MPC-RL-based bipedal robot control

Constant Roux

UPSSITECH

Research Project

MPC-RL-based bipedal robot control

Author	Constant ROUX
Promotion	SRI 2024 - Year 2023
Tutor	Viviane CADENAT
Supervisor	Olivier STASSE
Co-Supervisor	Philippe SOUERES
Dates	march 2024 — august 2024

Acknowledgement

First and foremost, I would like to thank Olivier Stasse. As my internship supervisor and future thesis director, he has guided me through my work over the past six months, providing his expertise in numerical optimization and humanoid robotics. He also facilitated my integration into the team.

I also wish to thank Thomas Flayols for his training and expertise in the mechatronic design of ODRI robots. Additionally, I am grateful to Jérôme Manhes, who helped me identify and fix various electronic issues.

I would like to express my gratitude to Viviane Cadenat for her support and supervision during my internship at LAAS-CNRS. I also thank the entire teaching staff at UPSSITECH.

Lastly, I extend my thanks to the interns, PhD students, postdoctoral researchers, and permanent members of the GEPETTO team for their expertise and warm welcome. I am also grateful to all the staff at LAAS-CNRS.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ANITI Artificial and Natural Intelligence Toulouse Institute. 7

CaT Constraints as Terminations. 2, 3, 6, 23, 29, 30, 35

CNRS National Centre for Scientific Research. 2, 3, 5, 6, 7

CoM Center of Mass. 3, 15, 16, 17, 18, 20, 25, 26, 30, 34

CoP Center of Pressure. 38

CPU central Processing Unit. 16

DCM Divergent Component of Motion. 2, 3, 5, 9, 10, 11, 12, 13, 14, 16, 17, 18, 20, 21, 22, 35, 38, 39, 40

DDP Differential Dynamic Programming. 2, 3, 5, 16, 41, 42, 43

FDDP Feasibility-Driven Differential Dynamic Programming. 16

iLQG Linear Quadratic Gaussian. 5

IMU Inertial Measurement Unit. 3, 32

LAAS Laboratory for Analysis and Architecture of Systems. 2, 3, 5, 6, 7

LIPM Linearized Inverted Pendulum Model. 5, 18, 21, 22, 38

MPC Model Predictive Control. 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 35

OCP Optimal Control Problem. 13, 41

ODRI Open Dynamic Robot Initiative. 1, 2, 7, 31, 32, 33, 35

QP Quadratic programming. 9, 10, 11

RL Reinforcement learning. 3, 5, 6, 8, 22, 23, 25, 35

SRDF Semantic Robot Description Format. 16

TO Trajectory Optimization. 5

URDF Unified Robot Description Format. 16, 33, 34, 35

WB Whole-Body. 5, 6, 11, 13, 15, 16, 21, 22, 35

ZMP Zero Moment Point. 5

1 Introduction

1.1 Context

Bipedal robotics, with its origins tracing back to the end of the last century, has witnessed a significant surge in recent years. This trend, driven by technological advancements in areas such as actuation, and computing, has opened up a flurry of new potential applications [1, 2, 3]. However, these new solutions require efficient controllers that can fully exploit the hardware's specificities to maximize utility.

Historically, Trajectory Optimization (TO) has been widely used for the control of bipedal robots. Early models, such as the Linearized Inverted Pendulum Model (LIPM) or centroidal Model Predictive Control (centroidal MPC), evolved into more sophisticated approaches like Whole-Body MPC (WB MPC) [4, 5, 6, 7, 8]. However, as the complexity of control problems increases, the limitations of MPC become apparent. These limitations include limited reactivity to unforeseen disturbances and insufficient computational capacity. Even though warm starting the solver near the optimal solution can drastically reduce computation time [9], these challenges persist.

Reinforcement learning (RL) techniques have emerged as a promising alternative [10, 11, 12]. RL nonetheless presents notable limitations. Bipedal robots are inherently very unstable, complicating the learning process. Additionally, RL requires vast amounts of data and computational time, and often lacks performance guarantees in real-world conditions. RL algorithms can also be sensitive to variations in the training environment, limiting their robustness and generalizability.

In response to these challenges, hybrid methods, combining the strengths of trajectory optimization and reinforcement learning, are gaining popularity. Approaches presented in [13, 14] aim to merge the best of both worlds to achieve optimal performance in terms of stability and adaptability. The enthusiasm around hybrid methods highlights the relevance of traditional MPC methods, especially when dealing with highly unstable humanoids. However, in most applications, they require an additional trajectory planner, which may not account for the dynamics of the MPC, to provide a reference trajectory for the robot.

Despite increasingly precise modeling and step sequencers based on simplified yet effective models, such as the Divergent Component of Motion (DCM) [15, 16] or the Zero Moment Point (ZMP) [17], these planners are still necessary to provide robust heuristics for bipedal walking. An example of such architecture is presented in [18]. It employs a step planner followed by a foot trajectory controller and a whole-body instantaneous controller. Similarly, [19] conducted research using a low-frequency step planner (10 Hz) and a WB controller. A method based on iterative Linear Quadratic Gaussian (iLQG) [20] provides the capabilities to find foot sequences autonomously, but struggles to efficiently work on real hardware [21]. However, Differential Dynamic Programming (DDP) with a rigid contact formulation such as the one formulated in [22] and tested on Talos in [7] shows that if a sequence of contacts is given, WB MPC is doable on a real size humanoid robot.

1.2 Report outline

In this report, we present the following:

- we begin with an introduction to LAAS-CNRS and the GEPETTO team.
- Then, we propose to build upon the work of Khadiv et al. [18] by replacing the combination of the WB instantaneous controller and the footstep trajectory controller with a unified WB trajectory controller. Our approach aims to operate both the step sequencer and

Fig. 1: Bolt walking at LAAS-CNRS using RL.

the whole-body model predictive controller (WB MPC) at the same frequency of 100 Hz, contrary to Yeganegi et al. [19]. Furthermore, we propose a parameter sensitivity analysis of the step sequencer based on the method proposed by Fiacco [23].

- Next, we describe the RL-based control approach using Constraints as Terminations (CaT), which enables the robot to learn to walk in simulation while also learning to avoid and recover from undesirable states. To this end, we modify the work of Chane-sane et al. [24], adapting their project from the quadrupede solo12 to a biped.
- We conclude with a brief summary of the findings and implications of our research.

The described technique was implemented in simulation on the Bolt robot (See Fig. 1), an affordable bipedal hardware platform designed for students and scholars. This platform utilizes an actuator concept developed by [25].

2 Host Organization

2.1 LAAS-CNRS

The Laboratory for Analysis and Architecture of Systems (LAAS) is a Toulouse, France-based research unit of the National Centre for Scientific Research (CNRS). LAAS is a globally recognized laboratory that was founded in 1983 and specializes in the analysis and design of complex systems. Its interdisciplinary approach, which combines knowledge of computer science, robotics, electronics, automation, and applied mathematics, sets it apart.

The research teams at LAAS CNRS work together on cutting-edge topics that span from embedded systems design to autonomous robotics. The lab is notable for its dedication to basic science and its development of useful applications in fields like mobility, health, and the environment. Modern facilities and solid partnerships with academic and industry partners are advantages for LAAS CNRS.

2.2 GEPETTO

The GEPETTO team, founded in 2006 and now consisting of over 40 members, focuses on the movement of anthropomorphic systems. It is currently best known for its work in humanoid robotics and motion generation.

One of the main activities of the team is developing new motion generation techniques and control laws executable on physical robots (See Fig. 2[1].). These efforts are primarily based on optimal control, utilizing for example the Crocoddyl software. Most of this work is conducted within the frameworks of Artificial and Natural Intelligence Toulouse Institute (ANITI) and AGIMUS.

Fig. 2: HRP2 walking with a dumbbell in a complex environment.

The team also works on the mechatronic design of robots, whether in collaboration with PAL Robotics on their humanoid robot Talos (See Fig. 3[2]) or on a smaller scale with ODRI's robots

[1] image extracted from this https://images.cnrs.fr/video/2159

[2] image extracted from this https://theconversation.com/pourquoi-marcher-est-il-si-difficile-pour-un-robot-201996.

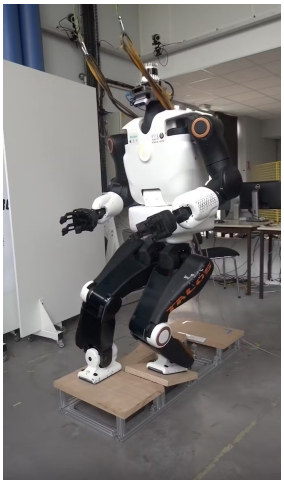


Fig. 3: Talos walking on uneven terrain.

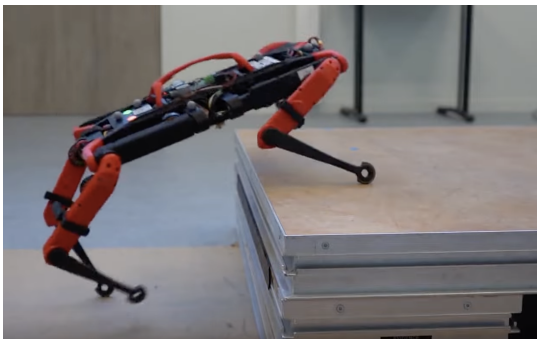


Fig. 4: Solo12 climbing a box using RL.

Fig. 5: Bolt walking using MPC.

Solo12 (See Fig. 4[3]) and Bolt (See Fig. 5[4]).

[3] image extracted from this https://www.youtube.com/watch?v=crWoYTb8QvU.
[4] image extracted from this https://www.youtube.com/watch?v=U8qYgGROyCM.

3 Whole-body MPC based control

The following section is largely taken from a paper submitted to the Humanoids conference [26].

3.1 Footsteps Sequencer

3.1.1 Review of previous work

Khadiv et al. [18] proposed to compute the position and timing of the next step so that the robot's center of mass adheres to a desired velocity command while stabilizing the DCM. The DCM is the unstable component of the linear inverted pendulum model (See Appendix A), whose dynamic equations are recalled as follows:

$$\dot{\mathbf{c}} = w_0(\zeta - \mathbf{c}) \tag{1a}$$

$$\dot{\zeta} = w_0(\zeta - \mathbf{p}_0) \tag{1b}$$

where $\mathbf{c} \in \mathbb{R}^2$ is the position of the robot's center of mass, $\omega_0 \in \mathbb{R}$ is the natural frequency of the pendulum ($\omega_0 = \sqrt{\frac{g}{z_c}}$, with g being the gravitational constant and z_c the fixed height of the center of mass), $\zeta \in \mathbb{R}^2$ is the robot's DCM, and $\mathbf{p}_0 \in \mathbb{R}^2$ is the position of the support foot.

