Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

# Enriching the Capabilities of Legged Robots with Libraries of Motions

**Sotirios Apostolopoulos**

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**

genehmigten Dissertation.

| | | |
|---|---|---|
| Vorsitzender: | | Prof. Dr.-Ing. Werner Hemmert |
| Prüfer der Dissertation: | | |
| | 1. | Prof. Dr.-Ing. Martin Buss |
| | 2. | Prof. Dr. Oskar von Stryk |

Die Dissertation wurde am 02.12.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 14.12.2021 angenommen.

# Foreword

This thesis summarizes my research at the Chair of Automatic Control Engineering at the Technical University of Munich. This thesis was not only an opportunity to explore new research areas, but also a chance to interact with dedicated and highly knowledgeable researchers.

I am grateful to Prof. Martin Buss who offered me the possibility to come to Germany and work on the field of Motion Primitives and Legged Robots. His assistance and insight helped me overcome research challenges and contribute in the fields of my studies and finally write this thesis. In addition, I want to thank Dr. Marion Leibold for sharing the same passion for walking robots and introducing me to new and interesting topics. She was always there for me to discuss the state of my research and proof read my publications. Furthermore, both gave me the opportunity to teach students which was a valuable experience for me.

I would also like to thank my colleagues in the SHRINE project for all our fruitful discussions and exchange of ideas. These meetings allowed me to become familiar with different and diverse research topics. Besides the academic exchange, I found friends in this team which I have kept also after leaving TUM and Munich: Michaela Semmler, my office-mate Stefan Kersting, Stefan Friedrich, Philine Donner, Alexander Pekarovskiy and Markus Schill.

Finally, I thank my parents, Andreas and Ermioni , my sister Vicky, my beloved grandparents Costas and Vasiliki and my uncle and aunt Dimitris and Christina for their love all these years. They were there for me when I needed them and always believed in me. Their encouragement and support allowed me to overcome difficulties and finally write this thesis.

Asperg, December 2019 Sotirios Apostolopoulos

## Abstract

Designing motions for legged robots is a computationally challenging problem due to the fact that their mathematical models are expressed as highly coupled nonlinear systems of equations subject to unilateral constraints. In order to counter this shortcoming this thesis introduces the utilization of databases/libraries of Motion Primitives. These motions are created offline through an optimization process and are later pre-processed in order to allow for online computations while retaining a degree of optimality. The feasibility and advantages of this approach are validated on different application scenarios.

At first, we tackle the challenge of online motion planning for walking robots with point feet on uneven terrain. For that, this thesis introduced an online motion planning algorithm and a motion generation methodology which uses a database of Motion Primitives as training examples for a Gaussian Process Regression. The regression is used when there is no match between the terrain variation and the Motion Primitives in the database. The key points that allowed our approach to be utilized online is the fact that the motion planning algorithm is based on a best first graph search approach and that the regression has a small inference time. Finally, the dynamic feasibility of the motions generated by the regression methodology is checked in the zero dynamics of the robot.

As a next step, the thesis introduces a novel way to improve the settling time of transitions between different periodic walking motions of an underactuated robot. The slow convergence is credited to the unactuated degree of freedom which prevents the state of the system from entering the domain of attraction of the target orbit close to the fixed point of the Poincaré Map. At first the problem is introduced with the help of the Hybrid Zero Dynamics framework and Optimal Control. Later a Markov Decision Process is formulated and solved with Reinforcement Learning in order to learn multi-step transitions that reduce the settling time. The experimental results suggest that the proposed methodology performs better than a one-step transition for 84.34% of all the considered transitions for a simulated walking robot.

The final contribution and proof of concept comes from the field of balancing, where motions are usually designed using simplified models of the Center of Mass and feedback control without accounting for energy efficiency. For that, a Motion Primitive switching methodology is introduced where samples of optimal motions are chosen online based on a Euclidean distance metric. The chosen sample is used to provide the reference trajectories, torques and Ground Reaction Forces to be tracked. In order to satisfy all of the modeling assumptions while tracking the reference values, a Quadratic Program is solved online where the dynamics of the robot, friction cone, Center of Pressure and torque and state bounds are treated as constraints. Convergence to the desired trajectories is dictated by a Control Lyapunov Function constraint which is introduced in the Quadratic Program. The methodology is evaluated on a four-link simulated robot where is shown that switching between Motion Primitives provides energy efficient balancing motions for different disturbance situations. At the same time the methodology provides more efficient motions for different disturbance forces when compared to a non-switching approach, where a Motion Primitive is chosen only once based on the post-impact robot state.

## Zusammenfassung

Der Entwurf von Bewegungen für Roboter mit Beinen ist ein rechnerisch anspruchsvolles Problem, da ihre mathematischen Modelle als hoch gekoppelte nichtlineare Gleichungssysteme mit einseitigen Nebenbedingungen ausgedrückt werden. Um diesem Mangel entgegenzuwirken, wird in dieser Promotion die Verwendung von Bewegungsprimitiven Datenbanken/Bibliotheken vorgestellt. Diese Bewegungen werden offline durch einen Optimierungsprozess erzeugt und später vorverarbeitet, um Online-Berechnungen unter Beibehaltung einer gewissen Optimalität zu ermöglichen. Die Machbarkeit und die Vorteile dieses Ansatzes werden an verschiedenen Anwendungsszenarien validiert.

Zuerst stellen wir uns der Herausforderung der Online-Bewegungsplanung für Laufroboter mit Spitzenfüßen auf unebenem Gelände. Für diese Dissertation wurde ein Online-Algorithmus zur Bewegungsplanung und eine Methodik zur Bewegungserzeugung eingeführt, die eine Datenbank mit Bewegungsprimitiven als Trainingsbeispiele für eine Gauß-Prozess-Regression verwendet. Die Regression wird verwendet, wenn keine Übereinstimmung zwischen der Geländevariation und den Bewegungsprimitiven in der Datenbank besteht. Die Schlüsselpunkte, die es uns ermöglichten, unseren Ansatz online zu nutzen, sind die Tatsache, dass der Bewegungsplanungsalgorithmus auf einem Best First Graph Suchansatz basiert und dass die Regression eine kleine Inferenzzeit hat. Schließlich wird die dynamische Machbarkeit der durch die Regressionsmethode erzeugten Bewegungen in der null Dynamik des Roboters überprüft.

Als nächster Schritt stellt die Dissertation eine neue Art zur Verbesserung der Beruhigungszeit von Übergängen zwischen verschiedenen periodischen Laufbewegungen eines unterbetätigten Roboters. Die langsame Konvergenz wird dem unbetätigten Freiheitsgrad zugeschrieben, der verhindert, dass der Zustand des Systems in die Anziehungsdomäne des Zielumlaufbahns nahe dem Fixpunkt der Poincaré-Karte gelangt. Zuerst wird das Problem mit Hilfe des Hybriden Nulldynamischen Rahmens und Optimaler Steuerung vorgestellt. Später wird ein Markov-Entscheidungsprozess formuliert und mit Reinforcement Learning gelöst, um mehrstufige Übergänge zu lernen, die die Beruhigungszeit verkürzen. Die experimentellen Ergebnisse deuten darauf hin, dass die vorgeschlagene Methodik besser funktioniert als ein einstufiger Übergang für 84,34% aller betrachteten Übergänge für einen simulierten Laufroboter.

Der letzte Beitrag und Machbarkeitsnachweis stammt aus dem Bereich des Balanzierens, wo die Bewegungen in der Regel mit vereinfachten Modellen des Massenschwerpunkts und Rückführung ohne Berücksichtigung der Energieeffizienz gestaltet werden. Für diese Bewegung wird eine Schaltmethode von Primitiven eingeführt, bei der Stichproben von optimalen Bewegungen online auf der Grundlage einer euklidischen Entfernungsmetrik ausgewählt werden. Die ausgewählte Stichprobe wird verwendet, um Referenztrajektorien, Drehmomente und Bodenreaktionskräfte bereitzustellen, die verfolgt werden sollen. Um alle Modellierungsannahmen bei der Verfolgung der Referenzwerte zu erfüllen, wird ein quadratisches Programm online gelöst, bei dem die Dynamik des Roboters, Reibungskegel, Druckmittelpunkt und Drehmoment- und Zustandsgrenzen als Nebenbedingungen behandelt werden. Die Konvergenz zu den gewünschten Trajektorien wird durch eine Control Lyapunov Funktion Nebenbedingung bestimmt, die im Quadratischen Programm eingeführt wird. Die Methodik wird an einem Vierkörper simulierten Roboter evaluiert, wo gezeigt wird, dass das Umschalten zwischen Bewegungsprimitiven energieeffiziente Balanzierbewegungen für verschiedene Störungssituationen bietet. Gleichzeitig bietet die Methodik effizientere Bewegungen für verschiedene Störkräfte im Vergleich zu einem nicht schaltenden Ansatz, bei dem ein Motion Primitive nur einmal basierend auf dem Zustand nach dem Aufprall des Roboters ausgewählt wird.

# Contents

# Notations

## Abbreviations

| | |
|---|---|
| CoM | Center of Mass |
| ZMP | Zero Moment Point |
| CoP | Center of Pressure |
| GRF | Ground Reaction Force |
| PID | Proportional Integral Derivative |
| PD | Proportional Derivative |
| QP | Quadratic Program |
| CPG | Central Pattern Generator |
| LIP | Linear Inverted Pendulum |
| RWLIP | Reaction Wheel Linear Inverted Pendulum |
| MPC | Model Predictive Control |
| DoF | Degree of Freedom |
| HZD | Hybrid Zero Dynamics |
| VMC | Virtual Model Control |
| GP | Gaussian Process |
| ODE | Ordinary Differential Equation |
| FRI | Foot Rotation Indicator |
| OCP | Optimal Control Problem |
| DMP | Dynamical Movement Primitives |
| CTC | Computed Torque Control |
| CLF | Control Lyapunov Function |
| RES-CLF | Rapidly Exponentially Stabilizing-Control Lyapunov Function |

## Subscripts and Superscripts

| | |
|---|---|
| $\dot{z}$ | Time derivative of $z$ |
| $z^+$ | Post-impact/post-transition value of $z$ |
| $z^-$ | Pre-impact/pre-transition value of $z$ |
| $z^x$ | $x$ Cartesian coordinate of $z$ (unless stated otherwise) |
| $z^y$ | $y$ Cartesian coordinate of $z$ (unless stated otherwise) |
| $z_{\mathrm{des}}$ | Desired value of $z$ |
| $\boldsymbol{X}^{-1}$ | Inverse of $\boldsymbol{X}$ |

| | |
|---|---|
| $\boldsymbol{r}^\top$ | Transpose of $\boldsymbol{r}$ |
| $\boldsymbol{X}^\top$ | Transpose of $\boldsymbol{X}$ |
| $z_0$ | Initial value of $z$ |
| $z_T$ | Final value of $z$ |

## Number Sets

| | |
|---|---|
| $\mathbb{R}$ | Real numbers |
| $\mathbb{N}^*$ | Physical numbers without 0 |

## Symbols

### General

| | |
|---|---|
| $\boldsymbol{x}$ | System state |
| $\boldsymbol{y}$ | Vector of continuous outputs |
| $\boldsymbol{u}$ | Vector of inputs for the continuous flow of a system |
| $\boldsymbol{h}$ | Output vector function |
| $\boldsymbol{f}$ | Drift/bias term in the control affine form |
| $\boldsymbol{g}$ | Non drift/bias term in the control affine form |
| $t$ | Time |
| $\mathbf{0}_n$ | Zero square matrix of dimension $n$ |
| $\mathbf{0}_{n\times k}$ | Zero matrix of dimensions $n \times k$ |
| $\boldsymbol{I}_n$ | Identity matrix of dimension $n$ |
| $n$ | Number of DoFs |

### Modeling of Planar Legged Robots

| | |
|---|---|
| $\boldsymbol{\sigma}$ | Vector state of hybrid system |
| $\boldsymbol{d}$ | Function for the equations of motion of a hybrid system |
| $x_d$ | Discrete mode of the state of a hybrid system |
| $y_d$ | Discrete output of a hybrid system |
| $u_d$ | Discrete input of a hybrid system |
| $S_{i\to j}$ | Transition surface from discrete mode $i$ to discrete mode $j$ |
| $\Sigma$ | Set of transitions surfaces |
| $I_d$ | Index set |
| $\Theta_s$ | Configuration space of the robot |
| $\boldsymbol{\Delta}_{i\to j}$ | Transition matrix to define the hybrid state after crossing a transition surface |
| $\boldsymbol{\theta}$ | Generalized coordinates of the robot |
| $\boldsymbol{q}$ | Joint positions of the robot |
| $\boldsymbol{p}_1$ | Position for the tip of the stance leg |
| $L$ | Lagrangian of the robot system |
| $K$ | Kinetic energy of the robot system |

| | |
|---|---|
| $V$ | Potential energy of the robot system |
| $\boldsymbol{U}$ | Vector of generalized torques and forces acting on the robot |
| $\boldsymbol{D}_e$ | Mass-inertia matrix for the unconstrained dynamics |
| $\boldsymbol{C}_e$ | Matrix of centrifugal and Coriolis terms for the unconstrained dynamics |
| $\boldsymbol{G}_e$ | Vector of gravity terms for the unconstrained dynamics |
| $\boldsymbol{S}_e$ | Input matrix for the unconstrained dynamics |
| $\boldsymbol{F}$ | GRFs |
| $F_x$ | $x$ Coordinate of the GRFs |
| $F_y$ | $y$ Coordinate of the GRFs |
| $\boldsymbol{J}_1$ | Jacobian matrix for the tip of the stance leg |
| $\delta\boldsymbol{F}_2$ | Externally applied force |
| $\boldsymbol{p}_2$ | Position of the tip of the swing leg |
| $\boldsymbol{F}_2$ | Impact force applied on the tip of the swing leg |
| $\boldsymbol{J}_2$ | Jacobian matrix for the tip of the swing leg |
| $\boldsymbol{\Delta}_F$ | Auxiliary matrix for the calculation of $\boldsymbol{F}_2$ |
| $\boldsymbol{\Delta}_q$ | Auxiliary matrix for the calculation of the $\boldsymbol{\theta}^+$ |
| $\boldsymbol{R}$ | Relabelling matrix |
| $\boldsymbol{\Delta}_q^R$ | Auxiliary matrix for the calculation of $\boldsymbol{\theta}^+$ including $\boldsymbol{R}$ |
| $\boldsymbol{D}$ | Mass-inertia matrix for the constrained dynamics |
| $\boldsymbol{C}$ | Matrix of centrifugal and Coriolis terms for the constrained dynamics |
| $\boldsymbol{G}$ | Vector of gravity terms for the constrained dynamics |
| $\boldsymbol{S}$ | Input matrix for the constrained dynamics |
| $\mathcal{S}$ | Walking surface |
| $\boldsymbol{\Delta}$ | Matrix to calculate the post-impact state in the hybrid system formulation |
| $\mu_s$ | Coulomb static friction coefficient |
| $\alpha$ | Inverse tangent of the Coulomb static friction coefficient |
| $\mathcal{C}$ | Friction cone |
| $M_z$ | Contact moment |
| $p_\ell$ | Distance from the ankle to the left foot edge |
| $p_r$ | Distance from the ankle to the right foot edge |
| $y_0$ | Constant height of the CoM for the simplified models |
| $\boldsymbol{p}_{\text{cm}}$ | Position of the CoM |
| $\boldsymbol{F}_{\text{cm}}$ | Force acting on the CoM |
| $\boldsymbol{g}_{\text{cm}}$ | Gravity vector field of the CoM |
| $g$ | Gravity constant |
| $\boldsymbol{p}_{\text{ankle}}$ | Position of the ankle |
| $\boldsymbol{\tau}_z$ | Vector of the ankle torque |
| $\tau_{\text{ankle}}$ | Ankle torque |
| $\boldsymbol{v}_{\text{cm}}$ | Velocity of the CoM |
| $\boldsymbol{p}_0$ | Position of the nonlinear spring damper system for a compliant walking model |
| $\boldsymbol{p}_{\text{foot}}$ | Position of the point of first contact on the robot foot |
| $K_P^x$ | Term for the $x$ coordinate of the position error for a compliant walking model |
| $K_P^y$ | Term for the $y$ coordinate of the position error for a compliant walking model |
| $K_D^x$ | Term for the $x$ coordinate of the velocity error for a compliant walking model |

| | |
|---|---|
| $K_D^y$ | Term for the $y$ coordinate of the velocity error for a compliant walking model |

## Hybrid Zero Dynamics of Walking Robots

| | |
|---|---|
| $\boldsymbol{h}_d$ | Desired trajectories for the actuated DoFs of the walking robot |
| $\boldsymbol{T}$ | Coordinate transformation |
| $\boldsymbol{v}$ | Feedback control input |
| $z$ | Transformed coordinates for the input-output feedback linearization |
| $\xi_1$, $\theta$ | Monotonically increasing variable to replace time |
| $\boldsymbol{c}$ | Vector to define $\theta$ |
| $\xi_2$, $\gamma$, $\bar{\sigma}_n$ | Conjugate angular momentum around $q_n$ |
| $\gamma_0$ | Last line of mass-inertia matrix $\boldsymbol{D}$ |
| $\boldsymbol{A}_u$ | Auxiliary matrix to define the output zeroing control law $\boldsymbol{u}^*$ |
| $\boldsymbol{b}_u$ | Auxiliary vector to define the output zeroing control law $\boldsymbol{u}^*$ |
| $\boldsymbol{\mathcal{Z}}$ | Zero dynamics manifold |
| $\boldsymbol{K}_d$, $\boldsymbol{K}_p$ | Derivative and proportional gain matrices for $\boldsymbol{v}$ |
| $I$ | Auxiliary function to facilitate the introduction of Lagrangian dynamics |
| $\kappa_2$ | Time derivative of $\xi_2$ |
| $\boldsymbol{C}_n$ | Last row of $\boldsymbol{C}$ |
| $\boldsymbol{G}_n$ | Last row of $\boldsymbol{G}$ |
| $K_{\text{zero}}$,$V_{\text{zero}}$ | Kinetic and potential energy for the zero dynamics manifold |
| $V_{\text{zero}}^{\text{MAX}}$ | Maximum value of the potential energy for the zero dynamics manifold |
| $\delta_{\text{zero}}$ | Kinetic energy exchange at the impact in the zero dynamics |
| $\boldsymbol{\phi}_{i\to f}(t)$ | Transition motion between the stable periodic orbits $\boldsymbol{\phi}_i(t)$ and $\boldsymbol{\phi}_f(t)$ |
| $\delta_{\text{zero}}^{i\to f}$ | Kinetic energy exchange at the impact for the transition motion $\boldsymbol{\phi}_{i\to f}$ |
| $\mathcal{P}$ | Poincaré Map for the zero dynamics |
| $\xi_2^*$ | Fixed point of the Poincaré Map $\mathcal{P}$ |
| $\mathcal{D}$ | Domain of definition of the Poincaré Map $\mathcal{P}$ |
| $\boldsymbol{\phi}_i(t)$, $\boldsymbol{\phi}_f(t)$ | Initial and final stable periodic orbits |
| $V_{\text{zero},f}^{\text{MAX}}$ | Maximum value of the $V_{\text{zero},f}$ for the target periodic orbit $\boldsymbol{\phi}_f(t)$ |
| $V_{\text{zero},i\to f}^{\text{MAX}}$ | Maximum value of $V_{\text{zero},i\to f}$ for the transition motion $\boldsymbol{\phi}_{i\to f}(t)$ |
| $V_{\text{zero}}^{i\to f}$ | Zero dynamics potential energy for $\boldsymbol{\phi}_{i\to f}(t)$ |
| $\boldsymbol{q}^d$ | Desired trajectories |
| $M$ | Order of the Bézier polynomial |
| $\boldsymbol{\alpha}$ | Coefficients of the Bézier polynomial |
| $s$ | Phase variable of the Bézier polynomial |
| $\boldsymbol{\omega}^-$ | Variable to connect the first two and last two Bézier coefficients |
| $\Gamma_{\boldsymbol{\alpha}}$,$\Gamma_{\boldsymbol{\beta}}$ | Controller corresponding to the Bézier coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ |
| $\Gamma_{\boldsymbol{\alpha}\to\boldsymbol{\beta}}$ | Controller corresponding to the Bézier coefficients $\boldsymbol{\alpha} \to \boldsymbol{\beta}$ |
| $M_{\boldsymbol{\alpha}\to\boldsymbol{\beta}}$ | Order of the Bézier polynomial for the transition motion |
| $J$ | Squared sum of torques |
| $J_{\text{mech}}$ | Mechanical power per step |
| $\boldsymbol{W}$ | Weighting matrix |

## Learning with Primitives for Online Motion Generation

| | |
|---|---|
| $P$ | Motion Primitive |
| $\ell^-$ | Stride length at the end of the motion |
| $h^-$ | Stride height at the end of the motion |
| $\mathcal{GP}$ | Gaussian Process |
| $\boldsymbol{\varphi}$ | Input to the Gaussian Process |
| $m$ | Mean function of the Gaussian Process |
| $\kappa$ | Covariance function of the Gaussian Process |
| $\mathcal{N}$ | Gaussian distribution |
| $\boldsymbol{\mu}$ | Mean of the Gaussian distribution |
| $\boldsymbol{\mathcal{K}}$ | Covariance matrix of the Gaussian distribution |
| $\mathbb{E}$ | Expected value |
| cov | Covariance |
| $\sigma_f^2$ | Length scale of the squared exponential covariance kernel |
| $\boldsymbol{\Sigma}$ | Covariance matrix of squared exponential covariance kernel |
| $\sigma_\varphi^2$ | Standard deviation of the squared exponential covariance kernel |
| $\boldsymbol{\lambda}$ | Hyperparameters of the Gaussian Process |
| $p$ | Currently executed node in the motion planning algorithm |
| $d$ | Best node in the motion planning algorithm |
| $D$ | Depth of the motion planning algorithm |
| $f$ | Predecessor of the best node in the motion planning algorithm |
| $GP_i$ | $i$-th Primitive generated by the Gaussian Process |
| $\Lambda$ | Number of generated Primitives in the motion planning algorithm |
| $\ell^-_{\text{MIN}}$ | Minimum stride length |
| $\ell^-_{\text{MAX}}$ | Maximum stride length |

## Settling Time Reduction Using Sequences of Motion Primitives

| | |
|---|---|
| $\phi^*$ | Target periodic orbit |
| $\phi_1, \phi_2, \phi_3, \phi_4$ | Successive solution of the equations of motion |
| $\boldsymbol{\Gamma}$ | Matrix to define the cost function in the Optimal Control Problem |
| $J_{\text{STR}}$ | Cost function for the Settling Time Reduction |
| $\alpha_J$ | Penalty term for the total duration of the motion |
| $w$ | Diagonal value for the weighting matrix $\boldsymbol{W}$ of the cost function $J_{\text{STR}}$ |
| $e_K$ | Convergence error for the Settling Time Reduction |
| $t^\star$ | Time when the multi-step sequence chooses $\Phi_{\text{target}}$ |
| $t^o$ | Time of the next impact of the one-step sequence after $t^\star$ |
| $\mathcal{P}$ | Tuple to define the Markov Decision Process |
| $\mathcal{S}$ | State space for the Markov Decision Process |
| $\mathcal{T}$ | Action space for the Markov Decision Process |
| $\mathcal{F}$ | State transition function for the Markov Decision Process |
| $\Phi$ | The set of periodic orbits for the Reinforcement Learning problem |
| $\rho$ | Reward function for the Markov Decision Process |
| $\chi$ | Discount factor for the Markov Decision Process |
| $\boldsymbol{s}_k$ | Current space for the Markov Decision Process |
| $\tau_k$ | Current action for the Markov Decision Process |

| | |
|---|---|
| $\pi$ | Policy for the action selection |
| $Q^{\pi}$ | State-action value function |
| $\epsilon$ | Probability to choose an action randomly |
| $\varphi_i$ | Basis function |
| $Z$ | Total number of basis functions |
| $\boldsymbol{v}$ | Parameter vector to be learnt for state parameterization |
| $\text{card}(\Phi)$ | Cardinality of $\Phi$ |
| $\eta_k$ | Current learning rate of the $Q$-Learning algorithm |
| $\vartheta$ | Scalar for defining the reward function $\rho$ |
| $e_K^+$ | Convergence error for the Reinforcement Learning problem |
| $e_v$ | Velocity error for the Reinforcement Learning problem |
| $\Delta e_K^+$ | Comparison criterion for the overall performance |

## Online Switching between Motion Primitives for Generalization to New Tasks - Balancing

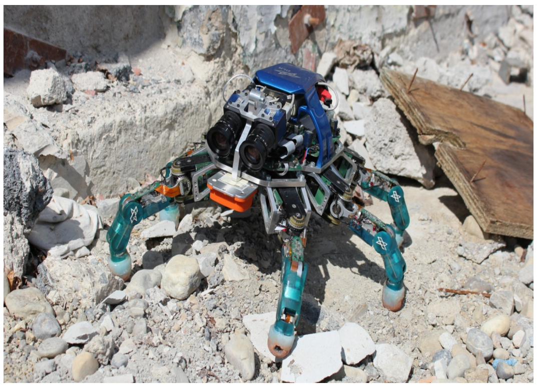
| | |
|---|---|
| $J_{\text{bal}}$ | Cost function for the balancing Motion Primitives |
| $\boldsymbol{W}_1$,$\boldsymbol{W}_2$,$\boldsymbol{W}_3$ | Weighting matrices for $J_{\text{bal}}$ |
| $P_{[j,k]}$ | $k-$th sample of the $j-$th balancing Motion Primitive |
| $T_s$ | Sampling time of each balancing Motion Primitive |
| $\mathcal{DB}$ | Database of Motion Primitives Samples |
| $\boldsymbol{A}$ | State matrix of the closed loop dynamics |
| $\boldsymbol{B}$ | Input matrix of the closed loop dynamics |
| $\boldsymbol{w}$ | Feedback control term |
| $\boldsymbol{P}$ | Solution of the continuous time algebraic Riccati equation |
| $\boldsymbol{Q}$ | Cost matrix of the state in the continuous time LQR problem |
| $\varepsilon$ | Rate of convergence of the RES-CLF |
| $V_{\text{Lyap}}$ | Lyapunov function |
| $\boldsymbol{P}_{\varepsilon}$ | Auxiliary matrix for defining the Lyapunov function $V_{\text{Lyap}}$ |
| $c_1, c_2$ | Constants for bounding the Lyapunov function $V_{\text{Lyap}}$ |
| $c_3$ | Constant for bounding the derivative of the Lyapunov function $V_{\text{Lyap}}$ |
| $\boldsymbol{\psi}_1$ | Auxiliary vector to define the Control Lyapunov Function Constraint |
| $\psi_0$ | Auxiliary term to define the Control Lyapunov Function Constraint |
| $\lambda$ | Eigenvalue of a matrix |
| $\delta$ | Violation of the Control Lyapunov Function Constraint |
| $p_{\text{CLF}}$ | Positive scalar that penalizes the violation of the CLF Constraint |
| $T_n$ | Time period for choosing a new Motion Primitive to track |
| $\boldsymbol{L}$ | Weighting matrix for the Euclidean distance metric |
| $\delta_{\text{thr}}$ | Violation threshold of the CLF Constraint for switching shut off |
| $t^*$ | Time when switching is shut off |
| $\Psi$ | Cost function for the Quadratic Program |
| $\boldsymbol{W}_{\text{QP}}$ | Weighting matrix for the cost function $\Psi$ of the Quadratic Program |
| $\varrho$ | Small positive number to express inequality constraints |
| $P^*$ | Currently tracked Motion Primitive |
| $\ddot{\boldsymbol{\theta}}^*, \boldsymbol{u}^*, \boldsymbol{F}^*$ | Accelerations, inputs and GRFs of $P^*$ |

| | |
|---|---|
| $\boldsymbol{J}_{\text{push}}^{\top}$ | Jacobian at the middle of the torso |
| $\boldsymbol{F}_{\text{push}}$ | Impact force during the balancing experiments |
| $J_{\text{bal}}^{\text{switch}}$ | Value of the cost function $J_{\text{bal}}$ when using the switching approach |
| $J_{\text{bal}}^{\text{classic}}$ | Value of the cost function $J_{\text{bal}}$ when not using the switching approach |

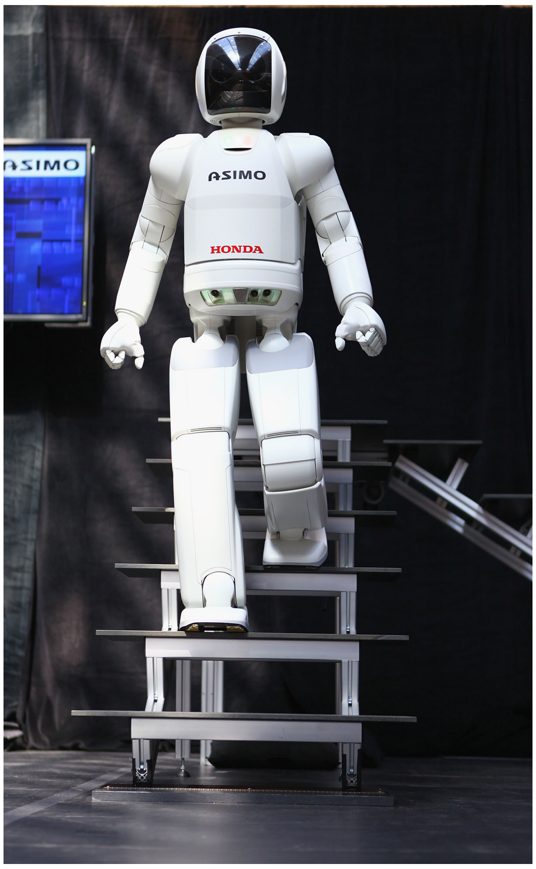
# 1 Introduction

Legged robots are electromechanical systems which use limbs in order to interact with their environment. Their design draws inspiration from nature and most specifically from the biomechanical properties of the human and animal body. Until now most of the legged robots are research platforms in order to advance control and perception algorithms, but the ultimate objective of the current research in legged robots is to be able to deploy them in real-world applications and assist or replace humans in different tasks. The most prominent examples are Search and Rescue missions in human unfriendly environments (Fig. 1.1a), planet exploration (Fig. 1.1b) and human care-taking (Fig. 1.1c).

**(a)** Lauron V developed by FZI suitable for Search & Rescue applications. [1]



**(b)** Crawler robot designed by DLR capable for planetary exploration. [2]

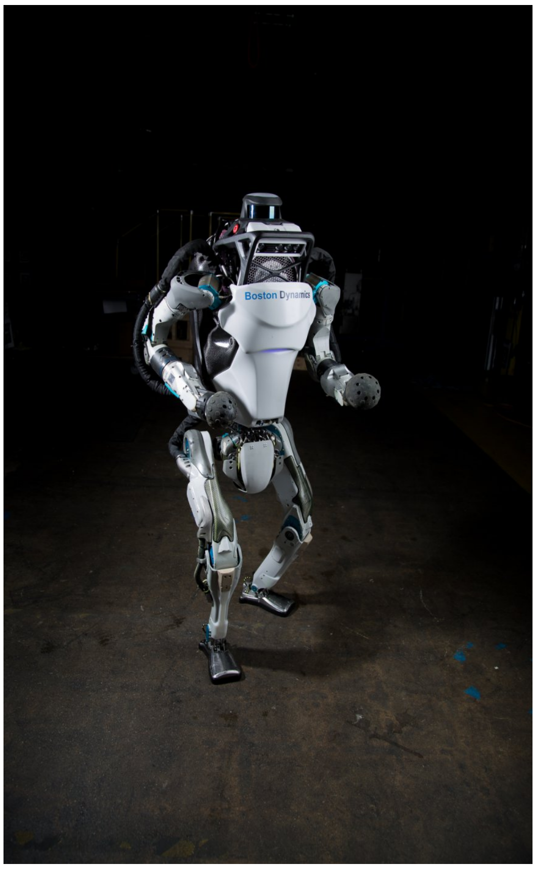


**(c)** Asimo by Honda with the potential of assisting elderly people. [3]

**Fig. 1.1:** Legged robots with the potential of assisting in human unfriendly environments and care-taking.

A special category of legged robots are the *humanoid robots* (like ASIMO in Fig. 1.1c). This kind of robots combine the mobility of legs - that theoretically allow them to maneuver on different terrains - and the manipulation capabilities of arms. There has been special interest for humanoids robots during the last years and special benchmark contests have been established in order to push forward their capabilities, such as the DARPA Robotics Challenge [4]. In this contest humanoid robots shall perform a rescue scenario which among others involves using a cutting machine, turning a valve and even driving a vehicle. Interesting competitors are the ATLAS robot which is used by various teams in the USA and the humanoid of the WALK-MAN project from the Italian Institute of Technology [5] (see Fig. 1.2).

In the real world applications, robots are expected to operate without failures or at least be able to overcome failures. In addition, robustness is expected with respect to model and sensor uncertainties. Due to this prioritization, energy efficiency on real world applications is often not taken into account during the description of the robot behavior or receives minor importance. Another reason for not prioritizing energy efficiency is the fact that online solutions of the robot and task dynamics are computationally expensive and the inclusion of efficiency will lead to an increase in the computational overhead. That is also a prominent reason why simplified models of the Center of Mass (CoM) have been introduced for humanoid robots, which exclude the dynamic coupling of the robot links through the joints [6, 7].


**(a)** ATLAS [8]

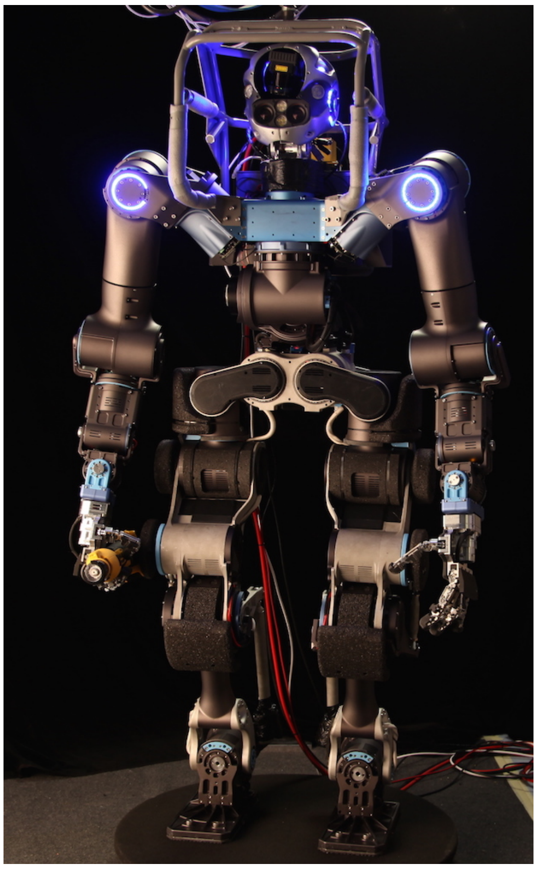

**(b)** WALK-MAN [9]

**Fig. 1.2:** Representative competitors of the 2016 DRC.

In order for legged robots to operate, a task description is necessary. This task description (for example walking with a desired velocity or reaching an object while not falling) is fed to a task planner which will provide desired trajectories on the chosen task space or torque/force inputs which solve the task. Typical task spaces are the joint space and the Cartesian space. In order for the task planner to provide a plan, a robot representation is necessary, i.e. a robot model. In case of walking, simplified models of the CoM are acceptable, but in reaching tasks a multi-body model of the robot shall be used. Afterwards, the task plan and - if necessary - the robot model are forwarded as inputs to a controller that will compute the necessary commands to the robot joints such that the task is realized. These commands are computed by combining the task plan and feedback from the state of the robot in order to compensate for model and sensor uncertainties. If necessary the robot state can also be fed back to the task planner, in case the task planner needs this information in order to decide the next or a new task. This is the typical operation scheme of legged robots and it is summarized in Fig. 1.3.

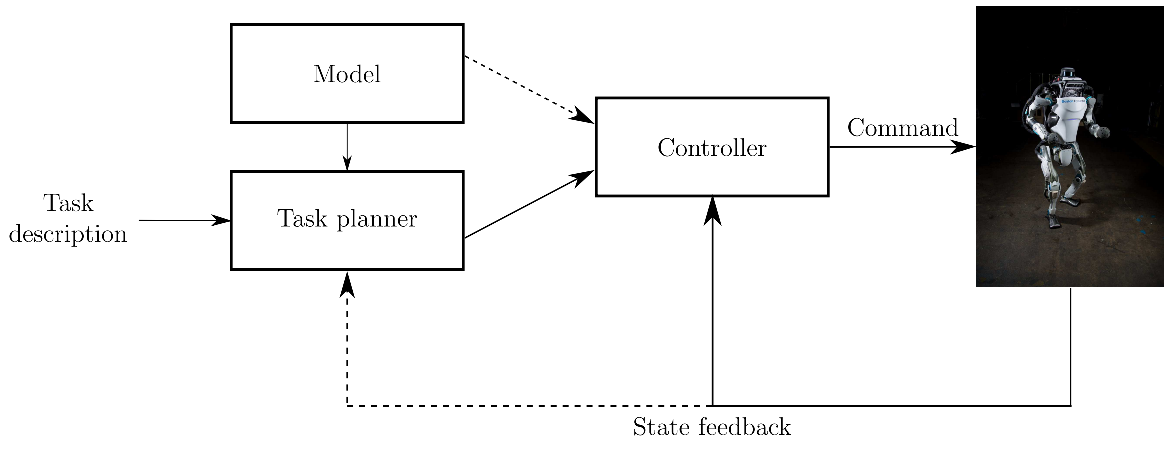
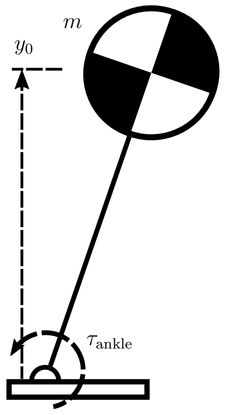


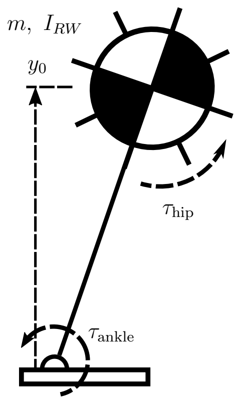
**Fig. 1.3:** Typical operational scheme of legged robots

Regarding the most common tasks that need to be undertaken by legged robots, these are walking and balancing. A walking task plan can be encoded as desired torque inputs and/or trajectories of the actuated robot joints that will propel the robot body towards the desired direction with a desired velocity and/or a final kinematic configuration. In addition, the walking plan can be defined as desired trajectories of the CoM, Zero Moment Point (ZMP) or Center of Pressure (CoP) [10, 11]. Similarly, a balancing task plan can also be encoded as desired trajectories and/or generalized forces for the joints or as desired trajectories of the CoM and ZMP or CoP, but with the difference that they will bring the robot to an equilibrium state. The manipulation of the ZMP or CoP consists of keeping it within the base of support of the robot and is necessary in order to prevent foot tilting and falling. If the manipulation of the ZMP or the CoP is not successful, the manipulation of the Capture Point (CP) can be utilized which will lead to a stepping strategy [12]. For these purposes, the knowledge and manipulation of the Ground Reaction Forces (GRFs) are necessary. Further common tasks are overtaking obstacles [13] which is an extension of walking with the additional task of clearing an obstacle twice with the current swing leg. There is also the task of running which can be defined as an extension of walking with an aerial face [14, 15]. For robots with point feet, the balancing task with both feet in place cannot be realized since the ZMP and CoP cannot be defined for such robots. Finally, the ATLAS robots by Boston Dynamics recently demonstrated the execution of highly dynamic tasks, such as parkour and gymnastic maneuvers [16].

As already hinted above, there are two approaches for the modeling of legged robots: The utilization of simplified models of the CoM and the use of multi-body models. The simplified models of the CoM assume that the total mass of the robot is concentrated on the CoM which moves at a constant height. Well known representatives of this category are the Linear Inverted Pendulum (LIP) (Fig. 1.4a) and the Reaction Wheel Linear Inverted Pendulum (RWLIP) (Fig. 1.4b) [6, 7].



**(a)** Linear Inverted Pendulum



**(b)** Linear Inverted Pendulum with flywheel

In these two simplified models $m$ is the total mass of the robot which is assumed to be concentrated on the CoM, $y_0$ is the constant height of the CoM, $\tau_{\text{ankle}}$ is the torque around the ankle joint and $\tau_{\text{hip}}$ is the torque around the hip which can model the upper body angular momentum of the robot. Finally, $I_{RW}$ is the inertia of the flywheel.

The multi-body dynamic models, on the other hand, are based on the assumption that the robot links are rigid bodies. With such models and depending on the task, the plan can be defined on the joint level and/or the task space. Once the desired positions and velocities have been computed, the corresponding accelerations can be provided through a derivation of the velocities. These three inputs can be used as inputs to the multi-body dynamic model in order to ensure that the desired trajectories are dynamically feasible. This will be done by checking if the resulting nominal torque inputs and GRFs are respecting specified modeling assumptions.

Once a task has been defined and a model representation has been chosen, the task planner will compute the reference motion that executes the task. There are three main approaches regarding the task planner: simplified model based approaches, optimization based approaches and the utilization of learning methods. Simplified model based approaches are combined with simplified models of the CoM. In the case of dynamic walking, a desired ZMP trajectory is computed and its tracking will yield the CoM trajectory. If static walking is used, the task planner will directly provide the CoM trajectory. Regarding the trajectory of the foot of the swing leg, it has been pre-defined. The solution of the inverse kinematics and differential kinematics of the CoM and the foot in swing motion will yield the desired trajectories of each Degree of Freedom (DoF) (Part A , Chapter 1 in [17]). If the robot has servo motors, the desired joint positions and velocities can be commanded to the servo-motor drive. Otherwise, if the robot is torque-driven the desired trajectories can be tracked with a PID controller.

An interesting approach for applying torque commands on robots which are driven by servo motors has been proposed in [18], where the concept of the torque transformer is introduced. With that approach the desired joint torques can be transformed into joint position command signals, given that the time constants of the servo motors are known.

The optimization based approaches use multi-body models with which they can compute feedforward inputs for the robot joints. Common approaches are the utilization of Optimal Control [19] or static optimization where the desired joint trajectories are expressed as parametrized polynomials [20]. These two approaches usually take place offline since they are computationally expensive. However, if the optimization problem can be brought in a computationally efficient form, like a Quadratic Program (QP), it can be potentially solved online, i.e. while the robot executes its motion [21, 22]. If parametrized polynomials are used for the desired trajectories, the approach of input-output feedback linearization with a PD feedback controller can be used to track the desired trajectories (Ch. 5 in [23]). If the optimization outcome is feedforward torque inputs, they can be directly commanded to the robot together with a PD feedback controller.

