\text{nodes}}^{\text{base}} & \quad \text{base-motion} \\ + n_{\text{feet}}(n_{\text{st}} - 1 + n_{\text{sw}}) & \quad \text{durations} \\ + n_{\text{feet}}[3n_{\text{st}} + (3+2)n_{\text{sw}}] & \quad \text{foot-motion} \\ + n_{\text{feet}}(3+3)n_{\text{nodes}}^{\text{force/foot/st}} n_{\text{st}} & \quad \text{forces.} \end{aligned} \tag{3.3}$$

The number of base nodes $n_{\text{nodes}}^{\text{base}}$ is calculated using

$$n_{\text{nodes}}^{\text{base}} = 1 + \text{round}(\Delta T_{\text{H}}/dt_{\text{base-poly}}) \tag{3.4}$$

where ΔT_{H} is the duration of the optimization horizon and $dt_{\text{base-poly}}$ is the time step of the base polynomials. For a quadruped, the number of legs n_{feet} is 4. The motion of each leg consists of a number of stance n_{st} and swing n_{sw} phases. The stance phase is defined by the linear position $\mathbf{p}_i(t) \in \mathbb{R}^3$ values, while the swing phase has both linear position $\mathbf{p}_i(t) \in \mathbb{R}^3$ and linear velocity $\dot{\mathbf{p}}_i(t) \in \mathbb{R}^2$ values. The velocity in the z-dimension is an implicit constraint set to 0.

The number of force nodes per foot per stance $n_{\text{nodes}}^{\text{force/foot/st}}$ is generally 3 as forces are constrained to $\mathbf{0}$ at the start and end of each *swing* phase. For every mid-motion stance phase, this number further decreases to 2 as the forces at the start and end are $\mathbf{0}$.

Fig. 3.1 shows these splines for some of the problem variables. The base trajectory is discretized using a fixed timestep $dt_{\text{base-poly}}$ ($0.1\,\text{s}$), while the feet trajectories and contact forces are discretized with a fixed number of polynomials

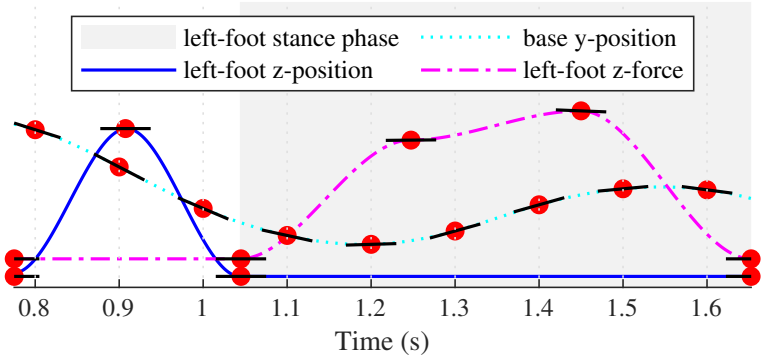


Figure 3.1: An illustrative sample of the trajectories computed by the numerical optimizer, for the left-front leg of a trotting quadruped. Red dots and black lines represent values and derivatives at optimization nodes, interpolated using cubic splines. Each of the polynomials that make up the base trajectory spline is 0.1 s long.

per phase (2 for the swing trajectory, 3 for the stance forces). Therefore, the number of variables for the feet and contact forces varies with the number of steps, while the number of base-related variables depends on the time horizon.

The formulation of the locomotion problem implicitly constrains some of the variables:

- Forces are set to zeros during swing phases.
- Derivatives of the forces and feet positions are set to zeros at the transitions between swing and stance phases.
- A swing node is the highest point of a swing trajectory and its z-dimension derivative is set to zero.
- Feet are fixed in place during a stance phase.

Consequently, Eq. 3.3 can be reduced to

$$\#\mathbf{w} = 12\Delta T_{\mathrm{H}}/dt_{\text{base-poly}} + 20S + \underbrace{2S}_{\text{if phase durations are optimized}} + 120, \tag{3.5}$$

where S is the number of steps.

The nonlinear problem is then transferred to a primal-dual interior-point method solver (IPOPT by Wächter and Biegler [159]) which searches for a solution respecting the following explicit constraints:

- Dynamics of the system (modelled as SRBD).
- Kinematic limits (the positions of feet, relative to the base, constrained within a 3D convex hull, e.g., a box).
- Maximum contact forces and friction pyramids.
- Feet at terrain height during the stance phase.

Since the original TOWR formulation poses only a feasibility problem, the NLP solver generally converges quickly (in less than 40 iterations). While the resulting motion plans are promising, realizing them on hardware is difficult. Oscillation and strong body rolling motions are common while footstep placement and leg terrain clearance are not considered. In this work, we tackle these issues as described in Sec. 3.3.2.

3.3.2 Analytical costs and kinematic constraints

Planning long-horizon, whole-body trajectories for legged robots with complex, nonlinear dynamics and discrete contacts on uneven terrain is a challenging research problem. These motion plans need to be feasible (respect the laws of physics), they should be optimal (facilitate sustained, energy-efficient motion) and they must be viable (executable by a real hardware system). State-of-the-art optimization-based approaches use high-fidelity models with carefully designed constraints and costs which steer the solver towards desirable solutions; without them, the outcome may not be optimal or even viable.

The addition of costs and constraints increases the optimization time. However, as shown in Table 3.2, the increase in the per-iteration time is rather insignificant; instead, it is primarily due to a higher number of optimization iterations which are required for the solver to converge to an acceptable solution. Chapter 5 will describe the use of machine learning to provide an efficient optimization seed which offsets the extra computation time.

3.3.2.1 Smoothing trajectories using integral costs

To generate more conservative trajectories for motions such as the one shown in Fig. 1.3, we extended TOWR with analytically derived costs. A set of i costs $J_i(t)$ are scaled by weights ω_i, to give a total cost of $J(t) = \sum \omega_i J_i(t)$.

Integral costs, which have the effect of minimizing the area between the parameterizing polynomials, were added to reduce linear velocity in the z-dimension and angular velocities of the base, to penalize the magnitude of the ground-reaction forces and their derivatives, and to maintain the desired magnitude of the normal force. The generic cost was

$$J_i(t) = \int_0^T \Big(\omega_{\text{err},i}\big(x_i(t) - x_{\text{ref},i}(t)\big)^2 + \omega_{\text{der},i}\,\dot{x}_i(t)^2\Big)dt, \tag{3.6}$$

where $x_i(t)$ is the optimized polynomial, $\omega_{\text{err},i}$ is the weight on the error between the polynomial and the reference value $x_{\text{ref},i}(t)$, and $\omega_{\text{der},i}$ is the weight on the derivative of $x_i(t)$.[4] These costs and their partial derivatives (with respect to the nodes) were derived analytically using Eq. 3.1 and Eq. 3.2 as shown in Appendix A.1.

3.3.2.2 Terrain-based foothold selection

We aim to generate dynamic motions to traverse slopes, steps and stairs. The original TOWR implementation does not take into account swing collisions between spline nodes. To resolve this, we add kinematic constraints to the base of the robot to enforce that it remains a certain distance above the feet; this eliminates the possibility of collision between the base and the ground. To prevent the solver from creating trajectories that pass through or collide with step edges, we added a cost to discourage the robot from selecting footsteps close to these step edges. The selected cost is a differentiable Gaussian function

$$J_{\text{G}}(t) = \sum \exp\left(-\frac{x_{\text{s}}(t)^2}{2\sigma^2}\right) \tag{3.7}$$

[4] In this work, we omitted $\dot{x}_{\text{ref},i}(t)$ in Eq. 3.6 for simplicity because (in the case of our chosen costs) it was always equal to zero. It would be required if a non-constant $x_{\text{ref},i}(t)$ were simultaneously minimized.

where x_{s} represents the perpendicular distance from each footstep to the edge of the stair and $\sigma^2 \in \mathbb{R}_{>0}$ is the variance. The value of σ is picked to penalize footholds placed close to the edge of a step[5]. Footholds positioned farther are unaffected by the function. The cost can act as a barrier function prohibiting the solver from moving a foothold from one side of the step to the other. However, in practice, if the value of J_{G} is low relative to the overall cost, its effectiveness will be reduced. In the case of a discontinuous step, the ability of the solver to move a foothold from one height level to another is primarily restricted by the terrain constraint (rather than the cost). Smoothing methods or terrain-aware initialization schemes can be used to tackle this nonconvexity.

To ensure that collisions with terrain are avoided, both an inequality constraint and a cost are applied to the height of the swinging feet. The constraint ensures that the swing node is at least a certain distance[6] above each of the adjacent stance nodes while the cost minimizes the swing height to prevent large leg motions that would create angular momentum on the real system, essentially acting as a soft constraint.

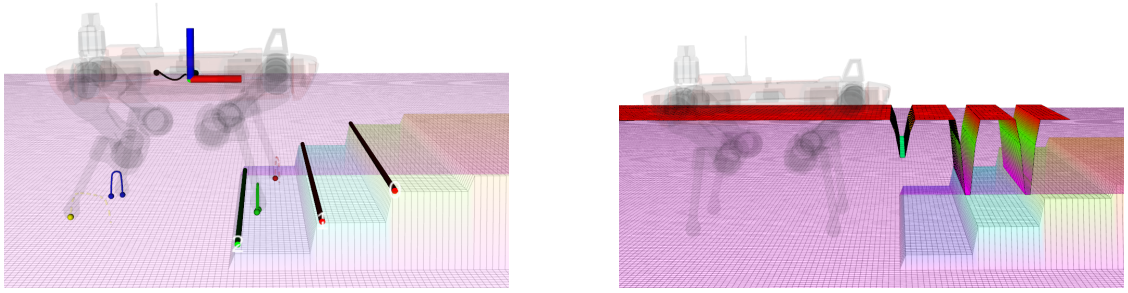
In this work, we identify edges using AprilTags or perform Canny edge detection via the OpenCV library as demonstrated in Fig. 3.2a. Alternatively, we process the elevation map and compute a general cost map shown in Fig. 3.2b. It simplifies the problem of robust detection[7], computation of the perpendicular distances x_{s} and the memorization of the locations of the edges.

3.3.2.3 End-effector range-of-motion constraint

In the latter stages of these studies, we employed a 3D geometric shape known as a *superquadric*,

$$\left|\frac{x}{A}\right|^a + \left|\frac{y}{B}\right|^b + \left|\frac{z}{C}\right|^c = 1 \tag{3.8}$$

[5]We set the value so that the symmetric Gaussian cost cuts off at $\pm 7\,\mathrm{cm}$ from the edge.
[6]We typically use between $5\,\mathrm{cm}$ to $10\,\mathrm{cm}$ for the swing height constraint minimum bound.
[7]The edges must be detected quickly and uniquely, and their poses should remain unchanged.



(a) Detected edges **(b)** Costmap

Figure 3.2: Heightmap processing for foothold selection resulting in (a) detected edges which can be used with the Gaussian edge avoidance cost computed using Eq. 3.7, and (b) a costmap for general foothold penalization. In this image, the costmap is inverted and offset to avoid obstructing the view of the stairs.

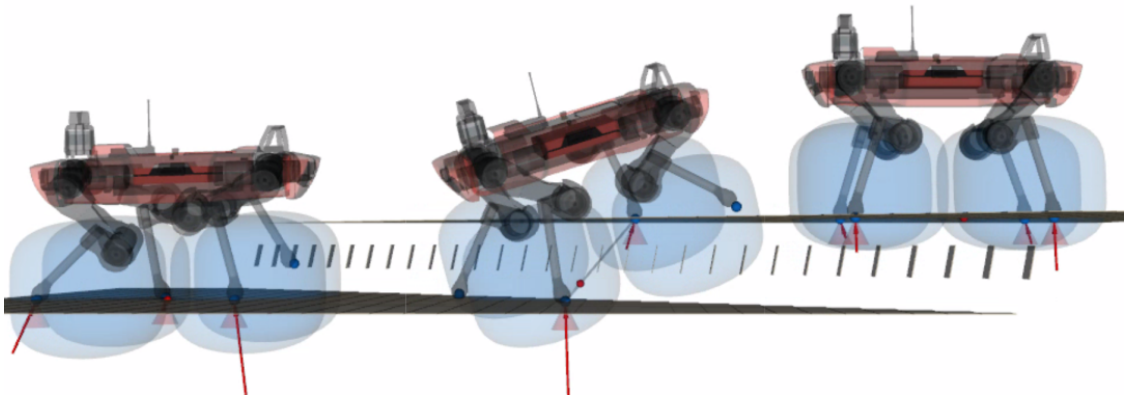


Figure 3.3: Using superquadrics (blue shapes) as end-effector range-of-motion constraints to climb a 0.3 m step oriented at 45°.

whose curvature is defined by the exponents $\{a, b, c\}$ and dimensions are specified by the scalings $\{A, B, C\}$ as shown in Fig. A.1. It can be used as an end-effector range-of-motion constraint forcing a foothold $\mathbf{p}_i(t)$ to lie inside the superquadric

$$\left|\frac{p_{i,x}(t)}{A}\right|^a + \left|\frac{p_{i,y}(t)}{B}\right|^b + \left|\frac{p_{i,z}(t)}{C}\right|^c \leq 1. \tag{3.9}$$

This shape conveys a more natural range of foothold choices and avoids unwanted behaviours that a box shape can promote, i.e., yawing to maximize the length of a stride.

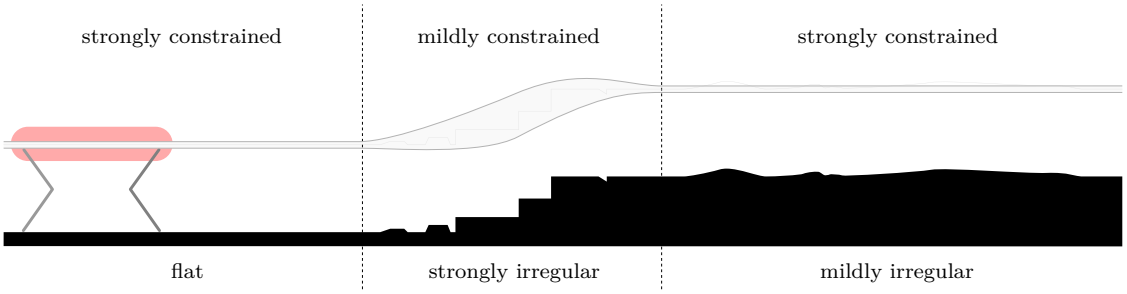


Figure 3.4: A conceptual diagram of a terrain-dependent base pose constraint.

3.3.2.4 Terrain-dependent base pose constraint

In order to generate trajectories that exhibit advantageous qualities—such as maintaining a certain position and orientation relative to the terrain—we leverage sensed terrain information in the form of a 2.5D elevation map (heightmap) to adapt the base motion. This has been achieved by 'terrain simplification', i.e., applying a directional Gaussian blurring to the preprocessed heightmap. To improve the speed of the process, the blurring occurs in a parallel thread and is applied to a subsection of the heightmap. We use the blurred output within the framework to formulate an inequality constraint which enforces that the optimized robot pose is within predefined limits allowing for only minor deviations from nominal values. This concept is shown in Fig 3.4.

To better understand the workings of the proposed method, consider a 2.5D heightmap $\mathbf{I}_{\text{full}}$ and its relevant part $\mathbf{I}_{\text{part}} \in \mathbb{R}^{k_{\mathbf{I},x} \times k_{\mathbf{I},y}}$ whose dimensions $k_{\mathbf{I},x} = l_x/r$ and $k_{\mathbf{I},y} = l_y/r$ are determined from the desired lengths, l_x and l_y, and resolution r at which it should be discretized. A directional Gaussian blur is applied using OpenCV's filter2d function. It correlates[8] $\mathbf{I}_{\text{part}}$ with the normalized, directional Gaussian kernel $\hat{\mathbf{G}}_{\text{dir}} = \mathbf{G}_{\text{dir}} / \left(\sum_{x=1}^{k_\mathbf{G}} \sum_{y=1}^{k_\mathbf{G}} \mathbf{G}_{\text{dir}}(x,y) \right) \in \mathbb{R}^{k_\mathbf{G} \times k_\mathbf{G}}$ to produce $\mathbf{I}_{\text{blur}} \in \mathbb{R}^{k_{\mathbf{I},x} \times k_{\mathbf{I},y}}$ whose entries can be computed as follows

$$\mathbf{I}_{\text{blur}}(x,y) = \sum_{\substack{0 \le x' < k_\mathbf{G} \\ 0 \le y' < k_\mathbf{G}}} \hat{\mathbf{G}}_{\text{dir}}(x',y')\, \mathbf{I}_{\text{part}}(x + x' - a_x,\ y + y' - a_y), \tag{3.10}$$

[8] *Correlation* is a process which determines similarity between two signals (matrices). It resembles *convolution* except one of the signals (matrices) is flipped row- and column-wise.

where $a_x = -1$ and $a_y = -1$ are the anchor points that specify the relative position of a point with the kernel, i.e., the centre[9]. The directional Gaussian kernel $\mathbf{G}_{\text{dir}} \in \mathbb{R}^{k_\mathbf{G} \times k_\mathbf{G}}$ is obtained using OpenCV's warpAffine and getRotationMatrix2d functions which apply an affine transformation, i.e., rotation. The latter produces a rotation matrix $\mathbf{R} \in \mathbb{R}^{2\times 3}$ from the estimated yaw angle of the robot's base ψ, a scaling factor $k = 1$ and the centre point (c_x, c_y), where $c_x = c_y = (k_\mathbf{G} - 1)/2$. It is of the form

$$\mathbf{R} = \begin{bmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \end{bmatrix} = \begin{bmatrix} k_cos\psi & k_sin\psi & (1 - k_cos\psi)c_x - (k_sin\psi)c_y \\ -k_sin\psi & k_cos\psi & (k_cos\psi)c_x + (1 - k_cos\psi)c_y \end{bmatrix}. \quad (3.11)$$

The matrix can then be used to transform existing entries of the pre-directional Gaussian kernel $\mathbf{G}_{\text{pre-dir}} \in \mathbb{R}^{k_\mathbf{G} \times k_\mathbf{G}}$ as shown below

$$\mathbf{G}_{\text{dir}}(x, y) = \mathbf{G}_{\text{pre-dir}}(R_{00}x + R_{01}y + R_{02}, R_{10}x + R_{11}y + R_{12}). \quad (3.12)$$

Its elements are zero except for the middle row which is the Gaussian kernel $\mathbf{G} \in \mathbb{R}^{k_\mathbf{G} \times 1}$

$$\mathbf{G}_{\text{pre-dir}} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \mathbf{G} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}^\intercal. \quad (3.13)$$

The i^{th} element of $\mathbf{G}$ is

$$G_i = \alpha e^{-[i-(k_\mathbf{G}-1)/2]^2/2\sigma^2}. \quad (3.14)$$

The scaling factor α is chosen so that $\sum_{i=1}^{k_\mathbf{G}} G_i = 1$. The aperture size $k_\mathbf{G} \in \mathbb{Z}^+$ must be odd[10]. The Gaussian standard deviation σ is calculated as follows,

$$\sigma = 0.3\left[(k_\mathbf{G} - 1)/2 - 1\right] + 0.8. \quad (3.15)$$

The directional derivative $D_\mathbf{u}\mathbf{I}_{\text{blur}} \in \mathbb{R}^{k_{\mathbf{I},x} \times k_{\mathbf{I},y}}$, in the direction of the unit vector $\mathbf{u} = (\cos(\psi)\ \sin(\psi))$, can be computed from the partial derivatives as follows

$$D_\mathbf{u}\mathbf{I}_{\text{blur}} = \nabla_x \mathbf{I}_{\text{blur}}\ \cos(\psi) + \nabla_y \mathbf{I}_{\text{blur}}\ \sin(\psi). \quad (3.16)$$

[9] Information on how the filter deals with boundary conditions here is omitted for simplicity but can be found in OpenCV's documentation.

[10] Here, $k_\mathbf{G} = 51$ was used as it results in a blurring effect equivalent to the size of the ANYbotics ANYmal C quadruped [5] and the types of uneven terrain it traverses, e.g., stairs.

This first-order derivative information can be used within the optimization as the nominal or desired pitch angle of the robot's body.

The implementation of the aforementioned description is outlined in Alg. 1.

Algorithm 1: Directional Gaussian blurring of a heightmap

Data: $\mathbf{I}_{\text{full}}, l_x, l_y, r, k_{\mathbf{G}}, c_x, c_y$
Result: $\mathbf{I}_{\text{blur}}, D_{\mathbf{u}}\mathbf{I}_{\text{blur}}$
while isFastParallelThreadRunning() **do**
  $\mathbf{I}_{\text{part}} \leftarrow$ updatePartialHeightmap($\mathbf{I}_{\text{full}}$)
  $\mathbf{G} \leftarrow$ getGaussianKernel($k_{\mathbf{G}}$) ▷ Using Eq. 3.14.
  $\mathbf{G}_{\text{pre-dir}} \leftarrow$ getPreDirectionalGaussianKernel($\mathbf{G}$) ▷ Using Eq. 3.13.
  $\psi \leftarrow$ getRobotYaw()
  $\mathbf{R} \leftarrow$ getRotationMatrix2d(c_x, c_y, ψ) ▷ Using Eq. 3.11.
  $\mathbf{G}_{\text{dir}} \leftarrow$ getDirectionalGaussianKernel($\mathbf{G}_{\text{pre-dir}}, \mathbf{R}$) ▷ Using Eq. 3.12.
  $\hat{\mathbf{G}}_{\text{dir}} \leftarrow$ normalizedirectionalGaussianKernel($\mathbf{G}_{\text{dir}}$)
  $\mathbf{I}_{\text{blur}} \leftarrow$ applyDirectionalGaussianBlur($\mathbf{I}_{\text{part}}, \hat{\mathbf{G}}_{\text{dir}}$) ▷ Using Eq. 3.10.
  $\nabla_x \mathbf{I}_{\text{blur}}, \nabla_y \mathbf{I}_{\text{blur}} \leftarrow$ computeGradients($\mathbf{I}_{\text{blur}}$)
  $D_{\mathbf{u}}\mathbf{I}_{\text{blur}} \leftarrow$ getFirstOrderDerivative($\mathbf{I}_{\text{blur}}, \psi$) ▷ Using Eq. 3.16.
end while

3.3.3 Whole-body control

We adopted the whole-body controller developed by Bellicoso et al. [184] to track base and end-effector trajectories at 400 Hz. The controller contains a state machine which adapts gains in response to slipping and other unexpected contact events.

3.4 Results

In this section, the evaluation of the proposed methods on the ANYbotics ANYmal series of quadrupedal robots is discussed. The results were obtained by testing in simulation (Gazebo and RaiSim by Hwangbo et al. [261]) as well as during real-world experiments.

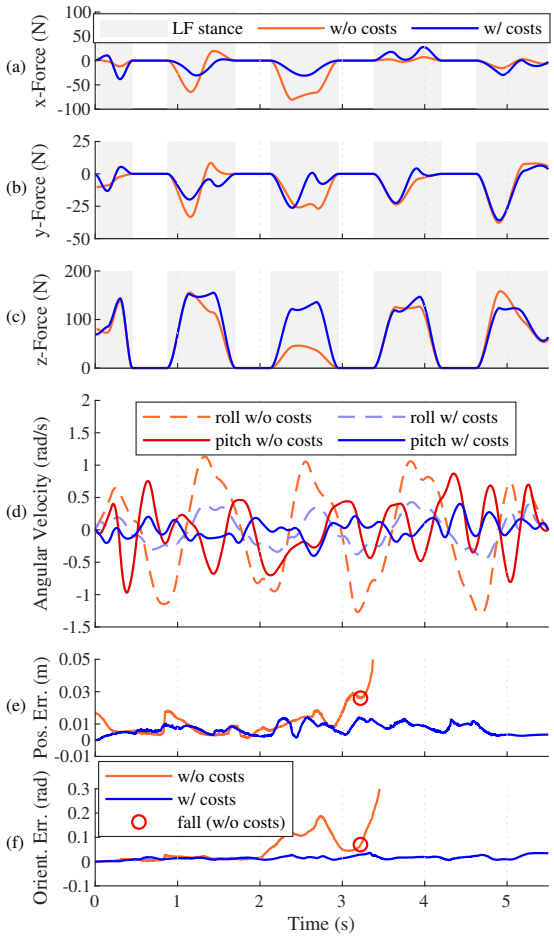


Figure 3.5: A comparison of the effects of the proposed analytical costs on the generated trajectories, tracking errors during the climbing of a single step, similar to the pallet shown in Fig. 3.7, in the Gazebo simulation. By adding costs to the optimization problem we can achieve motion plans that enable the robot to dynamically climb a step without falling.

3.4.1 Validation in simulation

Experiment 1 In Fig. 3.5, we demonstrate the need for more sophisticated planning for a trajectory up and over a step. We compare the performance with and without the base costs proposed in Sec. 3.3.2.1 and the end-effector costs proposed in Sec. 3.3.2.2.

Figure 3.5a and 3.5b show an overall decrease in peak x- and y-direction forces. Figure 3.5c shows an increase in the z-direction force during the third second of the trajectory resulting in a corresponding decrease in the x-direction force. Figure 3.5d shows the inertial stabilization of the base—a significant reduction in the roll and pitch angular velocity of the base can be observed.

Figure 3.5e and Fig. 3.5f show the norm of the tracking errors of the base during the execution of the trajectories by the whole-body controller in the Gazebo

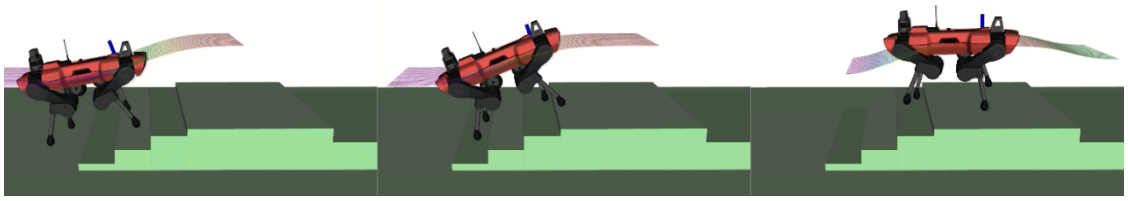


Figure 3.6: The proposed directional Gaussian blurring is applied to a heightmap discretization of a stairs model. The resulting envelope serves as inequality constraints, applied to the robot's pose, within our receding-horizon trajectory optimization framework called RHECALL. The constantly replanned trajectories—towards the joystick-controlled axis (blue) representing the final desired pose—are executed in RaiSim [261].

physics simulator. Initially, the controller manages to track both trajectories well; after $2\,\mathrm{s}$, the controller can no longer adequately follow the trajectory without the proposed costs, as indicated by the increase in the orientation error. At $3.2\,\mathrm{s}$, the robot's front foot collides with the step and the robot falls. Meanwhile, the controller tracking the trajectory generated with the addition of the proposed costs successfully completes the execution.

Experiment 2 We tested the terrain-dependent base pose constraint introduced in Sec. 3.3.2.4, in a receding-horizon fashion, on the ANYbotics ANYmal C quadrupedal robot in RaiSim [261], a realistic physics' simulator. As shown in Fig. 3.6, the base of the robot follows an 'envelope' positioned above the terrain.

3.4.2 Validation on hardware

Building upon the successful evaluation in simulation, we tested the viability of the approach on the real ANYmal B and C quadrupeds.

3.4.2.1 Locomotion on uneven terrain

To test the approach on uneven terrain in a controlled laboratory setting, modular structures have been set up. These terrains are described below:

Flat ground The generated trajectories were tested on flat ground for distances of up to $1.5\,\mathrm{m}$ with strides of up to $50\,\mathrm{cm}$. To have highly dynamic motions, the time horizon has been set to $3.5\,\mathrm{s}$ which resulted in a velocity of about

$0.6\,\mathrm{m\,s^{-1}}$ during the middle stage of the trajectory. For this setup, the number of optimization variables was 848.

Single Pallet This involved stepping onto a $1.2\,\mathrm{m} \times 1\,\mathrm{m}$ pallet of height $16.5\,\mathrm{cm}$ (see Fig. 3.7). For this experiment, the forward motion was about the same as for the Flat Ground, while the time horizon of the trajectory was increased to $5.5\,\mathrm{s}$. The initial orientation of the base was restricted such that the pallet stays within the field of view of the robot's camera (so it could detect the pallet position). In this scenario, there were 952 optimization variables.

Double Pallet This two-step environment is shown in Fig. 1.3. The steps were $14.5\,\mathrm{cm}$ and $16.5\,\mathrm{cm}$ high. The horizontal distance between steps was $40\,\mathrm{cm}$. The optimization parameters were kept the same as for the Single Pallet; again being 952.

Random steps For tests involving ANYmal C, we used a series of steps of different sizes which were positioned and oriented randomly. An example of one arrangement can be seen in Fig. 3.8.

Open-loop trajectories were evaluated on the ANYmal B quadruped using 3 different terrains of increasing difficulty. While the robot's sensors could have been used to create a model of the terrain (using its Intel RealSense depth camera), the model of the environment was instead loaded from a virtual model. This ensured repeatability and avoided limits in sensor field of view and the resolution. For the tests using pallets, the robot's front camera was used to read an AprilTag which gave the position and orientation of the obstacle with respect to the robot. The robot's onboard state estimator (Bloesch et al. [262]) was used as state input; measurable estimator drift was present.

Fig. 3.7 shows the kinematic model and the real robot executing the optimized trajectory on the single pallet while Fig. 1.3 shows the double pallet. The results show that despite the errors in state and terrain estimation, the robot realized the trajectory to a high degree of accuracy and precision.

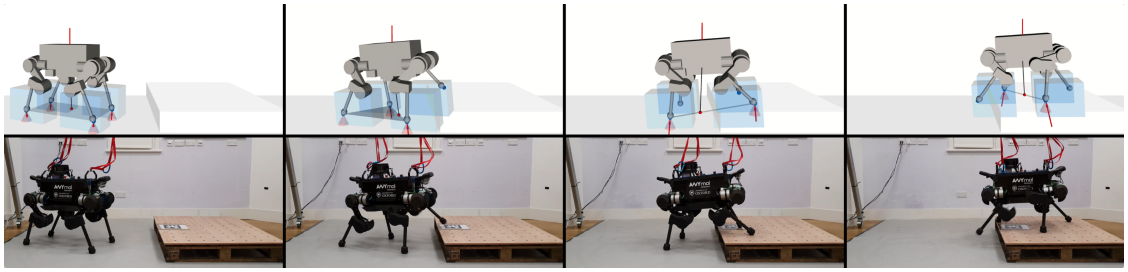


Figure 3.7: The comparison of a dynamic trajectory generated using the proposed approach (top) with the experimental evaluation (bottom) for a 5.5 s climbing of a pallet with a turn.

3.4.2.2 Onboard computation of long-horizon trajectories

The full system—the generation of the initial guess, the optimization of the trajectory and its execution by the whole-body controller—ran onboard the robot's computer.

As discussed in Sec. 3.3.1, the number of optimization variables depends on the number of steps and the time horizon of the trajectory. While the number of steps was kept constant (16 steps for dynamic walking, 14 for trotting), the time horizon was changed on a per-environment basis to make dynamic trajectories feasible.

Table 3.2 summarizes the computation times obtained on the onboard locomotion computer (Intel i7-7500U) of ANYmal B. Three variations of problem formulation have been tested: optimizing all variables, keeping only phase durations fixed, and keeping both phase durations and footstep locations fixed. Not re-optimizing the phase durations gives a great improvement in the computation time per iteration while the success rate is similar, or better. Additionally, the foot trajectories can be taken from the initialization (but eventually adapting the height to correspond to the terrain) and not re-optimized, but this results in a slight increase in the computation time. It appears that it is simpler for the optimization to adapt the foot positions than the base trajectory since the base is coupled to, and constrained by, all the feet.

Optimization Variables	Flat Ground (ms) 1st iter.	Mean	Pallets (ms) 1st iter.	Mean
Full	315 [304]	156 [126]	843 [700]	417 [305]
No Phase	210 [202]	69 [60]	385 [371]	117 [108]
No Phase&Feet	282 [282]	84 [74]	534 [524]	177 [145]

Table 3.2: The per-iteration computation time when solving the optimization problem using the IPOPT solver. The 1st iteration (iter.) takes longer due to the solver initialization, subsequent iteration times are approximately constant. The mean was computed for 100 iterations. The values without costs are shown in the square brackets.

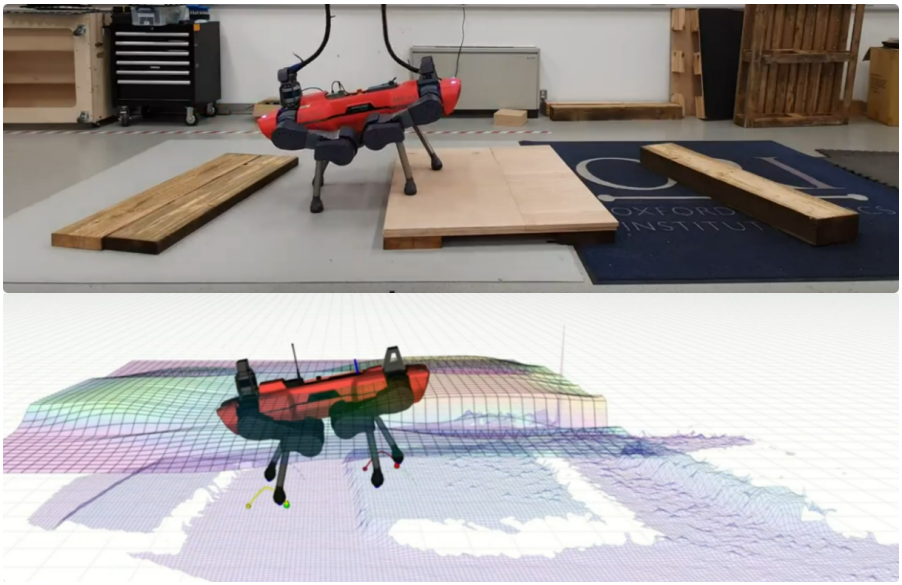


Figure 3.8: The proposed terrain-based base range-of-motion constraint has been tested on the real ANYbotics ANYmal C quadrupedal robot. The robot was tasked with traversing a series of steps of random depths, heights and orientations. Using its RGB-D cameras, it acquired a 2.5D elevation map of the terrain which was preprocessed (to remove holes) and then continuously filtered to produce an envelope which guided the motion of the robot's base.

3.4.2.3 Base pose constraint applied to ANYmal C

Similarly to the results obtained in simulation discussed in Sec. 3.4.1, in Fig. 3.8, we are showing the real robot traverse a series of random steps using the terrain-dependent base pose constraint proposed in Sec. 3.3.2.4.

3.5 Discussion

3.5.1 Computational challenges

As already discussed in Sec. 3.4.2.2, the proposed methods facilitate the generation of high-fidelity, long-horizon, whole-body trajectories which can be executed successfully on challenging terrains. However, due to the dimensionality and nonconvexity of the optimization problem, the computation time can be significant and convergence to the global optimum is unlikely. In Appendix A.4, we analyse how various components of the TOWR optimization framework affect the overall computation time. In order to offset the computation effort and deal with the nonconvexity of the locomotion problem, a near-optimal initialization of the solver can be used. A learning-based initialization scheme is investigated in Chapter 5.

3.5.2 Continuous locomotion

For continuous locomotion, it is paramount to replan the trajectories. This must be achieved at a high rate for the execution to be robust to unmodelled uncertainty, dynamic conditions or unexpected disturbances. In particular, we have found that the major challenge comes as a result of a drift in the robot's state estimate. A receding-horizon variant of the framework will be proposed in Chapter 4.

3.5.3 Perception

Testing algorithms related to perception and sensor data in simulation can be challenging as terrain models are not fully representative of real-world conditions. One potential solution is to integrate noise into the model (mesh) itself. In Appendix A.3, we show a process of reconstructing realistic terrain meshes from pointclouds acquired by RGB-D cameras.

3.6 Conclusion

This chapter focused on adapting a trajectory optimization framework for walking robots and a whole-body controller for real-world deployment. The proposed approach produces long-horizon, dynamic trajectories that are not only feasible in theory, but can also be reliably executed on real quadrupedal robots (ANYmal B and C) on a variety of terrains of increasing difficulty.

Leveraging a formulation based on direct collocation with state and control parameterized by cubic polynomials of Hermite form, enabled the development of fast-to-compute analytical costs. The integration of perception facilitated choosing advantageous footholds and reduced the risk of collisions with edges by the swinging end-effectors. Terrain-aware kinematic constraints have been proposed for the base which helped eliminate solutions where the base would exhibit superfluous vertical and angular motion. The application of superquadrics as range-of-motion constraints for the end-effectors resulted in a more appropriate approximation of the kinematic limits of the legs and promoted more natural-looking motions than the existing box constraints.

The next chapter will describe extending the proposed approach to enable sustained mobility through a receding-horizon formulation and the integration of the current state. Online replanning of terrain-aware dynamic locomotion with a several-step horizon will be demonstrated.

4

Receding-horizon adaptive trajectory optimization

This chapter includes elements of the following publications:

[2] **O. Melon**, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811

[3] R. Orsolino, S. Gangapurwala, **O. Melon**, M. Geisert, I. Havoutis, and M. Fallon, "Rapid Stability Margin Estimation for Contact-Rich Locomotion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 8485–8492

This chapter presents the reformulation and extension of the trajectory optimization based on direct collocation, described in Chapter 3, into a receding-horizon (RH) model. To this end, we propose a segmentation-based formulation with asynchronous base and end-effector timings and discretizations. We put forward a heuristic initialization scheme, which efficiently reuses the previous plan, and integrate real-time perception to enable continuous motion. We validate our approach on the ANYmal C quadruped on uneven terrain. Additionally, we investigate the performance of the replanner after replacing the SRBD constraint with our learned stability margin constraint [3] which considers torque limits.

4.1 Introduction

State-of-the-art legged-locomotion controllers can maintain dynamic motion while handling perturbations from external forces on moderately uneven terrain. Alternatively, locomotion planners can reason over longer horizons to strategically plan footholds and paths across more prominent obstacles, such as those seen in industrial settings. However, the benefits of these two paradigms—speed and foresight, respectively—are not trivial to combine. As a result legged robots have limited performance in scenarios that demand careful foothold selection, gait adaptation and momentum 'shaping'[1].

Dynamically complex, long-horizon robot motion plans are commonly generated through trajectory optimization (TO)—a process which minimizes an objective function while satisfying equality and inequality constraints. Our TO approach discussed in Chapter 3 uses the phase-based end-effector parametrization introduced by Winkler et al. [86] making it appropriate for handling the contact-induced discontinuities inherent to legged locomotion while not imposing overly rigid assumptions. Adapting this approach for online use carries several challenges, such as the need to minimize the required computation time and the need to employ the optimizer in a receding-horizon (RH) manner without compromising the solution's expressiveness.