To compute the best guest for the next step, Khadiv et al. [18] propose solving the following linearly constrained multi-objective QP problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \begin{cases} g_i(\mathbf{x}) \geq 0, & i = 1, \dots, 6 \\ \hat{h_j}(\mathbf{x}) = 0, & j = 1, \dots, 2 \end{cases} \end{aligned} \tag{2}$$

where $\mathbf{x} = \begin{pmatrix} \mathbf{p}_T^\top & \Gamma(T) & \mathbf{b}_T^\top \end{pmatrix}^\top$ with $\mathbf{p}_T = \begin{pmatrix} p_{T,x} & p_{T,y} \end{pmatrix}^\top \in \mathbb{R}^2$ representing the position of the next step, $\Gamma(T) = e^{w_0 T} \in \mathbb{R}$ the timing of the next step's contact, and $\mathbf{b}_T \in \mathbb{R}^2$ the DCM offset at the next step's contact instant ($\mathbf{b}_T = \begin{pmatrix} b_{T,x} & b_{T,y} \end{pmatrix}^\top = \zeta_T - \mathbf{p}_T$, where ζ_T is the DCM at the next step's contact instant). The multi-objective function is given as follows:

$$\begin{aligned} f(\mathbf{x}) = \alpha_1 & \left\| \mathbf{p}_T - \mathbf{p}_0 - \begin{pmatrix} l_{\text{nom}} \\ w_{\text{nom}} \end{pmatrix} \right\|^2 \\ & + \alpha_2 \left| \Gamma(T) - \Gamma(T_{\text{nom}}) \right|^2 \\ & + \alpha_3 \left\| \mathbf{b}_T - \begin{pmatrix} b_{x,\text{nom}} \\ b_{y,\text{nom}} \end{pmatrix} \right\|^2 \end{aligned} \tag{3a}$$

with α_1, α_2, α_3 as the weights of the different objectives, l_{nom} and w_{nom} as the desired step length and width respectively, $\Gamma(T_{\text{nom}})$ the desired contact time, and $b_{x,\text{nom}}$, $b_{y,\text{nom}}$ the desired DCM offsets whose analytical expression is available in Appendix B. The inequality constraints are given as follows:

$$g_1(\mathbf{x}) = p_{T,x} - p_{0,x} - l_{\min} \tag{4a}$$

$$g_2(\mathbf{x}) = l_{\max} - p_{T,x} + p_{0,x} \tag{4b}$$

which are constraints limiting the step length between $l_{\min}$ and $l_{\max}$,

$$g_3(\mathbf{x}) = p_{T,y} - p_{0,y} - w_{\min} \tag{4c}$$

$$g_4(\mathbf{x}) = w_{\max} - p_{T,y} + p_{0,y} \tag{4d}$$

which are constraints limiting the step width between $w_{\min}$ and $w_{\max}$,

$$g_5(\mathbf{x}) = \Gamma(T) - \Gamma(T_{\min}) \tag{4e}$$

$$g_6(\mathbf{x}) = \Gamma(T_{\max}) - \Gamma(T) \tag{4f}$$

which are constraints limiting the step contact time between $\Gamma(T_{\min})$ and $\Gamma(T_{\max})$.

The following equations are equality constraints ensuring that the DCM follows the dynamics imposed by Eq. (1b):

$$\hat{h_1}(\mathbf{x}) = p_{T,x} + b_{T,x} - p_{0,x} - (\hat{\zeta}_x - p_{0,x})e^{-w_0 t}\Gamma(T) \tag{5a}$$

$$\hat{h_2}(\mathbf{x}) = p_{T,y} + b_{T,y} - p_{0,y} - (\hat{\zeta}_y - p_{0,y})e^{-w_0 t}\Gamma(T) \tag{5b}$$

with t the time elapsed since the last foot contact and $\hat{\zeta}$ the measured DCM. Note that, as explained in [27], the DCM offset is not hard constrained because this could make the problem infeasible under certain conditions. In addition, w will sometimes be referred to as w_{left} or w_{right} depending on whether the next step is executed by the left foot or the right foot.

3.1.2 Sensitivity analysis to disturbances on the measured DCM

We now propose to analyze the sensitivity of the optimal solution to perturbations on the measured DCM. These perturbations can model measurement noise or an external force that would alter the value of the DCM.

Introducing a disturbance on the measured DCM such that $\hat{\zeta} = \zeta + \theta$, where $\zeta \in \mathbb{R}^2$ is the real value of the DCM and $\theta \in \mathbb{R}^2$ is a disturbance, the problem (2) then becomes:

$$\begin{aligned} &\min_{\mathbf{x}} \quad f(\mathbf{x}) \\ &\text{s.t.} \quad \begin{cases} g_i(\mathbf{x}) \geq 0, & i = 1, \dots, 6 \\ h_j(\mathbf{x}) + \theta_j c_j(\mathbf{x}) = 0, & j = 1, \dots, 2 \end{cases} \end{aligned} \tag{6}$$

$$h_1(\mathbf{x}) = p_{T,x} + b_{T,x} - p_{0,x} - (\zeta_x - p_{0,x})e^{-w_0 t}\Gamma(T) \tag{7a}$$

$$h_2(\mathbf{x}) = p_{T,y} + b_{T,y} - p_{0,y} - (\zeta_y - p_{0,y})e^{-w_0 t}\Gamma(T) \tag{7b}$$

$$c_1(\mathbf{x}) = -e^{-w_0 t}\Gamma(T) \tag{8a}$$

$$c_2(\mathbf{x}) = -e^{-w_0 t}\Gamma(T) \tag{8b}$$

The theoretical result of sensitivity of a QP problem with respect to a parameter is provided by [23]. In the following, we use the superscript * on an expression to denote that the latter is evaluated at $\theta = 0$. Noting that (a) the functions of Equation (6) are twice differentiable, (b) the second-order sufficiency conditions hold at $\mathbf{x}^*$, (c) $\{\nabla_{\mathbf{x}} g_i^*\}, i = 1, \dots, 6$, $\{\nabla_{\mathbf{h}} g_j^*\}, j = 1, \dots, 2$ are linearly independent, and (d) $u_i^* > 0$ when $g_i^*(\mathbf{x}) = 0$ where u_i and w_i are the Lagrangian multipliers, then the following set of equations is satisfied at $(\mathbf{x}, \mathbf{u}, \mathbf{w}) = (\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)$, $\theta = 0$:

$$\begin{aligned} &\nabla_{\mathbf{x}} f - \sum_{i=1}^{6} u_i \nabla_{\mathbf{x}} g_i + \sum_{j=1}^{2} w_j (\nabla_{\mathbf{x}} h_j + \theta_j \nabla_{\mathbf{x}} c_j) = 0 \\ &u_i g_i(\mathbf{x}) = 0, \qquad i = 1, \dots, 6 \\ &h_j(\mathbf{x}) + \theta_j c_j(\mathbf{x}) = 0, \quad j = 1, 2 \end{aligned} \tag{9}$$

Tab. 1: Parameters of the step sequencer used for walking

Parameter	min	nom	max
$(\alpha_1,\, \alpha_2,\, \alpha_3)$	-	(1e3, 1, 1e6)	-
z_c	-	0.31	-
p_0	-	(-0.12, 0.10)	-
t	-	0.229	-
$\hat{\zeta}$	-	(-0.12, -0.07)	-
l	-0.3	0.1	0.3
w_{left}	-0.40	-0.25	-0.10
w_{right}	0.10	0.25	0.40
T	0.1	0.3	1.0

Let $F(\mathbf{x}, \theta)$ denote the function composed of the terms on the left-hand side of Eq. 9. Applying the implicit function theorem, we deduce that:

$$\frac{\partial(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)}{\partial\theta} = -J_{F^*}^{-1}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) J_F(\theta) \tag{10}$$

$$J_{F^*}(\mathbf{x}^*) = \begin{pmatrix} \text{diag}(\alpha_1, \alpha_1, \alpha_2, \alpha_3, \alpha_3) & -G^* & H^* \\ \text{diag}(u_i^*)(G^*)^\top & \text{diag}(g_i^*) & [0]_{6\times 2} \\ (H^*)^\top & [0]_{2\times 6} & [0]_{2\times 2} \end{pmatrix} \tag{11}$$

$$J_F(\theta) = \begin{pmatrix} C^* \text{diag}(w_j^*) \\ [0]_{6\times 2} \\ \text{diag}(c_j^*) \end{pmatrix} \tag{12}$$

where $G^* = \begin{pmatrix} \nabla_{\mathbf{x}} g_1^* & \cdots & \nabla_{\mathbf{x}} g_6^* \end{pmatrix}$, $H^* = \begin{pmatrix} \nabla_{\mathbf{x}} h_1^* & \nabla_{\mathbf{x}} h_2^* \end{pmatrix}$ and $C^* = \begin{pmatrix} \nabla_{\mathbf{x}} c_1^* & \nabla_{\mathbf{x}} c_2^* \end{pmatrix}$.

Eq. (10) allows us to numerically evaluate the sensitivity of the optimal solution of the QP problem with respect to perturbations on the DCM. For example, Fig. 6 presents the optimal solutions $p_{T,y}$ and $b_{T,y}$ in the space of Gaussian noise perturbations following $\mathcal{N}(0, 0.005)$ where no inequality constraints are active, using the input parameters of the problem from Table 1. We observe that these spaces form planes, and the slope of the plane in the space of $p_{T,y}$ is steeper than that of $b_{T,y}$. This result is expected because $\alpha_3 \gg \alpha_1$, which is also reflected in $(\frac{\partial p_{T,y}^*}{\partial \theta_y} = 5.18) > (\frac{\partial b_{T,y}^*}{\partial \theta_y} = 5.18e-3)$, by using Eq. (10). Thus, we understand that the robot will prioritize balancing before responding to the speed command.