The utilization of learning approaches does not necessarily require the equations of motion of the robot, since a simulation environment is adequate. It could be argued that these models are extensions of the optimization based approaches, since they converge to a solution by searching for a minimum of an objective function, but we decided to consider them in a different category due to the stochasticity involved in learning approaches. In the case of neural networks the objective function to be minimized is the error between the training data and the current output of the network [24, 25]. In the case of Reinforcement Learning a reward function needs to be maximized [26, 27]. The desired trajectories and inputs can be tracked, respectively applied, with the techniques described above. Learning approaches can also be combined with each other, such that the weights of a neural network can be learned through evaluationary apporoaches [28]. Worth mentioning is also the case of the Central Pattern Generators (CPGs) where different parts of the robot are modeled as an oscillator whose parameters can be learned through Reinforcement Learning [29, 30].

As it is shown in this section, there are many approaches for generating motions for legged robots. However, as we will see in the next section, there is a challenging problem that still remains to be alleviated: Even though there are different kind of humanoids that can perform various tasks, when it comes to online motion generation, the vast majority of them is confined in quasi-static motions, since dynamic motions involve a great amount of computations that have to be performed online. This is due to the fact that humanoids possess highly nonlinear and non-convex dynamics and are subject to different kind of physical constraints as well as unilateral constraints that are imposed on the GRFs [31]. As a result, integrating the equations of motion online in order to compute a motion for the robot - as for example when using Optimal Control [32] - is computationally difficult if not impossible. And due to the computational overhead accompanying the integration of the equations of motion and the online solution of an optimization problem, valid joint torques that will lead to an energy efficient motion cannot be calculated online. However, despite this challenge, the research community has been able to make contributions towards the online motion generation for legged robots.

## 1.1 State of the Art

So far, considerable progress has been made towards the online generation of robot motions with the utilization of simplified models of the CoM like the LIP and the RWLIP, such that humanoids were able to walk on different even and uneven terrains, as well as perform stepping strategies for rejecting severe perturbations [33, 34]. Even though these models have proven very useful for the online motion generation since they allow for fast computations, they result in unnatural hip motions, since the ZMP has to be transferred from the previously supporting foot to the next one during the double support phase. Another limitation is that as already suggested, these models are oversimplifications. That means they cannot be used to design energy efficient motions, since they do not take into account the torque inputs to each actuated joint. In addition, even though the RWLIP has been successful in simplifying balancing studies by utilizing a reaction wheel as a representation of upper body angular momentum, a motion designed with it is not easily transferred on a real humanoid, but as suggested in [7] this topic is currently under work. Finally, there is a simplified model of the CoM, namely the Spring Loaded Inverted Pendulum [35], which has been used for the design of running motions for simulated robots and analysis of running [36, 37].

Further approaches for the online motion planning utilize the full body dynamics of the robot and can provide energy efficient motions which are easily transferable to robots. The only drawback is that online optimization approaches like Model Predictive Control (MPC) are more likely to be successful with robot models that possess few DoFs. When it comes to robots with more DoFs the application of MPC would require a linearization of the robot dynamics and the tuning of many parameters. Unfortunately, the linearization of multi-body systems is only valid within a very small vicinity around the point of linearization and therefore require very small control times. As a consequence, the online solutions of the underlying QP that will provide the joint inputs for the next control step shall take place in a very short time, which is computationally very challenging. A work worth mentioning towards this direction is the concept of Sequential Action Control (SAC) [38], which computes online an improvement of the nomimal control input. Additionally, the work on Nonlinear Model Predictive Control (NMPC) [39] might evolve and allow for its successful application on robots with multiple DoFs. Nonetheless, linear MPC has been applied by Powell *et al.* [40] in an intelligent way using a lower state representation of a 5-link biped robot without feet, namely the Hybrid Zero Dynamics (HZD). An alternative approach to confront the computation overhead inherent in MPC problems, is to make the prediction horizon equal to 0. That means that robustness is compromised in favor of feasibility, since in that case only one and smaller QP shall be solved for the current control step. Due to the lower dimensionality, this approach has been successful on the motion stabilization of simulated humanoids [22].

A different approach that allows for motion generation in an online fashion and is suitable for force controlled robots is the utilization of compliant methodologies which calculate the joint torques in order to comply with desired forces applied on different points of the robot. One of the first representatives of this category is the Virtual Model Control (VMC), where a fictional force is applied on the robot and through the Jacobian transformation a torque command for each DoF is calculated [41]. In the work of Stephens *et al.* [42] a dynamic balance force controller applies a pseudo-inverse approach on the desired CoM forces and momenta as well as the GRFs in order to calculate the joint torques. Another methodology is the Dynamics Filter proposed by Yamane [43] where reference and initially infeasible motions are tracked by human figures using the pseudo-inverse approach. Finally, Cheng *et al.* [44] proposed a passivity-based approach for disturbance rejection.

Another solution to the problem of online motion generation is the introduction of motion libraries/databases. These databases include motions that fulfill different task objectives, e.g. walking with a desired average velocity or balancing after a specific disturbance force is applied on a specific point of the robot body. This approach has the advantage that it uses the full body dynamics of the robot and can include energy efficiency terms in the design phase of the motion, since the construction of the database takes place offline. In order to be able to endow reactive characteristics to the robots or allow for generalization and - when possible - energy efficiency, these motions can be used, tracked, combined or even modified online. The main focus and contribution of this thesis are centered around these reactive and generalization capabilities that can be achieved through the use of motion libraries and are elaborated on the next section.

## 1.2 Contributions

This thesis proposes a methodology to overcome the computational overhead of designing motions online, especially for legged robots which are nonlinear systems with non-convex dynamics and multiple DoFs. In addition, we deal with both underactuation and full actuation, more specifically walking with point feet and balancing tasks. The approach suggests the utilization of motion databases that have been computed offline through an optimization process and each motion, alternatively Motion Primitive, fulfills a specific task objective. In contrast to other approaches that utilize motion databases and focus on the sequencing problem, i.e. how to solve a kinematic problem with them, this thesis focuses on generalization to new and unknown task objectives using the motion database. In addition, we investigate how to enhance the capabilities of the system with respect to the already known tasks and - when possible - retain energy efficiency.

The generalization is achieved with the use of machine learning techniques such as Gaussian Processes (GPs) and Reinforcement Learning, as well as by employing a Euclidean distance metric. The generalization of course shall not compromise the dynamic feasibility of the generated motion, therefore the dynamic feasibility is checked every time a new motion is generated or a new motion command is calculated. In order to demonstrate the generalization better, example applications are investigated, like underactuated walking on uneven terrain and balancing. In addition, the problem of Settling Time Reduction when switching between different walking periodic orbits is introduced from an Optimal Control perspective and as a later step, we use Reinforcement Learning to demonstrate how a database can provide a faster switching strategy than conventional approaches.

In more details, we contribute to the online generation of dynamically feasible motions and the enhancement of the robot system capabilities in the followings ways:

**Motion primitives as training examples for learning models.** The general problem of using reference motions as training examples for generating new motions online is investigated. More specifically, we use the parameters that define the trajectories of the Motion Primitives as training examples and are able to generate stable periodic motions online for a wide range of kinematic configurations.

(i) The parameters of the Motion Primitives are used as training examples for a Gaussian Process. The input to the GP is the desired final stride length and height of the robot. With this approach we demonstrate how Motion Primitives can be used to generalize to new and unknown task objectives. This is a novel methodology for trajectory generation through parameter inference and a novel application of GPs. Even though other learning models could be used, the GPs were favored due to their fast prediction time for the mean of the inferred parameters and their precision.

(ii) For aperiodic walking a best-first planning algorithm is introduced in order to allow the robot to traverse uneven terrain with step variations not corresponding to any Motion Primitive in the training examples. In comparison to other works like [45], the new terrain height specification is always satisfied since if no primitive with such specification exists within the training examples, a new one will be created to match the new terrain height. The dynamic feasibility of each generated primitive is checked online. If the check is negative, the generated primitive is not considered in the plan. The robot is assumed to not have full knowledge of the terrain, but only a limited one, therefore it can plan its actions in a receding horizon. After each step, the algorithm is executed again and a new plan is calculated.

*The methodology of motion generation using a database of Motion Primitives and GP regression was first published in the IEEE International Conference on Robotics and Automation (ICRA) article [46].*

**Sequencing of Motion Primitives for Settling Time Reduction.** The problem of Settling Time Reduction for a transition between two periodic orbits for underactuated walking is investigated. Instead of trying the difficult approach of minimizing the maximum eigenvalue of the Poincaré Map of the target periodic orbit, we employ a database of Motion Primitives to compute multi-step sequences that bring the robot state close to the fixed point of the Poincaré Map of the target periodic orbit.

(i) The problem of Settling Time Reduction is formulated as an Optimal Control problem. We demonstrate a novel approach of bringing the Hybrid Zero Dynamics framework into an Optimal Control formulation as equality and inequality constraints. The cost function to be minimized is the distance to the fixed point of the target periodic orbit and the solution of the problem provides an aperiodic transition motion between the initial and target periodic motion.

(ii) After the problem has been defined, we utilize Reinforcement Learning in order to find multi-step sequences that provide faster convergence to the target periodic orbit than an one-step transition. This is a new methodology for combining Motion Primitives and solving the Settling Time Reduction problem. Since we have Motion Primitives in our disposal, we use them to formulate the Markov Decision Process needed for computing the multi-step sequences and later solve it with a Q-Learning algorithm. The reward function is always checking if the transition from the current state of the robot to the target periodic orbit is more advantageous than performing an intermediate transition towards another periodic orbit. The solution of the problem demonstrates that in many of the possible cases where the multi-step sequences can outperform one-step traditional approaches [47, 20], this is indeed the case.

*The definition of the Settling Time Reduction problem was first published in the International Journal of Humanoid Robotics with [48]. The multi-step sequences approach using Reinforcement Learning was first published in the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) article [49].*

**Online switching of Motion Primitives for balancing.** The problem of balancing with respect to new and unknown disturbances is investigated. For that, a Motion Primitive database is generated for rejecting disturbances of different sizes which are applied on the middle of the robot torso. The motions are optimal with respect to the effort for bringing the robot back to the upright posture. After their computation, the primitives are tokenized in small samples and at a predefined rate the sample closest to the current robot state is chosen. Once a sample has been selected, the primitive owning the chosen sample, is tracked for a predefined time. After this time elapses, the samples are checked again with respect to their distance from the current robot state and if necessary a new one is chosen. This novel approach allows to reject new and unknown disturbances, continuous pushes, as well as multiple pushes by tracking always the currently optimal primitive and leads to efficient balancing motions. In addition, a robustness study is conducted to demonstrate that the approach is also suitable in cases of pushes under model uncertainties.

(i) Balancing motions are extracted in a novel way by parametrizing the robot trajectories with Bézier polynomials and formulating the balancing problem as a static optimization problem. The cost function minimizes deviations from the upright posture and the integral of the squared applied torques. The CoP and friction cone constraints are included as inequality constraints.

(ii) Each motion is split/tokenized in small samples, transforming the database of Motion Primitives into a collection of samples. This novel approach allows to track the primitive whose sample is nearest to the current state. When the switching is not needed anymore the robot tracks an optimal trajectory, therefore the rest of the motion is close-to-optimal with respect to the cost function minimized during the database extraction process, as described above. The tracking problem is formulated as a QP where a Control Lyapunov Function (CLF) Constraint is included. The violation of the CLF is included in the cost function of the QP, such that the robot state converges exponentially to the chosen primitive, when possible. The violation of the CLF is also used to turn off the switching.

(iii) The approach is successfully validated in different scenarios of new and unknown disturbances. Among others, we compare our switching methodology to a non-switching one, where the closest primitive is chosen based solely on the post-impact state of the robot and tracked thereafter. The evaluation demonstrates that our switching methodology results in more efficient balancing motions than the non-switching approach.

*The methodology of Motion Primitive switching for balancing with respect to new and unknown disturbances was first published in the International Journal of Humanoid Robotics with [50].*

The methodologies that led to the aforementioned contributions are summarized graphically in Fig. 1.5.

## 1.3 Thesis Outline

Following the introduction, Chapter 2 discusses the modeling of legged robots as hybrid systems. For that, the Lagrangian formulation for deriving the equations of motion is presented and the relevant constraints on the GRFs are defined.

The theory of Hybrid Zero Dynamics for underactuated walking robots is presented in Chapter 3. In the same chapter we present the static optimization problem which yields the periodic Motion Primitives for chapters 4 and 5. We also discuss interesting characteristics of Bézier polynomials which define the desired outputs to be tracked and lead to the HZD.

In Chapter 4 we present our methodology with respect to online motion generation using Gaussian Processes. A short introduction to the Gaussian Process is given as well as the presentation of the best first algorithm for motion planning on uneven terrain.

Chapter 5 defines the Settling Time Reduction problem using an Optimal Control formulation. In addition, the Markov Decision Process for multi-step sequences is defined and later solved with Reinforcement Learning.

Chapter 6 demonstrates the Motion Primitive switching methodology for balancing. The optimization problem for generating balancing motions is presented as well as the theory of Rapidly Exponentially Stabilizing Control Lyapunov functions. In addition, the overall Quadratic Problem is formulated and the approach is validated in different scenarios.

The thesis concludes with Chapter 7 where we use the achieved contributions as a starting point for possible future research.

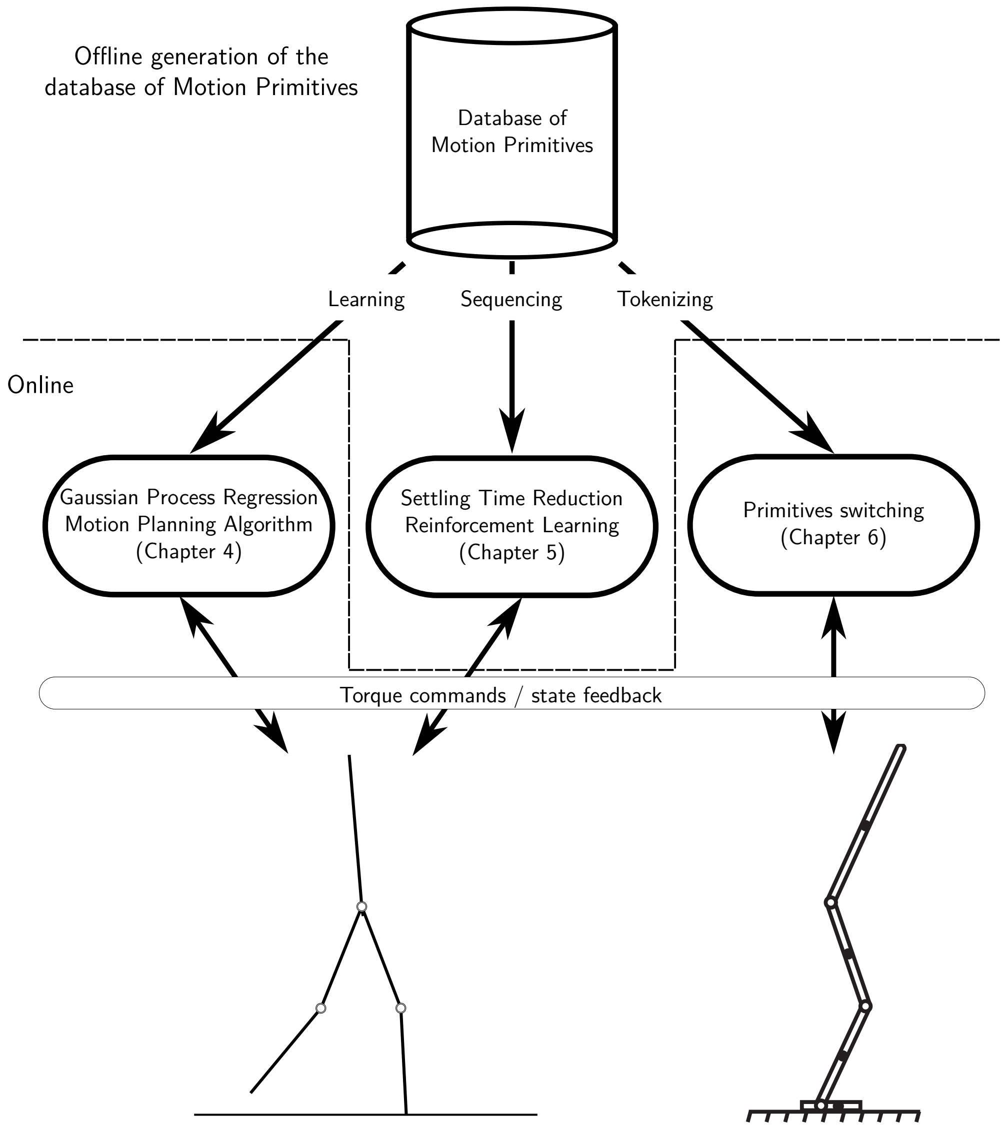

Offline generation of the database of Motion Primitives
Database of Motion Primitives
Learning
Sequencing
Tokenizing
Online
Gaussian Process Regression Motion Planning Algorithm (Chapter 4)
Settling Time Reduction Reinforcement Learning (Chapter 5)
Primitives switching (Chapter 6)
Torque commands / state feedback

**Fig. 1.5:** Graphic representation of the methodologies which led to the contributions of this thesis and their corresponding chapters. The arrowheads towards the robots denote the torque commands and the arrowheads originating from the robots denote the state feedback.

# 2 Modeling of Planar Legged Robots

This chapter introduces the dynamic modeling for walking and balancing robots. The motion is analyzed in the sagittal plane and we further assume that the robots consist of rigid links. Regarding the legs, we can assume that they include feet or not. Walking without feet is suitable for studying and designing gaits that exploit the natural dynamics of the robot which is the case during the heel and toe roll phases of underactuated walking.

Walking is described as a sequence of single and double support. Single support denotes the phase when only one leg - the stance leg - is in contact with the walking surface. At the same time the other leg - the swing leg - performs a swinging motion from the back of the stance foot to the front of it. The single support is succeeded by the double support, which is initiated when the swing leg impacts the walking surface. The double support can be assumed to have finite or infinitesimal duration.

For each walking phase different conditions and modeling assumptions have to be imposed in order for the equations of motion to remain valid. Such conditions include the satisfaction of the friction cone constraint, such that slip is avoided and in the case of robots with feet, the prevention of foot rolling if an underactuated walking phase is not desired. In addition, the position of the tip or the foot of the swing leg needs to be checked against the height of the walking surface in order to switch between the different phases of walking.

Since walking is a sequence of different phases its modeling and analysis can be facilitated by the use of a hybrid, event-based system modeling approach. In that case, a system of differential equations is introduced for the single support phase, another one for the double support and a switching condition which decides when each phase is active. In this thesis, however, we are studying walking with an instantaneous double support, therefore we will use an impact model for this phase, instead of a system of differential equations.

Contrary to walking, balancing in place does not require a hybrid system formulation in order to be described, since it does not consist of a sequence of different phases. In that kind of balancing the feet of the robot are required to retain a flat contact with the ground and avoid foot rolling which leads to underactuation. The other balancing approach allows the robot to perform a stepping sequence in order to dissipate kinetic energy with each step through the impact with the ground and finally come to a stop. In that case, foot rolling is allowed. In this thesis we are dealing with balancing in place and for that we require only a system of differential equations for the balancing model and an impact model to initialize its state after a disturbance.

The structure of this chapter is as follows: In section 2.1 we will introduce the hybrid system approach used in this thesis. In section 2.2 we will derive the governing equations of motion for walking and balancing and we will present the impact model. In section 2.3 we will elaborate on the role of the Ground Reaction Forces for walking and balancing. Finally, section 2.4 will introduce the robot models used in this thesis.

## 2.1 Hybrid Systems

Hybrid systems have been used to describe systems that experience both discrete and continuous dynamics. In this section we will give a brief introduction of the terminology of hybrid systems. More information regarding the theory of hybrid systems can be acquired from [51, 52, 53, 54, 55, 56, 57].

A hybrid system is defined as a dynamical system which is subject to continuous and discrete dynamics. As a consequence the state vector of a hybrid system $\boldsymbol{\sigma}$ is described by a vector of continuous variables $\boldsymbol{x} \in \mathbb{R}^n$ and a discrete mode $x_d \in \mathbb{N}^*$.

$$\boldsymbol{\sigma} = \begin{pmatrix} \boldsymbol{x} \\ x_d \end{pmatrix} \in \mathbb{R}^n \times \mathbb{N}^* \tag{2.1}$$

Similarly to the state, a hybrid system can have both a discrete input $u_d(t)$ and continuous ones $\boldsymbol{u}(t)$. The same holds for the outputs which are determined by an output vector function $\boldsymbol{h}$

$$\begin{pmatrix} \boldsymbol{y} \\ y_d \end{pmatrix} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, x_d, u_d, t) \tag{2.2}$$

The continuous dynamics are also referred to as flow and are described by ordinary differential equations (ODEs)

$$\dot{\boldsymbol{x}} = \boldsymbol{d}_i(\boldsymbol{x}, \boldsymbol{u}, t), \tag{2.3}$$

where $\boldsymbol{d}_i$ is a smooth vector field and is valid for a given discrete mode $x_d = i$. Neglecting the discrete inputs for the rest of the section since they are not applicable in our case, the continuous dynamics can be written in the form of

$$\dot{\boldsymbol{x}} = \begin{cases} \boldsymbol{d}_1(\boldsymbol{x}, \boldsymbol{u}, t), \ x_d = 1 \\ \boldsymbol{d}_2(\boldsymbol{x}, \boldsymbol{u}, t), \ x_d = 2 \\ \dots \\ \boldsymbol{d}_N(\boldsymbol{x}, \boldsymbol{u}, t), \ x_d = N \end{cases} \tag{2.4}$$

A transition from a discrete mode $x_i$ to $x_j$ will cause a switching of the continuous dynamics. Such a transition occurs when the transition surface $S_{i\to j}$ is crossed. The transition surfaces are functions of the discrete mode, continuous states and the continuous inputs

$$\Sigma : S_{i\to j}(\boldsymbol{x}, \boldsymbol{u}) = 0, i \in I_d, \ j \in I_d, \ i \neq j, \tag{2.5}$$

where $I_d \in \mathbb{Z}$ is a finite index set. Please note that in Eq. (2.5) the discrete modes $x_i$ and $x_j$ are dropped since they are implied by the subscript of the transition surface $i \to j$.

When a transition $i \to j$ takes place, the initial state of the system for the vector field $\boldsymbol{d}_j$ will be given by a transition function $\boldsymbol{\Delta}_{i\to j}$. This function depends on both the discrete mode and continuous states as well as the continuous inputs and is allowed to cause a discontinuous jump in the continuous state $\boldsymbol{x}$ of the system. Therefore, the state $\boldsymbol{x}^+$ after crossing the new transition surface $S_{i\to j}$ is defined as

$$\boldsymbol{x}^+ = \boldsymbol{\Delta}_{i\to j}(\boldsymbol{x}^-, \boldsymbol{u}), \tag{2.6}$$

where $\boldsymbol{x}^-$ is the state before crossing $S_{i\to j}$. Again, the discrete modes $x_i$ and $x_j$ are omitted in Eq. (2.6) since they are implied by the subscript of the transition function $i \to j$.

Finally, the formal notation of a hybrid system is as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{d}_i(\boldsymbol{x}, \boldsymbol{u}, t),\ x_d = i \wedge S_{i\to j}(\boldsymbol{x}, \boldsymbol{u}) \neq 0, \forall j \in I_d, j \neq i \tag{2.7}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2.8}$$

$$\boldsymbol{x}^+ = \boldsymbol{\Delta}_{i\to j}\left(\boldsymbol{x}^-, \boldsymbol{u}\right),\ S_{i\to j}(\boldsymbol{x}, \boldsymbol{u}) = 0 \tag{2.9}$$

### Legged Robots as Hybrid Systems

The modeling approach of walking robots with the help of hybrid systems assumes two discrete modes for the single support with each leg. Each of these two discrete modes has its own continuous dynamics. It is also possible to have discrete modes for the double support, where the differentiation between these two modes is made by identifying which leg established contact with the ground as last. However in this thesis the double support is always assumed to be instantaneous and as a consequence there is no need to introduce any continuous dynamics for this phase. Instead, a transition matrix will be enough to model the discontinuous jump in the velocities at the time of impact. The transition surfaces, which dictate when the impact matrix has to be applied, are defined as the states of the robot where the currently swinging leg touches the walking surface. As we will see later, symmetries between the discrete modes of single support can be exploited in order to reduce the number of discrete modes. In Fig. 2.1 we provide a graphical representation of the hybrid model of a compass robot [58]. In order to simplify our representation, foot scuffing is neglected and the continuous input $\boldsymbol{u}$ is omitted in the transition functions, since such a dependency does not hold for this model.

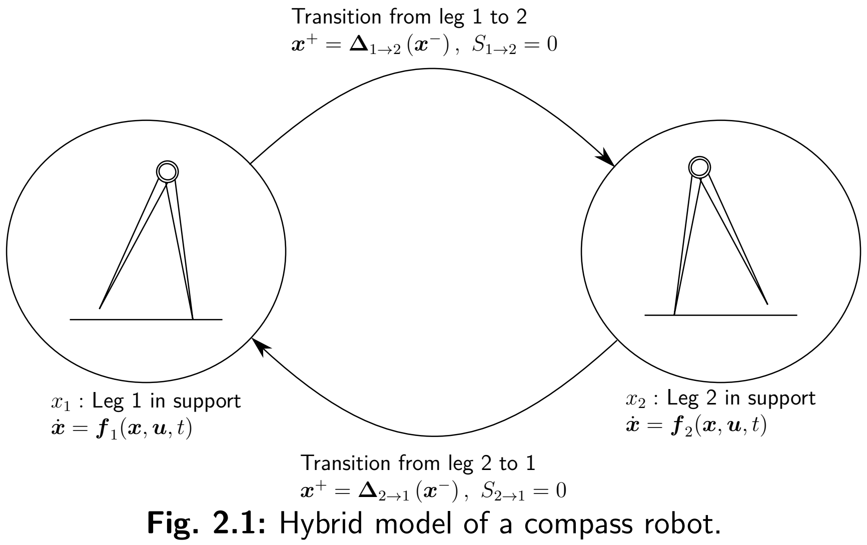

Transition from leg 1 to 2
$\boldsymbol{x}^+ = \boldsymbol{\Delta}_{1\to 2}(\boldsymbol{x}^-),\ S_{1\to 2} = 0$
$x_1$ : Leg 1 in support
$\dot{\boldsymbol{x}} = \boldsymbol{f}_1(\boldsymbol{x}, \boldsymbol{u}, t)$
$x_2$ : Leg 2 in support
$\dot{\boldsymbol{x}} = \boldsymbol{f}_2(\boldsymbol{x}, \boldsymbol{u}, t)$
Transition from leg 2 to 1
$\boldsymbol{x}^+ = \boldsymbol{\Delta}_{2\to 1}(\boldsymbol{x}^-),\ S_{2\to 1} = 0$

**Fig. 2.1:** Hybrid model of a compass robot.

## 2.2 Equations of Motion

This section develops the mathematical modeling of planar bipedal walking robots where the legs are symmetric and connected to a common point which we will identify as the

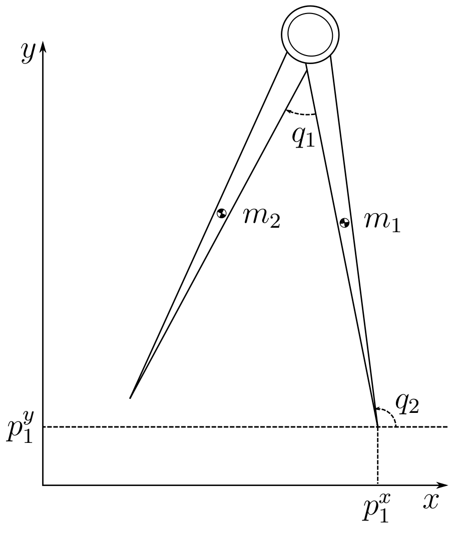


**Fig. 2.2:** Compass gait biped with actuation on the hip joint.

hip. The inertial frame is assumed to be oriented such that the $y$-axis points upwards while the gravity vector points downwards. A further assumption is that the height of the walking surface is zero with respect to the inertial frame. For clarification purposes we will accompany the modeling of walking robots with the academic example of a compass gait biped depicted in Fig. 2.2 which resembles the simplest walking machine introduced in [58] with the difference that the masses are not concentrated on the hip and leg tips but they are distributed on the leg links. In addition, the hip joint is assumed to be actuated.

## Single Support Phase

During the single support phase the robot can be modeled as an open kinematic chain. Let $\Theta_s$ denote the configuration space of the robot and $\boldsymbol{\theta} = [q_1 \ \cdots \ q_n \ p_1^x \ p_1^y]^\top$ denote a vector of generalized coordinates where $q_i, i = 1, ..., n$ refer to the rotational DoFs and $\boldsymbol{p}_1 = [p_1^x \ p_1^y]^\top$ refer to the $x - y$ Cartesian coordinates of a point of the robot. For this thesis we will assume that this point is the tip of the stance leg.

The equations of motion will be derived using the Lagrangian method and the assumption of rigid bodies [59]. According to the method of Lagrange, the first step is to define the Lagrangian equation as

$$L\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) = K\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right) - V\left(\boldsymbol{\theta}\right), \tag{2.10}$$

where $K$ is the sum of the kinetic energies of each link and $V$ is the sum of potential energies of each link. As a consequence, $K$ and $V$ can be assumed as the total kinetic and potential energy, respectively, of the robot. As a next step, the formulation of Lagrange's equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{\theta}}} - \frac{\partial L}{\partial \boldsymbol{\theta}} = \boldsymbol{U} \tag{2.11}$$

will yield the equations of motion for the robot. Please note that the term $\boldsymbol{U}$ in Eq. (2.11) refers to the vector of generalized torques and forces acting on the robot.

Applying the method of Lagrange and taking into account the assumption that the stance leg should remain pinned on the ground, the equations of motion are given as

$$\boldsymbol{D}_e(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}_e(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{G}_e(\boldsymbol{\theta}) = \boldsymbol{S}_e\boldsymbol{u} + \boldsymbol{J}_1^\top(\boldsymbol{\theta})\boldsymbol{F} \tag{2.12a}$$

$$\boldsymbol{J}_1(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{J}}_1(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} = \boldsymbol{0}, \tag{2.12b}$$

where $\boldsymbol{D}_e(\boldsymbol{\theta}) \in \mathbb{R}^{(n+2)\times(n+2)}$ is the mass-inertia matrix, $\boldsymbol{C}_e(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \mathbb{R}^{(n+2)\times(n+2)}$ is the matrix of centrifugal and Coriolis terms, $\boldsymbol{G}_e(\boldsymbol{\theta}) \in \mathbb{R}^{n+2}$ summarizes the gravitational terms, $\boldsymbol{u} \in \mathbb{R}^n$ is the vector of generalized torques and $\boldsymbol{S}_e \in \mathbb{R}^{(n+2)\times n}$ is the input matrix. Finally, $\boldsymbol{F} = [F_x \; F_y]^T$ is the vector of Ground Reaction Forces (GRFs) and $\boldsymbol{J}_1(\boldsymbol{\theta}) \in \mathbb{R}^{2\times(n+2)}$ is the Jacobian matrix of the tip of the stance leg.

Since we assume that $\boldsymbol{p}_1$ are the Cartesian coordinates of the tip of the stance leg the Jacobian of the tip is given as

$$\boldsymbol{J}_1 = \frac{\partial \boldsymbol{p}_1}{\partial \boldsymbol{\theta}} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_n & \boldsymbol{I}_2 \end{bmatrix} \tag{2.13}$$

Solving Eq. (2.12b) for $\ddot{\boldsymbol{p}}_1$ will result in $\ddot{\boldsymbol{p}}_1 = \boldsymbol{0}$. In addition, since the tip of the stance leg shall remain pinned on the ground, we have $\dot{\boldsymbol{p}}_1 = \boldsymbol{0}$. Let us now re-write Eq. (2.12a) as

$$\begin{bmatrix} \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{D}_{21} & \boldsymbol{D}_{22} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \ddot{\boldsymbol{p}}_1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}}_1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_2 \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{F}, \tag{2.14}$$

where we omit the arguments of the matrix and vector functions for brevity. For a robot in single support it holds that $\boldsymbol{S}_2 = \boldsymbol{0}$. Using this information and substituting $\ddot{\boldsymbol{p}}_1 = \dot{\boldsymbol{p}}_1 = \boldsymbol{0}$ in Eq. (2.14) will yield the constrained dynamics of the robot, i.e. the first line of Eq. (2.14)

$$\boldsymbol{D}_{11}\ddot{\boldsymbol{q}} + \boldsymbol{C}_{11}\dot{\boldsymbol{q}} + \boldsymbol{G}_1 = \boldsymbol{S}_1\boldsymbol{u} \tag{2.15}$$

Using the second line of Eq. (2.14) we acquire the expression for the GRFs

$$\boldsymbol{F} = \boldsymbol{D}_{21}\ddot{\boldsymbol{q}} + \boldsymbol{C}_{21}\dot{\boldsymbol{q}} + \boldsymbol{G}_2. \tag{2.16}$$

### Impact

The impact is assumed to be inelastic and instantaneous and takes place when the swing leg of the robot touches the walking surface, i.e. when $p_2^y = 0$. When the swing leg impacts the ground the previous stance leg lifts from the ground without interaction. The impact between the swing leg and the ground results in externally applied forces and causes a discontinuity in the velocities, but not the positions. In order to find the externally applied forces at the impact and the post-impact velocities we have to utilize Eq. (2.12a) with $\boldsymbol{S}_2 = \boldsymbol{0}$ and include the externally applied forces $\delta\boldsymbol{F}_2$, such that

$$\boldsymbol{D}_e(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}_e(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{G}_e(\boldsymbol{\theta}) = \boldsymbol{S}_e\boldsymbol{u} + \boldsymbol{J}_2^\top(\boldsymbol{\theta})\delta\boldsymbol{F}_2, \tag{2.17}$$

where $\boldsymbol{J}_2(\boldsymbol{\theta}) = \frac{\partial \boldsymbol{p}_2}{\partial \boldsymbol{\theta}}$ denotes the Jacobian of the tip of the swing leg. During the impact the conservation of momentum holds and it will yield

$$\int_{t^-}^{t^+} \left\{ \boldsymbol{D}_e(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \boldsymbol{C}_e(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{G}_e(\boldsymbol{\theta}) \right\} \mathrm{d}t = \int_{t^-}^{t^+} \left\{ \boldsymbol{S}_e \boldsymbol{u} + \boldsymbol{J}_2^\top(\boldsymbol{\theta})\delta \boldsymbol{F}_2 \right\} \mathrm{d}t, \tag{2.18}$$

where $t^-$ and $t^+$ are the times just before and after the impact, respectively. Keeping in mind that during the impact the positions do not change and setting the integral of the actuators equal to 0 since they cannot generate impulses we get

$$\boldsymbol{D}_e(\boldsymbol{\theta}^-) \left( \dot{\boldsymbol{\theta}}^+ - \dot{\boldsymbol{\theta}}^- \right) = \boldsymbol{J}_2^\top(\boldsymbol{\theta}^-)\boldsymbol{F}_2, \tag{2.19}$$

where $\boldsymbol{F}_2$ is the impulsive impact force acting on the tip of the swing leg at the moment of the impact.

Another assumption that holds for the impact is that it does not result in any rebound or slipping of the swing leg. This can be expressed as

$$\boldsymbol{J}_2(\boldsymbol{\theta}^-)\dot{\boldsymbol{\theta}}^+ = 0 \tag{2.20}$$

The system of Eq. (2.19) and Eq. (2.20)

$$\begin{bmatrix} \boldsymbol{D}_e(\boldsymbol{\theta}^-) & -\boldsymbol{J}_2^\top(\boldsymbol{\theta}^-) \\ \boldsymbol{J}_2(\boldsymbol{\theta}^-) & \boldsymbol{0}_2 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}}^+ \\ \boldsymbol{F}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_e(\boldsymbol{\theta}^-)\dot{\boldsymbol{\theta}}^- \\ \boldsymbol{0}_2 \end{bmatrix} \tag{2.21}$$

provides $n+4$ equations which when solved gives the post-impact velocities $\dot{\boldsymbol{\theta}}^+$ and impact force $\boldsymbol{F}_2$. Given that the tip of the stance foot is pinned on the ground, i.e $\dot{\boldsymbol{p}}_1^- = \boldsymbol{0}$, we can use an alternative expression for Eq. (2.21) which maps the pre-impact joint velocities $\dot{\boldsymbol{q}}^-$ to the post-impact velocities and impulse forces at the impact

$$\begin{bmatrix} \boldsymbol{D}_e(\boldsymbol{\theta}^-) & -\boldsymbol{J}_2^\top(\boldsymbol{\theta}^-) \\ \boldsymbol{J}_2(\boldsymbol{\theta}^-) & \boldsymbol{0}_2 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\theta}}^+ \\ \boldsymbol{F}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}_e(\boldsymbol{\theta}^-) \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{2\times n} \end{bmatrix} \\ \boldsymbol{0}_2 \end{bmatrix} \dot{\boldsymbol{q}}^- \tag{2.22}$$

Solving this system results in the impact forces

$$\boldsymbol{F}_2 = -\left( \boldsymbol{J}_2 \boldsymbol{D}_e^{-1} \boldsymbol{J}_2^\top \right)^{-1} \boldsymbol{J}_2 \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{2\times n} \end{bmatrix} \dot{\boldsymbol{q}}^- = \boldsymbol{\Delta}_F \dot{\boldsymbol{q}}^- \tag{2.23}$$

and the post-impact velocities

$$\dot{\boldsymbol{\theta}}^+ = \left( \boldsymbol{D}_e^{-1} \boldsymbol{J}_2^\top \boldsymbol{\Delta}_F + \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{2\times n} \end{bmatrix} \right) \dot{\boldsymbol{q}}^- = \boldsymbol{\Delta}_q \dot{\boldsymbol{q}}^- \tag{2.24}$$

In order to avoid introducing a new discrete mode after the impact where the roles of the stance and swing legs swap, we introduce a relabelling matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ to relabel the coordinates such that only Eq. (2.15) is used for both single support phases. Please note that $\boldsymbol{R}$ must be a circular matrix, i.e. $\boldsymbol{R}\boldsymbol{R} = \boldsymbol{I}$. With this relabelling, the post-impact joint positions are given as

$$\boldsymbol{q}^{+} = \boldsymbol{R}\boldsymbol{q}^{-} \tag{2.25}$$

and the post-impact joint velocities are given as,

$$\dot{\boldsymbol{q}}^{+} = [\boldsymbol{R}\ \boldsymbol{0}_{n\times 2}]\, \boldsymbol{\Delta}_{q}\left(\boldsymbol{q}^{-}\right)\dot{\boldsymbol{q}}^{-} \tag{2.26}$$

$$\dot{\boldsymbol{q}}^{+} = \boldsymbol{\Delta}_{q}^{R}\left(\boldsymbol{q}^{-}\right)\dot{\boldsymbol{q}}^{-} \tag{2.27}$$

## Hybrid Model of Walking

The hybrid model of walking includes the single support continuous dynamics and the reset map for the state of the system which is employed at the impact. As state of the system $\boldsymbol{x}$ we define the joint positions and velocities such that $\boldsymbol{x} = \left[\boldsymbol{q}^{\top}\ \dot{\boldsymbol{q}}^{\top}\right]^{\top}$. In that case and by introducing $\boldsymbol{D}_{11} = \boldsymbol{D}$, $\boldsymbol{C}_{11} = \boldsymbol{C}$, $\boldsymbol{G}_{1} = \boldsymbol{G}$ and $\boldsymbol{S}_{1} = \boldsymbol{S}$ the accelerations $\ddot{\boldsymbol{q}}$ during the single support can be written in affine form with respect to the inputs $\boldsymbol{u}$ as

$$\ddot{\boldsymbol{q}} = \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right) + \boldsymbol{D}^{-1}\boldsymbol{S}\boldsymbol{u} \tag{2.28}$$

Using that, the state space system equations that describe the state evolution during the single support are given as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right) + \boldsymbol{D}^{-1}\boldsymbol{S}\boldsymbol{u} \end{bmatrix} \tag{2.29}$$

or in affine form

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{D}^{-1}\boldsymbol{S} \end{bmatrix}\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \tag{2.30}$$

Regarding the state resetting it is given as

$$\boldsymbol{x}^{+} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Delta}_{q}^{R} \end{bmatrix}\boldsymbol{x}^{-} = \boldsymbol{\Delta}\boldsymbol{x}^{-} = \boldsymbol{\Delta}\left(\boldsymbol{x}^{-}\right) \tag{2.31}$$

and it is applied when the swing leg touches the ground and is in front of the stance leg. With this condition the switching surface can be defined as the set

$$\mathcal{S} = \{\boldsymbol{x} | p_2^{y}(\boldsymbol{x}) = 0, p_2^{x}(\boldsymbol{x}) > 0\} \tag{2.32}$$

and the complete description of the hybrid system of walking is given as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u},\ \boldsymbol{x}^{-} \notin \mathcal{S} \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{x}^{+} &= \boldsymbol{\Delta}\boldsymbol{x}^{-},\ \boldsymbol{x}^{-} \in \mathcal{S} \end{aligned} \tag{2.33}$$

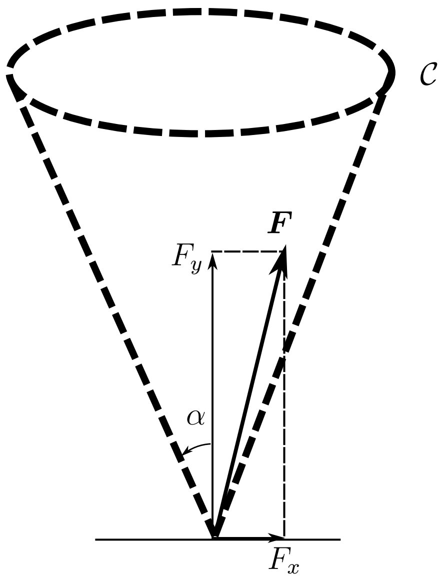


**Fig. 2.3:** Graphic representation of the friction cone constraint.

## 2.3 The Role of the Ground Reaction Forces

The GRFs determine the interaction between the robot and the standing or walking surface. This interaction can be verified by checking the validity of different conditions imposed on the GRFs. For this thesis, the important conditions are the unilateral contact constraint, the no slipping and the no rotation condition.

### Unilateral Contact Condition

The unilateral contact constraint $F_y \geq 0$ denotes the mechanical constraint that prevents penetration between the robot foot and the surface. In order to enforce a constant contact between the two bodies, the constraint can be stricter by using only the inequality $F_y > 0$.