We propose a problem formulation which enables RH replanning and facilitates the systematic use of an initialization scheme. We integrate real-time perception to enable online foothold optimization while preserving the ability to optimize the durations of stance and swing phases. As shown in Fig. 4.1, we successfully deploy this system on the real ANYmal C quadruped. Figure 4.2 provides an overview of our replanning framework, which is detailed throughout Sec. 4.3.

[1] 'Shaping' a body's momentum involves planning advantageous trajectories (velocities) that produce momenta which enable smooth and stable motion. Coordinated body motion is required when traversing large obstacles.

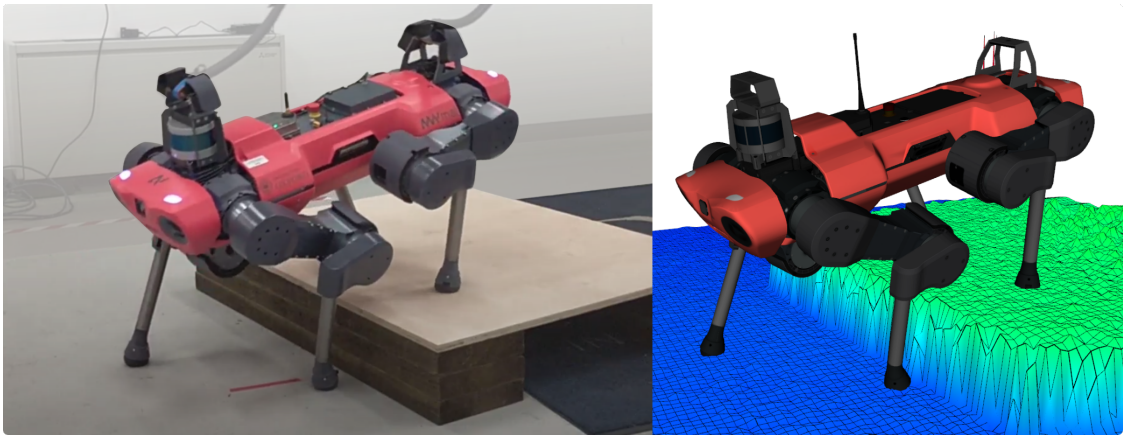


Figure 4.1: The ANYmal C quadruped using online, receding-horizon trajectory optimization and real-time perception to cross a 0.20 m step.

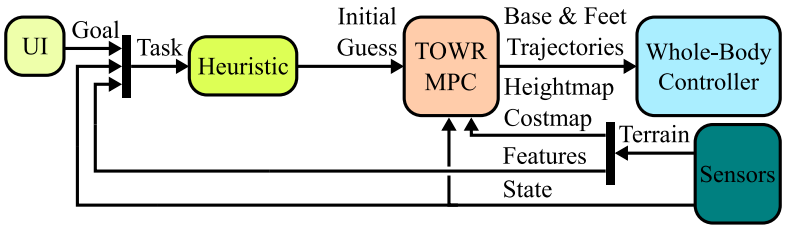


Figure 4.2: Overview of the proposed onboard system. The user, or a high-level planning algorithm, specifies the goal as a $SE(2)$ base pose within a couple meters of the robot. The sensors provide the estimated state, along with environment features. Together, these form a task that is input to the proposed initializer, Heuristic, which produces an asynchronously parametrized trajectory segment for the trajectory optimizer (TOWR MPC) to refine so that it satisfies dynamic and kinematic constraints. It uses the heightmap, a dense 2.5D elevation map produced using depth cameras or LiDAR, and the costmap to plan favourable footholds. The resulting base and feet trajectories are tracked by the whole-body controller.

4.1.1 Contributions

- Implementation of an *asynchronous* and *segmentation*-based trajectory optimization formulation for legged robots based on the SRBD model to enable RH replanning.
- A heuristic method for initializing the NLP problem that efficiently reuses the previous plan in an MPC fashion.
- The integration of a learned stability margin criterion, capable of enforcing joint-torque limits, in place of a dynamics (SRBD) and force constraints, within a real-time RH legged locomotion replanner.

- Evaluation of the proposed system in simulation and on hardware using a real ANYmal C quadruped.

4.2 Related works

4.2.1 Motion planning via model-predictive control

Many of the most successful approaches to dynamic legged locomotion use TO. To model the dynamics of the robot, they may use the full rigid-body dynamics (RBD) model (Mastalli et al. [145]), or simplified models such as centroidal dynamics (CD) (Orin et al. [77], Dai et al. [83], Carpentier and Mansard [263]), and single-rigid-body dynamics (SRBD) (Winkler et al. [86]). They can also adopt different strategies for contact scheduling such as completely predefining the timings, only predefining the sequence (Ponton et al. [264]) or fully discovering both (Posa et al. [125], Mordatch et al. [153]). However, few of these methods are suitable for online motion generation.

To be applicable online, approaches usually rely on simplified dynamics criteria like the zero-moment point (ZMP) (Vukobratovic and Borovac [265]), or linearize the CoM part of the SRBD (Di Carlo et al. [266], Bledt and Kim [182]). Some works decouple the perceptive foothold selection to reduce computation time (Fankhauser et al. [267], Kim et al. [268]). Jenelten et al. [269] used batch search optimization to find locally optimal footholds. Villarreal et al. [270] trained a convolutional neural network for this purpose. While simultaneous optimization of footholds has been achieved (Herdt et al. [271], Naveau et al. [272], Winkler et al. [273]), doing it on rough terrain remains a challenge. Jenelten et al. [274] eliminated contact forces from the SRBD to achieve real-time terrain-aware receding-horizon planning that jointly optimizes footholds and the base pose.

More sophisticated nonlinear MPC approaches use the whole-body model (Dai et al. [83]) of the legged robot but have optimization horizons that are too short to be able to adequately traverse large obstacles (Farshidian et al. [142]).

Other implementations based on the full robot model employ differential dynamic programming (DDP) (Mastalli et al. [145]) with an efficient feasibility-driven formulation that promises to enable high-frequency RH control. Another family of approaches is mixed-integer programming (MIP) (Apgar et al. [275], Aceituno-Cabezas et al. [276]) which, however, does not scale well with the number of decision variables. Other authors successfully used the SRBD which represents a compromise between model accuracy and complexity that allows online replanning (Cebe et al. [277]). While the SRBD assumes the inertia to be invariant, the changing property can be accounted and compensated for without increasing model complexity (Villarreal et al. [270]).

4.3 Methodology

In this section we present our approach to online replanning and execution of dynamic locomotion. The developed framework is shown in Fig. 4.2. Notably, in this chapter we consider the heuristic initialization scheme. It is based on the traditional MPC-style approach which reuses a part of the previous solution. The remaining section of the new initial guess, for which no prior data is available, is set up using linear interpolation.

4.3.1 Segmentation-based problem formulation

Our formulation is based on direct collocation which represents the state and control variables using cubic polynomials of Hermite form, i.e., the splines are defined at the nodes by the values $\mathcal{N}$ and the derivatives $\dot{\mathcal{N}}$ which make up the optimization variables $\mathbf{w} = (\mathcal{N}, \dot{\mathcal{N}})$.[2]

We use phase-based end-effector parametrization (Winkler et al. [86]), which enables the base and end-effector nodes to be asynchronous, allowing the phase durations to be optimized to achieve an aperiodic contact sequence. To enable MPC and facilitate its initialization, we keep the dimensionality of the problem

[2]The optimization variables $\mathbf{w}$ exclude *explicitly set* values or derivatives, denoted as $\mathcal{N}^{\square}$, which act as *implicit constraints*.

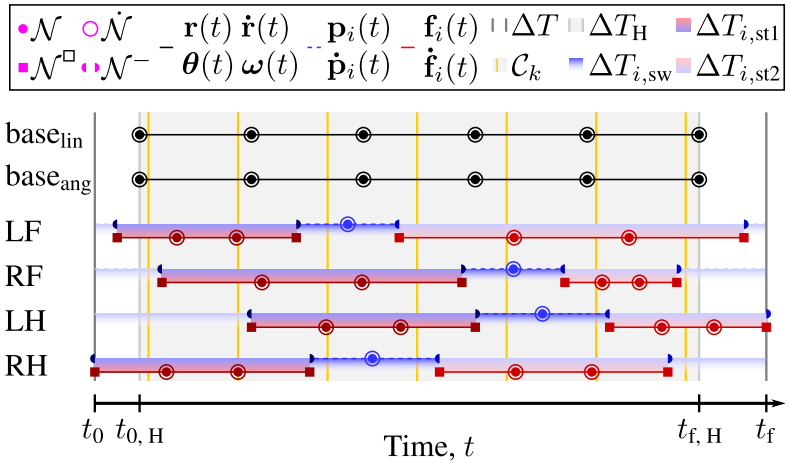


Figure 4.3: Our direct collocation, *segmentation*-based problem formulation uses phase-based end-effector (LF, RF, LH, RH) parametrization [86]. To facilitate initialization of the solver and allow the contact phase durations to be optimized, we keep the dimensionality constant by defining base and feet time horizons *asynchronously*.

constant by proposing a *segmentation*-based formulation with *asynchronous* base and feet *time horizons* as shown in Fig. 4.3[3].

The problem is defined over a duration ΔT between time t_0 (earliest contact) and the time t_{f} (latest take-off) during which each foot takes a step resulting in the *first* $\Delta T_{i,\mathrm{st1}}$ and the *second* $\Delta T_{i,\mathrm{st2}}$ stance phase. In between, the swing phase $\Delta T_{i,\mathrm{sw}}$ occurs. If a leg is in swing at $t_{0,\mathrm{H}}$, that swing is not re-optimized. Instead, the next stance-swing-stance sequence is considered in the optimization.

Figure 4.3 illustrates the problem formulation for the base horizon $\Delta T_{\mathrm{H}} = 5\, dt_{\text{base-poly}}$ which is defined with respect to the first and last base nodes occurring at $t_{0,\mathrm{H}}$ and $t_{\mathrm{f},\mathrm{H}}$, respectively. The figure shows all the constructing nodes, implicit constraints $\mathcal{N}^{\square}$, polynomials, and durations. In addition, a constraint $\mathcal{C}_k$, discretized at a fixed time step k, is shown; this type of constraint can be used to enforce the dynamics equality[4] or the range-of-motion inequality constraints. Notably, these can be enforced not only at the nodes but anywhere along the

[3] *Segmentation* is vital for the training of a RH regressor-based initializer (introduced in the following chapter in Sec. 5.3.2) which produces a fixed-size output of a given set of optimization variables $\boldsymbol{\tau}$ representing a trajectory with a predefined time horizon.

[4] We use the single-rigid-body dynamics (SRBD) model, described in Sec. 2.3.3, as the dynamics equality constraint.

polynomials. The optimization values that solely represent values that remain constant for a period of time are denoted by $\mathcal{N}^{-}$.

Using Eq. 3.3, the size of the proposed RH problem can be computed. Typically, when using the heuristic initializer, the following parameters were used:

- The duration of the base-motion horizon ΔT_{H} and base polynomial discretization $dt_{\text{base-poly}}$ were set to $1.4\,\mathrm{s}$ and $0.1\,\mathrm{s}$, respectively; using Eq. 3.4, the number of base nodes is determined as $n_{\text{nodes}}^{\text{base}} = 15$.

- For a quadruped, the number of legs n_{feet} is 4. The chosen number of stance n_{st} and swing n_{sw} phases for each foot i is 2 and 1, respectively.

- Under the formed assumption that each stance phase occurs in-between swing phases, and the fact that the forces are constrained to $\mathbf{0}$ at the start and end of each *swing* phase. the number of force nodes per foot per stance $n_{\text{nodes}}^{\text{force/foot/st}}$ is 2.

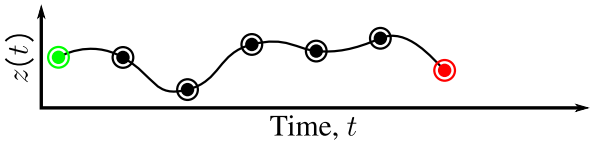
We employ *superquadrics*, described in Sec. 3.3.2.3, as end-effector range-of-motion constraints.

4.3.2 Heuristic initialization

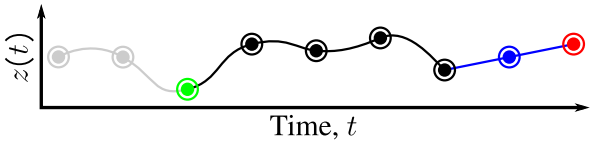
An illustrative example explaining the heuristic initialization is shown in Fig. 4.4. The scheme initializes the base trajectory variables using the corresponding values of the solution obtained at the previous replanning cycle. The non-overlapping part of the new base trajectory is linearly interpolated up to the user-defined goal. This stage is shown in Fig. 4.4b.

Meanwhile, the footholds corresponding to the first stance phase coincide with the location given by the previous plan whilst the footholds of the second stance phase are initialized at a location set relative to the goal state.

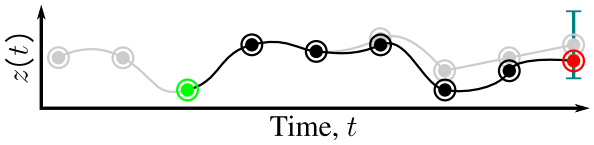
The heuristic initializes the NLP problem with periodic gaits (e.g., trot) which result in the two stance phases having the same duration ($\Delta T_{i,\text{st1}} = \Delta T_{i,\text{st2}} = \Delta T_{i,\text{st}}$) and in all the feet having the same stance and swing durations



(a) Previous trajectory



(b) Initialization via reused and interpolated segments



(c) Current trajectory

Figure 4.4: A 1D illustration of the receding-horizon Heuristic initialization method applied to the z-dimension of the base trajectory spline made up of third-order polynomials connected at the nodes (circles); the green and red nodes are the initial z_0 and final (goal) z_f state. The 3 main stages involve (a) the previous solution; (b) the initialization using reused (black) and interpolated (blue) segments; and (c) the current, optimized trajectory. The teal line in Fig. 4.4c represents limits (inequality constraints) on the goal state.

ΔT_st and ΔT_sw; however, these values can still be varied during the optimization. The durations can be set using the duty factor d and the cycle factor c such that $\Delta T_\mathrm{st} + \Delta T_\mathrm{sw} = (1.0 - c)\Delta T_\mathrm{H}$ and $d = \Delta T_\mathrm{st} / \Delta T_\mathrm{cycle}$ (where $\Delta T_\mathrm{cycle} = \Delta T_\mathrm{st} + \Delta T_\mathrm{sw}$ is the gait cycle time). The relative time offset $t_{0,i}$ between the start of the base trajectory $t_{0,\mathrm{H}}$ and that of the ith foot can be set as the fraction $\alpha_i \in (0, 1)$ such that $t_{0,i} = \alpha_i \Delta T_\mathrm{H}$.

4.3.3 State prediction due to the optimization delay

Most legged locomotion MPC algorithms target relatively high frequencies of 40 Hz to 200 Hz which facilitate shorter-horizon reactive behaviours (Bledt and Kim [182], Bledt and Kim [241]). Because our framework is concerned with solving longer-horizon dynamic motion, it operates at a much lower, fixed frequency of 1 Hz to 3.3 Hz. As a result, it experiences delays measured in hundreds of milliseconds between measurements of the state and the execution of resulting solutions. To manage this issue, the starting point of the computed trajectory corresponds to the time at which it will be executed and occurs after the computation begins. We implement a simple *predictor* to handle this by using the

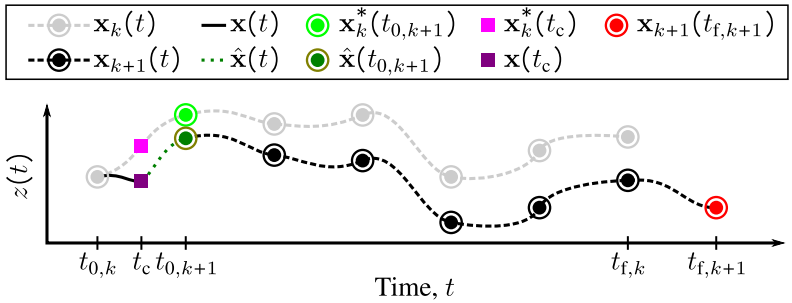


Figure 4.5: A 1D illustration—focusing on the z-dimension of the base trajectory spline previously shown in Fig. 4.4c—of the method of predicting the initial state for the next plan $\hat{\mathbf{x}}(t_{0,k+1})$ using the desired initial state from the current plan $\mathbf{x}_k^*(t_{0,k+1})$, the desired initial state from the current plan $\mathbf{x}_k^*(t_c)$ and the actual state $\mathbf{x}(t_c)$ at the current time t_c. This approach is needed due to the delay introduced by the long optimization time.

desired position computed by the previous plan and the current tracking error:

$$\hat{\mathbf{x}}(t_{0,k+1}) = \mathbf{x}_k^*(t_{0,k+1}) + \boldsymbol{\alpha}(t) \odot \overline{\mathbf{x}}(t), \tag{4.1}$$

$$\overline{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_k^*(t), \tag{4.2}$$

where t is the current time, $t_{0,k+1}$ is the time at start of the new plan $k+1$, $\hat{\mathbf{x}}(t_{0,k+1})$ is the predicted state used as initial state for the next plan, $\mathbf{x}_k^*(t)$ is the desired state from the current plan, $\odot$ is the Hadamard (element-wise) product of the scaling vector $\boldsymbol{\alpha}$ and the current tracking error $\overline{\mathbf{x}}(t)$.

For the scaling, values of 0 mean that the prediction relies only on the desired state while values of 1 fully take into account the tracking error and correspond to the actual state of the robot when prediction is with a short horizon. For each foot, the scaling vector is set to $\mathbf{1}$ if it is in stance and this phase corresponds to the same one at the beginning of the next plan, or 0 otherwise. For the base, the scaling factors are tuned between those two values to allow feedback on the actual state while still being close enough to the previous plan for stable numerical behaviour. This kind of simple prediction is only valid if the horizon is short with respect to the dynamics of the tracking error. To have a more accurate initial state for the optimized trajectory, the constraint on the initial state is updated with a new prediction between each iteration of the optimizer.

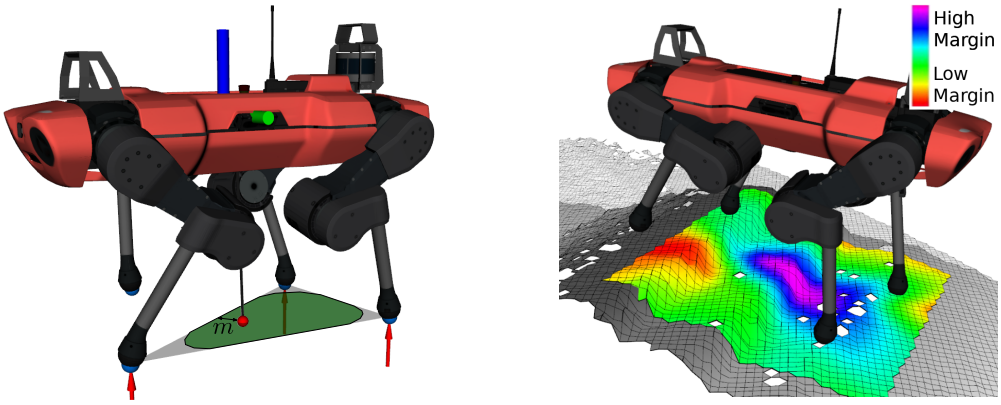
4.3.4 Terrain perception

The onboard RealSense depth cameras and the GridMap library (Fankhauser and Hutter [66]) are used to perceive the terrain. GridMap constructs a 2.5D heightmap which constrains the footholds and a costmap to aid obstacle avoidance. A mildly processed version of the heightmap (holes removed) is used to match the z-coordinate of the planned footholds to the actual terrain and a smooth version of the heightmap is used to set a cost on each candidate foothold. By making the cost proportional to the slope of the terrain, the solver is incentivized to choose footholds away from edges and inclined surfaces. In addition, the same heightmap data is processed to detect edges. Both the heightmap and the edges are updated at a frequency of 6 Hz.

4.3.5 Whole-body controller and robustness to uncertainty in perception

To execute the generated plans, the whole-body controller (Bellicoso et al. [184]) is used to track the trajectory of the base and the end-effectors at 400 Hz. The controller includes different heuristics to improve its behaviour in case of slip or other unexpected events.

To be more robust to errors in the terrain height measured by the depth cameras and to be able to make contact at the right instant, the swing trajectory has been modified. On the planner side, the formulation of splines in TOWR has been adapted to allow non-zero vertical velocities for the foot motions at the transition from swing to stance. Additionally, the controller is able to modify the setpoints of the feet computed by the planner by continuing the swing trajectory downward until contact is detected. Those two features allow our controller to quickly react to delayed contacts while having smooth foot trajectories despite the low frequency of the proposed planner.



(a) Analytical stability margin **(b)** Learned stability margin

Figure 4.6: As presented in Orsolino et al. [3], the stability margin can be (a) determined analytically as the perpendicular distance between the contour of the feasible region (green region) and the Instantaneous Capture Point (ICP) (red sphere), or (b) approximated by a neural network and easily visualized as a dynamic colormap.

4.3.6 Locomotion replanning using a dynamic stability criterion

The limitation of our RH nonlinear TO framework is that it relies on the simplified SRBD model, which ignores joint torques and their limits, and uses a number of costs and constraints to achieve a desirable motion of the CoM. Subsequently, we tested removing the equality constraint and eliminating force variables, and related inequality constraints, and integrating a *learned stability margin*—a data-driven estimator of the dynamic stability of a legged robot proposed by Orsolino et al. [3]. This learned approximation offsets the computational complexity of its analytical equivalent, as shown in Fig. 4.6 and described in Sec. 4.3.6.1.

In Sec. 4.3.6.2, we discuss how we formulated the underlying TO problem to integrate this dynamic stability criterion.

4.3.6.1 Learned stability margin

The analytical stability margin can be computed as follows

$$m = \min\left(\mathbf{N} \cdot [\boldsymbol{\xi}^{\mathsf{T}} \quad 1]^{\mathsf{T}}\right) \approx \boldsymbol{\pi}_m\left(\boldsymbol{\lambda}\right). \tag{4.3}$$

It involves minimizing the norm of the vector resulting from the multiplication of the matrix of the normals and offsets, $\mathbf{N} \in \mathbb{R}^{n_{\text{edges}} \times 3}$, that defines the contour (edges) of the feasible region, and a second vector ($\mathbb{R}^3$) which includes the Instantaneous Capture Point (ICP) $\boldsymbol{\xi} \in \mathbb{R}^2$. These concepts are illustrated in Fig. 4.6a. The *feasible region* which builds upon the *support region*[5] by integrating joint-torque limits (Orsolino et al. [279]) and external wrenches (Abdalla et al. [280])[6]. It can be computed using an incremental projection (IP) algorithm (Orsolino et al. [3]). The ICP is the instantaneous stability point to which the CoM projection is converging (Pratt et al. [281]).

The stability margin can be approximated by a multi-layer perceptron (MLP), trained in a supervised fashion, using a policy $\boldsymbol{\pi}_m(\boldsymbol{\lambda})$ whose input $\boldsymbol{\lambda}(t) \in \mathbb{R}^{47}$ is

$$\begin{aligned}\boldsymbol{\lambda} = \Big(& {}^{\mathcal{B}}\mathbf{e}_{\text{g}}, {}^{\mathcal{L}}\dot{\mathbf{r}}, {}^{\mathcal{L}}\ddot{\mathbf{r}}, {}^{\mathcal{L}}\dot{\boldsymbol{\omega}}, {}^{\mathcal{L}}\mathbf{f}_{\text{ext}}, {}^{\mathcal{L}}\boldsymbol{\tau}_{\text{ext}}, \\ & {}^{\mathcal{L}}\mathbf{p}_1, \dots, {}^{\mathcal{L}}\mathbf{p}_{n_{\text{ee}}}, c_1, \dots c_{n_{\text{ee}}}, {}^{\mathcal{L}}\mathbf{n}_1, \dots, {}^{\mathcal{L}}\mathbf{n}_{n_{\text{ee}}}, \mu \Big)\end{aligned} \tag{4.4}$$

which includes quantities in base $\mathcal{B}$ and local $\mathcal{L}$[7] frames: the gravity axis[8] ${}^{\mathcal{B}}\mathbf{e}_{\text{g}} \in \mathbb{R}^3$; CoM linear velocity ${}^{\mathcal{L}}\dot{\mathbf{r}} \in \mathbb{R}^3$ and acceleration ${}^{\mathcal{L}}\ddot{\mathbf{r}} \in \mathbb{R}^3$; CoM angular acceleration ${}^{\mathcal{L}}\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$; the external wrench $({}^{\mathcal{L}}\mathbf{f}_{\text{ext}}, {}^{\mathcal{L}}\boldsymbol{\tau}_{\text{ext}}) \in \mathbb{R}^6$; and each end-effectors' position ${}^{\mathcal{L}}\mathbf{p}_i \in \mathbb{R}^3$ and contact state $c_i \in \{0, 1\}$, the normal of the terrain at the contact point ${}^{\mathcal{L}}\mathbf{n}_i \in \mathbb{R}^3$; and the friction coefficient $\mu \in \mathbb{R}$ (assumed singular and constant).

4.3.6.2 Problem formulation

The formulation of the TO problem was augmented. Instead of the SRBD equality constraint, the stability margin was used. Because the NN has learned offline the relationship between task-space variables and a feasible state of the robot, it

[5] Bretl and Lall [278] proposed an algorithm which computes the *support region*, a projection of a nonlinear convex set onto a 2D linear subspace.

[6] The *feasible region* can also include the joint-position limits though this has not been implemented in this work.

[7] The *local* frame is the *base* frame with zero roll and pitch angles. It is also known as a *control* or *horizontal* frame in literature.

[8] The gravity axis ${}^{\mathcal{B}}\mathbf{e}_{\text{g}}$ corresponds to the third row of the rotation matrix ${}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \in SO(3)$.

is not necessary to include the contact forces as optimization variables or force-related constraints as they were already included in the IP algorithm. The new *minimum stability margin* inequality constraints were added

$$\boldsymbol{\pi}_m\left(\boldsymbol{\lambda}\left(t_k\right)\right) \geq m_{\min}. \tag{4.5}$$

They were enforced at each discretization node of the base spline at time t_k where $k = \{1, \ldots, n_{\text{nodes}}^{\text{base}}\}$. The value of $m_{\min}$ is gait dependent; using a trotting gait, for example, the feasible region is a 1D segment and the maximum achievable value will thus be zero (when the capture point $\boldsymbol{\xi}$ belongs to the segment). A minimum margin value $m_{\min} = -0.05$, for example, will guarantee that $\boldsymbol{\xi}$ does not diverge during the optimization horizon and that, consequently, the robot does not fall.

As we provide an analytical Jacobian of the constraints to the solver, the required partial derivative[9] of the learned stability margin $\delta m / \delta \boldsymbol{\lambda}$ is obtained during training via the backpropagation algorithm. The inference of the derivative takes just $100\,\mu\text{s}$, instead of $940\,\text{ms}$ taken by the finite difference method (approximation), enabling real-time optimization on hardware.

4.4 Results

4.4.1 Heuristic initialization

The initializations provided by the heuristic, based on a user-commanded target robot base pose, enabled the continuous execution of a walk and a trotting gait during which the robot traversed obstacles of heights up to $0.2\,\text{m}$ ($38\,\%$ of nominal height) as shown in Fig. 4.1.

Fig. 4.7 shows the robot using the 2.5D elevation map to plan footholds and swing motions allowing it to climb a step. The base tracking error is shown in Fig. 4.8 for a hardware experiment where the robot walked over obstacles up to $0.13\,\text{m}$ high for over $5.5\,\text{min}$.

[9] We ensure that the derivative information is expressed in the correct frame (with respect to the optimization variables) by applying the chain rule.

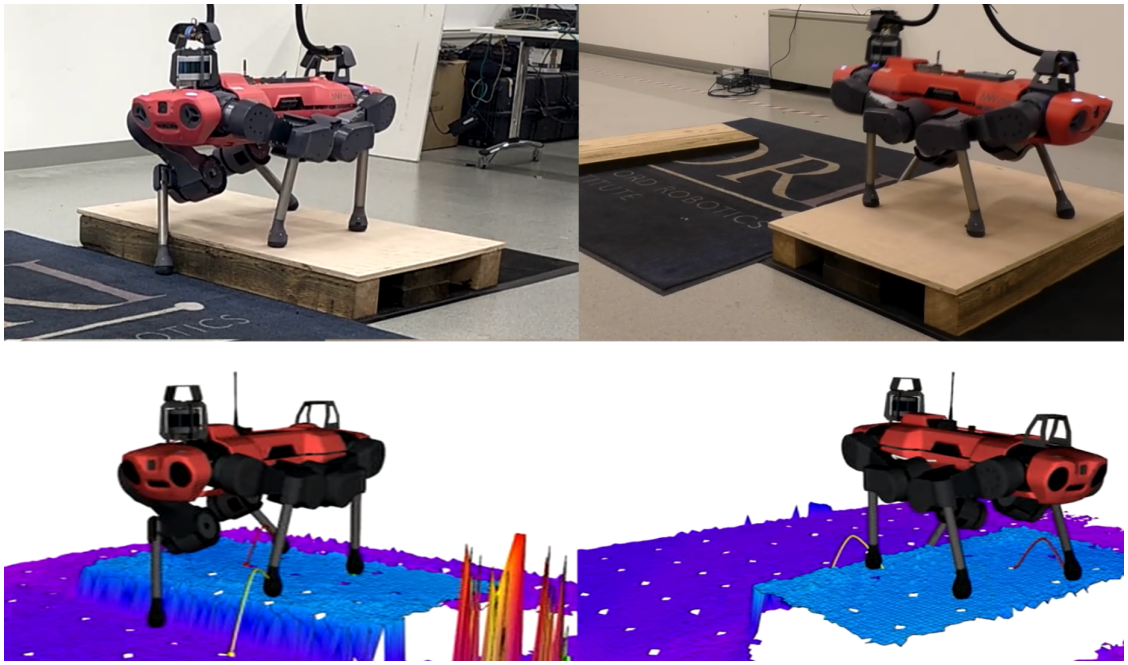


Figure 4.7: Rear (left) and front (right) angles of the ANYmal C quadruped replanning its motion using the Heuristic initialization to continuously traverse a series of steps of heights reaching 0.13 m. The bottom figures show the sensed elevation map and the planned end-effector trajectories.

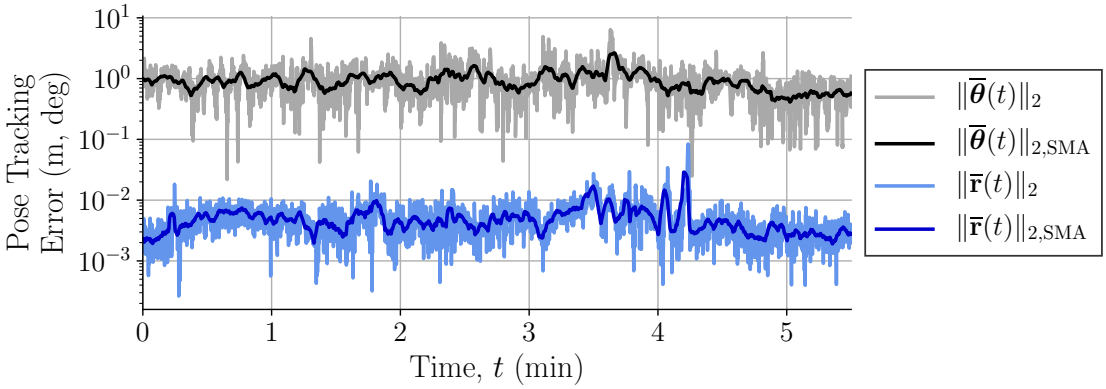


Figure 4.8: Despite a low replanning frequency (1.4 Hz) and a large time step $dt_{\text{base-poly}}$ (0.1 s), the direct collocation formulation produces trajectories which satisfy the dynamics to third-order enabling accurate and precise tracking by the whole-body controller. The figure shows base pose tracking errors, and their simple moving averages (SMA), of a 5.5 min execution of the MPC initialized by the heuristic, on terrain, by the real robot. The root-mean-squared error (RMSE) values of the linear position and orientation were 6.55×10^{-3} m and 1.07°, respectively.

The replanning frequency for the walking gait was $1\,\mathrm{Hz}$ with an optimization horizon of $3\,\mathrm{s}$. The trot could run at $2\,\mathrm{Hz}$ over a replanning window of $1\,\mathrm{s}$. For both walk and trot, the Heuristic was operated at speeds of about $0.1\,\mathrm{m\,s^{-1}}$ on flat ground and while negotiating the obstacles shown in the accompanying video. The average step length on flat ground was $0.25\,\mathrm{m}$ for the trot and $0.3\,\mathrm{m}$ for the walk.

Depending on the gait and the irregularity of the traversed terrain, the solver needed between 15 and 25 iterations to converge to the termination criteria specified by the desired convergence tolerance of 10^{-4} and the constraint and variable bound violation of 10^{-3}. Default values of the desired thresholds for the dual infeasibility and complementarity conditions were used. Occasionally, when the solver did not converge to the desired tolerances and exceeded the allowed number of iterations, we used one of the following two approaches. Because the planning horizon was at least twice as long as the executed plan, we could continue using the previous plan and reattempt the optimization from the new state. We also used a heuristic to decide whether a software-based safety stop should be engaged (commanding the motors to 'freeze' the robot in its current state) or if any of the intermediate solutions (we stored each iterate) satisfied a lower 'acceptable' tolerance based on the primal infeasibility that allowed the robot to continue operation (of course, increasing the risk of failure).

4.4.2 Learned stability margin

We tested the formulation described in Sec. 4.3.6 in simulation (Gazebo) and on hardware. We used a time horizon ΔT_{H} of $3\,\mathrm{s}$, a base trajectory discretization of $dt_{\text{base-poly}}$ of $0.2\,\mathrm{s}$ and a walking gait. This resulted in an overall problem of 200 optimization variables and 500 constraints. Convergence took about 9 iterations using the primal-dual interior-point method solver IPOPT (Wächter and Biegler [159]).

A desktop with 4-core/8-thread Intel Xeon CPU E3-1505M v6 @ $3\,\mathrm{GHz}$ took to about $70\,\mathrm{ms}$ per iteration, while on the onboard computer, $40\,\mathrm{ms}$ were achieved.

We ran the replanner at a frequency of $1\,\mathrm{Hz}$. As can be seen in Fig. 4.9, the replanner enables the ANYmal C robot to adapt its body trajectory and footsteps location to the terrain with slopes and obstacles with heights from $0.06\,\mathrm{m}$ to $0.13\,\mathrm{m}$. In simulation, taller $0.18\,\mathrm{m}$ steps were used.

The benefit of measuring the stability of the robot using the feasible region is particularly apparent in load-intensive tasks or in scenarios that require the robot to take on complex body configurations. In the bottom-right image of Fig. 4.9, for example, the robot walks along a side step, causing its left feet to be on a surface $0.13\,\mathrm{m}$ above its right feet. In this case, any simplified model (e.g., SRBD) would plan the CoM motion in the middle of the left and right feet, exactly as if they were all on the same flat surface. The feasible region considers the joint-torque limits bringing the CoM closer to the most extended legs—capable of carrying more body weight. Additionally, the approach caused the robot to use a more steady, wider stance.

4.5 Discussion

During the experimental trials, it became apparent that reliable and sustained execution on terrain requires higher replanning frequency (likely at least $10\,\mathrm{Hz}$). Our proposed state prediction (Sec. 4.3.3), which continuously updated the initial state constraint, negatively impacted the convergence of the IPOPT solver. It was often insufficient to guarantee robustness e.g., when the robot collided with the environment and lost tracking of the planned path. In an attempt to accelerate the computation, as discussed in Appendix A.4, we have assessed the sparsity patterns of the formulation and concluded that the slow optimization times are primarily due to the inefficient (overly generalized) implementation of the framework rather than the NLP solver (IPOPT), its linear solvers (MUMPS or HSL) or their subroutines. Our framework acted as a prototype and a proof of concept. We hope that future researchers working on similar approaches will consider and address the discussed limitations of this framework.

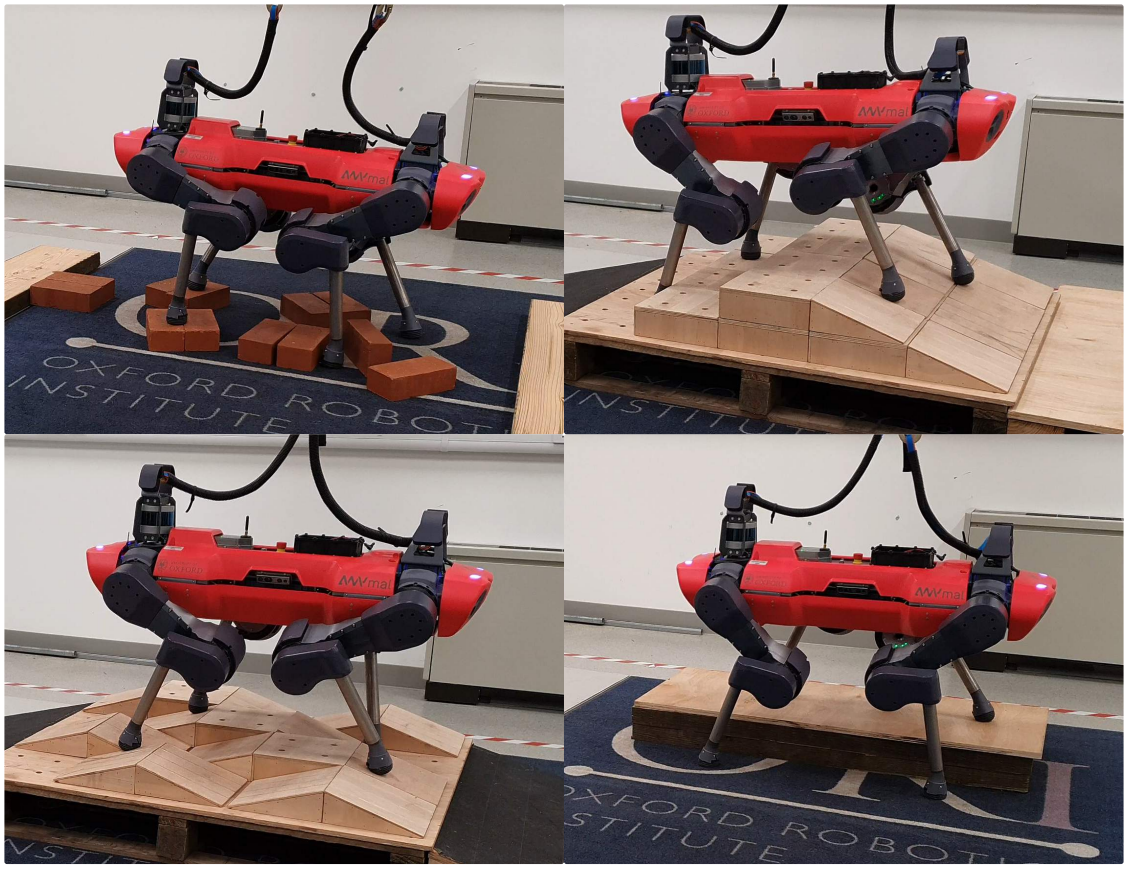


Figure 4.9: The proposed RH replanner using the learned stability margin (Orsolino et al. [3]), in place of the SRBD model, to traverse uneven terrains including bricks (top left), steps (top right), inclines (bottom left) and side steps (bottom right).