3.1.3 Fixed-Horizon MPC Sequencer

To enable the WB MPC to predict a state trajectory throughout its prediction horizon (see Section 3.2), the step sequence has to extend at least to the end of the MPC's horizon. To generate a step sequence up to an appropriate time H_u, we propose to iteratively solve problem (2) and reuse the solution computed at time k as the initial condition of the problem solved at time $k+1$. This process can be carried out until a contact time exceeding the desired time horizon is obtained. The problem can be summarized as follows:

$$\begin{aligned} &\min_{\mathbf{x}_{k+1}} \quad f(\mathbf{x}_{k+1}, \mathbf{x}_k) \\ &\text{s.t.} \quad \begin{cases} g_i(\mathbf{x}_{k+1}, \mathbf{x}_k) \geq 0, & i = 1, \ldots, 6 \\ h_j(\mathbf{x}_{k+1}, \mathbf{x}_k) = 0, & j = 1, \ldots, 2 \end{cases} \end{aligned} \tag{13}$$

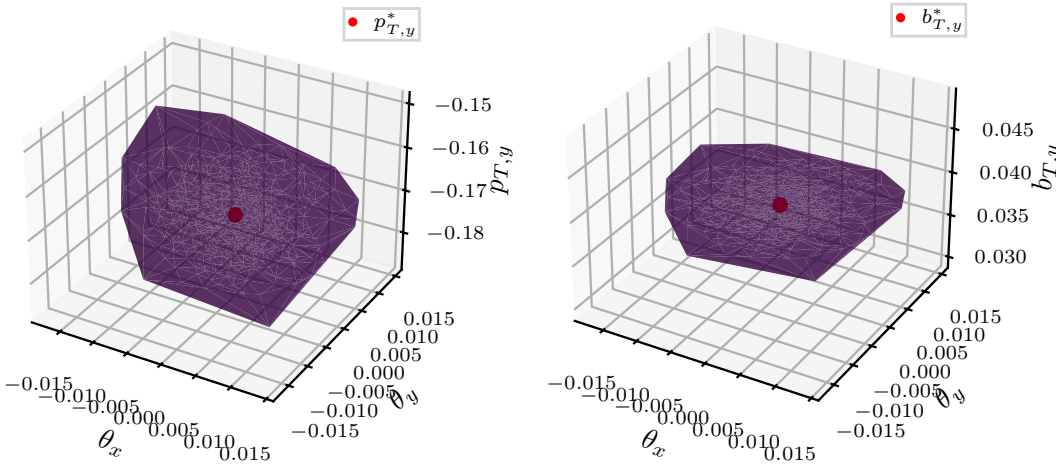


Fig. 6: Solution space example of $p_{T,y}$ (left) and $b_{T,y}$ (right) as a function of the DCM disturbance $\theta \sim \mathcal{N}(0, 0.005)$ generated with 1000 samples. The optimal solution is represented in red, and no inequality constraints is active.

$$
\begin{aligned}
f(\mathbf{x}_{k+1}, \mathbf{x}_k) = \alpha_1 &\left\| \mathbf{p}_{T_{k+1}} - \mathbf{p}_{T_k} - \begin{pmatrix} l_{\text{nom}} \\ w_{k,\text{nom}} \end{pmatrix} \right\|^2 \\
&+ \alpha_2 \left| \Gamma(T_{k+1}) - \Gamma(T_k + T_{\text{nom}}) \right|^2 \\
&+ \alpha_3 \left\| \mathbf{b}_{T_{k+1}} - \begin{pmatrix} b_{k,x,\text{nom}} \\ b_{k,y,\text{nom}} \end{pmatrix} \right\|^2
\end{aligned} \tag{14a}
$$

$$
\begin{aligned}
g_1(\mathbf{x}_{k+1}, \mathbf{x}_k) &= p_{T_{k+1},x} - p_{T_k,x} - l_{\min} && \text{(15a)} \\
g_2(\mathbf{x}_{k+1}, \mathbf{x}_k) &= l_{\max} - p_{T_{k+1},x} + p_{T_k,x} && \text{(15b)} \\
g_3(\mathbf{x}_{k+1}, \mathbf{x}_k) &= p_{T_{k+1},y} - p_{T_k,y} - w_{k,\min} && \text{(15c)} \\
g_4(\mathbf{x}_{k+1}, \mathbf{x}_k) &= w_{k,\max} - p_{T_{k+1},y} + p_{T_k,y} && \text{(15d)} \\
g_5(\mathbf{x}_{k+1}, \mathbf{x}_k) &= \Gamma(T_{k+1}) - \Gamma(T_k + T_{\min}) && \text{(15e)} \\
g_6(\mathbf{x}_{k+1}, \mathbf{x}_k) &= \Gamma(T_k + T_{\max}) - \Gamma(T_{k+1}) && \text{(15f)}
\end{aligned}
$$

$$
\begin{aligned}
h_1(\mathbf{x}_{k+1}, \mathbf{x}_k) &= p_{T_{k+1},x} + b_{T_{k+1},x} - p_{T_k,x} \\
&\quad - (\hat{\zeta}_{k,x} - p_{T_k,x}) e^{-w_0 T_k} \Gamma(T_{k+1}) && \text{(16a)} \\
h_2(\mathbf{x}_{k+1}, \mathbf{x}_k) &= p_{T_{k+1},y} + b_{T_{k+1},y} - p_{T_k,y} \\
&\quad - (\hat{\zeta}_{k,y} - p_{T_k,y}) e^{-w_0 T_k} \Gamma(T_{k+1}) && \text{(16b)}
\end{aligned}
$$

where $\mathbf{x}_{k+1}$ represents the next solution and $\mathbf{x}_k$ the previous solution. The parameter w_k varies depending on whether the next step is taken by the left foot or the right foot.

Algorithm 1 summarizes the MPC step sequencer:

Algorithm 1 Footsteps sequencer

$t \leftarrow t_{\text{mea}}$, $t_0 \leftarrow t_{\text{mea}}$, $\hat{\zeta}_k \leftarrow \zeta_{\text{mea}}$
$\mathbf{x}_k \leftarrow \begin{pmatrix} \mathbf{p}_{\text{ini}}^\top & \Gamma(t_{\text{mea}}) & [0]_{1\times 2} \end{pmatrix}^\top$
$S \leftarrow \emptyset$
repeat
  $\mathbf{x}_{k+1} \leftarrow$ Solve (13) using $\mathbf{x}_k$
  $t \leftarrow T_{k+1}$, $\hat{\zeta}_k \leftarrow \mathbf{p}_{T_{k+1}} + \mathbf{b}_{T_{k+1}}$
  $S \leftarrow S \cup \mathbf{x}_{k+1}$
  $\mathbf{x}_k \leftarrow \mathbf{x}_{k+1}$
until $t \geq t_0 + H_u$
Return S

where t_{mea} is the time at which the algorithm is called, ζ_{mea} is the measured DCM at $t = t_{\text{mea}}$, and $\mathbf{p}_{\text{ini}}$ is the position of the foot currently in support. S is the list containing the K steps of the generated sequence. The value of the nominal DCMs and the changes in w depending on whether the next step is taken by the left foot or the right foot are not detailed here for the sake of clarity. Fig. 7 illustrates an example of a step sequence generated using the parameters from Table 1 and a horizon H_u of 3 s for walking at a velocity of $V_x^* = 0.3\ m.s^{-1}$. The figure also shows that all hard constraints are satisfied, and the DCM remains non-divergent over time. This sequence can subsequently serve as a reference for WB MPC.

3.2 Whole-body MPC

3.2.1 Discretized Optimal Control Formulation

Given the step sequence generated by the sequencer, we now aim to compute a torque sequence to control the robot. To account for all dynamic effects, we employ a whole-body MPC, which iteratively solves the Optimal Control Problem (OCP) described as follows:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{i=0}^{N-1} l_i(\mathbf{x_i}, \mathbf{u_i}) + l_N(\mathbf{x_N}) \\ \text{s.t.} \quad & \mathbf{x}_0 = \mathbf{x}_{init} \\ & f_i(\mathbf{x_i}, \mathbf{u_i}) = \mathbf{x_{i+1}} \quad \forall i \in [0, N-1] \end{aligned} \tag{17}$$

where $\mathbf{x}$ and $\mathbf{u}$ represent the decision variables for the state and control inputs, respectively. N denotes the discrete horizon length (discretization of H_u), while $l_i(\mathbf{x}, \mathbf{u})$ corresponds to the running costs and $l_N(\mathbf{x_N})$ the terminal cost. The initial state, $\mathbf{x}_0$, is set to $\mathbf{x}_{\text{init}}$, and $f_i(\mathbf{x}_i, \mathbf{u}_i)$ represents the system dynamics from time i to $i+1$, including the contact dynamics constraints for both the left and right foot. The specific time instances i for these contacts are provided for each foot by the step sequencer.

3.2.2 Costs functions

The running cost $l_i(\mathbf{x_i}, \mathbf{u_i})$ is the sum of the following costs:

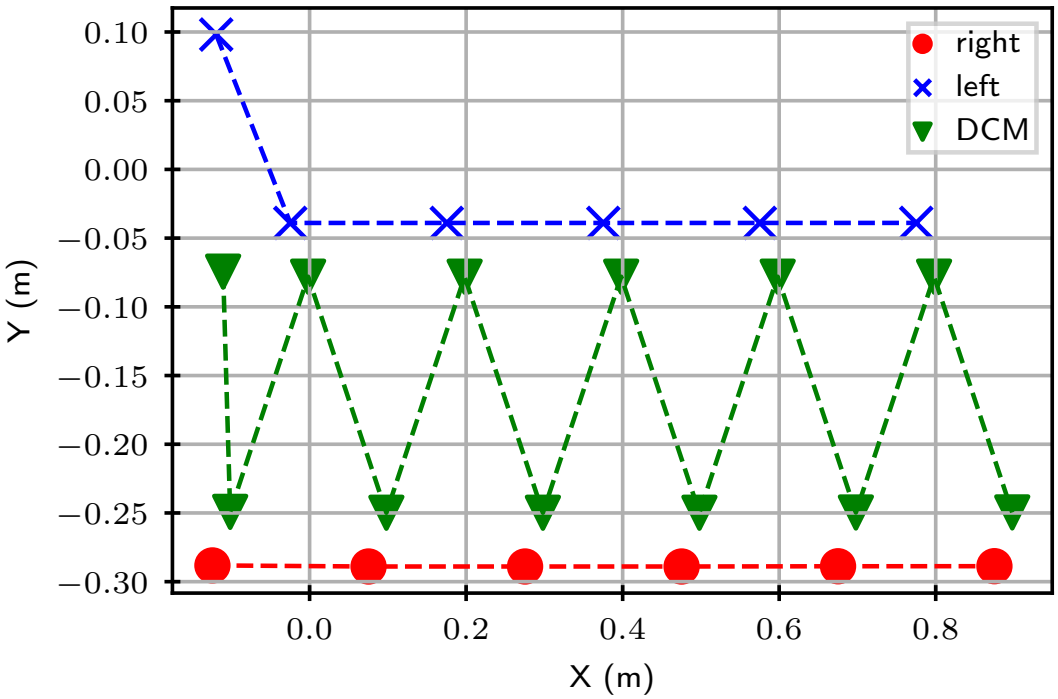


Fig. 7: Position of the feet and DCM in the Cartesian plane along a generated step sequence. The walking sequence was generated with $H_u = 3\ s$ using the input parameters from Table 1. The imposed speed is $V_x^* = \frac{l_{\text{nom}}}{T_{\text{nom}}} = 0.33 m.s^{-1}$. All constraints are satisfied, and the nominal speed is achieved.