### No Slipping Condition

Slipping is an undesired phenomenon for walking robots. As a consequence, the friction cone constraint which states that the resultant GRF shall stay within the friction cone $\mathcal{C}$ (see Fig. 2.3) has to be respected during a walking or balancing motion. This is formally defined as

$$-\mu_s F_y \leq F_x \leq \mu_s F_y \tag{2.34}$$

where $\mu_s$ is the Coulomb static friction coefficient of the walking or standing surface. In Fig. 2.3 the friction angle (or angle of repose) $\alpha$ is equal to the inverse tangent of the Coulomb static friction coefficient such that $\alpha = \tan^{-1} \mu_s$.

### No Rotation Condition

This condition ensures that the robot foot retains a flat contact with the ground. Retaining a flat contact with the ground is the main objective of balancing, unless stepping is also accepted as a balancing strategy. This condition shall also be checked and respected if a

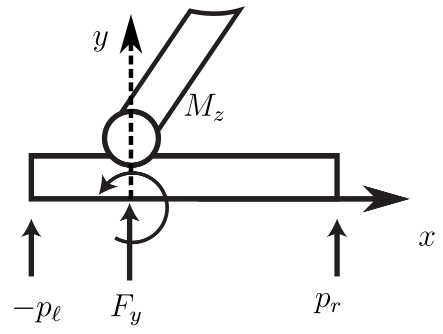


**Fig. 2.4:** Contact force $F_y$ and contact moment $M_z$.

three phase walking strategy is followed. More specifically, during a three phase walking strategy (which resembles human walking) the heel strike event, is followed by a flat contact and then a toe roll phase [60]. During the flat phase the no rotation condition shall hold.

The no rotation condition can be expressed with the help of the Center of Pressure (CoP) [11] and/or the Zero Moment Point (ZMP) [10].

The CoP is defined as the point on the standing/walking surface where the GRF is acting. The ZMP is defined as the point where the total moment acting on the robot due to gravity and inertia forces is zero. On flat ground the ZMP and the CoP coincide with each other if the robot is balancing. If the robot foot is rotating, then the CoP still exists, but the ZMP does not.

Focusing on the case of a flat contact being already established, the no rotation condition requires that the CoP or the ZMP lies within the foot surface. In that case, the robot motion is assumed to be dynamically stable. If the robot has both feet on the ground, then the condition requires that the CoP or the ZMP lies within the convex hull formed by the feet.

For the definition of the CoP/ZMP we will restrict ourselves to the sagittal plane and assume a contact between the robot foot and the standing/walking surface. Given that the tangential component of the GRF $F_x$ does not generate any moment around the foot, the moment balance around the foot can be expressed as

$$M_z = F_y \cdot \mathrm{CoP} \tag{2.35}$$

where $M_z$ is the contact moment and the CoP is defined with respect to the ankle as the point where the GRF is acting (see Fig. 2.4). Therefore, the CoP is defined as the fraction $\mathrm{CoP} = M_z/F_y$ and the stability condition is expressed as

$$-p_\ell < \mathrm{CoP} < p_r \tag{2.36}$$

where $p_\ell$ is the distance from the ankle to the left foot edge and $p_r$ is the distance from the ankle to the right foot edge . If $\mathrm{CoP} = -p_\ell$, then the robot will start rotating around the left foot edge. Similarly, if $\mathrm{CoP} = p_r$, the robot will start rotating around the right foot edge. Finally, if $F_y = 0$, the robot is in the air where neither the CoP nor the ZMP are defined.

As an alternative to the ZMP and the CoP, the Foot Rotation Indicator (FRI) was introduced by Goswami [61] as the point on the walking/standing surface where the net GRF would have to act in order to keep the foot stationary. The stability criterion is similar to the one for the CoP and ZMP. In order for the robot to balance, the FRI must remain within the surface of the foot or the support polygon formed by both feet. In contrast to the ZMP, however, the FRI still exists if the robot rotates around the foot. In addition, the distance of the FRI to the rotating foot edge serves as an indication of the severity of the destabilizing moment around the rotating foot edge. If the FRI remains within the foot (or the convex hull of both feet), then it coincides with the CoP and as a consequence with the ZMP.

## 2.4 Models Used in this Thesis

As already discussed in section 1.2, this thesis uses undeactuated walking and balancing as benchmarks to demonstrate the advantages of motion databases with respect to online motion generation and generalization to new and previously unknown task objectives. For all the walking related studies within this thesis a planar walking robot with point feet was used whose parameters match those of RABBIT [62] (see Fig. 2.5 and Table 2.1). The reason was the fact that RABBIT is a robot capable of underactuated dynamic walking on even as well uneven terrain. In addition, due to the fact that RABBIT contains a torso link, it can utilize its angular momentum to control its walking speed in an easier and faster way in comparison to robots without a torso. The equations of motion were generated using the software AUTOLEV [63]. For the balancing experiments, feet were

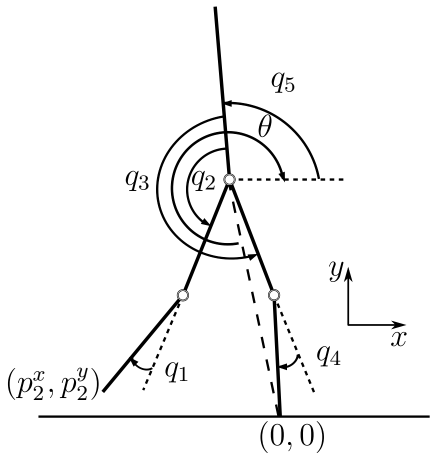


**Fig. 2.5:** Kinematic model of the biped under study. The unactuated DoF is the torso angle $q_5$, but other modeling options are valid as well.

introduced to the planar biped in Fig. 2.5 and the assumption was made that the ankle, knee and hip joints of each leg execute the same trajectories. Therefore, the femur, tibia and feet were modeled as one. The mass and inertia of each new resulting link is the sum of the mass and inertia of the corresponding individual links (see Fig. 2.6 and Table 2.2).

**Tab. 2.1:** Robot parameters for the walking studies

| | **Tibia** | **Femur** | **Torso** |
|---|---|---|---|
| **Mass** (kg) | 3.2 | 6.8 | 12 |
| **Length** (m) | 0.4 | 0.4 | 0.63 |
| **Inertia** (kg·m$^2$) | 0.2 | 0.47 | 1.33 |
| **Mass Center** (m) | 0.16 | 0.29 | 0.24 |
| | (from ankle) | (from knee) | (from hip) |

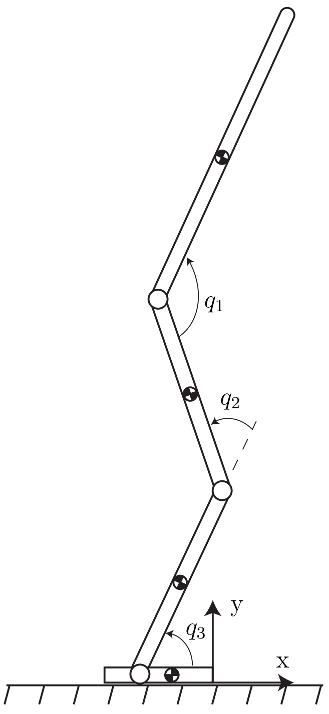


**Fig. 2.6:** Model of the planar robot used for balancing. The Newtonian reference frame is assumed to be fixed on the ground at the tip of the foot.

**Tab. 2.2:** Robot parameters for the balancing studies

| | **Foot** | **Tibia** | **Femur** | **Torso** |
|---|---|---|---|---|
| **Mass** (kg) | 0.4 | 6.4 | 13.6 | 12 |
| **Length** (m) | 0.2 (0.05 from heel to ankle) | 0.4 | 0.4 | 0.63 |
| **Inertia** (kg·m$^2$) | 0.1 | 0.4 | 0.94 | 1.33 |
| **Mass Center** (m) | 0.12 | 0.16 | 0.29 | 0.24 |
| | (from tip) | (from ankle) | (from knee) | (from hip) |

# 3 Hybrid Zero Dynamics of Walking Robots

Task objectives can be fulfilled by designing a set of desired outputs for each actuated DoF and then providing a control law in order to track these outputs. A very popular method for tracking control of such outputs is the *computed torque control* [64, 65]. This methodology uses the concept of input-output feedback linearization [66] and leads to a control law that will force the outputs to asymptotically converge to zero. Once the outputs are tracked or *zeroed*, the dynamics of the system will evolve on the *zero dynamics* manifold, which is the internal dynamics of the system that are compatible with the outputs being zero.

The dimension of the zero dynamics is defined as the difference between the dimension of the vector output relative degree of the system and the dimension of the state vector. As we will see later, walking robots with one degree of underactuation have a vector relative degree which is smaller than the dimension of the state vector. As a consequence, additional equations are required in order to create a coordinate transformation and proceed with input-output feedback linearization. In addition, the zero dynamics need to also be defined at the impact and of course be checked for stability.

This chapter presents the concept of the Hybrid Zero Dynamics for walking robots, which was originally introduced in [67]. The description of the Hybrid Zero Dynamics (HZD) in section 3.1 follows our narrative, which compared to the initial definition, it describes the Hybrid Zero Dynamics concept in a compact form which focuses also on the Langrangian formulation of the zero dynamics. Besides the introduction of the Hybrid Zero Dynamics, [67] introduced a methodology to design outputs for periodic walking while at the same time checking the feasibility and stability of the whole motion in the lower dimensional zero dynamics manifold. As we will see in section 3.2, the outputs are expressed as *virtual constraints*, which are functional relations on the configuration variables of the robot that are dynamically imposed through feedback control [68, 69]. The design of the virtual constraints is realized by parametrizing the desired trajectories of the actuated DoFs with Bézier polynomials and finally computing the Bézier coefficients through the solution of a nonlinear static optimization problem, which is presented in section 3.3. The chapter concludes with a brief summary in section 3.4.

## 3.1 Zero Dynamics of Underactuated Walking

The design of periodic orbits requires conditions which ensure that the orbits are dynamically feasible and stable. Such conditions can be provided in the HZD framework. We show that the zero dynamics allow for a Lagrangian formulation which facilitates the derivation of these conditions. Furthermore, we define the Poincaré Map and its corresponding fixed point in the zero dynamics manifold. In order to do so, however, we need to start with the computation of the output relative degree of our system.

### 3.1.1 Output Relative Degree

As mentioned in the introduction of this chapter, the outputs will be designed using the concept of virtual constraints, which will be defined as desired trajectories $\boldsymbol{h}_d$ for the actuated DoFs, such that the outputs $\boldsymbol{y}$ are zeroed as long as the desired trajectories $\boldsymbol{h}_d$ are tracked with the help of a feedback controller. The outputs $\boldsymbol{y}$ are formally defined as

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{H}_0\boldsymbol{q} - \boldsymbol{h}_d, \tag{3.1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$, $\boldsymbol{h}_d \in \mathbb{R}^{n-1}$ and

$$\boldsymbol{H}_0 = \begin{bmatrix}\boldsymbol{I}_{n-1} & \boldsymbol{0}_{(n-1)\times 1}\end{bmatrix}. \tag{3.2}$$

Differentiating the output vector $\boldsymbol{y}$ twice will result in

$$\begin{aligned}
\frac{d^2\boldsymbol{y}}{dt^2} &= \begin{bmatrix}\frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right) & \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\end{bmatrix}\underbrace{\begin{bmatrix}\dot{\boldsymbol{q}}\\ \ddot{\boldsymbol{q}}\end{bmatrix}}_{\dot{\boldsymbol{x}}} \text{ (recall Eq. (2.30))}\\
&= \begin{bmatrix}\frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right) & \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\end{bmatrix}\left[\begin{bmatrix}\dot{\boldsymbol{q}}\\ \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right)\end{bmatrix} + \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{D}^{-1}\boldsymbol{S}\end{bmatrix}\boldsymbol{u}\right] \qquad (3.3)\\
&= \begin{bmatrix}\frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right) & \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{q}}\\ \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right)\end{bmatrix} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\boldsymbol{D}^{-1}\boldsymbol{S}\boldsymbol{u},
\end{aligned}$$

showing that each output has a relative degree $r$ of 2 and therefore the vector relative degree of the system is $(n-1)r = 2n-2 \neq 2n$ (Ch. 5 in [23]). As a consequence, we will need two additional equations to be able to find a coordinate transformation that brings the system (2.33) into an input-output linearizable form. These two additional equations together with the output vector $\boldsymbol{y}$ and its first order time derivative $\dot{\boldsymbol{y}}$ will lead to a diffeomorphism and its inversion will yield the state $\boldsymbol{x}$.

### 3.1.2 Swing Phase Zero Dynamics

The coordinate transformation $\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x})$ can be chosen as

$$\boldsymbol{z} = \begin{bmatrix} z_1\\ z_2\\ \vdots\\ z_{n-1}\\ ---\\ z_n\\ \vdots\\ z_{2n-2}\\ ---\\ z_{2n-1}\\ z_{2n}\end{bmatrix} = \begin{bmatrix} h_1\\ h_2\\ \vdots\\ h_{n-1}\\ ---\\ \dot{h}_1\\ \vdots\\ \dot{h}_{n-1}\\ ---\\ \xi_1\\ \xi_2\end{bmatrix} \tag{3.4}$$

where in the $\boldsymbol{z}$ coordinates the system will be brought into the input-output linearizable form

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{z}_1 \\ \vdots \\ \dot{z}_{n-1} \\ - \; - \; - \\ \dot{z}_n \\ \vdots \\ \dot{z}_{2n-2} \\ - \; - \; - \\ \dot{z}_{2n-1} \\ \dot{z}_{2n} \end{bmatrix} = \begin{bmatrix} z_n \\ \vdots \\ z_{2n-2} \\ - \; - \; - \; - \; - \\ \left[\frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right) \;\; \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\right]\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right)\end{bmatrix} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\boldsymbol{D}^{-1}\boldsymbol{S}\boldsymbol{u} \\ - \; - \; - \; - \; - \\ \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} \tag{3.5}$$

In order to linearize the system we need to apply the appropriate zeroing input $\boldsymbol{u}$. Therefore we will solve $\frac{d^2\boldsymbol{y}}{dt} = 0$ for $\boldsymbol{u}$ and find the nomimal input

$$\begin{aligned} \boldsymbol{u}^* &= -\left(\underbrace{\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\boldsymbol{D}^{-1}\boldsymbol{S}}_{\boldsymbol{A}_u}\right)^{-1}\left(\underbrace{\frac{\partial}{\partial \boldsymbol{q}}\left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right)\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\boldsymbol{D}^{-1}\left(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}\right)}_{\boldsymbol{b}_u}\right) \\ \boldsymbol{u}^* &= -\boldsymbol{A}_u^{-1}\boldsymbol{b}_u \end{aligned} \tag{3.6}$$

However the stabilization of the first $n-2$ lines of (3.5) will require the consideration of a feedback input $\boldsymbol{v}(\boldsymbol{z})$ in the nominal input $\boldsymbol{u}^*$ such that

$$\boldsymbol{u} = \boldsymbol{A}_u^{-1}\left(\boldsymbol{v}(\boldsymbol{z}) - \boldsymbol{b}_u\right) \tag{3.7}$$

Applying the input (3.7) will yield the input-output linearized system

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{z}_1 \\ \vdots \\ \dot{z}_{n-1} \\ - \; - \; - \\ \dot{z}_n \\ \vdots \\ \dot{z}_{2n-2} \\ - \; - \; - \\ \dot{z}_{2n-1} \\ \dot{z}_{2n} \end{bmatrix} = \begin{bmatrix} z_n \\ \vdots \\ z_{2n-2} \\ - \; - \; - \; - \; - \; - \\ \boldsymbol{v}(\boldsymbol{z}) \\ - \; - \; - \; - \; - \; - \\ \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} \tag{3.8}$$

Choosing $\boldsymbol{v}$ as a PD term

$$\boldsymbol{v}(\boldsymbol{z}) = -\boldsymbol{K}_d\dot{\boldsymbol{h}} - \boldsymbol{K}_p\boldsymbol{h}, \tag{3.9}$$

with $\boldsymbol{K}_d > 0$, $\boldsymbol{K}_p > 0$ being derivative and proportional gain matrices, respectively, of appropriate dimension will force the outputs $\boldsymbol{h}$ and their time derivative $\dot{\boldsymbol{h}}$ to converge asymptotically to zero and the state $\boldsymbol{x}$ will evolve on the zero dynamics manifold

$$\boldsymbol{\mathcal{Z}} = \left\{ \boldsymbol{x} \middle| \boldsymbol{h}(\boldsymbol{q}) = \boldsymbol{0}, \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = \boldsymbol{0} \right\}, \tag{3.10}$$

Since we know that on the zero dynamics manifold the outputs, the variables $z_1$ to $z_{2n-2}$ are known to be zero and the focus is shifted towards $z_{2n-1}$ and $z_{2n}$. These two coordinates have to be chosen such that an invertible diffeomorphism $\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{x})$ is built. That means, knowledge of $\boldsymbol{z}$ will automatically provide knowledge for $\boldsymbol{x}$. Another constraint when choosing $z_{2n-1}$ and $z_{2n}$ is that the input $\boldsymbol{u}$ does not appear in their time derivatives. This can be formally defined as $\frac{\partial z_{2n-1}}{\partial \boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) = \frac{\partial z_{2n}}{\partial \boldsymbol{x}} \boldsymbol{g}(\boldsymbol{x}) = 0$, since

$$\dot{z}_{2n-1} = \frac{\partial z_{2n-1}}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = \frac{\partial z_{2n-1}}{\partial \boldsymbol{x}} \left( \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x}) \boldsymbol{u} \right) \tag{3.11}$$

Equation (3.11) holds for $\dot{z}_{2n}$ as well. A valid choice is

$$\begin{aligned} \xi_1 &:= z_{2n-1} = \boldsymbol{c}^T \boldsymbol{q} =: \theta(\boldsymbol{q}) \\ \xi_2 &:= z_{2n} = \gamma(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\gamma}_0(\boldsymbol{q}) \dot{\boldsymbol{q}} =: \bar{\sigma}_n(\boldsymbol{q}, \dot{\boldsymbol{q}}), \end{aligned} \tag{3.12}$$

where $\boldsymbol{\gamma}_0(\boldsymbol{q})$ is the $n-$th line of the mass-inertia matrix $\boldsymbol{D}$ and $\xi_2$ is the conjugate angular momentum around the unactuated DoF $q_n$. The vector $\boldsymbol{c}$ has to be chosen such that its $n-$th element is nonzero. Within this thesis and with respect to Fig. 2.5, $\boldsymbol{c}$ is chosen as $\boldsymbol{c} = [0\ 0\ -1\ -0.5\ -1]$. The variable $\theta$ is marked on Fig. 2.5 which is the robot model used in the thesis when the HZD framework is utilized.

Furthermore, the variable $\theta$ in (3.12) is chosen to be monotonically increasing and with that it can replace the time variable $t$, which results in an autonomous closed-loop system. By using $\theta$ as a replacement of time, the original coordinates can be reconstructed as

$$\boldsymbol{q} = \boldsymbol{H}^{-1} \begin{bmatrix} \boldsymbol{h}_d \\ \theta \end{bmatrix} \text{ and } \dot{\boldsymbol{q}} = \boldsymbol{H}^{-1} \begin{bmatrix} \frac{\partial \boldsymbol{h}_d}{\partial \theta} \\ 1 \end{bmatrix} \dot{\theta}, \tag{3.13}$$

where $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_0 \\ \boldsymbol{c} \end{bmatrix}$.

Now that we established a control law to steer the state $\boldsymbol{x}$ on the zero dynamics manifold, we have to check the zero dynamics for stability.

### 3.1.3 Form of the Zero Dynamics

The zero dynamics can be brought into a form that facilitates the derivation of Lagrangian dynamics through the identification of a potential and a kinetic energy function. In turn, the definition of these two functions will allow us to impose a feasibility condition for periodic and aperiodic walking.

In order to be able to introduce Lagrangian dynamics for the zero dynamics we manipulate the expression for $\xi_2$ from Eq. (3.12) to come up with an expression of the form $\dot{\xi}_1 = \frac{1}{I(\xi_1)}\xi_2$, where $I(\xi_1) \neq 0$. For that we write

$$\begin{aligned} \boldsymbol{\gamma_0}\dot{\boldsymbol{q}} &= \xi_2 \Rightarrow \quad \text{(recall Eq. (3.13))} \\ \boldsymbol{\gamma_0}\boldsymbol{H}^{-1}\begin{bmatrix} \frac{\partial \boldsymbol{h}_d}{\partial \theta} \\ 1 \end{bmatrix}\dot{\theta} &= \xi_2 \overset{\dot{\theta}=\dot{\xi}_1}{\Rightarrow} \\ I(\xi_1)\dot{\xi}_1 &= \xi_2 \Rightarrow \\ \dot{\xi}_1 &= \frac{1}{I(\xi_1)}\xi_2. \end{aligned} \tag{3.14}$$

For $z_{2n} = \xi_2$ we can derive an expression of the form $\dot{\xi}_2 = \kappa_2(\xi_1)$ as follows:

$$\begin{aligned} \dot{\xi}_2 &= \left[\frac{\partial \gamma}{\partial \boldsymbol{q}}\ \frac{\partial \gamma}{\partial \dot{\boldsymbol{q}}}\right]\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{D}^{-1}(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}) \end{bmatrix} \quad \text{(recall } \gamma(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\gamma}_0(\boldsymbol{q})\dot{\boldsymbol{q}}) \\ &= \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{\gamma}_0^T}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\gamma}_0\boldsymbol{D}^{-1}(-\boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{G}) \\ &= \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{\gamma}_0^T}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} - \boldsymbol{C}_n\dot{\boldsymbol{q}} - G_n. \end{aligned} \tag{3.15}$$

Now, $\boldsymbol{C}_n = \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{\gamma}_0^T}{\partial \boldsymbol{q}} - \frac{1}{2}\dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{D}}{\partial q_n}$ and since $q_n$ is cyclic, i.e. $\frac{\partial \boldsymbol{D}}{\partial q_n} = 0$, we get $\boldsymbol{C}_n = \dot{\boldsymbol{q}}^T\frac{\partial \boldsymbol{\gamma}_0^T}{\partial \boldsymbol{q}}$. With that,

$$\dot{\xi}_2 = -G_n = -\frac{\partial V}{\partial q_n} = \kappa_2(\xi_1), \tag{3.16}$$

where $V$ is the potential energy function of Eq. (2.29).

### 3.1.4 Lagrangian Dynamics in the Zero Dynamics Manifold

If we differentiate the expression (3.14) for $\dot{\xi}_1$ we get

$$\begin{aligned} \ddot{\xi}_1 &= \frac{\dot{\xi}_2}{I(\xi_1)} - \xi_2\frac{\partial I(\xi_1)}{\partial \xi_1}\frac{1}{I^2(\xi_1)}\dot{\xi}_1 \Rightarrow \\ I^2(\xi_1)\ddot{\xi}_1 + \frac{\partial I(\xi_1)}{\partial \xi_1}I(\xi_1)\dot{\xi}_1^2 - \kappa_2(\xi_1)I(\xi_1) &= 0 \end{aligned} \tag{3.17}$$

In this expression of Lagrangian dynamics, we can identify a kinetic energy function

$$K_{\text{zero}}(\xi_1) = \frac{1}{2}I^2(\xi_1)\dot{\xi}_1^2 \tag{3.18}$$

or

$$K_{\text{zero}}(\xi_2) = \frac{1}{2}\xi_2^2 \tag{3.19}$$

and a potential energy function

$$V_{\text{zero}}(\xi_1) = -\int_{\theta^+}^{\xi_1} I(\xi)\kappa_2(\xi)d\xi \tag{3.20}$$

As a next step we will consider the impact in the zero dynamics description.

### 3.1.5 Impact Invariance

The zero dynamics remain invariant under the impact event of the swing leg with the ground if the state of the system after the application of the impact matrix $\boldsymbol{\Delta}$ remains on the zero dynamics. This can be formally expressed as

$$\boldsymbol{\Delta}\left(\mathcal{S} \cap \mathcal{Z}\right) \subset \mathcal{Z} \tag{3.21}$$

where $\mathcal{S}$ is the impact surface. The intersection $\mathcal{S} \cap \mathcal{Z}$ is the pre-impact system state on the zero dynamics manifold $\boldsymbol{z}^-$. Please recall that as shown with Eq. (2.31) the impact matrix $\boldsymbol{\Delta}$ is a function of $\boldsymbol{x}$ and through the diffeomorphism $\boldsymbol{T}$ can also be written as a function of $\boldsymbol{z}$ once the system state is on the zero dynamics manifold.

Now that the impact invariance has been established we can write the hybrid zero dynamics model of the system (2.30) as

$$\begin{aligned} \dot{\boldsymbol{z}} &= f_{\text{zero}}(\boldsymbol{z}), \;\; \boldsymbol{z}^- \notin \mathcal{S} \cap \mathcal{Z} \\ \boldsymbol{z}^+ &= \boldsymbol{\Delta}\left(\boldsymbol{z}^-\right), \;\; \boldsymbol{z}^- \in \mathcal{S} \cap \mathcal{Z} \end{aligned}$$

and we can proceed with finding conditions such that a walking motion is dynamically feasible and stable.

### 3.1.6 Feasibility and Stability

The feasibility condition states that the post-impact kinetic energy in the zero dynamics manifold $K_{\text{zero}}(\xi_2^+)$ has to be greater than the maximum value $V_{\text{zero}}^{\text{MAX}}$ of the potential energy function. Given that during the impact an exchange of kinetic energy takes place we can write $K_{\text{zero}}(\xi_2^+) = \delta_{\text{zero}} K_{\text{zero}}(\xi_2^-)$, where $\delta_{\text{zero}}$ is the kinetic energy exchange expressed in the zero dynamics. With that, the feasibility condition can be expressed as

$$\delta_{\text{zero}} K_{\text{zero}}(\xi_2^-) > V_{\text{zero}}^{\text{MAX}} \tag{3.22}$$

where the minus superscript in $\xi_2^-$ denotes the value of $\xi_2$ at the end of the motion (before the impact). The constant $\delta_{\text{zero}}$ has an analytic expression and can be computed *a priori* according to the following equation (with the help of (2.27)):

$$\delta_{\text{zero}} = \left( \boldsymbol{\gamma}_0\left(\boldsymbol{q}^+\right) \boldsymbol{\Delta}_q^R\left(\boldsymbol{q}^-\right) \begin{bmatrix} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\left(\boldsymbol{q}^-\right) \\ \boldsymbol{\gamma}_0\left(\boldsymbol{q}^-\right) \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{0}_{n-1} \\ 1 \end{bmatrix} \right)^2 \tag{3.23}$$

At the end of each step the pre-impact kinetic energy $K_{\text{zero}}(\xi_2^-)$ is given by:

$$K_{\text{zero}}(\xi_2^-) = K_{\text{zero}}(\xi_2^+) - V_{\text{zero}}(\xi_1^-) \tag{3.24}$$

Now a return map $\mathcal{P}$ (namely *Poincaré map*) [70] can be defined which will map the zero dynamics on a surface or section (namely *Poincaré section*) transverse to the zero dynamics flow after each step. The Poincaré section will be chosen to be a section of the walking surface and since the zero dynamics are of dimension 2, their intersection with the Poincaré section will be of dimension 1. With that in mind, we can manipulate Eq. (3.24) in order to map the pre-impact kinetic energy of the $k-$th step $K^k_{\text{zero}}(\xi_2^-)$ to the pre-impact kinetic energy of the next step, such that

$$K^{k+1}_{\text{zero}}(\xi_2^-) = \mathcal{P}(K^k_{\text{zero}}(\xi_2^-)) \Rightarrow \text{ (recall that } \mathrm{K}_{\text{zero}}(\xi_2^+) = \delta_{\text{zero}}\mathrm{K}_{\text{zero}}(\xi_2^-)) \tag{3.25}$$

$$K^{k+1}_{\text{zero}}(\xi_2^-) = \delta_{\text{zero}} K^k_{\text{zero}}(\xi_2^-) - V_{\text{zero}}(\xi_1^-) \tag{3.26}$$

The fixed point of the Poincaré map $\mathcal{P}$ is defined as its point of convergence where $K^*_{\text{zero}}(\xi_2^-) = \mathcal{P}(K^*_{\text{zero}}(\xi_2^-))$ and is obtained by setting $K^{k+1}_{\text{zero}}(\xi_2^-) = K^k_{\text{zero}}(\xi_2^-) = K^*_{\text{zero}}(\xi_2^-)$ and solving for $K^*_{\text{zero}}(\xi_2^-)$. Following this approach yields,

$$K^*_{\text{zero}}(\xi_2) = \frac{V_{\text{zero}}(\xi_1^-)}{\delta_{\text{zero}} - 1} \tag{3.27}$$

or in terms of $\xi_2$

$$\xi_2^* = (\cdot)\sqrt{2\frac{V_{\text{zero}}(\xi_1^-)}{\delta_{\text{zero}} - 1}} \tag{3.28}$$

and the domain of definition $\mathcal{D}$ of $\mathcal{P}$ is given by

$$\mathcal{D} = \left\{\xi_2^- | \delta_{\text{zero}} K_{\text{zero}}(\xi_2^-) > V^{\text{MAX}}_{\text{zero}}\right\}. \tag{3.29}$$

The $(\cdot)$ in Eq. (3.28) denotes that the sign is based on the angle notation used (in correspondence to Fig. 2.5 it should be "-"). Finally, if $0 < \delta_{\text{zero}} < 1$ the orbit is exponentially stable and its domain of attraction is $\mathcal{D}$, the whole domain of definition of $\mathcal{P}$.

Please note that the Poincaré map $\mathcal{P}$ can also be defined using the post-impact state of the robot. In that case, the Poincaré map $\mathcal{P}$ is given as:

$$K^{k+1}_{\text{zero}}(\xi_2^+) = \delta_{\text{zero}}\left(K^k_{\text{zero}}(\xi_2^+) - V_{\text{zero}}(\xi_1^-)\right) \tag{3.30}$$

and the fixed point as:

$$\begin{aligned} K_{\text{zero}}(\xi_2^*) &= \frac{\delta_{\text{zero}} V_{\text{zero}}(\xi_1^-)}{\delta_{\text{zero}} - 1} \Rightarrow \\ \xi_2^* &= (\cdot)\sqrt{2\frac{\delta_{\text{zero}} V_{\text{zero}}(\xi_1^-)}{\delta_{\text{zero}} - 1}} \end{aligned} \tag{3.31}$$

Now that the feasibility and stability conditions for periodic walking have been defined, we can proceed with feasibility checks for transitions between periodic orbits as well as aperiodic walking.

### 3.1.7 Transition between Periodic Orbits

Assume we are given two stable periodic orbits $\boldsymbol{\phi}_i(t)$ and $\boldsymbol{\phi}_f(t)$ and the fixed points of their Poincaré Maps correspond to $\boldsymbol{x}^-$, i.e. the state of the robot before the impact. These orbits are periodic solutions of the robot hybrid dynamics described by Eq. (2.33) which include the impact at the end of the motion and fulfill different task objectives. Assuming that

$$\delta_{\text{zero},i} K_{\text{zero}}(\xi_2^-) > V_{\text{zero},i\to f}^{\text{MAX}} \tag{3.32}$$

a transition $\boldsymbol{\phi}_{i\to f}(t)$ between these two orbits is feasible when the post-impact state $\boldsymbol{x}^+$ of the system after the transition is inside the domain of attraction of the target orbit. In the zero dynamics manifold and in a similar fashion to Eq. (3.22) and Eq. (3.32) this condition is expressed as

$$\delta_{\text{zero}}^{i\to f} K_{\text{zero}}(\xi_2^-) > V_{\text{zero},f}^{\text{MAX}} \tag{3.33}$$

$$K_{\text{zero}}(\xi_2^+) > V_{\text{zero},f}^{\text{MAX}} \tag{3.34}$$

The convergence to the target periodic orbit is guaranteed, since $0 < \delta_{\text{zero},f} < 1$.

### 3.1.8 Aperiodic Walking

If the robot transitions between two periodic orbits, but does not settle to any of them, then the robot executes aperiodic walking. The condition that has to be checked in that case is similar to Eq. (3.34) and evaluates if the post-impact kinetic energy in the zero dynamics is greater than the maximum value of the potential energy of the aperiodic motion in the zero dynamics. Assuming that the transition is between the initial kinematic configuration of the periodic orbit $\boldsymbol{\phi}_i(t)$ and the final kinematic configuration of the periodic orbit $\boldsymbol{\phi}_f(t)$, the feasibility condition can be expressed as

$$K_{\text{zero}}(\xi_2^+) > V_{\text{zero},i\to f}^{\text{MAX}}. \tag{3.35}$$

In addition, the energy evolution of the system in the zero dynamics manifold is given as:

$$K_{\text{zero}}^{k+1}(\xi_2^+) = \delta_{\text{zero}}^{i\to f} \left( K_{\text{zero}}^{k}(\xi_2^+) - V_{\text{zero}}^{i\to f}(\xi_1^-) \right) \tag{3.36}$$

Now that we have presented the theory of the HZD, we can proceed with the presentation of the virtual constraints and how they facilitate the design of output functions which respect the feasibility and stability conditions we described within this section.

## 3.2 Bézier Polynomials as Virtual Constraints

The virtual constraints used commonly within the HZD framework are the desired trajectories $\boldsymbol{q}^d$ of the output functions $\boldsymbol{h}$ which are parametrized as Bézier polynomials of order

$M$ with coefficients $\boldsymbol{\alpha}$, such that

$$q_i^d(s) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} s^k (1-s)^{M-k},\ i = 1, ..., n-1,\ k = 0, ..., M \tag{3.37}$$

where $s \in [0,1]$ is a phase variable and is defined as $s(\boldsymbol{q}) = \frac{\theta(\boldsymbol{q})-\theta^+}{\theta^- - \theta^+}$. Please note that $s(\boldsymbol{q})$ is monotonically increasing and therefore replaces time. With that, the desired trajectories $\boldsymbol{q}^d$ become state-dependent.

The Bézier polynomials come with interesting and useful properties:

- They allow for an analytic expressions for the derivative of the desired trajectory $\frac{\partial \boldsymbol{q}_i^d(s)}{\partial s}$.
- The first and last Bézier coefficients can be easily computed by the initial and final desired configuration of the corresponding DoF

$$q_i^d(0) = \alpha_0^i \text{ and } q_i^d(1) = \alpha_M^i.$$

In addition, the computation of the initial and final joint positions and velocities is straightforward with the help of the Bézier coefficients:

- Initial joint positions

$$\boldsymbol{q}^+ = \boldsymbol{H}^{-1} \begin{bmatrix} \boldsymbol{\alpha}_0 \\ \theta^+ \end{bmatrix} \tag{3.38}$$

- Final joint positions

$$\boldsymbol{q}^- = \boldsymbol{H}^{-1} \begin{bmatrix} \boldsymbol{\alpha}_M \\ \theta^- \end{bmatrix} \tag{3.39}$$

- Initial joint velocities

$$\dot{\boldsymbol{q}}^+ = \boldsymbol{H}^{-1} \begin{bmatrix} \frac{M}{\theta^- - \theta^+} (\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_0) \\ 1 \end{bmatrix} \dot{\theta}^+ \tag{3.40}$$

- Final joint velocities

$$\dot{\boldsymbol{q}}^- = \boldsymbol{H}^{-1} \begin{bmatrix} \frac{M}{\theta^- - \theta^+} (\boldsymbol{\alpha}_M - \boldsymbol{\alpha}_{M-1}) \\ 1 \end{bmatrix} \dot{\theta}^- \tag{3.41}$$

Other important relations are the ones that guarantee the impact invariance of the zero dynamics. These relations are defined with the help of the relabelling matrix $\boldsymbol{R}$ and the impact matrix with relabelling included $\boldsymbol{\Delta}_q^R$ as follows:

$$\begin{bmatrix} \boldsymbol{\alpha}_0 \\ \theta^+ \end{bmatrix} = \boldsymbol{H}\boldsymbol{R}\boldsymbol{H}^{-1} \begin{bmatrix} \boldsymbol{\alpha}_M \\ \theta^- \end{bmatrix} \tag{3.42}$$

$$\boldsymbol{\alpha}_1 = \boldsymbol{H}_0 \boldsymbol{\Delta}_q^R \boldsymbol{\omega}^- \frac{\theta^- - \theta^+}{M} \left(\boldsymbol{c}\boldsymbol{\Delta}_q^R \boldsymbol{\omega}^-\right)^{-1} + \boldsymbol{\alpha}_0 \tag{3.43}$$

where

$$\boldsymbol{\omega}^{-} = \boldsymbol{H}^{-1} \begin{bmatrix} \frac{M}{\theta^{-}-\theta^{+}} \left(\boldsymbol{\alpha}_M - \boldsymbol{\alpha}_{M-1}\right) \\ 1 \end{bmatrix} \tag{3.44}$$

In order, however, for (3.43) to hold, $\left(\boldsymbol{c}\boldsymbol{\Delta}_q^R \boldsymbol{\omega}^-\right) \neq 0$ must be true. In addition, since $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are a function of $\boldsymbol{\alpha}_{M-1}$ and $\boldsymbol{\alpha}_M$, the order of the Bézier polynomials has to be chosen as $M \geq 3$.

As a next step we will show how transition motions can be designed with the help of the Bézier coefficients of the initial and target periodic orbits.

## Bézier Coefficients for a Transition Motion

A transition motion between two periodic orbits connects the zero dynamics of each periodic orbit. Assume that the initial periodic orbit of the robot is characterized by the Bézier coefficients $\boldsymbol{\alpha}$, the target periodic orbit is characterized by the Bézier coefficients $\boldsymbol{\beta}$ and the transition motion by the coefficients $(\boldsymbol{\alpha} \to \boldsymbol{\beta})$. In addition, the corresponding controllers that will zero the outputs which are defined through these Bézier coefficients are denoted as $\Gamma_{\boldsymbol{\alpha}}$, $\Gamma_{\boldsymbol{\beta}}$ and $\Gamma_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$. The switching to the controllers $\Gamma_{\boldsymbol{\beta}}$ and $\Gamma_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$ will be synchronized with the impact events.

The objective of the controller $\Gamma_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$ is to map a subset of the one-dimensional manifold $\Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\alpha}}\right)$ into a subset of the one-dimensional manifold $\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\beta}}$. The zero dynamics manifold $\boldsymbol{\mathcal{Z}}_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$ defined by the Bézier coefficients $(\boldsymbol{\alpha} \to \boldsymbol{\beta})$ and parameters $\theta^+_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$, $\theta^-_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$:

$$\begin{aligned}
(\boldsymbol{\alpha} \to \boldsymbol{\beta})_0 &= \boldsymbol{\alpha}_0 \\
(\boldsymbol{\alpha} \to \boldsymbol{\beta})_1 &= \frac{M_{\boldsymbol{\alpha}}}{M_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}} \frac{\theta^-_{\boldsymbol{\beta}} - \theta^+_{\boldsymbol{\alpha}}}{\theta^-_{\boldsymbol{\alpha}} - \theta^+_{\boldsymbol{\alpha}}} \left(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_0\right) + \boldsymbol{\alpha}_0 \\
(\boldsymbol{\alpha} \to \boldsymbol{\beta})_{M_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}-1} &= \frac{M_{\boldsymbol{\beta}}}{M_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}} \frac{\theta^-_{\boldsymbol{\beta}} - \theta^+_{\boldsymbol{\alpha}}}{\theta^-_{\boldsymbol{\beta}} - \theta^+_{\boldsymbol{\beta}}} \left(\boldsymbol{\beta}_{M_{\boldsymbol{\beta}}-1} - \boldsymbol{\beta}_{M_{\boldsymbol{\beta}}}\right) + \boldsymbol{\beta}_{M_{\boldsymbol{\beta}}} \\
(\boldsymbol{\alpha} \to \boldsymbol{\beta})_{M_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}} &= \boldsymbol{\beta}_{M_{\boldsymbol{\beta}}} \\
\theta^+_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})} &= \theta^+_{\boldsymbol{\alpha}} \\
\theta^-_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})} &= \theta^-_{\boldsymbol{\beta}}
\end{aligned} \tag{3.45}$$

will satisfy

$$\boldsymbol{\mathcal{Z}}_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})} \cap \Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\alpha}}\right) = \Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\alpha}}\right) \tag{3.46}$$

and

$$\Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}\right) = \Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\beta}}\right) \tag{3.47}$$

Equation (3.46) states that when switching to $\Gamma_{(\boldsymbol{\alpha}\to\boldsymbol{\beta})}$ at the beginning of the transition motion, the state of the robot belongs to the manifold $\Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\alpha}}\right)$. Equation (3.47) states that when switching to $\Gamma_{\boldsymbol{\beta}}$ after the impact event, the post-impact state of the robot will belong to the manifold $\Delta\left(\mathcal{S} \cap \boldsymbol{\mathcal{Z}}_{\boldsymbol{\beta}}\right)$. Please note that it was assumed that the initial and final periodic orbits have the same switching surface, but the same results and conclusions hold also for the case when the two periodic orbits have different switching surfaces.

The intermediate coefficients can be chosen as

$$(\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta})_i = \frac{\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i}{2}, \;\; i = 2, ..., M_{(\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta})} - 2 \tag{3.48}$$

or can be calculated through optimization.

## 3.3 Designing Virtual Constraints through Nonlinear Static Optimization

The virtual constraints can be computed by solving a static optimization problem. In this thesis we used the optimization problem proposed in [20, 67] to calculate the walking primitives which are used later in chapters 4 and 5. The optimization variables are the last $M-1$ coefficients of the Bézier polynomial of each actuated DoF, since the first two can be computed by Eq. (3.42) and Eq. (3.43). For the optimization problem there are different constraints to be taken into account with respect to the dynamic feasibility, stability, style of motion and task objective which are presented below.

### Constraints

- The vertical component of the GRFs should be positive $F_y > 0$, i.e. the tip of the stance leg is not penetrating the walking surface.
- The friction cone constraint has to be respected, such that the robot is not slipping on the walking surface $-\mu_s F_y \leq F_x \leq \mu_s F_y$.
- The same has to hold for the impact forces too, i.e. $F_2^y > 0$ and $-\mu_s F_2^y \leq F_2^x \leq \mu_s F_2^y$.
- The tip of the swing leg starts behind the tip of the stance leg and concludes its motion in front of the tip of the stance leg. Since the reference frame of the robot is fixed on the stance leg, this constraint can be expressed as

  $$p_2^{x+} < 0 < p_2^{x-} \tag{3.49}$$

  where the superscript ”+” denotes the beginning of the motion and ”-” the end of it.
- The height of the swing leg has to be greater than 0 during the motion and 0 at the beginning and end of the motion. In a compact form this constraint can be expressed as

  $$p_2^y \geq 0 \tag{3.50}$$

  The condition $p_2^{y+} = p_2^{y-} = 0$ can be satisfied by solving $p_2^{y+} = p_2^{y-} = 0$ for $q_5$.
- The post-impact velocity of the vertical component of the tip of the swing leg has to be positive $\dot{p}_2^{y+} > 0$
- A fixed point shall exist, which means that (3.22) has to hold.