Our proposed segmentation approach lacked the capacity to change the motion of an end-effector already in a swing phase. However, due to the short swing phase durations, this trade-off was deemed acceptable.

During the original experiments outlined in Sec. 4.4.1, where the heuristic initialization scheme was used on uneven terrain, it was observed that the solver would return base trajectories characterized by superfluous vertical motion. These negatively impacted the stability of the robot. This motivated us to develop the terrain-aware range-of-motion base pose constraint presented in Sec. 3.3.2.4 whose application is shown in Fig. 3.8. Although it was generally effective at steering the solution trajectories towards more desirable outcomes, it was not always reliable. It was susceptible to imperfections in the terrain map. Also, implementing it as a constraint meant that it was not always satisfied

causing issues with convergence, e.g., when the initial or goal state was outside the bounds. In hindsight, implementing it as a well-weighted, flexible cost could deliver more consistent performance on real hardware.

When using the formulation with the learned stability margin discussed in Sec 4.3.6.1, we optimized footholds but not end-effector motion which was computed by a separate swing trajectory generator to achieve more squarish steps.

The feasible region can be used to account for the joint-position limits, in our future work, we would extend the training of the learned stability margin to incorporate this element allowing us to remove the end-effector range-of-motion constraints from the TO formulation.

4.6 Conclusion

This work presents a pipeline for iterative perception-aware planning of dynamic legged locomotion in a receding-horizon fashion via online trajectory optimization with a novel asynchronous, segmentation-based formulation using the single rigid body dynamics (SRBD) model.

We deployed this pipeline on a quadrupedal robot, ANYbotics ANYmal C, using fully onboard sensing and computation.

The proposed *heuristic* initialization combines previous solutions with a linearly interpolated guess to accelerate the computation. Hardware trials with heuristic demonstrate the ability to traverse terrain obstacles using walking and trotting gaits.

We have also shown that it is feasible to use the learned stability margin, as a measure of dynamic stability for legged robots in contact-rich scenarios, for online replanning of CoM trajectories and footholds while respecting the torque limits of the actuators.

5
Data-driven solver initialization

This chapter includes elements of the following publications:

[1] **O. Melon**, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1410–1416

[4] D. Surovik, **O. Melon**, M. Geisert, M. Fallon, and I. Havoutis, "Learning an expert skill-space for replanning dynamic quadruped locomotion over obstacles," in *Proceedings of the Conference on Robot Learning*, PMLR, 2021

[2] **O. Melon**, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811

This chapter investigates the use of a data-driven initialization scheme for a nonlinear programming solver to accelerate its convergence. It describes the developed pipeline for the generation, processing and learning of optimal trajectories. A supervised and an unsupervised behaviour cloning approach is presented and tested on the real ANYmal B and C quadrupeds. The former highlights the computational advantage of near-optimal initialization. The latter generalizes the idea, by leveraging the latent space, to enable the reconstruction of temporally extended segments of diverse behaviours for replanning.

5.1 Introduction

Solving a long-horizon problem involving agile motion with aperiodic contact sequences from scratch is impractical because of limited onboard computational resources. Research on real-time quadrupedal locomotion has therefore focused mostly on near-term planning.

One increasingly popular strategy is to produce high-quality, near-optimal initial guesses by leveraging offline experience, so that only minor online computation is needed to adapt to a scenario seen at runtime. Many earlier examples of this approach focused on manipulators (Jetchev and Toussaint [218]), leading to later studies on more dynamic systems (Mansard et al. [236], Lembono et al. [239]). Recent works have considered more complex variants that may be needed when the relevant solution space is multimodal (Lynch et al. [233], Surovik et al. [4]). For the most part, these works impose strong assumptions about the beginning and end of the motion plan being produced, i.e., static initial and final conditions. For generalized, sustained behaviour by dynamic quadrupeds, a formulation is needed that can handle their multimodality—and to do so without being required to routinely revisit a small and restrictive class of checkpoint states.

The objective of our work is to produce proactive plans over suitably long time horizons with minimal online computation effort. We first combine a learning-based data-driven initialization with an enhanced formulation of the optimization problem of dynamic motion generation (presented in Chapter 3), which we demonstrate for a quadrupedal robot walking on uneven terrain—an overview of the system for open-loop planning is shown in Fig. 5.1. Then, using the segmentation technique proposed in Chapter 4, we extend our initialization scheme with an aim to enable *online replanning* of dynamic, temporally extended motions that can be *sustained indefinitely* while anticipating upcoming terrain features by employing *diverse modes of behaviour*. The overview of the system for closed-loop receding-horizon replanning is shown Fig. 5.2.

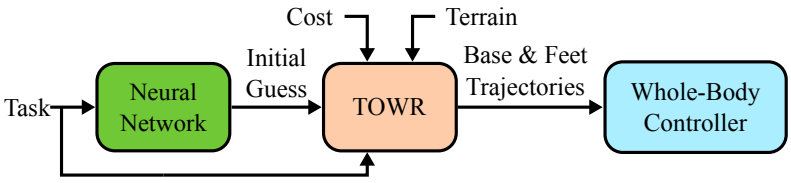


Figure 5.1: Overview of the proposed *open-loop* motion planner initialized by the LearnedInit neural network introduced in Sec 5.3.1. The initial guess is optimized using our extension of the TOWR solver (Sec. 3.3.2). The resulting trajectories are executed on the ANYmal B quadruped using the whole-body control (WBC) (Sec. 3.3.3).

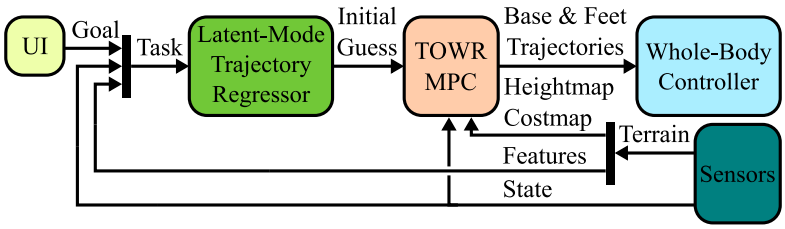


Figure 5.2: The system for *closed-loop replanning* previously shown and described in Fig. 4.2 modified with the LMTR initialization module which is discussed in Sec. 5.3.2.

5.1.1 Contributions

- A data-driven method to produce good initial guesses which speeds up optimization convergence. The method can compute 16-footstep trajectories in less than 1 s while avoiding poor local minima.
- Experiments in which the real ANYmal robot trots or walks across flat or uneven terrain while using only onboard sensors for state estimation, verifying the validity of our approach and its suitability for real hardware.
- Incorporation of LMTR (Surovik et al. [4]), a data-driven receding-horizon initializer of the NLP program, and its adaptation for uneven terrain and real-time deployment.

5.2 Related works

5.2.1 Data-driven initialization

Data-driven trajectory initialization schemes have been applied in the domain of manipulation to speed up the computation of smooth paths when reaching past

obstacles (Berenson et al. [235], Jetchev and Toussaint [218], Merkt et al. [237]). Related methods produce multiple initializations in different basins of attraction so as to identify distinct ways of approaching the object to be grasped (Dragan et al. [220], Lembono et al. [227]). Dynamic constraints can additionally be met for tasks such as quickly reaching to catch a thrown object (Lampariello et al. [282]) or rejecting large disturbances on underactuated aerial vehicles (Mansard et al. [236]). For legged locomotion on terrain, a related idea was used to plan individual feasible footsteps, which were then combined by another process into a full motion plan (Stolle et al. [217]).

Most of these efforts map tasks to initial solutions, while others map to segments of solutions (Stolle et al. [217]), or to additional constraints that convexify the problem (Dragan et al. [220]). Nearly all consider nearest-neighbour lookup and regression on an experience library as a mapping method, while many also consider other function approximators. These include support vector machines (SVMs) (Dragan et al. [220], Lampariello et al. [282]), GP regression (Lembono et al. [227], Lampariello et al. [282]), or artificial neural networks (ANNs) (Dragan et al. [220], Mansard et al. [236]). Another research strand is the use of feature-spaces and dimensionality reduction techniques (Stolle et al. [217], Jetchev and Toussaint [218], Merkt et al. [237], Dragan et al. [220], Lembono et al. [227]). In each case, the goal is to sufficiently represent past experiences in a manner which can inform future decisions.

5.2.2 Data-driven locomotion

Model-based approaches to motion generation may be limited by optimization time or the discovery effort needed to construct complex or subtle behaviour online. This has lead to data-driven approaches that 'amortize' this cost and effort by conducting it offline. Bledt and Kim [241] observed correlations between tasks and optimized behaviours to identify heuristic costs for regularizing an online planner whose implementation was presented in Bledt and Kim [182].

Alternately, machine learning approaches reduce the role of humans in curating the knowledge.

A hierarchy of two reinforcement learning (RL) policies was used in Tsounis et al. [283]—one for planning footholds and phase durations, and one for base and end-effector motion. To gradually build up the complexity of the learned behaviours, Xie et al. [198] used curriculum learning for bipedal stepping-stone motions, evaluated in simulation. However, these methods become sample-inefficient when dealing with a large state-space and typically require considerable reward shaping.

To produce longer-horizon actions, Mansard et al. [236] 'warm-started' non-linear MPC using a model learned from a sampling-based approach. Guided trajectory learning uses consensus optimization to learn a function approximator that only reconstructs feasible motions (Duburcq et al. [284]). In Lembono et al. [239], a "memory of motion" of dynamic, collision-free locomotion has been generated using the HPP Loco3D planner (Tonneau et al. [240]) for the CROCCODYL solver (Mastalli et al. [145]).

To adequately capture highly varied behaviour, multiple regressors can be used, as in computer animation of dogs and humans (Zhang et al. [224], Starke et al. [225]). For legged robots, Carius et al. [228] used imitation learning of optimal control demonstrations, improving sample efficiency over RL. To remove the need for multiple networks and discrete categories of actions, latent variables can be used to express continuous modal variation (Lynch et al. [233], Surovik et al. [4]).

5.3 Methodology

Optimization frameworks such as TOWR use solvers which require a guess $\mathbf{y}_0$ of the solution $\mathbf{y}$. It can be generated automatically by some map

$$A\colon \mathbf{x} \mapsto \mathbf{y}_0 \tag{5.1}$$

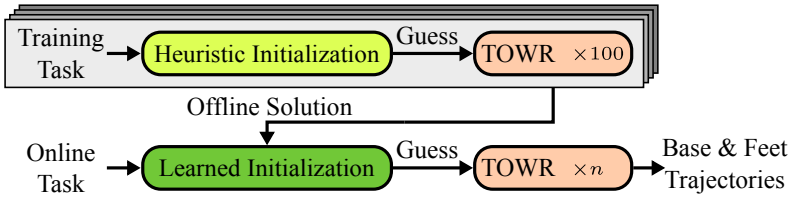


Figure 5.3: Learning from previous fully converged outcomes allows the optimizer to be initialized close to a good solution.

that acts upon the task $\mathbf{x}$, e.g., the pair of initial and desired robot states. Here, primal variables of the interior-point method are initialized with a guess which substantially affects not only the rate of convergence but also the quality of $\mathbf{y}$. As more costs and constraints are used, more local minima arise, and the initial guess becomes even more influential.

The conventional guess generator for TOWR, termed heuristic, linearly interpolates a path for the floating base between the start and goal locations from $\mathbf{x}$. Footsteps are evenly spaced and transitions between them are evenly timed according to the selected gait, with contact forces equally distributed to counteract the robot's weight.

This section consists of two subsections; the open-loop LearnedInit initializer (**Melon** et al. [1]) is described in Sec. 5.3.1 and the RH LMTR initializer—introduced in (Surovik et al. [4]) and deployed in (**Melon** et al. [2])—is discussed in Sec. 5.3.2.

5.3.1 Learned initialization of an optimization solver

The objective of this section is to produce a data-driven initializer, LearnedInit, to replace the heuristic such that far less optimization effort (iterations) is required, as sketched in Fig. 5.3, while furthermore avoiding poor local minima.

Denoting the heuristic as A_h and an iteration of the optimizer as another map T, optimization can be expressed as $\mathbf{y}_{\mathrm{h},n}(\mathbf{x}) = T^n A_\mathrm{h}\mathbf{x}$, with iteration count n selected based upon convergence or available computation time. The improved initializer is a function approximator $A_{\boldsymbol{\theta}}$ trained to mimic the behaviour of the

optimization process $T^n A_\mathrm{h}$. For a given task space $\mathcal{X}$, its optimal parameters are then defined by

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}'}{\operatorname{argmin}} \sum_{\mathbf{x} \in \mathcal{X}} \| A_{\boldsymbol{\theta}'} \mathbf{x} - \mathbf{y}_{\mathrm{h},n}(\mathbf{x}) \|_{\mathbf{W}} \tag{5.2}$$

with a positive-definite weight matrix $\mathbf{W}$.

In this work, $A_{\boldsymbol{\theta}}$ is a fully connected neural network with 2 to 3 hidden layers, where $\boldsymbol{\theta}$ refers to the connection weights. These are determined through supervised learning on a dataset $\mathcal{D} = \left\{ \mathbf{x}_i, \mathbf{y}_{\mathrm{h},n}(\mathbf{x}_i) \right\} = (\mathcal{D}_\mathbf{x}, \mathcal{D}_{\boldsymbol{\tau}})$ with samples $\mathbf{x}_i \in \mathcal{X}$. At present, a given $\boldsymbol{\theta}$ is learned for a specific environment; however, this serves as a first step toward contextual planning with $\mathbf{x}$ augmented by local environmental features. The form of $\mathbf{y}$, whose length depends on the total duration and footstep count, is also kept constant.

Given the significant risk of converging to poor local minima, the dataset $\mathcal{D}$ is not guaranteed to imply a well-behaved map from $\mathcal{X}$ to the solution space $\mathcal{Y}$. Two additional steps are thus taken to ensure the tractability of the learning problem in Eq. 5.2 and the quality of its result.

First, $\mathcal{D}$ is filtered based on a threshold of solution cost J_max to exclude poor solutions from training:

$$\mathcal{D}_\mathrm{good} = \{ \mathbf{x}_i, \mathbf{y}_i \mid J(\mathbf{y}_i) < J_\mathrm{max} \}. \tag{5.3}$$

Second, since even the dataset containing 'good' solutions $\mathcal{D}_\mathrm{good}$ is unlikely to contain only globally optimal solutions, the average performance and uniformity of the learned initialization can potentially be increased by repeating the process of Fig. 5.3 with $\boldsymbol{\theta}$ retrained on optimization outcomes resulting from its previous value. This cycle of moderate exploration and filtering thus lends an aspect of reinforcement learning to the scheme, with particular similarity to the alternation between local optimization and global supervised learning of control laws in guided policy search (GPS) (Levine and Abbeel [285]).

The learning method is summarized in Alg. 2. Notably, during refinement, fewer optimization iterations $n_{\boldsymbol{\theta}} < n_\mathrm{h}$ can be used due to the faster convergence observed when using learned initialization relative to the heuristic.

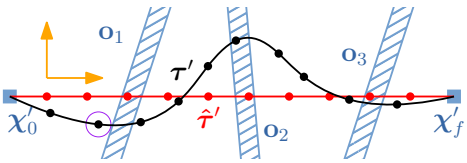
Algorithm 2: Using supervised behaviour cloning to train LearnedInit.

Input: A set of sampled tasks $\mathcal{D}_{\mathbf{x}}$
Output: Learned initializer parameters $\boldsymbol{\theta}$

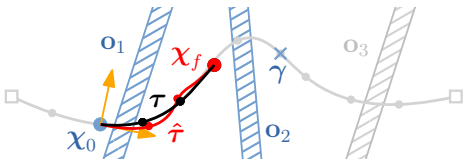
$\mathcal{D}_{\boldsymbol{\tau}} \leftarrow \text{towr}(A_{\text{h}}(\mathcal{D}_{\mathbf{x}}); n_{\text{h}})$ ▷ Using heuristic.
for n_{train} **do**
  $\mathcal{D}_{\text{good}} \leftarrow \text{filter}(\mathcal{D}_{\mathbf{x}}, \mathcal{D}_{\boldsymbol{\tau}}; J_{\max})$ ▷ Using Eq. 5.3.
  $\boldsymbol{\theta} \leftarrow \text{supervizedLearning}(\mathcal{D}_{\text{good}})$ ▷ Using Eq. 5.2.
  $\mathcal{D}_{\boldsymbol{\tau}} \leftarrow \text{towr}(A_{\boldsymbol{\theta}}(\mathcal{D}_{\mathbf{x}}); n_{\boldsymbol{\theta}})$ ▷ Using LearnedInit.
end for
return $\boldsymbol{\theta}$

For each of the test environments, detailed in Sec. 3.4.2.1, a set of about 2000 tasks were sampled from the task space. This space expresses variation of the initial base location and yaw angle. The distribution $J(\mathcal{D}_{\boldsymbol{\tau}})$ of costs resulting from optimization with heuristic and $n_{\text{h}} = 100$ generally exhibited a long tail of outliers, as seen in Fig. 5.7, that correspond to poor local minima. The filtering threshold $J_{\max}$ was set at the start of this tail, reducing the initial training set $\mathcal{D}_{\text{good}}$ to about 300 to 500 sampled tasks, and a two-layer network was fit to this small dataset. Subsequent learning cycles used $n_{\boldsymbol{\theta}} = 25$ due to fast convergence, as well as a 3-layer network due to the larger number of samples passing through the filter.

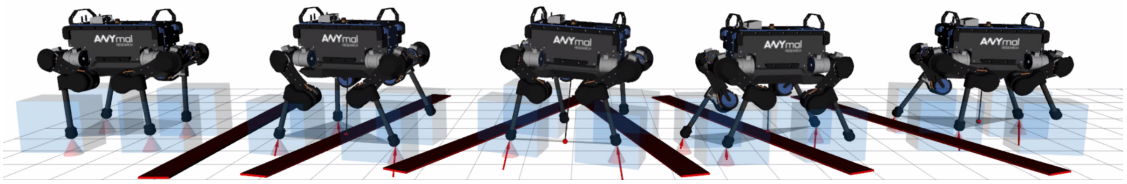
The loss-weighting matrix $\mathbf{W}$ consistently weighted all optimization variables within a given category. An initial check was conducted in which some categories were initialized by $A_{\boldsymbol{\theta}}$ and others by A_{h}. This revealed that the base linear positions, stepping phase durations, and contact forces were the most crucial values to improve through learned initialization. The implication is that it is easier for the optimizer to move along some dimensions of the solution space than others, and that the initializer should prioritize the accuracy of the strongest nonlinear influences. Ultimately, the best performance was obtained using coarsely tuned weights that included all categories.



(a) Stationary Horizon (SH)



(b) Receding Horizon (RH)



(c) Snapshots from a typical rollout of LMTR which produces RH trajectory segments enabling ANYmal B to traverse a series of obstacles.

Figure 5.4: Illustration of two scopes with which to view trajectories that cross discrete obstacles (striped) illustrated in (c) [4]. Blue indicates components of the 'task', i.e., inputs. Red and black denote outputs of the initializer and optimizer, respectively. (a) SH scope, with static robot states (squares) at the boundaries of a long trajectory, and naive initialization. (b) RH scope, with a trajectory segment bounded by dynamic states (circles), and learning-based initialization (LMTR).

5.3.2 Multimodal imitation learning

The approach outlined in the previous section is limited as the regressor can only be trained to reconstruct a unimodal family of long trajectories. Our proposed framework refines these trajectories to adapt them to the task at hand. Using shorter horizons reduces the necessary computation; however, it also decreases the expressiveness of the trajectory as the dynamic (involving positions and velocities) final boundary conditions are difficult to define. We approach these issues via imitation learning of a set of long-horizon expert trajectories that have been processed into segments, as originally detailed by Surovik et al. [4]. This produces a regression model (LMTR) that expresses a diverse continuum of behaviours suited for handling a range of obstacle configurations.

5.3.2.1 Change of scope

Key to the imitation learning setup is the use of segmentation (Sec. 4.3.1) to convert expert data from stationary-horizon (SH) to RH scope, as sketched in Fig. 5.4, with SH variables using $\cdot'$ notation. SH involves static boundary states

$\boldsymbol{\chi}'$ and long trajectories $\boldsymbol{\tau}'$, while in RH the boundary states $\boldsymbol{\chi}$ are dynamic and the trajectories $\boldsymbol{\tau}$ are shorter segments ($\boldsymbol{\tau} \in \mathbb{R}^{312}$ when the base horizon is $1.2\,\mathrm{s}$). Values provided by an initialization scheme are denoted $\hat{\cdot}$.

The initial boundary state is $\boldsymbol{\chi}_0 = \left({}^{\mathcal{B}}\dot{\mathbf{r}}, \boldsymbol{\theta}, {}^{\mathcal{B}}\boldsymbol{\omega}, {}^{\mathcal{B}}\mathbf{p}_i, \ldots, {}^{\mathcal{B}}\mathbf{p}_{n_{\mathrm{ee}}}, \Delta t_i, \ldots, \Delta t_{n_{\mathrm{ee}}}\right) \in \mathbb{R}^{24}$ where ${}^{\mathcal{B}}\dot{\mathbf{r}} \in \mathbb{R}^3$ is the linear velocity of the base in the base frame $\mathcal{B}$, $\boldsymbol{\theta} = (\phi\ \theta\ \psi)^\intercal \in \mathbb{R}^3$ is a vector of Euler angles (roll, pitch, yaw), ${}^{\mathcal{B}}\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the base in the base frame, ${}^{\mathcal{B}}\mathbf{p}_i \in \mathbb{R}^3$ is the linear position of the ith end effector's current foothold (or the planned foothold if it is in swing phase), and $\Delta t_i \in \mathbb{R}$ is the elapsed time since the ith end effector made contact (or the time it will take till the next planned contact event occurs if it is in swing phase). Note that the linear position of the base in the base frame ${}^{\mathcal{B}}\mathbf{r} = (0\ 0\ 0)^\intercal \in \mathbb{R}^3$ is not included in the task.

$\mathbf{o}_i = \left({}^{\mathcal{B}}x_{\mathbf{o}}, {}^{\mathcal{B}}y_{\mathbf{o}}, {}^{\mathcal{B}}h_{\mathbf{o}}, {}^{\mathcal{B}}\psi_{\mathbf{o}}\right) \in \mathbb{R}^4$ are discrete terrain elements which specify the position[1], height and yaw orientation of the obstacle (edge) in the base frame, respectively.

In RH scope, the task $\mathbf{x} = (\boldsymbol{\chi}_0, \boldsymbol{\gamma}, \mathbf{o}_i, \mathbf{o}_{i+1}) \in \mathbb{R}^{35}$ includes $\mathbf{o}_i$ within a local window and an $SE(2)$ goal $\boldsymbol{\gamma} = \left({}^{\mathcal{B}}x_{\boldsymbol{\gamma}}, {}^{\mathcal{B}}y_{\boldsymbol{\gamma}}, {}^{\mathcal{B}}\psi_{\boldsymbol{\gamma}}\right) \in \mathbb{R}^3$ taken from a base pose on the SH trajectory. With $\boldsymbol{\gamma}$ occurring further along than $\boldsymbol{\chi}_f$, the size of $\boldsymbol{\tau}$ can be controlled separately from the amount of foresight encompassed by $\mathbf{x}$.

5.3.2.2 Expert dataset

Training data are generated in bulk as shown in Fig. 5.5. Within SH scope, tasks $\mathbf{x}'$ are sampled from a distribution of static robot states and terrains and $\hat{\boldsymbol{\tau}}'$ are initialized naively with linear interpolation and standard gait timings. Allowed a high number $n_{\max}$ of iterations per sample, TOWR with analytical costs [1] returns expert trajectories $\boldsymbol{\tau}'$. The segmentation process of Sec. 4.3.1 is then used to generate the RH $\mathcal{D}$, i.e., pairs of RH tasks and segments $\{\mathbf{x}, \boldsymbol{\tau}\}$, desired for imitation learning (IL).

[1] To determine the 'position' coordinates of the obstacle (edge), the perpendicular point on the edge (closest to the robot) is identified using real-time perception and the edge-detection algorithm discussed in Sec. 3.3.2.2.

5.3.2.3 Generalization via LMTR

Due to the nonconvexity of the NLP, similar $\mathbf{x}$ may be paired with substantially different $\boldsymbol{\tau}$, reflecting a rich inventory of behaviours. This prevents directly fitting a 'unimodal' regressor $\mathbf{f}_{\text{uni}}\colon \mathbf{x} \mapsto \hat{\boldsymbol{\tau}}$, which must smoothly vary based on $\mathbf{x}$ alone. Sohn et al. [231] and Lynch et al. [233] demonstrated a solution capable of representing multimodal distributions. This solution was adopted by Surovik et al. [4] and integrated into this work. The idea is based on the use of a 'multimodal' regressor $\mathbf{f}_{\text{multi}}\colon (\mathbf{x};\mathbf{z}) \mapsto \hat{\boldsymbol{\tau}}$ which is conditioned on the 'mode' variable $\mathbf{z}$. Instead of manually specifying the form of $\mathbf{z}$, it can be represented as a latent variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$ in a CVAE. The CVAE consists of an autoencoder which is essentially an identity function that learns the features of the input data and stores this compressed representation in the hidden latent space $\mathcal{Z}$ where 'similar' data appear 'closer together'[2]. In our case, the latent space is the embedding of the pairs of tasks and trajectory segments within a lower-dimensional manifold and $\mathbf{z}$ represents the 'style' of the trajectory, e.g., how long the stride of each foot should be, as shown in Fig. A.5a in Appendix A.5.2.

The CVAE consists of an encoder $Q_{\boldsymbol{\theta}}\colon (\boldsymbol{\tau};\mathbf{x}) \mapsto (\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$ and a decoder $P_{\boldsymbol{\phi}}\colon (\mathbf{z};\mathbf{x}) \mapsto \hat{\boldsymbol{\tau}}$ whose parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are trained using Adam optimization algorithm with an L1 reconstruction loss and a Kullback-Leibler (KL) divergence (a type of variational inference) regularization term $D_{\text{KL}}(\cdot)$ weighted by

[2]Note that 'closeness' is not necessarily based on the Euclidean distance metric.

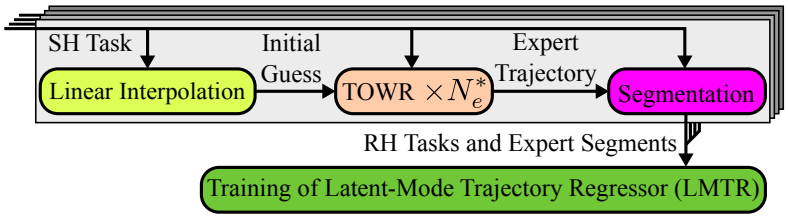


Figure 5.5: A diagram of bulk data generation for imitation learning. Stationary-horizon tasks are sampled and extensively optimized into high-quality trajectories. The segmentation process converts data into receding-horizon segments and tasks, used to train the multimodal trajectory regressor LMTR.

$\kappa = 0.001$, and $1\,\%$ noise added to task inputs:

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} \Big[\|\boldsymbol{\tau} - \hat{\boldsymbol{\tau}}\|_1 + \kappa D_{\mathrm{KL}}\big(\mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}\right) \parallel \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)\big) \Big]. \tag{5.4}$$

Here, $\mathcal{D}$ is the dataset, $\boldsymbol{\tau}$ and $\hat{\boldsymbol{\tau}}$ are the original and reconstructed trajectories, $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\sigma}_{\mathbf{z}}$ are the outputs of the encoder that parameterize the normal distribution (of the hidden, latent-space variable $\mathbf{z}$) which is compared with its prior standard normal distribution with zero mean $\mathbf{0}$ and unit variance $\mathbf{I}$. This training process essentially disentangles the degrees of freedom between different locally optimal solutions.

The decoder module of this model, hereby known as the LMTR—as introduced by Surovik et al. [4]—then serves as the regressor P_ϕ deployed online as shown in Fig. 5.6b. During deployment, when the regressor is presented with a novel task $\mathbf{x}$, it requires a mode-selecting policy $\boldsymbol{\pi}$. As shown by Lynch et al. [233], this policy can be trained in parallel with the CVAE to reconstruct the (Gaussian) output distribution of the encoder $Q_{\boldsymbol{\theta}}$ such that $\boldsymbol{\pi}_{\mathcal{N}} \colon \mathbf{x} \mapsto (\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$. The selection of $\mathbf{z}$ is done using k-NN lookup on expert pairs $\mathcal{D}_{\mathbf{z}} = \{\mathbf{x}, \mathbf{z}\}$ (which is responsible for the majority of the online memory footprint of the proposed initialization method) via the mode-selecting policy

$$\boldsymbol{\pi}(\mathbf{x}) = \mathbf{z}_i \mid \underset{\mathbf{x}_i \in \mathcal{D}_{\mathbf{z}}}{\operatorname{argmin}}\, d\left(\mathbf{x}_i, \mathbf{x}\right), \tag{5.5}$$

where $\mathbf{x}_i$ is a task encountered during training and $d(\cdot)$ is a weighted distance metric; in our proof of concept, we used the L1 norm and weighted certain elements of $\mathbf{x}$ (information related to the state, obstacles and goal) based on a data-driven analysis of successful execution in a physics simulator. Fig. A.5c demonstrates the need for the mode-selecting policy where a task We acknowledge the limitation of this relatively simple approach of being unimodal when the distribution of the modes can be multimodal as shown in Fig A.5c and suggest that a more appropriate (but also more challenging to apply) multimodal policy is investigated, e.g., normalizing flows (Mazoure et al. [286]). The decoder reconstructs a near-optimal segment as shown in Fig. 5.4b. Relative



(a) CVAE training **(b)** LMTR deployment

Figure 5.6: Diagrams showing the key stages of multimodal data-driven trajectory initialization (DDTI), an approach adopted from Surovik et al. [4], which involves (a) the training of a CVAE (orange encoder and green decoder) to reconstruct an input trajectory segment $\boldsymbol{\tau}$ and an input task $\mathbf{x}$ by generating a latent mode $\mathbf{z}$, and (b) the deployment of the LMTR (green decoder) using a mode-selecting policy $\boldsymbol{\pi}(\mathbf{x})$.

to the heuristic initializer presented in Sec. 4.3, more aggressive and adaptive motions can be expressed via the LMTR, although its deployability depends upon the quality of its training data.

5.4 Results

This section is divided into two parts. The results obtained on the ANYmal B quadruped using the open-loop LearnedInit initializer [1] are discussed in Sec. 5.4.1. The experiments involving the ANYmal C quadruped and the RH LMTR initializer [2] will be covered in Sec. 5.4.2.

5.4.1 Open-loop planning using LearnedInit

This section provides a detailed look at the effect of Alg. 2 upon the optimization process. Descriptions of all the environments used during tests are provided in Sec. 3.4.2.1. The results presented here reflect the *Single Pallet* problem of ascending a ledge. For this problem, the starting base position ranges from between $-0.5\,\mathrm{m}$ to $-1.5\,\mathrm{m}$ back from the step and $\pm\,0.75\,\mathrm{m}$ laterally from the goal position, with yaw variation in the range $\pm\,30°$. All experiments were conducted using the ANYmal B quadruped and the system shown in Fig. 5.1.

Figure 5.7 shows the benefits of LearnedInit in terms of convergence rate, final cost, and the maximum force experienced by the robot. heuristic initialization often results in poor local minima, and sometimes causes divergence of the optimization process as indicated by high costs and violations of the maximum

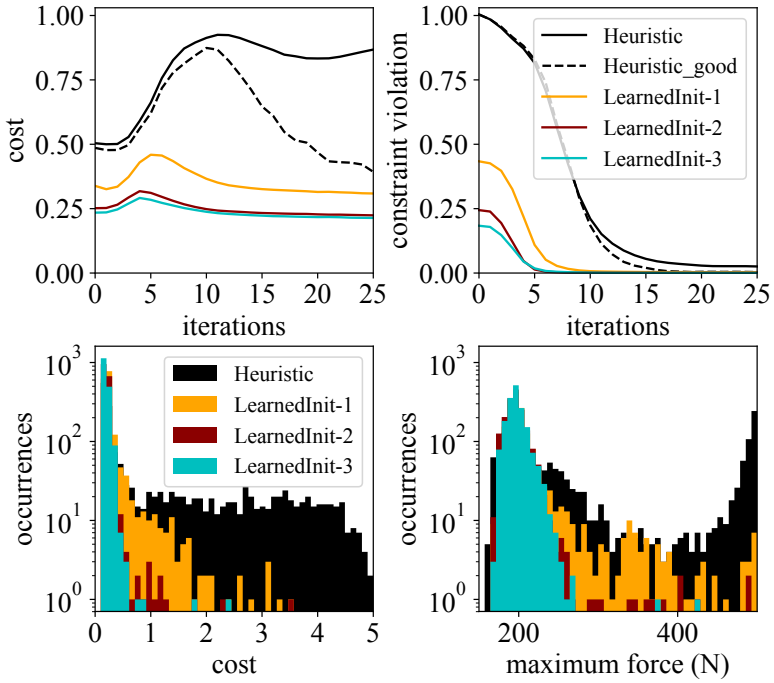


Figure 5.7: Optimization performance under different initialization schemes on a large task set. With heuristic, many sampled tasks diverge, producing high costs and forces. Each cycle of training for LearnedInit improves the convergence rate, final cost, and maximal force, while eliminating outliers.

force constraint (truncated from the plot). Due to the use of filtering, LearnedInit exhibits better and more consistent performance, succeeding on most tasks that were failed by heuristic. Retraining LearnedInit after optimizing its original output set increases these benefits and eliminates nearly all outliers; this was done $n_{\text{train}} = 3$ times, creating the 3 LearnedInit variants shown in Fig. 5.7.

5.4.1.1 Test in dynamic simulation

To evaluate the performance of the learning-based approach, solutions from three initialization methods were tested on the Double Pallet shown in Fig. 5.8. While varying the maximum iteration count, a sample set of 100 tasks was optimized for each. Trials that did not reach the goal state were marked as failures. Tracking accuracy was measured for successful runs to indicate how well-suited the planned trajectories were to the closed-loop system.

Fig. 5.9 demonstrates that the initializing solutions produced by LearnedInit are directly executable more than half the time without any further optimization,

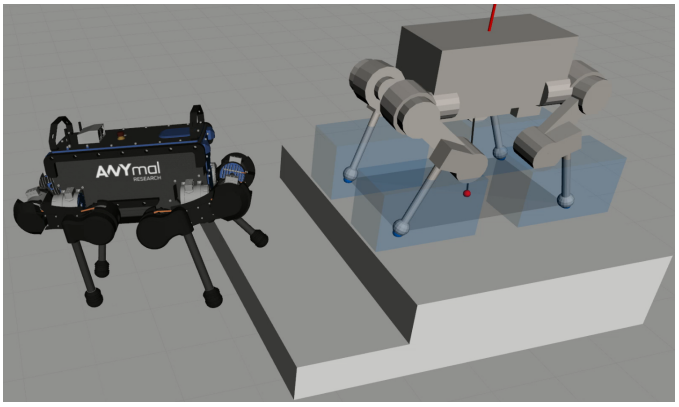


Figure 5.8: ANYmal B quadruped executing planned trajectories (with the goal state visualized by the grey robot) initialized by LearnedInit, on the Double Pallet in the Gazebo physics simulator.

and the success rate approaches $95\,\%$ within about 5 iterations of re-optimization. This is because the network has been retrained 3 times using data of only successful trajectories for a particular problem—essentially overfitting to the problem. Similar results were achieved for all tested environments types (using separate networks). When phase durations were fixed at their initialized values (represented by the dotted lines in Fig. 5.9), success rates remained high regardless of the number of iterations. However, when the phase durations were optimized (represented by the solid lines in Fig. 5.9), a higher number of iterations caused the optimization to unexpectedly diverge, resulting in a drop in success rates. This convergence issue was attributed to the extreme nonlinearity of the problem and the numerical instability resulting from some elements of the constraints Jacobian matrix changing from zero to non-zero values. This is the intended behaviour of the phase-based end-effector parameterization (Winkler et al. [86]) which initializes all elements that can change during the course of the optimization as 'potentially' non-zero. This results in a denser sparsity structure being provided to the optimizer. Optimizing over phase durations affects the robustness of the convergence and the time it takes to find a solution. More research on the solver's robustness to this atypical use case is needed.

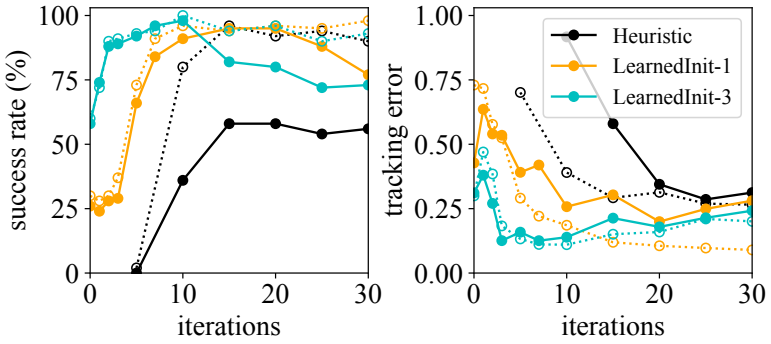


Figure 5.9: Performance of trajectories climbing a single pallet under whole-body control in simulation (100 tasks per data point). LearnedInit produces solutions that are easier to track, reaching high success rates after minimal online refinement. Fixing phase durations at initialized values (dotted lines) improves optimization stability.