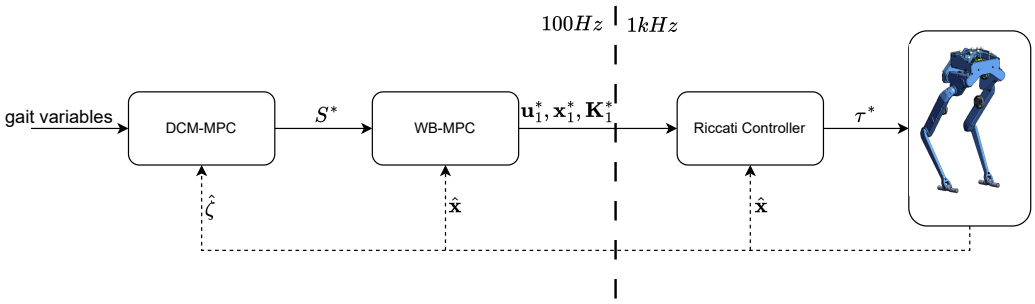


Fig. 8: Simplified architecture of Bolt's walking controller.

1. *Foot height tracking cost*: The cost of foot height tracking ensures that the robot lifts its foot appropriately. It is described by the following equation:

$$\ell^f_{track}(\mathbf{x}) = \left\| x^f_z - x^{f^*}_z \right\|^2_{w_{track}} \tag{18}$$

where f denotes the flying foot, $\mathbf{x}$ the state of the robot, x^f_z, $x^{f^*}_z \in \mathbb{R}$ respectively the measured and desired flying foot height, and $w_{track} \in \mathbb{R}$ a weighting hyperparameter.

The reference trajectory of the foot height is a polynomial function ensuring that the height and the velocity of the foot height are zero at both the initial and final instants, and that the foot reaches a predefined height at the midpoint of the duration, as described by:

$$x^{f^*}_z(t) = \frac{16H}{t_f^4}t^4 - \frac{32H}{t_f^3}t^3 + \frac{16H}{t_f^2}t^2 \tag{19}$$

where $t \in \mathbb{R}$ is a variable ranging from 0 to $t_f \in \mathbb{R}$, t_f is the total duration of the step, and $H \in \mathbb{R}$ is the maximum foot height at $\frac{t_f}{2}$.

Note that the foot contact positions calculated by the sequencer are not predefined, they naturally emerge from the solver used to address the WB MPC problem.

2. *Regularization costs*: A regularization cost around zero command indirectly limits the torques applied to the joints. It is implemented according to the following equation:

$$\ell_{\mathbf{u}}(\mathbf{u}) = \|\mathbf{u}\|^2_{w_{\mathbf{u}}} \tag{20}$$

where $w_{\mathbf{u}} \in \mathbb{R}$ is a weighting hyperparameter.

Additionally, a regularization cost of the state around a reference posture is added to ensure that the robot's posture tends towards it. The cost is described as:

$$\ell_X(\mathbf{x}) = \|(\mathbf{x} - \mathbf{x}^*)\|^2_{w_{\mathbf{x}}} \tag{21}$$

where $\mathbf{x}^* \in \mathbb{R}^{n_{\mathbf{x}}}$ (where $n_{\mathbf{x}}$ is the size of the robot state vector) is the desired reference posture, and $w_{\mathbf{x}} \in \mathbb{R}^{n_{\mathbf{x}} \times n_{\mathbf{x}}}$ is a weighting hyperparameter.

3. *Boundary cost*: A barrier cost is added to the control to ensure that it never exceeds the physical limits of the robot's actuators. The cost is given as follows:

$$\ell^f_{bu}(\mathbf{x}) = \|\min(\max(\mathbf{u}, \bar{\mathbf{u}}), \underline{\mathbf{u}})\|^2_{w_{bu}} \tag{22}$$

where $\underline{\mathbf{u}} \in \mathbb{R}^{n_{\mathbf{u}}}$ (where $n_{\mathbf{u}}$ is the size of the robot control vector) and $\bar{\mathbf{u}} \in \mathbb{R}^{n_{\mathbf{u}}}$ are respectively the lower and upper boundaries of the robot torque outputs. $w_{bu} \in \mathbb{R}$ is a weighting hyperparameter.

The cost $l_N(\mathbf{x_N})$ is the sum of the previous state-dependent costs along with an additional cost on the position of the center of mass:

$$\ell_{\text{CoM}}(\mathbf{x}) = \|\mathbf{c} - \mathbf{c}^*\|^2_{w_{\text{CoM}}} \tag{23}$$

where $\mathbf{c} \in \mathbb{R}^3$ is the position of the CoM, $\mathbf{c}^* \in \mathbb{R}^3$ is the reference position of the CoM, and $w_{\text{CoM}} \in \mathbb{R}$ is a weighting hyperparameter.

The reference for the CoM is calculated by solving Eq. (1a):

$$\mathbf{c}^* = (\hat{\mathbf{c}}_{K-1} - \hat{\zeta}_{K-1})e^{w_0(T_{K-1} - H_u)} + \hat{\zeta}_{K-1} \tag{24}$$

where the subscript $_{K-1}$ represents the index of the last step whose contact is before the horizon of the list S. This ensures that the robot's center of mass reaches a stable position while adhering to the desired setpoint.

3.2.3 Control pipeline

The full control structure consists of two parallel processes (see Fig. 8):

- The high-level process runs at 100 Hz and solves the DCM MPC and WB MPC using OSQP [28] and Feasibility-Driven Differential Dynamic Programming (FDDP) algorithm from Crocoddyl [22], respectively. A demonstration of Differential Dynamic Programming (DDP), similar to FDDP, is given in the appendix C. The WB MPC uses a 10 ms time discretization interval to balance problem-solving time and numerical integration quality. The high-level frequency is set to use the solution from node 1 (after 10 ms) to account for computational delay.
- The optimization process yields a by-product that is leveraged for low-level control of the robot at a frequency of 1 kHz. We designate these as Riccati gains throughout the remainder of this paper. The control law, derived from [29], is given by:

$$\tau = \mathbf{u}_1 + K_1(\hat{\mathbf{x}} - \mathbf{x_1}) \tag{25}$$

  where $\mathbf{u}_1 \in \mathbb{R}^{n_\mathbf{u}}$ and $\mathbf{x}_1 \in \mathbb{R}^{n_\mathbf{x}}$ are the optimal control input and state computed by the solver, respectively. $\hat{\mathbf{x}} \in \mathbb{R}^{n_\mathbf{x}}$ is the robot's measured state at 1 kHz. Finally, the matrix $K_1 \in \mathbb{R}^{6\times(n_\mathbf{x}-1)}$ is the Riccati gain matrix at node 1 given by:

$$K_1 \triangleq \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_1} \tag{26}$$

This setup ensures high-frequency feedback and robust real-time control.

3.3 Simulation Results

3.3.1 Setup

The objective of these experiments in simulation is to evaluate the robot's ability to follow a given velocity command, demonstrate robustness against external perturbations, and adapt its gait dynamically. The simulation is conducted in PyBullet on an Apple Mac M3 MAX CPU clocked at 4.05 GHz with 16 cores. A simplified URDF model of the Bolt robot is utilized[5], with details in the associated paper [25]. Initial conditions are identical for all simulations: Bolt starts with both feet on the ground and in the configuration specified by the SRDF. Three scenarios are designed:

1. *Walking with and without disturbance*: Bolt follows a walking command while experiencing occasional external perturbations.
2. *Cluttered terrain walking*: Bolt walks on a cluttered terrain.
3. *Velocity reference transition*: Bolt transitions from a velocity command to another.

3.3.2 Nominal walking with and without perturbations

We propose to compare the walking motion of Bolt with a target speed command $v_X^* = 3.3\,m.s^{-1}$ with and without perturbations. Fig. 9 shows the y-axis positions of the CoM, DCM and feet positions of bolt over time. The top plot depicts unperturbed walking over a two-second interval, while the bottom plot shows walking with a perturbation of 6.3 N applied at the base for 0.1 s

[5] `https://github.com/Gepetto/example-robot-data/tree/master/robots/bolt_description`

(equivalent to 0.63 N.s) at $t = 4$ s. We observe that Bolt is capable of rejecting the perturbation by adjusting the step sequence and subsequently resumes nominal walking.