- The periodic orbit has to be exponentially stable, i.e. $0 < \delta_{\text{zero}} < 1$ has to hold.
- A task objective can also be imposed as a constraint. In this thesis two different task objectives have been used:
  - the average walking velocity
    $$v_{\text{des}} = \frac{p_2^{x-}}{T}, \tag{3.51}$$
    with $T$ the total duration of the motion
  - and a fixed final configuration $\boldsymbol{p}_2^- = \boldsymbol{p}_{2,\text{des}}^-$.

#### Cost Functions

In this thesis the squared sum of the torques per step is used as a cost function

$$J = \frac{1}{p_2^-(\boldsymbol{\alpha})} \int_0^T \|\boldsymbol{u}(t,\boldsymbol{\alpha})\|_2^2 \text{dt}, \tag{3.52}$$

where the contribution of each torque input can be optionally scaled by a weighting matrix $\boldsymbol{W}$.

As an alternative the mechanical power per step can also be used

$$J_{\text{mech}} = \frac{1}{p_2^-(\boldsymbol{\alpha})} \int_0^T \dot{\boldsymbol{q}}^\top(t,\boldsymbol{\alpha}) \boldsymbol{S} \boldsymbol{u} \text{dt}, \tag{3.53}$$

where the mechanical power of each joint can also be scaled through $\boldsymbol{W}$.

## 3.4 Summary

This chapter presented the theory of hybrid zero dynamics and their usefulness in designing stable gaits for biped robots. The theory is general for different walking robot systems with one degree of underactuation. As it was shown, the zero dynamics can be defined with the help of Lagrangian dynamics, which facilitate the understanding of the theory. Since the zero dynamics manifold is two dimensional, the Poincaré Map becomes one dimensional and together with the domain of attraction and the fixed point they receive an analytic expression. This is a very important property, since usually for arbitrary systems the Poincaré Map can only be arithmetically computed and for that, the integration of the robot dynamics in order to calculate the robot state trajectory is mandatory.

Interesting works for gait analysis can also be found in the works of Zutven and Dehghani *et al.*[71, 72]. In addition, one can use the alternative approach of Djoudi *et al.* for designing periodic orbits for underactuated walking [73]. In our thesis, however, we decided in favor of the HZD framework since in comparison to [73], this framework provides the stabilizing control law once the virtual constraints have been defined.

# 4 Learning with Primitives for Online Motion Generation

Underactuated walking robots are trading off control authority with more efficient and human-like locomotion. In order to achieve these objectives, careful understanding of the dynamics underlying their motion is necessary. By that, the natural dynamics of such a robot can be exploited and utilized in the synthesis of sophisticated feedback controllers, which in turn are able to cope with the aforementioned reduced control authority.

However, the dynamics of such robots are highly nonlinear due to the dynamic coupling of the rotational degrees of freedom. Thus, designing walking motions online is computationally costly. As an alternative, a repertoire of walking primitives can be computed offline and employed online in order to steer the motion of the robot based on different terrain specifications.

The idea of utilizing a database of Motion Primitives in order to plan the motion of a robot has been studied in the literature for different kinds of tasks like balancing and motion planning for an aerial vehicle [74, 75]. In the field of walking robots, an early work can be found in [76], where a database of walking primitives was employed in order to navigate JOHNNIE, a fully-actuated humanoid robot. In [77], a compass gait robot was steered through an environment with obstacles using only 3 asymptotically stable walking primitives. Furthermore, in [45] it was shown how a database of Motion Primitives can be used for motion planning on uneven terrain for a compass gait and a 5-link walking robot. In that work however, the authors assume that the feedback terms of the controller can compensate for an error between the encountered terrain and the final stride height of the chosen primitive.

Depending on the size of the database and the knowledge of the terrain, different motion planning techniques can be used. For cases with full knowledge of the environment, classic graph search algorithms can be utilized, like in [76], [77]. If a large database renders graph search intractable, online predictive approaches can be utilized so that the search horizon is limited to a few steps and new plans can be generated in each step very fast. The challenge however is to know which sequences of Motion Primitives are feasible, since in underactuated walking robots there is always the risk that the robot does not have enough kinetic energy to overcome the potential energy barrier. Concrete guaranties whether a sequence of such primitives is feasible or not are given in subsection 3.1.8. In addition, worth mentioning is the work on LQR-trees [78], where reachable sequences of controllers can be generated, even though the region of attraction for each controller is estimated in a conservative way using Sum-of-Squares optimization.

Another challenge is that, even with a very large database of Motion Primitives, there will always be cases in which there is no primitive corresponding to the current terrain height [45]. In this chapter we alleviate this limitation. In order to do so, a regression technique for the generation of walking primitives is introduced and utilized when the database does not contain a primitive which corresponds to the specific terrain variation. This is achieved by designing a database of primitives for walking, ascending and descending stairs and then training a Gaussian Process which can generate new gaits. Thus, we are able to reduce the cardinality of the database, in order to use efficient motion planning techniques and still react to unforeseen terrain variations. An interesting evaluation of different regression methodologies (including the Gaussian Process Regression) with respect to constraint violation for a fully actuated biped can be found in [79].

The Motion Primitives which we will use are generated with the help of the Hybrid Zero Dynamics framework. With this methodology we will be able to evaluate the feasibility of transition motions between Motion Primitives. Another advantage is that the motion planning utilizes the 2-dimensional state representation of the zero dynamics instead of the full system state which is always higher-dimensional and with that the planning can be executed online. As already mentioned in section 2.4, the robotic model under study is the 5-link biped, which constitutes a minimalistic model capable of walking on uneven terrain. Please note that the methodology can be generalized to any robotic system with one degree of underactuation.

The structure of this chapter is as follows: In section 4.1 the structure of the Motion Primitives is introduced. In section 4.2, the learning process is explained and evaluated. In section 4.3, the motion planning algorithm is introduced which is based on a best first approach. Section 4.4 presents two sample cases that validate our approach. Section 4.5 concludes the chapter with a summary.

## 4.1 Motion Primitives

For the definition of the Motion Primitives we are using the virtual constraints which were described in section 3.2. These virtual constraints are parametrized as Bézier polynomials of order $M$ and have been already presented in section 3.2. Nonetheless, they are presented once more here for clarity:

$$q_i^d(s) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} s^k (1-s)^{M-k}, \tag{4.1}$$

where $s$ is a phase variable and is defined as $s(\boldsymbol{q}) = \frac{\theta(\boldsymbol{q})-\theta^+}{\theta^- - \theta^+}$. Please note that $s(\boldsymbol{q})$ is monotonically increasing and therefore it replaces time.

The coefficients $\boldsymbol{\alpha}$ are obtained using the optimization process described in section 3.3 and are used in the definition of the Motion Primitives.

### 4.1.1 Database of Motion Primitives

The number of primitives in the database is dependent on the expected terrain variations and possibly on the computational resources available. For the proposed methodology, the cardinality of the database does not need to be large, since the regression technique - as is going to be presented in section 4.2 - is able to enrich the capabilities of a small database by generating new primitives when they are needed.

The important quantities in the Motion Primitives and their explanations are presented in Table 4.1. After defining the role of each term in Table 4.1, a Motion Primitive $P$ is formally defined as the tuple

$$P = \left\{\boldsymbol{\alpha}, \theta^-, V_{\text{zero}}, V_{\text{zero}}^{\text{MAX}}, \delta_{\text{zero}}, \ell^-, h^-\right\} \tag{4.2}$$

and describes both periodic and aperiodic primitives.

**Tab. 4.1:** Structure of a Motion Primitive

| Quantity | Explanation |
|---|---|
| $\boldsymbol{\alpha}$ | The Bézier coefficients, which describe the desired trajectories and the feedforward control input (3.6) |
| $\theta^-$ | Pre-impact value of $\theta$. It is used to define $s(\boldsymbol{q})$ |
| $V_{\text{zero}}$ | Used for the computation of $K_{\text{zero}}^+$ during motion planning according to (3.30) (periodic walking) or (3.36) (aperiodic walking) |
| $\delta_{\text{zero}}$ | Used for the computation of $K_{\text{zero}}^+$ during motion planning according to (3.30) (periodic walking) or (3.36) (aperiodic walking) |
| $V_{\text{zero}}^{\text{MAX}}$ | Necessary in order to evaluate the feasibility of a transition according to (3.32) and (3.33) (periodic walking) or (3.35) (aperiodic walking) |
| $\ell^-$ | Final stride length. Even though it can be reconstructed by the Bézier coefficients $\boldsymbol{\alpha}$ and the value $\theta^-$, it is desirable to be included in the primitive definition in order to facilitate the motion planning and primitive generation. For brevity $\ell^-$ is used instead of $p_2^{x-}$ |
| $h^-$ | Final stride height. The justification is the same as for the stride length $\ell^-$. For brevity $h^-$ is used instead of $p_2^{y-}$ |

In this work, the cost $J$ from Eq. (3.52) associated with each primitive is not included in the definition (4.2), since it describes the total cost of the associated periodic orbit and during aperiodic walking the state $\boldsymbol{x}$ does not evolve necessarily on the periodic orbit.

## 4.2 Regression Method

In this section, the regression method is presented, which enables the online generation of walking patterns, when there is no primitive in the database which matches the terrain variation.

For walking on uneven terrain the input is the desired final stride length $\ell^-$ and height $h^-$. The outputs are the Bézier coefficients $\boldsymbol{\alpha}$, except the first two, since according to (3.38) and (3.40) they depend on the post-impact state of the robot $\boldsymbol{x}^+$ and can be easily determined by solving these two equations for $\alpha_0$ and $\alpha_1$. In this work, the learning model employed is the Gaussian Process Regression, due to its fast learning and inference time and the fact that it was able to learn the nonlinear dependencies between the gaits in a very satisfying way. A brief introduction to the Gaussian Process follows.

### 4.2.1 Gaussian Process

The Gaussian process $\mathcal{GP}$ [80] is a stochastic process in which any linear combination of the input variables $\boldsymbol{\varphi}_i,\ i = 1, ..., N$ has a joint Gaussian distribution $\mathcal{N}$. In our case, each of the Bézier coefficients $\boldsymbol{\alpha}_{2:M}^{1:n-1}$ will be predicted from a Gaussian process $\mathcal{GP}$ which can be written as

$$\boldsymbol{\alpha}_{2:M}^{1:n-1}(\boldsymbol{\varphi}) \sim \mathcal{GP}(m(\boldsymbol{\varphi}), \kappa(\boldsymbol{\varphi}, \boldsymbol{\varphi}_i)), \tag{4.3}$$

where $\boldsymbol{\varphi} = [\ell^-\ h^-]^\top$ is the input, $m$ the mean function and $\kappa$ the covariance function. The mean and covariance functions can also be defined as

$$m(\boldsymbol{\varphi}) = \mathbb{E}[\boldsymbol{\alpha}_{2:M}^{1:n-1}] \tag{4.4}$$

$$\kappa(\boldsymbol{\varphi}, \boldsymbol{\varphi}_i) = \mathrm{cov}[m(\boldsymbol{\varphi}), m(\boldsymbol{\varphi}_i)]. \tag{4.5}$$

The joint Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\mathcal{K}})$ is defined by a covariance matrix $\mathcal{K}_{ij} = \kappa(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)$ of dimension $N \times N$, where $N$ is the number of existing primitives and a mean $\boldsymbol{\mu} = [m(\boldsymbol{\varphi}_1), m(\boldsymbol{\varphi}_2), ..., m(\boldsymbol{\varphi}_N)]^\top$. The behaviour of the output functions is determined by the covariance kernels. It is common practice to employ a zero mean function $\boldsymbol{\mu}$ and a squared exponential covariance kernel given by

$$\kappa(\boldsymbol{\varphi}, \boldsymbol{\varphi}_i) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{\varphi} - \boldsymbol{\varphi}_i)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\varphi} - \boldsymbol{\varphi}_i)\right), \tag{4.6}$$

where $\sigma_f^2$ is the length scale and $\boldsymbol{\Sigma} = \sigma_\varphi^2 \boldsymbol{I}$ the standard deviation with $\boldsymbol{I}$ the identity matrix. The quantities $\sigma_f$ and $\sigma_\varphi$ are combined in $\boldsymbol{\lambda} = [\sigma_f\ \sigma_\varphi]^\top$ and are referred to as the hyperparameters of the Gaussian Process which are estimated through maximum likelihood methods.

### 4.2.2 Evaluation of the Regression Method for Periodic Gaits

The regression method is trained offline and can be utilized online for motion planning. The task constraints which were used for the computation of the Motion Primitives according to the optimization problem described in section 3.3 are the final step length $\ell^-$ and the height $h^-$ of the swing foot. The optimization was utilized for a grid from 0.4 to 0.7 m for $\ell^-$ and from -0.15 to 0.10 m for $h^-$. The step increment for $\ell^-$ is 0.05 m and for $h^-$ it is 0.01 m. With these specifications, the database contains 182 periodic primitives as well as all the transitions between them, where the transitions coefficients are chosen according to the methodology described in section 3.2.

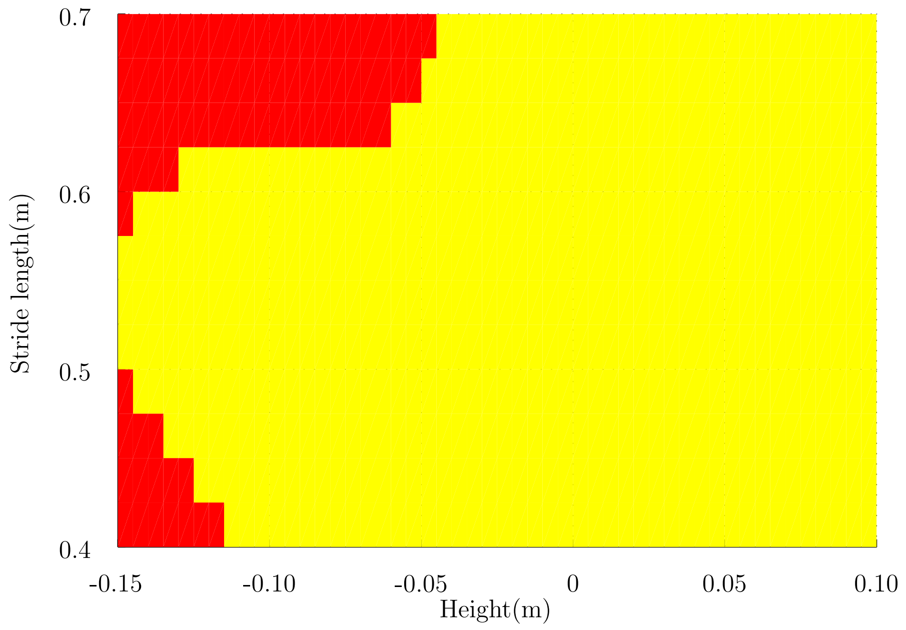


**Fig. 4.1:** The range of values for which the regression method can generate feasible gaits. Yellow color indicates feasible gaits, while red indicates infeasible ones.

The important quantities when a generated gait is employed are $\delta_{\text{zero}}$, $V_{\text{zero}}$, $V_{\text{zero}}^{\text{MAX}}$ and the justification is the same as in Table 4.1. The verification of the feasibility condition for periodic walking described by Eq. (3.22) is illustrated in Fig. 4.1 and the quantity $\delta_{\text{zero}}$ is presented in Fig. 4.2 for a densely sampled grid of points corresponding to the stride length and height of the Motion Primitives in the database. For a gait to be periodic and exponentially stable, Eq. (3.22) as well as $0 < \delta_{\text{zero}} < 1$ must hold. Since, the latter holds for the whole grid, the interest is shifted towards Fig. 4.1. There, it is obvious that the regression method can generate periodic gaits for a very large range of $\ell^-$ and $h^-$ values.

A limitation arises for stair descent with a very large or small step length. At this range, the nonlinear relations between the Bézier coefficients of the periodic gaits are highly uncorrelated and cannot be easily captured by the Gaussian Process. This finding was expected, since during stair descent the robot is taking advantage of gravity and utilizes more of its natural dynamics. Therefore the range of solutions is very small for these gaits. This finding is also supported by the fact that as the height decreases, the range of step lengths corresponding to periodic gaits decreases.

### 4.2.3 Online Generation of Motion Primitives

After the description and analysis of the regression methodology, it is time to discuss its online capabilities. In order to obtain the value of the unactuated DoF $q_5$ at the end of the gait, we only need to equate the vertical component $y_s$ of the swing leg with the desired final stride height $h^-$, i.e. solve the equation $y_s(\boldsymbol{\alpha}_M, q_5^-) = h^-$ for $q_5^-$. This ensures that the generated gaits will always match the encountered terrain variations. In order to compute $V_{\text{zero}}$ and $V_{\text{zero}}^{\text{MAX}}$ very fast, the function $V_{\text{zero}}$ is computed on a sufficiently dense grid of points of $\xi_1$, so that it can be numerically integrated using the cumulative sum. Since the function $V_{\text{zero}}$ corresponds to potential energy, it is concave which makes the computation

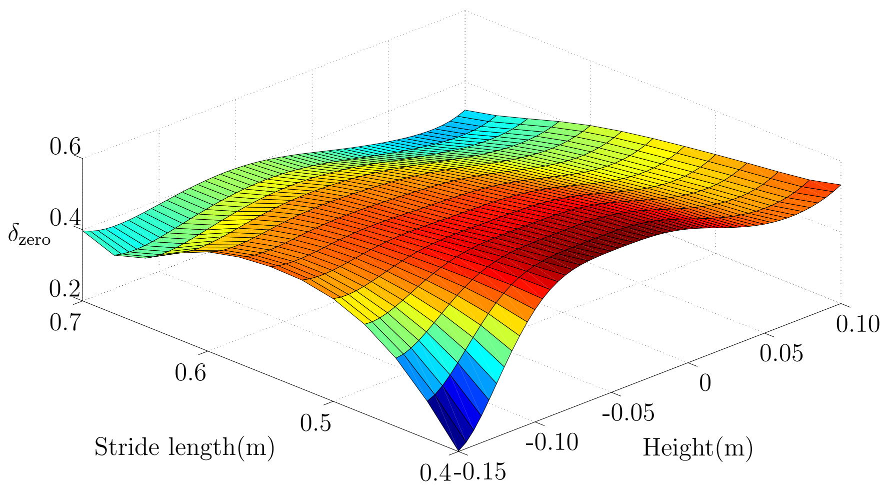


**Fig. 4.2:** The quantity $\delta_{\text{zero}}$ for the generated walking gaits.

of $V_{\text{zero}}^{\text{MAX}}$ very easy and fast. The computation of $\delta_{\text{zero}}$ requires only a simple evaluation of Eq. (3.23).

More precisely, regarding the computational cost, the Gaussian Process prediction of the mean has an algorithmic complexity of $O(N)$, where $N$ in this case is the total number of training examples. Regarding the dynamic feasibility, the cumulative sum for integrating the function $V_{\text{zero}}$ is a linear operation which depends on the size of the grid of $\xi_1$ points. Finding the maximum $V_{\text{zero}}^{\text{MAX}}$ is also a linear operation.

On the other hand, finding a feasible solution online using optimization constitutes a very difficult problem. The decision variables are $(M-1)\times(n-1)$ and we have nonlinear constraints regarding the feasibility and the stability of the gait. Even if the optimization is only with respect to the constraints, i.e. finding a set of decision variables that respects the terrain variation and satisfies the constraints, the Gaussian Process is going to produce a gait much faster.

Finally, for aperiodic walking an evaluation of the feasibility of transitions is not helpful, since it is subject to the post-impact state of the robot, which is based on the history of transitions executed so far. A solution to this problem is presented in section 4.3, where the feasibility is checked online with the aid of a look-ahead approach.

## 4.3 Motion Planning Algorithm

The motion planning algorithm takes as input the terrain description and gives as an output a sequence of primitives which can traverse this terrain. It is based on a best first approach. This is due to the limited computation time, which is dictated by the duration of the gait. In a few words, we want the motion planning algorithm to terminate before the robot concludes its current step. The sketch of the algorithm is presented in Fig. 4.3. The search node of the list 'TREE' is assumed to have the structure described in Table 4.2.

Initially the structure TREE contains the currently executed node $p$. At each execution step we choose the best node in TREE. The evaluation criterion assumed for selecting the

**Tab. 4.2:** Structure of a search node

| Quantity | Justification |
|---|---|
| P | Associated primitive as described in table 4.1 |
| $K^+_{\text{zero}}$ | Necessary for checking the feasibility of transitions and choosing primitives |
| predecessor | Father of current node in the search tree |
| k | Depth of the current node in the search tree |
| (x,y) | Cartesian coordinates of the swing leg on the terrain |
| ID | Takes discrete values from the set {'N','G'}, denoting nominal and generated node respectively |

best node $d$ is defined as

$$d \leftarrow \arg\max_i\{K^+_{\text{zero},i}\} \tag{4.7}$$

The motivation behind this comes from Eq. (3.35). That is, with a greater value of $K^+_{\text{zero}}$, the number of available primitives gets larger. The maximum allowed depth of the TREE is $D$ and thus we check if the best node is within this depth.

- If this is true, we backtrack in TREE to find the predecessor $f$ of $d$ in depth 1. If the execution of $f$ will bring the robot past the goal value "terrainX", then the algorithm is terminated. Otherwise, the structure TREE is re-initialized when the robot concludes its current step with the node $f$, whose primitive is going to be executed.
- If this is not true, we check further the database for suitable primitives to populate the structure TREE.

This is done by iterating along all the $(\ell^-, h^-)$ combinations corresponding to primitives in the database in order to find the ones that match the terrain variations. For that, the ID information of the best node $d$ is important.

- If the best node is a nominal one, the primitives $P_i$ that correspond to the terrain variations already exist in the database.
- If the best node is a generated one, the quantities $\delta_{\text{zero}}$, $V_{\text{zero}}$ and $V^{\text{MAX}}_{\text{zero}}$ of the "connecting" primitives $P_i$ whose final kinematic configuration corresponds to the terrain variations need to be calculated online, since they do not exist in the database.

If there is no primitive in the database that matches the terrain variations, we utilize the Gaussian Process to generate primitives. This procedure is executed for a grid of $\Lambda$ equally distributed step lengths $\ell^-$ in the range $\left[\ell^-_{\text{MIN}}, \ell^-_{\text{MAX}}\right]$. The desired value $h^-$ is equal to the terrain variation in a distance $\ell^-$ from the stance leg of the robot.

The dynamic feasibility of the generated primitives is checked by computing the quantities $V_{\text{zero}}$ and $V^{\text{MAX}}_{\text{zero}}$ as described in subsection 4.2.3 and then evaluating Eq. (3.35). In any case, dynamically feasible primitives are appended in TREE. Finally, it should be

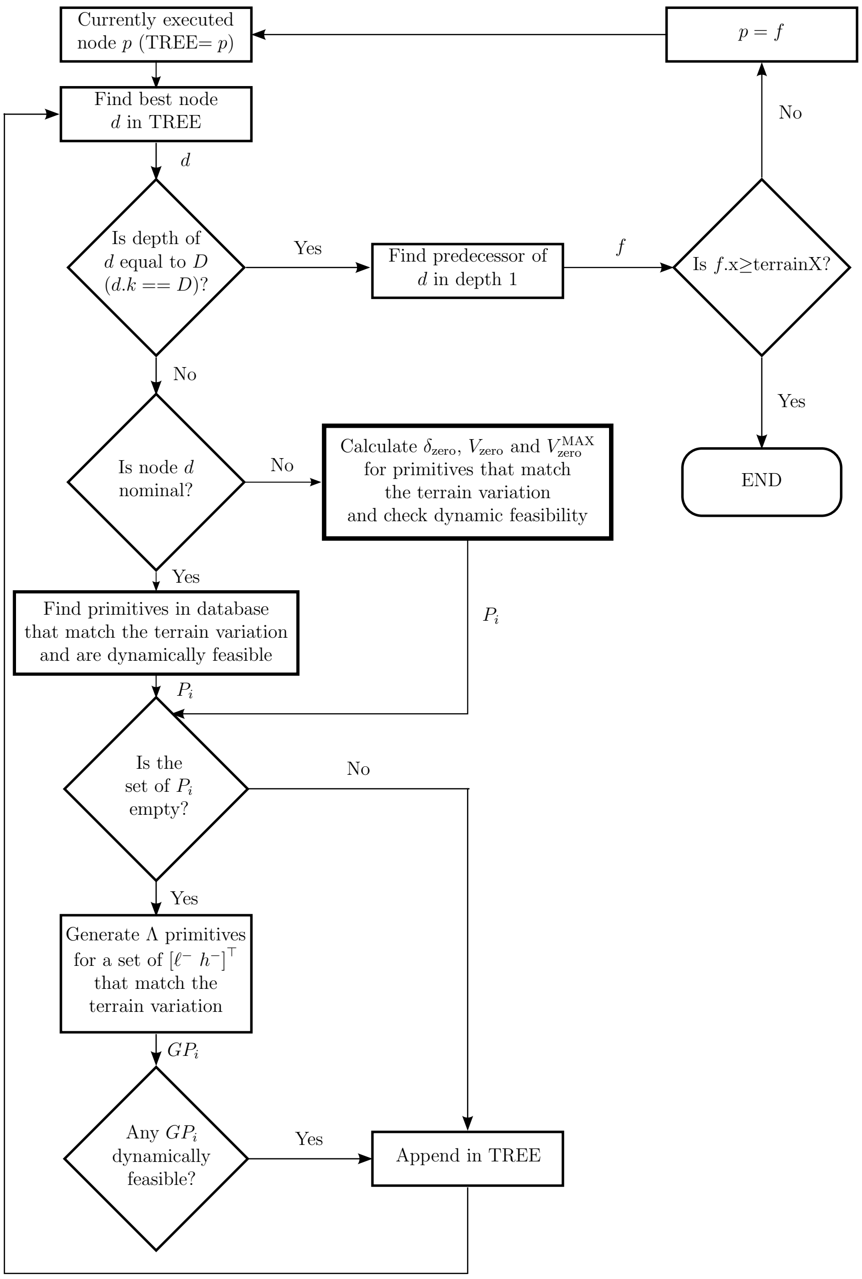

Currently executed node $p$ (TREE= $p$)
$p = f$
Find best node $d$ in TREE
No
$d$
Is depth of $d$ equal to $D$ ($d.k == D$)?
Yes
Find predecessor of $d$ in depth 1
$f$
Is $f$.x≥terrainX?
No
Yes
Is node $d$ nominal?
No
Calculate $\delta_{\text{zero}}$, $V_{\text{zero}}$ and $V_{\text{zero}}^{\text{MAX}}$ for primitives that match the terrain variation and check dynamic feasibility
END
Yes
Find primitives in database that match the terrain variation and are dynamically feasible
$P_i$
$P_i$
Is the set of $P_i$ empty?
No
Yes
Generate $\Lambda$ primitives for a set of $[\ell^- \ h^-]^\top$ that match the terrain variation
$GP_i$
Any $GP_i$ dynamically feasible?
Yes
Append in TREE

**Fig. 4.3:** Flow diagram describing the motion planning algorithm. The dynamic feasibility is checked for each primitive that is either chosen from the database or generated from the Gaussian Process.

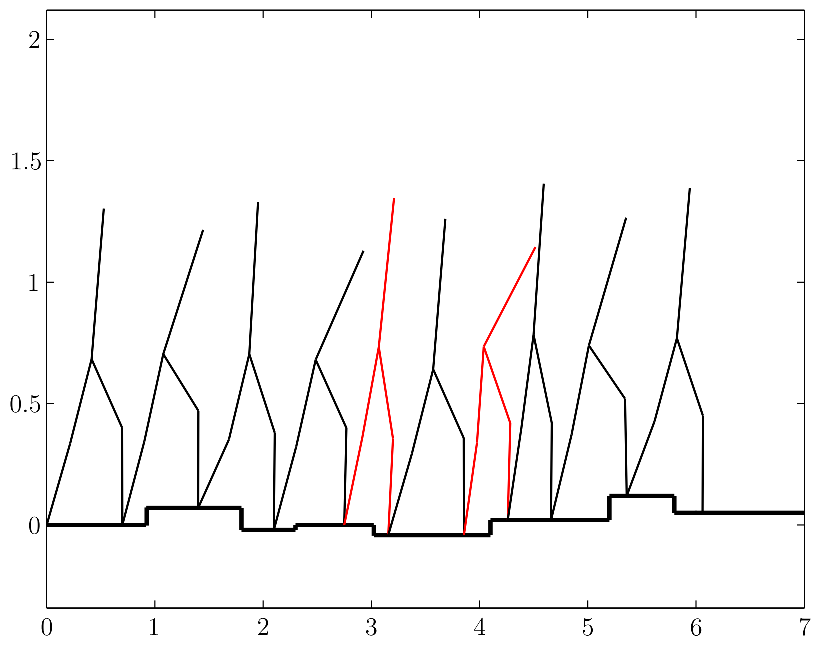


**Fig. 4.4:** Walking sequence produced with the algorithm proposed in section 4.3. The black configurations correspond to gaits that match the aforementioned grid, while the red ones do not.

noted that $D$ and $\Lambda$ are design parameters that dependent on the computational resources available.

## 4.4 Simulation Evaluation

This section presents an evaluation of the algorithm proposed in section 4.3.

In Fig. 4.4 a horizon of $D$=3 steps ahead is utilized and the total amount of primitives which are allowed to be generated by the Gaussian Process is $L$=7. The red configurations denote final robot poses with a $[\ell^- \; h^-]^\top$ specification that does not belong to the aforementioned grid and the regression technique had to be utilized for the transition to them. Since the algorithm involves only lower dimensional dynamics and the inference time of the Gaussian Process is small, the proposed methodology can be utilized for online motion planning. An interesting result arises when the second generated gait shows that the robot leans forward in order to gain more momentum and overtake a big step.

In Fig. 4.5 another evaluation is presented for a more challenging terrain. For this case, the robot has to take an initial step with a height equal to 0.099 m followed by a step with a height equal to -0.148 m. The step down generated transition (2nd red configuration) has a final stride length $\ell^-$ of 0.4039 m and a final stride height $h^-$ of $-0.148$ m. According to Fig. 4.1 a periodic gait with this specification cannot be generated. This example shows that even though the generation of such a periodic gait is infeasible, a feasible transition that ends up in a pose with the specified $\ell^-$ and $h^-$ values can be generated.

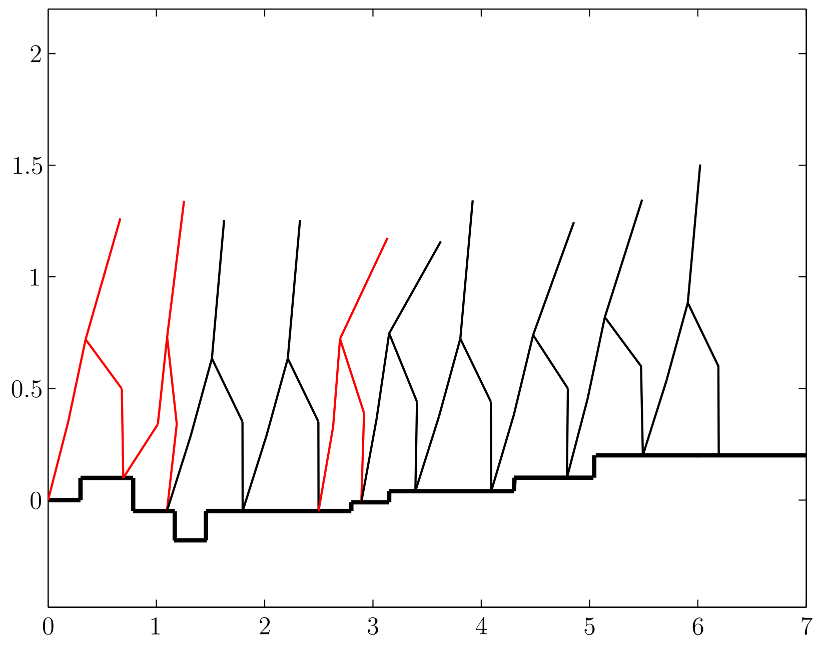


**Fig. 4.5:** Walking sequence for a terrain with a challenging transition at the beginning. The Gaussian Process has to be employed to take a transition from a step with a height of 0.099 m to another one with a height of -0.148 m.

## 4.5 Summary

This chapter proposed an online motion planning algorithm for walking on uneven terrain, using Motion Primitives which are extracted based on the Hybrid Zero Dynamics approach. The key idea is that, when there is a mismatch between the final stride height of the Motion Primitives in the database and the terrain height, a regression technique can be used to generate a primitive that matches the terrain variation. The regression method is a Gaussian Process and it uses the Motion Primitives in the database as training examples. The motion planning algorithm is shown to be efficient since it uses the HZD of the robot which are 2-dimensional and the inference time of the Gaussian Process is very small. Simulation studies are presented in section 4.4.

# 5 Settling Time Reduction Using Sequences of Motion Primitives

Underactuated robots are systems that possess less actuators than degrees of freedom (DoFs). In that case the controller design becomes a challenging task, since the control signals to the actuated DoFs need to induce a motion that stabilizes the whole system. Despite this challenge, there are numerous successful applications for a wide range of systems using various techniques. In the case of academic example systems, there is the swing-up control of the Acrobot which uses partial feedback linearization and LQR control [81]. For free-flying mechanical systems like quadrotors, there is a successfully applied framework for position control and trajectory tracking based on backstepping and sliding mode control [82]. In addition, recent interesting applications emerged from the field of ship-mounted cranes where stabilizing control is achieved through a nonlinear controller design based on the method of Lyapunov [83, 84]. Last but not least, as we will see in the sequel, there are also different approaches in the field of walking robots, which is the main focus of this thesis.

Periodic walking for an underactuated robots is dictated by a periodic orbit. Stable periodic orbits have a domain of attraction where any deviation from the nominal orbit can be compensated by the dynamics of the system and the feedback terms of the control law. As a consequence, for the feasibility of a transition between two periodic orbits, it is sufficient that the state of the system enters the domain of attraction of the target orbit. When that happens, convergence is guaranteed and its rate is dictated by the maximum eigenvalue of the Poincaré Map. The time until convergence to the target periodic orbit is defined as the settling time.

Even though convergence can be guaranteed, it might be desired to improve its rate. In the case of velocity control, for example, one might want the state of the robot to converge to the target periodic orbit as soon as possible, such that the target velocity is acquired very fast. Another reason might be that the target periodic orbit is optimized with respect to an energy criterion and fast convergence to it allows the robotic system to comsume less power. In general, a target periodic orbit is usually designed to satisfy a task objective and faster convergence to the periodic orbit means that the task is fulfilled faster. In such cases, a possible solution is to try to minimize the maximum eigenvalue of the Poincaré Map associated with the target periodic orbit, as was done for a running robot model and a somersaulting simulated robot [14, 85]. In these two related works, the cost function to be minimized is the maximum eigenvalue of the Poincaré map itself. As suggested however in the mentioned works this minimization constitutes a difficult problem. One of the reasons is that the maximum eigenvalue function of the non-symmetric Jacobian matrix of the Poincaré Map is non-differentiable and possibly even non-Lipschitz at points where multiple eigenvalues coalesce. In addition, different modifications of existing optimization algorithms were required for the success of this optimization. Another issue is that this minimization objective might be used against the satisfaction of other objectives such as torque consumption and/or minimum foot clearance.

Another possible solution to improve the settling time could be given by the concept of Virtual Model Control (VMC), where the virtual force could be used to make the robot transition between different walking cycles [41]. Despite the fact that the VMC is generally applicable and very promising, it is not accompanied by concrete stability properties. In addition, with this methodology we do not converge to a desired periodic orbit per se, but instead allow the robot to walk in a fashion that satisfies the task objective encoded in the virtual force. Finally, this methodology might induce high torques, such that the motion is energy inefficient or it violates the actuator limits. A recent methodology which can also employ virtual forces can be found in the work of Veer *et al.* but as it can be seen the settling time is slow [86].

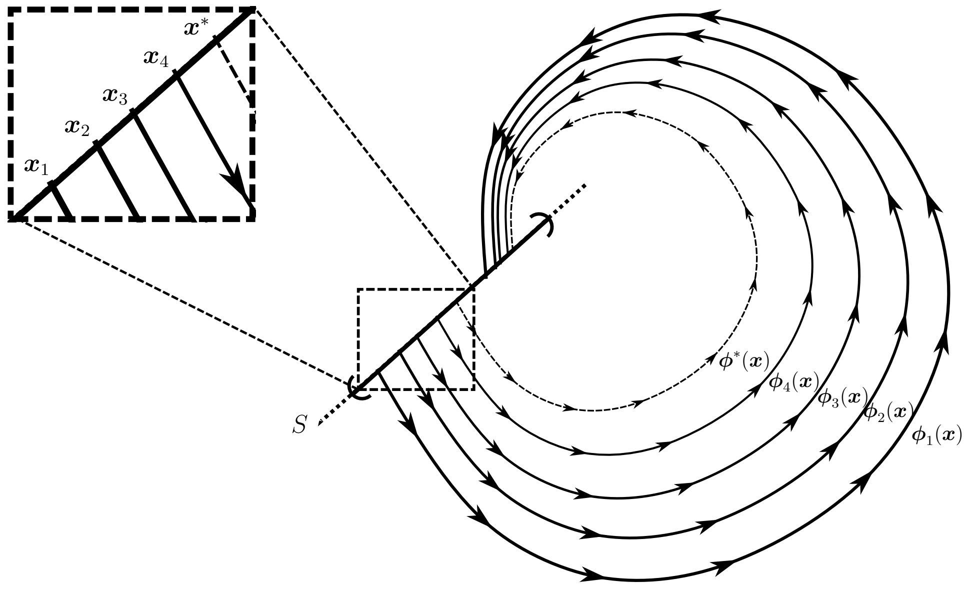
In this chapter we introduce with section 5.1 the Settling Time Reduction problem with the help of Optimal Control. As a later step in section 5.2 we propose a Reinforcement Learning algorithm in order to find sequential controllers which enforce multi-step sequences that can minimize the settling time of different transitions within a database of Motion Primitives. Finally, section 5.3 concludes this chapter with a short discussion.

## 5.1 Optimal Control Formulation

One alternative to Settling Time Reduction without the shortcomings of the VMC and the minimization of the maximum eigenvalue is to enter the domain of attraction of the target periodic orbit very close to the fixed point of the Poincaré map. Then, convergence to the periodic orbit will require less crossings of the Poincaré section, i.e. less time (see Fig. 5.1). If such an alternative is to be undertaken, a concrete feasibility guarantee is required which in order to be provided, knowledge of the domain of attraction of the target periodic orbit is required. The reason for this requirement is due to the fact that this condition characterizes a transition as feasible if and only if the state of the robot is in the domain of attraction of the target periodic orbit after the impact with the ground.

In general, computing the domain of attraction of a periodic orbit is challenging. In order to overcome this challenge we utilize the framework of HZD. One more reason why we utilize the HZD is because we want to compare our approach with the transition methodology proposed within the HZD [47], which is also described in section 3.2.

The assumption is made that the periodic orbits are given and we focus on the problem of transitioning between them in a way that reduces the settling time to the target periodic orbit. In order to compute this aperiodic transition motion, we employ Optimal Control to formulate and solve the Settling Time Reduction problem. The Optimal Control approach is adopted since it allows us to easily impose all the constraints that ensure a feasible, valid and natural motion and there are already examples of its successful utilization in walking robots [76, 13]. The initial state of the robot, state and actuators limits as well as modeling assumptions are treated as constraints. The feasibility condition is introduced as an inequality constraint imposed at the end of the motion taking into account the impact that occurs when the swing leg touches the ground. The cost function penalizes deviations from the fixed point of the Poincaré Map of the target periodic orbit in the zero dynamics



$\boldsymbol{x}^*$
$\boldsymbol{x}_4$
$\boldsymbol{x}_3$
$\boldsymbol{x}_2$
$\boldsymbol{x}_1$
$S$
$\boldsymbol{\phi}^*(\boldsymbol{x})$
$\boldsymbol{\phi}_4(\boldsymbol{x})$
$\boldsymbol{\phi}_3(\boldsymbol{x})$
$\boldsymbol{\phi}_2(\boldsymbol{x})$
$\boldsymbol{\phi}_1(\boldsymbol{x})$

**Fig. 5.1:** Different orbits around an arbitrary periodic orbit $\boldsymbol{\phi}^*(\boldsymbol{x})$ marked with the dashed line. The fixed point of the periodic orbit is denoted with $\boldsymbol{x}^*$ and the Poincaré section with $S$ which in case of walking robots is usually chosen to be the walking surface, i.e. the state of the robot when the tip of the swing leg impacts the ground (either pre- or post-impact state). In addition, the domain of attraction is denoted with the curled brackets on the Poincaré section $S$. As is shown, if we are able to enter the domain of attraction closer to the fixed point $\boldsymbol{x}^*$, less iterations towards the periodic orbit will be required until convergence.

manifold and avoids torques with large magnitude, while keeping the transition time low.

The results are compared - as mentioned above - against the one-step approach proposed by Yang *et al.* [47] (see Fig. 5.2) where the transition motion is designed with the help of the Bézier coefficients of the initial and target orbits and show that our methodology of optimizing a transition improves the settling time. Even though we demonstrate the usefulness of our approach with the example of a 5-link underactuated robot, the methodology for Settling Time Reduction can be applied to any underactuated walking robot with one unactuated DoF. Of course, our approach of Settling Time Reduction is not only limited to the case study of walking, but can also be extended to the case of running [36, 87, 88, 89].

The section is structured as follows: In subsection 5.1.1 we introduce the OPC and in subsection 5.1.2 we present our numerical results. The section concludes with a discussion in subsection 5.1.3 regarding alternative ways to construct transition motions.