5.4.2 Receding-horizon replanning using LMTR

The receding-horizon (RH) planner was tested with both trajectory initialization options, heuristic[3] and LMTR. The robot was tasked with computing dynamic motions toward a goal. The quality of the initial guesses and the refined trajectories were quantified by analysing the cost and constraint violation computed at each iteration of the nonlinear solver. Execution of the replanning pipeline, shown in Fig. 5.2, on the ANYmal C quadruped demonstrated its viability on hardware for most scenarios.

In the most challenging case—aggressive behaviours on terrain planned by LMTR—viability on hardware has not been fully achieved. Instead, we provide *numerical*[4] trials in which the replanner receives the task that would be seen if the previous plan were followed nominally for the duration of one replanning cycle, and refinement is conducted using the same computation time limit as hardware trials.

An LMTR regressor was trained for both environment types (described below), using $\mathbf{z} \in \mathbb{R}^{12}$ and with $\boldsymbol{\tau} \in \mathbb{R}^{312}$ containing a 1.2 s base horizon and stance-swing-stance phases. The environments were:

[3] In the RH context, heuristic refers to the RH variant proposed in Chapter 4.

[4] By describing certain trials as *numerical*, we refer to controlled tests in which we imitate real-time deployment, ensuring that the replanner finds feasible solutions within a realistic time budget *without* executing them in a physics simulator in order to guarantee deterministic outcomes.

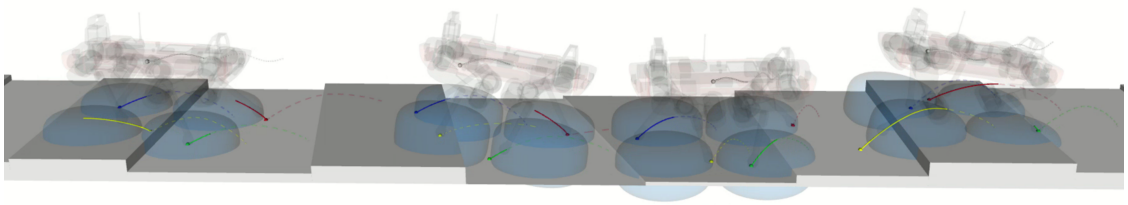


Figure 5.10: Snapshots of a typical numerical trial of the LMTR which leverages similar prior experience to quickly plan very dynamic trajectories in a receding-horizon fashion while being informed by real-time perception. Steps were of arbitrary depth, height and orientation. The generated plans were refined online by the solver. The dashed lines indicate planned base and feet trajectories while the solid lines emphasize the portion of the plan executed during each MPC control cycle. The blue superquadrics approximate the operational space of the end-effectors by serving as range-of-motion constraints.

Flat ground For the *flat-ground* scenario, the offline expert's initial conditions used random offsets of the lateral position and the yaw relative to a desired line of travel. Without any obstacles, $\mathbf{x} \in \mathbb{R}^{27}$.

Terrain The *terrain* scenario included randomized stairstep features $\mathbf{o}_i$ spaced apart by $0.25\,\mathrm{m}$ to $1\,\mathrm{m}$, with height changes of $-0.16\,\mathrm{m}$ to $0.16\,\mathrm{m}$ and edge orientations of $-15°$ to $15°$. During deployment, these terrain features are extracted from the elevation map, and the closest two are included in $\mathbf{x} \in \mathbb{R}^{35}$.

Numerical results shown in Fig. 5.10 demonstrate motions computed at $2.5\,\mathrm{Hz}$ that reach mathematical feasibility under the SRBD-based formulation. Attaining reliable deployment in physics simulators and on hardware is a prime focus of our ongoing work.

When facing a new scenario within the bounds of the experience set, the network consistently reconstructed a valid initial guess which was refined to a satisfactory level of optimality within the iteration budget available for online use, as shown in Fig. 5.11. Occasionally, some cycles (e.g., 42) presented conditions that were challenging to satisfy, potentially due to gaps in the regressor's coverage. If refinement failed, the remaining portion of the previous valid trajectory was used for an additional replanning cycle, which was possible due to a greater than 2-to-1 ratio of these two durations.

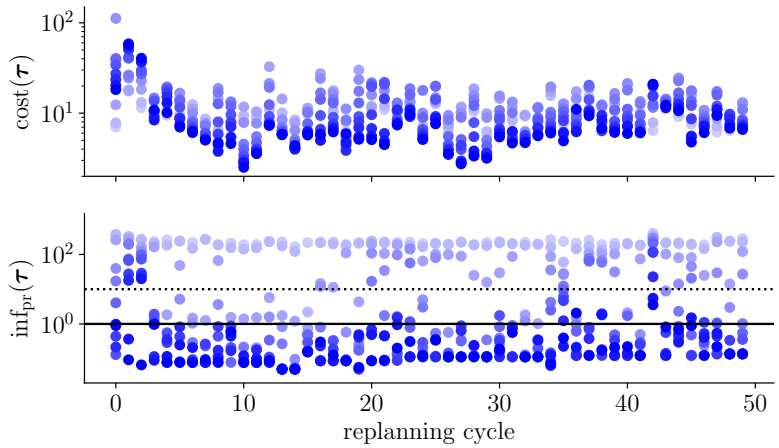


Figure 5.11: The values of the cost and primal infeasibility (inf_{pr}) during refinement of initial guesses $\boldsymbol{\tau}$ provided by LMTR while traversing the terrain shown in Fig. 5.10 for 50 replanning cycles. The gradient (light to dark) represents subsequent solver iterations limited to a maximum of 10 per cycle. Horizontal lines show preferred (solid) and acceptable (dotted) thresholds of $\text{inf}_{\text{pr}}(\boldsymbol{\tau})$.

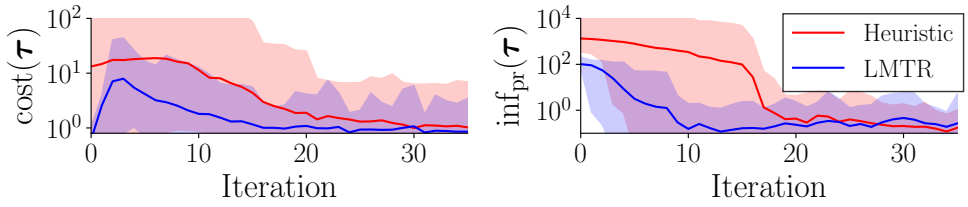


Figure 5.12: Mean values of the cost and primal infeasibility (inf_{pr}) at each refinement iteration of 1.2 s long initial guesses $\boldsymbol{\tau}$ provided by the heuristic (linear interpolation) and LMTR (near-optimal trajectories) on flat terrain at 2 Hz. Here, both cases use the same mid-motion final state $\boldsymbol{\chi}_f$ intelligently provided by LMTR as shown in Fig. 5.4—a key strength of the regressor.

Figure 5.12 shows the evolution of the cost and the constraint violation along the iteration of the solver for the heuristic and the LMTR initialization. While the heuristic could be used with any predefined gait, to have a fair comparison with respect to the difficulty of the gait, both were evaluated on the contact timing and footholds generated by the regressor. The results show that the LMTR initialization was able to converge approximately twice as fast as the heuristic.

As shown in Fig. 5.13, the LMTR was used to enable a $4\,\text{m}$, $12\,\text{s}$ trot-like motion on flat ground which resulted in the base velocity in the x-direction $\dot{r}_x(t)$ reaching $0.7\,\text{m}\,\text{s}^{-1}$ and swing lengths of $0.2\,\text{m}$ to $0.6\,\text{m}$ with durations $\boldsymbol{\Delta}\mathbf{T}_{\text{sw}}$ of $0.3\,\text{s}$ to $0.5\,\text{s}$. Thanks to the LMTR initialization, such a plan can be adapted

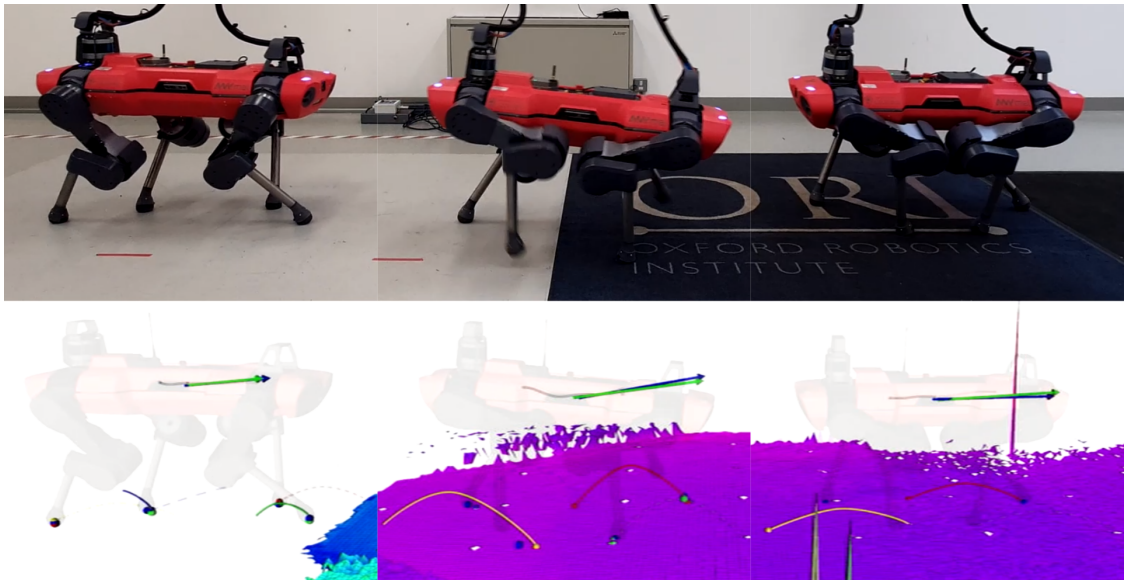


Figure 5.13: The initial guesses provided by the LMTR, trained on flat ground, were adapted online. The real robot (top) used real-time perception and tracked the desired CoM and end-effector trajectories (bottom); the green and blue arrows represent the desired and measured velocities.

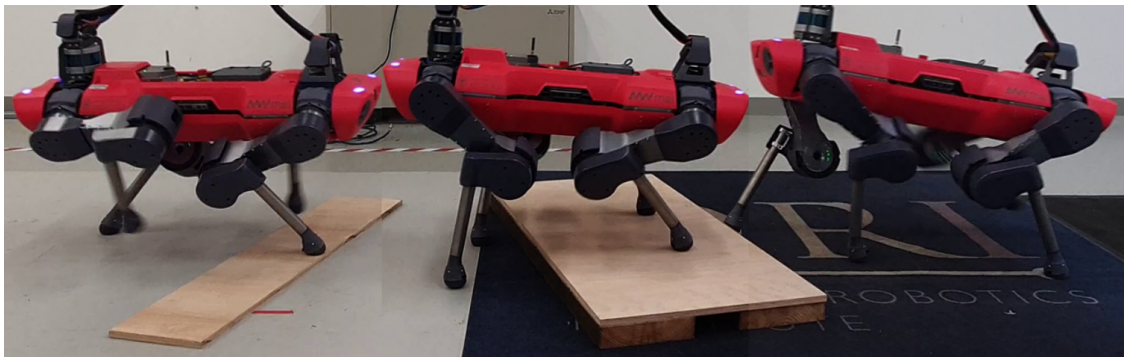


Figure 5.14: The initial guesses provided by the LMTR trained on flat-ground can be adapted to mild height variation of 0.02 m to 0.08 m by the receding-horizon optimization, showing the ability to cope with minor imprecisions.

online by the optimizer at frequencies from 2 Hz to 3.3 Hz with planning horizons of 1.2 s to take into account the model of the terrain received from the sensors and walk over obstacles as shown in Fig. 5.14.

5.5 Discussion

Using the LearnedInit initializer, the convergence of the optimization solver has been improved and accelerated. This confirmed that the primal variables of

the NLP solver IPOPT can be effectively initialized with a near-optimal guess. However, this learning-based method was limited to a specific problem for which a single family of solutions was determined. The inability to incorporate state feedback made the approach susceptible to disturbances.

The LMTR initialization scheme addressed these limitations. It was demonstrated that the regressor can efficiently represent many categories of trajectories and be used for RH planning. Occasionally, we have experienced difficulty obtaining good initializations using the LMTR initialization. In future work, we would consider replacing the *k*-nearest neighbors (*k*-NN) mode selector with an alternative better suited to multimodal distributions (e.g., normalizing flows).

When generating the expert trajectories used for IL, the superquadric range-of-motion constraint, introduced in Sec. 3.3.2.3, substantially improved numerical behaviour of the optimizer and prevented divergent cases that sometimes occurred with a simple box-shaped constraint; specifically, the combination of the two constraints shown in Fig. A.2 appeared beneficial.

The optimizer autonomously decided the sequence of steps and the durations of stance and swing phases for each of the end-effectors. The regressor learned the optimal relationship between tasks and aperiodic phase durations. Therefore, during online refinement by the optimizer, the phase durations were kept constant; however, although we argue against it, the proposed system does allow the phase durations to be further modified, albeit at a significant computational expense and a risk of solver instability discussed earlier.

The capacity of the optimizer to refine the initialization allowed to adapt the plan in real time when the robot faced obstacles which were not experienced during training as demonstrated in Fig. 5.14.

During deployment on the real ANYmal C quadruped, achieving robust execution was challenging. The robot would fall when the trajectories involved high base linear velocities (e.g., $0.6\,\mathrm{m\,s^{-1}}$), narrow stances, long strides (e.g., $0.5\,\mathrm{m}$) and fast swinging legs. We believe this is in part due to the ANYmal C quadruped having heavy legs due to the large knee flexion/extension (KFE)

actuators mounted directly in its knees. Positioned far from the CoM, its inertial effects limit the possible joint acceleration and act like a disturbance. Many quadrupedal robots e.g., Boston Dynamics Spot, mount the KFE motors close to the robot's base and actuate the joints using linkages to reduce the weight of the legs for more agile locomotion. Our planner, which uses the simple SRBD model, does not account for the effects of the legs' inertias. Using a more complete model e.g., CD or RBD, would address this issue; however, adding more decision variables and incorporating Coriolis, centrifugal and nonlinear effects into the model increases the complexity of the optimization problem which makes it more difficult to solve in real time.

5.6 Conclusion

This chapter focused on using imitation learning to accelerate the convergence of an optimization solver. Behaviour cloning of optimal whole-body locomotion trajectories—generated using the approach described in Chapter 3—produced two regressors which served as schemes which intelligently initialized the primal variables of the NLP problem solved by the IPOPT optimizer.

The LearnedInit initializer was trained in a supervised fashion on filtered experiences gathered offline. It was used for the open-loop reconstruction of long-horizon trajectories to offset the strong nonconvexity and increased computational expense associated with the optimization problem. It was deployed on the ANYmal B quadruped, enabling it to dynamically traverse uneven terrain including industrial pallets. While it relied on models of the environment it used real-time perception to identify their position and orientation.

The LMTR initializer was used for receding-horizon replanning. It imitated segments of diverse long-horizon trajectories, proposing a behaviour dependent on the task at hand, which included prominent terrain features such as edges, if present. It was deployed on the ANYmal C quadruped, using fully onboard sensing and computation. The LMTR enabled more dynamic, trot-like locomotion but with aperiodic contact sequences that hint at its capacity to

express a diverse inventory of viable behaviours. Numerical results show how its expressiveness is suitable for adapting to challenging terrain while moving at a fast pace.

Future work will seek to realize these precise and highly dynamic motions by explicitly compensating for the approximations in the planning model, in lower-level control, and by updating the LMTR based on deployed experiences.

6

Dynamics models for inertia shaping

In this chapter, we extend the SRBD model, used in the preceding chapters, to account for the effects of leg inertias on the angular momentum of the robot.

We provide analytical derivations of the lump leg single-rigid-body dynamics (LL-SRBD) and centroidal dynamics (CD) models including the first-order derivative information needed by an NLP solver.

Both models account—to varying degrees—for the nonlinear effects of moving joints and their impact on the distribution of the inertia of the robot. While the LL-SRBD model approximates these effects to a large extent, it arguably offers ample fidelity for some robots. The CD model is very accurate but also highly nonlinear and therefore more challenging to be solved quickly.

The optimization variables are parameterized by Euclidean-space polynomials of Hermite form. This gives rise to an intuitive description of CoM and end-effector trajectories and enables the discovery of contact locations, an aperiodic gait sequence, and the timings of contact events.

If the joint torques were also optimized, e.g., to explicitly ensure that their limits are not exceeded, the proposed derivation could be extended to the full RBD model.

6.1 Introduction

Traditionally, simple and intuitive models were used to describe the dynamics of legged robots, and heuristic strategies were used for locomotion planning on flat terrain. As more challenging problems were being tackled—such as athletic traversal of uneven terrain informed by perception—a new generation of sophisticated robots emerged. These state-of-the-art robots carry sensors or payloads, and have stronger but larger actuators. Such components have considerable inertia which induce nonlinear effects that affect the motion of the robot. In order to generate realizable locomotion plans, these effects need to be accounted for by the dynamics model.

6.1.1 Contributions

- The derivation of analytical first-order partial derivatives of the LL-SRBD model.
- The deduction of a CD model parameterized by Euclidean-space variables, which introduces Cartesian foothold locations as decision variables, and the analytical derivation of the resulting first-order partial derivatives.
- The discretization of the aforementioned models using third-order polynomials of Hermite form which enables the optimization of end-effector contact phase durations.

6.2 Related works

One of the simplest models applied to legged robots is the spring-loaded inverted pendulum (SLIP) model (Blickhan [92]). While it has been used for dynamic translational motion, such as running and jumping, it ignores the angular momentum and cannot be applied to problems involving rotations. Other simple models with similar limitations include the centre of pressure (CoP) or zero-moment point (ZMP) stability criterion (Kajita et al. [95]).

In general, more accurate models involve nonlinearities, nonconvexities and discontinuities which adversely affects the performance of optimization-based techniques. The single-rigid-body dynamics (SRBD) model uses Newton-Euler equations to model translational and rotational dynamics. It has been used by Winkler et al. [86], Bledt et al. [87], Di Carlo et al. [181], and **Melon** et al. [1] for quadrupedal robots. This model struggles to capture the dynamics of robots with heavy legs (**Melon** et al. [2]). as it assumes the composite rigid body (CRB) inertia to be constant and the momentum due to joint velocities to be negligible. Orin et al. [77] introduced the centroidal dynamics (CD) model which describes the momentum of the CoM of the robot. Dai et al. [83], Kuindersma et al. [43] and Herzog et al. [84] used this model with full kinematics for highly dynamic whole-body motion planning for humanoid robots. Budhiraja et al. [78] proposed a rigorous mathematical framework which enforces consensus between the CD model and the rigid-body dynamics (RBD) model which—more explicitly—describes the motion of individual joints. The latter model has been applied offline to high-dimensional humanoids by Buss et al. [74], Mombaur [75] and Lengagne et al. [76].

Many methods have focused on extending simpler models with specialized components. To compensate for the wrench exerted by the swinging legs, Villarreal et al. [270] incorporated a feedforward term into the convex formulation by Di Carlo et al. [181]. Ahn et al. [287] proposed a composite rigid body (CRB) model, which improves the SRBD model, by integrating a centroidal inertia network. While the approach is computationally efficient, it requires a representative dataset for training. The model also disregards other nonlinear effects captured by more sophisticated models. Wang et al. [288] introduced the LL-SRBD model (called LL-SRBM therein) which extends the SRBD model by partly accounting for the inertias of the legs, i.e., representing each leg as a point mass positioned along the line between the end-effector and the hip, to enable highly dynamic motions like a twisting jump while adding little computational cost.

Recently, Zhou et al. [289] used an alternating strategy of solving the SRBD via TOWR (Winkler et al. [86]) with an updated inertia tensor and the RBD using CROCCODYL (Mastalli et al. [145]), until achieving consensus, to generate linear and angular momenta reference trajectories offline. They leveraged the equimomental ellipsoid parameterization of the inertia tensor proposed by Lee and Goswami [290]. While also using direct collocation, our approach differs by directly integrating and solving the complete CD with the help of the Pinocchio library (Carpentier et al. [151]).

6.3 Methodology

6.3.1 Preliminaries

The mathematical expressions derived in this section involve multiplying (third-order) tensors (multidimensional arrays) with matrices and vectors. This is enabled by the n-mode product presented in Def. 6.3.1.

Definition 6.3.1 (The n-mode product). As presented by Kolda and Bader [291], the n-mode product can be used to multiply tensors with matrices or vectors.

A product of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ produces a tensor $\mathbf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$, and is denoted as

$$\mathbf{Y} = \mathbf{X} \times_n \mathbf{U} \Leftrightarrow \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)}$$

where $\mathbf{Y}_{(n)}$ and $\mathbf{X}_{(n)}$ are unfolded tensors[1]. Elementwise, the (matrix) product is

$$(\mathbf{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} \, u_{j i_n}.$$

The n-mode vector product of the tensor $\mathbf{X}$ and a vector $\mathbf{v} \in \mathbb{R}^{I_n}$ is denoted as $\mathbf{X} \,\bar{\times}_n\, \mathbf{v}$. Its result is of order $N - 1$. It can be computed elementwise as follows

$$(\mathbf{X} \,\bar{\times}_n\, \mathbf{v})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} \, v_{i_n}.$$

[1] Using *matricization*, also known as *unfolding* or *flattening*, a tensor can be transformed into a matrix as shown by Kolda and Bader [291].

Specifically in our case, the tensor $\mathbf{X} = \frac{\partial \mathbf{A}}{\partial \mathbf{a}} \in \mathbb{R}^{3\times3\times3}$ is a partial derivative of a matrix $\mathbf{A}(\mathbf{a}) \in \mathbb{R}^{3\times3}$ with respect to a vector $\mathbf{a} \in \mathbb{R}^3$ and can be multiplied with another matrix $\mathbf{B} \in \mathbb{R}^{3\times3}$, as follows $\mathbf{X} \times_3 \mathbf{B}^\intercal \in \mathbb{R}^{3\times3\times3}$, or with another vector $\mathbf{b} \in \mathbb{R}^3$, as follows $\mathbf{X} \bar{\times}_2 \mathbf{b} \in \mathbb{R}^{3\times3}$. Note that the matrix $\times_n$ and vector $\bar{\times}_n$ n-mode product symbols are unrelated to the cross product symbol $\times$.

Definition 6.3.2 (The outer product of tensors). In addition to being applied to two vectors, the outer product of tensors $\otimes\colon\ (\mathbf{A}, \mathbf{b}) \mapsto \mathbf{X}$ can be used to multiply a matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$ with a vector $\mathbf{b} \in \mathbb{R}^3$ to produce a tensor $\mathbf{X} \in \mathbb{R}^{3\times3\times3}$.

Lemma 6.3.1 (The partial derivative of an inverse of a matrix with respect to a vector). Consider a matrix $\mathbf{A}(\mathbf{a}) \in \mathbb{R}^{3\times3}$, a vector $\mathbf{a} = (a_1\, a_2\, a_3)^\intercal \in \mathbb{R}^3$ and a scalar $a \in \mathbb{R}$. It can be shown through the following steps

$$\begin{aligned} \mathbf{A}\mathbf{A}^{-1} &= \mathbb{1}, \\ \frac{\partial \mathbf{A}}{\partial a_1}\mathbf{A}^{-1} + \mathbf{A}\frac{\partial \mathbf{A}^{-1}}{\partial a_1} &= \mathbb{0}, \\ \mathbf{A}\frac{\partial \mathbf{A}^{-1}}{\partial a_1} &= -\frac{\partial \mathbf{A}}{\partial a_1}\mathbf{A}^{-1}, \end{aligned}$$

that the partial derivative of an inverse of a matrix with respect to a scalar is

$$\frac{\partial \mathbf{A}^{-1}}{\partial a_1} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial a_1}\mathbf{A}^{-1}.$$

Similarly, the element-wise partial derivative with respect to a vector is

$$\begin{aligned} \frac{\partial \mathbf{A}^{-1}_{ij}}{\partial \mathbf{a}_k} &= -\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \mathbf{a}_k}\mathbf{A}^{-1}\right)_{ij} \\ &= -\mathbf{A}^{-1}_{il}\frac{\partial \mathbf{A}_{lm}}{\partial \mathbf{a}_k}\mathbf{A}^{-1}_{mj}. \end{aligned}$$

Therefore, using the notation proposed in Definition 6.3.1, we can write

$$\frac{\partial \mathbf{A}^{-1}}{\partial \mathbf{a}} = -\left(\frac{\partial \mathbf{A}}{\partial \mathbf{a}} \times_1 \mathbf{A}^{-1}\right) \times_2 \left(\mathbf{A}^{-1}\right)^\intercal.$$

Definition 6.3.3 (Skew-symmetric matrix). Consider vectors $\mathbf{a} = (a_1\, a_2\, a_3)^\intercal, \mathbf{b} = (b_1\, b_2\, b_3)^\intercal \in \mathbb{R}^3$. The cross product between them can be written as $\mathbf{a}\times\mathbf{b} = [\mathbf{a}]_\times\, \mathbf{b}$, where

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \quad \in \mathbb{R}^{3\times 3}$$

is the skew-symmetric matrix.

Lemma 6.3.2 (The partial derivatives of cross and dot products with respect to a vector). Consider vectors $\mathbf{a}(\mathbf{c}), \mathbf{b}(\mathbf{c}), \mathbf{c} \in \mathbb{R}^3$. The partial derivative of the cross product, with respect to $\mathbf{c}$, eliminating the cross product using the skew-symmetric matix $\mathbf{a}\times\mathbf{b} = [\mathbf{a}]_\times\, \mathbf{b} = [\mathbf{b}]_\times^\intercal\, \mathbf{a} = -[\mathbf{b}]_\times\, \mathbf{a}$, is

$$\frac{\partial\,(\mathbf{a}\times\mathbf{b})}{\partial\mathbf{c}} = [\mathbf{a}]_\times \frac{\partial\mathbf{b}}{\partial\mathbf{c}} - [\mathbf{b}]_\times \frac{\partial\mathbf{a}}{\partial\mathbf{c}}.$$

The partial derivative of the dot product, with respect to $\mathbf{c}$, eliminating the dot product using the commutative property $\mathbf{a}\cdot\mathbf{b} = \mathbf{a}^\intercal\mathbf{b} = \mathbf{b}\cdot\mathbf{a} = \mathbf{b}^\intercal\mathbf{a}$, is

$$\frac{\partial\,(\mathbf{a}\cdot\mathbf{b})}{\partial\mathbf{c}} = \mathbf{a}^\intercal\frac{\partial\mathbf{b}}{\partial\mathbf{c}} + \mathbf{b}^\intercal\frac{\partial\mathbf{a}}{\partial\mathbf{c}}.$$

Definition 6.3.4 (The triple product expansion). A triple cross product can be expanded into vector products using the 'acb-abc' or 'bac-cab' rules as follows

$$\begin{aligned}\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) &= (\mathbf{a}\cdot\mathbf{c})\,\mathbf{b} - (\mathbf{a}\cdot\mathbf{b})\,\mathbf{c} \\ &= \mathbf{b}\,(\mathbf{a}\cdot\mathbf{c}) - \mathbf{c}\,(\mathbf{a}\cdot\mathbf{b})\,.\end{aligned}$$

Definition 6.3.5 (The time derivative of a rotation matrix). As derived by Zhao [292], the time derivative of a rotation matrix $\dot{\mathbf{R}}$ can be expressed in terms of angular velocity $\boldsymbol{\omega}$ and the rotation matrix $\mathbf{R}$, as follows

$$\begin{aligned}{}^{\mathcal{W}}\dot{\mathbf{R}}_{\mathcal{B}} &= \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}, \\ {}^{\mathcal{W}}\dot{\mathbf{R}}_{\mathcal{B}} &= {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}} \left[{}^{\mathcal{B}}\boldsymbol{\omega}\right]_\times, \\ {}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{W}} &= -{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times, \\ {}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{W}} &= -\left[{}^{\mathcal{B}}\boldsymbol{\omega}\right]_\times {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}},\end{aligned}$$

where $\mathcal{W}$ is the world (fixed) frame and $\mathcal{B}$ it the body or CoM (rotating) frame.

Definition 6.3.6 (Numeric vector and matrix notation). For conciseness, we denote a vector of zeros as $\mathbf{0} = (0 \ldots 0)^\intercal \in \mathbb{R}^n$, a vector of ones as $\mathbf{1} = (1 \ldots 1)^\intercal \in \mathbb{R}^n$, a matrix of zeros as $\mathbb{0} \in \mathbb{R}^{m \times n}$, and the identity matrix as $\mathbb{1} = \text{diag}(1, 1, \ldots, 1) \in \mathbb{R}^{n \times n}$, where m and n are typically 3.

Definition 6.3.7 (Mass moment of inertia). The mass moment of inertia[2], denoted as $\mathbf{I} \in \mathbb{R}^{3\times3}$, of a rigid object made up of N distributed masses approximated as point masses m_k, positioned at $\boldsymbol{\rho}_k = (\rho_{k,1}, \rho_{k,2}, \rho_{k,3}) = (\rho_{k,x}, \rho_{k,y}, \rho_{k,z})$ relative to the point about which it is computed, can be defined elementwise as follows

$$I_{ij} = I^{\text{CoM}}_{k,ij} + \sum_{k=1}^{N} m_k \left((\boldsymbol{\rho}_k \cdot \boldsymbol{\rho}_k)\, \delta_{ij} - \rho_{k,i}\rho_{k,j} \right)$$

where $I^{\text{CoM}}_{k,ij}$ is a component of $\mathbf{I}^{\text{CoM}}_k \in \mathbb{R}^{3\times3}$, i.e., the inertia of the kth *distributed* mass calculated at the centre of mass (CoM), δ_{ij} is the Kronecker delta, and $\boldsymbol{\rho}_k \cdot \boldsymbol{\rho}_k = \boldsymbol{\rho}_k^\intercal \boldsymbol{\rho}_k$ is the dot product. Alternatively, the moment of inertia of the kth point mass is

$$\begin{aligned} \mathbf{I}_k &= \mathbf{I}^{\text{CoM}}_k + m_k \left((\boldsymbol{\rho}_k \cdot \boldsymbol{\rho}_k)\, \mathbb{1} - \boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_k \right) \\ &= \mathbf{I}^{\text{CoM}}_k + m_k \left[\boldsymbol{\rho}_k\right]_\times^\intercal \left[\boldsymbol{\rho}_k\right]_\times \\ &= \mathbf{I}^{\text{CoM}}_k - m_k \left[\boldsymbol{\rho}_k\right]_\times^2 \end{aligned}$$

where $\boldsymbol{\rho}_k \otimes \boldsymbol{\rho}_k = \boldsymbol{\rho}_k \boldsymbol{\rho}_k^\intercal$ is the outer product.

Definition 6.3.8 (Equimomental ellipsoid). As derived by Lee and Goswami [290], the *equimomental ellipsoid* is computed from the CRB inertia ${}^{\mathcal{B}}\mathbf{I}$ and the mean density ρ of a rigid body. It is an ellipsoid that has the same rotational inertia (equal mass moments of inertia) about any arbitrary axis as the rigid body. Its main purpose is to serve as a visualization aid when trying to understand the mass distribution of the rigid body and its mass moments of inertia about chosen axes.

[2]Also known as the moment of inertia or rotational inertia. It is often referred to as composite rigid body (CRB) inertia in robotics.

The size of the ellipsoid is defined by the lengths of its semi-axes $\mathbf{a} = (a_1\, a_2\, a_3) \in \mathbb{R}^3$. Each length is computed using

$$a_i = \frac{(\lambda_1\lambda_2\lambda_3)^{2/5}}{\lambda_i\,(8\pi\rho/15)^{1/5}}$$

where $(\lambda_1, \lambda_2, \lambda_3)$ are the eigenvalues of ${}^{\mathcal{B}}\mathbf{I}$. This formula is related to the inertia of an ellipsoid equation. Note that unlike when visualizing a matrix (e.g., an inertia tensor), $a_i \neq 1/\sqrt{\lambda_i}$ in the case of the *equimomental* ellipsoid.

The orientation of the ellipsoid in the base frame $\mathcal{B}$ is given by its eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. Note that when representing the orientation as a quaternion, one must use right-handed orthonormal vectors which satisfy $(\mathbf{e}_1 \times \mathbf{e}_2) \cdot \mathbf{e}_3 > 0$.

6.3.2 Direct collocation

As discussed in Sec. 3.3.1, we formulate the nonlinear programming (NLP) problem using the direct collection transcription method. More specifically, we represent the state $\mathbf{x}$ and control $\mathbf{u}$ as third-order polynomials of Hermite form. In this section, we will derive the three-dimensional mathematical description of our approach. This information will be helpful in explaining two dynamics models proposed in the following sections which use the same transcription method.

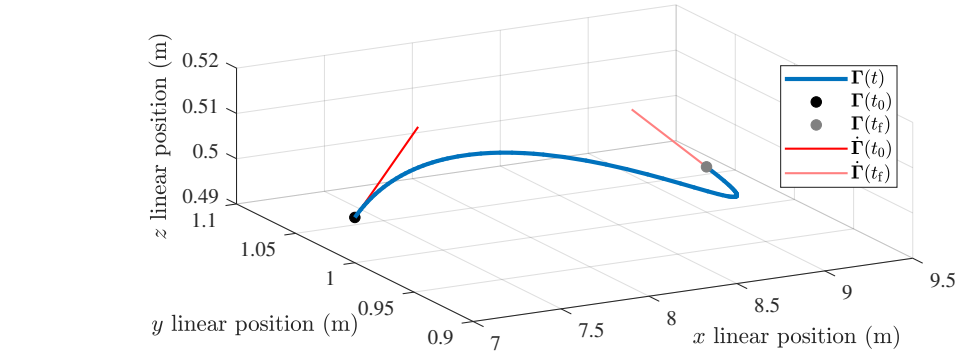
Consider a vector-valued function $\mathbf{\Gamma}(t; \mathbf{w}_{\mathbf{\Gamma}})$, of time t, parameterized by $\mathbf{w}_{\mathbf{\Gamma}} \in \mathbb{R}^{13}$, such that $\mathbf{\Gamma}\colon \mathbb{R}^{14} \rightarrow \mathbb{R}^3$

$$\mathbf{\Gamma}(t; \mathbf{w}_{\mathbf{\Gamma}}) = x(t; \mathbf{w}_x)\,\hat{\mathbf{i}} + y(t; \mathbf{w}_y)\,\hat{\mathbf{j}} + z(t; \mathbf{w}_z)\,\hat{\mathbf{k}} \tag{6.1}$$

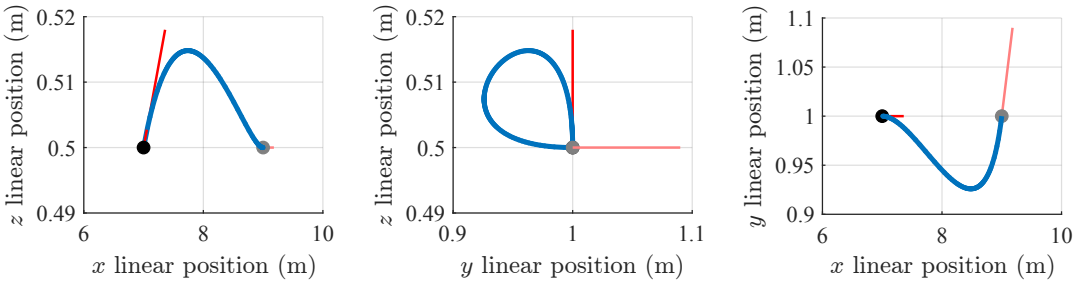
where $x(\cdot), y(\cdot)$ and $z(\cdot)$ are coordinate functions of time t, which are third-order polynomials of Hermite form parameterized by $\mathbf{w}_x, \mathbf{w}_y$ and $\mathbf{w}_z \in \mathbb{R}^5$, respectively. $\left(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\right)$ are the normalized unit vectors in the direction of the x, y and z axes of a three-dimensional, Cartesian coordinate system. The parameters $\mathbf{w}_{\mathbf{\Gamma}} \in \mathbb{R}^{13}$ are a collection of the time duration ΔT, and the values $(\mathbf{\Gamma}(t_0), \mathbf{\Gamma}(t_\mathrm{f}))$ and derivatives $\left(\dot{\mathbf{\Gamma}}(t_0), \dot{\mathbf{\Gamma}}(t_\mathrm{f})\right)$ at the start (0) and end (f) points of $\mathbf{\Gamma}$, respectively:

$$\mathbf{w}_{\mathbf{\Gamma}} = (\Delta T, \underbrace{x_0, y_0, z_0}_{\mathbf{\Gamma}(t_0)}, \underbrace{x_\mathrm{f}, y_\mathrm{f}, z_\mathrm{f}}_{\mathbf{\Gamma}(t_\mathrm{f})}, \underbrace{\dot{x}_0, \dot{y}_0, \dot{z}_0}_{\dot{\mathbf{\Gamma}}(t_0)}, \underbrace{\dot{x}_\mathrm{f}, \dot{y}_\mathrm{f}, \dot{z}_\mathrm{f}}_{\dot{\mathbf{\Gamma}}(t_\mathrm{f})}). \tag{6.2}$$

A 3D illustration of such a parameterized polynomial is shown in Fig. 6.1.



(a) 3D view



(b) zx-axes view **(c)** zy-axes view **(d)** yx-axes view

Figure 6.1: An example of a function of third-order polynomial of Hermite form $\mathbf{\Gamma}(t; \mathbf{w}_{\mathbf{\Gamma}})$ with $\mathbf{w}_{\mathbf{\Gamma}} = (5,\ 7, 1, 0.5,\ 9, 1, 0.5,\ 0.4, 0, 0.02,\ 0.2, 0.1, 0)$.

6.3.2.1 Partial derivatives

The Jacobian matrix of Eq. 6.1 is

$$\mathbf{J}_{\mathbf{w}_{\mathbf{\Gamma}}} = \frac{\partial \mathbf{\Gamma}}{\partial \mathbf{w}_{\mathbf{\Gamma}}} = \frac{\partial (x, y, z)}{\partial (\Delta T, x_0, y_0, z_0, x_\mathrm{f}, y_\mathrm{f}, z_\mathrm{f}, \dot{x}_0, \dot{y}_0, \dot{z}_0, \dot{x}_\mathrm{f}, \dot{y}_\mathrm{f}, \dot{z}_\mathrm{f})} \quad \in \mathbb{R}^{3\times 13}. \tag{6.3}$$

To understand how the parameters of the polynomial change the output of some function $\mathbf{f}_t$, dependent on the polynomial $\mathbf{\Gamma}_t$, evaluated at some time t, we can apply the chain rule as follows

$$\frac{\partial \mathbf{f}_t}{\partial \mathbf{w}_{\mathbf{\Gamma}}} = \frac{\partial \mathbf{f}_t}{\partial \mathbf{\Gamma}_t} \frac{\partial \mathbf{\Gamma}_t}{\partial \mathbf{w}_{\mathbf{\Gamma}}} \quad \in \mathbb{R}^{3\times 13}. \tag{6.4}$$

6.3.3 Lump leg single rigid body dynamics

In this section, we will derive the LL-SRBD model and its partial derivatives with respect to the optimization variables of our problem formulation presented in Sec. 6.3.2.