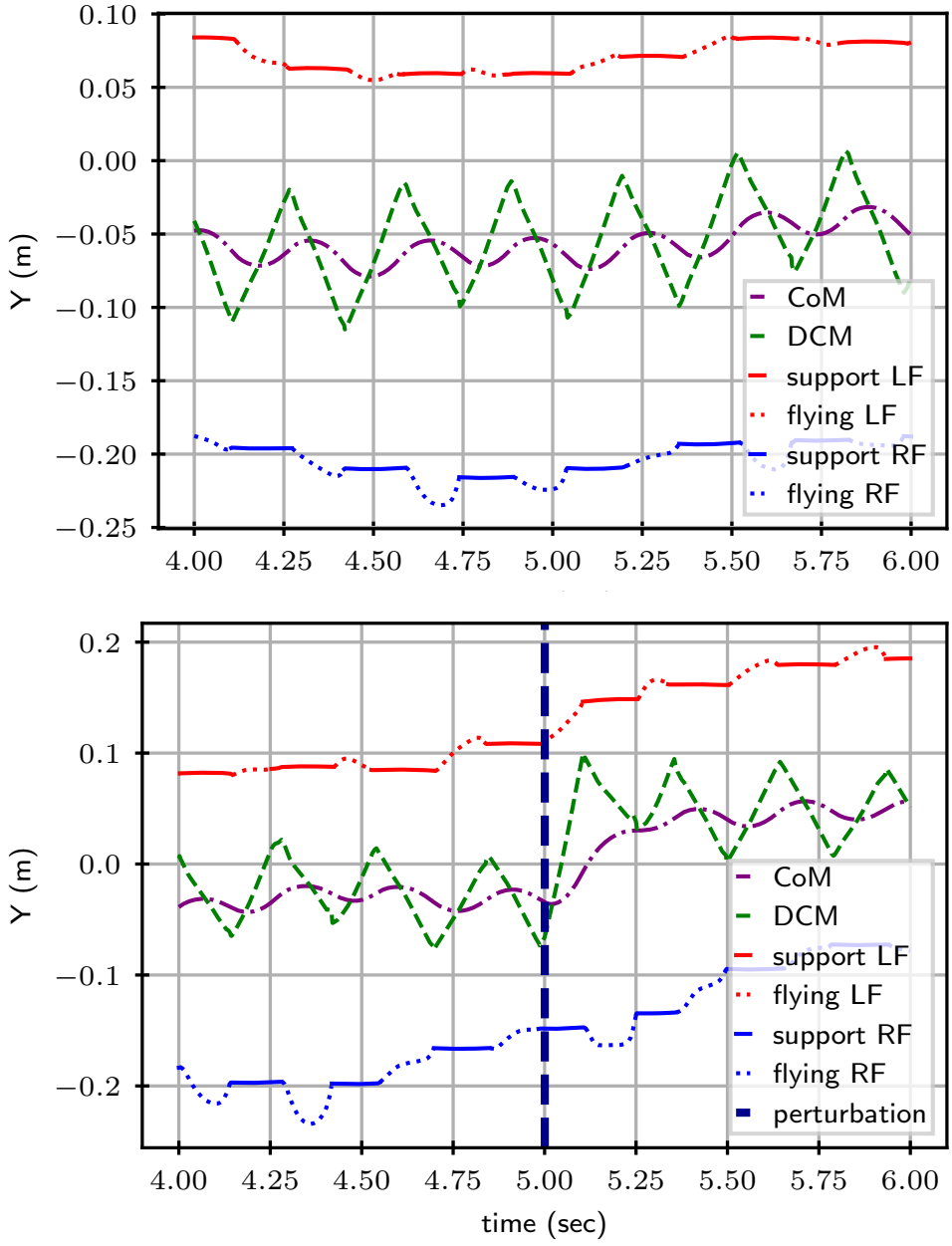


Fig. 9: Comparison of the CoM (purple), the DCM (green), and the feet positions (red for left, blue for right) along the y-axis over time during walking at $V_x^* = 0.33\, m.s^{-1}$. Top: Unperturbed walking. Bottom: Walking perturbed at $t = 5$ s by an applied force of $F = 6.3$ N along the y-axis at the robot's base for 0.1 s.

Next, we analyze the base velocity of Bolt. Fig. 10 shows the base velocities in the x and y directions over time, both without and with the same perturbation as before. The top plot presents the unperturbed velocities, whereas the bottom plot shows the velocities under perturbation. We observe that the accuracy of x-axis velocity tracking decreases at the onset of the perturbation and returns to the target velocity once the perturbation is rejected. This behavior is expected because, as explained in Section 3.1.2, in case of a DCM perturbation, the solver prioritizes maintaining Bolt's balance at the cost of velocity tracking. Additionally, we note on the top plot an average error $\epsilon_{\bar{\mathbf{V}}} = \left|\bar{\mathbf{V}} - \mathbf{V}^*\right| = \begin{pmatrix}5.2 & 1.1\end{pmatrix}^\top \, mm.s^{-1}$ in the velocity tracking. It is mainly due to discrepancies between the LIPM and the simulator model, particularly the fact that angular momentum is neglected in the LIPM. Solutions exploring 3D DCM-based methods [15] address this issue, however such an extension is beyond the scope of our work.

3.3.3 Nominal walking on cluttered terrain

We now demonstrate the robustness of the controller on Bolt against slips and rough terrain (see Fig. 11) with a reference speed $V_x^* = 0.3\, m.s^{-1}$.

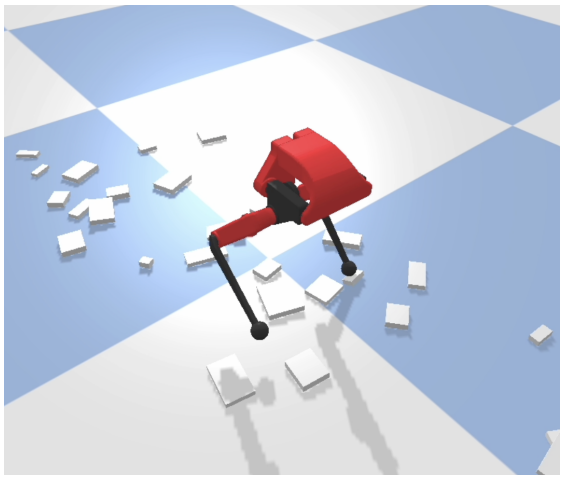


Fig. 11: Bipedal robot bolt walking in PyBullet using whole- body MPC on cluttered terrain.

The perturbations involve the generation of rectangular parallelepipeds with side lengths ranging from 1 cm to 5 cm and heights ranging from 5 mm to 8 mm, placed on the robot's path, with an average density of 10 obstacles per square meter. Fig. 12 shows the y axis position of the CoM, the DCM, and the foot positions over time, highlighting an instance of left foot slippage. We observe that the controller adapts the step sequence to compensate for slips and reject the resulting perturbations.

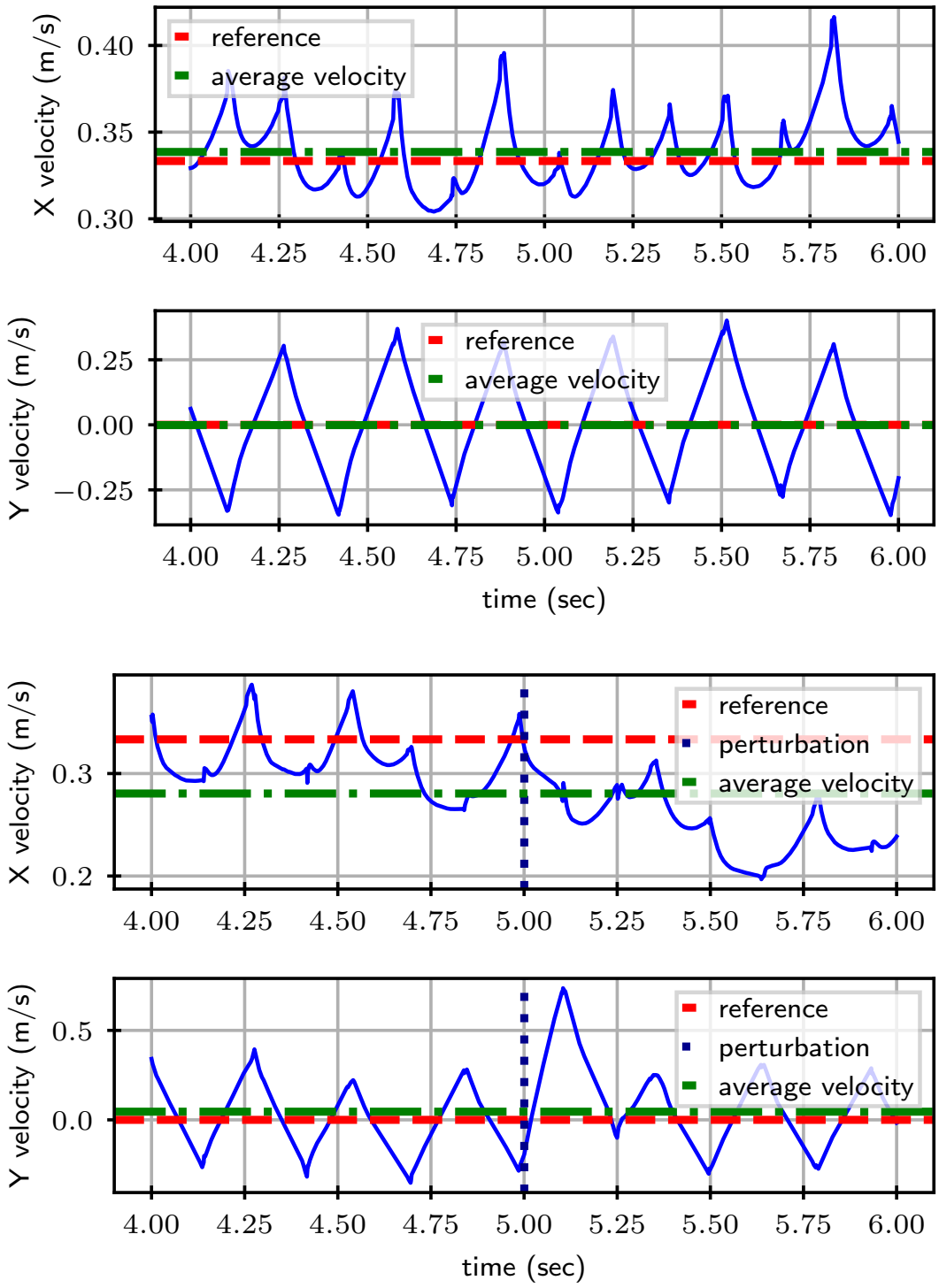


Fig. 10: Comparison of the base velocity (blue) and the average velocity (green) along the x and y axes over time when walking at $V_x^* = 0.33\,m.s^{-1}$ (shown in red). Top: Unperturbed walking. Bottom: Walking perturbed at $t = 5\,\mathrm{s}$ by an applied force of $F = 6.3\,\mathrm{N}$ along the y-axis at the robot's base for 0.1 s.