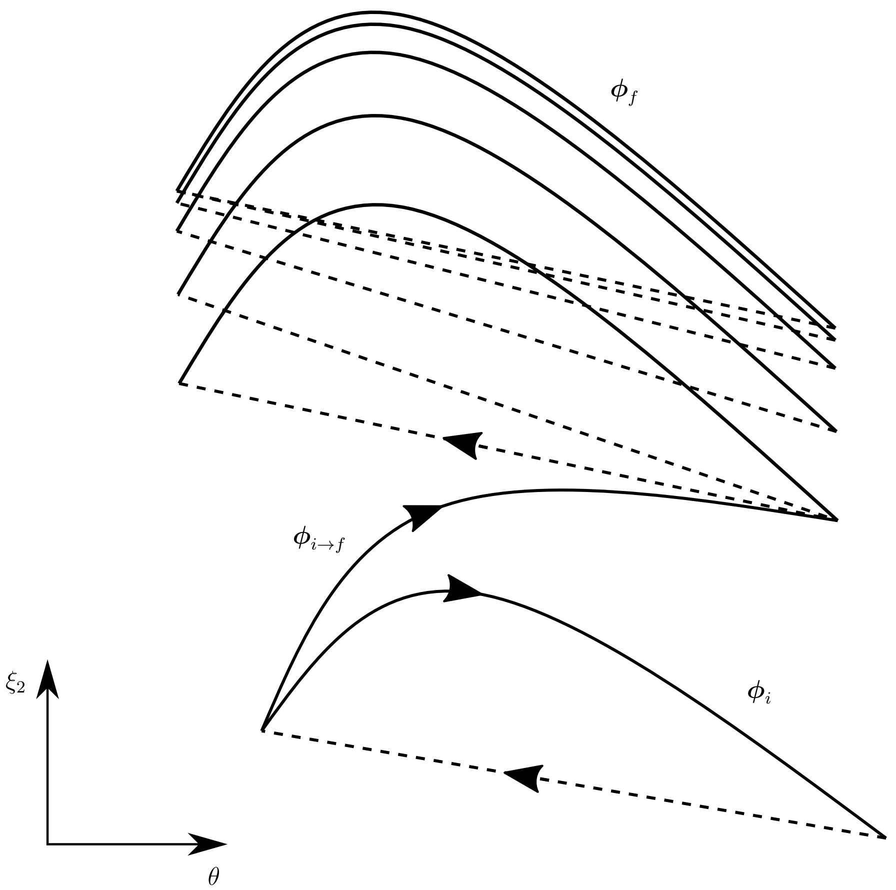


**Fig. 5.2:** Example one-step transition according to [47]. A transition motion $\phi_{i\to f}$ is taken towards the target periodic orbit $\phi_f$. Once this transition is executed, the state of the robot will iterate towards $\phi_f$ until convergence to the periodic orbit.

## 5.1.1 Settling Time Reduction as an Optimal Control Problem

The objective of the Settling Time Reduction is to find a transition motion $\phi_{i\to f}(t)$ (from the initial periodic orbit $\phi_i$ to the final/target periodic orbit $\phi_f$) s.t.

- (3.33) is fulfilled, i.e. $\delta_{\text{zero}}^{i\to j} K_{\text{zero}}(\xi_2^-) > V_{\text{zero},f}^{\text{MAX}}$
- its overall duration $T$ is relatively small and
- the distance $\|\dot{\boldsymbol{q}}_{i\to f}(T) - \dot{\boldsymbol{q}}_f(T)\|$ is minimized.

Such a transition motion will bring relatively fast the state of the system $\boldsymbol{x}$ inside the domain of attraction of $\boldsymbol{\phi}_f$ and is expected to drive the state very close to the fixed point $K_{\text{zero}}^*$ or $\xi_2^*$ of the HZD Poincaré Map of $\boldsymbol{\phi}_f$.

Please note that for the Optimal Control approach the fixed point is chosen with respect to the pre-impact state of the robot. This decision was made, because since we are working with the robot equations of motion and not the HZD, a multiplication of the terminal joint velocities $\dot{\boldsymbol{q}}_{i\to f}(T)$ with the impact matrix $\boldsymbol{\Delta}_q$ would be necessary in the formulation of the cost function as it will be shown later. This multiplication would however introduce additional and unnecessary complexity in the problem formulation and therefore it was rejected.

In order to solve the Settling Time Reduction problem with Optimal Control, we first have to define constraints that have to be fulfilled by the transition motion and a cost function which when minimized attaches the desired characteristics to the motion.

### Constraints

In an Optimal Control Problem (OCP) different equality and inequality constraints are imposed involving the system state and controls. These constraints can be either linear or nonlinear. The ones related to the Settling Time Reduction are listed and described below:

- The OCP is always subject to the dynamics of the system that are described by (2.29)
- The initial state of the transition is the initial state of $\boldsymbol{\phi}_i(0)$ such that

$$\boldsymbol{x}_0 = \boldsymbol{\phi}_i(0). \tag{5.1}$$

- The terminal state is constrained only in terms of the joint positions $\boldsymbol{q}_T$, i.e.

$$\boldsymbol{q}_T = \boldsymbol{q}_f(T) \tag{5.2}$$

  which correspond to the terminal joint positions of $\boldsymbol{\phi}_f$. The joint velocities $\dot{\boldsymbol{q}}$ are not included because of the underactuation that makes it difficult to reach them exactly, preventing us from forming a 2 point boundary value problem. Instead, we leave them free and penalize their deviation from the desired ones $\dot{\boldsymbol{q}}_f(T)$ in the cost function.

- Since we are dealing with systems with impact effects, we have to impose a constraint that ensures that the impact is valid. In order to so, the impact force $\boldsymbol{F}_2$ has to respect the friction cone constraint

$$-\mu_s F_2^y \le F_2^x \le \mu_s F_2^y \tag{5.3}$$

and the vertical impact force component $F_2^y$ has to be positive

$$F_2^y > 0. \tag{5.4}$$

According to (2.23) the matrix $\boldsymbol{\Delta}_F$ is a function of the terminal configuration of the robot. As a consequence, since we know the desired terminal joint positions $\boldsymbol{q}_T = \boldsymbol{q}_f$ and the Cartesian coordinates of the foot of the swing leg $p_2^x = 0$ and $p_2^y = 0$, we can compute the desired elements of the matrix $\boldsymbol{\Delta}_F$ at the end of the transition motion. With that, the constraints on the impact force $\boldsymbol{F}_2$ can be expressed linearly in $\dot{\boldsymbol{q}}_T$. In order to express the constraint (5.4) linearly to $\dot{\boldsymbol{q}}_T$, we decompose the desired impact matrix to its lines as $\boldsymbol{\Delta}_F = \left[\boldsymbol{\Delta}_F^{\top,1}\ \boldsymbol{\Delta}_F^{\top,2}\right]^\top$. In such a way, the constraints on the impact force $\boldsymbol{F}_2$ can be written as

$$-\left(\mu_s \boldsymbol{\Delta}_F^2 - \boldsymbol{\Delta}_F^1\right)\dot{\boldsymbol{q}}_T \leq 0 \tag{5.5}$$

$$-\left(\mu_s \boldsymbol{\Delta}_F^2 + \boldsymbol{\Delta}_F^1\right)\dot{\boldsymbol{q}}_T \leq 0 \tag{5.6}$$

$$\boldsymbol{\Delta}_F^2 \dot{\boldsymbol{q}}_T > 0. \tag{5.7}$$

- The friction cone constraint that holds for the impact force $\boldsymbol{F}_2$ has to hold also for the ground reaction forces $\boldsymbol{F}$ during the transition, i.e.

$$-\mu_s F_y \leq F_x \leq \mu_s F_y \tag{5.8}$$

and the vertical force component $F_y$ has to be positive:

$$F_y > 0. \tag{5.9}$$

- The transition must bring the state of the robot inside the domain of attraction of the target orbit as dictated by (3.33). This condition can be expressed linearly to the terminal joint velocities $\dot{\boldsymbol{q}}_T$ in the following way: At first, the maximum value of the potential energy at the zero dynamics manifold $V_{\text{zero},f}^{\max}$ is known. In addition, we need to multiply the last line of the mass-inertia matrix $\boldsymbol{D}_n$ with the post-impact joint velocities $\dot{\boldsymbol{q}}^+$. Using (2.24) and excluding the last two lines, the post-impact joint velocities are given as $\dot{\boldsymbol{q}}^+ = \boldsymbol{\Delta}_q^R \dot{\boldsymbol{q}}_T$. The matrix $\boldsymbol{\Delta}_q^R$ is a function of the terminal configuration of the robot and therefore the desired elements of $\boldsymbol{\Delta}_q^R$ can be computed. This is due to the fact that the desired terminal configuration $\boldsymbol{q}_f(T)$ is known and for the tip of the stance leg it holds that $p_2^x(T) = p_2^y(T) = 0$. Finally, as already mentioned we need the last line of the mass-inertia matrix $\boldsymbol{D}_n$ which can be evaluated using the post-impact joint positions $\boldsymbol{q}^+$ whose desired values are $\boldsymbol{q}_f(0)$. Bringing everything together yields

$$\boldsymbol{D}_n(\boldsymbol{q}_f(0))\boldsymbol{\Delta}_q^R(\boldsymbol{q}_f(T))\dot{\boldsymbol{q}}_T < -\sqrt{2V_{\text{zero},f}^{\max}}. \tag{5.10}$$

- The swing foot has to be above the ground during the transition motion:

$$p_2^y(\boldsymbol{q}) > 0. \tag{5.11}$$

  The desired final and initial posture of the robot guarantees that $p_2^y = 0$ for $t = 0$ and $t = T$.

- The state $\boldsymbol{x}$, input $\boldsymbol{u}$ and total duration of the motion $T$ should satisfy the following box constraints:

$$\boldsymbol{x}_{\text{min}} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\text{max}} \tag{5.12}$$

$$\boldsymbol{u}_{\text{min}} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\text{max}} \tag{5.13}$$

$$T_{\text{min}} \leq T \leq T_{\text{max}} \tag{5.14}$$

  The constraint on the state $\boldsymbol{x}$ is imposed to ensure the satisfaction of physical and style constraints. The input $\boldsymbol{u}$ is constrained such that we do not exceed any imposed actuator limits. Finally, the time duration $T$ is constrained to prevent the solution search space from unreasonably increasing.

### Cost function

As already described, the terminal joint velocities $\dot{\boldsymbol{q}}_T$ are free and we penalize their deviation from the pre-impact joint velocities $\dot{\boldsymbol{q}}_f(T)$ of the orbit $\boldsymbol{\phi}_f$. At the same time, we want to penalize the final time $T$ such that we enter the domain of attraction of the target periodic orbit not only close to the fixed point, but also fast. Finally, we want the magnitude of the torques to be low, such that "energy" criteria are also taken into account and a smooth transition motion is produced. Therefore the employed cost function is given by

$$J_{\text{STR}}(\boldsymbol{u}, T) = \underbrace{\left(\dot{\boldsymbol{q}}_f - \dot{\boldsymbol{q}}_T\right)^\top \boldsymbol{\Gamma} \left(\dot{\boldsymbol{q}}_f - \dot{\boldsymbol{q}}_T\right)}_{\|\xi_2^* - \xi_2\|^2} + \alpha_J T + \int_0^T \boldsymbol{u}^\top \boldsymbol{W} \boldsymbol{u} \tag{5.15}$$

where $\boldsymbol{\Gamma} = \boldsymbol{\gamma}_0^\top(\boldsymbol{q}_f(T))\boldsymbol{\gamma}_0(\boldsymbol{q}_f(T))$. The first term is equal to $\|\xi_2^* - \xi_2\|^2$ and penalizes deviations from the fixed point of the target periodic orbit, but through $\xi_2$ instead of $K_{\text{zero}}$. The use of $\xi_2$ retains the information of the fixed point and prevents the term from becoming unnecessarily complicated. The constant $\alpha_J$ penalizes large values of the total time duration $T$ and the matrix $\boldsymbol{W}$ has weighting and scaling purposes and favors or disfavors "energy" consumption against convergence error and total time duration. These weighting factors assist in determining how much different control inputs are penalized or how much the deviation from the desired fixed point is penalized.

#### Optimal Control Formulation

The problem of finding a transition motion between two periodic orbits such that the settling time to the target periodic orbit is minimized can be formulated as an OCP as

$$\begin{aligned} \min_{\boldsymbol{u},T} \quad & J_{\mathrm{STR}}(\boldsymbol{u}, T) \\ \text{s.t.} \quad & \text{dynamics model (2.29)} \\ & \text{equality \& inequality constraints (5.1) – (5.14)} \end{aligned}$$

For the solution of this OCP we used the ACADO software [90]. This software implements a direct multiple shooting algorithm with an equidistant grid for control discretization. In such a way, the dynamic optimization problem is transformed into a static Nonlinear Program. For more details on multiple shooting algorithms, refer to further readings [91].

### 5.1.2 Numerical Results

In this section we provide numerical results by solving the presented problem for a transition from a periodic orbit corresponding to a velocity of 1 m/s to one that enables the robot to walk with a velocity of 1.5 m/s. The robot model used is the 5-link biped as depicted in Fig. 2.5 and the periodic orbits that dictate walking with the aforementioned velocities have been designed using the optimization methodology as described in section 3.3.

In Tab. 5.1 we compare the convergence error between the transition methodology from Yang *et al.*[47] and our OCP approach for the transition under study. For this comparison, we use different combinations of the weighting factors in the cost function $J_{\mathrm{STR}}$ (5.15), where we choose a diagonal $\boldsymbol{W}$ matrix with the same diagonal value $w$. The convergence error is defined as

$$e_K = |K^{-}_{\text{zero}} - K^{*}_{\text{zero}}|. \tag{5.16}$$

and as expected, the less we penalize the magnitude of the torques, the less the convergence error of the OCP approach gets in comparison to the one of the approach of Yang *et al.* The penalization of the total time duration $\alpha_J$ has not the same impact as $w$ but we can observe an advantageous behavior for $\alpha_J = 10$. In the sequel, we will discuss the behavior

**Tab. 5.1:** $e_{K,\mathrm{OCP}}/e_{K,\mathrm{one-step}} - 1$

| $\alpha_J$ \ $w$ | 0.001 | 0.01 | 0.1 |
|---|---|---|---|
| 1 | −22.67% | −8.90% | −2.78% |
| 10 | −24.93% | −9.25% | −2.76% |
| 100 | −24.38% | −6.64% | −2.59% |

Relative difference of the convergence error when using the OCP approach and one step approach. The relative difference is presented with respect to the weighting constants $\alpha_J$ and $w_i$.

of the transition motion for the values $\alpha_J = 50$ and $w = 0.05$. Regarding the torques,

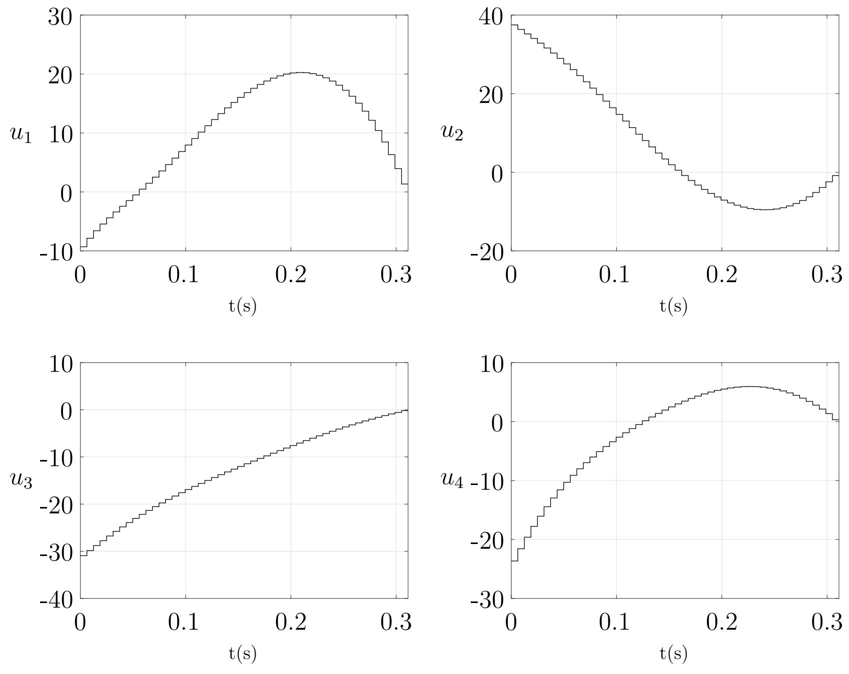


**Fig. 5.3:** Joint torques for the considered transition which is the outcome of the solution of the OCP with $\alpha_J = 50$ and $w = 0.05$.

in Fig. 5.3 we see that the lowest torque requirements are for the knee of the swing leg $u_1$, since it has to manipulate only the relatively smaller mass of the tibia of the swing leg and the greatest effort is taken by the hips, i.e. $u_2$ and $u_3$ where the peak values are observed at the beginning of the motion. This behavior shows that the robot is trying to generate enough momentum at the beginning of the motion to reach a higher velocity fast. The torque profile of the knee of the stance leg $u_4$ exhibits also a similar behavior. A nice feature of the generated motion is that all torques settle close to zero at the end of the motion. As also seen in Fig. 5.3, the peak torques are always inside the assumed limits of $\pm 100$ N·m. Finally. as shown in Fig. 5.4, the friction cone constraint is always respected for the assumed static friction coefficient of $\mu_s = 0.7$.

We also provide a stick diagram animation of the motion (Fig. 5.5) where it can be seen that the robot in order to accelerate fast to the desired velocity of 1.5 m/s is utilizing the inertia of its torso and concludes the transition with the torso leaned forward. Finally, we show the values of the matrix $\mathbf{\Gamma}$ (see (5.15))

$$\mathbf{\Gamma} = \begin{bmatrix} 0.0011 & 0.0016 & 0.6073 & 0.3094 & 0.7512 \\ 0.0016 & 0.0023 & 0.8897 & 0.4532 & 1.1004 \\ 0.6073 & 0.8897 & 339.7983 & 173.1101 & 420.2884 \\ 0.3094 & 0.4532 & 173.1101 & 88.1909 & 214.1158 \\ 0.7512 & 1.1004 & 420.2884 & 214.1158 & 519.8447 \end{bmatrix}$$

where it is shown that most of the non-diagonal elements have large values and as a

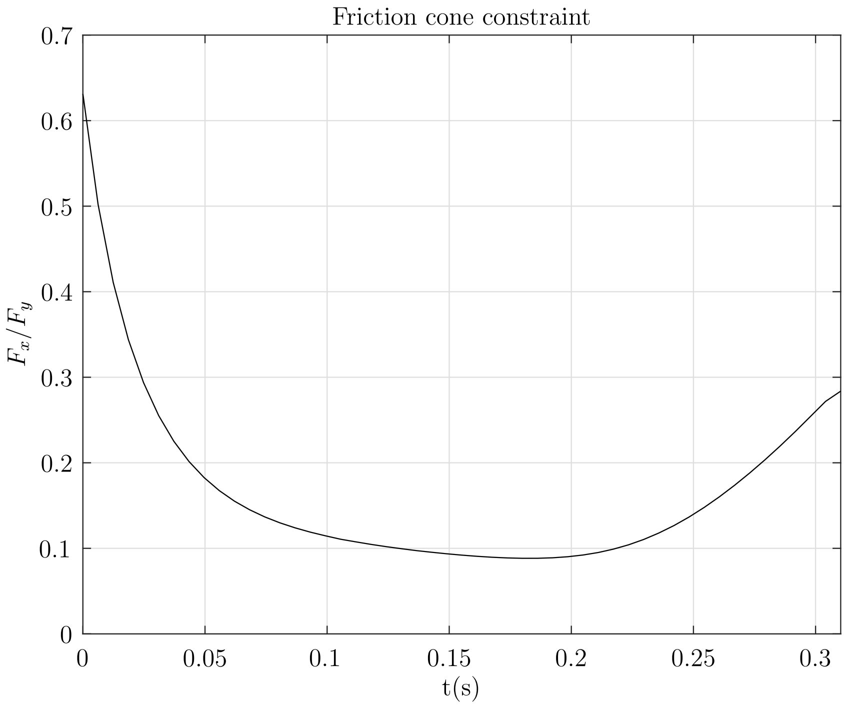


**Fig. 5.4:** Friction cone constraint for the considered transition which is the outcome of the solution of the OCP with $\alpha_J = 50$ and $w = 0.05$.

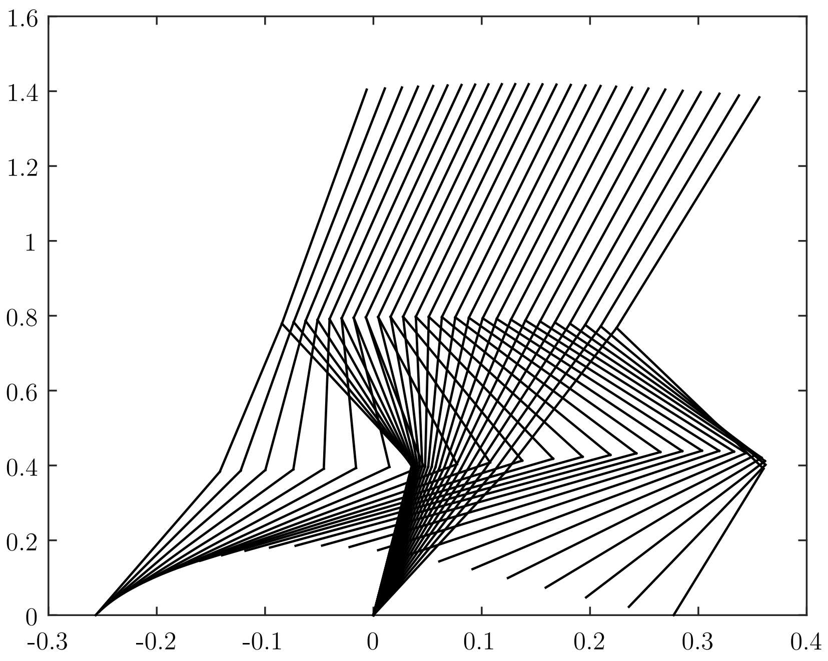


**Fig. 5.5:** Stick diagram animation of the considered transition which is the outcome of the solution of the OCP with $\alpha_J = 50$ and $w = 0.05$.

consequence there is a coupling between the corresponding deviations of the terminal joint velocities. The stronger coupling however is between the deviation for the DoFs of the stance leg ($q_3$ and $q_4$) and the unactuated DoF $q_5$, both between themselves and each other. This finding is expected since $q_3$ and $q_4$ belong to the stance leg which manipulates the whole mass of the robot and together with $q_5$ they have a major contribution in the determination of the pre-impact state of the robot and its distance from the desired fixed point $\xi_2^*$.

### 5.1.3 Discussion

As was presented in section 3.2, a transition between two stable periodic orbits can be designed with Bézier polynomials of $M-$th order which are employed for each actuated DoF [47]. In this Bézier polynomial based approach, the first two and the last two coefficients of each actuated DoF are determined such that the initial and terminal state of the transition are valid with respect to the pre-impact and post-impact states of $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_f$, respectively. The remaining $M-3$ coefficients are calculated by averaging the corresponding coefficients of the two periodic orbits ($4 \times (M-3)$ coefficients in total). As stated in the literature, they can also be found through optimization. We decided against such an approach due to the following reason.

As stated in the book of Agoston (Chapter 11) [92], when employing Bézier polynomials, the further a control point is from a point on the curve, the smaller its effect on that point is. By fixing the first and last two coefficients of each polynomial, we have less freedom on shaping the desired trajectories such that they satisfy the aforementioned constraints (especially (5.3) - (5.10)). As a consequence, this approach provides limited user influence on the distance from the fixed point $\xi_2^*$ of $\boldsymbol{\phi}_f$ and by that, on the settling time to the periodic orbit of $\boldsymbol{\phi}_f$.

## 5.2 Reinforcement Learning Approach

Another way of dealing with the problem of Settling Time Reduction is to consider a sequence of steps that will enable the system to enter the domain of attraction of the target orbit closer to its fixed point, than if a one-step transition was taken as suggested in [47] and section 3.2. When such methods are utilized, the feasibility of the step sequence has to be investigated, in order to ensure that the sequence will indeed drive the state of the system in the domain of attraction of the target orbit.

The feasibility problem has been treated in [78] and [93]. In [78], the idea of LQR-Trees is introduced, where the state space is partitioned in different regions. Each region corresponds to the domain of attraction of a LQR controller which is computed based on a conservative approach using Sum-of-Squares optimization. Thus, the system can be driven to a final state from any initial one by finding a sequence of LQR controllers. This idea is claimed to be extendable to walking robots as well. In [93] the framework of Sequential Composition Control is introduced. The idea is more generic than the LQR-Trees, since it describes how any kind of controllers can be composed into a sequence in order to accomplish a high-level plan. The feasibility condition states that a transition between different controllers is feasible only when the image of the funnel of one controller belongs to the domain of attraction of the other one.

In related work [94], the idea of Sequential Composition Control was used to define feasibility conditions for transitions between different controllers. When a transition was not feasible, a connecting controller was learnt online. Despite the fact that these ideas describe ways to create a composite controller as a concatenation of different ones, they do not take into account any optimality criteria for the way this composite controller is generated, but only focus on feasibility. In this thesis we overcome this limitation by expressing the problem of controller composition as a Markov Decision Process and by introducing a reward function that takes into account optimality criteria related to the task under study, which is the Settling Time Reduction.

In our case study we assume a set of walking motions and consider the problem of reducing the settling time of a transition between different periodic orbits. Due to the underactuation, the feasibility of a transition is not pre-determined but rather has to be verified at each impact event (i.e. swing leg establishing contact with the ground). The Markov Decision Process for finding composite controllers is solved with a realization of Reinforcement Learning and leads to multi-step transition motions. A contribution of using the Hybrid Zero Dynamics framework is that the proposed methodology can be extended to any walking robot with one degree of underactuation. In addition, due to the utilization of the Hybrid Zero Dynamics the state of the Markov Decision Process has lower dimensionality than the one of the robot system.

The section is structured as follows: In subsection 5.2.1 we formulate the problem of finding multi-step transitions as a Markov Decision Process and the Reinforcement Learning algorithm is presented. The simulation results are presented in subsection 5.2.2.

### 5.2.1 Learning for Settling Time Reduction

The purpose of this work is to find a sequence of controllers/motions in order to reduce the settling time of a transition from an initial periodic $\boldsymbol{\phi}_{\text{init}}$ to another one $\boldsymbol{\phi}_{\text{target}}$. In order to do so, we formulate the problem of finding multi-step transitions which can reduce the settling time of the aforementioned transition as a Markov Decision Process, which we solve with Reinforcement Learning. For this work, each orbit corresponds to a desired average walking velocity.

#### One-step Transition

In the one-step approach and under the assumption that Eq. (3.32) holds, a transition between two different periodic orbits is realized by checking first if Eq. (3.34) holds. If this is the case, the transition motion is executed and then the state of the robot will enter the domain of attraction of the target orbit. Otherwise if Eq. (3.34) does not hold, an intermediate transition is taken towards the orbit whose domain of attraction can be reached by the initial orbit and is closest (in terms of fixed point) to the target one. If the target orbit is still unreachable, this process can be repeated. In any case, once in the domain of attraction of the target orbit, the convergence to the fixed point is dictated by the quantities $\delta_{\text{zero}}$ and $V_{\text{zero}}(\theta^-)$, thus it is possible that convergence requires a lot of time and steps.

### Multi-Step Transition

In this work, this slow convergence is confronted by requiring that the transition to the target orbit $\boldsymbol{\phi}_{\text{target}}$ does not have to be realized following the one-step approach, since there might be a sequence of transitions which is enforced by a composite controller and can potentially reduce the settling time (see Fig. 5.6). More formally, assume that in the multi-step transition the state of the robot is commanded towards the target orbit $\boldsymbol{\phi}_{\text{target}}$ at time $t^\star$ and $K^*_{\text{zero}}$ denotes the fixed point of the periodic orbit $\boldsymbol{\phi}_{\text{target}}$ which is chosen as the post-impact kinetic energy in the zero dynamics manifold. Since the robot might be executing a step at time $t^\star$ under the one-step transition, we allow it to conclude the step and measure $K^+_{\text{zero}}$ at time $t^o > t^\star$.

**Definition 5.2.1** A composite controller is successful if and only if

$$|K^+_{\text{zero}}(t^\star) - K^*_{\text{zero}}| \leq |K^+_{\text{zero}}(t^o) - K^*_{\text{zero}}|, \tag{5.17}$$

□

This evaluation criterion is adopted because it is desirable to evaluate the two different transition strategies on the pre-impact state of the robot. Since though, the impact events between the two strategies are not synchronized, the one-step strategy is allowed to complete the currently executed step and get closer to the fixed point $K^*_{\text{zero}}$.

A sequence of transitions is equivalent to a composite controller, since - as seen in section 3.2 - each transition $\boldsymbol{\phi}_{i\to j}$ is commanded by a controller $\Gamma_{(\boldsymbol{\alpha}_i \to \boldsymbol{\alpha}_j)}$, where $(\boldsymbol{\alpha}_i \to \boldsymbol{\alpha}_j)$ the Bézier coefficients of the transition motion.

If the state of the system under a multi-step transition crosses the Poincaré section of the target orbit closer to the fixed point than the one-step transition, the principle of optimality can be utilized and claim that convergence is achieved faster. In order to find such multi-step transitions, we use Reinforcement Learning.

### Reinforcement Learning

Reinforcement Learning has been proposed as a semi-supervised optimization method [95] and has been extensively used in the field of Robotics [96]. An optimization problem in the context of Reinforcement Learning can be defined as a Markov Decision Process described by the tuple $\mathcal{P}(\mathcal{S}, \mathcal{T}, \mathcal{F}, \rho, \chi)$, where $\mathcal{S}$ is the state space, $\mathcal{T}$ is the action space, $\mathcal{F} : \mathcal{S} \times \mathcal{T} \to \mathcal{S}$ is the state transition function, $\rho$ is the reward function and $\chi \in [0, 1]$ is a discount factor. The state transition function $\mathcal{F}$ returns the state $\boldsymbol{s}_{k+1}$ when applying the action $\tau_k$ at state $\boldsymbol{s}_k$. The reward function returns a scalar value $\rho_k$ after such a transition. The action selection is dictated by a policy $\pi$, such that $\pi : \mathcal{S} \to \mathcal{T}$.

The idea behind Reinforcement Learning is to find this policy $\pi$ such that the expected discounted sum of future rewards is maximized. This expected discounted sum is represented by a state-action value function $Q^\pi : \mathcal{S} \times \mathcal{T} \to \mathbb{R}$, such that

$$Q^\pi(\boldsymbol{s}_k, \tau_k) = \mathbb{E}\left(\sum_{i=0}^{\infty} \chi^i r_{k+i+1}\right), \tag{5.18}$$

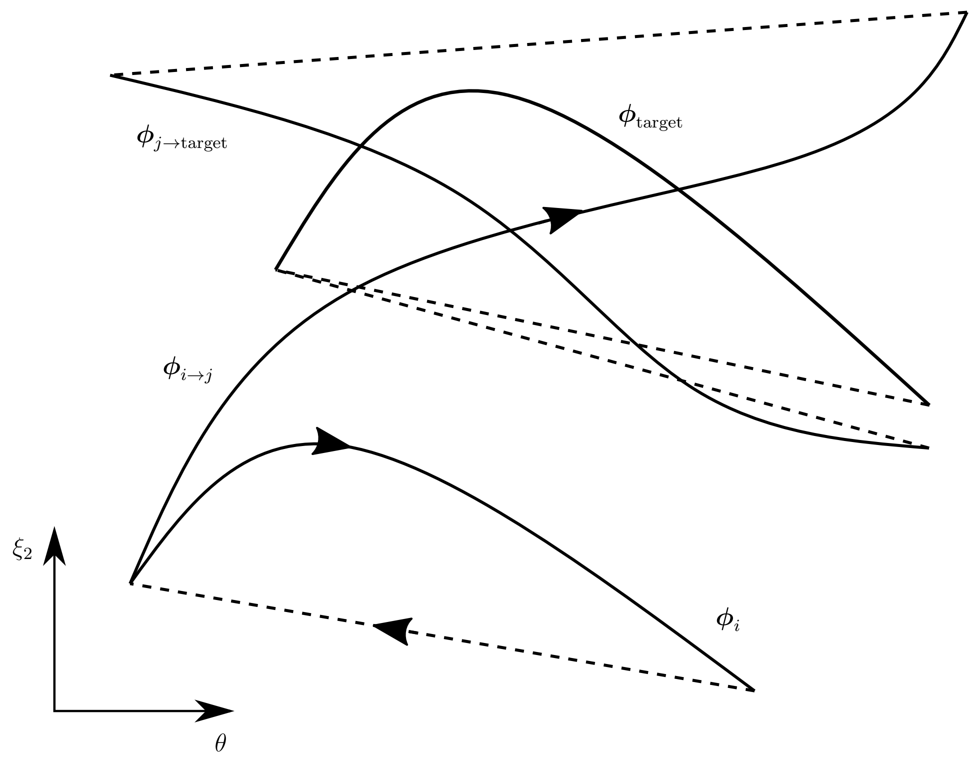


**Fig. 5.6:** Example multi-step transition with only one intermediate transition. Instead of having a transition motion $\phi_{i \to \text{target}}$, an intermediate transition is taken towards the periodic orbit $\phi_j$. Once the transition motion $\phi_{i \to j}$ is executed, the state of the robot does not iterate towards $\phi_j$, but executes another transition towards the target periodic orbit $\phi_{\text{target}}$. Once this transition is executed, the state of the robot will iterate towards $\phi_{\text{target}}$ until convergence to the periodic orbit.

where $\mathbb{E}$ the expectation operator. In other words, we are trying to find an optimal policy $\pi^*$ such that

$$\pi^* \in \arg\max_\tau Q^*(\boldsymbol{s}, \tau) \tag{5.19}$$

where $Q^*(\boldsymbol{s}, \tau) = \max_\pi Q^\pi(\boldsymbol{s}, \tau)$.

Different realizations have been proposed in the literature for finding such a policy. When dealing with large state-action spaces, a practical solution is to use approximation techniques in order to learn the $Q^\pi$ function. In this work, a $Q$-learning method is adopted with linear parametrization such that $Q^\pi(\boldsymbol{s}_k, \tau_k) = \boldsymbol{\varphi}^T(\boldsymbol{s}_k, \tau_k)\boldsymbol{v}$ and $\epsilon$-greedy action selection, where a random action is taken with probability $\epsilon$ and an optimal one with probability $1-\epsilon$. In this linear parametrization of $Q^\pi$, the parametrization of the state is given by the vector $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, ..., \varphi_Z]^T$, where each $\varphi_i$ corresponds to a basis function and $Z$ is the total number of basis functions. The parameter vector to be learnt is $\boldsymbol{v} \in \mathbb{R}^Z$.

In the Reinforcement Learning framework, the state-action space has to be defined for the Settling Time Reduction:

- The **state space** $\mathcal{S} = [K^+_{\text{zero,min}}, K^+_{\text{zero,max}}] \times \{1, ..., \text{card}(\Phi)\}$, where

  $$\Phi = \cup_i \boldsymbol{\phi}_i, \; i = 1, ..., \text{total number of periodic orbits} \tag{5.20}$$

  the set of periodic orbits . For this representation, $K^+_{\text{zero}}$ is already defined and "card" denotes the cardinality of the set of periodic orbits $\Phi$. The discrete set $\{1, ..., \text{card}(\Phi)\}$ describes the periodic orbit towards which the robot currently moved. The limits of $K^+_{\text{zero}}$ are determined by the set of orbits $\Phi$.

- The **action space** $\mathcal{T} = \{1, ..., \text{card}(\Phi)\}$ corresponds to the periodic orbit towards which we want the robot to take an intermediate transition.

Note that if the state $K^+_{\text{zero}}$ was replaced with the average velocity of the robot, the integration of the equations of motion would be necessary in order to calculate the time duration of the motion and with that the average velocity which requires a considerable amount of time. When utilizing the HZD framework though, an integration is not necessary.

In addition, Eq. (3.36)

$$K^{k+1}_{\text{zero}}(\xi_2^+) = \delta^{i\to f}_{\text{zero}}\left(K^k_{\text{zero}}(\xi_2^+) - V^{i\to f}_{\text{zero}}(\xi_1^-)\right)$$

will serve as the state transition function $\mathcal{F}$.

For each transition, a different parameter vector $\boldsymbol{v}$ is learnt. An outline of the Reinforcement Learning algorithm is presented in Algorithm 1. Once the learning is concluded, actions are selected in a greedy way according to

$$\tau \leftarrow \arg\max_{\bar{\tau}} \boldsymbol{\varphi}^\top(\boldsymbol{s}, \bar{\tau})\boldsymbol{v}$$

Details regarding the convergence proof of this algorithm can be found in [97] (Ch. 3.4).

**Algorithm 1** $Q$-Learning with linear parametrization

```
1: for all transitions $\boldsymbol{\phi}_i \to \boldsymbol{\phi}_j$ do
2:     for all epochs do
3:         Initialize learning rate $\eta_0$
4:         Initialize $\epsilon$
5:         Initialize state $\boldsymbol{s}_0 = \left[K^*_{\text{zero},i}, i\right]$
6:         Initialize randomly the parameter vector $\boldsymbol{\upsilon}$
7:         for all episodes do
8:             $\tau_k \leftarrow \begin{cases} \text{uniform random action in } \mathcal{T} \text{ with probability } \epsilon \\ \arg\max_{\bar{\tau}} \boldsymbol{\varphi}^\top(\boldsymbol{s}_k, \bar{\tau})\boldsymbol{\upsilon}_k \text{ with probability } 1-\epsilon \end{cases}$
9:             Apply $\tau_k$, measure $\boldsymbol{s}_{k+1}$ and reward $\rho_{k+1}$
10:            $\boldsymbol{\upsilon}_{k+1} \leftarrow \boldsymbol{\upsilon}_k + \eta_k \left[\rho_{k+1} + \chi \max_{\tau'}(\boldsymbol{\varphi}^\top(\boldsymbol{s}_{k+1}, \tau')\boldsymbol{\upsilon}_k) - \boldsymbol{\varphi}^\top(\boldsymbol{s}_k, \tau_k)\boldsymbol{\upsilon}_k\right] \boldsymbol{\varphi}(\boldsymbol{s}_k, \tau_k)$
11:            Reduce learning rate $\eta$ and $\epsilon$
12:        end for
13:    end for
14: end for
```

### Reward Function for Settling Time Reduction

Assuming that the fixed point $K^*_{\text{zero}}$ corresponds to the target periodic orbit $\boldsymbol{\phi}_{\text{target}}$, the **reward function** $\rho$ is chosen as

$$\rho(K^+_{\text{zero}}, i, j) = \exp(-\vartheta|K^+_{\text{zero}}(k+2) - K^*_{\text{zero}}|), \tag{5.21}$$

where

$$K^+_{\text{zero}}(k+1) = \delta^{i\to j}_{\text{zero}}\left(K^+_{\text{zero}} - V^{i\to j}_{\text{zero}}\right) \tag{5.22a}$$

$$K^+_{\text{zero}}(k+2) = \delta^{j\to\text{target}}_{\text{zero}}\left(K^+_{\text{zero}}(k+1) - V^{j\to\text{target}}_{\text{zero}}\right) \tag{5.22b}$$

The justification behind this reward function follows from the fact that the target periodic orbit $\boldsymbol{\phi}_{\text{target}}$ is pre-determined. The reward function accounts for the distance between the fixed point of the target orbit $K^*_{\text{zero}}$ and the value of $K^+_{\text{zero}}$, if at first a transition towards the intermediate orbit $\boldsymbol{\phi}_j$ and then to the target orbit $\boldsymbol{\phi}_{\text{target}}$ is taken. In a few words, we are interested in how the transition $\boldsymbol{\phi}_{i\to\text{target}}$ is influenced by an intermediate transition towards the orbit $\boldsymbol{\phi}_j$. If the transition is not feasible, the reward is equal to $-1$.

## 5.2.2 Experimental Evaluation

This section presents the evaluation of the learning scheme proposed in the previous subsection. The set of periodic orbits $\Phi$ is populated by 81 orbits corresponding to average desired velocities ranging from 0.7 to 1.5 m/s with a step of 0.01 m/s. Each orbit is determined by the optimization procedure as described in section 3.3 where the desired walking velocity was imposed as a task constraint. The one-step transitions between them are pre-computed according to the methodology described in section 3.2 and stored. The parameter $\vartheta$ in the reward function (5.21) is 0.2. Regarding the learning algorithm itself,

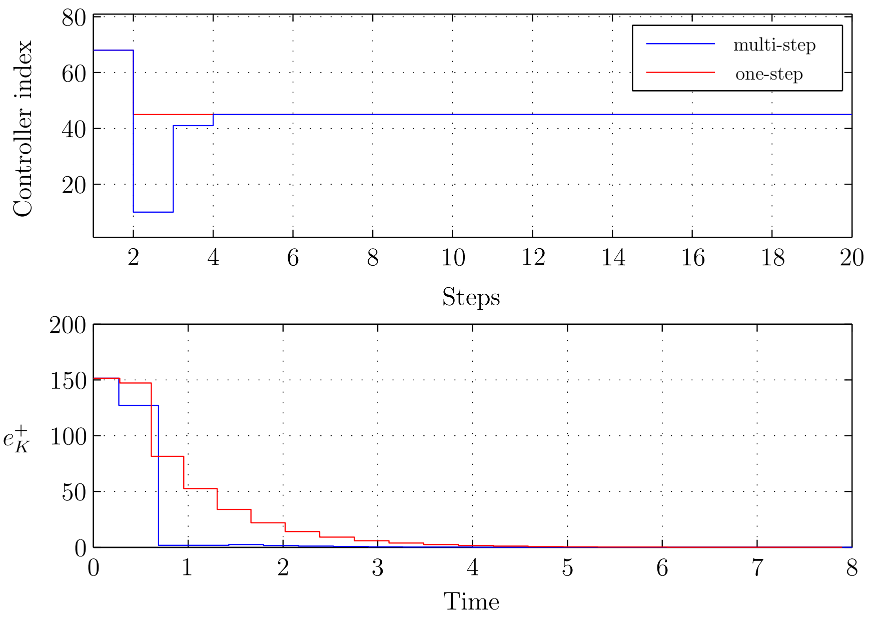


**Fig. 5.7:** The multi-step policy and the error convergence for a transition from a velocity of 1.37 $\mathrm{m/s}$ to that of 1.14 $\mathrm{m/s}$ ($\boldsymbol{\phi}_{68} \to \boldsymbol{\phi}_{45}$).

the discount factor $\chi$ is 0.7 and the probability of taking a random action $\epsilon$ is 30%. The learning procedure lasts for 30000 epochs, while each epoch lasts for 40 episodes. The aforementioned parameters were experimentally chosen.

#### Learning a single transition

This paragraph presents the experimental results obtained for learning a single transition from a velocity of 1.37 m/s to that of 1.14 m/s. As illustrated in Fig. 5.7, the multi-step policy tries to decelerate the robot by commanding it to walk with a much lower velocity than the target one. Then, it commands the robot to take a transition towards the target periodic orbit after two more steps. The validity of our approach can also be justified by plotting the convergence of the velocity error $e_v = |\bar{v} - v_{\text{target}}|$, where $\bar{v}$ is the average velocity of the robot during a step and $v_{\text{target}}$ is the target velocity. As shown in Fig. 5.8, there is a high correlation between the way the velocity and $K^+_{\text{zero}}$ converge to their desired values.