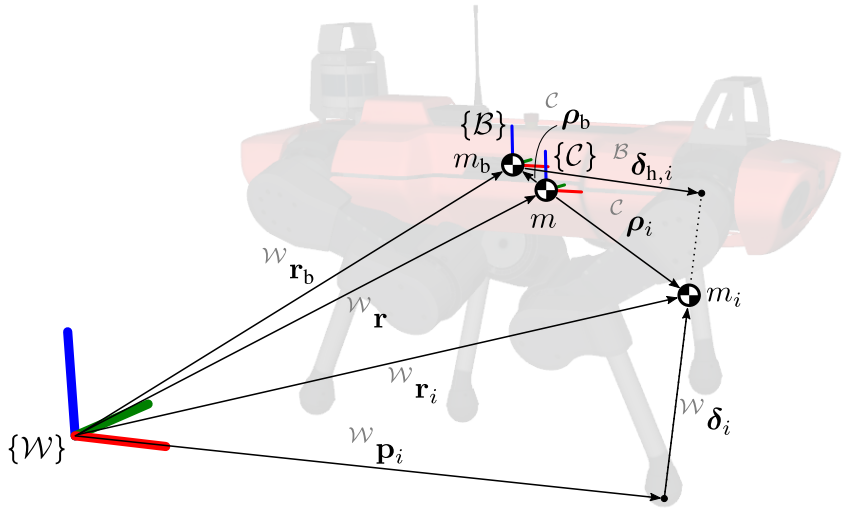


Figure 6.2: The key components of the lump leg single-rigid-body dynamics (LL-SRBD) model. The model distinguishes between the world $\mathcal{W}$, base $\mathcal{B}$ and CoM $\mathcal{C}$ reference frames. $\mathcal{B}$ represents the geometric centre (centroid) of the base which is the location of the mass of the base link m_b (approximated as a point mass). The location ${}^{\mathcal{W}}\mathbf{r}_i$ of the mass m_i of the ith leg (approximated as a point mass) is assumed to be between its end effector and its hip (h) joint, offset by ${}^{\mathcal{W}}\boldsymbol{\delta}_i$ relative to ${}^{\mathcal{W}}\mathbf{p}_i$. These components can be used to determine the linear position ${}^{\mathcal{W}}\mathbf{r}$ of the CoM frame $\mathcal{C}$ and the mass moment of inertia of the whole robot which depends on ${}^{\mathcal{C}}\boldsymbol{\rho}_\mathrm{b}$ and ${}^{\mathcal{C}}\boldsymbol{\rho}_i$, the relative positions of the base and leg masses, respectively.

The linear (translational) and angular (rotational) governing equations of the LL-SRBD model, shown in Fig. 6.2, are:

$$m\,{}^{\mathcal{W}}\ddot{\mathbf{r}} = m\,\mathbf{g} + \sum_{i=1}^{n_\mathrm{ee}} {}^{\mathcal{W}}\mathbf{f}_i, \tag{6.5a}$$

$${}^{\mathcal{W}}\mathbf{I}\,{}^{\mathcal{W}}\dot{\boldsymbol{\omega}} + {}^{\mathcal{W}}\boldsymbol{\omega} \times {}^{\mathcal{W}}\mathbf{I}\,{}^{\mathcal{W}}\boldsymbol{\omega} = \sum_{i=1}^{n_\mathrm{ee}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right)\right). \tag{6.5b}$$

They appear equivalent to the SRBD; however, the model distinguishes between the mass of the base m_b and the mass m_i of ith leg (approximated as a point mass) which make up the total mass $m = m_\mathrm{b} + \sum_{i=1}^{n_\mathrm{ee}} m_i$ and accounts for the inertia of the ith leg by assuming its mass m_i lies between its end effector ${}^{\mathcal{W}}\mathbf{p}_i \in \mathbb{R}^3$ and its hip which is known to be offset by ${}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},i}$ relative to $\mathcal{B}$. Therefore, the centre of mass of each leg is ${}^{\mathcal{W}}\mathbf{r}_i = {}^{\mathcal{W}}\mathbf{p}_i + {}^{\mathcal{W}}\boldsymbol{\delta}_i \in \mathbb{R}^3$. The offset ${}^{\mathcal{W}}\boldsymbol{\delta}_i$, and its time derivatives, can be calculated using the rotation matrix ${}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}(\boldsymbol{\theta}) = {}^{\mathcal{W}}\mathbf{R}_{\mathcal{C}}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the

orientation (parameterized by Euler angles) of the robot, as follows

$$
\begin{aligned}
{}^{\mathcal{W}}\boldsymbol{\delta}_i\left({}^{\mathcal{W}}\mathbf{r}_\mathrm{b}, \boldsymbol{\theta}, {}^{\mathcal{W}}\mathbf{p}_i\right) &= k\left({}^{\mathcal{W}}\mathbf{r}_\mathrm{b} + {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\,{}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},i} - {}^{\mathcal{W}}\mathbf{p}_i\right), && \text{(6.6a)}\\
{}^{\mathcal{W}}\dot{\boldsymbol{\delta}}_i\left({}^{\mathcal{W}}\dot{\mathbf{r}}_\mathrm{b}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, {}^{\mathcal{W}}\dot{\mathbf{p}}_i\right) &= k\left({}^{\mathcal{W}}\dot{\mathbf{r}}_\mathrm{b} + \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\,{}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},i} - {}^{\mathcal{W}}\dot{\mathbf{p}}_i\right), && \text{(6.6b)}\\
{}^{\mathcal{W}}\ddot{\boldsymbol{\delta}}_i\left({}^{\mathcal{W}}\ddot{\mathbf{r}}_\mathrm{b}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}, {}^{\mathcal{W}}\ddot{\mathbf{p}}_i\right) &= k\left({}^{\mathcal{W}}\ddot{\mathbf{r}}_\mathrm{b} + \left(\left[{}^{\mathcal{W}}\dot{\boldsymbol{\omega}}\right]_\times + \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times^2\right){}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\,{}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},i} - {}^{\mathcal{W}}\ddot{\mathbf{p}}_i\right), && \text{(6.6c)}
\end{aligned}
$$

where $k \in \mathbb{R}$ is a scaling factor which determines the length of the ${}^{\mathcal{W}}\boldsymbol{\delta}_i$ vector[3].

Remark 6.3.1. Importantly, one should observe that in the case of a symmetric body (as is the case for a quadrupedal robot), $\sum_{i=1}^{n_\mathrm{ee}} {}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},i} = 0$; this eliminates the middle term in Eq. 6.6 and therefore the dependence on $\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right)$.

The linear position and acceleration of the CoM of the whole robot are

$$
\begin{aligned}
{}^{\mathcal{W}}\mathbf{r}(\mathbf{V}_\mathbf{r}; \boldsymbol{\Theta}) &= \frac{{}^{\mathcal{W}}\mathbf{r}_\mathrm{b} m_\mathrm{b} + \sum_{i=1}^{n_\mathrm{ee}} {}^{\mathcal{W}}\mathbf{r}_i m_i}{m_\mathrm{b} + \sum_{i=1}^{n_\mathrm{ee}} m_i}\\
&= \frac{1}{m}\left[{}^{\mathcal{W}}\mathbf{r}_\mathrm{b} m_\mathrm{b} + \sum_{i=1}^{n_\mathrm{ee}}\left[\left({}^{\mathcal{W}}\mathbf{p}_i + {}^{\mathcal{W}}\boldsymbol{\delta}_i\left(\cdot\right)\right) m_i\right]\right], && \text{(6.7a)}\\
{}^{\mathcal{W}}\ddot{\mathbf{r}}\left(\mathbf{V}_{\ddot{\mathbf{r}}}; \boldsymbol{\Theta}\right) &= \frac{1}{m}\left[{}^{\mathcal{W}}\ddot{\mathbf{r}}_\mathrm{b} m_\mathrm{b} + \sum_{i=1}^{n_\mathrm{ee}}\left[\left({}^{\mathcal{W}}\ddot{\mathbf{p}}_i + {}^{\mathcal{W}}\ddot{\boldsymbol{\delta}}_i\left(\cdot\right)\right) m_i\right]\right], && \text{(6.7b)}
\end{aligned}
$$

where:

- $\mathbf{V}_\mathbf{r} = \left({}^{\mathcal{W}}\mathbf{r}_\mathrm{b}, {}^{\mathcal{W}}\mathbf{p}_1, \ldots, {}^{\mathcal{W}}\mathbf{p}_{n_\mathrm{ee}}, \boldsymbol{\theta}\right)$ are the variables of ${}^{\mathcal{W}}\mathbf{r}$,
- $\mathbf{V}_{\ddot{\mathbf{r}}} = \left({}^{\mathcal{W}}\ddot{\mathbf{r}}_\mathrm{b}, {}^{\mathcal{W}}\ddot{\mathbf{p}}_1, \ldots, {}^{\mathcal{W}}\ddot{\mathbf{p}}_{n_\mathrm{ee}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right)$ are the variables of ${}^{\mathcal{W}}\ddot{\mathbf{r}}$, and
- $\boldsymbol{\Theta} = \left({}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},1}, \ldots, {}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h},n_\mathrm{ee}}, m_\mathrm{b}, m_1, \ldots, m_{n_\mathrm{ee}}\right)$ are the parameters.

The moment of inertia of the whole robot ${}^{\mathcal{W}}\mathbf{I}(\cdot) \in \mathbb{R}^{3\times3}$ in the world frame $\mathcal{W}$ is computed by transforming ${}^{\mathcal{B}}\mathbf{I}(\cdot) \in \mathbb{R}^{3\times3}$ in the body frame $\mathcal{B}$ using the rotation matrix ${}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}} \in \mathbb{R}^{3\times3}$

$$
{}^{\mathcal{W}}\mathbf{I}(\mathbf{V}_\mathbf{r}; \boldsymbol{\Theta}) = {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\,{}^{\mathcal{B}}\mathbf{I}\,{}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}^\intercal \qquad \text{(6.8)}
$$

[3]By varying k, m_i can be placed anywhere between the foothold ($k = 0$) and the hip ($k = 1$); we used a value of $k = 1/2$.

with

$$^{\mathcal{B}}\mathbf{I}(\mathbf{V_r};\boldsymbol{\Theta}) = {}^{\mathcal{B}}\mathbf{I}_{\mathrm{b}} + \sum_{i=1}^{n_{\mathrm{ee}}} {}^{\mathcal{B}}\mathbf{I}_i. \tag{6.9}$$

$^{\mathcal{B}}\mathbf{I}_{\mathrm{b}} \in \mathbb{R}^{3\times3}$ is the moment of inertia of the robot's base (assumed to be a *distributed* mass) in the body frame, which can be computed using the parallel axes theorem as follows,

$$^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}(\mathbf{V_r};\boldsymbol{\Theta}) = {}^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}^{\mathrm{CoM}} - m_{\mathrm{b}} \left[{}^{\mathcal{C}}\boldsymbol{\rho}_{\mathrm{b}}\right]_{\times}^{2}, \tag{6.10}$$

and $^{\mathcal{B}}\mathbf{I}_i \in \mathbb{R}^{3\times3}$ is the moment of inertia of the ith leg (assumed to be a *point* mass) in the body frame

$$^{\mathcal{B}}\mathbf{I}_i(\mathbf{V_r};\boldsymbol{\Theta}) = -m_i \left[{}^{\mathcal{C}}\boldsymbol{\rho}_i\right]_{\times}^{2} \tag{6.11}$$

In Eq. 6.10 and Eq. 6.11 above, $^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}^{\mathrm{CoM}}$ is the inertia of the base calculated at its CoM, $^{\mathcal{C}}\boldsymbol{\rho}_{\mathrm{b}} \in \mathbb{R}^3$ is the vector between the CoM of the robot and the centre of the base

$$^{\mathcal{C}}\boldsymbol{\rho}_{\mathrm{b}}(\mathbf{V_r};\boldsymbol{\Theta}) = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left(-{}^{\mathcal{W}}\mathbf{r} + {}^{\mathcal{W}}\mathbf{r}_{\mathrm{b}}\right), \tag{6.12}$$

and $^{\mathcal{C}}\boldsymbol{\rho}_i \in \mathbb{R}^3$ represents the vector between the CoM of the robot and the CoM of the ith leg

$$^{\mathcal{C}}\boldsymbol{\rho}_i(\mathbf{V_r};\boldsymbol{\Theta}) = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left(-{}^{\mathcal{W}}\mathbf{r} + {}^{\mathcal{W}}\mathbf{p}_i + {}^{\mathcal{W}}\boldsymbol{\delta}_i\right). \tag{6.13}$$

6.3.3.1 Equality constraint

The linear (translational) and angular (rotational) dynamics equality constraints are

$$\mathbf{f}_{\text{dyn-lin}}\left(\mathbf{V}_{\ddot{\mathbf{r}}}, {}^{\mathcal{W}}\mathbf{f}\right) := m\,{}^{\mathcal{W}}\ddot{\mathbf{r}} - m\,\mathbf{g} - \sum_{i=1}^{n_{\mathrm{ee}}} {}^{\mathcal{W}}\mathbf{f}_i = \mathbf{0}, \tag{6.14a}$$

$$\mathbf{f}_{\text{dyn-ang}}\left(\mathbf{V_r}, {}^{\mathcal{W}}\mathbf{f}\right) := {}^{\mathcal{W}}\mathbf{I}\,{}^{\mathcal{W}}\dot{\boldsymbol{\omega}} + {}^{\mathcal{W}}\boldsymbol{\omega} \times {}^{\mathcal{W}}\mathbf{I}\,{}^{\mathcal{W}}\boldsymbol{\omega} - \sum_{i=1}^{n_{\mathrm{ee}}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right)\right) = \mathbf{0}. \tag{6.14b}$$

6.3.3.2 Partial derivatives

The partial derivatives of Eq. 6.14a are

$$\frac{\partial \mathbf{f}_{\text{dyn-lin}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{r}}_{\text{b}}} = \left(m_{\text{b}} + k\sum_{i=1}^{n_{\text{ee}}} m_i\right)\mathbb{1}, \tag{6.15a}$$

$$\frac{\partial \mathbf{f}_{\text{dyn-lin}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{p}}_i} = (1-k)m_i\,\mathbb{1}, \tag{6.15b}$$

$$\frac{\partial \mathbf{f}_{\text{dyn-lin}}}{\partial \left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right)} = \mathbb{0}, \qquad \text{(follows from remark 6.3.1)} \tag{6.15c}$$

$$\frac{\partial \mathbf{f}_{\text{dyn-lin}}}{\partial {}^{\mathcal{W}}\mathbf{f}_i} = -\mathbb{1}. \tag{6.15d}$$

The partial derivatives of Eq. 6.10 are:

$$\begin{aligned}\frac{\partial {}^{\mathcal{B}}\mathbf{I}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} &= m_{\text{b}}\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} - \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}{}^{\intercal} - {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \otimes \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}}^{\intercal}\right] \\ &= m_{\text{b}}\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} - 2\left(1 - \frac{m_{\text{b}}}{m}\right){}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}{}^{\intercal}\right],\end{aligned} \tag{6.16a}$$

$$\begin{aligned}\frac{\partial {}^{\mathcal{B}}\mathbf{I}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} &= m_{\text{b}}\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} - \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}{}^{\intercal} - {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \otimes \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}}{\partial {}^{\mathcal{W}}\mathbf{p}_i}^{\intercal}\right] \\ &= m_{\text{b}}\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}} \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} - \frac{2m_i}{m}(1-k){}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_{\text{b}}{}^{\intercal}\right],\end{aligned} \tag{6.16b}$$

$$\frac{\partial {}^{\mathcal{B}}\mathbf{I}_{\text{b}}}{\partial \boldsymbol{\theta}} = \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \,\bar{\times}_2\, \left(-{}^{\mathcal{W}}\mathbf{r} + {}^{\mathcal{W}}\mathbf{r}_{\text{b}}\right), \tag{6.16c}$$

The partial derivatives of Eq. 6.11 are:

$$\begin{aligned}\frac{\partial {}^{\mathcal{B}}\mathbf{I}_i}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} &= m_i\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_i\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} - \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_i{}^{\intercal} - {}^{\mathcal{C}}\boldsymbol{\rho}_i \otimes \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}}^{\intercal}\right] \\ &= m_i\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_i\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{r}_{\text{b}}} - \frac{2m_{\text{b}}}{m}(k-1){}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_i{}^{\intercal}\right],\end{aligned} \tag{6.17a}$$

$$\begin{aligned}\frac{\partial {}^{\mathcal{B}}\mathbf{I}_i}{\partial {}^{\mathcal{W}}\mathbf{p}_i} &= m_i\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_i\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} - \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i}{\partial {}^{\mathcal{W}}\mathbf{p}_i} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_i{}^{\intercal} - {}^{\mathcal{C}}\boldsymbol{\rho}_i \otimes \frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i}{\partial {}^{\mathcal{W}}\mathbf{p}_i}^{\intercal}\right] \\ &= m_i\left[\frac{\partial {}^{\mathcal{C}}\boldsymbol{\rho}_i \cdot {}^{\mathcal{C}}\boldsymbol{\rho}_i\mathbb{1}}{\partial {}^{\mathcal{W}}\mathbf{p}_i} - 2\left(1 - \frac{m_i}{m}\right)(1-k){}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \otimes {}^{\mathcal{C}}\boldsymbol{\rho}_i{}^{\intercal}\right],\end{aligned} \tag{6.17b}$$

$$\frac{\partial {}^{\mathcal{B}}\mathbf{I}_i}{\partial \boldsymbol{\theta}} = \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \,\bar{\times}_2\, \left(-{}^{\mathcal{W}}\mathbf{r} + {}^{\mathcal{W}}\mathbf{p}_i + {}^{\mathcal{W}}\boldsymbol{\delta}_i\right). \tag{6.17c}$$

The first terms in Eq. 6.16a, 6.16b, 6.17a, and 6.17b can be computed element-wise as follows

$$\left[\frac{\partial \boldsymbol{\rho} \cdot \boldsymbol{\rho}\mathbb{1}}{\partial \boldsymbol{a}}\right]_{ijl} = 2\boldsymbol{\rho} \cdot \frac{\partial \boldsymbol{\rho}}{\partial a_l} \cdot \delta_{ij}.$$

The partial derivatives of Eq. 6.9 are:

$$\frac{\partial^{\mathcal{B}}\mathbf{I}}{\partial^{\mathcal{W}}\mathbf{r}_{\mathrm{b}}} = \frac{\partial^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}}{\partial^{\mathcal{W}}\mathbf{r}_{\mathrm{b}}} + \sum_{i=1}^{n_{\mathrm{ee}}} \frac{\partial^{\mathcal{B}}\mathbf{I}_i}{\partial^{\mathcal{W}}\mathbf{r}_{\mathrm{b}}}, \tag{6.18a}$$

$$\frac{\partial^{\mathcal{B}}\mathbf{I}}{\partial^{\mathcal{W}}\mathbf{p}_i} = \frac{\partial^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}}{\partial^{\mathcal{W}}\mathbf{p}_i} + \sum_{i=1}^{n_{\mathrm{ee}}} \frac{\partial^{\mathcal{B}}\mathbf{I}_i}{\partial^{\mathcal{W}}\mathbf{p}_i}, \tag{6.18b}$$

$$\frac{\partial^{\mathcal{B}}\mathbf{I}}{\partial\boldsymbol{\theta}} = \frac{\partial^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}}{\partial\boldsymbol{\theta}} + \sum_{i=1}^{n_{\mathrm{ee}}} \frac{\partial^{\mathcal{B}}\mathbf{I}_i}{\partial\boldsymbol{\theta}}, \tag{6.18c}$$

The product of the partial derivative of Eq. 6.8 (with respect to $\mathbf{a} \in \mathbf{V}_{\mathbf{r}}$), with any vector $\mathbf{b} \in \mathbb{R}^3$ (which is assumed to be constant), results in the following matrix taking values in $\mathbb{R}^{3\times3}$:

$$\begin{aligned}\frac{\partial^{\mathcal{W}}\mathbf{I}}{\partial\mathbf{a}} \bar{\times}_2 \mathbf{b} = &\frac{\partial^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}}{\partial\mathbf{a}} \bar{\times}_2 \left({}^{\mathcal{B}}\mathbf{I}\, {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\, \mathbf{b}\right) \\ &+ {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}} \left[\frac{\partial^{\mathcal{B}}\mathbf{I}}{\partial\mathbf{a}} \bar{\times}_2 \left({}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}^{\intercal}\, \mathbf{b}\right)\right] \\ &+ {}^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}\, {}^{\mathcal{B}}\mathbf{I} \left(\frac{\partial^{\mathcal{W}}\mathbf{R}_{\mathcal{B}}}{\partial\mathbf{a}}^{\intercal} \bar{\times}_2 \mathbf{b}\right).\end{aligned} \tag{6.19a}$$

Therefore, the partial derivatives of Eq. 6.14b are:

$$\begin{aligned}\frac{\partial \mathbf{f}_{\text{dyn-ang}}}{\partial\mathbf{a}} = &\frac{\partial^{\mathcal{W}}\mathbf{I}}{\partial\mathbf{a}} \bar{\times}_2 {}^{\mathcal{W}}\dot{\boldsymbol{\omega}} + {}^{\mathcal{W}}\mathbf{I}\frac{\partial^{\mathcal{W}}\dot{\boldsymbol{\omega}}}{\partial\mathbf{a}} \\ &+ \frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\mathbf{a}} \times {}^{\mathcal{W}}\mathbf{I}\, {}^{\mathcal{W}}\boldsymbol{\omega} + {}^{\mathcal{W}}\boldsymbol{\omega} \times \left(\frac{\partial^{\mathcal{W}}\mathbf{I}}{\partial\mathbf{a}} \bar{\times}_2 {}^{\mathcal{W}}\boldsymbol{\omega}\right) + {}^{\mathcal{W}}\boldsymbol{\omega} \times {}^{\mathcal{W}}\mathbf{I}\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\mathbf{a}} \\ &- \sum_{i=1}^{n_{\mathrm{ee}}} \left[{}^{\mathcal{W}}\mathbf{f}_i \times \left(\frac{\partial^{\mathcal{W}}\mathbf{r}}{\partial\mathbf{a}} - \frac{\partial^{\mathcal{W}}\mathbf{p}_i}{\partial\mathbf{a}}\right)\right],\end{aligned} \tag{6.20a}$$

$$\frac{\partial \mathbf{f}_{\text{dyn-ang}}}{\partial^{\mathcal{W}}\mathbf{f}_i} = -\left[{}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right]_{\times}. \tag{6.20b}$$

6.3.4 Centroidal dynamics in Euclidean space

Recalling Eq. 2.7, the centroidal dynamics describe the time derivatives of the linear momentum $\mathbf{l}$ and angular momentum $\mathbf{k}$ as

$$\dot{\mathbf{h}}(\mathbf{q}, \mathbf{v}, \dot{\mathbf{v}}) = \begin{bmatrix}\dot{\mathbf{l}} \\ \dot{\mathbf{k}}\end{bmatrix} = \begin{bmatrix} m\,\mathbf{g} + \sum_{i=1}^{n_{\mathrm{ee}}} {}^{\mathcal{W}}\mathbf{f}_i \\ \sum_{i=1}^{n_{\mathrm{ee}}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right)\right)\end{bmatrix} \tag{6.21}$$

where

- $\mathbf{q} = \left[{}^{\mathcal{W}}\mathbf{r}^{\intercal}\ Q_w\ Q_x\ Q_y\ Q_z\ \mathbf{q}_{\mathrm{j},i}{}^{\intercal}\ \ldots\ \mathbf{q}_{\mathrm{j},n_{\mathrm{ee}}}{}^{\intercal}\right]^{\intercal} \in \mathbb{R}^{(3+4+n_{\mathrm{j}})}$,

- $\mathbf{v} = \begin{bmatrix} {}^{\mathcal{B}}\dot{\mathbf{r}}^{\intercal} & {}^{\mathcal{B}}\boldsymbol{\omega}^{\intercal} & \mathbf{v}_{\mathrm{j},i}{}^{\intercal} & \dots & \mathbf{v}_{\mathrm{j},n_{\mathrm{ee}}}{}^{\intercal} \end{bmatrix}^{\intercal} \in \mathbb{R}^{(3+3+n_{\mathrm{j}})}$, and
- $\dot{\mathbf{v}} = \begin{bmatrix} {}^{\mathcal{B}}\ddot{\mathbf{r}}^{\intercal} & {}^{\mathcal{B}}\dot{\boldsymbol{\omega}}^{\intercal} & \dot{\mathbf{v}}_{\mathrm{j},i}^{\intercal} & \dots & \dot{\mathbf{v}}_{\mathrm{j},n_{\mathrm{ee}}}^{\intercal} \end{bmatrix}^{\intercal} \in \mathbb{R}^{(3+3+n_{\mathrm{j}})}$

are column vectors of generalized coordinates, velocities and accelerations, respectively. At a time t, the elements of these vectors can be computed from the following Euclidean-space variables:

$$\mathbf{V}(t; \mathbf{w}_{\mathbf{V}}) = \left({}^{\mathcal{W}}\mathbf{r}, {}^{\mathcal{W}}\dot{\mathbf{r}}, {}^{\mathcal{W}}\ddot{\mathbf{r}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}, {}^{\mathcal{W}}\mathbf{p}_i, \dots, {}^{\mathcal{W}}\mathbf{p}_{n_{\mathrm{ee}}}, {}^{\mathcal{W}}\dot{\mathbf{p}}_i, \dots, {}^{\mathcal{W}}\dot{\mathbf{p}}_{n_{\mathrm{ee}}}, {}^{\mathcal{W}}\ddot{\mathbf{p}}_i, \dots, {}^{\mathcal{W}}\ddot{\mathbf{p}}_{n_{\mathrm{ee}}} \right)$$

that each belong to a *single*[4] polyonomial. This collection of single polynomials is parameterized, using Eq. 3.1, by the following optimization variables:

$$\begin{aligned} \mathbf{w}_{\mathbf{V}} = \big(& {}^{\mathcal{W}}\mathbf{r}_0, {}^{\mathcal{W}}\mathbf{r}_{\mathrm{f}}, {}^{\mathcal{W}}\dot{\mathbf{r}}_0, {}^{\mathcal{W}}\dot{\mathbf{r}}_{\mathrm{f}}, \boldsymbol{\theta}_0, \boldsymbol{\theta}_{\mathrm{f}}, \dot{\boldsymbol{\theta}}_0, \dot{\boldsymbol{\theta}}_{\mathrm{f}}, \\ & {}^{\mathcal{W}}\mathbf{p}_{1,0}, {}^{\mathcal{W}}\mathbf{p}_{1,\mathrm{f}}, \dots, {}^{\mathcal{W}}\mathbf{p}_{n_{\mathrm{ee}},0}, {}^{\mathcal{W}}\mathbf{p}_{n_{\mathrm{ee}},\mathrm{f}}, {}^{\mathcal{W}}\dot{\mathbf{p}}_{1,0}, {}^{\mathcal{W}}\dot{\mathbf{p}}_{1,\mathrm{f}}, \dots, {}^{\mathcal{W}}\dot{\mathbf{p}}_{n_{\mathrm{ee}},0}, {}^{\mathcal{W}}\dot{\mathbf{p}}_{n_{\mathrm{ee}},\mathrm{f}} \big). \end{aligned}$$

Note that $Q = w + x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}} \in \mathbb{H}$ is a quaternion and $\boldsymbol{\theta} = (\phi \ \theta \ \psi)^{\intercal} \in \mathbb{R}^3$ is a vector of Euler angles (roll, pitch, yaw) used in our formulation to parameterize the orientation and to construct the rotation matrix ${}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}(\boldsymbol{\theta})$. For visual clarity, in the remainder of this section, we omit the subscript i, e.g., ${}^{\mathcal{B}}\mathbf{p} = {}^{\mathcal{B}}\mathbf{p}_i \in \mathbb{R}^3$.

In this section, we will show the necessary conversions from world frame $\mathcal{W}$ to body (CoM) frame $\mathcal{B}$ and from Euclidean space to joint space.

The angular velocity and acceleration of the CoM in the world frame are

$${}^{\mathcal{W}}\boldsymbol{\omega}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) = \mathbf{C}(\boldsymbol{\theta})\ \dot{\boldsymbol{\theta}} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{6.22a}$$

$${}^{\mathcal{W}}\dot{\boldsymbol{\omega}}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right) = \dot{\mathbf{C}}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)\ \dot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta})\ \ddot{\boldsymbol{\theta}}, \tag{6.22b}$$

where the order of the application is $\{\psi\ \theta\ \phi\}$. The angular velocity and acceleration of the CoM in the body frame are

$${}^{\mathcal{B}}\boldsymbol{\omega}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\, {}^{\mathcal{W}}\boldsymbol{\omega}, \tag{6.23a}$$

$$\begin{aligned} {}^{\mathcal{B}}\dot{\boldsymbol{\omega}}\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\, {}^{\mathcal{W}}\dot{\boldsymbol{\omega}} + {}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{W}}\, {}^{\mathcal{W}}\boldsymbol{\omega} \\ &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\, {}^{\mathcal{W}}\dot{\boldsymbol{\omega}} - {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \underbrace{\left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_{\times}}_{=0}\, {}^{\mathcal{W}}\boldsymbol{\omega} \\ &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\, {}^{\mathcal{W}}\dot{\boldsymbol{\omega}}. \end{aligned} \tag{6.23b}$$

[4] Each trajectory spline, e.g., CoM linear position, consists of multiple connected polynomials.

The linear position, velocity and acceleration of an end effector in the body frame are:

$$
\begin{aligned}
{}^{\mathcal{B}}\mathbf{p}\left(\mathbf{V}_{\mathbf{p}}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right), && \text{(6.24a)}\\
{}^{\mathcal{B}}\dot{\mathbf{p}}\left(\mathbf{V}_{\dot{\mathbf{p}}}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[-{}^{\mathcal{W}}\boldsymbol{\omega}\times\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right) + \left({}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right)\right], && \text{(6.24b)}\\
{}^{\mathcal{B}}\ddot{\mathbf{p}}\left(\mathbf{V}_{\ddot{\mathbf{p}}}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\Bigg[\overbrace{{}^{\mathcal{W}}\boldsymbol{\omega}\times\left({}^{\mathcal{W}}\boldsymbol{\omega}\times\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)\right)}^{\text{centrifugal acceleration}} - \overbrace{2\,{}^{\mathcal{W}}\boldsymbol{\omega}\times\left({}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right)}^{\text{Coriolis acceleration}}\\
&\qquad - \underbrace{{}^{\mathcal{W}}\dot{\boldsymbol{\omega}}\times\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)}_{\text{Euler acceleration}} + \left({}^{\mathcal{W}}\ddot{\mathbf{p}} - {}^{\mathcal{W}}\ddot{\mathbf{r}}\right)\Bigg], && \text{(6.24c)}
\end{aligned}
$$

where:

- $\mathbf{V}_{\mathbf{p}} = \left({}^{\mathcal{W}}\mathbf{r}, {}^{\mathcal{W}}\mathbf{p}, \boldsymbol{\theta}\right)$ are the variables of ${}^{\mathcal{B}}\mathbf{p}$,
- $\mathbf{V}_{\dot{\mathbf{p}}} = \left({}^{\mathcal{W}}\mathbf{r}, {}^{\mathcal{W}}\dot{\mathbf{r}}, {}^{\mathcal{W}}\mathbf{p}, {}^{\mathcal{W}}\dot{\mathbf{p}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)$ are the variables of ${}^{\mathcal{B}}\dot{\mathbf{p}}$, and
- $\mathbf{V}_{\ddot{\mathbf{p}}} = \left({}^{\mathcal{W}}\mathbf{r}, {}^{\mathcal{W}}\dot{\mathbf{r}}, {}^{\mathcal{W}}\ddot{\mathbf{r}}, {}^{\mathcal{W}}\mathbf{p}, {}^{\mathcal{W}}\dot{\mathbf{p}}, {}^{\mathcal{W}}\ddot{\mathbf{p}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right)$ are the variables of ${}^{\mathcal{B}}\ddot{\mathbf{p}}$.

The linear velocity and acceleration of the CoM in the body frame are

$$
\begin{aligned}
{}^{\mathcal{B}}\dot{\mathbf{r}}\left({}^{\mathcal{W}}\dot{\mathbf{r}}, \boldsymbol{\theta}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\,{}^{\mathcal{W}}\dot{\mathbf{r}}, && \text{(6.25a)}\\
{}^{\mathcal{B}}\ddot{\mathbf{r}}\left({}^{\mathcal{W}}\dot{\mathbf{r}}, {}^{\mathcal{W}}\ddot{\mathbf{r}}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\,{}^{\mathcal{W}}\ddot{\mathbf{r}} + {}^{\mathcal{B}}\dot{\mathbf{R}}_{\mathcal{W}}\,{}^{\mathcal{W}}\dot{\mathbf{r}}\\
&= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\,{}^{\mathcal{W}}\ddot{\mathbf{r}} - {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_{\times}{}^{\mathcal{W}}\dot{\mathbf{r}}. && \text{(6.25b)}
\end{aligned}
$$

Let us introduce the Jacobian matrix ${}^{\mathcal{B}}\mathbf{J}(\mathbf{q}_{\mathrm{j}}) \in \mathbb{R}^{3\times 3}$ which depends on the joint angles $\mathbf{q}_{\mathrm{j}} = \{q_{\mathrm{HAA}}, q_{\mathrm{HFE}}, q_{\mathrm{KFE}}\} \in \mathbb{R}^3$ of the ith leg. It transforms a small change in $\mathbf{q}_{\mathrm{j}}$ into a small change in the linear position of the leg's end effector ${}^{\mathcal{B}}\mathbf{p} \in \mathbb{R}^3$ as follows

$$
\delta\,{}^{\mathcal{B}}\mathbf{p} = {}^{\mathcal{B}}\mathbf{J}(\mathbf{q}_{\mathrm{j}})\,\delta\mathbf{q}_{\mathrm{j}} = \begin{bmatrix} \frac{\partial p_x}{\partial q_{\mathrm{HAA}}} & \frac{\partial p_x}{\partial q_{\mathrm{HFE}}} & \frac{\partial p_x}{\partial q_{\mathrm{KFE}}} \\ \frac{\partial p_y}{\partial q_{\mathrm{HAA}}} & \frac{\partial p_y}{\partial q_{\mathrm{HFE}}} & \frac{\partial p_y}{\partial q_{\mathrm{KFE}}} \\ \frac{\partial p_z}{\partial q_{\mathrm{HAA}}} & \frac{\partial p_z}{\partial q_{\mathrm{HFE}}} & \frac{\partial p_z}{\partial q_{\mathrm{KFE}}} \end{bmatrix} \delta\mathbf{q}_{\mathrm{j}} \qquad \text{(6.26)}
$$

where each element of ${}^{\mathcal{B}}\mathbf{J}$ is a function of $\mathbf{q}_{\mathrm{j}}$. Note that, in our formulation, $\mathbf{q}_{\mathrm{j}} = \mathbf{q}_{\mathrm{j}}\left({}^{\mathcal{B}}\mathbf{p}\left({}^{\mathcal{W}}\mathbf{r}, \boldsymbol{\theta}, {}^{\mathcal{W}}\mathbf{p}\right)\right)$. Meanwhile, the time derivatives of ${}^{\mathcal{B}}\mathbf{p}$ are:

$$
{}^{\mathcal{B}}\dot{\mathbf{p}} = {}^{\mathcal{B}}\mathbf{J}\,\mathbf{v}_{\mathrm{j}}, \tag{6.27a}
$$

$$
{}^{\mathcal{B}}\ddot{\mathbf{p}} = {}^{\mathcal{B}}\dot{\mathbf{J}}\,\mathbf{v}_{\mathrm{j}} + {}^{\mathcal{B}}\mathbf{J}\,\dot{\mathbf{v}}_{\mathrm{j}}. \tag{6.27b}
$$

$\mathbf{J}^{-1}\left(\mathbf{q}_{\mathrm{j}}\right)$, can be used for inverse kinematics (IK); the joint positions (angles) can be obtained by rearranging Eq. 6.26 as follows

$$
\begin{aligned}
\mathbf{q}_{\mathrm{j}}\left(\mathbf{V}_{\mathbf{p}}\right) &= \mathbf{J}^{-1}\,{}^{\mathcal{B}}\mathbf{p} \\
&= \mathbf{J}^{-1}\,{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)
\end{aligned} \tag{6.28}
$$

The joint velocities can be obtained by rearranging Eq. 6.27a as follows

$$
\begin{aligned}
\mathbf{v}_{\mathrm{j}}\left(\mathbf{V}_{\dot{\mathbf{p}}}\right) &= \mathbf{J}^{-1}\,{}^{\mathcal{B}}\dot{\mathbf{p}} \\
&= \mathbf{J}^{-1}\,{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[-{}^{\mathcal{W}}\boldsymbol{\omega} \times \left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right) + \left({}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right)\right].
\end{aligned} \tag{6.29}
$$

The joint accelerations can be obtained by rearranging Eq. 6.27b as follows

$$
\begin{aligned}
\dot{\mathbf{v}}_{\mathrm{j}}\left(\mathbf{V}_{\ddot{\mathbf{p}}}\right) &= \mathbf{J}^{-1}\left({}^{\mathcal{B}}\ddot{\mathbf{p}} - {}^{\mathcal{B}}\dot{\mathbf{J}}\,\mathbf{v}_{\mathrm{j}}\right) \\
&= \mathbf{J}^{-1}\Bigg({}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\Big[{}^{\mathcal{W}}\boldsymbol{\omega} \times \left({}^{\mathcal{W}}\boldsymbol{\omega} \times \left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)\right) - 2\,{}^{\mathcal{W}}\boldsymbol{\omega} \times \left({}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right) \\
&\qquad - {}^{\mathcal{W}}\dot{\boldsymbol{\omega}} \times \left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right) + \left({}^{\mathcal{W}}\ddot{\mathbf{p}} - {}^{\mathcal{W}}\ddot{\mathbf{r}}\right)\Big] - \underbrace{{}^{\mathcal{B}}\dot{\mathbf{J}}\,\mathbf{v}_{\mathrm{j}}}_{\text{constant "drift"}}\Bigg)
\end{aligned} \tag{6.30}
$$

6.3.4.1 Equality constraint

The linear (translational) and angular (rotational) dynamics equality constraint is

$$
\mathbf{f}_{\mathrm{cd}}\left(\mathbf{w}\right) \coloneqq \dot{\mathbf{h}} - \dot{\mathbf{h}}_{\mathrm{ext}} = \dot{\mathbf{h}} - \begin{bmatrix} m\,\mathbf{g} - \sum_{i=1}^{n_{\mathrm{ee}}} {}^{\mathcal{W}}\mathbf{f}_i \\ \sum_{i=1}^{n_{\mathrm{ee}}} \left({}^{\mathcal{W}}\mathbf{f}_i \times \left({}^{\mathcal{W}}\mathbf{r} - {}^{\mathcal{W}}\mathbf{p}_i\right)\right) \end{bmatrix} = \mathbf{0} \tag{6.31}
$$

where $\mathbf{w} = \left(\mathbf{w}_{\mathbf{V}}, {}^{\mathcal{W}}\mathbf{f}_1, \ldots, {}^{\mathcal{W}}\mathbf{f}_{n_{\mathrm{ee}}}\right)$.