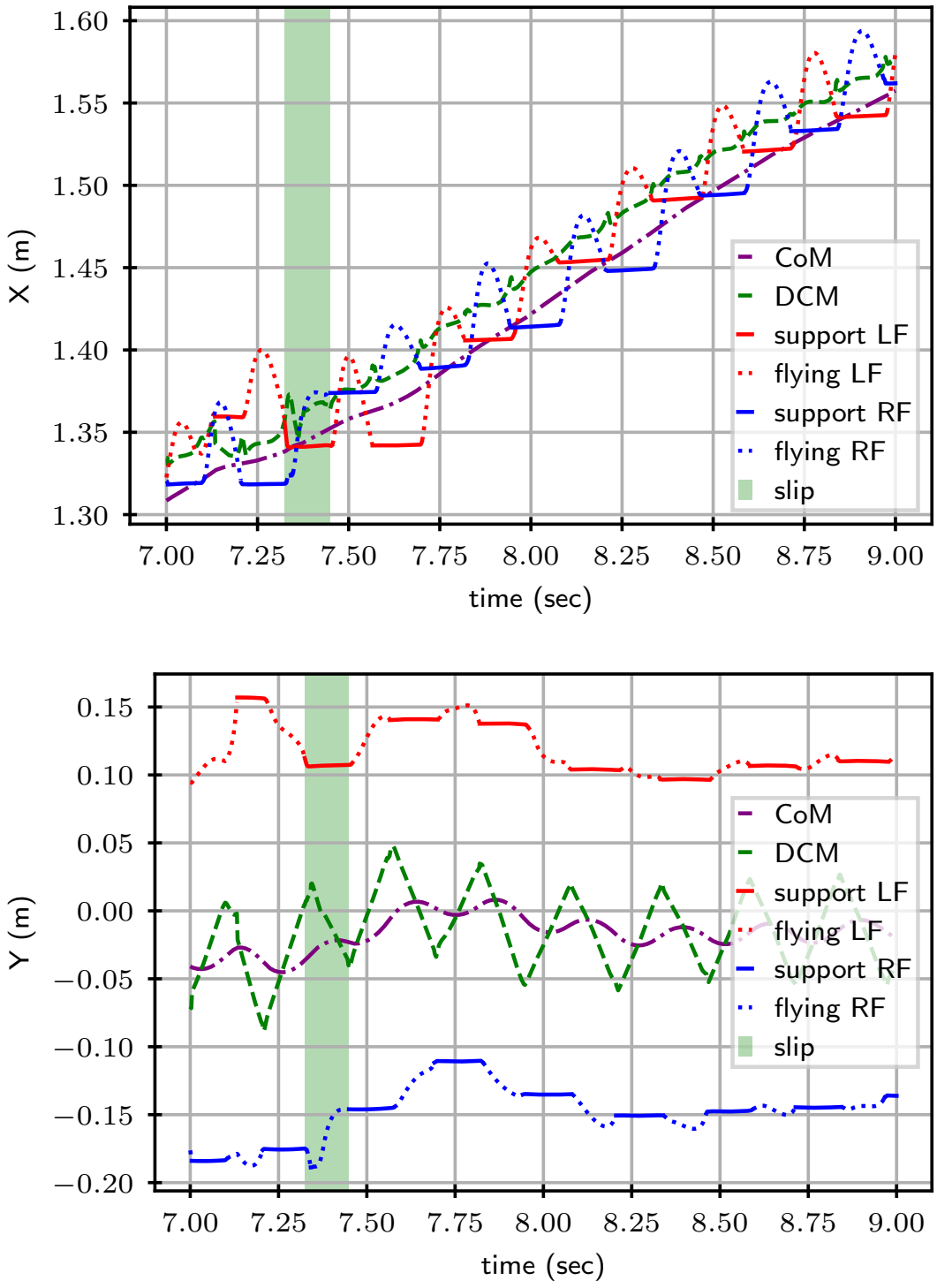


Fig. 12: CoM (purple), the DCM (green), and the feet positions (red for left, blue for right) along the y-axis over time when walking at $V_x^* = 0.33\,m.s^{-1}$. The left foot is slipping in the green area.

3.3.4 Transition from standing to walking

Fig. 13 shows the robot's base velocity along the x axis over time. The command initially sets the velocity to 0 $m.s^{-1}$ (step in place), transitioning to 0.3 $m.s^{-1}$ at $t = 5$s. We measure a rise time of 4.35s for the robot to reach its final velocity value. This delay can be attributed to a significant increase in angular moments caused by the velocity change, leading to a larger discrepancy between the LIPM and the simulator model. This difference can be modeled as a perturbation on the DCM. As explained in section 3.1.2, foot placements are more sensitive to perturbations than robot balance. Therefore, while striving to maintain balance, the feet positions deviate from the optimal position, resulting in a slower achievement of the desired velocity.

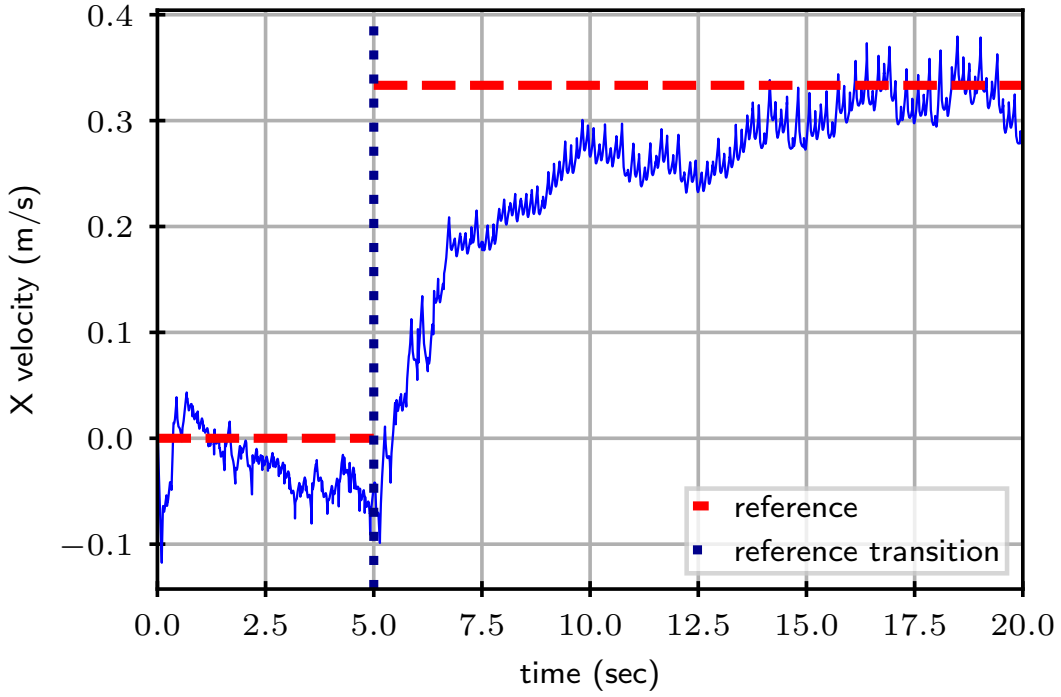


Fig. 13: Base velocity (blue) and the average velocity (green) along x-axis over time when walking at $V_x^* = 0.0\, m.s^{-1}$ before $t = 5s$ and $V_x^* = 0.33\, m.s^{-1}$ after $t = 5s$ (shown in red).

3.4 Discussion and conclusion

In this study, we demonstrated that successful control of a highly unstable bipedal robot can be achieved using a footstep sequencer and a WB MPC operating at 100 Hz. Our approach eliminates the need for explicit footstep trajectories, as they naturally emerge when solving the WB MPC. Our results show that the robot exhibits significant capabilities in executing velocity transitions, recovering from perturbations, and managing foot slippage. Sensitivity analysis provides additional insights, aligning with previous research [18, 19] while offering a deeper theoretical understanding. However, we observed that the desired speed was not always attained and that velocity transitions were slow, which can limit the robot's performance in scenarios

requiring rapid speed changes. This outcome was anticipated due to the assumptions imposed by the LIPM.

In the short term, we plan to deploy and validate this controller on the physical Bolt robot, something we have not yet managed to achieve for various reasons (complex initial conditions, measurement noise, defective robot, etc...).. For future work, we plan to leverage the sensitivity analysis computed in this study as a foundational work to estimate the sensitivity of the whole control structure (DCM MPC + WB MPC). We would additionally need to compute the sensitivity WB MPC with respect to the contact timings calculated by the step sequencer. This study could allow to provide more depth in the theoretical interpretation of the performance of our control structure as well as offering robustness guarantees.

As we were unable to control Bolt using the WB MPC, we propose a more robust control method, RL, in the following section.

4 Reinforcement learning based control

4.1 CaT: Constraints as Terminations

4.1.1 Theory

In RL, we often aim to find a policy π that maximizes the expected cumulative reward over time. The first equation describes the standard RL objective:

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi,\mathcal{T}} \left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t\right) \right] \tag{27}$$

where π is the policy. $\mathbb{E}_{\tau\sim\pi,\mathcal{T}}$ denotes the expectation over trajectories τ generated by policy π and transition dynamics $\mathcal{T}$. γ is the discount factor, which ensures that future rewards are weighted less than immediate rewards and $r(s_t, a_t)$ is the reward received at time step t when the agent is in state s_t and takes action a_t.

Chane-Sane et al. [24] propose to modify the standard RL objective to include constraints as terminations:

$$\max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \left(\prod_{t'=0}^{t} \gamma\left(1-\delta\left(s_{t'}, a_{t'}\right)\right) \right) r\left(s_t, a_t\right) \right] \tag{28}$$

where the product term $\prod_{t'=0}^{t} \gamma\left(1-\delta\left(s_{t'}, a_{t'}\right)\right)$ adjusts the discount factor dynamically based on the random variable $\delta(s_{t'}, a_{t'})$. $\delta(s_{t'}, a_{t'})$ is a random variable function that represents whether a constraint is violated. If a constraint is violated, δ changes, which decreases the product term and effectively reduces the contribution of future rewards.

The δ random variables is defined as follows:

$$\delta = \max_{i\in I} p_i^{\max} \operatorname{clip}\left(\frac{c_i^+}{c_i^{\max}}, 0, 1 \right) \tag{29}$$

where i indexes the set of constraints I and $p_i^{\max}$ is the maximum allowable probability of violating constraint i. c_i^+ represents the current value of constraint i and $c_i^{\max}$ is a moving threshold for constraint i. $\operatorname{clip}(x, 0, 1)$ restricts the value of x to be between 0 and 1.

The following equation updates the maximum constraint value $c_i^{\max}$:

$$c_i^{\max} \leftarrow \tau^c c_i^{\max} + \left(1-\tau^c\right) \max_{(s,a)\in \text{ batch}} c_i^+(s,a) \tag{30}$$

where τ^c is a decay rate and $\max_{(s,a)\in\text{batch}} c_i^+(s,a)$ is the maximum value of c_i^+ observed in the current batch of state-action pairs.

4.1.2 Rewards

The reward function guides the agent towards desirable behaviors. The following equation defines the reward function used on Bolt:

$$r = e^{-\frac{\left\| v_{xy}^{\text{des}} - v_{xy} \right\|_2^2}{0.25}} + \frac{1}{2} e^{-\frac{\left| \omega_z^{\text{des}} - \omega_z \right|^2}{0.25}} \tag{31}$$

where v_{xy}^{des} is the desired velocity in the xy-plane and v_{xy} is the actual velocity in the xy-plane. ω_z^{des} is the desired angular velocity around the z-axis and ω_z is the actual angular velocity around the z-axis. The terms $e^{-\frac{\left\|v_{xy}^{\text{des}}-v_{xy}\right\|_2^2}{0.25}}$ and $\frac{1}{2}e^{-\frac{\left|\omega_z^{\text{des}}-\omega_z\right|^2}{0.25}}$ ensure that the reward decreases exponentially as the actual velocities deviate from the desired velocities, promoting behavior that achieves the desired velocities.