The superiority of the multi-step transition is clearly highlighted when the error $e_K^+ = |K^+_{\text{zero}} - K^*_{\text{zero}}|$ is taken into account. The multi-step policy takes a transition towards the periodic orbit $\boldsymbol{\phi}_{45}$ at time $t^\star \approx 1.1$ s (see Fig. 5.8) with an error $e_K^+ = 1.85$, while at time $t^o \approx 1.3$ s (see Fig. 5.8) the one-step transition has an error $e_K^+ = 34$. The error $e_v$ following the multi-step policy becomes negligible in approximately 2.2 s, while for the one-step it requires approximately 3.8 s.

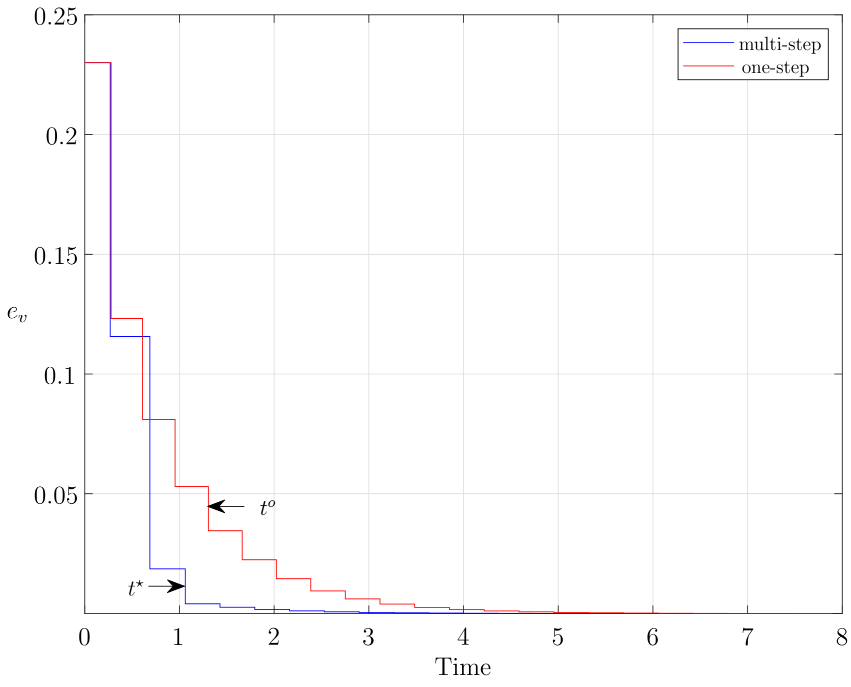


**Fig. 5.8:** The convergence of the velocity error for the transition $\phi_{68} \rightarrow \phi_{45}$

## Dealing with non-stationary policies

When it comes to Reinforcement Learning, one might end up dealing with policies that oscillate around the desired terminal state or in our case the orbit index that corresponds to the desired target velocity. In order to overcome this situation, once the policies for all the transitions are learnt, a post-processing procedure is initiated to detect such repeating patterns. An example of such a policy can be seen in Fig. 5.9 for a transition from a velocity of 0.84 m/s to that of 1.39 m/s. As shown there, the policy dictates to take a transition towards an orbit which corresponds to a larger velocity than the target one and then decelerates the robot towards the desired velocity, but does not eventually reach it, rather it oscillates between the orbits that correspond to the velocities of 1.38 m/s and 1.40 m/s. Since the target velocity is known, once such policies are detected, the pattern can be removed by fixing the action $\tau$ to the orbit corresponding to the target velocity as illustrated in Fig. 5.10.

## Overall performance

When evaluated on all possible transitions $\phi_{i\rightarrow j}$, $i \neq j$, the proposed methodology has a success rate of 84.34%, meaning that 84.34% of the overall transitions are performed faster with this framework. For the remaining 15.66% of the transitions, the one-step approach can be utilized, since it is known that it will perform better. There is always the possibility to fine tune the policies that perform worse than the one-step approach, but it is desired to have a uniform framework. Fig. 5.11 presents the overall performance of the proposed methodology. Finally, Fig. 5.12 gives a "heat" map, which shows how much better the

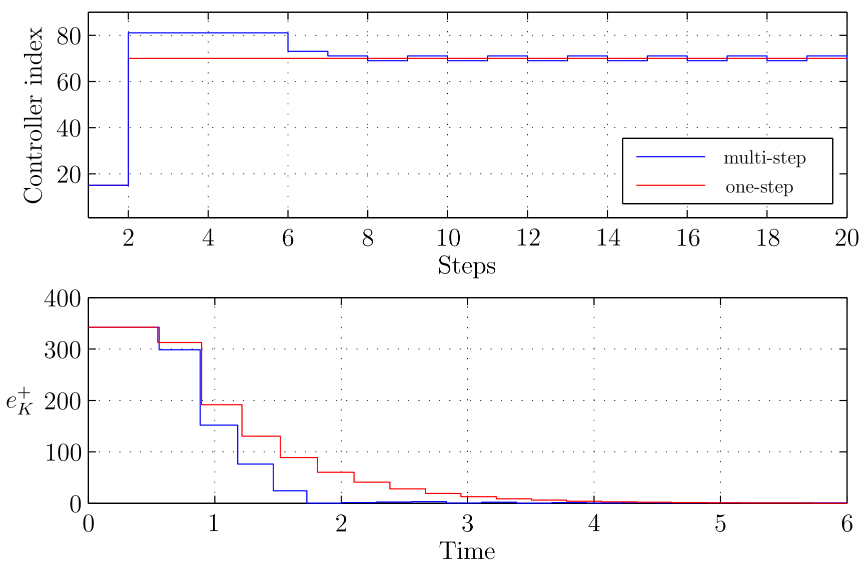


**Fig. 5.9:** An exemplary oscillating policy. The desired transition is from 0.84 $\mathrm{m/s}$ to 1.39 $\mathrm{m/s}$.

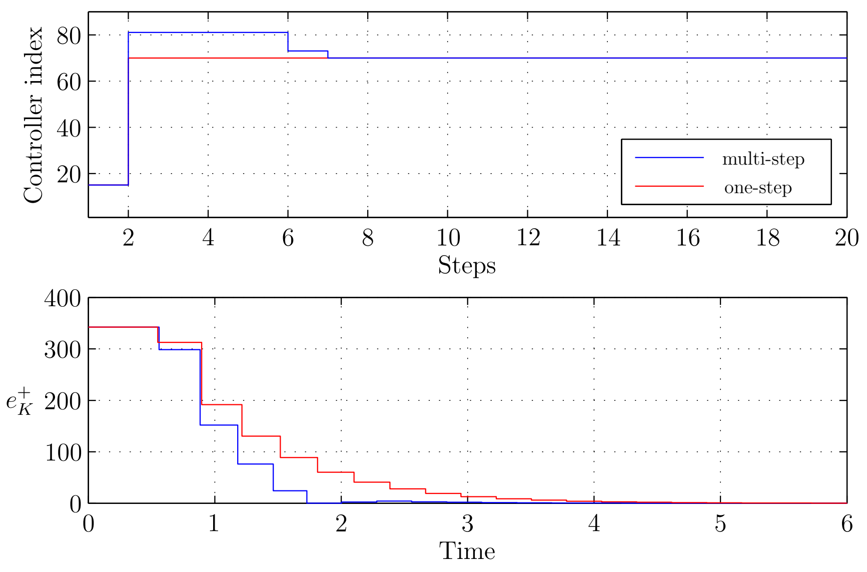


**Fig. 5.10:** The solution to the problem of the non-stationary policy. When the pattern 71 $\rightarrow$ 69 $\rightarrow$ 71 is discovered, the policy is fixed to the target orbit $\phi_{70}$.

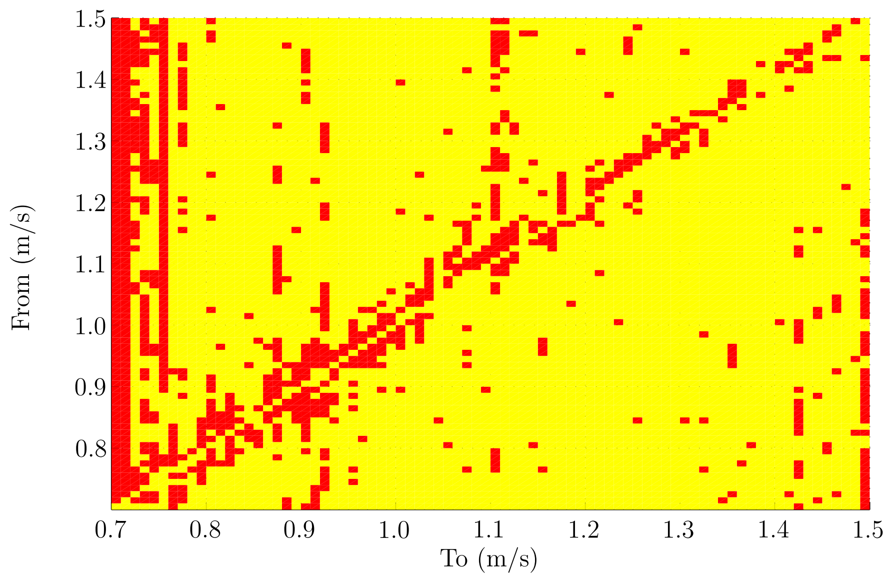


**Fig. 5.11:** The overall score for the proposed methodology. Yellow denotes that the multi-step policy performs better than the one-step approach, while red suggests the opposite. The periodic transitions (secondary diagonal) are not taken into account.

proposed methodology can perform in comparison to the one-step approach. The "heat" corresponds to the value $\Delta e_K^+ = |K_{\text{zero}}^+(t^\star) - K_{\text{zero}}^*| - |K_{\text{zero}}^+(t^o) - K_{\text{zero}}^*|$.

It is evident that the proposed methodology cannot outperform the one-step approach in cases where a transition is taken between "neighboring" orbits, i.e. transitions close to the secondary diagonal. For these cases, a sequential controller is not expected to offer much, since the fixed points of the initial and target periodic orbits are in general close to each other. On the other hand, learning the one-step transition for these cases depends strongly on the randomly selected actions at the beginning of each epoch.

The second big class of cases where the one-step approach has better performance, comprises transitions where the target orbit index is close to the limits (1 or 81). In that case, a multi-step transition cannot perform better than an one-step transition, since a possible deceleration or acceleration below and above the target velocity is not possible.

## 5.3 Summary

This chapter dealt with the problem of Settling Time Reduction for transitions between periodic orbits. We defined this problem with the utilization of Optimal Control and proposed a Reinforcement Learning approach for learning multi-step transitions which can reduce the settling time.

The cost function to be minimized in the Optimal Control Problem takes into account the convergence error to the fixed point of the target periodic orbit, the torque consumption and the total time duration of the motion. The feasibility condition is stated as an inequality constraint and further constraints regarding the transition and the style of motion are imposed. For the numerical evaluation we utilize a 5-link biped robot and present

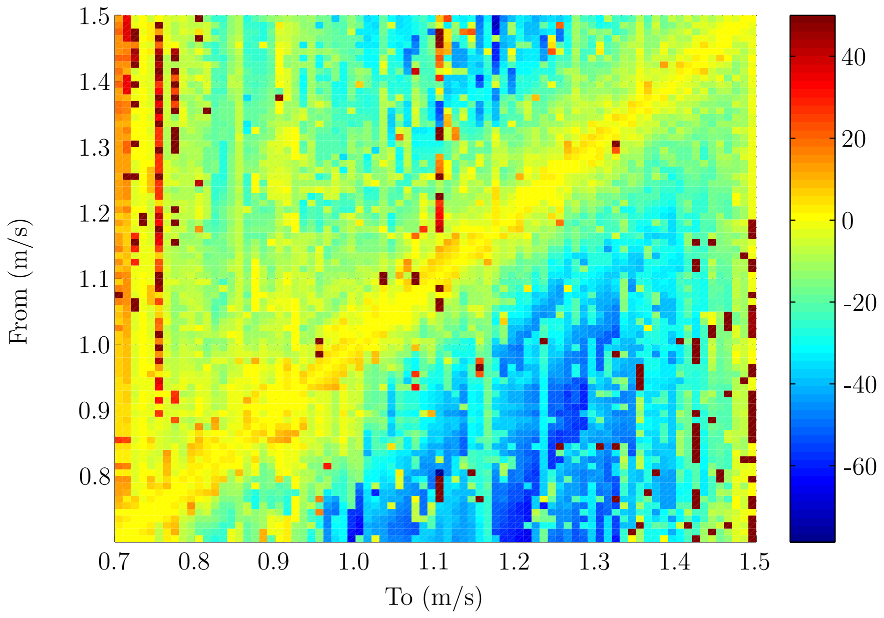


**Fig. 5.12:** A "heat" map showing how much better the proposed methodology performs in comparison to the one-step approach. The evaluation criterion is $\Delta e_K^+$. For the purpose of facilitating the illustration, differences above the value of 50 have been truncated.

the effect of different weighting of the terms in the cost function for a sample transition motion. As shown by the results, optimizing such a transition is advantageous in terms of entering the domain of attraction of the target periodic orbit closer to the fixed point in comparison to the one-step transitioning approach that is usually utilized.

In the Reinforcement Learning approach we expressed the problem of finding multi-step transitions as a Markov Decision Process. Using the HZD framework assisted in reducing the state representation for the Markov Decision Process. The experimental results demonstrate that the proposed framework can perform better for 84.34% in a total of 6480 assumed transitions for the 5-link biped which was used in this study.

# 6 Achieving New Tasks with Online Switching of Motion Primitives - Balancing

In the last years a lot of effort is targeted towards endowing humanoid robots with capabilities for outdoors operation and human assistance. In both cases, the humanoid is expected to be able to retain its balance, i.e. be able to reject or handle external disturbances such that a fall is avoided. An unforeseen disturbance force to a robot does not constitute a rare case, since an only partially observed environment can lead to collisions. In such a case, a control input has to be computed online in order to balance the robot. Also, in assistive tasks like human-robot collaborative object transportation, the robot should be able to counteract or comply with the forces applied by the human partner. In order to reduce the effort from the human side the appropriate control inputs have to be computed online.

Both cases demonstrate the need for fast computations online. One way to do so is through the utilization of simplified models, like the Linear Inverted Pendulum or variations of it, which assume that the mass of the system is concentrated on the Center of Mass and is constrained to move at constant height [6, 7]. In that case, the dynamics become linear and the derivation of a control input is simplified. Using such models, different balancing strategies can be achieved like the ankle, the hip, the squat or the stepping strategy [98, 99]. Also, since the dynamics are linear online optimization approaches can be utilized to derive balancing and walking motions and to choose between different balancing strategies [100, 101, 102].

Unfortunately, designing balancing motions using these simplified models does not consider energy consumption since an online optimization using the nonlinear equations of motion of a humanoid is computationally intractable. However, it has been shown that energy consumption is part of the optimization process utilized by humans when performing balancing motions [103]. Another limitation of using simplified models, is that even though the LIP with Reaction Wheel model [7] has been frequently used in studies of humanoid balancing, the direct application of the resulting motion to the humanoid itself is not straightforward.

Approaches towards the utilization of nonlinear systems of equations, suggest the linearization of the system equations around the state of the robot and the use of an LQR controller in the vicinity of the state. For this approach to be used, an estimation of the area of validity of the linearization of the system is necessary. Additionally, the domain of attraction of this LQR controller has to be computed using optimization methods [104]. Following this process offline for different state instances, the whole state space of the robot can be partitioned in disjoint areas based on the validity of the linearization and the domain of attraction of the corresponding LQR controller leading to the form of an LQR tree [78]. Humanoid robots however are subject to unilateral constraints on the contact forces, like the friction cone and the Center of Pressure constraints [11], which are difficult to be taken into account in the LQR tree approach. If these constraints are violated, the modeling assumptions are not valid anymore and slipping or underactuation might occur.

A possible solution is to use a database of Motion Primitives, that satisfy all the modeling assumptions. These Motion Primitives are the outcome of an offline optimization process which takes into account task objectives as well as energy criteria. Each Motion Primitive corresponds to a trajectory/controller that balances the robot with respect to a specific disturbance force. For the online utilization, it is only necessary to properly select the most suitable primitive. Related work regarding balancing with a database of Motion Primitives has been done by Liu *et al.* where local policies around trajectories are extracted using Differential Dynamic Programming [74]. In their work however, information of the push magnitude and application point is required for the selection of the local model.

An important aspect when utilizing Motion Primitives is robustness to a wide range of new disturbances, i.e. the ability to balance the robot with respect to disturbances that were not considered during the extraction of the database of the Motion Primitives. For that, online approaches can be utilized to correct any mismatch between the reference values and the measured ones. An example is the Dynamic Balance Force Control [42] where valid GRFs are found through the solution of a QP. As a next step, dynamically feasible accelerations and torques are generated with the use of a weighted pseudo-inverse approach. Another methodology using pseudo-inversion for animated characters is the Dynamics Filter [43] and the approach for full body passivity based control as proposed in the work of Cheng *et al.* [44] As an alternative, in the work of Lee *et al.* the balancing motion is generated by solving many QPs sequentially while at the same time desired accelerations are taken into account [105]. An additional methodology that employs QPs in order to provide motions which satisfy multiple tasks can be found in the work of Herzog *et al.*, where the solution of the proposed QP sequence can achieve rates of 1 kHz [106, 107]. Worth mentioning is also the work of Escande *et al.*, where the solution of a hierarchical QP can provide control rates of 1 kHz as well [108]. In addition, Kuindersma *et al.* solved a QP in order to apply a motion generated by a LIP model to the ATLAS humanoid in a simulated environment [21]. In the same fashion, in the work of Hopkins *et al.* a QP was used to render the motion generated using the Divergent Component of Motion dynamically feasible and consistent [109, 110].

In this thesis, the reference balancing motions are provided by Motion Primitives. For that, we apply numeric optimization using the nonlinear equations of motion of the robot, while at the same time we take both the CoP and the friction cone constraints into account. For this chapter, we work with the four link robot model (see section 2.4) which is a minimalistic approach of a humanoid robot employing its upper body for angular momentum regulation. Such a model can demonstrate all the balancing strategies, i.e. ankle, hip and squat strategy, but not stepping. In order to create Motion Primitives for different balancing situations, we apply pushes on the middle of the torso with different push strengths.

Afterwards, equidistant samples of each trajectory with their corresponding accelerations, torques and GRFs are stored, leading in a database of Motion Primitive Samples. During balancing we apply a Euclidean distance metric on the state to choose the best sample, forming in the end a Motion Primitive switching methodology. By using only the state of the robot, we gain also the advantage that we do not depend on information about the size and strength of the push. Otherwise we would need to implement an estimator like the Extended Kalman Filter, such that information regarding the strength and the size of the push could be provided [74]. We also do not need to apply intesive pre-processing of the data, as was done in the work of Liu *et al.* [74] The reason for that is that in [74], due to the lack of any online optimization, a global controller has to be designed such that any control signal applied to the robot can ensure that the modeling assumptions on the GRFs are satisfied. In our QP formulation, such an issue is not the case, since we can online provide balancing motions that satisfy these assumptions. Finally, designing a global controller scales badly with the size of the state of the system in comparison to a static QP.

The robustness to disturbance scenarios not considered during the design of the Motion Primitives is achieved by solving a single QP online where the adopted cost function penalizes deviations from the nominal accelerations, torques and GRFs that are provided by the chosen Motion Primitive. At the same time, torque limits and constraints on the GRFs (friction cone and CoP) are treated as inequality constraints and a Control Lyapunov Function constraint is used in order to provide exponential convergence - when posible - to the desired trajectories at a pre-specified rate [111]. In that way, the solution of the QP provides an optimal compromise between the tracking of optimized trajectories, the dynamic feasibility of the motion and the satisfaction of the modeling assumptions. We adopted a QP approach instead of a pseudo-inverse one due to the fact that we need to include a CLF constraint. In addition, even though a pseudo-inverse approach can yield solutions that respect the system dynamics, the resulting torques might violate the assumed actuator limits.

The main contribution of this chapter lies in the switching between pre-computed Motion Primitives in order to provide reference balancing solutions to the robot that bring it back to the rest posture while at the same time they provide robustness to unknown disturbance forces and energy efficient motions. The Motion Primitive switching is proposed, since the solution of the QP might steer the robot state closer to a sample that belongs to another Motion Primitive. An important characteristic of our approach that allows us to monitor the tracking performance is the use of a CLF constraint in the QP. By applying such a switching mechanism, we were able to have a more efficient motion than by committing to a single Motion Primitive from the beginning, something which in our opinion is a finding of major interest.

The remainder of the chapter is structured as follows: In section 6.1 balancing is formulated as a numeric optimization problem and we explain the procedure for creating the balancing motions and then the database of Motion Primitive Samples. In section 6.2 we demonstrate how a CLF constraint can be used for tracking. Section 6.3 presents the QP that we solve online in order to ensure the satisfaction of the constraints that will provide a dynamically feasible and convergent motion, as well as the selection methodology. The simulation results are illustrated and discussed in section 6.4. Section 6.5 concludes the chapter.

## 6.1 Motion Primitives for Humanoid Balancing

In this section we describe the static optimization problem that leads to the design of the Motion Primitives. Afterwards, we explain how each primitive is sampled and stored in the memory in order to form a database of Motion Primitive Samples.

### 6.1.1 Balancing as an Optimization Problem

For this thesis we choose to formulate the balancing problem as a trajectory optimization problem. As we already did for walking, we parametrize each joint trajectory as a Bézier polynomial. In order to facilitate the analysis we repeat the definition of the Bézier polynomial, but this time as a function of time $t$, i.e.

$$q_i(t) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \left(\frac{t}{T}\right)^k \left(\frac{T-t}{T}\right)^{M-k}, \tag{6.1}$$

where $M$ is the order and $\alpha_k^i$ the coefficients of the Bézier polynomial. The total duration of the balancing motion $T$ is fixed. We do not include in the trajectory optimization the position of the tip and the orientation of the foot, since they have to be constantly zero in order to satisfy Eq. (2.12b), i.e. the contact constraint. In addition, in Fig. 6.1 we depict once more the 4 link planar robot model used for the balancing studies in order to facilitate the presentation. The cost function to be minimized is adopted by the work from

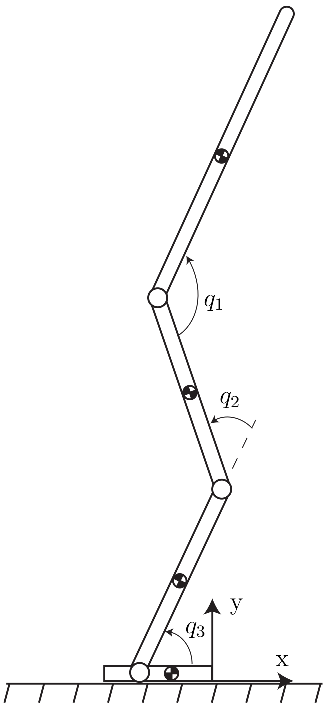


**Fig. 6.1:** Model of the planar robot used for balancing. The Newtonian reference frame is assumed to be fixed on the ground at the tip of the foot.

Atkeson *et al.*[98] and is given by

$$J_{\text{bal}}(\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, \boldsymbol{\alpha}^3) = \int_0^T \Big\{ (\boldsymbol{q} - \boldsymbol{q}_T)^\top \boldsymbol{W}_1 (\boldsymbol{q} - \boldsymbol{q}_T) + \dot{\boldsymbol{q}}^\top \boldsymbol{W}_2 \dot{\boldsymbol{q}} + \boldsymbol{u}^\top \boldsymbol{W}_3 \boldsymbol{u} \Big\} \, \mathrm{dt} \tag{6.2}$$

where $\boldsymbol{\alpha}^i \in \mathbb{R}^{(M+1)}$ are the Bézier coefficients of each joint trajectory and $\boldsymbol{W}_i$ are weighting matrices of appropriate dimensions. Note that $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\boldsymbol{u}$ depend on $\boldsymbol{\alpha}^i$ but in the definition this notation is suppressed for brevity. Finally, $\boldsymbol{q}_T$ corresponds to the desired rest posture of the robot. The cost function minimizes deviations from the rest posture, while at the same time it keeps the joint torques relatively low, such that energy efficiency is also taken into account. The weighting between these two objectives is determined by the matrices $\boldsymbol{W}_i$. It should be noted that this cost function can provide motions that use the ankle, hip and squat strategies.

During the optimization the following constraints have to be imposed such that the modeling assumptions are satisfied, foot tilting is avoided as well as actuator and joint limits are respected.

- Center of Pressure (CoP) constraint: The CoP [11] has been defined in section 2.3. To facilitate the presentation we cite here the definition again which is

  $$\text{CoP} = \frac{M_z}{F_y} \tag{6.3}$$

  In this equation $M_z$ is the ground reaction moment and $F_y$ the vertical component of the Ground Reaction Force. As long as the CoP is inside the base of support, foot tilting is avoided and the CoP is equivalent to the Zero Moment Point [10]. With correspondence to Fig. 6.1 the CoP constraint can be stated as

  $$-\ell_{\text{foot}} < \text{CoP} < 0 \tag{6.4}$$

  where $\ell_{\text{foot}}$ is the length of the foot, equal to 0.2 m.

- Friction Cone Constraint: In order to ensure that the foot is not sliding, the following relation must be satisfied:

  $$-\mu_s F_y \leq F_x \leq \mu_s F_y, \tag{6.5}$$

  where $\mu_s \in \mathbb{R}$ is the coefficient of static friction.

- Positive vertical contact force:

  $$F_y > 0 \tag{6.6}$$

- Joint position and joint velocity limits:

  $$\begin{aligned} \boldsymbol{q}_{\min} &\leq \boldsymbol{q} \leq \boldsymbol{q}_{\max} \\ \dot{\boldsymbol{q}}_{\min} &\leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_{\max} \end{aligned} \tag{6.7}$$

- Input torque saturation:

  $$\boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max} \tag{6.8}$$

This optimization process is directly applicable to fully actuated robots. The advantage of using parametrized trajectories is that the accelerations $\ddot{\boldsymbol{q}}$, control inputs $\boldsymbol{u}$ and GRFs $\boldsymbol{F}$ have analytical expressions. On the other hand, Bézier polynomials have the advantage that the first two coefficients $a_0^i$ and $a_1^i$ of each DoF $i$ are determined by the initial state of the robot and the last two coefficients $a_{M-1}^i$ and $a_M^i$ are determined by the terminal state of the robot. Another advantage of the Bézier parametrization is that the control inputs $\boldsymbol{u}$ are smooth in contrast to the controls inputs returned from an Optimal Control algorithm, which are in principle piecewise linear or piecewise constant. Of course, one can assume that the control inputs are states and the new control inputs are the torque derivatives. As a consequence though, the dimensionality of both the system and the optimization problem is increased and convergence is more difficult to be achieved. More insight on how Optimal Control could have been used to generate motions for humanoids can be found in the work of Denk *et al.* [76] and Koch *et al* [13].

Alternatives include Machine Learning approaches like Reinforcement Learning [27, 26, 112] or the use of Dynamical Movement Primitives (DMPs) which have been widely used for robotic applications [113, 114, 115]. In comparison to the Bézier parametrization, they are described by more coefficients which correspond to each basis function and its weight. In lack of demonstration data, these weights can be learned by Reinforcement Learning approaches, which in principle require more computational time [116]. However, the DMPs are in general characterized by better scaling properties both spatial and temporal.

### 6.1.2 Database of Motion Primitives Samples

In order to generate the database of Motion Primitives, we apply impact forces on the middle of the torso and use Eq. (2.19) to find the initial state of the robot. Solving Eq. (2.19) for $\dot{\boldsymbol{\theta}}^+$ and keeping in mind that $\dot{\boldsymbol{\theta}}^+ = \boldsymbol{0}$ will yield,

$$\dot{\boldsymbol{\theta}}^{+} = \boldsymbol{D}_e^{-1}(\boldsymbol{\theta}^{-})\boldsymbol{J}_{\text{push}}^{\top}(\boldsymbol{\theta}^{-})\boldsymbol{F}_{\text{push}}, \tag{6.9}$$

where $\boldsymbol{F}_{\text{push}}$ is the applied impulsive force and $\boldsymbol{J}_{\text{push}}$ is the Jacobian matrix at the middle of the torso. After the post-impact state is determined, we solve the aforementioned static optimization problem (subsection 6.1.1) using the `fmincon` function in MATLAB and a database of balancing motions (Motion Primitives) is generated.

As a next step, each Motion Primitive is sampled every $T_s$ seconds. For each Motion Primitive the important quantities that are stored in the database of Motion Primitive Samples are the joint positions, velocities and accelerations, as well as the control inputs and GRFs at each time step $kT_s$, $k = 0, ..., \frac{T}{T_s}$. The time step $kT_s$ that corresponds to the sample is also needed as well as the index $j$ of the current Motion Primitive. The Motion Primitive with index $j$ will be used to generate the reference values until we choose a new sample or until the end of the balancing motion in the case where the switching is shut off, as no other primitive is used. Summarizing, a Motion Primitive Sample is formally defined as the tuple

$$P_{[j,k]} = \Big(\boldsymbol{\theta}(kT_s),\ \dot{\boldsymbol{\theta}}(kT_s),\ \ddot{\boldsymbol{\theta}}(kT_s),\ \boldsymbol{u}(kT_s),\ \boldsymbol{F}(kT_s)\Big) \tag{6.10}$$

Therefore, the database $\mathcal{DB}$ of Motion Primitives Samples is defined as the collection $\mathcal{DB} = \cup_j \cup_k P_{[j,k]}$. By keeping the index $j$ constant and iterating through all the samples with $k$ values from $k = 0$ to $k = \frac{T}{T_s}$ we can reproduce a Motion Primitive.

Please note, that as an alternative we could store only the Bézier coefficients $\boldsymbol{\alpha}^i$ and generate the reference trajectories online. Then, by computing the system matrices we can compute the input torques and the GRFs. This is a trade-off between memory and CPU power. In this thesis we do not want to introduce any computational overhead and as a consequence we avoid storing only the Bézier coefficients $\boldsymbol{\alpha}^i$.

## 6.2 Control Lyapunov Function for Trajectory Tracking

In this section we briefly introduce the concept of the CLF which is utilized to provide feedback policies such that the desired trajectories will be tracked exponentially [117, 118]. The CLF is utilized because we do not want to explicitly determine any feedback gain matrices, but allow them to be decided online. The reason for that is that predetermining the gains might lead to undesired behavior. Low gains can lead to poor tracking, while large gains can lead to large accelerations that as a consequence can induce GRFs that violate the modeling assumptions or can produce large torques that exceed the actuator constraints.

### 6.2.1 Motion Primitive Tracking

For trajectory tracking we define outputs $\boldsymbol{y}$ of the form

$$\boldsymbol{y} = \boldsymbol{\theta} - \boldsymbol{\theta}_{\text{des}} \tag{6.11}$$

where $\boldsymbol{\theta}_{\text{des}}$ are the desired trajectories that are designed using the methodology that was described in subsection 6.1.1.

In order to zero the outputs we choose the Computed Torque Control (CTC) [64] scheme which is based on Input-Output Feedback Linearization. In CTC the control input is defined using the dynamics of the robot (2.12) as

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} = \boldsymbol{D}_e \left( \ddot{\boldsymbol{\theta}}_{\text{des}} + \frac{1}{\varepsilon} \boldsymbol{K}_d \left( \dot{\boldsymbol{\theta}}_{\text{des}} - \dot{\boldsymbol{\theta}} \right) + \frac{1}{\varepsilon^2} \boldsymbol{K}_p \left( \boldsymbol{\theta}_{\text{des}} - \boldsymbol{\theta} \right) \right) + \boldsymbol{C}_e \dot{\boldsymbol{\theta}} + \boldsymbol{G}_e \tag{6.12}$$

where $\boldsymbol{K}_d$ and $\boldsymbol{K}_p$ are positive definite constant diagonal matrices (gain matrices) and $\boldsymbol{S}_e$ is omitted since it is equal to the identity matrix $\boldsymbol{I}$. Then, we define

$$\begin{aligned} \ddot{\boldsymbol{\theta}} &= \ddot{\boldsymbol{\theta}}_{\text{des}} + \frac{1}{\varepsilon} \boldsymbol{K}_d \left( \dot{\boldsymbol{\theta}}_{\text{des}} - \dot{\boldsymbol{\theta}} \right) + \frac{1}{\varepsilon^2} \boldsymbol{K}_p \left( \boldsymbol{\theta}_{\text{des}} - \boldsymbol{\theta} \right) \\ \ddot{\boldsymbol{\theta}} &= \ddot{\boldsymbol{\theta}}_{\text{des}} + \boldsymbol{w} \end{aligned} \tag{6.13}$$

and then the closed loop dynamics become

$$\begin{aligned}\begin{bmatrix}\dot{\boldsymbol{y}}\\ \ddot{\boldsymbol{y}}\end{bmatrix} &= \begin{bmatrix}\mathbf{0} & \boldsymbol{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{y}\\ \dot{\boldsymbol{y}}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{w} \Rightarrow\\ \Rightarrow \dot{\boldsymbol{\eta}} &= \boldsymbol{A\eta} + \boldsymbol{Bw} \end{aligned} \tag{6.14}$$

Considering the continuous time algebraic Riccati equation

$$\boldsymbol{A}^\top\boldsymbol{P} + \boldsymbol{PA} - \boldsymbol{PBB}^\top\boldsymbol{P} + \boldsymbol{Q} = \mathbf{0} \tag{6.15}$$

where the solution is $\boldsymbol{P}$, we can use the solution $\boldsymbol{P}$ to construct a Rapidly Exponentially Stabilizing (RES)-CLF that can stabilize (6.14) at a rate $0 < \varepsilon < 1$. Note that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are positive definite symmetric matrices.

The Lyapunov function $V_{\text{Lyap}}$ has a quadratic form

$$V_{\text{Lyap}}(\boldsymbol{\eta}) = \boldsymbol{\eta}^\top \begin{bmatrix}\frac{1}{\varepsilon}\boldsymbol{I} & 0\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{P}\begin{bmatrix}\frac{1}{\varepsilon}\boldsymbol{I} & 0\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{\eta} = \boldsymbol{\eta}^\top\boldsymbol{P}_\varepsilon\boldsymbol{\eta} \tag{6.16}$$

and as defined in the work of Ames *et al.* [111] it is a RES-CLF if the following two conditions hold:

- The CLF is bounded as:

$$c_1\|\boldsymbol{\eta}\|^2 \leq V_{\text{Lyap}}(\boldsymbol{\eta}) \leq \frac{c_2}{\varepsilon^2}\|\boldsymbol{\eta}\|^2, \tag{6.17}$$

  where $c_1 > 0$ and $c_2 > 0$.

- The CLF is exponentially decreasing:

$$\begin{aligned}\dot{V}_{\text{Lyap}}(\boldsymbol{\eta}) &\leq -\frac{c_3}{\varepsilon}V_{\text{Lyap}}(\boldsymbol{\eta})\\ \boldsymbol{\eta}^\top\left(\boldsymbol{A}^\top\boldsymbol{P}_\varepsilon + \boldsymbol{P}_\varepsilon\boldsymbol{A}\right)\boldsymbol{\eta} + 2\boldsymbol{\eta}^\top\boldsymbol{P}_\varepsilon\boldsymbol{Bw} &\leq -\frac{c_3}{\varepsilon}V_{\text{Lyap}}(\boldsymbol{\eta})\\ \boldsymbol{\eta}^\top\left(\boldsymbol{A}^\top\boldsymbol{P}_\varepsilon + \boldsymbol{P}_\varepsilon\boldsymbol{A}\right)\boldsymbol{\eta} + \frac{c_3}{\varepsilon}V_{\text{Lyap}}(\boldsymbol{\eta}) + 2\boldsymbol{\eta}^\top\boldsymbol{P}_\varepsilon\boldsymbol{Bw} &\leq 0\\ \psi_0 + \boldsymbol{\psi}_1^\top\boldsymbol{w} &\leq 0\end{aligned} \tag{6.18}$$

  where $c_3$ is defined as

$$c_3 = \frac{\lambda_{\min}(\boldsymbol{Q})}{\lambda_{\max}(\boldsymbol{P})} > 0 \tag{6.19}$$

  and $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ denote the minimum and maximum eigenvalues, respectively, of a given symmetric matrix.

For the derivation of (6.18) we made use of (6.14) and the fact that $\boldsymbol{P}_\varepsilon$ is symmetric.

### 6.2.2 Feedback determination through QP with CLF Constraint

The auxiliary input $\boldsymbol{w}$ can be determined online through the solution of a QP. A simple version of it finds the minimum norm $\boldsymbol{w}$ which satisfies (6.18) with a tolerance $\delta$ and can be formulated as

$$\begin{aligned} &\operatorname{argmin}_{(\delta,\boldsymbol{w})} \left\{p_{\mathrm{CLF}}\delta^2 + \boldsymbol{w}^\top \boldsymbol{w}\right\} \\ &\text{s.t. } \psi_0 + \boldsymbol{\psi}_1^\top \boldsymbol{w} \leq \delta \end{aligned}$$

where $p_{\mathrm{CLF}}$ is a positive scalar that penalizes the violation $\delta$ of (6.18). The CLF constraint is relaxed because otherwise it can lead to a feedback $\boldsymbol{w}$ that in order to be realized needs torques that exceed the actuator limits or GRFs that violate the modeling assumptions. As shown in the work of Ames *et al.*, this formulation can satisfy bounds on the torques and GRFs [22] which we will introduce in the sequel.

## 6.3 Robustness to Unknown Disturbances

The purpose of using a database of Motion Primitives Samples is to be able to provide an energy efficient and dynamically feasible reference balancing motion that satisfies all the modeling assumptions for a wide range of impact situations. At the same time it assists in steering the motion of the robot towards the rest posture, after it being disturbed with an impact force that was not considered during the generation of the Motion Primitives.

As a consequence, our approach will be able to balance the robot with respect to unknown situations, demonstrating robustness characteristics. For this thesis, disturbances correspond to impact forces with a magnitude and/or an application point which was not considered during the generation of the Motion Primitives, as described in subsection 6.1.2. In principle, robustness is achieved by choosing a suitable Motion Primitive Sample, then solving a QP at each time step to ensure that the constraints on the GRFs $\boldsymbol{F}$ and control inputs $\boldsymbol{u}$ which were introduced at the static optimization problem in subsection 6.1.1 are satisfied and that the resulting motion is dynamically feasible and stabilizing according to the CLF (6.16).

### 6.3.1 Selection of New Samples - Motion Primitive Switching

The selection methodology takes place every $T_n$ seconds, where $T_n$ is greater or equal to the sampling time $T_s$. This relation is important because if $T_s$ is small and $T_s \approx T_n$, the frequent switching of the reference trajectories might give rise to a chattering behavior. After these two time constants are chosen, we apply every $T_n$ seconds a weighted Euclidean distance metric on the current state of the robot $\boldsymbol{x}(kT_n) = \left[\boldsymbol{q}^\top(kT_n)\ \dot{\boldsymbol{q}}^\top(kT_n)\right]^\top$ and the samples in the database $\mathcal{DB}$, i.e.

$$P^* = \operatorname{argmin}_{i,k} \left(P_{[i,k]} - \boldsymbol{x}(kT_n)\right)^\top \boldsymbol{L} \left(P_{[i,k]} - \boldsymbol{x}(kT_n)\right) \tag{6.20}$$

Here, the matrix $\boldsymbol{L}$ is a weighting matrix.

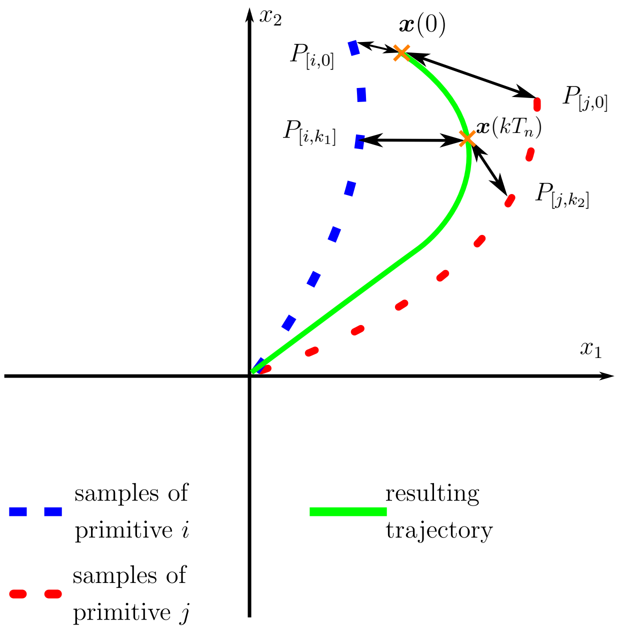


**Fig. 6.2:** Graphic representation of the switching approach. The initial state $\boldsymbol{x}(0)$ is closer to the state of the sample $P_{[i,0]}$, but after $kT_n$ seconds we find that the sample of another Motion Primitive is closer to $\boldsymbol{x}(kT_n)$ and for that reason we choose to track the Motion Primitive with index $j$. Afterwards, either the trajectory of the $j-$th Motion Primitive is always the closest in terms of Euclidean distance or the CLF constraint violation at time $T_n$ was below the user defined threshold $\delta_{\text{thr}}$ and we shut off the switching. The dummy states $x_1$ and $x_2$ are introduced for presentation purposes and (0,0) is the equilibrium state.

The reason for the switching is due to the fact that the tracking might have poor performance at the expense of satisfying the modeling assumptions or the torque limits. Therefore, it might be preferable to find a better Motion Primitive to track.

If switching is preferable, for the next $T_n$ seconds the index $j$ of $P^*$ will be used to generate the desired joint positions, velocities, accelerations, torques and GRFs for the QP. The QP is solved at each execution step, i.e. every $T_s$ seconds. The switching is turned off at a time $t^*$ when the CLF violation $\delta$ is smaller than a user defined threshold $\delta_{\text{thr}}$ and for the remaining $T - t^*$ seconds the desired values are generated from the last selected Motion Primitive. In Fig. 6.2 we provide a graphic representation of the switching methodology.