6.3.4.2 Partial derivatives

In this section, we derive by hand the analytical partial derivatives with respect to the Euclidean-space optimization variables. The total derivative of Eq. 6.31 is

$$
\begin{aligned}
\frac{\mathrm{d}\mathbf{f}_{\mathrm{cd}}}{\mathrm{d}\mathbf{w}} &= \frac{\partial \dot{\mathbf{h}}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{V}}\frac{\partial \mathbf{V}}{\partial \mathbf{w}} + \frac{\partial \dot{\mathbf{h}}}{\partial \mathbf{v}}\frac{\partial \mathbf{v}}{\partial \mathbf{V}}\frac{\partial \mathbf{V}}{\partial \mathbf{w}} + \frac{\partial \dot{\mathbf{h}}}{\partial \dot{\mathbf{v}}}\frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{V}}\frac{\partial \mathbf{V}}{\partial \mathbf{w}} \\
&\quad - \frac{\partial \dot{\mathbf{h}}_{\mathrm{ext}}}{\partial {}^{\mathcal{W}}\mathbf{r}}\frac{\partial {}^{\mathcal{W}}\mathbf{r}}{\partial \mathbf{w}} - \frac{\partial \dot{\mathbf{h}}_{\mathrm{ext}}}{\partial {}^{\mathcal{W}}\mathbf{p}}\frac{\partial {}^{\mathcal{W}}\mathbf{p}}{\partial \mathbf{w}} - \frac{\partial \dot{\mathbf{h}}_{\mathrm{ext}}}{\partial {}^{\mathcal{W}}\mathbf{f}}\frac{\partial {}^{\mathcal{W}}\mathbf{f}}{\partial \mathbf{w}},
\end{aligned} \tag{6.32}
$$

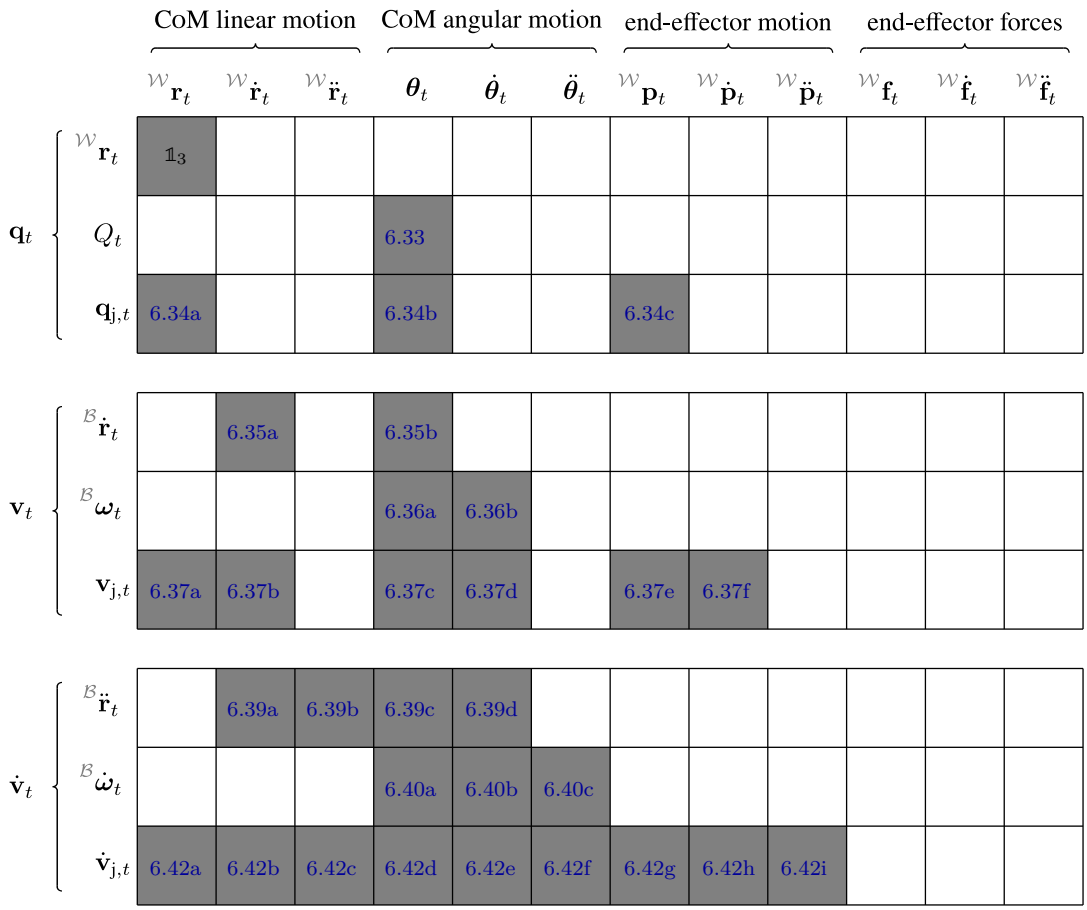


Figure 6.3: The sparsity pattern of the Jacobian matrix which illustrates the dependencies of the $\mathbf{q}$, $\mathbf{v}$ and $\dot{\mathbf{v}}$ vectors—defined in equations 6.28, 6.29, and 6.30, respectively—on the *indirectly* controlled variables CoM and end-effector variables $\mathbf{V}_t$, evaluated at time t. The end-effector forces do not affect the vectors.

where

- $\frac{\partial \dot{\mathbf{h}}}{\partial (\cdot)}$ can be obtained via the Pinocchio library (Carpentier et al. [151]),
- $\frac{\partial (\cdot)}{\partial \mathbf{V}}$ and $\frac{\partial \dot{\mathbf{h}}_{\text{ext}}}{\partial (\cdot)}$ will be derived in this section, and
- $\frac{\partial (\cdot)}{\partial \mathbf{w}}$ has been introduced in Eq. 6.3.

The sparsity pattern of the $\frac{\partial \mathbf{q}}{\partial \mathbf{V}}$, $\frac{\partial \mathbf{v}}{\partial \mathbf{V}}$ and $\frac{\partial \dot{\mathbf{v}}}{\partial \mathbf{V}}$ partial derivatives is shown in Fig 6.3. It should be noted that the Pinocchio library offers $\frac{\partial \dot{\mathbf{h}}\,(Q \oplus \Delta Q)}{\partial \Delta Q} \in \mathbb{R}^{6\times 3}$, i.e., the partial derivative of the rate of change of momenta $\dot{\mathbf{h}}\,(Q \oplus \Delta Q)$, where $\oplus$ is a tangent application operator, with respect the angular velocity ΔQ expressed

in the tangent space. Pinocchio also provides $\frac{\partial (Q \oplus \Delta Q)}{\partial \Delta Q} \in \mathbb{R}^{4\times 3}$ whose pseudo-inverse can be used to obtain the coefficient-wise derivative

$$\frac{\partial \dot{\mathbf{h}} (Q \oplus \Delta Q)}{\partial Q} = \frac{\partial \dot{\mathbf{h}} (Q \oplus \Delta Q)}{\partial \Delta Q} \frac{\partial \Delta Q}{\partial (Q \oplus \Delta Q)}.$$

Then, it can be further combined with the partial derivative of a quaternion with respect to Euler angles which is denoted as

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} \quad \in \mathbb{R}^{4\times 3}, \tag{6.33}$$

and is expanded fully in Eq. A.5.

The partial derivatives of Eq. 6.28 are:

$$\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{W}}\mathbf{r}} = \frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial {}^{\mathcal{W}}\mathbf{r}} = \mathbf{J}^{-1} \left(-{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \right), \tag{6.34a}$$

$$\frac{\partial \mathbf{q}_\mathrm{j}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial \boldsymbol{\theta}} = \mathbf{J}^{-1} \left(-{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \right), \tag{6.34b}$$

$$\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{W}}\mathbf{p}} = \frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial {}^{\mathcal{W}}\mathbf{p}} = \mathbf{J}^{-1} \left({}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \right). \tag{6.34c}$$

The partial derivatives of Eq. 6.25a are:

$$\frac{\partial {}^{\mathcal{B}}\dot{\mathbf{r}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}, \tag{6.35a}$$

$$\frac{\partial {}^{\mathcal{B}}\dot{\mathbf{r}}}{\partial \boldsymbol{\theta}} = \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \bar{\times}_2 {}^{\mathcal{W}}\dot{\mathbf{r}}. \tag{6.35b}$$

The partial derivatives of Eq. 6.23a are:

$$\frac{\partial {}^{\mathcal{B}}\boldsymbol{\omega}}{\partial \boldsymbol{\theta}} = \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \bar{\times}_2 {}^{\mathcal{W}}\boldsymbol{\omega} + {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \frac{\partial {}^{\mathcal{W}}\boldsymbol{\omega}}{\partial \boldsymbol{\theta}}, \tag{6.36a}$$

$$\frac{\partial {}^{\mathcal{B}}\boldsymbol{\omega}}{\partial \dot{\boldsymbol{\theta}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\, \mathbf{C}. \tag{6.36b}$$

The partial derivatives of Eq. 6.29 are

$$\begin{aligned}
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial {}^{\mathcal{W}}\mathbf{r}} &= \mathbf{J}^{-1}\, {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times \\
&\quad + \frac{\partial \mathbf{J}^{-1}}{\partial \mathbf{q}_\mathrm{j}} \times_3 \left(\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial {}^{\mathcal{W}}\mathbf{r}}\right)^\mathsf{T} \bar{\times}_2 {}^{\mathcal{B}}\mathbf{J}\, \mathbf{v}_\mathrm{j}, && \text{(6.37a)} \\
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}} &= -\mathbf{J}^{-1}\, {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}, && \text{(6.37b)} \\
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial \boldsymbol{\theta}} &= \mathbf{J}^{-1}\, {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right]_\times \frac{\partial {}^{\mathcal{W}}\boldsymbol{\omega}}{\partial \boldsymbol{\theta}} \\
&\quad + \mathbf{J}^{-1} \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \times_3 \left[\left({}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right) - {}^{\mathcal{W}}\boldsymbol{\omega} \times \left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)\right]^\mathsf{T} \\
&\quad + \frac{\partial \mathbf{J}^{-1}}{\partial \mathbf{q}_\mathrm{j}} \times_3 \left(\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial \boldsymbol{\theta}}\right)^\mathsf{T} \bar{\times}_2 {}^{\mathcal{B}}\mathbf{J}\, \mathbf{v}_\mathrm{j}, && \text{(6.37c)} \\
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial \dot{\boldsymbol{\theta}}} &= \mathbf{J}^{-1}\, {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right]_\times \mathbf{C}(\boldsymbol{\theta}), && \text{(6.37d)} \\
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial {}^{\mathcal{W}}\mathbf{p}} &= -\mathbf{J}^{-1}\, {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times \\
&\quad + \frac{\partial \mathbf{J}^{-1}}{\partial \mathbf{q}_\mathrm{j}} \times_3 \left(\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} \frac{\partial {}^{\mathcal{B}}\mathbf{p}}{\partial {}^{\mathcal{W}}\mathbf{p}}\right)^\mathsf{T} \bar{\times}_2 {}^{\mathcal{B}}\mathbf{J}\, \mathbf{v}_\mathrm{j}, && \text{(6.37e)} \\
\frac{\partial \mathbf{v}_\mathrm{j}}{\partial {}^{\mathcal{W}}\dot{\mathbf{p}}} &= -\frac{\partial \mathbf{v}_\mathrm{j}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}} && \text{(6.37f)}
\end{aligned}$$

The above equations can be further simplified by noting

$$\frac{\partial \mathbf{q}_\mathrm{j}}{\partial {}^{\mathcal{B}}\mathbf{p}} = \mathbf{J}^{-1}. \tag{6.38}$$

The partial derivatives of Eq. 6.25b are:

$$\begin{aligned}
\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{r}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}} &= -{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times, && \text{(6.39a)} \\
\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{r}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{r}}} &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}, && \text{(6.39b)} \\
\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{r}}}{\partial \boldsymbol{\theta}} &= \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \bar{\times}_2 {}^{\mathcal{W}}\ddot{\mathbf{r}} - \frac{\partial {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial \boldsymbol{\theta}} \bar{\times}_2 \left({}^{\mathcal{W}}\boldsymbol{\omega} \times {}^{\mathcal{W}}\dot{\mathbf{r}}\right) \\
&\quad - {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \left(\frac{\partial {}^{\mathcal{W}}\boldsymbol{\omega}}{\partial \boldsymbol{\theta}} \times {}^{\mathcal{W}}\dot{\mathbf{r}} + {}^{\mathcal{W}}\boldsymbol{\omega}\right), && \text{(6.39c)} \\
\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{r}}}{\partial \dot{\boldsymbol{\theta}}} &= {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}} \frac{\partial {}^{\mathcal{W}}\boldsymbol{\omega}}{\partial \dot{\boldsymbol{\theta}}}. && \text{(6.39d)}
\end{aligned}$$

The partials of Eq. 6.23b are:

$$\frac{\partial^{\mathcal{B}}\boldsymbol{\omega}}{\partial\boldsymbol{\theta}} = \frac{\partial^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial\boldsymbol{\theta}} \bar{\times}_2 {}^{\mathcal{W}}\boldsymbol{\omega} + {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\boldsymbol{\theta}}, \tag{6.40a}$$

$$\frac{\partial^{\mathcal{B}}\boldsymbol{\omega}}{\partial\dot{\boldsymbol{\theta}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[\frac{\partial\dot{\mathbf{C}}}{\partial\dot{\boldsymbol{\theta}}} \bar{\times}_2 \dot{\boldsymbol{\theta}} + \dot{\mathbf{C}}\right], \tag{6.40b}$$

$$\frac{\partial^{\mathcal{B}}\boldsymbol{\omega}}{\partial\ddot{\boldsymbol{\theta}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\mathbf{C}. \tag{6.40c}$$

Before taking partial derivatives of Eq. 6.30 as a whole, we will consider its terms. For the centrifugal acceleration term in Eq. 6.30, the following 'bac-cab' triple cross product identity can be used $\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = \mathbf{b}(\mathbf{a}^{\mathsf{T}}\mathbf{c}) - \mathbf{c}(\mathbf{a}^{\mathsf{T}}\mathbf{b})$. The partials of Eq. 6.24c are

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\mathbf{r}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left({}^{\mathcal{W}}\boldsymbol{\omega}^{\mathsf{T}}{}^{\mathcal{W}}\boldsymbol{\omega}\,\mathbb{1} - {}^{\mathcal{W}}\boldsymbol{\omega}\,{}^{\mathcal{W}}\boldsymbol{\omega}^{\mathsf{T}} + \left[{}^{\mathcal{W}}\dot{\boldsymbol{\omega}}\right]_\times\right), \tag{6.41a}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\dot{\mathbf{r}}} = 2\,{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[{}^{\mathcal{W}}\boldsymbol{\omega}\right]_\times, \tag{6.41b}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\ddot{\mathbf{r}}} = -{}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}, \tag{6.41c}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial\boldsymbol{\theta}} = \frac{\partial^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}}{\partial\boldsymbol{\theta}} \bar{\times}_2 \left({}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}^{\mathsf{T}}{}^{\mathcal{B}}\ddot{\mathbf{p}}\right) \tag{6.41d}$$

$$+ {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left\{\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\boldsymbol{\theta}}\left[{}^{\mathcal{W}}\boldsymbol{\omega}\cdot\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)\right] + 2\left[{}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right]_\times\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\boldsymbol{\theta}}\right.$$

$$\left. + \left[{}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right]_\times\frac{\partial^{\mathcal{W}}\dot{\boldsymbol{\omega}}}{\partial\boldsymbol{\theta}}\right\}, \tag{6.41e}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial\dot{\boldsymbol{\theta}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left\{\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\dot{\boldsymbol{\theta}}}\left[{}^{\mathcal{W}}\boldsymbol{\omega}\cdot\left({}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right)\right] + 2\left[{}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right]_\times\frac{\partial^{\mathcal{W}}\boldsymbol{\omega}}{\partial\dot{\boldsymbol{\theta}}}\right.$$

$$\left. + \left[{}^{\mathcal{W}}\mathbf{p} - {}^{\mathcal{W}}\mathbf{r}\right]_\times\frac{\partial^{\mathcal{W}}\dot{\boldsymbol{\omega}}}{\partial\dot{\boldsymbol{\theta}}}\right\}, \tag{6.41f}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial\ddot{\boldsymbol{\theta}}} = {}^{\mathcal{B}}\mathbf{R}_{\mathcal{W}}\left[{}^{\mathcal{W}}\dot{\mathbf{p}} - {}^{\mathcal{W}}\dot{\mathbf{r}}\right]_\times\mathbf{C}, \tag{6.41g}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\mathbf{p}} = -\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\mathbf{r}}, \tag{6.41h}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\dot{\mathbf{p}}} = -\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\dot{\mathbf{r}}}, \tag{6.41i}$$

$$\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\ddot{\mathbf{p}}} = -\frac{\partial^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial^{\mathcal{W}}\ddot{\mathbf{r}}}. \tag{6.41j}$$

Therefore, while ignoring the ${}^{\mathcal{B}}\dot{\mathbf{J}}\,\mathbf{v}_{\mathrm{j}}$ term, the partials of Eq. 6.30 are

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\mathbf{r}} = \frac{\partial \mathbf{J}^{-1}}{\partial {}^{\mathcal{W}}\mathbf{r}} \bar{\times}_2 {}^{\mathcal{B}}\ddot{\mathbf{p}} + \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\mathbf{r}}, \tag{6.42a}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{r}}}, \tag{6.42b}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{r}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{r}}}, \tag{6.42c}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\theta}} \bar{\times}_2 {}^{\mathcal{B}}\ddot{\mathbf{p}} + \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial \boldsymbol{\theta}}, \tag{6.42d}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial \dot{\boldsymbol{\theta}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial \dot{\boldsymbol{\theta}}}, \tag{6.42e}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial \ddot{\boldsymbol{\theta}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial \ddot{\boldsymbol{\theta}}}, \tag{6.42f}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\mathbf{p}} = \frac{\partial \mathbf{J}^{-1}}{\partial {}^{\mathcal{W}}\mathbf{p}} \bar{\times}_2 {}^{\mathcal{B}}\ddot{\mathbf{p}} + \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\mathbf{p}}, \tag{6.42g}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{p}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\dot{\mathbf{p}}}, \tag{6.42h}$$

$$\frac{\partial \dot{\mathbf{v}}_{\mathrm{j}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{p}}} = \mathbf{J}^{-1}\frac{\partial {}^{\mathcal{B}}\ddot{\mathbf{p}}}{\partial {}^{\mathcal{W}}\ddot{\mathbf{p}}}. \tag{6.42i}$$

6.4 Results

When computing the LL-SRBD, we approximated the hip abduction/adduction (HAA) and hip flexion/extension (HFE) joints (motors) as connected to the base; we used the masses and the moments of inertia of the base and the joints to calculate ${}^{\mathcal{B}}\mathbf{I}_{\mathrm{b}}$. We intuitively used the HAA frame position, e.g., ${}^{\mathcal{B}}\boldsymbol{\delta}_{\mathrm{h,LF}} = (0.36, 0.19, 0)$ (front left), the mass of the KFE joint ($m_i = 0.59\,\mathrm{kg}$), and the scaling $k = 1/2$ to calculate ${}^{\mathcal{B}}\mathbf{I}_i$.[5]

While we intended to implement the derived partial derivatives efficiently in code, we have tested the proposed models using only derivatives obtained using the finite difference (FD) method.

We compared the rotational inertia tensors obtained for both dynamics models for 5 different joint configurations (denoted as "C") in Table 6.1. The config-

[5]It is difficult to decide how to best approximate the rigid body, in a way that works as good as possible for an arbitrary configuration.

C.	Model	Rotational Inertia ${}^{\mathcal{B}}\mathbf{I}$ ($\mathrm{kg\,m^2}$)					
		I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
C0	SRBD	1.469	4.300	4.534	0.009	0.087	0
C1	LL-SRBD	1.636	4.705	4.984	0.009	0.089	0
	CD	1.729	4.405	4.881	0.009	0.087	0
C2	LL-SRBD	1.636	4.574	4.853	0.009	0.089	0
	CD	1.658	4.187	4.735	0.009	0.084	0
C3	LL-SRBD	1.546	4.705	4.893	0.009	0.089	0
	CD	1.321	4.317	4.348	0.010	0.091	0
C4	LL-SRBD	1.544	4.573	4.762	0.009	0.089	0
	CD	1.264	4.067	4.264	0.009	0.085	0
C5	LL-SRBD	1.504	4.625	4.867	−0.013	0.110	−0.005
	CD	1.202	4.054	4.314	0.013	0.241	0.047

Table 6.1: A comparison of the components of the rotational inertia of the SRBD, LL-SRBD and CD models for the 5 leg configurations (C.) shown in Fig. 6.4.

urations with superimposed equimomental ellipsoids are shown in Fig. 6.4. We also calculate the percentage deviation as shown in Table 6.2. In all the tests, the robot's height was ${}^{\mathcal{W}}r_z = 0.52\,\mathrm{m}$. The end-effector positions can be determined from the figure, e.g., ${}^{\mathcal{W}}\mathbf{p}_{\mathrm{LF}} = (0.3, 0.3, 0)$. All feet were on the ground—except for C5 when $\left({}^{\mathcal{W}}p_{\mathrm{LF},z}, {}^{\mathcal{W}}p_{\mathrm{RF},z}, {}^{\mathcal{W}}p_{\mathrm{LH},z}, {}^{\mathcal{W}}p_{\mathrm{RH},z}\right) = (0.2, 0, 0.05, 0.15)$. For reference, we use the SRBD model in its nominal configuration (C0) with a symmetric stance with, e.g., ${}^{\mathcal{W}}\mathbf{p}_{\mathrm{LF}} = (0.29, 0.19, 0)$.

As demonstrated, when the feet move away from the nominal locations C0, the components of the inertia matrix calculated using the CD model generally deviate more from their nominal values than those of the LL-SRBD model. Although the latter can shape the inertia, it tends to disproportionally change the wrong components.

We recognize that a metric based on the angular momentum (e.g., the L^2-norm or a divergence plot) could potentially be more appropriate and indicative of the superiority of the CD model.

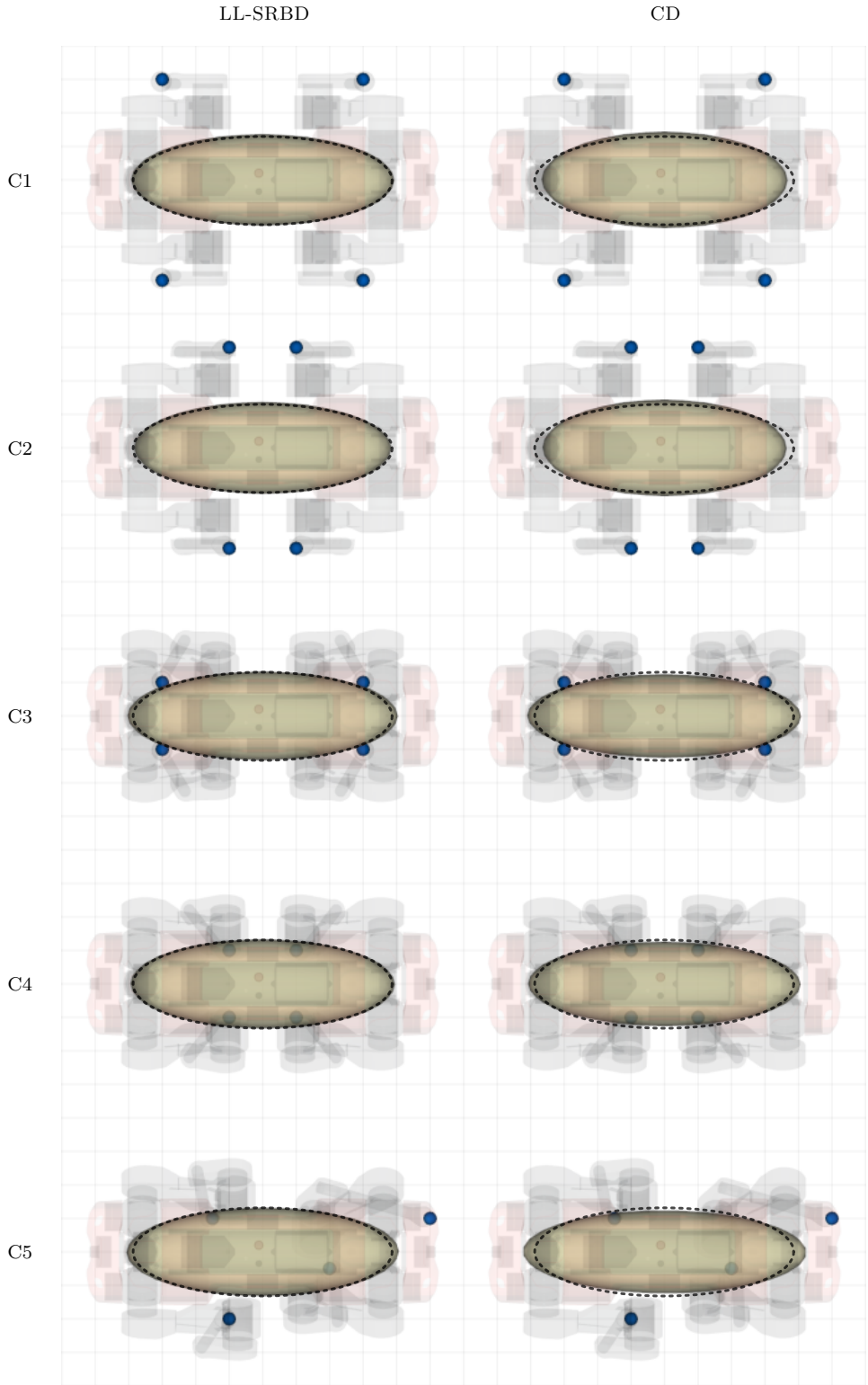


Figure 6.4: A top-down orthographic view comparing the equimomental ellipsoids of the LL-SRBD (left) and CD (right) models for 5 leg configurations (top-bottom). The grid spacing is 0.1 m. The dashed oval represents the equimomental ellipsoid of the SRBD model for a default configuration (nominal stance). The corresponding rotational inertia values are presented in Table 6.1.

C.	Model	$\Delta^{\mathcal{B}}\mathbf{I}$ (%)		
		ΔI_{xx}	ΔI_{yy}	ΔI_{zz}
C1	LL-SRBD	11.4	9.4	9.9
	CD	17.7	2.4	7.7
C2	LL-SRBD	11.4	6.4	7.0
	CD	12.9	−2.6	4.4
C3	LL-SRBD	5.2	9.4	7.9
	CD	−10.1	0.4	−4.1
C4	LL-SRBD	5.1	6.3	5.0
	CD	−14.0	−5.4	−6.0
C5	LL-SRBD	2.4	7.6	7.3
	CD	−18.2	−5.7	−4.9

Table 6.2: A comparison of the percentage change of the I_{xx}, I_{yy} and I_{zz} components of the rotational inertia of the LL-SRBD and CD models, relative to the fixed values of the SRBD model, shown in Table 6.1.

6.5 Discussion

Despite Wang et al. [288] having successfully applied their proposed LL-SRBD model to the ANYbotics ANYmal B quadruped, following our analysis in Sec. 6.4, we argue that this simplified model is not sufficiently applicable to robots with heavy knee joints and highly bent legs. While the model offers the capacity to shape the rotational inertia of the robot, the assumption that the point mass of a leg lies between its end effector and hip does not hold. Sometimes, the obtained inertia tensor deviates from the real values. The model is much better suited to bipedal robots (e.g., Draco or Cassie), especially those with straight legs like Slider. An alternative model by Ficht and Behnke [293] uses a triangle mass distribution to better represent the inertia of a bent leg.

Meanwhile, the values calculated by the accurate CD model closely agree with the true mass moments of inertia enabling the computation of angular velocity that produces the desired angular momentum needed for nimble motion.

Typically, the RBD model has been used for open-loop planning; however, Mastalli et al. [188] recently demonstrated it being applied in a RH fashion be leveraging a differential dynamic programming (DDP) method which ex-

ploits the temporal structure of the optimal-control problem (OCP). This novel approach promises to enable the real-time planning of acrobatic motions by leveraging the properties of the whole body.

In this chapter, we derived the partial derivatives analytically, by hand, resulting in arguably strenuous algebra. Despite the drawbacks of being prone to errors and consuming to implement, this analytical solution is general and exact, and can be efficiently implemented (with proper memory management). The explicit equations provide for a better understanding of the problem's structure. To reduce the computation time, one can decide to identify and ignore the terms that are slow to compute but make an insignificant contribution to the value of the derivative (e.g., when terms differ by orders of magnitude). Alternatives include applying finite difference (FD), or using code-generating software such as those designed for symbolic differentiation (SD) or automatic differentiation (AD), e.g., SymPy (Meurer et al. [294]) or CasADi (Andersson et al. [295]). FD is slow and prone to precision errors; however, it is easy to use and is a helpful validation tool. Symbolic differentiation (SD) produces code that is memory intensive and slow. AD is equivalently fast as an efficient implementation of analytical derivatives (Margossian [296]).

6.6 Conclusion

This chapter focused on a thorough derivation of the LL-SRBD and CD models and their first-order partial derivatives required by an optimization solver for efficient computation. To varying degrees of exactness, these models are aware of the properties of the rigid bodies that make up the robot. This enables the rotational inertia of the robot to be shaped in a way that facilitates the generation of nimble locomotion trajectories.

We have showed that, due to its assumptions, the LL-SRBD model is not sufficiently representative of the dynamics governing a quadrupedal robot with heavy knee joints. The model produces moments of inertia which occasionally deviate from the values obtained using the more accurate CD model.

In our formulation we used a direct collocation transcription method which parameterizes the Euclidean-space variables using third-order polynomials of Hermite form. This approach enables the optimization of the contact locations and the timings of contact events.

While we intended to implement the derived partial derivatives, we have resorted to using FD at the time of writing. Future work will involve comparing the computational time of evaluating the hand-coded derivatives with those generated by CasADi as well as comparing the performance of the two models when applied to a real robot.

7 Conclusion

The objective of this thesis was to develop new methods for kinodynamic planning of quadrupedal robot locomotion. We aspired to advance motion-generation research in legged robotics beyond the typical limitations such as short-horizon planning, simplified dynamics models, predefined gaits, footholds chosen according to heuristics, and quasi-static traversal of mildly irregular terrain. We aimed to achieve agile locomotion on complex terrain by availing of models of higher fidelity and real-time perception to simultaneously plan body motion and footsteps. Additionally, we leveraged expert demonstrations to imitate desirable locomotion styles. During this study, we investigated combining numerical optimization with machine learning techniques within our proposed framework called Receding-Horizon Experience-Controlled Adaptive Legged Locomotion (RHECALL).

7.1 Concluding summary

In Chapter 3, we described how we adapted the trajectory optimization framework of Winkler et al. [86], which uses the single-rigid-body dynamics (SRBD) model, and a whole-body controller of Bellicoso et al. [184] for real-world deployment using the ANYbotics ANYmal B and C quadrupeds. We presented

the direct collocation discretization method which was used to transcribe an optimal-control problem (OCP) into a nonlinear programming (NLP) problem. This problem was solved with an interior-point method (IPM) solver IPOPT. The method parameterizes the state and control variables using third-order polynomials of Hermite form, i.e., polynomials defined by the values and derivatives at their start and end points. This approach facilitates the independent discretization of the body and end-effector trajectories enabling the generation of aperiodic contact sequences. We leveraged the parameterization to derive computationally efficient analytical costs which we applied to minimize body velocities, and forces and their derivatives. To maximize the use of the robot's end-effector operational space, we proposed using superquadrics as range-of-motion constraints. Perception information in the form of a 2.5D heightmap was integrated; we used OpenCV filters to detect edges and compute a costmap to select advantageous footholds on uneven terrain and to create an 'envelope' constraint which we used to guide the linear and angular position of the robot's centre of mass (CoM), relative to the traversed terrain.

In Chapter 4, we presented a receding-horizon extension of our approach. We reformulated the NLP problem to use segments with dynamic initial and final conditions and defined separate CoM and end-effector time horizons of different durations and asynchronous start and end points. To enable a sustained and quicker generation of consecutive dynamic segments, we developed a heuristic initialization scheme which combined parts of a previous solution with a linearly interpolated guess towards the goal pose specified by the user. Thanks to real-time perception, we were able to optimize foot placement. Additionally, we explored replacing the SRBD equality constraint with our learned stability margin (Orsolino et al. [3]) as a measure of dynamic stability. This allowed the optimization to implicitly consider the torque limits of the actuators and cause the robot to assume a more cautious stance on uneven terrain.

In Chapter 5, we investigated accelerating the convergence of the IPM solver to a locally optimal solution using a data-driven initialization of the primal vari-

ables of the NLP problem. We described the system used for the generation, processing and imitation learning (behaviour cloning) of optimal trajectories generated using the approach outlined in Chapter 3. We analysed two approaches. First, the LearnedInit initializer was used for open-loop reconstruction of long-horizon trajectories. While this regressor was trained in a supervised fashion and was overfitted to a specific task, we demonstrated the ability to offset the computational cost of the nonlinear optimization by quickly producing a high-quality guess which required no or very few solver iterations to refine the trajectories until they were executable on the real robot. Then, we deployed the LMTR initializer (Surovik et al. [4]) within our receding-horizon locomotion planning framework. We showed that using the latent space offers the ability to represent and reconstruct dynamic segments of multimodal trajectories dependent on the task which included terrain description, i.e., prominent features such as edges.

In Chapter 6, the main limitation of the SRBD model was addressed. Specifically, an attempt was made to extend the applicability of the proposed approach to quadrupedal robots with significant leg inertias such as the ANYmal robots. Two more accurate dynamics models and their first-order partial derivatives were derived analytically. The lump leg single-rigid-body dynamics (LL-SRBD) model approximates each leg as a point mass positioned along the line between the end-effector and the hip. The centroidal dynamics (CD) model is more representative of the real system as it accounts for the inertia of each link. To facilitate intuitive usage of costs, constraints and initializations, we parameterized these models by Euclidean-space variables. We demonstrated that these models can shape the centroidal inertia enabling more agile behaviours.

In this thesis, we have focused on a real-world systems-level integration of many optimization- and learning-based components. The methods proposed herein were implemented and tested in realistic physics simulators, such as Gazebo and RaiSim, and validated experimentally on the ANYbotics ANYmal B and C quadrupeds. While the models derived in Chapter 6 have not been

tested on real hardware, we believe this mainly theoretical contribution may inspire further research.

7.2 Future work

During the studies conducted as part of this thesis, some limitations of the proposed approaches have become apparent and interesting avenues for further research have been identified.

At its core, RHECALL uses numerical optimization to generate explainable locomotion trajectories. The framework is flexible in the way it allows an arbitrary performance metric or constraint to be defined. Using a direct collocation discretization method enabled us to produce natural-looking, high-fidelity, long-horizon trajectories.

7.2.1 Leg inertia compensation

By replacing the SRBD model with a more accurate but tractable representation, e.g., LL-SRBD or CD models presented in Chapter 6, the framework could be used for walking robots with significant leg inertias, including humanoids.

7.2.2 Faster receding-horizon planning

Solving the formulated NLP problem is challenging because it is nonconvex. While a sparse nonlinear solver can find a locally optimal solution, the per-iteration computation time is prohibitive for receding-horizon use cases.

Careful consideration should be given to ensure that an optimization-based framework intended for real-time deployment is efficiently implemented. Specifically, we would like to redesign RHECALL to use more cache-friendly data structures, for reasons discussed in App. A.4.

Changing the transcription method could also be considered. Mastalli et al. [146] demonstrated that an alternative numerical method based on differential dynamic programming (DDP) outperforms direct transcription. It exploits

the temporal structure of the optimal-control problem (OCP) by solving a sequence of smaller subproblems. These subproblems require very small matrix factorizations relative to the factorizations needed by an incremental projection method, i.e., to compute the search directions to satisfy the Karush-Kuhn-Tucker (KKT) conditions.

7.2.3 Improved learned initialization

To substantially increase the chances of convergence and to offset the computation cost of the optimization, we integrated a neural network as a means of initializing the NLP solver. The receding-horizon LMTR initializer was trained to imitate segments of expert, long-horizon trajectories. To ensure a constant number of variables, the CoM trajectories were cut at the start and end nodes of the polynomials, i.e., at the discretization interval of the dynamics, and the end-effector trajectories included stance-swing-stance phase sequences. This approach forced the initialization to be synchronized with the discretization interval (essentially limiting the achievable replanning frequency) and prevented the solver from modifying the path of a moving end-effector.

By devising a suitable segmentation method and an appropriate problem formulation which would retain the polynomial parameterization, the replanning could be made more reactive to disturbances such as contact slippage.

7.2.4 Reactive policy via inverse reinforcement learning

Although optimal control requires substantial design and tuning effort, it is capable of generating elaborate trajectories which are otherwise hard to discover through trial and error. Meanwhile, locomotion policies trained using reinforcement learning (RL) exhibit impressive robustness to uncertainty and perturbations. Recent works, like that of Brakel et al. [212] and Vollenweider et al. [215], utilized inverse reinforcement learning (IRL) to obtain reactive policies that imitate styles of expert demonstrations. The trajectories generated using RHECALL would be well-suited for such an application. The formulation, costs

and constraints proposed in Chapter 3 offer extensive control over the style of the trajectories. As discussed in Chapter 5, we already developed an automated process used to generate large datasets of solutions for imitation learning.