4.1.3 Constraints

The table 2 summarizes the constraints applied during the training of the bipedal robot Bolt. These constraints are critical for ensuring both the safety and performance of the robot. They are categorized into three distinct groups:

1. *Hard Constraints for Safety:* These constraints are essential and non-negotiable to ensure the robot's safety. They include:
   - *Knee or Base Collision:* The constraint is represented by $c_{\text{knee/base contact}} = \mathbb{1}_{\text{knee/base contact}}$, where $\mathbb{1}_{\text{knee/base contact}}$ is an indicator function that activates if a collision between the knee or base occurs.
   - *Upsidedown:* Denoted as $c_{\text{upsidedown}} = \mathbb{1}_{\text{upsidedown}}$, this constraint is an indicator function that is triggered when the robot is in an upside-down position.
   - *Foot Contact Force:* This is given by $c_{\text{foot contact }_j} = \left\|f^{\text{foot }_j}\right\|_2 - f^{\text{lim}}$, where $\left\|f^{\text{foot }_j}\right\|_2$ represents the Euclidean norm of the contact force at foot j, and f^{lim} is the maximum allowable force.
2. *Soft Constraints for Safety:* These constraints are flexible and are used to maintain safety within permissible limits. They include:
   - *Torque Limits:* Represented by $c_{\text{torque }_k} = |\tau_k| - \tau^{\text{lim}}$, where $|\tau_k|$ is the magnitude of the torque on joint k, and τ^{lim} is the torque limit.
   - *Joint Velocity Limits:* Given by $c_{\text{joint velocity }_k} = |\dot{q}_k| - \dot{q}^{\text{lim}}$, where $|\dot{q}_k|$ denotes the velocity of joint k, and $\dot{q}^{\text{lim}}$ is the maximum allowable velocity.
   - *Joint Acceleration Limits:* Expressed as $c_{\text{joint acceleration }_k} = |\ddot{q}_k| - \ddot{q}^{\text{lim}}$, where $|\ddot{q}_k|$ is the acceleration of joint k, and $\ddot{q}^{\text{lim}}$ is the acceleration limit.
   - *Action Rate Limits:* Defined by $c_{\text{action rate }_k} = \frac{\left|\Delta q_{t,k}^{\text{des}} - \Delta q_{t-1,k}^{\text{des}}\right|}{dt} - \dot{q}^{\text{des lim}}$, where $\frac{\left|\Delta q_{t,k}^{\text{des}} - \Delta q_{t-1,k}^{\text{des}}\right|}{dt}$ denotes the rate of change of desired joint angles, and $\dot{q}^{\text{des lim}}$ is the maximum allowable rate.
3. *Soft Constraints for Style:* These constraints pertain to the robot's performance style and aesthetics. They include:
   - *Base Orientation:* Described by $c_{\text{ori}} = \| \text{ base ori }_{\text{xy}}\|_2 - \text{ base }^{\text{lim}}$, where $\| \text{ base ori }_{\text{xy}}\|_2$ is the Euclidean norm of the base's orientation in the xy-plane, and $\text{ base }^{\text{lim}}$ is the orientation limit.
   - *Hip Orientation:* Given by $c_{\text{hip }_j} = | \text{ hip ori }_j | - \text{ hip }^{\text{lim}}$, where $| \text{ hip ori }_j |$ represents the magnitude of the hip orientation for joint j, and $\text{ hip }^{\text{lim}}$ is the orientation limit.
   - *Foot Air Time:* Expressed as $c_{\text{air time }_j} = t_{\text{air time}}^{\text{des}} - t_{\text{air time }_j}$, where $t_{\text{air time}}^{\text{des}}$ is the desired time the foot is in the air, and $t_{\text{air time }_j}$ is the actual air time for foot j.

Tab. 2: Constraints used to train the bipedal robot Bolt

Hard constraints for safety	
Knee or base collision	$c_{\text{knee/base contact}} = \mathbb{1}_{\text{knee/base contact}}$
Upsidedown	$c_{\text{upsidedown}} = \mathbb{1}_{\text{upsidedown}}$
Foot contact force	$c_{\text{foot contact }_j} = \left\| f^{\text{foot }_j} \right\|_2 - f^{\text{lim}}$
Soft constraints for safety	
Torque limits	$c_{\text{torque }_k} = \lvert\tau_k\rvert - \tau^{\text{lim}}$
Joint velocity limits	$c_{\text{joint velocity }_k} = \lvert\dot{q}_k\rvert - \dot{q}^{\text{lim}}$
Joint acceleration limits	$c_{\text{joint acceleration }_k} = \lvert\ddot{q}_k\rvert - \ddot{q}^{\text{lim}}$
Action rate limits	$c_{\text{action rate }_k} = \frac{\lvert\Delta q^{\text{des}}_{t,k} - \Delta q^{\text{des}}_{t-1,k}\rvert}{dt} - \dot{q}^{\text{des lim}}$
Soft constraints for style	
Base orientation	$c_{\text{ori}} = \lVert \text{ base ori }_{\text{xy}} \rVert_2 - \text{ base }^{\text{lim}}$
Hip orientation	$c_{\text{hip }_j} = \lvert \text{ hip ori }_j \rvert - \text{ hip }^{\text{lim}}$
Foot air time	$c_{\text{air time }_j} = t^{\text{des}}_{\text{air time}} - t_{\text{air time }_j}$
Number of foot contacts	$c_{\text{n foot contacts}} = \left\lvert n_{\text{foot contact}} - n^{\text{des}}_{\text{foot contact}} \right\rvert$

- *Number of Foot Contacts:* Defined by $c_{\text{n foot contacts}} = \left\lvert n_{\text{foot contact}} - n^{\text{des}}_{\text{foot contact}} \right\rvert$, where $n_{\text{foot contact}}$ is the actual number of foot contacts and $n^{\text{des}}_{\text{foot contact}}$ is the desired number of foot contacts.

4.1.4 Domain randomization

Domain randomization is a technique in RL that aims to enhance the robustness and generalization capabilities of RL agents by exposing them to a diverse range of environmental conditions during training. The table 3 outlines various sources of uncertainty that are randomized in the training process of Bolt. These sources of uncertainty include:

- *Robot Initial State*: Variations in the initial position and velocity of joints, as well as base pose, are essential for simulating diverse starting conditions.
- *Robot Sensors*: Noise introduced in joint position, velocity, and sensor measurements reflects the imperfections and variations in real-world sensors.
- *Robot Dynamics*: Inertial parameters such as mass, matrix, and CoM are varied to account for differences in robot dynamics.
- *Terrain Dynamics*: Changes in friction represent different terrain conditions the robot might operate on.
- *Commands*: Variability in commanded velocities and angular velocities to simulate different control inputs.
- *External Perturbations*: Randomization of push intervals helps to prepare the robot for unexpected disturbances.

Tab. 3: Domain randomization of Bolt training

Sources of uncertainty	Parameter	Range	Unit
Robot initial state	Initial joint position noise	$[-1.03, 1.03]\times$ default	rad
	Initial joint velocity noise	$[-1.03, 1.03]\times$ default	rad/s
	Initial base x noise	$[-0.05, 0.05]$	m
	Initial base y noise	$[-0.05, 0.05]$	m
	Initial base pitch noise	$[0.3, 0.3]$	rad
	Initial base roll noise	$[0.3, 0.3]$	rad
	Initial base yaw noise	$[-\pi, \pi]$	rad
Robot sensors	Joint position static bias noise	$[-0.05, 0.05]$	rad
	Joint position noise	$[-0.03, 0.03]$	rad
	Joint velocity noise	$[-0.03, 0.03]$	rad/s
	Base angular velocity noise	$[-0.01, 0.01]$	rad/s
	Measured gravity noise	$[-0.01, 0.01]$	m/s^2
Robot Dynamics	Inertial mass links noise	$[0.8, 1.2]\times$ default	kg
	Inertial matrix links noise	$[0.8, 1.2]\times$ default	$kg.m^2$
	Inertial CoM links noise	$[0.8, 1.2]\times$ default	m
Terrain Dynamics	Friction Range	0.25 - 1.5	-
Commands	x-linear velocity	$[-0.5, 0.5]$	m/s
	y-linear velocity	$[-0.3, 0.3]$	m/s
	Yaw angular velocity	$[-0.2, 0.2]$	rad/s
External perturbations	Push Interval	8	s

4.2 Simulation Results

In this section, we present the results of a training session. Fig. 14, 15, and 16 respectively illustrate the violation rate of hard safety constraints, soft safety constraints, and soft style constraints. We observe that all safety constraints are close to zero, which is a necessary result. We notice that most style constraints are respected, except for the airtime constraint. This could be due to insufficient training or conflicting constraints (for example, relaxing the base orientation constraint decreases airtime for the same number of epochs).