## 6.3.2 Quadratic Program for Motion Primitive Tracking and Constraint Satisfaction

The tracking of a Motion Primitive is expressed as a QP, allowing to be solved online. For the QP, we have to define a quadratic cost function and express all the necessary equality and inequality constraints in a linear way. In this thesis we choose to give more freedom to the optimization problem by allowing it to manipulate not only the violation $\delta$ of the CLF constraint (6.18), the torques $\boldsymbol{u}$ and the GRFs $\boldsymbol{F}$, but also the accelerations $\ddot{\boldsymbol{q}}$, as opposed to the work of Ames *et al.* [22] The cost function $\Psi$ to be minimized is defined as

$$\Psi = \left\{ \left( \begin{bmatrix} \ddot{\boldsymbol{\theta}}^* \\ 0 \\ \boldsymbol{u}^* \\ \boldsymbol{F}^* \end{bmatrix} - \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} \right)^\top \boldsymbol{W}_{\mathrm{QP}} \left( \begin{bmatrix} \ddot{\boldsymbol{\theta}}^* \\ 0 \\ \boldsymbol{u}^* \\ \boldsymbol{F}^* \end{bmatrix} - \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} \right) \right\} \tag{6.21}$$

and is penalizing deviations from the reference values provided by the selected primitive which is given by the index $j$ of $P^*$. In the function above, $\boldsymbol{W}_{\mathrm{QP}}$ is a positive definite weighting matrix, where the diagonal element $\boldsymbol{W}_{\mathrm{QP}_{7,7}}$ equals $p_{\mathrm{CLF}}$ and controls the penalty $\delta$ on the violation of the CLF constraint (6.18). Regarding the linear constraints introduced in the QP, they are listed below with a brief description of their purpose.

### Dynamically Feasible Motion

The motion that is required to be performed by the robot should be dynamically feasible. In order to ensure this, we express the dynamics of the robot (2.12) as

$$\underbrace{\begin{bmatrix} \boldsymbol{D}_e & 0 & -\boldsymbol{S}_e & -\boldsymbol{J}_1^\top \\ \boldsymbol{J}_1 & 0 & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{A}_{eq}} \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} = \underbrace{\begin{bmatrix} -\boldsymbol{C} - \boldsymbol{G} \\ -\dot{\boldsymbol{J}}_1 \dot{\boldsymbol{\theta}} \end{bmatrix}}_{\boldsymbol{b}_{eq}}$$

which will be imposed in the QP as an equality constraint.

### Control Lyapunov Function Constraint

The CLF constraint has been already discussed in section 6.2 and here is re-written in a form that is suitable for the proposed QP. To that end, Eq. (6.18) can be brought in the form of

$$\underbrace{\begin{bmatrix} \boldsymbol{\psi}_1^\top & -1 & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}}_{\boldsymbol{A}_{in}^1} \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} \leq \underbrace{-\psi_0 + \boldsymbol{\psi}_1^\top \ddot{\boldsymbol{\theta}}_{\mathrm{des}}}_{\boldsymbol{b}_{in}^1},$$

since $\boldsymbol{w} = \ddot{\boldsymbol{\theta}} - \ddot{\boldsymbol{\theta}}_{\mathrm{des}}$.

### Constraints on the Contact Forces

The constraints imposed on the contact forces can be expressed as inequality constraints which are linear to the forces. This can be expressed as follows

$$\underbrace{\begin{bmatrix} \mathbf{0} & 0 & \mathbf{0} & 1 & -\mu_s & 0 \\ \mathbf{0} & 0 & \mathbf{0} & -1 & -\mu_s & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & -1 & 0 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & 0 & 1 \\ \mathbf{0} & 0 & \mathbf{0} & 0 & -\ell_{\text{foot}} + \varrho & -1 \end{bmatrix}}_{\boldsymbol{A}_{in}^2} \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} \leq \underbrace{\begin{bmatrix} 0 \\ 0 \\ -\varrho \\ -\varrho \\ 0 \end{bmatrix}}_{\boldsymbol{b}_{in}^2}$$

where $\varrho$ is a small positive number. The first two lines correspond to the friction cone constraint. The third one ensures that the ground is pushing instead of pulling. The last two lines correspond to the CoP constraint and the variable $\ell_{\text{foot}}$ is the length of the foot. The last three columns of $\boldsymbol{A}_{in}^2$ are expanded to shift the focus towards the GRFs $\boldsymbol{F} = [F_x \; F_y \; M_z]^\top$.

### Torque Limits

Finally, we impose a linear inequality constraint for the torque limits such that unacceptable joint torques are avoided.

$$\underbrace{\begin{bmatrix} \mathbf{0} & 0 & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & 0 & -\boldsymbol{I} & \mathbf{0} \end{bmatrix}}_{\boldsymbol{A}_{in}^3} \begin{bmatrix} \ddot{\boldsymbol{\theta}} \\ \delta \\ \boldsymbol{u} \\ \boldsymbol{F} \end{bmatrix} \leq \underbrace{\begin{bmatrix} \boldsymbol{u}_{\max} \\ -\boldsymbol{u}_{\min} \end{bmatrix}}_{\boldsymbol{b}_{in}^3}$$

## 6.3.3 Overall Quadratic Program

For brevity, the optimization variables can be concatenated in $\boldsymbol{\kappa} = \left[\ddot{\boldsymbol{q}}^\top \; \delta \; \boldsymbol{u}^\top \; \boldsymbol{F}^\top\right]^\top$ and then the overall QP is formulated as

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{\kappa}} \; \Psi \\ &\quad \text{s.t.} \quad \boldsymbol{A}_{eq} \; \boldsymbol{\kappa} = \; \boldsymbol{b}_{eq} \\ &\qquad\quad \begin{bmatrix} \boldsymbol{A}_{in}^1 \\ \boldsymbol{A}_{in}^2 \\ \boldsymbol{A}_{in}^3 \end{bmatrix} \boldsymbol{\kappa} \leq \begin{bmatrix} \boldsymbol{b}_{in}^1 \\ \boldsymbol{b}_{in}^2 \\ \boldsymbol{b}_{in}^3 \end{bmatrix} \end{aligned}$$

This QP finds a compromise between staying close to the reference acceleration, torques and GRFs provided by the primitive which corresponds to the sample $P^*$ and keeping the CLF violation $\delta$ close to zero (or optimally zero) such that we achieve good tracking performance. At the same time, we avoid unacceptable torques $\boldsymbol{u}$ and GRFs $\boldsymbol{F}$ that violate the modeling assumptions or accelerations $\ddot{\boldsymbol{\theta}}$ which in combination with the torques $\boldsymbol{u}$ and

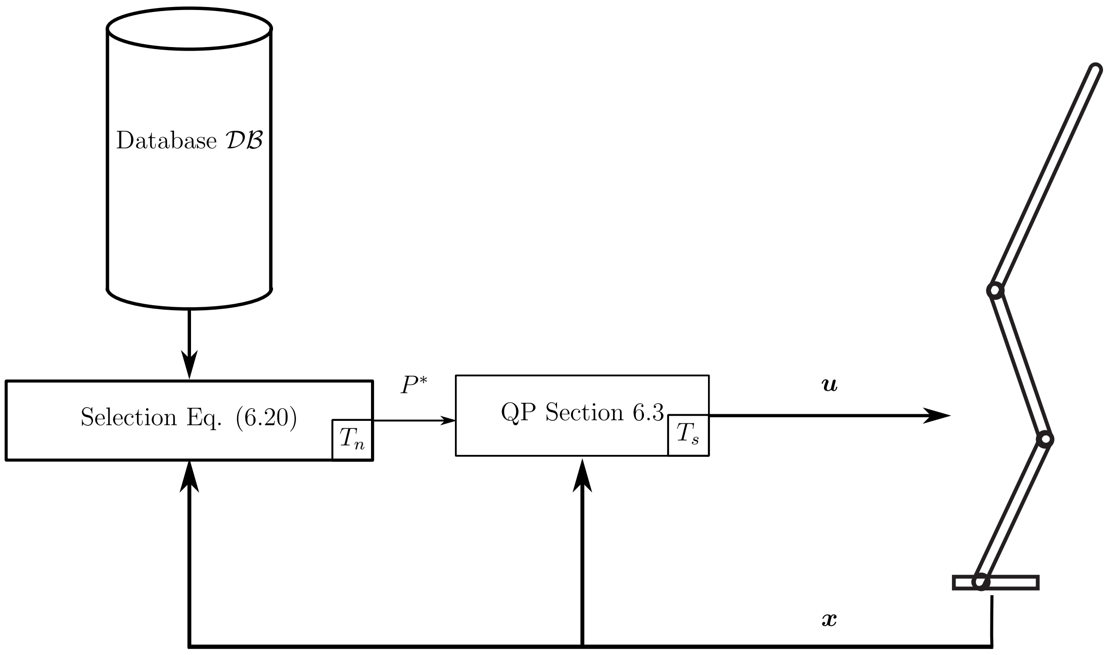


**Fig. 6.3:** Schematic diagram of the overall proposed architecture. The working frequency of the QP block is $T_s$ seconds while a new sample is selected every $T_n$ seconds. The selection block is shut off if the CLF violation $\delta$ is smaller than a threshold $\delta_{\text{thr}}$. If this is the case the index of the last selected sample dictates the Motion Primitive that provides reference values to the QP for the remaining of the balancing motion.

the GRFs $\boldsymbol{F}$ will give rise to dynamically infeasible motions. In Fig. 6.3 we depict the complete architecture of the proposed methodology.

At this point we would like to mention that the QP approach can provide balancing solutions for disturbance scenarios where the offline optimization presented in subsection 6.1.1 fails. This is due to the fact that the offline optimization utilizes Bézier polynomials which are smooth and as a consequence provide smooth control inputs $\boldsymbol{u}$ and GRFs $\boldsymbol{F}$. In contrast, the QP provides control inputs and GRFs that do not necessarily have to be fitted by any spline since they are computed at every time step $T_s$. As a consequence the QP allows for more arbitrarily shaped control inputs and GRFs to be generated that in turn can provide balancing motions that are capable of dealing with a wider variety of disturbance scenarios.

One might have been successful in generating balancing solutions for the case where the offline Bézier based approach fails by using Optimal Control with piecewise constant or linear parametrization for the control inputs. Finally, an enhancement of the Bézier polynomials is possible by increasing their order $M$ or allowing for non-equally spaced control points, but an increase in the complexity of the offline optimization problem is expected.

**Tab. 6.1:** Parameters and variables used for the extraction of the Motion Primitives

| | | |
|---|---|---|
| | $\boldsymbol{W}_1$ | diag([1 1 1]) |
| Cost function $J_{\text{bal}}$ | $\boldsymbol{W}_2$ | diag([1 1 1]) |
| | $\boldsymbol{W}_3$ | diag([0.002 0.002 0.002]) |
| Total Time (s) | $T$ | 3 |
| Order of Bézier polynomial | $M$ | 5 |
| Static Friction Coefficient | $\mu_s$ | 0.7 |

## 6.4 Simulation Evaluation

This section presents the evaluation of the aforementioned methodology on the simulated four-link robot. We first list the values of the parameters and the variables used in the simulation experiments in order to facilitate the reproducibility of the results reported here. Later, we show different evaluation scenarios.

### 6.4.1 Database Extraction - Values and Parameters

The balancing motions are generated by solving the static optimization problem in section 6.1 for impact forces of the form $\boldsymbol{F}_{\text{push}} = [F^x_{\text{push}}\ 0]^\top$ with $F^x_{\text{push}} = 1, ..., 80$ N and an interval of 1 N. The application point of the impact force is at the middle of the torso. Therefore, we have a total of 80 optimization problems to solve offline. After that, each primitive is sampled every $T_s$ seconds leading to a database $\mathcal{DB}$ of $\left(\frac{T}{T_s}+1\right)\times 80 = 3001\times 80 = 240080$ Motion Primitive Samples. Note that in the sequel the application point of an impact force will be described by a number between 0 and 1, where 0 means the hip and 1 the end-point of the torso. Based on this notation, the application point for the generation of the Motion Primitives was 0.5.

The parameters of the static optimization problem are presented in Table 6.1 and the ones of the QP and the selection methodology in Table 6.2. For the solution of the QP in the online phase we used the open source software qpOASES [119].

In Table 6.2 the zero elements of $\boldsymbol{L}$ correspond to the tip position and foot orientation as well as the tip velocity and foot angular velocity, which are always zero. The nonzero elements correspond to the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$. The rest of the values of the weighting matrix $\boldsymbol{L}$ are experimentally tuned. In principle, however, for $\boldsymbol{L}$ and $\boldsymbol{Q}$ we use higher penalties for elements that correspond to DoFs that are closer to the foot. The reason for that is that they have to manipulate the masses and inertia of all the links above them in the kinematic chain, therefore they influence the balancing motion to a greater extend than the DoFs away from the foot.

### 6.4.2 Trajectory switching

In this experiment we apply an impact force $\boldsymbol{F}_{\text{push}} = [40\ 25]^\top$ with an application point of 0.7. The nominal cost $J_{\text{bal}}$ (see Eq. (6.2)) of the overall produced motion is 7.1755. For

**Tab. 6.2:** Parameters and variables used in the Motion Primitive Sample selection and in the Quadratic Program

| | | |
|---|---|---|
| Selection method | $\boldsymbol{L}$ | diag([1 5 10 0 0 0 0.1 0.5 1 0 0 0]) |
| | $T_s$ (ms) | 1 |
| | $T_n$ (ms) | 10 |
| | $\delta_{\text{thr}}$ | 0.001 |
| CLF Constraint | $\varepsilon$ | 0.04545 |
| | $p_{\text{CLF}}$ | 100 |
| | $\boldsymbol{Q}$ | diag([1 5 10 10 10 10 10 50 100 100 100 100]) |
| Cost function | $\boldsymbol{W}_{\text{QP}}$ | diag([1 1 1 1 1 1 $p_{\text{CLF}}$ 0.002 0.002 0.002 0.002 0.002 0.002]) |
| Torque Limits | $\boldsymbol{u}_{\max}$ (N· m) | $[60\ 60\ 60]^\top$ |
| | $\boldsymbol{u}_{\min}$ (N· m) | $[-60\ \ -60\ \ -60]^\top$ |

comparison purposes, we also used the offline optimization process described in section 6.1 for the same impact force. Remarkably, the offline optimization for this disturbance scenario gave a solution with an overall cost of 7.1740, initialized from different Bézier coefficients randomly. This is a very interesting finding, since the QP solution was able to come up with a balancing motion that is very close to the optimal one, only by switching between optimized balancing motions. As shown in Fig. 6.4, the robot returns finally to the rest posture. In addition, the torques (Fig. 6.5) stay always in the limits and after a transition phase at the beginning, they remain smooth. The constraints on the GRFs (Fig. 6.6) are satisfied since the CoP stays between $-\ell_{\text{foot}} = -0.2$ and 0 and the friction stays between the limits, since $-\mu_s \leq \frac{Fx}{Fy} \leq \mu_s$, where $\mu_s$ equals 0.7.

We compared our methodology with a classic non-switching approach where we apply Eq. (6.20) only once at the post-impact state of the robot and the reference values of the initially chosen primitive are tracked for the whole motion. The cost $J_{\text{bal}}$ using this approach was 52.6697, much higher than what the trajectory switching offers.

The sequence of best fitting Motion Primitives is presented in Fig. 6.7. As is shown, at 0.13 s the switching is shut off since the CLF constraint violation is below the assumed threshold of $\delta_{\text{thr}} = 0.001$. Before that, it is shown that different Motion Primitives were chosen and tracked for either shorter or longer time.

Finally, in Fig. 6.8 we provide the final posture of the robot after an impact force of $\boldsymbol{F}_{\text{push}} = [60\ 25]^\top$ with an application point of 0.7. As is shown, the trajectory switching methodology allows for balancing while the approach of following only one primitive chosen at the beginning is not able to bring the robot to the rest posture in the allotted time. Regarding the cost $J_{\text{bal}}$ of the motion, it is 14.8088 for the switching methodology and 122.41 for the non-switching approach mainly due to the fact that the deviation between the desired and the actual terminal state of the robot is very large.

As is shown, our system is able to reject disturbances that were not presented during the

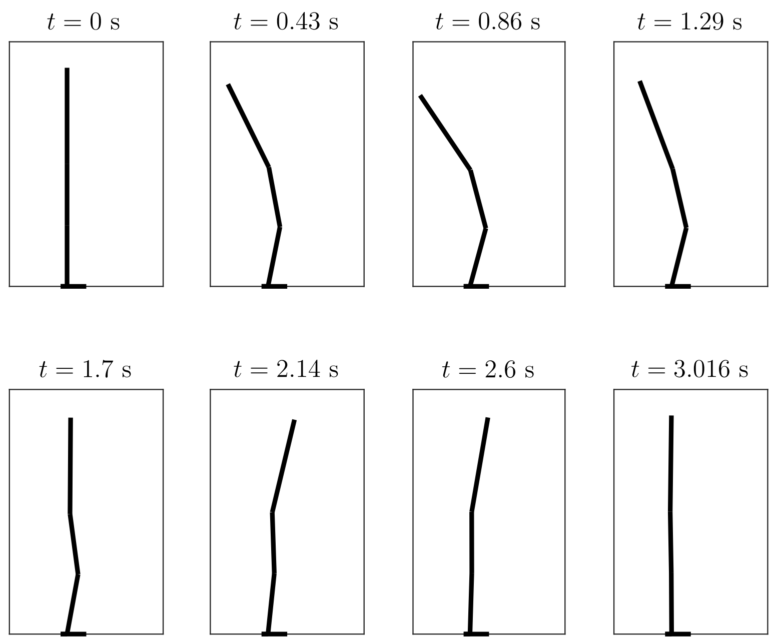


**Fig. 6.4:** Snapshots of the resulting balancing motion.

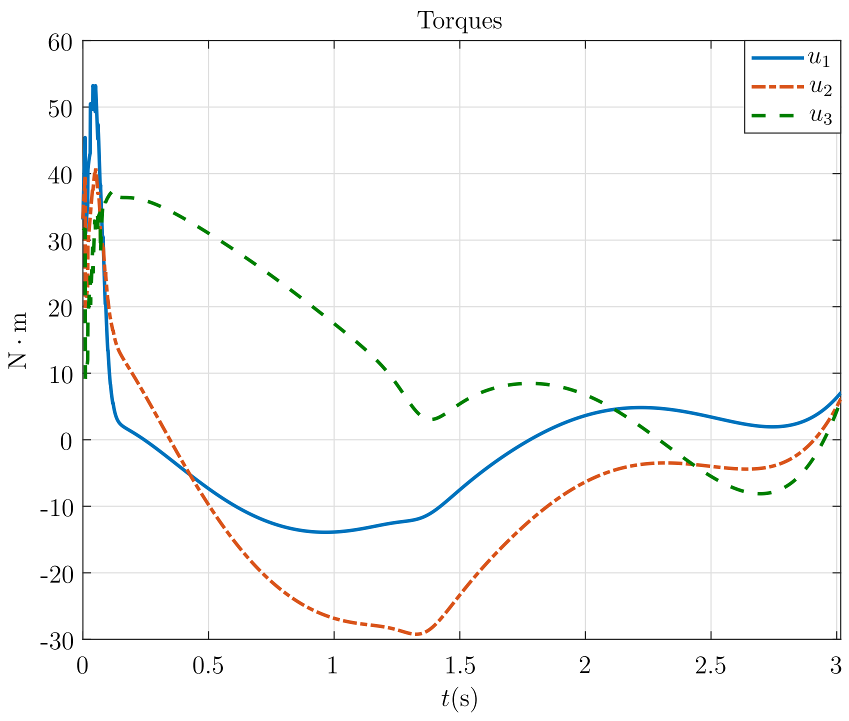


**Fig. 6.5:** Torques of the resulting balancing motion.

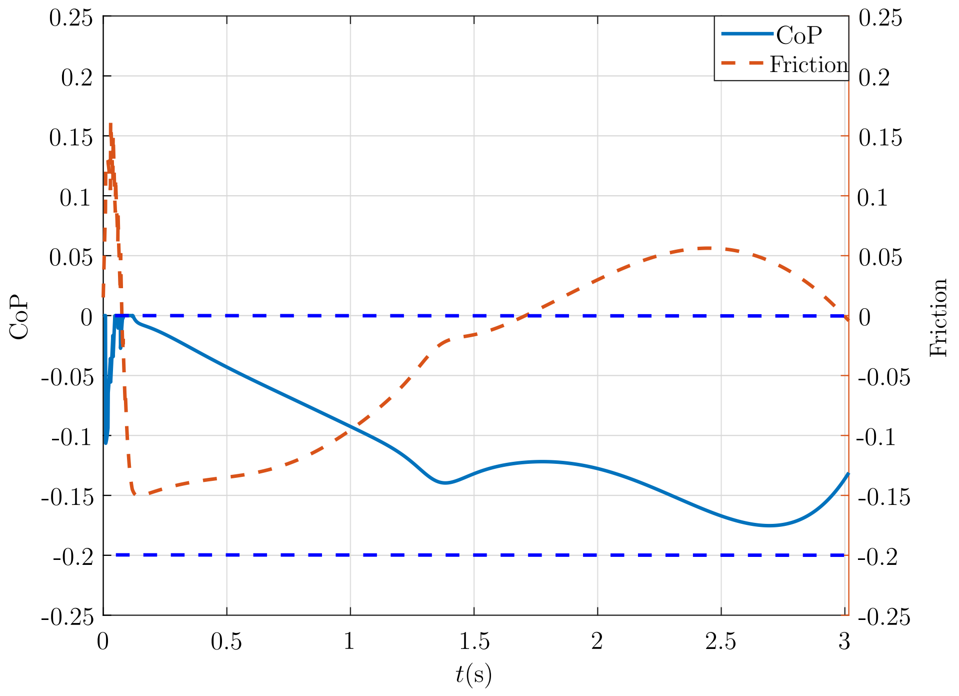


**Fig. 6.6:** CoP and friction associated with the resulting balancing motion. The dashed horizontal lines mark the limits for the CoP.

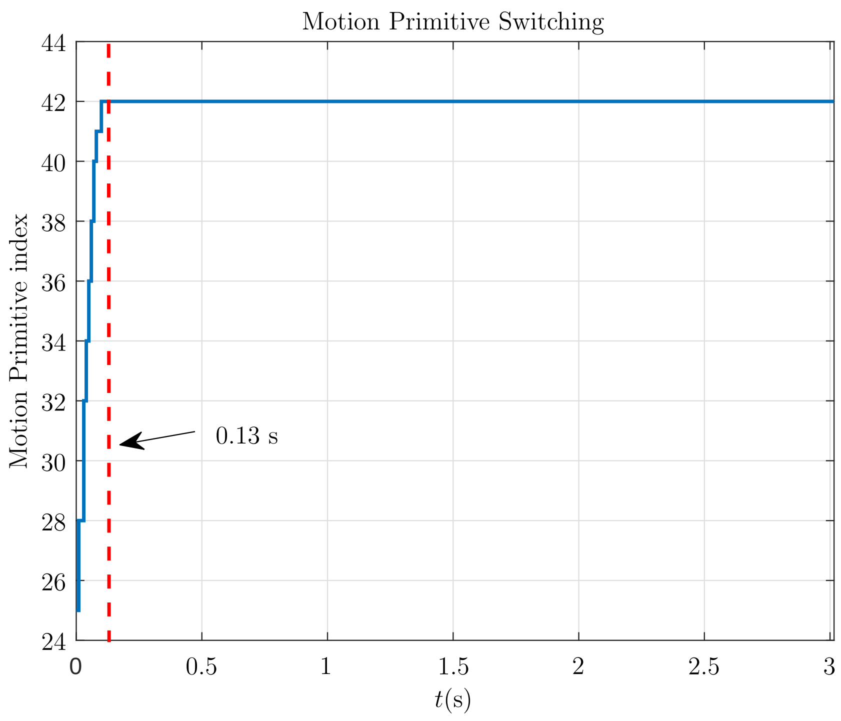


**Fig. 6.7:** Motion Primitives that choose the reference values to be forwarded to the QP. The red dashed line indicates when the CLF constraint violation becomes smaller than the threshold $\delta_{\text{thr}}$ and the switching is shut off.

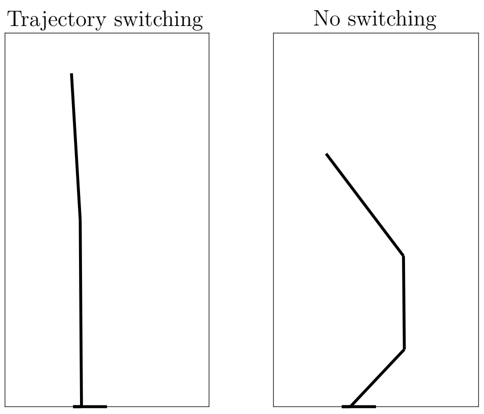


**Fig. 6.8:** Final posture of the robot after an impact force of $\boldsymbol{F}_{\text{push}} = [60\ 25]^{\top}$ with an application point of 0.7 when following the switching approach (left) and when following the non-switching approach (right).

offline optimization phase by employing the trajectory switching approach. This switching methodology allowed to generate balancing motions online which provide a lower cost in comparison to a non-switching approach. Even for larger impact forces like in Fig. 6.8, our switching approach provides a balancing motion which is able to bring the robot closer to the upright posture in comparison to the non-switching approach. At the same time, all the constraints imposed on the GRFs and torque limits are satisfied.

### 6.4.3 Double Impact Experiment

This experiment aims to demonstrate how the proposed methodology is able to balance the robot when it is further disturbed while not in the rest posture yet. For that, we start by applying an impact force $\boldsymbol{F}^{1}_{\text{push}} = [20\ 5]^{\top}$ with an application point of 0.68. After 1 s, while the robot has not reached the rest posture yet, we apply a second impact force $\boldsymbol{F}^{2}_{\text{push}} = [20\ 10]^{\top}$ with an application point of 0.65. The resulting balancing motion has an overall cost of 4.457.

By storing the database in terms of samples and applying such a switching approach we are able to endow reactive characteristics to our approach. In order to detect the second impact we can take advantage of the fact that an impact will cause a considerable discontinuity in the joint velocities. If we have not chosen to switch again after the second impact, we would have to deal with a great deviation between the reference values provided by the primitive chosen at $t = 0.13$ s (when switching is shut off in Fig. 6.12) and the ones corresponding to the robot. Finally, if we chose one trajectory at $t = 0$ s and then another one at $t = 1$ s (after each impact) the cost is 6.9969, greater than our switching approach.

As is shown in Fig. 6.9, the robot is able to return to the rest posture while at the same time the torque limits (Fig. 6.10) and the CoP and friction constraints (Fig. 6.11) are all satisfied. Finally, Fig. 6.12 shows the Motion Primitive switching that takes place during the double impact experiment.

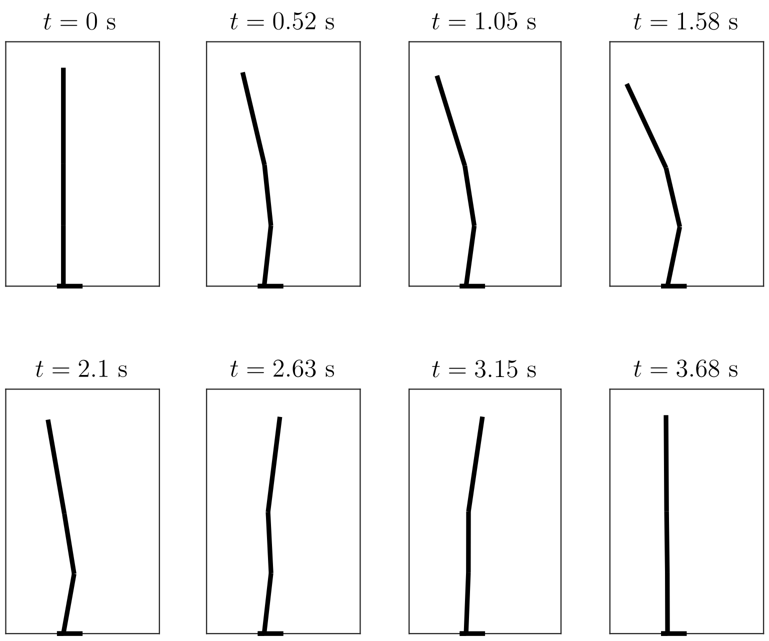


**Fig. 6.9:** Snapshots of the resulting balancing motion in the double impact experiment.

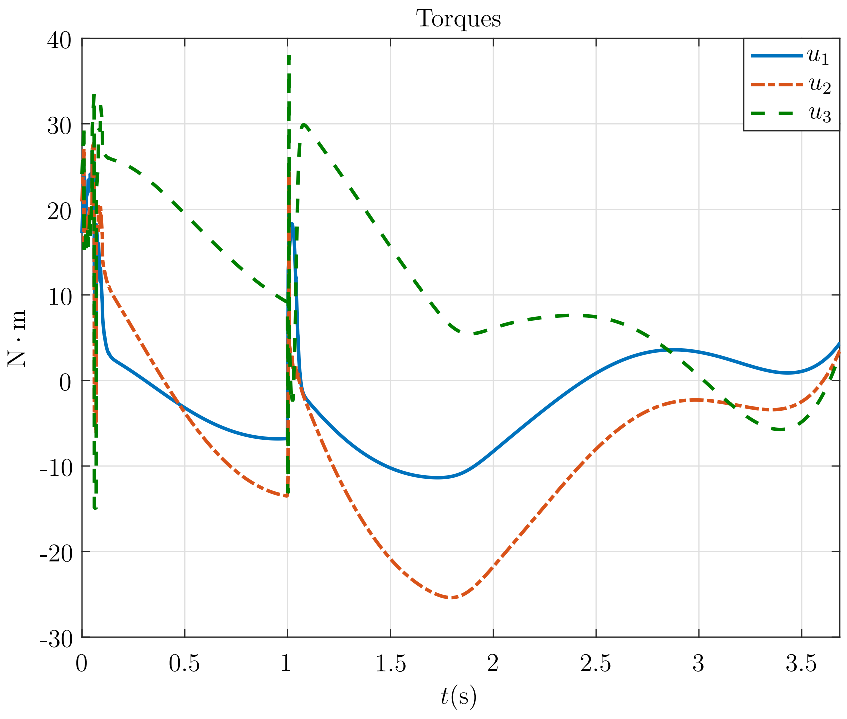


**Fig. 6.10:** Torques during the double impact experiment.

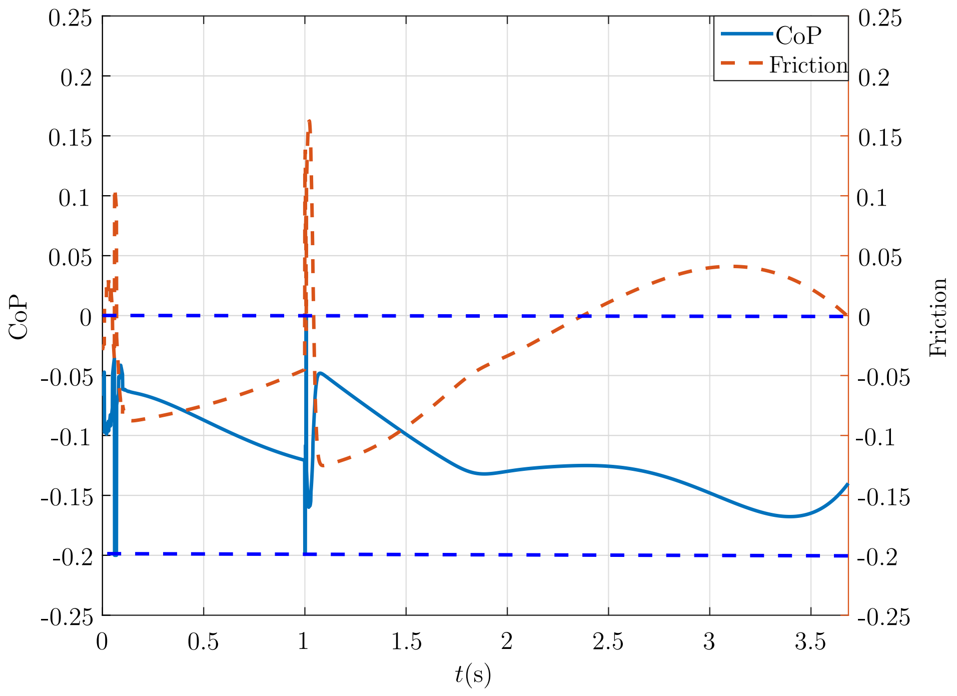


**Fig. 6.11:** CoP and friction associated with the double impact experiment. The dashed horizontal lines mark the limits for the CoP.

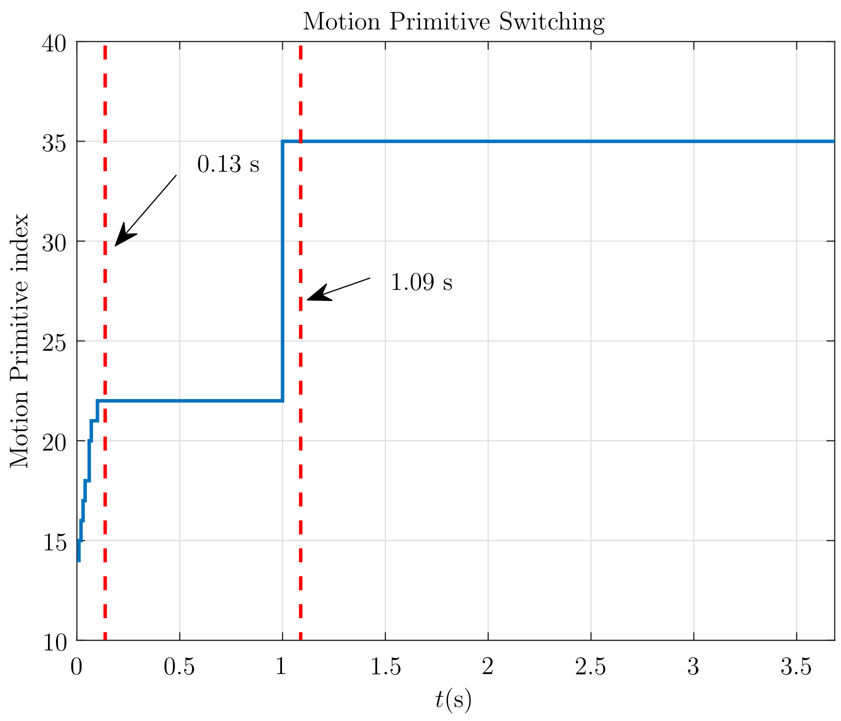


**Fig. 6.12:** Motion Primitives that choose the reference values to be forwarded to the QP for the double impact experiment. The red dashed lines indicate when the CLF constraint violation becomes smaller than the threshold $\delta_{\text{thr}}$ and the switching is shut off.

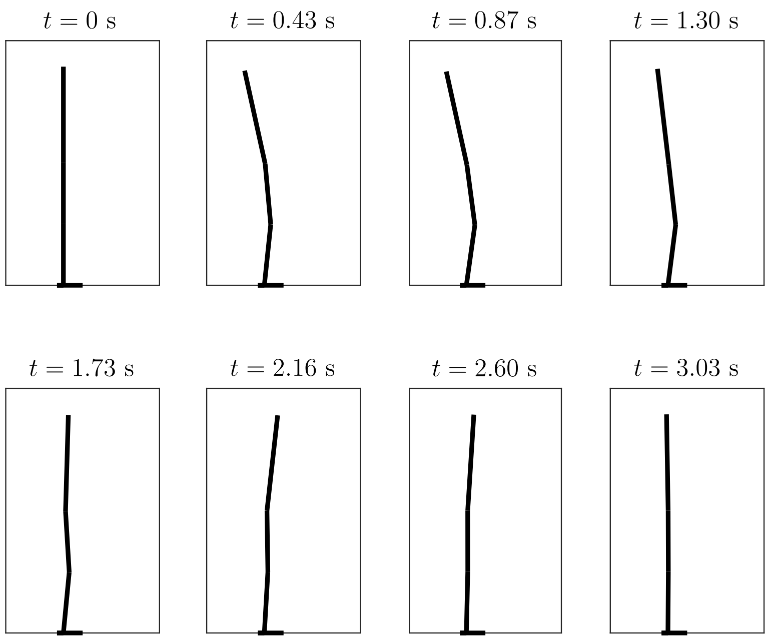


**Fig. 6.13:** Snapshots of the resulting balancing motion in the continuous push experiment.

### 6.4.4 Continuous Push

In this experiment we demonstrate that our approach can also deal with continuous pushes. For this purpose we apply for the first 400 ms a force $\boldsymbol{F}_{\text{push}} = [15\ 0]$ with an application point of 1, i.e. at the tip of the torso. What is interesting in this experiment is that as we can see in Fig. 6.13 the robot is able to return to the upright posture when applying a continuous push even though the database of primitives is generated for instantaneous impact forces. This is due to the fact that we have stored each primitive in samples and we can switch between samples of different trajectories at each time step, enhancing in that way the capabilities of the database and our approach. In addition, even though the switching is turned off after 130 ms, for the remaining 270 ms the robot is still able to withstand the push, due to the fact that after this point the motion dictated by the push and the one of the selected primitive are in good accordance with each other. Finally, if we discard the switching and try to balance the robot only with one trajectory chosen at the beginning of the motion, we are not able to bring the robot back in the upright posture (see Fig. 6.17).

### 6.4.5 Model Uncertainties - Noisy Measurements

This experiment is intended to demonstrate the capabilities of our approach to handle model uncertainties and noisy measurements. For that purpose we assume that the robot model differs from the one we used in our offline optimization phase. More precisely, we assume that the inertia of the torso is 5% more, i.e. instead of 1.33 it is 1.3965 and the inertia of the femur is 2% less, i.e. instead of 0.94 it is now 0.9212 (compare also with

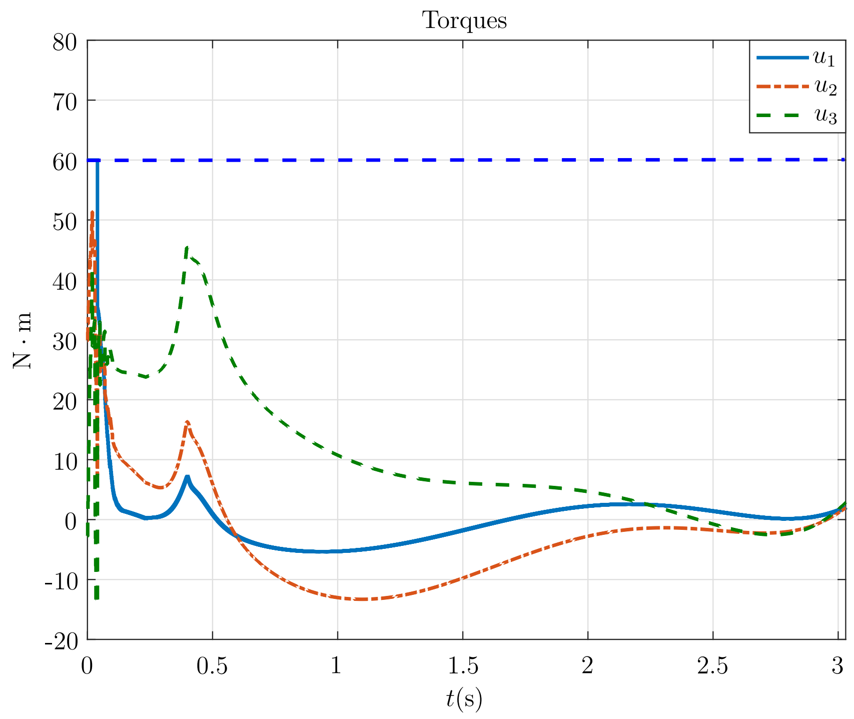


**Fig. 6.14:** Torques during the continuous push experiment.

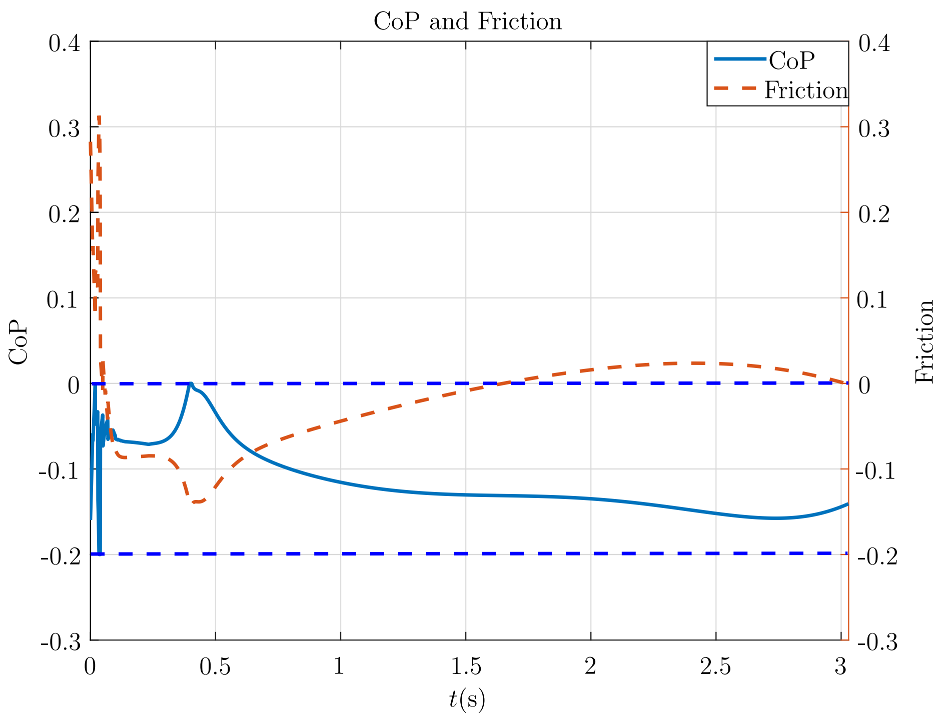


**Fig. 6.15:** CoP and friction associated with the continuous push experiment. The dashed horizontal lines mark the limits for the CoP.

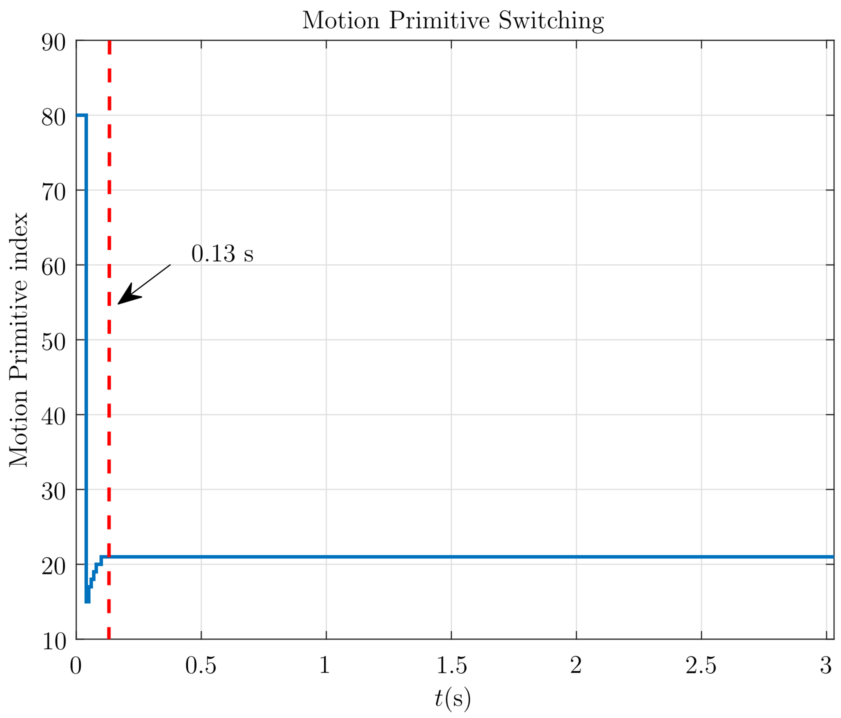


**Fig. 6.16:** Motion Primitives that choose the reference values to be forwarded to the QP for the continuous push experiment. The red dashed line indicates when the CLF constraint violation becomes smaller than the threshold $\delta_{\text{thr}}$ and the switching is shut off.