7.2.5 Three-dimensional environment representation

In this work, we used a 2.5D heightmap to model the terrain for planning contact locations and controlling the motion of the CoM. Tranzatto et al. [297] highlighted navigation challenges experienced during the Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge; they showed a confined space overhung by a stalactite appearing as a phantom obstacle (artefact) within the 2.5D map. A detailed 3D representation of the environment, such as that proposed by Schmid et al. [298], would enable more general navigation in such situations. Scherer [299] used a volumetric representation during the aforementioned Challenge. While such a representation is often nonconvex, a sampling-based method (discussed in Sec. 2.4.2), or the learning-based initialization proposed in Chapter 5, could be used to guide the solver to a specific locally optimal solution.

A
Appendices

A.1 Analytical costs and their partial derivatives

Recalling the integral cost defined in Eq. 3.6 and letting $x = x_i(t)$, $x_\mathrm{r} = x_{\mathrm{ref},i}(t)$, $\alpha = \omega_{\mathrm{err},i}$, $\beta = \omega_{\mathrm{der},i}$, the cost becomes

$$J_i(t) = \int_0^T \left(\alpha \left(x - x_\mathrm{r}\right)^2 + \beta \dot{x}^2\right) dt. \tag{A.1}$$

Expressing x in terms of the coefficients (a, b, c, d) defined in Eq. 3.2, Eq. A.1 can be evaluated to give

$$\begin{aligned} J_i(t) = {} & t^7 d^2 \alpha / 7 \\ & + t^6 cd\alpha / 3 \\ & + t^5 \left[\alpha \left(c^2 + 2bd\right) + 9d^2\beta\right] / 5 \\ & + t^4 \left[\alpha \left(2bc + 2d\left(a - x_\mathrm{r}\right)\right) + 12cd\beta\right] / 4 \\ & + t^3 \left[\alpha \left(b^2 + 2c\left(a - x_\mathrm{r}\right)\right) + \beta \left(4c^2 + 6bd\right)\right] / 3 \\ & + t^2 \left[2bc\beta + b\alpha\left(a - x_\mathrm{r}\right)\right] \\ & + t \left[b^2\beta + \alpha\left(a - x_\mathrm{r}\right)^2\right]. \end{aligned} \tag{A.2}$$

The partial derivatives of Eq. A.2 with respect to the duration of the polynomial (ΔT), the values (x_0, x_1) and the derivatives $(\dot{x}_0, \dot{x}_1)$ at the start and end nodes are as follows:

$$\begin{aligned} \frac{\partial J_i}{\partial x_0} = {} & \Big[\alpha T^2 \Big(156x_0 + 54x_1 + 22T\dot{x}_0 - 13T\dot{x}_1 - 210x_\mathrm{r}\Big) \\ & + \beta\Big(504x_0 - 504x_1 + 42T\dot{x}_0 + 42T\dot{x}_1\Big)\Big] / (210T), \end{aligned} \tag{A.3a}$$

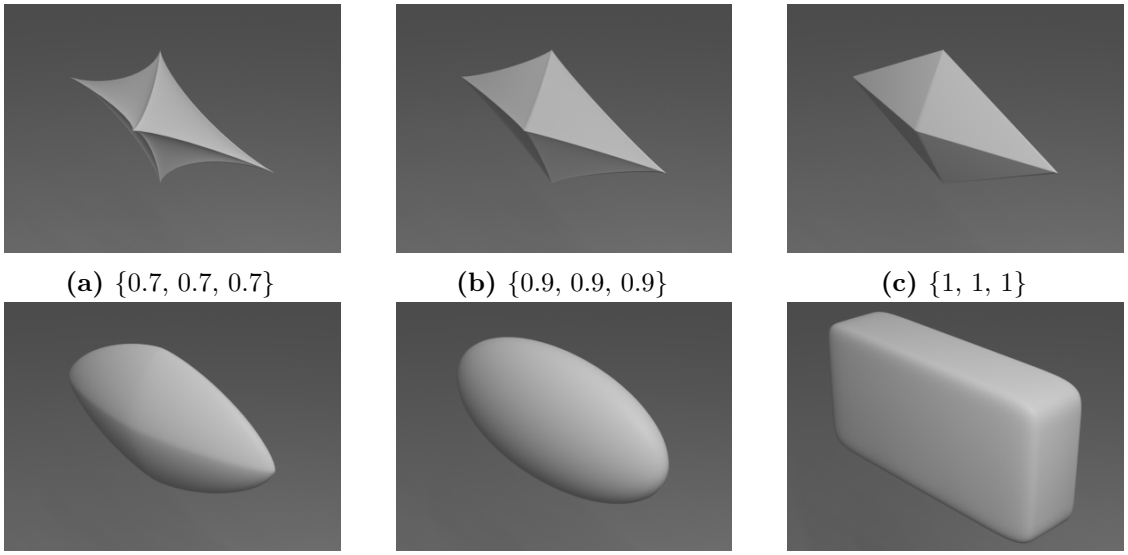
$$\begin{aligned} \frac{\partial J_i}{\partial x_1} = {} & \Big[\alpha T^2 \Big(54x_0 + 156x_1 + 13T\dot{x}_0 - 22T\dot{x}_1 - 210x_\mathrm{r}\Big) \\ & - \beta\Big(504x_0 - 504x_1 + 42T\dot{x}_0 + 42T\dot{x}_1\Big)\Big] / (210T), \end{aligned} \tag{A.3b}$$

$$\begin{aligned} \frac{\partial J_i}{\partial \dot{x}_0} = {} & \Big[\alpha T^2 \Big(22x_0 + 13x_1 + 4T\dot{x}_0 - 3T\dot{x}_1 - 35x_\mathrm{r}\Big) \\ & + \beta\Big(42x_0 - 42x_1 + 56T\dot{x}_0 - 14T\dot{x}_1\Big)\Big] / 210, \end{aligned} \tag{A.3c}$$

$$\begin{aligned} \frac{\partial J_i}{\partial \dot{x}_1} = {} & \Big[\alpha T^2 \Big(-13x_0 - 22x_1 - 3T\dot{x}_0 + 4T\dot{x}_1 + 35x_\mathrm{r}\Big) \\ & + \beta\Big(42x_0 - 42x_1 - 14T\dot{x}_0 + 56T\dot{x}_1\Big)\Big] / 210, \end{aligned} \tag{A.3d}$$

$$\begin{aligned}\frac{\partial J_i}{\partial T} = \Big\{ & \alpha T^2 \Big[78x_0^2 + 78x_1^2 + 54x_0x_1 \\ & + 44Tx_0\dot{x}_0 - 44Tx_1\dot{x}_1 - 26Tx_0\dot{x}_1 + 26Tx_1\dot{x}_0 \\ & + 6T^2\dot{x}_0^2 + 6T^2\dot{x}_1^2 - 9T^2\dot{x}_0\dot{x}_1 \\ & + 70x_{\mathrm{r}}\Big(-3x_0 - 3x_1 - T\dot{x}_0 + T\dot{x}_1 + 3x_{\mathrm{r}}\Big)\Big] \\ & + 14\beta\Big(-18x_0^2 - 18x_1^2 + 36x_0x_1 \\ & + 2T^2\dot{x}_0^2 + 2T^2\dot{x}_1^2 - T^2\dot{x}_0\dot{x}_1\Big)\Big\}/(210T^2).\end{aligned} \tag{A.3e}$$

A.2 Examples of superquadrics



(a) {0.7, 0.7, 0.7} **(b)** {0.9, 0.9, 0.9} **(c)** {1, 1, 1}

(d) {1.5, 1.5, 1.5} **(e)** {2.0, 2.0, 2.0} **(f)** {10, 10, 10}

Figure A.1: Examples of superquadrics defined by Eq. 3.8 with fixed scalings $\{A, B, C\} = \{2, 0.5, 1\}$ and varying exponents $\{a, b, c\}$. Only the convex and differentiable variants ($\min(a, b, c) > 1$) are useful for optimization.

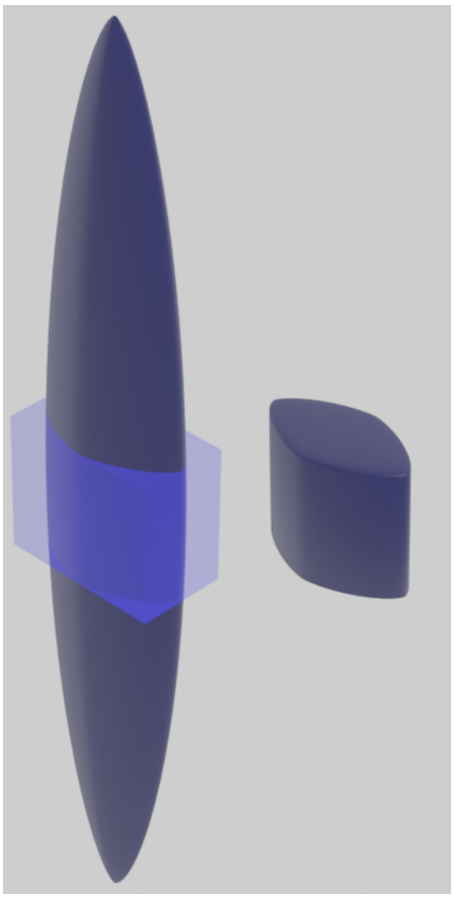


Figure A.2: A range-of-motion constraint resulting from an intersection of a superquadric and a box.

A.3 Terrain reconstruction for realistic simulation

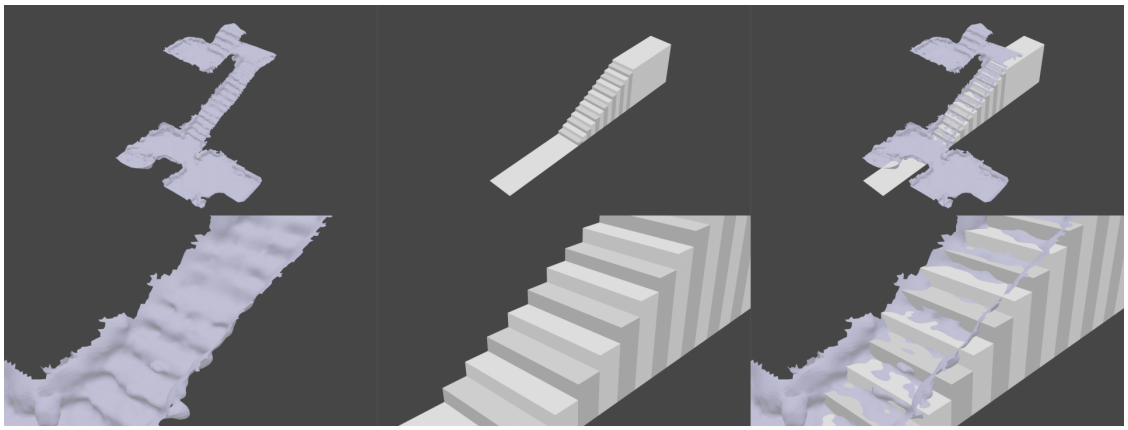


Figure A.3: A full (top) and close-up (bottom) comparison of meshes of a staircase which can be reconstructed from noisy RGB-D camera's depth pointclouds (left) or modelled using true information (middle). The accuracy of the reconstruction is demonstrated by overlaying the meshes (right). The information necessary for such terrain reconstruction can be obtained using a hand-held sensor. Such realistic model of the terrain can be used in simulation to test algorithms before real-world deployment.

A.4 Computation time

A.4.1 Discussion of the effects of TOWR's architecture

The TOWR (Winkler et al. [86]) framework uses a heavily abstracted architecture. The modular and flexible structure allows it to hide the implementation details of the trajectory optimization components and methods while simplifying and generalizing the process of formulating a legged locomotion problem involving many constraints and parameters. The framework is well-written from the perspective of a user interested in an intuitive way to formulate problems and generate trajectories in an offline setting.

Unfortunately, TOWR is not suitable for real-time applications. Its design does not leverage CPU (cache) and memory resources effectively. The combination of many levels of abstraction with separate data structures results in large objects and cache misses, and poor memory locality. Additionally, extensive heap memory allocations occur due to many methods creating temporary objects or unnecessarily copying data.

After the publication of our receding-horizon work (**Melon** et al. [2]), we refactored parts of TOWR that could be conveniently modified. We precomputed certain quantities to avoid repeated function calls, avoided recursively resizing some vectors, and reduced the number of local copies by using constant references wherever it was possible. Thanks to these changes, we achieved a nearly $50\,\%$ improvement in the computation speed of TOWR (which corresponds to a $33\,\%$ reduction in the overall optimization time). We concluded that we needed an even greater improvement to achieve satisfactory replanning frequencies which would enable robust locomotion in the presence of disturbances; however, further changes would require redesigning the framework from scratch which was beyond the scope of this project.

Based on the vast experience acquired during the process, we advise future researchers interested in using the phase-based parameterization (Winkler et al. [86]) for a real-time application to consider writing their own software.

Linear solver	Computation time (s)	Improvement (%)
MUMPS	27.56	0
MA27	26.76	2.9
MA57	24.84	9.9
MA77	30.50	−10.7

Table A.1: A comparison of the effects of different linear solvers on the total computation time (s) reported by the IPOPT solver when used with the TOWR-based framework presented in Chapter 3. Each trial involved 100 solver iterations. A Dell Precision 7520 laptop with the 4-core/8-thread Intel Xeon E3-1505M v6 @ 3 GHz CPU was used.

An approach worth investigating involves separating the classes containing the algorithms from the cache-friendly data structures as demonstrated by more computationally efficient frameworks PINOCCHIO (Carpentier et al. [151]) and CROCCODYL (Mastalli et al. [145]).

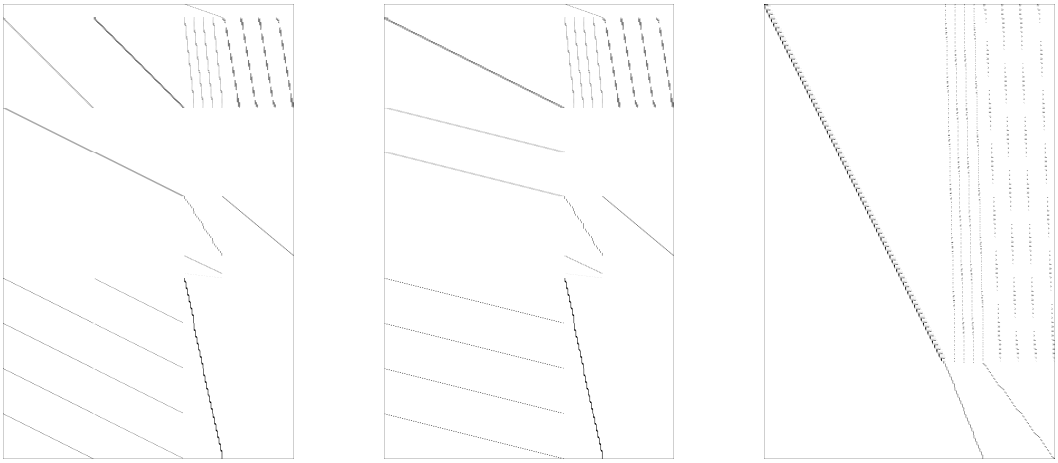
In Sec. A.4.2, Sec. A.4.3 and Sec. A.4.4, we discuss some results which demonstrate the components of our optimization pipeline that affect the total computation time. We investigate the structure of the problem and analyse the output information from the IPOPT solver.

A.4.2 Comparison of linear solvers

As part of the work on long-horizon, open-loop planning discussed in Chapter 3, we investigated replacing MUMPS, the default linear solver, to reduce the total computation time. As shown in Table A.1, the MA57 solver reduced it by 9.9 %. This conclusion aligns well with the recommendations of Tasseff et al. [165] and Ferrolho et al. [166] to use MA57 for small, robotics problems.

A.4.3 Reordering of the Jacobian matrix

As shown in Fig. A.4, we investigated the constraint Jacobian matrix sparsity patterns for a problem with 1560 optimization variables, 2444 constraints and 27 592 non-zero values (0.72 %). While it is possible to reorder the matrix by columns and/or rows, it does not have any effect on the overall computa-



(a) Original **(b)** Columns **(c)** Columns and rows

Figure A.4: The constraint Jacobian matrix sparsity patterns a without reordering, b with columns reordered, and c with columns and rows reordered. This problem involved an optimization horizon of 6 s with 6 steps per leg, and a discretization of 0.1 s.

tion time because the solver performs its own reordering regardless of the provided structure.

Disabling phase optimization can easily reduce the number of non-zero values by a factor of two. In our case, when the dynamics constraints are enforced at the polynomial nodes, there are tiny residuals (whose magnitude is less than 0.0001) which are caused by rounding errors due to the polynomial-based parameterization. These residuals account for $30\,\%$ of non-zero values and impact the computation time.

A.4.4 Analysis of the timing statistics of IPOPT

It was observed that the relationship between the base-motion horizon and the overall optimization time was linear with a slope in the range of 6 to 7 as can be deduced from the numbers in Table. A.2.

Using a base-motion horizon of $1.4\,\mathrm{s}$, the following timing statistics (in seconds) were obtained from IPOPT after 100 solver iterations:

```
OverallAlgorithm....................: 1.236 (wall: 1.262)
 PrintProblemStatistics.............: 0.000 (wall: 0.000)
 InitializeIterates.................: 0.029 (wall: 0.029)
```

Base-motion horizon (s)	**Computation time** (s)
0.8	0.115
1.0	0.160
1.4	0.220

Table A.2: A comparison of the effects of different base-motion horizons ΔT_{H} (s) on the total computation time (s) reported by the IPOPT solver when used with the RHECALL framework presented in Chapter 4. Each trial involved 10 solver iterations. A desktop with the 4-core/8-thread Intel Core i7-6700K @ 4 GHz CPU was used.

```
 UpdateHessian.....................: 0.006 (wall: 0.006)
 OutputIteration...................: 0.000 (wall: 0.001)
 UpdateBarrierParameter............: 0.103 (wall: 0.107)
 ComputeSearchDirection............: 0.090 (wall: 0.091)
 ComputeAcceptableTrialPoint.......: 0.109 (wall: 0.113)
 AcceptTrialPoint..................: 0.001 (wall: 0.002)
 CheckConvergence..................: 0.895 (wall: 0.913)
PDSystemSolverTotal................: 0.182 (wall: 0.185)
 PDSystemSolverSolveOnce...........: 0.172 (wall: 0.175)
 ComputeResiduals..................: 0.006 (wall: 0.007)
 StdAugSystemSolverMultiSolve......: 0.149 (wall: 0.152)
 LinearSystemScaling...............: 0.000 (wall: 0.000)
 LinearSystemSymbolicFactorization..: 0.000 (wall: 0.001)
 LinearSystemFactorization.........: 0.091 (wall: 0.093)
 LinearSystemBackSolve.............: 0.047 (wall: 0.047)
 LinearSystemStructureConverter.....: 0.000 (wall: 0.000)
  LinearSystemStructureConverterInit: 0.000 (wall: 0.000)
QualityFunctionSearch..............: 0.013 (wall: 0.014)
TryCorrector.......................: 0.000 (wall: 0.000)
Task1..............................: 0.003 (wall: 0.003)
Task2..............................: 0.002 (wall: 0.002)
Task3..............................: 0.001 (wall: 0.001)
Task4..............................: 0.000 (wall: 0.000)
```

```
Task5.............................: 0.001 (wall: 0.001)
Function Evaluations..............: 0.979 (wall: 0.999)
 Objective function...............: 0.025 (wall: 0.026)
 Objective function gradient......: 0.073 (wall: 0.077)
 Equality constraints.............: 0.068 (wall: 0.070)
 Inequality constraints...........: 0.001 (wall: 0.001)
 Equality constraint Jacobian.....: 0.000 (wall: 0.000)
 Inequality constraint Jacobian...: 0.812 (wall: 0.825)
 Lagrangian Hessian...............: 0.000 (wall: 0.000)
```

`CheckConvergence` appears to take significant portion of the overall time ($0.913\,\mathrm{s}$ out of $1.262\,\mathrm{s}$); however, rather than reporting the time taken to check the convergence criteria, the reported time is mainly due to the computation of the constraint Jacobian matrix (which is computed and cached during the `CheckConvergence` call) as explained at https://github.com/coin-or/Ipopt/issues/509.

More importantly, for 100 iterations, various NLP function evaluations in RHECALL take $0.999\,\mathrm{s}$ of which `Inequality constraint Jacobian` takes $0.825\,\mathrm{s}$. This indicates that $80\,\%$ of the time is spent outside IPOPT. As the number of iterations decreases, the percentage also decreases due to, e.g., the initialization in `InitializeIterates` ($0.029\,\mathrm{s}$) and the solving of the linear system in `PDSystemSolverTotal` ($0.185\,\mathrm{s}$). For small problems (receding-horizon), the portion of the solve time taken to solve the linear problem becomes significant; alternatives to solvers based on the IP method should be considered.

A.5 Analysis of LMTR results

The figures and some descriptions in this appendix have been adopted from Surovik et al. [4] to better explain the workings and performance of the LMTR initializer $P_\phi\colon\ (\mathbf{z};\mathbf{x}) \mapsto \hat{\boldsymbol{\tau}}$ and the mode-selecting policy $\boldsymbol{\pi}(\mathbf{x})$ used for the real-time RH work presented in Chapter 5.

A.5.1 Analysis of mode selection

Inspection of the distribution of expert modes $\mathbf{z} \in \mathcal{D}_\mathbf{z}$, shown in Fig. A.5a, reveals that the latent space $\mathcal{Z}$ is mainly used for representing decisions that cannot be unambiguously determined through the task-condition input to the regressor. For example, a foothold could avoid an obstacle through either a short stride or a long one, but not an average one—and thus $\mathcal{Z}$ shows clear organization in terms of stride length.

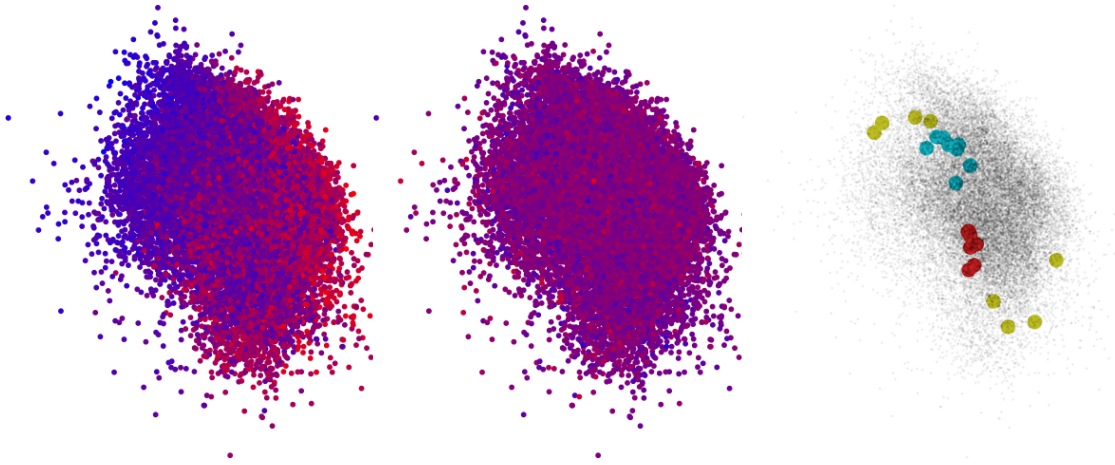
Meanwhile, as shown in Fig A.5b, height information was not contained in $\mathcal{Z}$; it can be deduced that this information is provided by the initial state which is part of the task $\mathbf{x}$.

Fig. A.5c shows a bimodal distribution of yellow modes which indicates that the mode-selecting policy $\boldsymbol{\pi}$ can be multimodal.

A.5.2 Analysis of mode dimensionality

A comparison of three types of initializers—a unimodal regressor, LMTR with varying mode dimensionality, and a TL—is shown in Fig. 5.4c. All initializers were used in a RH fashion to cross gaps as is shown in Fig. 5.4c. Notably, the experiments happened in non-real time and without online refinement by the optimizer—we call such a deployment a 'rollout'. 100 rollouts were generated for each type of initializer, with each rollout lasting $30\,\mathrm{s}$ (100 RH replanning cycles), corresponding to around $11\,\mathrm{m}$ travelled and about 13 obstacles (gaps) crossed.

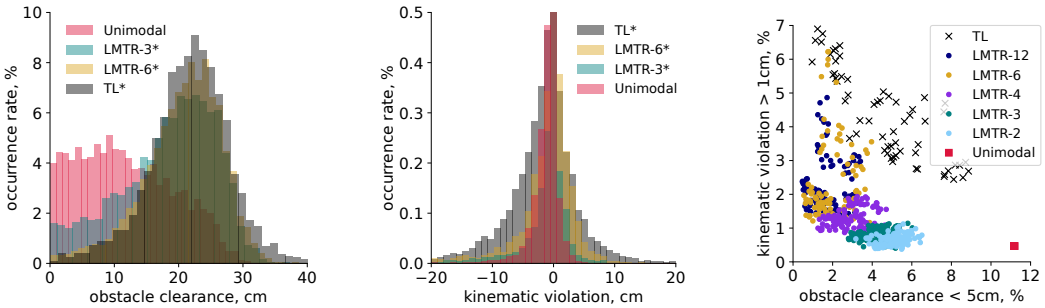
As shown in Fig. A.6a, the unimodal regressor commonly selects footholds very close to the obstacle, while LMTR and TL both strongly bias their footholds



(a) Left-front stride **(b)** Height change **(c)** Neighbouring tasks

Figure A.5: Mode choices $\mathbf{z} \in \mathbb{R}^3$ coloured based on the properties of the associated trajectory segments $\hat{\boldsymbol{\tau}}$ for a quadruped crossing fixed-sized gaps of random orientations. A distinct and independent trend in step lengths (blue is short, red is long) of (a) left-front stride was observed as associated with the need to either take normal or longer steps (when crossing a gap). (b) Height change information appeared to not be contained in the latent space $\mathcal{Z}$. (c) Four clusters (two yellow) of three distinct modes (of which the yellow group is bimodal) associated with three neighbouring sets of tasks.

away. Performance of LMTR plateaus at around 6 to 12 latent dimensions, at which point it roughly matches TL in terms of obstacle avoidance. Fig. A.6b, however, reveals that TL has a much higher rate and degree of kinematic violations. Performance trade-offs can be made by prioritizing some categories of task variables over others in the weight vector used for nearest neighbors, as shown in Fig. A.6c, where weighting the state and obstacle elements of $\mathbf{x}$, respectively, promotes the kinematic and obstacle metrics. Altogether, LMTR offers about 5 times less frequent kinematic violation than TL for a given level of obstacle avoidance performance.



(a) Foot gap clearance **(b)** Foot box violation **(c)** Correlation of failures

Figure A.6: A comparison of a unimodal regressor, LMTR with varying mode dimensionality, and a TL: (a) probability distribution of the closest foothold to each obstacle being a certain distance away, averaged across all four feet, (b) probability distribution of a foot violating its nominal range of motion by a given amount, and (c) rates of exceeding 1 cm kinematic violation and of failing to ensure 5 cm obstacle clearance, where each point per series represents aggregate performance under a different nearest-neighbors weight vector. In the legends, the numbers indicate the dimensionality of $\mathbf{z}$, and the asterisk symbol * indicates that a weight vector which resulted in the best performance (specific to the particular initializer type and metric) was used.

A.6 Quaternion and its partial derivatives

The terms of a quaternion $Q = w + x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}} \in \mathbb{H}$ can be expressed in terms of the Euler angles $\boldsymbol{\theta} \in \mathbb{R}^3$, i.e., roll ϕ, pitch θ and yaw ψ as follows

$$Q_w = \cos\left(\phi/2\right)\cos\left(\theta/2\right)\cos\left(\psi/2\right) + \sin\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right), \tag{A.4a}$$

$$Q_x = \sin\left(\phi/2\right)\cos\left(\theta/2\right)\cos\left(\psi/2\right) - \cos\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right), \tag{A.4b}$$

$$Q_y = \cos\left(\phi/2\right)\sin\left(\theta/2\right)\cos\left(\psi/2\right) + \sin\left(\phi/2\right)\cos\left(\theta/2\right)\sin\left(\psi/2\right), \tag{A.4c}$$

$$Q_z = \cos\left(\phi/2\right)\cos\left(\theta/2\right)\sin\left(\psi/2\right) - \sin\left(\phi/2\right)\sin\left(\theta/2\right)\cos\left(\psi/2\right). \tag{A.4d}$$

The term-wise partial derivatives are

$$\frac{\partial Q_w}{\partial \phi} = \left(1/2\right)\left(\cos\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right) - \cos\left(\theta/2\right)\cos\left(\psi/2\right)\sin\left(\phi/2\right)\right), \tag{A.5a}$$

$$\frac{\partial Q_w}{\partial \theta} = \left(1/2\right)\left(\cos\left(\theta/2\right)\sin\left(\phi/2\right)\sin\left(\psi/2\right) - \cos\left(\phi/2\right)\cos\left(\psi/2\right)\sin\left(\theta/2\right)\right), \tag{A.5b}$$

$$\frac{\partial Q_w}{\partial \psi} = \left(1/2\right)\left(\cos\left(\psi/2\right)\sin\left(\phi/2\right)\sin\left(\theta/2\right) - \cos\left(\phi/2\right)\cos\left(\theta/2\right)\sin\left(\psi/2\right)\right), \tag{A.5c}$$

$$\frac{\partial Q_x}{\partial \phi} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\theta/2\right)\cos\left(\psi/2\right) + \sin\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right)\right), \tag{A.5d}$$

$$\frac{\partial Q_x}{\partial \theta} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\theta/2\right)\sin\left(\psi/2\right) - \cos\left(\psi/2\right)\sin\left(\phi/2\right)\sin\left(\theta/2\right)\right), \tag{A.5e}$$

$$\frac{\partial Q_x}{\partial \psi} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\psi/2\right)\sin\left(\theta/2\right) - \cos\left(\theta/2\right)\sin\left(\phi/2\right)\sin\left(\psi/2\right)\right), \tag{A.5f}$$

$$\frac{\partial Q_y}{\partial \phi} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\theta/2\right)\sin\left(\psi/2\right) - \cos\left(\psi/2\right)\sin\left(\phi/2\right)\sin\left(\theta/2\right)\right), \tag{A.5g}$$

$$\frac{\partial Q_y}{\partial \theta} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\theta/2\right)\cos\left(\psi/2\right) - \sin\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right)\right), \tag{A.5h}$$

$$\frac{\partial Q_y}{\partial \psi} = \left(1/2\right)\left(\cos\left(\theta/2\right)\cos\left(\psi/2\right)\sin\left(\phi/2\right) - \cos\left(\phi/2\right)\sin\left(\theta/2\right)\sin\left(\psi/2\right)\right), \tag{A.5i}$$

$$\frac{\partial Q_z}{\partial \phi} = \left(1/2\right)\left(\cos\left(\phi/2\right)\cos\left(\psi/2\right)\sin\left(\theta/2\right) - \cos\left(\theta/2\right)\sin\left(\phi/2\right)\sin\left(\psi/2\right)\right), \tag{A.5j}$$

$$\frac{\partial Q_z}{\partial \theta} = \Big(1/2\Big)\left(\cos\Big(\theta/2\Big)\cos\Big(\psi/2\Big)\sin\Big(\phi/2\Big) - \cos\Big(\phi/2\Big)\sin\Big(\theta/2\Big)\sin\Big(\psi/2\Big)\right), \tag{A.5k}$$

$$\frac{\partial Q_z}{\partial \psi} = \Big(1/2\Big)\left(\cos\Big(\phi/2\Big)\cos\Big(\theta/2\Big)\cos\Big(\psi/2\Big) + \sin\Big(\phi/2\Big)\sin\Big(\theta/2\Big)\sin\Big(\psi/2\Big)\right). \tag{A.5l}$$

References

[1] **O. Melon**, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable Trajectories for Dynamic Quadrupeds using Analytical Costs and Learned Initializations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1410–1416 [cited on pages 5, 6, 9, 10, 38, 52, 91, 96, 100, 103, 115].

[2] **O. Melon**, R. Orsolino, D. Surovik, M. Geisert, I. Havoutis, and M. Fallon, "Receding-Horizon Perceptive Trajectory Optimization for Dynamic Legged Locomotion with Learned Initialization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 9805–9811 [cited on pages 5, 6, 9, 10, 52, 73, 91, 96, 103, 115, 152].

[3] R. Orsolino, S. Gangapurwala, **O. Melon**, M. Geisert, I. Havoutis, and M. Fallon, "Rapid Stability Margin Estimation for Contact-Rich Locomotion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 8485–8492 [cited on pages 6, 10, 73, 83, 84, 89, 141].

[4] D. Surovik, **O. Melon**, M. Geisert, M. Fallon, and I. Havoutis, "Learning an expert skill-space for replanning dynamic quadruped locomotion over obstacles," in *Proceedings of the Conference on Robot Learning*, PMLR, 2021 [cited on pages 6, 10, 91–93, 95, 96, 99, 101–103, 142, 157].

[5] ANYbotics. "ANYbotics | Autonomous Legged Robots for Inspection and More," ANYbotics. [], [Online]. Available: `https://www.anybotics.com/` [cited on pages 7, 14, 64].

[6] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 38–44 [cited on pages 7, 14].

[7] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "ANYmal - toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, Sep. 2, 2017 [cited on pages 7, 14].

[8] A. L. Mitchell, M. Engelcke, O. P. Jones, D. Surovik, S. Gangapurwala, **O. Melon**, I. Havoutis, and I. Posner, "First Steps: Latent-Space Control with Semantic Constraints for Quadruped Locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5343–5350 [cited on page 10].

[9] R. Hoggett. "Walking Machines inc. Steam Man Timeline," cyberneticzoo.com. [2022], [Online]. Available: https://cyberneticzoo.com/walking-machine-time-line/ [cited on page 12].

[10] H.-S. Yan, *Reconstruction Designs of Lost Ancient Chinese Machinery*, red. by M. Ceccarelli, ser. History Of Mechanism And Machine Science. Dordrecht: Springer Netherlands, 2007, vol. 3 [cited on page 12].

[11] M. E. Moran, "The da Vinci robot," *Journal of Endourology*, vol. 20, no. 12, pp. 986–990, Dec. 2006 [cited on page 12].

[12] "Hooke Folio Online." [], [Online]. Available: http://www.livesandletters.ac.uk/cell/Hooke/Hooke.html [cited on page 12].

[13] M. H. Raibert, "Legged robots," *Communications of the ACM*, vol. 29, no. 6, pp. 499–514, Jun. 1, 1986 [cited on page 12].

[14] E. Muybridge, *Animals in motion.* New York: Dover Publ, 1957 [cited on page 12].

[15] E. Muybridge, *The Human Figure in Motion.* London: Chapman & Hall, ld, 1901 [cited on page 12].

[16] L. Rygg, "Mechanical horse," U.S. Patent 491927A, Feb. 14, 1893 [cited on page 12].

[17] H. W. Wallace, "Vehicle," U.S. Patent 2371368A, Mar. 13, 1945 [cited on page 12].

[18] K. H. Lundberg and T. W. Barton, "History of Inverted-Pendulum Systems," *IFAC Proceedings Volumes*, 8th IFAC Symposium on Advances in Control Education, vol. 42, no. 24, pp. 131–135, Jan. 1, 2010 [cited on page 12].

[19] D. J. Todd, *Walking Machines*. Boston, MA: Springer US, 1985 [cited on pages 12, 15].

[20] P. McKerrow and H. Zima, *Introduction to Robotics*. Addison-Wesley Publishing Company, 1991, 834 pp. [cited on page 12].

[21] V. S. Gurfinkel, E. V. Gurfinkel, A. Y. Shneider, E. A. Devjanin, A. V. Lensky, and L. G. Shtilman, "Walking robot with supervisory control," *Mechanism and Machine Theory*, vol. 16, no. 1, pp. 31–36, Jan. 1, 1981 [cited on page 12].

[22] D. Gorinevsky and A. Y. Shneider, "Force Control in Locomotion of Legged Vehicles over Rigid and Soft Surfaces," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 4–23, Apr. 1, 1990 [cited on page 12].

[23] M. H. Raibert, "Hopping in legged systems — Modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, May 1984 [cited on page 13].

[24] M. H. Raibert, H. B. Brown, and M. Chepponis, "Experiments in Balance with a 3D One-Legged Hopping Machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, Jun. 1, 1984 [cited on page 13].

[25] M. Raibert, M. Chepponis, and H. Brown, "Running on four legs as though they were one," *IEEE Journal on Robotics and Automation*, vol. 2, no. 2, pp. 70–82, Jun. 1986 [cited on page 13].

[26] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato, "The Realization of Dynamic Walking by the Biped Walking Robot WL-10 RD," *Journal of the Robotics Society of Japan*, vol. 3, no. 4, pp. 325–336, 1985 [cited on page 13].

[27] R. Playter and M. Raibert, "Control Of A Biped Somersault In 3D," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Jul. 1992, pp. 582–589 [cited on page 13].

[28] D. J. Todd, "An evaluation of mechanically co-ordinated legged locomotion (The Iron Mule Train revisited)," *Robotica*, vol. 9, no. 4, pp. 417–420, Dec. 1991 [cited on page 13].

[29] M. Fujita and H. Kitano, "Development of an Autonomous Quadruped Robot for Robot Entertainment," *Autonomous Robots*, vol. 5, no. 1, pp. 7–18, Mar. 1, 1998 [cited on page 13].

[30] "Sony Global - Press Release - Sony Launches Four-Legged Entertainment Robot "AIBO" Creates a New Market for Robot-Based Entertainment." [Oct. 28, 2012], [Online]. Available: `https://web.archive.org/web/20121028033310/http://www.sony.net/SonyInfo/News/Press_Archive/199905/99-046/index.html` [cited on page 13].

[31] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: System overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Sep. 2002, 2478–2483 vol.3 [cited on page 13].

[32] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "BigDog, the Rough-Terrain Quadruped Robot," *IFAC Proceedings Volumes*, 17th IFAC World Congress, vol. 41, no. 2, pp. 10 822–10 825, Jan. 1, 2008 [cited on page 13].

[33] M. P. Murphy, A. Saunders, C. Moreira, A. A. Rizzi, and M. Raibert, "The LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 145–149, Feb. 1, 2011 [cited on page 13].

[34] C. D. Remy, O. Baur, M. Latta, A. Lauber, M. Hutter, M. A. Hoepflinger, C. Pradalier, and R. Siegwart, "Walking and crawling with ALoF - a robot for autonomous locomotion on four legs," *The industrial robot*, vol. 38, no. 3, pp. 264–268, 2011 [cited on page 13].