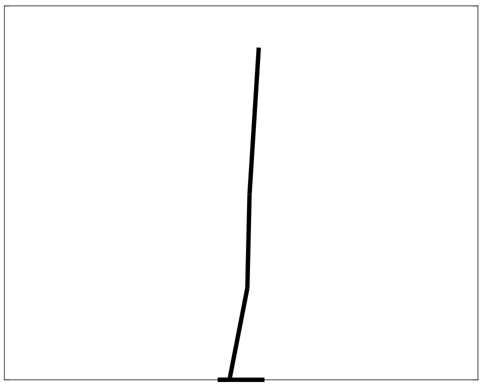


**Fig. 6.17:** Final posture of the robot that corresponds to the continuous push experiment when a non-switching approach is used.

**Tab. 6.3:** Cost of different trials of the model uncertainty and measurement noise experiment

| Trial | **1** | **2** | **3** | **4** | **5** | **average** |
|---|---|---|---|---|---|---|
| Non-switching | 18.1472 | 12.2432 | 16.3817 | 14.9073 | 13.0104 | 14.938 |
| Switching | 7.2763 | 9.2084 | 7.7048 | 8.4291 | 8.2057 | 8.2476 |

Table 2.2). At the same time we assume that the information from the joint velocities is noisy and for this purpose we add Gaussian white noise with a variance of $\sigma^2 = 0.01$. We use the same disturbance scenario as in experiment 6.4.2, i.e. $\boldsymbol{F}_{\text{push}} = [40\ 25]^\top$ with an application point of 0.7.

In Fig. 6.18 we see that the robot is able to return to the upright posture by applying the switching methodology. In Fig. 6.19 and Fig. 6.21 we see that the torques stay inside the assumed bounds and that the CoP and friction cone constraints are both satisfied. The torques and the GRFs, however, have spikes which is expected, since we have severely changed the parameters of our model and have included noise. The chattering phase of the torques and the constraints on the GRFs between 0.5 s and 1 s corresponds to high noise in the joint velocities. For that reason, we also plot the joint velocities during the uncertainty experiment in Fig. 6.20.

In order to get an estimate of the cost, we ran this experiment ten times, five times with the switching approach and five with the non-switching one. The cost of each trial is shown in Table 6.3, where the average cost for the switching approach is 8.2476 and 14.938 for the non-switching one. The figures correspond to the fourth trial which has a cost of 7.7048, which is close to the one during the experiment without any uncertainties and noise. As is shown, the parameters in Table 6.2 provide satisfactory results in case of model and measurement uncertainties. A positive aspect of the values adopted in our $\boldsymbol{L}$ matrix of the selection process is that we rely more on the joint positions in comparison to the joint velocities, since in a practical application the joint velocities are more noisy than the positions.

## 6.4.6 Overall Evaluation

This experiment is intended to demonstrate how the switching methodology can enhance the performance of the system in comparison to the classic approach of choosing a primitive only at the beginning of the motion in order to provide the reference motion. The comparison is performed in terms of cost $J_{\text{bal}}$ (Eq. (6.2)). The chosen grid consists of impact forces with a tangent and normal component and an application point of 0.7. As shown in Fig. 6.23, the cost difference is in favor of the trajectory switching methodology, showing that in terms of efficiency there is a direct advantage in performing trajectory switching rather than committing to only one trajectory chosen at the beginning of the motion. The advantage is more evident as the normal component of the impact force $F^x_{\text{push}}$ increases since the difference between the post-impact state of the robot and the state provided by the initially chosen Motion Primitive Sample increases. As a consequence, the non-switching policy will need time to converge to the optimal values provided by the

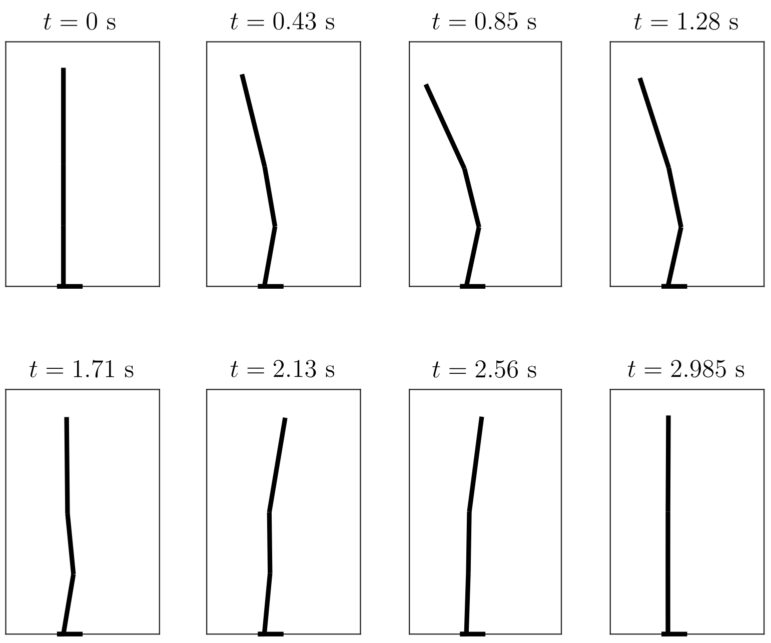


**Fig. 6.18:** Snapshots of the resulting balancing motion in the model uncertainty experiment.

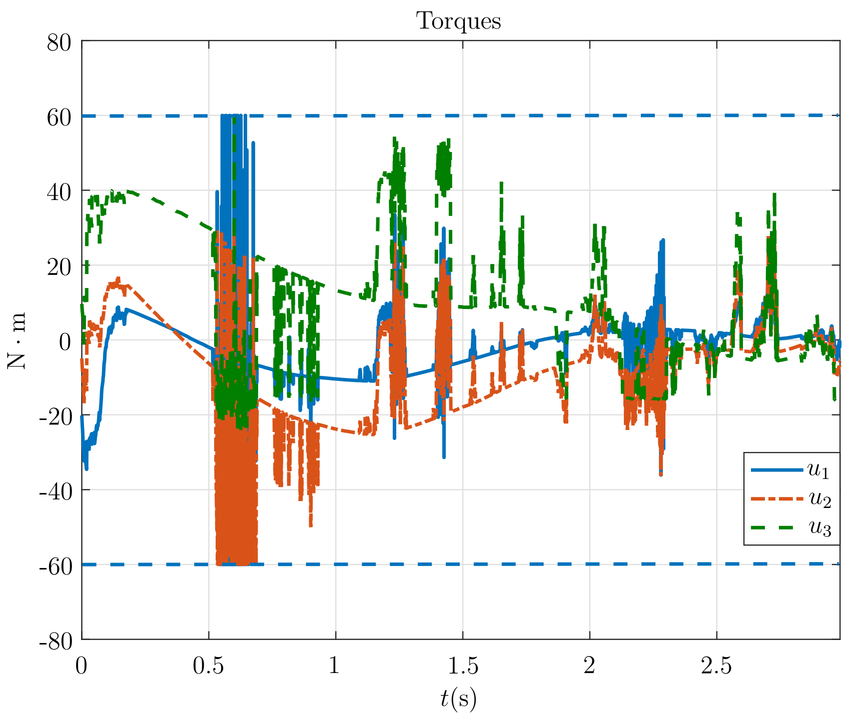


**Fig. 6.19:** Torques during the model uncertainty experiment.

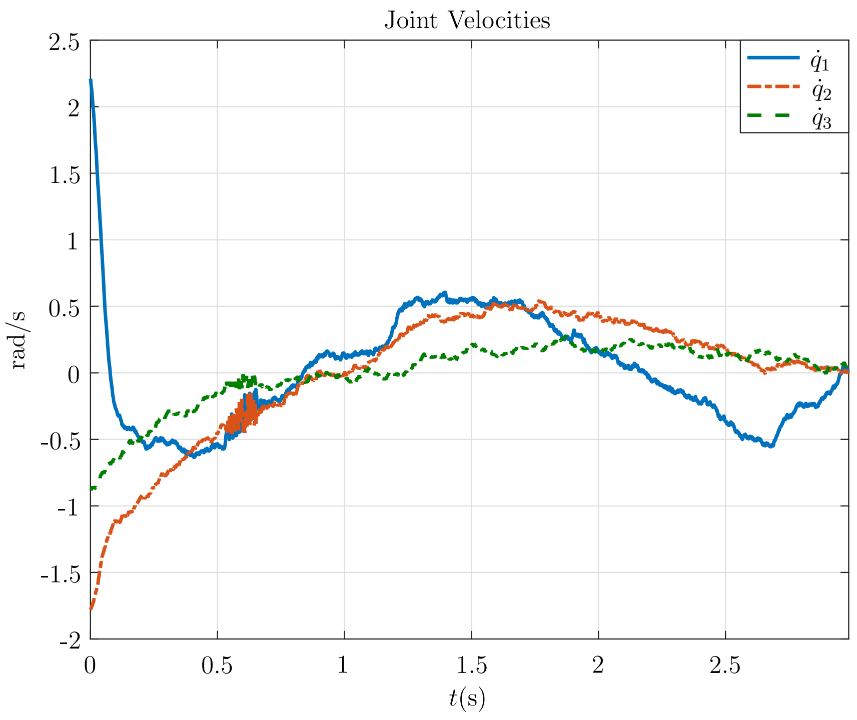


**Fig. 6.20:** Joint velocities during the model uncertainty experiment.

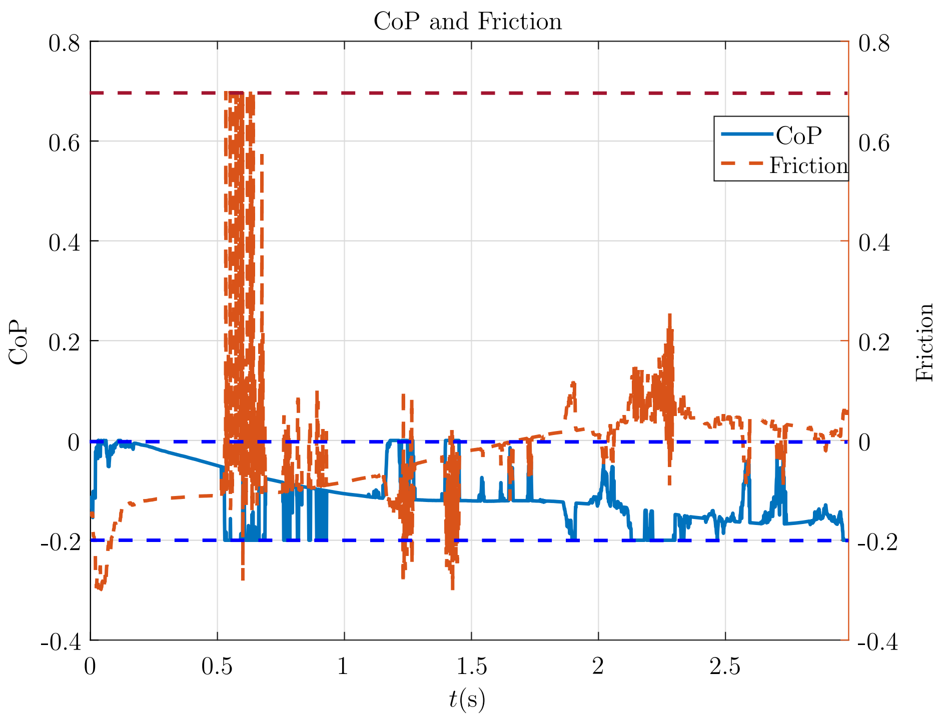


**Fig. 6.21:** CoP and friction associated with the model uncertainty experiment. The dashed horizontal lines mark the limits for the CoP and the friction.

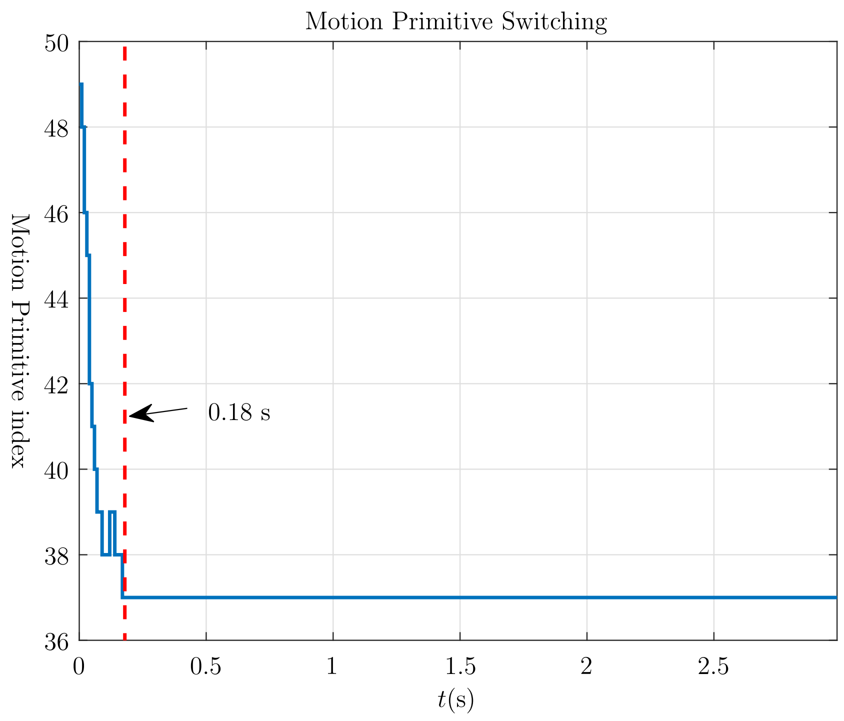


**Fig. 6.22:** Motion Primitives that choose the reference values to be forwarded to the QP for the model uncertainty experiment. The red dashed line indicates when the CLF constraint violation becomes smaller than the threshold $\delta_{\text{thr}}$ and the switching is shut off.

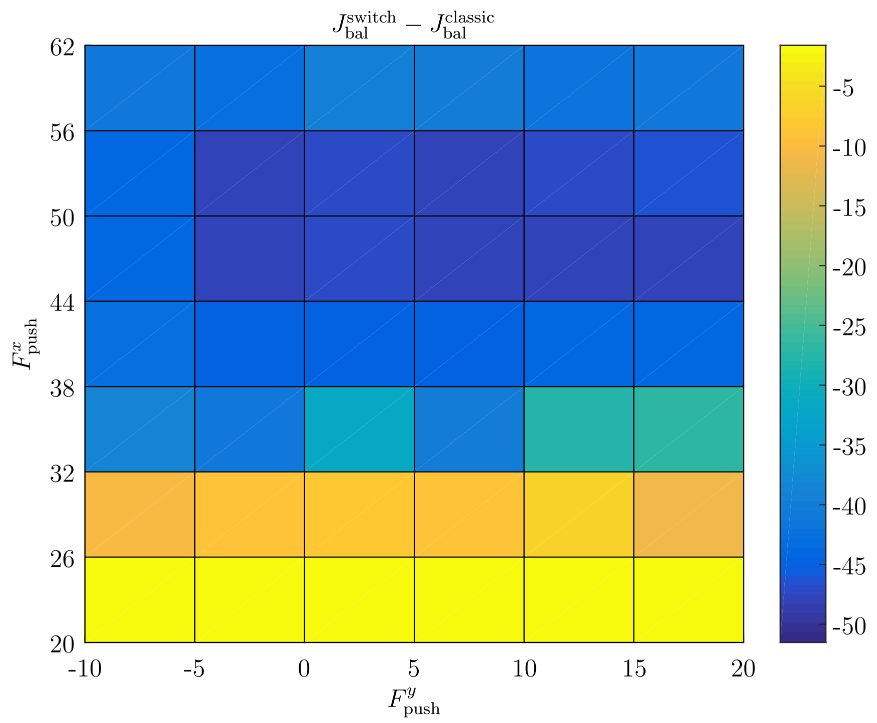


**Fig. 6.23:** An overall evaluation of the cost of the proposed methodology $J_{\text{bal}}^{\text{switch}}$ compared to the cost of the classic non-switching approach $J_{\text{bal}}^{\text{classic}}$ where we use the same primitive during the whole motion. As shown, the switching methodology can provide more efficient balancing motions for a wide range of impact forces $F_{\text{push}}$.

chosen Motion Primitive. In contrast, the switching policy has the opportunity to choose a Motion Primitive Sample $P_{[i,k]}^{*}$ that has a smaller Euclidean distance to the current state of the robot and - once switching is shut off - converge to the optimal values of the $i-$th Motion Primitive faster.

## 6.5 Summary

This chapter introduced a balancing methodology based on Motion Primitives Samples. At first, different balancing motions are calculated. Each motion brings the robot back to the rest posture after a different impact force is applied. As a further step, the trajectories, accelerations, torques and Ground Reaction Forces associated with each balancing motion are sampled and form a database of Motion Primitive Samples. When the robot is disturbed with an unknown impact force, a Motion Primitive switching algorithm takes place. Every $T_n$ seconds the database is searched and the sample closest to the current state of the robot - in terms of a Euclidean distance metric - is chosen. In such a way, for the next $T_n$ seconds reference trajectories, accelerations, torques and Ground Reaction Forces to be tracked are generated by the Motion Primitive which corresponds to the last selected sample.

In order to ensure that the tracking will not result in a motion that violates the modeling assumptions, we solve a Quadratic Program online every $T_s \leq T_n$ seconds were all these modeling assumptions are treated as constraints. The cost function penalizes the deviation from the input torques, Ground Reaction Forces and accelerations dictated by the current Motion Primitive. In addition, a Control Lyapunov Constraint is introduced that ensures exponential convergence to the desired positions and velocities at a pre-defined rate $0 < \varepsilon < 1$. The violation of the Control Lyapunov Constraint is penalized in the cost function and when it becomes lower than a user defined threshold the switching is discarded and the reference values to be tracked are generated by the Motion Primitive which corresponds to the last selected sample. The results show that this methodology can provide solutions for different impact scenarios in a robust fashion with a smaller cost in comparison to a non-switching approach where only one Motion Primitive is chosen right after the impact. Additionally, it can provide balancing motions for impact forces for which an offline optimization fails to calculate a motion.

# 7 Conclusion and Future Work

The thesis presented different ways of utilizing a database of Motion Primitives to enrich the capabilities of walking and balancing robots. We focused on different learning methods as well as optimization approaches to allow the robots to achieve generalization with respect to new task objectives. The databases were calculated offline and the generalization was achieved both online and offline. Our methodology allowed the robots to traverse uneven terrain with terrain variations which do not necessarily match the specification of the Motion Primitives in the database. In addition, our approach was able to outperform classic approaches on the task of reducing the settling time of the transition between different periodic primitives. Finally, we endowed a balancing robot with the ability to reject new and unknown disturbances by switching between the motion/balancing primitives in the database.

The focus of this thesis was walking and balancing robots, since the control of such robots is challenging and can serve as a benchmark for demonstrating the capabilities of our approach. The ideas presented within this thesis can be extended to other robotic systems, as well.

This chapter serves as a summary of the achievements of this thesis and provides future research directions.

## 7.1 Conclusions

The main conclusion of this thesis is that motion databases is a viable and beneficial approach to allow robots react to new task objectives as well as retain a degree of optimality. This was demonstrated through various studies performed in a simulation environment.

The simulation studies are realized with the use of robot models, which are described in chapter 2. In this thesis, in order to facilitate the analysis, planar robot models were used which were extracted based on the assumption of rigid bodies. A rigid contact model with the ground was used which played an important role in defining constraints for avoiding slip and foot tilting. The robot model for walking was finally defined as a hybrid system, where the dynamics of the robot undergo a discontinuous velocity jump when the swing leg impacts the ground. In order to avoid an unnecessary additional model for the swing phase with the other leg in support, the technique of coordinates relabelling was used. The models used in this thesis are a 5-link biped with point feet and a 4-link balancing robot.

A walking robot with point feet is an underactuated system. In our thesis, the 5-link biped has 5 DoFs, but only 4 actuators. Outputs can be defined only for the actuated DoFs, in the form of desired trajectories. When the outputs are zeroed, the dynamics evolve on the zero dynamics manifold. As shown in chapter 3, this manifold is of dimension 2 and facilitated the analysis of the robot motion and the dynamic feasibility of a walking motion, taking into account the impact event as well. The desired trajectories were expressed as

Bézier polynomials, which were easily integrated in the mathematical formulation of the stability and feasibility conditions. In addition, they facilitated the optimization problem for calculating walking and balancing Motion Primitives.

Online motion generation for biped robots with point feet is a challenging task, since the motion needs to be computed and checked for dynamic feasibility. Our approach to this problem was the introduction of a motion database which serves as a set of training examples for learning a regression model in order to generate new motions online. In chapter 4 we showed how a Gaussian Process can be used in order to calculate online new reference motions for such robots. This approach enabled the robot to traverse uneven terrain with terrain variations that did not match any motion in the database. The inputs to the Gaussian Process were the desired terrain variation, i.e. step height and length and the outputs were the necessary Bézier coefficients for defining the desired trajectories of the actuated DoFs. The results were presented for both periodic and aperiodic walking. Especially for aperiodic walking, in order to be able to traverse diverse terrains, we introduced a best first planning algorithm that provided a plan for a pre-defined planning horizon. The dynamic feasibility was checked on the two-dimensional zero dynamics manifold of the robot.

With chapter 5 we introduced the problem of Settling Time Reduction, which minimizes the convergence time to a target periodic orbit. This problem is interesting since convergence to orbital stable periodic orbits is dictated by the maximum eigenvalue of the Poincaré Map, whose minimization constitutes a difficult optimization problem. In general, if the state of the robot enters the domain of attraction of a stable periodic orbit, it will converge to the periodic orbit after some steps. However, a maximum eigenvalue with a great value will require the robot to take many steps before converging to the periodic orbit.

In this thesis we introduced the Settling Time Reduction problem with the help of Optimal Control. Later, we used Reinforcement Learning in order to compute composite controllers (multi-step transitions), which reduce the setting time. An important lesson from this approach was that Motion Primitives can be used as actions in a policy improvement methodology, given that the state space has a reasonably low dimension. In our thesis, this was achieved by using a function approximation for the state of the robot in its 2-dimensional zero dynamics. Another lesson is that multi-step sequences of Motion Primitives towards a periodic orbit can outperform a direct transition towards this periodic orbit in terms of speed of convergence.

Finally, in chapter 6 we demonstrated how to make good use of motion sample databases of high cardinality in terms of optimality preservation and generalization, but also in terms of robustness against model uncertainties. Our case study was balancing and we were able to endow a balancing robot with the capability to react to new and previously not known objectives while keeping a degree of optimality. In order to achieve this, we broke Motion Primitives down into samples and applied a Euclidean distance metric to choose the sample closest to the current state of the robot. The motion was tracked by solving a Quadratic Program where the constraints on the Ground Reaction Forces were imposed. In addition, the robot dynamics and the state and torque constraints were included in the formulation of the Quadratic Program. A Control Lyapunov Function constraint was introduced as well, in order to exponentially track the selected Motion Primitive and its violation was penalized. Once the Control Lyapunov Function constraint violation undershot a threshold, the switching between primitives was shut off. The methodology was successfully evaluated on different scenarios and was successfully compared to a non-switching methodology, i.e. a methodology where the Euclidean distance metric was applied only at the post-impact state of the robot.

## 7.2 Directions for Future Work

This thesis contributed towards allowing legged robots to respond to new and previously not known task objectives through the use of motion databases. After demonstrating the advantages of such an approach, different research directions can be followed to achieve further enhancements in the concept of Motion Primitives databases. We suggest the following directions:

- **Application of the method to other robotic systems:** The methodology of motion databases can find application in other domains as well. Such a methodology is attractive for systems that need to compute motions in order to solve new task objectives fast while keeping a degree of optimality. Our best candidate is the automotive field where energy efficiency plays an important role. There the methodology of switching between Motion Primitives can be applied to compute collision avoidance or overtake maneuvers that can be tracked by a driving assistance system.
- **Extending the Motion Primitive switching with Model Predictive Control:** The introduced approach was myopic in order to be computationally efficient. An interesting approach would be to extend our methodology by solving more than one QPs in a sequential manner. As a consequence, we will introduce a prediction horizon in our approach. This extension can allow to converge to one of the optimized trajectories in the database faster.
- **Real robot application:** Our thesis focused on simulation experiments, since the focus laid on the methodology. The application to real robots is an interesting step. For that, the methodology of Motion Primitive switching seems promising, since it can counteract model inaccuracies. In addition, the regression methodology and the Optimal Control formulation of the Settling Time Reduction problem could be accompanied with a robustness metric to also account for model inaccuracies.
- **Extending the regression methodology to counteract slip:** The regression and motion planning methodology could be accompanied with a QP approach similar to the one of chapter 6 in order to be able to counteract for slip. As a consequence, the regression methodology will be generating reference motions for the QP. With this approach, however, special care is needed in order to make sure that the motion will conclude with a pre-impact state that matches the terrain variation.
- **Regression with different models:** In this thesis we worked with Gaussian Processes due to their fast inference time. Of course there are other alternatives worth investigating. Among them, Deep Learning is of outermost interest. With such kind

of learning, the primitives of a database could still be used as training examples, but now the expectation would rise towards generating motions which are feasible for a wider range of stride lengths and heights than in chapter 4, while at the same time they respect the friction cone constraint. Another interesting topic would be an investigation on how optimality could be retained in the generated motions.

# Bibliography

[1] FZI Forschungszentrum Informatik Karlsruhe - Abteilung IDS. Lauron V, 2013, 2013 (accessed October 24, 2020). Retrieved from: https://de.wikipedia.org/wiki/Lauron_(Roboter).

[2] DLR - Institute of Robotics and Mechatronics. The DLR Crawler within a search and rescue test area, Accessed October 24, 2020. Retrieved from: https://www.dlr.de/rm-neu/en/desktopdefault.aspx/tabid-11705/#gallery/28668.

[3] Honda. Honda's humanoid robot ASIMO makes its first appearance at the FIRST Championship in St. Louis, Mo., Accessed October 24, 2020. Retrieved from: https://asimo.honda.com/gallery.aspx.

[4] DARPA Robotics Challenge (DRC) (Archived). http://www.darpa.mil/program/darpa-robotics-challenge.

[5] WALK-MAN: Whole-body Adaptive Locomotion and Manipulation. https://www.walk-man.eu/.

[6] 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. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:239–246, 2001.

[7] Sung-Hee Lee and A. Goswami. Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. *IEEE International Conference on Robotics and Automation*, pages 4667–4672, 2007.

[8] Boston Dynamics. Atlas Next Generation, 2016 (accessed October 24, 2020). Retrieved from: https://de.wikipedia.org/wiki/Atlas_(Roboter).

[9] IIT. WALK-MAN, not from Sony, Accessed October 24, 2020. Retrieved from: https://spectrum.ieee.org/automaton/robotics/humanoids/walkman-humanoid-robot-iit.

[10] Miomir Vukobratović and Branislav Borovac. Zero-Moment Point — Thirty Five Years Of Its Life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004.

[11] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(5):630–637, 2004.

[12] Twan Koolen, Tomas de Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113, 2012.

[13] K.H. Koch, K. Mombaur, O. Stasse, and P. Soueres. Optimization based exploitation of the ankle elasticity of HRP-2 for overstepping large obstacles. *IEEE-RAS International Conference on Humanoid Robots*, pages 733–740, 2014.

[14] Katja D. Mombaur, Richard W. Longman, Hans Georg Bock, and Johannes P. Schloeder. Open-loop stable running. *Robotica*, 23:21–33, 2005.

[15] G. Schultz and K. Mombaur. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783–792, 2010.

[16] More parkour atlas. https://www.youtube.com/watch?v=_sBBaNYex3E. Accessed: 20-10-2019.

[17] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag, Berlin, Heidelberg, 2007.

[18] Oussama Khatib, Peter Thaulad, Taizo Yoshikawa, and Jaeheung Park. Torque-position transformer for task control of position controlled robots. *IEEE International Conference on Robotics and Automation*, pages 1729–1734, 2008.

[19] Kai Henning Koch, Katja Mombaur, and Philippe Soueres. Optimization-based walking generation for humanoid robot. *IFAC Proceedings Volumes*, 45(22):498 – 504, 2012. 10th IFAC Symposium on Robot Control.

[20] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003.

[21] S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. *IEEE International Conference on Robotics and Automation*, pages 2589–2594, 2014.

[22] Aaron D. Ames and Matthew Powell. *Towards the Unification of Locomotion and Manipulation through Control Lyapunov Functions and Quadratic Programs*, pages 219–240. Springer's Lecture Notes in Control and Information Science, 2013.

[23] Alberto Isidori. *Nonlinear Control Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 1995.

[24] Jianjuen Hu, J. Pratt, and G. Pratt. Stable adaptive control of a bipedal walking robot with CMAC neural networks. *IEEE International Conference on Robotics and Automation*, 2:1050–1056, 1999.

[25] W. T. Miller. Real-time neural network control of a biped walking robot. *IEEE Control Systems Magazine*, 14(1):41–48, 1994.

[26] Jungho Lee and Jun Ho Oh. Biped walking pattern generation using reinforcement learning. *International Journal of Humanoid Robotics*, 06(01):1–21, 2009.

[27] J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin. A simple reinforcement learning algorithm for biped walking. *IEEE International Conference on Robotics and Automation*, 3:3030–3035, 2004.

[28] M. A. Lewis, A. H. Fagg, and A. Solidum. Genetic programming approach to the construction of a neural network for control of a walking robot. *IEEE International Conference on Robotics and Automation*, 3:2618–2623, 1992.

[29] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot. *The International Journal of Robotics Research*, 27(2):213–228, 2008.

[30] Takeshi Mori, Yutaka Nakamura, Masa-aki Sato, and Shin Ishii. Reinforcement learning for CPG-driven biped robot. *AAAI Proceedings of the 19th National Conference on Artificial Intelligence*, pages 623–630, 01 2004.

[31] Ch. Glocker and F. Pfeiffer. Dynamical systems with unilateral contacts. *Nonlinear Dynamics*, 3(4):245–259, 1992.

[32] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2nd edition, 2000.

[33] 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. *IEEE International Conference on Robotics and Automation*, 2:1620–1626, 2003.

[34] Jerry Pratt, Twan Koolen, Tomas de Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter Neuhaus. Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133, 2012.

[35] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behavior explains basic dynamics of walking and running. *Proceedings of Biological sciences*, 273:2861–2867, 2006.

[36] I. Poulakakis. Spring loaded inverted pendulum embedding: Extensions toward the control of compliant running robots. *2010 IEEE International Conference on Robotics and Automation*, pages 5219–5224, 2010.

[37] André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003.

[38] A. R. Ansari and T. D. Murphey. Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems. *IEEE Transactions on Robotics*, 32(5):1196–1214, 2016.

[39] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear model predictive control: A tutorial and survey. *IFAC Proceedings Volumes*, 27(2):185 – 197, 1994.

[40] M. J. Powell, E. A. Cousineau, and A. D. Ames. Model predictive control of underactuated bipedal robotic walking. *IEEE International Conference on Robotics and Automation*, pages 5121–5126, 2015.

[41] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001.

[42] B.J. Stephens and C.G. Atkeson. Dynamic balance force control for compliant humanoid robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1248–1255, 2010.

[43] K. Yamane and Y. Nakamura. Dynamics filter - concept and implementation of online motion generator for human figures. *IEEE Transactions on Robotics and Automation*, 19(3):421–432, 2003.

[44] S. H. Hyon, J. G. Hale, and G. Cheng. Full-body compliant human-humanoid interaction: Balancing in the presence of unknown external forces. *IEEE Transactions on Robotics*, 23(5):884–898, 2007.

[45] I.R. Manchester and J. Umenberger. Real-time planning with primitives for dynamic walking over uneven terrain. *IEEE International Conference on Robotics and Automation*, pages 4639–4646, 2014.

[46] S. Apostolopoulos, M. Leibold, and M. Buss. Online motion planning over uneven terrain with walking primitives and regression. *IEEE International Conference on Robotics and Automation*, pages 3799–3805, 2016.

[47] T. Yang, E.R. Westervelt, and A. Serrani. A framework for the control of stable aperiodic walking in underactuated planar bipeds. *IEEE International Conference on Robotics and Automation*, pages 4661–4666, 2007.

[48] Sotiris Apostolopoulos, Marion Leibold, and Martin Buss. Transitioning between underactuated periodic orbits: An optimal control approach for settling time reduction. *International Journal of Humanoid Robotics*, 15(06):1850027, 2018.

[49] S. Apostolopoulos, M. Leibold, and M. Buss. Settling time reduction for underactuated walking robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6402–6408, 2015.

[50] Sotiris Apostolopoulos, Marion Leibold, and Martin Buss. Energy efficient and robust balancing with motion primitive switching. *International Journal of Humanoid Robotics*, 14(03):1750009, 2017.

[51] Marion Sobotka. *Hybrid Dynamical System Methods for Legged Robot Locomotion with Variable Ground Contact.* Dissertation, Technische Universität München, München, 2007.

[52] Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors. *Hybrid Systems.* Springer-Verlag Berlin Heidelberg, 1 edition, 1993.

[53] Panos Antsaklis, Wolf Kohn, Anil Nerode, and S.S. Sastry, editors. *Hybrid Systems II.* Springer-Verlag Berlin Heidelberg, 1 edition, 1995.

[54] R. Alur, T.A. Henziger, and E.D. Sontag, editors. *Hybrid Systems III.* Springer-Verlag Berlin Heidelberg, 1 edition, 1996.

[55] Panos Antsaklis, Wolf Kohn, Anil Nerode, and S.S. Sastry, editors. *Hybrid Systems IV.* Springer-Verlag Berlin Heidelberg, 1 edition, 1997.

[56] Panos Antsaklis, Wolf Kohn, Michael Lemmon, Anil Nerode, and S.S. Sastry, editors. *Hybrid Systems V.* Springer-Verlag Berlin Heidelberg, 1 edition, 1999.

[57] M. Buss. Hybrid control of mechatronic systems. *Journal on Systems, Control and Information; Institute of Systems, Control and Information Engineers, ISCIE*, 46(3):126, 2002.

[58] Mariano Garcia, Anindya Chatterjee, Andy Ruina, and Michael Coleman. The simplest walking model: Stability, complexity, and scaling. *ASME Journal of Biomechanical Engineering*, 120:281–288, 1998.

[59] David Morin. *Introduction to Classical Mechanics: With Problems and Solutions.* Cambridge University Press, 2008.

[60] P. Sardain and G. Bessonnet. Gait analysis of a human walker wearing robot feet as shoes. *IEEE International Conference on Robotics and Automation*, 3:2285–2292 vol.3, 2001.

[61] Ambarish Goswami. Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point. *The International Journal of Robotics Research*, 18:523–533, 1999.

[62] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. Westervelt, C. Canudas-de Wit, and J. Grizzle. RABBIT: a testbed for advanced control theory. *IEEE Control Systems Magazine*, pages 57–79, 2003.

[63] David A. Levinson and Thomas R. Kane. *Multibody Systems Handbook*, chapter AUTOLEV - A New Approach to Multibody Dynamics, pages 81–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[64] Rafael Kelly, Victor Santibáñez Davila, and Julio Antonio Loría Perez. *Control of Robot Manipulators in Joint Space.* 2005.

[65] L. Sciavicco, Bruno Siciliano, and B. Sciavicco. *Modelling and Control of Robot Manipulators*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 2000.

[66] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[67] Eric Westervelt, Jessy Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC Press, Boca Raton, 2007.

[68] Carlos Canudas de Wit. On the concept of virtual constraints as a tool for walking robot control and balancing. *Annual Reviews in Control*, 28(2):157 – 166, 2004.

[69] Brent Griffin and Jessy Grizzle. Nonholonomic virtual constraints and gait optimization for robust walking control. *The International Journal of Robotics Research*, 36(8):895–922, 2017.

[70] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. CRC Press, 2015.

[71] P. W. M. van Zutven. *Control and identification of bipedal humanoid robots : stability analysis and experiments*. Dissertation, Technische Universiteit Eindhoven, 2014.

[72] Reza Dehghani and Abbas Fattah. Stability analysis and robust control of a planar underactuated biped robot. *International Journal of Humanoid Robotics*, 07(04):535–563, 2010.

[73] D. Djoudi, C. Chevallereau, and Y. Aoustin. Optimal reference motions for walking of a biped robot. *IEEE International Conference on Robotics and Automation*, pages 2002–2007, 2005.

[74] Chenggang Liu and C.G. Atkeson. Standing balance control using a trajectory library. *IEEE International Conference on Intelligent Robots and Systems*, pages 3031–3036, 2009.

[75] E. Frazzoli, M.A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[76] J. Denk and G. Schmidt. Synthesis of walking primitive databases for biped robots in 3D environments. *IEEE International Conference on Robotics and Automation, 2003*, 1:1343–1349, 2003.

[77] R.D. Gregg, A.K. Tilton, S. Candido, T. Bretl, and M.W. Spong. Control and Planning of 3D Dynamic Walking With Asymptotically Stable Gait Primitives. *IEEE Transactions on Robotics*, 28(6):1415–1423, 2012.

[78] Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[79] A. Werner, D. Trautmann, Dongheui Lee, and R. Lampariello. Generalization of optimal motion trajectories for bipedal walking. *IEEE International Conference on Intelligent Robots and Systems, 2015*, pages 1571–1577, 2015.

[80] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *J. Mach. Learn. Res.*, 11:3011–3015, 2010.

[81] M. W. Spong. The swing up control problem for the acrobot. *IEEE Control Systems*, 15(1):49–55, 1995.

[82] S. Bouabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. *IEEE International Conference on Robotics and Automation*, pages 2247–2252, 2005.

[83] N. Sun, Y. Fang, H. Chen, Y. Fu, and B. Lu. Nonlinear stabilizing control for ship-mounted cranes with ship roll and heave movements: Design, analysis, and experiments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–13, 2017.

[84] N. Sun, Y. Fang, H. Chen, and B. Lu. Amplitude-saturated nonlinear output feedback antiswing control for underactuated cranes with double-pendulum cargo dynamics. *IEEE Transactions on Industrial Electronics*, 64(3):2135–2146, 2017.

[85] K. D. Mombaur, H. G. Bock, J. P. Schloder, and R. W. Longman. Self-stabilizing somersaults. *IEEE Transactions on Robotics*, 21(6):1148–1157, 2005.

[86] Sushant Veer, Mohamad Shafiee Motahar, and Ioannis Poulakakis. On the adaptation of dynamic walking to persistent external forcing using hybrid zero dynamics control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 997–1003, 2015.

[87] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W. Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL. *The International Journal of Robotics Research*, 32(3):324–345, 2013.

[88] Van Oijen, Tim P. and Karssen, J. G. Daniël and Wisse, Martijn. The effect of center of mass offset on the disturbance rejection of running robots. *International Journal of Humanoid Robotics*, 10(02):1350004, 2013.

[89] Yang Yi, Zhiyun Lin, and Gangfeng Yan. Variable speed running on kneed biped robot with underactuation degree two. *International Journal of Humanoid Robotics*, 11(02):1450015, 2014.

[90] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[91] M. Diehl, H.G. Bock, H. Diedam, and P.-B. Wieber. *Fast Direct Multiple Shooting Algorithms for Optimal Robot Control*, pages 65–93. Springer Berlin Heidelberg, 2006.

[92] Max K. Agoston. *Computer Graphics and Geometric Modelling: Implementation & Algorithms.* Springer-Verlag New York, Inc., 2004.

[93] R.R. Burridge, A.A. Rizzi, and D.E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18:534–555, 1999.

[94] E. Najafi, G.A.D. Lopes, and R. Babuska. Reinforcement learning for sequential composition control. *IEEE 52nd Annual Conference on Decision and Control*, pages 7265–7270, 2013.

[95] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[96] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 2013.

[97] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators.* CRC press, 2010.

[98] C. G. Atkeson and B. Stephens. Multiple balance strategies from one optimization criterion. *IEEE-RAS International Conference on Humanoid Robots*, pages 57–64, 2007.

[99] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. *IEEE-RAS International Conference on Humanoid Robots*, pages 200–207, 2006.

[100] Juan Alejandro Castano, Zhibin Li, Chengxu Zhou, Nikos Tsagarakis, and Darwin Caldwell. Dynamic and reactive walking for humanoid robots based on foot placement control. *International Journal of Humanoid Robotics*, 2015.

[101] Z. Aftab, T. Robert, and P. B. Wieber. Ankle, hip and stepping strategies for humanoid balance recovery with a single model predictive control scheme. *IEEE-RAS International Conference on Humanoid Robots*, pages 159–164, 2012.

[102] B. J. Stephens and C. G. Atkeson. Push recovery by stepping for humanoid robots with force controlled joints. *IEEE-RAS International Conference on Humanoid Robots*, pages 52–59, 2010.

[103] A.D. Kuo. An optimal control model for analyzing human postural balance. *IEEE Transactions on Biomedical Engineering*, 42(1):87–101, 1995.

[104] Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization.* Dissertation, California Institute of Technology, 2000.

[105] Sung-Hee Lee and Ambarish Goswami. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, 33(4):399–414, 2012.

[106] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

[107] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 981–988, 2014.

[108] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[109] J. Englsberger, C. Ott, and A. Albu-Schäffer. Three-dimensional bipedal walking control using divergent component of motion. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2600–2607, 2013.

[110] Michael A. Hopkins, Alexander Leonessa, Brian Y. Lattimer, and Dennis W. Hong. Optimization-based whole-body control of a series elastic humanoid robot. *International Journal of Humanoid Robotics*, 13(01):1550034, 2016.

[111] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 59(4):876–891, 2014.

[112] Jemin Hwangbo, Christian Gehring, Hannes Sommer, Roland Siegwart, and Jonas Buchli. Policy learning with an efficient black-box optimization algorithm. *International Journal of Humanoid Robotics*, 12(03):1550029, 2015.

[113] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[114] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. *IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[115] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. *15th International Conference on Neural Information Processing Systems*, pages 1547–1554, 2002.

[116] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

[117] Zvi Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163 – 1173, 1983.

[118] Eduardo D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117 – 123, 1989.

[119] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.