[35] M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, C. D. Remy, and R. Siegwart, "STARLETH: A compliant quadrupedal robot for fast, efficient, and versatile locomotion," presented at the 15th International Conference on Climbing and Walking Robot (CLAWAR 2012), Autonomous Systems Lab, ETH Zurich, 2012 [cited on page 13].

[36] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ – a hydraulically and electrically actuated quadruped robot," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, Sep. 1, 2011 [cited on page 13].

[37] C. Semini, V. Barasuol, J. Goldsmith, M. Frigerio, M. Focchi, Y. Gao, and D. G. Caldwell, "Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 635–646, Apr. 2017 [cited on page 13].

[38] E. Ackerman. "Watch the HyQReal Robot Pull an Airplane," IEEE Spectrum. [May 23, 2019], [Online]. Available: https://spectrum.ieee.org/watch-iits-new-hyqreal-quadruped-robot-pull-an-airplane [cited on page 13].

[39] S. Kajita, "HRP Robots (e.g. HRP-4)," in *Humanoid Robotics: A Reference*, A. Goswami and P. Vadakkepat, Eds., Dordrecht: Springer Netherlands, 2016, pp. 1–16 [cited on page 14].

[40] L. Natale, C. Bartolozzi, F. Nori, G. Sandini, and G. Metta, "iCub," in *Humanoid Robotics: A Reference*, A. Goswami and P. Vadakkepat, Eds., Dordrecht: Springer Netherlands, 2019, pp. 291–323 [cited on page 14].

[41] A. Ajoudani, J. Lee, A. Rocchi, M. Ferrati, E. M. Hoffman, A. Settimi, D. G. Caldwell, A. Bicchi, and N. G. Tsagarakis, "A manipulation framework for compliant humanoid COMAN: Application to a valve turning task," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov. 2014, pp. 664–670 [cited on page 14].

[42] DARPA. "Debut of Atlas Robot." [], [Online]. Available: https://www.darpa.mil/about-us/timeline/debut-atlas-robot [cited on page 14].

[43] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, Mar. 1, 2016 [cited on pages 14, 23, 115].

[44] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "TALOS: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 689–695 [cited on page 14].

[45] H.-W. Park and S. Kim, "The MIT Cheetah, an Electrically-Powered Quadrupedal Robot for High-Speed Running," *Journal of the Robotics Society of Japan*, vol. 32, no. 4, pp. 323–328, 2014 [cited on page 14].

[46] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design Principles for Energy-Efficient Legged Locomotion and Implementation on the MIT Cheetah Robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, Jun. 2015 [cited on page 14].

[47] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 2245–2252 [cited on page 14].

[48] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT Cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, Feb. 1, 2017 [cited on page 14].

[49] M. Chignoli, S. Morozov, and S. Kim, "Rapid and Reliable Quadruped Motion Planning with Omnidirectional Jumping," arXiv, arXiv:2111.13648, Mar. 5, 2022 [cited on pages 14, 38].

[50] Ghost Robotics. "Home | Philadelphia | Ghost Robotics." [], [Online]. Available: https://www.ghostrobotics.io/ [cited on page 14].

[51] Boston Dynamics. "Boston Dynamics | Changing Your Idea of What Robots Can Do," Boston Dynamics. [], [Online]. Available: https://www.bostondynamics.com/ [cited on pages 14, 15].

[52] Z. Chen. "Unitree," unitree. [], [Online]. Available: https://www.unitree.com/ [cited on page 15].

[53] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An Open Torque-Controlled Modular Robot Architecture for Legged Locomotion Research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, Apr. 2020 [cited on page 15].

[54] Agility Robotics. "Agility Robotics," Agility Robotics. [], [Online]. Available: https://agilityrobotics.com [cited on page 15].

[55] Wandercraft. "Wandercraft | Ordinary Life for Extraordinary People." [], [Online]. Available: https://en.wandercraft.eu/ [cited on page 15].

[56] G. Endo and S. Hirose, "Study on Roller-Walker - Adaptation of characteristics of the propulsion by a leg trajectory -," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 1532–1537 [cited on page 15].

[57] F. Cordes, C. Oekermann, A. Babu, D. Kuehn, T. Stark, and F. Kirchner, "An Active Suspension System for a Planetary Rover," *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, p. 7, 2014 [cited on page 15].

[58] W. Reid, A. H. Gokto˘gan, and S. Sukkarieh, "Moving MAMMOTH: Stable Motion for a Reconfigurable Wheel-On-Leg Rover," p. 10, 2014 [cited on page 15].

[59] T. Klamt and S. Behnke, "Anytime hybrid driving-stepping locomotion planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada: IEEE Press, Sep. 24, 2017, pp. 4444–4451 [cited on page 15].

[60] M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, and S. Coros, "Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels," *ACM Transactions on Graphics*, vol. 37, no. 4, 160:1–160:12, Jul. 30, 2018 [cited on page 15].

[61] V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, F. Weber, R. Deuber, and R. Siegwart, "Ascento: A Two-Wheeled Jumping Robot," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7515–7521 [cited on page 15].

[62] T. Klamt, M. Schwarz, C. Lenz, L. Baccelliere, D. Buongiorno, T. Cichon, A. DiGuardo, D. Droeschel, M. Gabardi, M. Kamedula, N. Kashiri, A. Laurenzi, D. Leonardis, L. Muratore, D. Pavlichenko, A. S. Periyasamy, D. Rodriguez, M. Solazzi, A. Frisoli, M. Gustmann, J. Roßmann, U. Süss, N. G. Tsagarakis, and S. Behnke, "Remote mobile manipulation with the centauro robot: Full-body telepresence and autonomous operator assistance," *Journal of Field Robotics*, vol. 37, no. 5, pp. 889–919, 2020 [cited on page 15].

[63] M. Bjelonic, P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter. "Rolling in the Deep – Hybrid Locomotion for Wheeled-Legged Robots using Online Trajectory Optimization." [Feb. 2020] [cited on page 15].

[64] D. Jud, S. Kerscher, M. Wermelinger, E. Jelavic, P. Egli, P. Leemann, G. Hottiger, and M. Hutter, "HEAP - The autonomous walking excavator," *Automation in Construction*, vol. 129, p. 103 783, Sep. 1, 2021 [cited on page 15].

[65] T. D. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017 [cited on page 16].

[66] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ser. Studies in Computational Intelligence, A. Koubaa, Ed., Cham: Springer International Publishing, 2016, pp. 99–120 [cited on pages 16, 82].

[67] Y. Wang, N. Funk, M. Ramezani, S. Papatheodorou, M. Popović, M. Camurri, S. Leutenegger, and M. Fallon, "Elastic and Efficient LiDAR Reconstruction for Large-Scale Exploration Tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 5035–5041 [cited on page 16].

[68] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Dec. 8, 2015 [cited on page 17].

[69] J. K. Hodge, E. Marshall, and G. Patterson, "Gerrymandering and Convexity," *College Mathematics Journal*, vol. 41, no. 4, pp. 312–324, Sep. 2010 [cited on page 18].

[70] D. Wisth, M. Camurri, and M. Fallon, "Robust Legged Robot State Estimation Using Factor Graph Optimization," *IEEE Robotics and Automation Letters*, pp. 1–1, 2019 [cited on page 18].

[71] A. W. Winkler, *Optimization-based motion planning for legged robots*, Doctoral thesis, May 14, 2018 [cited on page 20].

[72] D. N. Nenchev, A. Konno, and T. Tsujita, *Humanoid Robots: Modeling and Control*. Butterworth-Heinemann, Nov. 21, 2018, 510 pp. [cited on page 21].

[73] D. Nenchev, K. Yoshida, P. Vichitkulsawat, and M. Uchiyama, "Reaction null-space control of flexible structure mounted manipulator systems," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 6, pp. 1011–1023, Dec. 1999 [cited on page 22].

[74] M. Buss, M. Hardt, J. Kiener, M. Sobotka, M. Stelzer, O. von Stryk, and D. Wollherr, "Towards an Autonomous, Humanoid, and Dynamically Walking Robot: Modeling, Optimal Trajectory Planning, Hardware Architecture, and Experiments," presented at the Proceedings of the 3rd International Conference on Humanoid Robots, 2003 [cited on pages 22, 115].

[75] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, May 2009 [cited on pages 22, 115].

[76] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, Aug. 1, 2013 [cited on pages 22, 115].

[77] D. Orin, A. Goswami, and S. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, pp. 161–176, 2013 [cited on pages 23, 76, 115].

[78] R. Budhiraja, J. Carpentier, and N. Mansard, "Dynamics Consensus between Centroidal and Whole-Body Models for Locomotion of Legged Robots," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6727–6733 [cited on pages 23, 115].

[79] D. E. Orin and A. Goswami, "Centroidal Momentum Matrix of a humanoid robot: Structure and properties," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 653–659 [cited on page 23].

[80] 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 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, Oct. 2003, 1644–1650 vol.2 [cited on page 23].

[81] P. M. Wensing and D. E. Orin, "Improved Computation of the Humanoid Centroidal Dynamics and Application for Whole-Body Control," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1 550 039, Mar. 2016 [cited on page 23].

[82] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, Nov. 1, 2012 [cited on page 23].

[83] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Madrid: IEEE, Nov. 2014, pp. 295–302 [cited on pages 23, 40, 76, 115].

[84] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 2703–2710 [cited on pages 23, 115].

[85] A. W. Winkler, "Optimization-based motion planning for legged robots," Doctoral Thesis, ETH Zurich, May 14, 2018 [cited on page 24].

[86] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, Jul. 2018 [cited on pages 24, 38, 39, 44, 50, 53–56, 74, 76–78, 105, 115, 116, 140, 152].

[87] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 4102–4109 [cited on pages 24, 115].

[88] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3D linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 1, Oct. 2001, 239–246 vol.1 [cited on page 24].

[89] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st. Springer Publishing Company, Incorporated, 2008, 632 pp. [cited on page 25].

[90] R. Full and D. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, Dec. 1, 1999 [cited on page 25].

[91] M. Raibert, *Legged Robots That Balance*, red. by D. G. Bobrow, M. Brady, R. Davis, and P. H. Winston, ser. Artificial Intelligence Series. Cambridge, MA, USA: MIT Press, Mar. 26, 1986, 250 pp. [cited on page 25].

[92] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, vol. 22, no. 11, pp. 1217–1227, Jan. 1, 1989 [cited on pages 25, 114].

[93] S. Kajita, T. Yamaura, and A. Kobayashi, "Dynamic walking control of a biped robot along a potential energy conserving orbit," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 4, pp. 431–438, Aug. 1992 [cited on page 25].

[94] E. Westervelt, J. Grizzle, and D. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 42–56, Jan. 2003 [cited on page 25].

[95] 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 *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, 1620–1626 vol.2 [cited on pages 25, 55, 114].

[96] S. Caron, Q.-C. Pham, and Y. Nakamura, "ZMP Support Areas for Multicontact Mobility Under Frictional Constraints," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 67–80, Feb. 2017 [cited on page 26].

[97] H. Dai and R. Tedrake, "Optimizing robust limit cycles for legged locomotion on unknown terrain," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec. 2012, pp. 1207–1213 [cited on page 26].

[98] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, Robotics and Neuroscience, vol. 21, no. 4, pp. 642–653, May 1, 2008 [cited on page 26].

[99] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1, 1959 [cited on pages 26, 27].

[100] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Jul. 1968 [cited on page 26].

[101] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artificial Intelligence*, vol. 31, no. 3, pp. 295–353, Mar. 1, 1987 [cited on page 26].

[102] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," *IEEE Access*, vol. 2, pp. 56–77, 2014 [cited on page 26].

[103] S. LaValle, *Rapidly-exploring random trees: A new tool for path planning*, TR 98-11, Computer Sci- ence Dept., Iowa State Univ., Oct. 1998 [cited on page 26].

[104] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996 [cited on page 26].

[105] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 1, 2001 [cited on page 26].

[106] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, Apr. 2000, 995–1001 vol.2 [cited on page 27].

[107] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-Based Methods for Motion Planning with Constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018 [cited on page 27].

[108] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 7681–7687 [cited on page 27].

[109] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5041–5047 [cited on page 27].

[110] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3067–3074 [cited on page 27].

[111] M. P. Strub and J. D. Gammell, "AIT* and EIT*: Asymmetric bidirectional sampling-based path planning," Nov. 2, 2021 [cited on page 27].

[112] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, Feb. 1, 2011 [cited on page 27].

[113] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain:" *The International Journal of Robotics Research*, Nov. 2010 [cited on pages 27, 48].

[114] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 167–172 [cited on page 27].

[115] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5148–5154 [cited on pages 27, 40].

[116] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 489–494 [cited on page 28].

[117] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574 [cited on page 28].

[118] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, Aug. 1, 2014 [cited on page 28].

[119] M. Toussaint, "Newton methods for k-order Markov Constrained Motion Problems," arXiv, arXiv:1407.0414, Jul. 1, 2014 [cited on page 28].

[120] M. Mukadam, X. Yan, and B. Boots, "Gaussian Process Motion planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 9–15 [cited on page 28].

[121] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs," in *Robotics: Science and Systems XII*, Robotics: Science and Systems Foundation, 2016 [cited on page 28].

[122] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in Motion Optimization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4202–4209 [cited on page 28].

[123] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, Mar. 1, 1998 [cited on page 29].

[124] M. Diehl, H. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast Motions in Biomechanics and Robotics*, M. Diehl and K. Mombaur, Eds., vol. 340, Berlin, Heidelberg: Springer, Berlin Heidelberg, 2006, pp. 65–93 [cited on pages 29, 50, 54].

[125] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Intl. J. of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014 [cited on pages 29, 54, 76].

[126] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, Jan. 1, 2017 [cited on pages 29, 31].

[127] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, ser. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, Jan. 1995, 618 pp. [cited on page 29].

[128] R. E. Bellman, *Dynamic Programming*. Princeton University Press, Aug. 25, 2021 [cited on page 29].

[129] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. Society for Industrial and Applied Mathematics, Jan. 2010 [cited on pages 30, 54].

[130] D. Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, Jan. 1, 1966 [cited on pages 30, 33].

[131] Robotics Today, director, *"Recent Progress on Atlas, the World's Most Dynamic Humanoid Robot" - Scott Kuindersma*, Jun. 30, 2020 [cited on page 31].

[132] M. P. Kelly, "Transcription Methods for Trajectory Optimization: A beginners tutorial," Jul. 2, 2017 [cited on page 32].

[133] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. American Elsevier Publishing Company, 1970, 232 pp. [cited on page 33].

[134] Weiwei Li and Emanuel Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement System," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*, Setúbal, Portugal: SciTePress - Science and and Technology Publications, 2004, pp. 222–229 [cited on page 33].

[135] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.*, Jun. 2005, 300–306 vol. 1 [cited on page 33].

[136] T. C. Lin and J. S. Arora, "Differential dynamic programming technique for constrained optimal control," *Computational Mechanics*, vol. 9, no. 1, pp. 41–53, Jan. 1, 1991 [cited on page 33].

[137] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1168–1175 [cited on pages 33, 34].

[138] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 695–702 [cited on page 33].

[139] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 61–66 [cited on page 34].

[140] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 7674–7679 [cited on page 34].

[141] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5674–5680 [cited on page 34].

[142] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 577–584 [cited on pages 34, 41, 76].

[143] A. Sideris and J. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," in *Proceedings of the 2005, American Control Conference, 2005.*, Jun. 2005, 2275–2280 vol. 4 [cited on pages 34, 41].

[144] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9 [cited on page 34].

[145] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2536–2542 [cited on pages 34, 47, 76, 77, 95, 116, 153].

[146] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Sola, N. Mansard, and S. Vijayakumar, "A Direct-Indirect Hybridization Approach to Control-Limited DDP," Oct. 1, 2020 [cited on pages 34, 143].

[147] J. N. Nganga and P. M. Wensing, "Accelerating Second-Order Differential Dynamic Programming for Rigid-Body Systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7659–7666, Oct. 2021 [cited on page 34].

[148] B. Plancher and S. Kuindersma, "Realtime Model Predictive Control using Parallel DDP on a GPU," presented at the International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada: IEEE, May 2019, p. 2 [cited on page 34].

[149] B. Plancher and S. Kuindersma, "A Performance Analysis of Parallel Differential Dynamic Programming on a GPU," in *Algorithmic Foundations of Robotics XIII*, ser. Springer Proceedings in Advanced Robotics, M. Morales, L. Tapia, G. Sánchez-Ante, and S. Hutchinson, Eds., vol. 14, Cham: Springer International Publishing, 2020, pp. 656–672 [cited on page 34].

[150] B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, and V. J. Reddi, "Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA," *IEEE Robotics and Automation Letters*, pp. 1–1, 2021 [cited on page 34].

[151] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, Jan. 2019, pp. 614–619 [cited on pages 34, 116, 130, 153].

[152] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, 1:1–1:37, Oct. 27, 2014 [cited on page 34].

[153] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, 43:1–43:8, Jul. 1, 2012 [cited on pages 35, 38, 54, 76].

[154] M. Posa and R. Tedrake, "Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact," in *Algorithmic Foundations of Robotics X*, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds., ser. Springer Tracts in Advanced Robotics, Berlin, Heidelberg: Springer, 2013, pp. 527–542 [cited on page 35].

[155] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1366–1373 [cited on page 35].

[156] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, Dec. 11, 2006, 686 pp. [cited on page 36].

[157] S. Mehrotra, "On the Implementation of a Primal-Dual Interior Point Method," *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 575–601, Nov. 1992 [cited on page 36].

[158] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, Jan. 2002 [cited on page 37].

[159] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006 [cited on pages 37, 50, 58, 87].

[160] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An Interior Point Algorithm for Large-Scale Nonlinear Programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, Jan. 1999 [cited on page 37].

[161] H. Mittelmann. "Benchmarks for Optimization Software: AMPL-NLP Benchmark." [], [Online]. Available: `http://plato.asu.edu/bench.html` [cited on page 37].

[162] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, "MUMPS: A General Purpose Distributed Memory Sparse Solver," in *Applied Parallel Computing. New Paradigms for HPC in Industry and Academia*, T. Sørevik, F. Manne, A. H. Gebremedhin, and R. Moe, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2001, pp. 121–130 [cited on page 37].

[163] O. Schenk, K. Gärtner, W. Fichtner, and A. Stricker, "PARDISO: A high-performance serial and parallel sparse linear solver in semiconductor device simulation," *Future Generation Computer Systems*, I. High Performance Numerical Methods and Applications. II. Performance Data Mining: Automated Diagnosis, Adaption, and Optimization, vol. 18, no. 1, pp. 69–78, Sep. 1, 2001 [cited on page 37].

[164] I. S. Duff, "MA57—a code for the solution of sparse symmetric definite and indefinite systems," *ACM Transactions on Mathematical Software*, vol. 30, no. 2, pp. 118–144, Jun. 1, 2004 [cited on page 37].

[165] B. Tasseff, C. Coffrin, A. Wächter, and C. Laird, *Exploring Benefits of Linear Solver Parallelism on Modern Nonlinear Optimization Applications*, Sep. 17, 2019 [cited on pages 37, 153].

[166] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Inverse Dynamics vs. Forward Dynamics in Direct Transcription Formulations for Trajectory Optimization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 12 752–12 758 [cited on pages 37, 153].

[167] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1071–1076 [cited on pages 38, 54].

[168] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 4914–4919 [cited on pages 38, 54].

[169] V. S. Medeiros, E. Jelavic, M. Bjelonic, R. Siegwart, M. A. Meggiolaro, and M. Hutter, "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Driving in Challenging Terrain," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4172–4179, Jul. 2020 [cited on page 38].

[170] E. Jelavic, Y. Berdou, D. Jud, S. Kerscher, and M. Hutter, "Terrain-Adaptive Planning and Control of Complex Motions for Walking Excavators," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 2684–2691 [cited on page 38].

[171] E. Jelavic, F. Farshidian, and M. Hutter, "Combined Sampling and Optimization Based Planning for Legged-Wheeled Robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8366–8372 [cited on page 38].

[172] A. Herzog, *Optimization-based motion generation for multiped robots in contact scenarios*, Doctoral thesis, Oct. 2, 2017 [cited on page 38].

[173] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 1, 2000 [cited on page 39].

[174] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 3555–3561 [cited on pages 40, 54, 55].

[175] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1096–1103 [cited on pages 40, 54].

[176] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic Locomotion Through Online Nonlinear Motion Optimization for Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2261–2268, Jul. 2018 [cited on pages 40, 43, 54].

[177] B. Ponton, A. Herzog, S. Schaal, and L. Righetti, "A convex model of humanoid momentum dynamics for multi-contact motion generation," in *IEEE/RSJ Int. Conf. on Humanoid Robots*, Cancun, Mexico, Nov. 2016, pp. 842–849 [cited on pages 40, 54, 55].

[178] M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini, "High-slope terrain locomotion for torque-controlled quadruped robots," *Autonomous Robots*, vol. 41, no. 1, pp. 259–272, Jan. 1, 2017 [cited on pages 40, 49].

[179] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter, "Dynamic locomotion and whole-body control for quadrupedal robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3359–3365 [cited on page 40].

[180] S. Caron and Q.-C. Pham, "When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov. 2017, pp. 522–528 [cited on page 40].

[181] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9 [cited on pages 40, 115].

[182] G. Bledt and S. Kim, "Implementing Regularized Predictive Control for Simultaneous Real-Time Footstep and Ground Reaction Force Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6316–6323 [cited on pages 40, 76, 80, 94].

[183] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3346–3351 [cited on page 40].

[184] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, "Perception-less terrain adaptation through whole body control and hierarchical optimization," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 558–564 [cited on pages 41, 43, 49, 51, 65, 82, 140].

[185] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018 [cited on page 41].

[186] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for Torque-Controlled Legged Robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 4730–4737 [cited on page 41].

[187] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-Aware Model Predictive Control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, Apr. 2019 [cited on pages 41, 55].

[188] C. Mastalli, W. Merkt, G. Xin, J. Shim, M. Mistry, I. Havoutis, and S. Vijayakumar, "Agile Maneuvers in Legged Robots: A Predictive Control Approach," arXiv, arXiv:2203.07554, Mar. 14, 2022 [cited on pages 41, 48, 137].

[189] S. Schaal, "Learning from demonstration," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, ser. NIPS'96, Cambridge, MA, USA: MIT Press, Dec. 3, 1996, pp. 1040–1046 [cited on page 42].

[190] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot Programming by Demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., Berlin, Heidelberg: Springer, 2008, pp. 1371–1394 [cited on page 42].

[191] R. S. Sutton, A. G. Barto, C.-D. A. L. L. A. G. Barto, and F. Bach, *Reinforcement Learning: An Introduction*. MIT Press, 1998, 356 pp. [cited on page 42].

[192] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," arXiv, arXiv:1312.5602, Dec. 19, 2013 [cited on page 42].

[193] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 7587 Jan. 2016 [cited on page 42].

[194] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, Mar. 2017 [cited on page 42].

[195] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 1, 2015, pp. 1889–1897 [cited on page 42].

[196] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv, arXiv:1707.06347, Aug. 28, 2017 [cited on page 42].

[197] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, Jul. 10–15, 2018, pp. 1861–1870 [cited on page 43].

[198] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne. "ALLSTEPS: Curriculum-Driven Learning of Stepping Stone Skills." [Aug. 2020] [cited on pages 43, 95].

[199] Z. Yin, Z. Yang, M. Van De Panne, and K. Yin, "Discovering diverse athletic jumping strategies," *ACM Transactions on Graphics*, vol. 40, no. 4, 91:1–91:17, Jul. 19, 2021 [cited on page 43].

[200] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30 [cited on page 43].

[201] E. Valassakis, Z. Ding, and E. Johns, "Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5372–5379 [cited on page 43].

[202] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, eaau5872, Jan. 16, 2019 [cited on page 43].

[203] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, eabc5986, Oct. 21, 2020 [cited on page 43].

[204] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Robotics: Science and Systems XVII*, Robotics: Science and Systems Foundation, Jul. 12, 2021 [cited on page 43].

[205] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," arXiv, arXiv:2109.11978, Oct. 30, 2021 [cited on page 43].

[206] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter. "DeepGait: Planning and Control of Quadrupedal Gaits using Deep Reinforcement Learning." [Sep. 18, 2019], [Online]. Available: `http://arxiv.org/abs/1909.08399` [cited on page 43].

[207] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, 2022 [cited on page 43].

[208] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, eabk2822, Jan. 19, 2022 [cited on page 44].

[209] R. E. Kalman, "When Is a Linear Control System Optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, Mar. 1, 1964 [cited on page 44].

[210] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 29, 2000, pp. 663–670 [cited on page 44].

[211] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, 143:1–143:14, Jul. 30, 2018 [cited on page 44].

[212] P. Brakel, S. Bohez, L. Hasenclever, N. Heess, and K. Bousmalis, "Learning Coordinated Terrain-Adaptive Locomotion by Imitating a Centroidal Dynamics Planner," arXiv, arXiv:2111.00262, Oct. 30, 2021 [cited on pages 44, 144].

[213] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016 [cited on page 44].

[214] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "AMP: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics*, vol. 40, no. 4, 144:1–144:20, Jul. 19, 2021 [cited on page 44].

[215] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning," arXiv, arXiv:2203.14912, Mar. 23, 2022 [cited on pages 44, 144].

[216] M. Stolle and C. Atkeson, "Policies based on trajectory libraries," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 3344–3349 [cited on page 45].

[217] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, Oct. 2007, pp. 2981–2986 [cited on pages 45, 94].

[218] N. Jetchev and M. Toussaint, "Fast motion planning from experience: Trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, no. 1, pp. 111–127, Jan. 2013 [cited on pages 45, 92, 94].

[219] N. Jetchev and M. Toussaint, "Trajectory prediction: Learning to map situations to robot trajectories," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09, New York, NY, USA: Association for Computing Machinery, Jun. 14, 2009, pp. 449–456 [cited on page 45].

[220] A. Dragan, G. J. Gordon, and S. Srinivasa, "Learning from experience in manipulation planning: Setting the right goals," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, Springer, Jul. 2011 [cited on pages 45, 94].

[221] S. Levine. "Exploring Deep and Recurrent Architectures for Optimal Control." [Nov. 7, 2013], [Online]. Available: `http://arxiv.org/abs/1311.1761` [cited on page 45].

[222] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural Path Planning: Fixed Time, Near-Optimal Path Generation via Oracle Imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3965–3972 [cited on page 46].

[223] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1327–1339, Oct. 2012 [cited on page 46].

[224] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Aug. 2018 [cited on pages 46, 95].

[225] S. Starke, Y. Zhao, T. Komura, and K. Zaman, "Local motion phases for learning multi-contact character movements," *ACM Transactions on Graphics*, vol. 39, no. 4, 54:54:1–54:54:13, Jul. 2020 [cited on pages 46, 95].

[226] S. Hong, D. Han, K. Cho, J. S. Shin, and J. Noh, "Physics-based full-body soccer motion control for dribbling and shooting," *ACM Transactions on Graphics*, vol. 38, no. 4, 74:1–74:12, Jul. 2019 [cited on page 46].

[227] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of Motion for Warm-Starting Trajectory Optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, Apr. 2020 [cited on pages 46, 47, 94].

[228] J. Carius, F. Farshidian, and M. Hutter, "MPC-Net: A First Principles Guided Policy Search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, Apr. 2020 [cited on pages 46, 95].

[229] G. Tang and K. Hauser, "Discontinuity-Sensitive Optimal Control Learning by Mixture of Experts," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7892–7898 [cited on page 46].

[230] W. X. Merkt, V. Ivan, T. Dinev, I. Havoutis, and S. Vijayakumar, "Memory Clustering Using Persistent Homology for Multimodality- and Discontinuity-Sensitive Learning of Optimal Control Warm-Starts," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1649–1660, Oct. 2021 [cited on page 46].

[231] K. Sohn, H. Lee, and X. Yan, "Learning Structured Output Representation using Deep Conditional Generative Models," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015 [cited on pages 47, 101].

[232] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-Conditioned Variational Autoencoders for Learning Movement Primitives," in *Conference on Robot Learning*, May 2020, pp. 933–944 [cited on page 47].

[233] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning Latent Plans from Play," in *Conference on Robot Learning*, May 2020, pp. 1113–1132 [cited on pages 47, 92, 95, 101, 102].

[234] Y. Tassa, T. Erez, and W. Smart, "Receding Horizon Differential Dynamic Programming," in *Advances in Neural Information Processing Systems*, vol. 20, Curran Associates, Inc., 2007 [cited on page 47].

[235] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*, St Paul, MN, USA: IEEE, May 2012, pp. 3671–3678 [cited on pages 47, 94].

[236] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD: IEEE, May 2018, pp. 2986–2993 [cited on pages 47, 92, 94, 95].

[237] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 5877–5884 [cited on pages 47, 94].

[238] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, R. Berka, R. Ambrose, M. Myles Markee, N. J. Fraser-Chanpong, C. McQuin, J. D. Yamokoski, S. Hart, R. Guo, A. Parsons, B. Wightman, P. Dinh, B. Ames, C. Blakely, C. Edmondson, B. Sommers, R. Rea, C. Tobler, H. Bibby, B. Howard, L. Niu, A. Lee, M. Conover, L. Truong, R. Reed, D. Chesney, R. Platt Jr, G. Johnson, C.-L. Fok, N. Paine, L. Sentis, E. Cousineau, R. Sinnet, J. Lack, M. Powell, B. Morris, A. Ames, and J. Akinyode, "Valkyrie: NASA's First Bipedal Humanoid Robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015 [cited on page 47].

[239] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard, and S. Calinon. "Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion." [Feb. 28, 2020], [Online]. Available: `http://arxiv.org/abs/2001.11751` [cited on pages 47, 92, 95].

[240] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An Efficient Acyclic Contact Planner for Multiped Robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, Jun. 2018 [cited on pages 47, 95].

[241] G. Bledt and S. Kim, "Extracting Legged Locomotion Heuristics with Regularized Predictive Control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 406–412 [cited on pages 47, 80, 94].

[242] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on HRP-2," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 2974–2979 [cited on page 48].

[243] P. D. Neuhaus, J. E. Pratt, and M. J. Johnson, "Comprehensive summary of the Institute for Human and Machine Cognition's experience with LittleDog," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 216–235, Feb. 1, 2011 [cited on page 48].

[244] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *Intl. J. of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001 [cited on page 48].

[245] T. Boaventura, C. Semini, J. Buchli, M. Frigerio, M. Focchi, and D. G. Caldwell, "Dynamic torque control of a hydraulic quadruped robot," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 1889–1894 [cited on page 48].

[246] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3287–3292 [cited on page 48].

[247] G. Chen, S. Guo, B. Hou, and J. Wang, "Virtual Model Control for Quadruped Robots," *IEEE Access*, vol. 8, pp. 140 736–140 751, 2020 [cited on page 48].

[248] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2010, pp. 3406–3412 [cited on pages 48, 49].

[249] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Inverse dynamics control of floating-base robots with external constraints: A unified view," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1085–1090 [cited on page 48].

[250] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 02, no. 04, pp. 505–518, Dec. 2005 [cited on page 48].

[251] L. Sentis, J. 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, Jun. 2010 [cited on page 48].

[252] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 814–820 [cited on page 49].

[253] F. Farshidian, E. Jelavić, A. W. Winkler, and J. Buchli, "Robust whole-body motion control of legged robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 4589–4596 [cited on page 49].

[254] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive Whole-Body Control for Quadruped Robots: Experimental Validation over Challenging Terrain," Nov. 2018 [cited on page 49].

[255] G. Xin, H. Lin, J. Smith, O. Cebe, and M. Mistry, "A Model-Based Hierarchical Controller for Legged Systems Subject to External Disturbances," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018-05, 2018, pp. 4375–4382 [cited on page 49].

[256] M. Hutter, H. Sommer, C. Gehring, M. Hoepflinger, M. Bloesch, and R. Siegwart, "Quadrupedal locomotion using hierarchical operational space control," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1047–1062, Jul. 1, 2014 [cited on page 49].

[257] N. Mansard, "A dedicated solver for fast operational-space inverse dynamics," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 4943–4949 [cited on page 49].

[258] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, Jul. 1, 1985 [cited on page 55].

[259] Y. Uno, M. Kawato, and R. Suzuki, "Formation and control of optimal trajectory in human multijoint arm movement," *Biological Cybernetics*, vol. 61, no. 2, pp. 89–101, Jun. 1, 1989 [cited on page 55].

[260] C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature*, vol. 394, no. 6695, pp. 780–784, 6695 Aug. 1998 [cited on page 55].

[261] J. Hwangbo, J. Lee, and M. Hutter, "Per-Contact Iteration Method for Solving Contact Dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, Apr. 2018 [cited on pages 65, 67].

[262] M. Bloesch, M. Burri, H. Sommer, R. Siegwart, and M. Hutter, "The Two-State Implicit Filter Recursive Estimation for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 573–580, Jan. 2018 [cited on page 68].

[263] J. Carpentier and N. Mansard, "Multi-contact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, Dec. 2018 [cited on page 76].

[264] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5776–5782 [cited on page 76].

[265] M. Vukobratovic and B. Borovac, "Zero-Moment Point — Thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004 [cited on page 76].

[266] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9 [cited on page 76].

[267] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust Rough-Terrain Locomotion with a Quadrupedal Robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5761–5768 [cited on page 76].

[268] D. Kim, D. Carballo, J. D. Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision Aided Dynamic Exploration of Unstructured Terrain with a Small-Scale Quadruped Robot," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2464–2470 [cited on page 76].

[269] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter, "Perceptive Locomotion in Rough Terrain – Online Foothold Optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5370–5376, Oct. 2020 [cited on page 76].

[270] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-Based Controller with Terrain Insight for Dynamic Legged Locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2436–2442 [cited on pages 76, 77, 115].

[271] A. Herdt, N. Perrin, and P. Wieber, "Walking without thinking about it," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 190–195 [cited on page 76].

[272] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–17, Jan. 2017 [cited on page 76].

[273] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5308–5313 [cited on page 76].

[274] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, *TAMOLS: Terrain-Aware Motion Optimization for Legged Systems*, Jul. 5, 2022 [cited on page 76].

[275] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast Online Trajectory Optimization for the Bipedal Robot Cassie," in *Robotics: Science and Systems XIV*, vol. 14, Jun. 2018 [cited on page 77].

[276] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2018 [cited on page 77].

[277] O. Cebe, C. Tiseo, G. Xin, H.-c. Lin, J. Smith, and M. Mistry. "Online Dynamic Trajectory Optimization and Control for a Quadruped Robot." [Sep. 2020] [cited on page 77].

[278] T. Bretl and S. Lall, "Testing Static Equilibrium for Legged Robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, Aug. 2008 [cited on page 84].

[279] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, D. G. Caldwell, and C. Semini, "Feasible Region: An Actuation-Aware Extension of the Support Region," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1239–1255, Aug. 2020 [cited on page 84].

[280] A. Abdalla, M. Focchi, R. Orsolino, and C. Semini, *An Efficient Paradigm for Feasibility Guarantees in Legged Locomotion*, Nov. 16, 2020 [cited on page 84].

[281] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture Point: A Step toward Humanoid Push Recovery," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, Dec. 2006, pp. 200–207 [cited on page 84].

[282] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2011, pp. 3719–3726 [cited on page 94].

[283] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, Apr. 2020 [cited on page 95].

[284] A. Duburcq, Y. Chevaleyre, N. Bredeche, and G. Boéris, "Online Trajectory Planning Through Combined Trajectory Optimization and Function Approximation: Application to the Exoskeleton Atalante," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 3756–3762 [cited on page 95].

[285] S. Levine and P. Abbeel, "Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014 [cited on page 97].

[286] B. Mazoure, T. Doan, A. Durand, J. Pineau, and R. D. Hjelm, "Leveraging exploration in off-policy algorithms via normalizing flows," p. 15, [cited on page 102].

[287] J. Ahn, S. J. Jorgensen, S. H. Bang, and L. Sentis, "Versatile Locomotion Planning and Control for Humanoid Robots," *Frontiers in Robotics and AI*, vol. 8, p. 257, 2021 [cited on page 115].

[288] K. Wang, G. Xin, S. Xin, M. Mistry, S. Vijayakumar, and P. Kormushev, "A Unified Model with Inertia Shaping for Highly Dynamic Jumps of Legged Robots," Sep. 9, 2021 [cited on pages 115, 137].

[289] Z. Zhou, B. Wingo, N. Boyd, S. Hutchinson, and Y. Zhao, "Momentum-Aware Trajectory Optimization and Control for Agile Quadrupedal Locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7755–7762, Jul. 2022 [cited on page 116].

[290] S.-H. Lee and A. Goswami, "Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 4667–4672 [cited on pages 116, 119].

[291] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 6, 2009 [cited on page 116].

[292] S. Zhao, *Time Derivative of Rotation Matrices: A Tutorial*, Sep. 20, 2016 [cited on page 118].

[293] G. Ficht and S. Behnke, "Fast Whole-Body Motion Control of Humanoid Robots with Inertia Constraints," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 6597–6603 [cited on page 137].

[294] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, "SymPy: Symbolic computing in Python," *PeerJ Computer Science*, vol. 3, e103, Jan. 2, 2017 [cited on page 138].

[295] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 1, 2019 [cited on page 138].

[296] C. C. Margossian, "A review of automatic differentiation and its efficient implementation," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, e1305, 2019 [cited on page 138].

[297] M. Tranzatto, F. Mascarich, L. Bernreiter, C. Godinho, M. Camurri, S. Khattak, T. Dang, V. Reijgwart, J. Löje, D. Wisth, S. Zimmermann, H. Nguyen, M. Fehr, L. Solanka, R. Buchanan, M. Bjelonic, N. Khedekar, M. Valceschini, F. Jenelten, M. Dharmadhikari, T. Homberger, P. De Petris, L. Wellhausen, M. Kulkarni, T. Miki, S. Hirsch, M. Montenegro, C. Papachristos, F. Tresoldi, J. Carius, G. Valsecchi, J. Lee, K. Meyer, X. Wu, J. Nieto, A. Smith, M. Hutter, R. Siegwart, M. Mueller, M. Fallon, and K. Alexis, "CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge," *Field Robotics*, vol. 2, no. 1, pp. 274–324, Mar. 10, 2022 [cited on page 145].

[298] L. Schmid, V. Reijgwart, L. Ott, J. Nieto, R. Siegwart, and C. Cadena, "A Unified Approach for Autonomous Volumetric Exploration of Large Scale Environments Under Severe Odometry Drift," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4504–4511, Jul. 2021 [cited on page 145].

[299] S. Scherer, "Resilient and Modular Subterranean Exploration with a Team of Roving and Flying Robots," *Field Robotics*, p. 57, 2022 [cited on page 145].