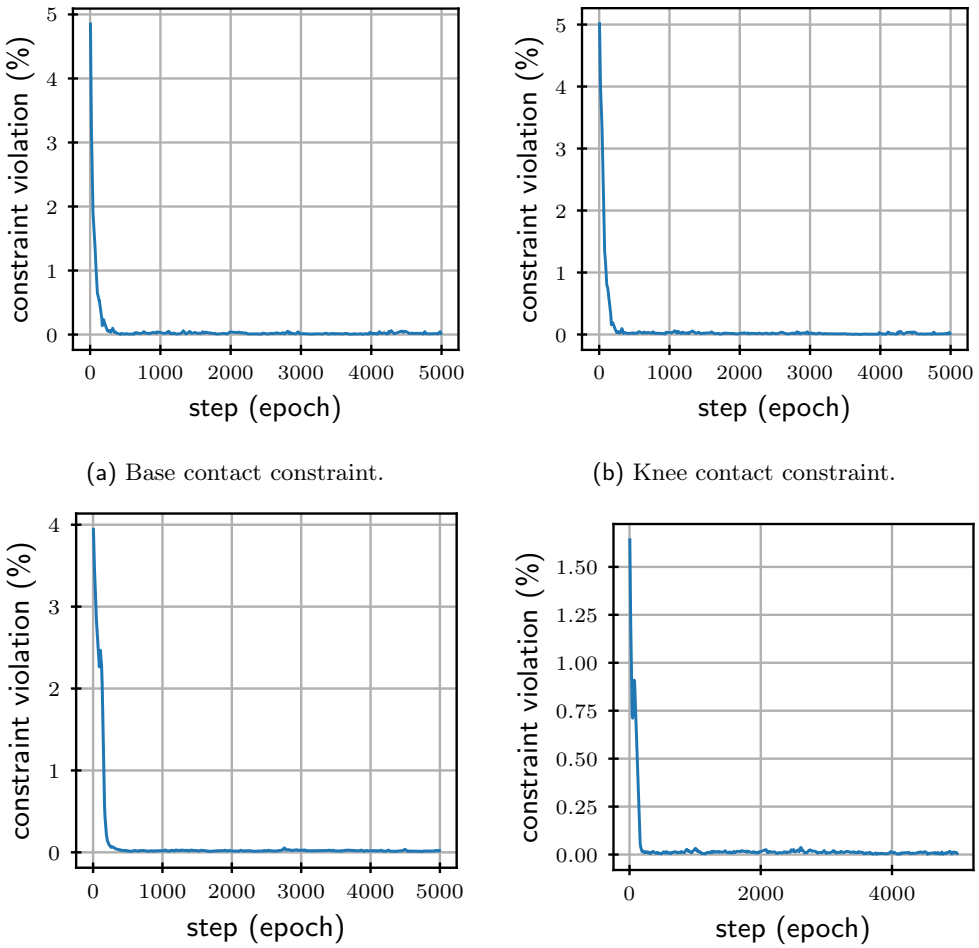
Fig. 17 presents a simulated walk of Bolt at a speed of 0.5 m/s, executed by applying the policy learned in the previously discussed training session. The walk satisfies all hard and soft safety constraints as well as the soft style constraints (except for the air time, which is not met in 14.7% of cases).

With the simulation fulfilling all necessary constraints and having implemented domain randomization, we can proceed to the sim-to-real phase.

4.3 Experimental Results

We now present the experiment detailed in Section 4.2. The policy deployed on Bolt enabled the robot to maintain balance by taking steps in place (see Fig. 18).

However, in order to maintain the horizontal base velocity close to zero, we had to command a forward velocity of $v_x = 0.3$ m/s. Without this input, the robot tends to move backward and eventually falls. We hypothesize that this issue may arise from several factors, primarily related to modeling inaccuracies, such as:



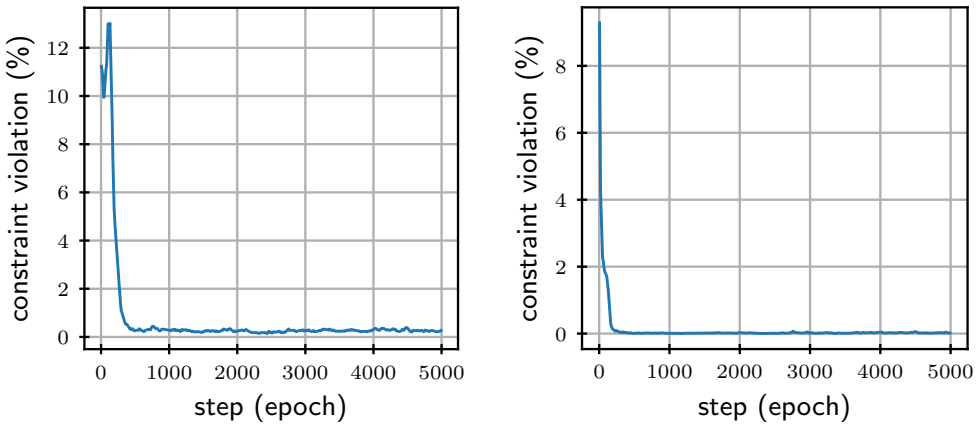
(a) Base contact constraint.

(b) Knee contact constraint.

(c) Foot contact constraint.

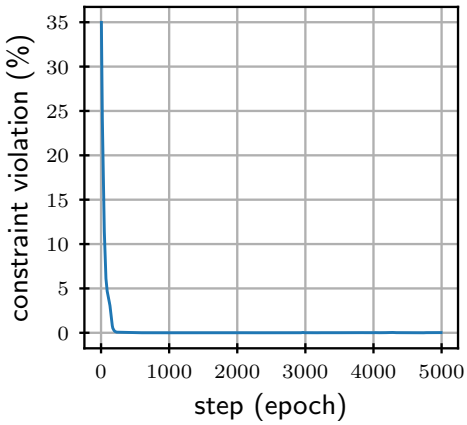
(d) Upsidedown constraint.

Fig. 14: Hard constraints for safety violation. Constraints at the end of training are all very close to zero.

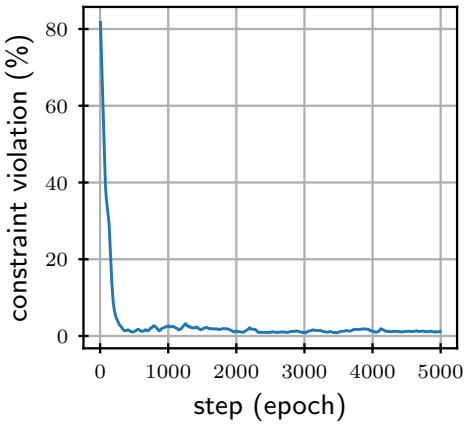


(a) Torque constraint.

(b) Joint velocity constraint.

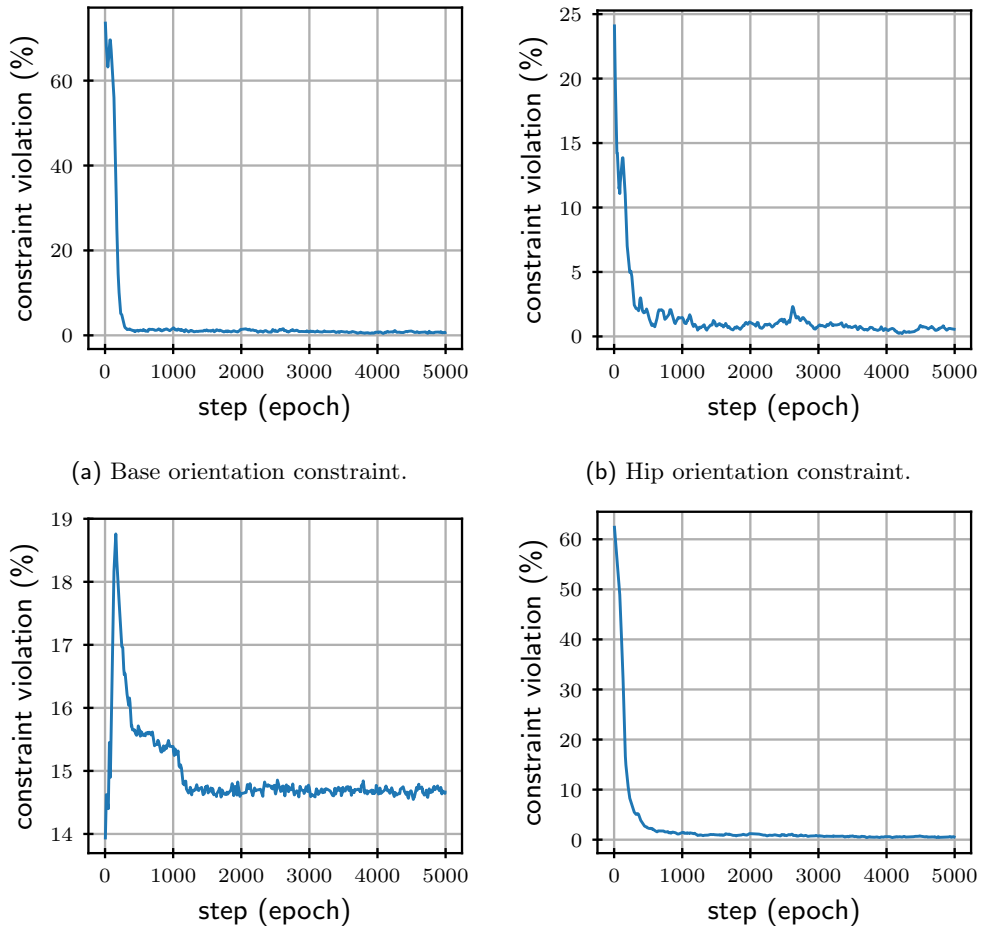


(c) Joint acceleration constraint.



(d) Action rate constraint.

Fig. 15: Soft constraints for safety violation. Constraints at the end of training are all very close to zero.



(a) Base orientation constraint.

(b) Hip orientation constraint.

(c) Foot air time constraint.

(d) Number of foot contacts constraint.

Fig. 16: Soft constraints for style violation. Constraints at the end of training are all very close to zero, except for air time, which stagnates at around 14.7%.

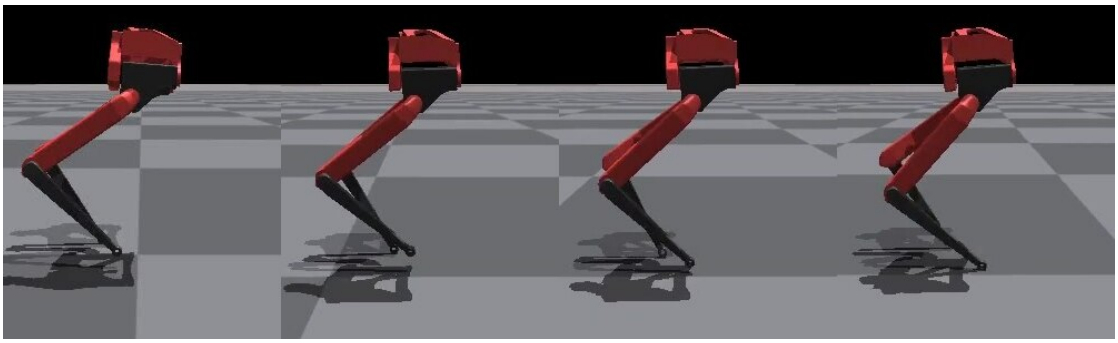


Fig. 17: Bolt walking with CaT at a speed of 0.5 m/s. Each image is separated by a time step of 0.5 s.

Fig. 18: Real Bolt walking with CaT at a speed of 0.0 m/s. Each image is separated by a time step of 1.0 s.

- The actual CoM of the robot's base does not align with the model's CoM.
- The inertia matrices of the robot's links (particularly the base) differ significantly from those in the model.

Through a protocol not detailed here, we found that the center of mass of the base link is nearly identical between the model and the real robot. Regarding the inertia matrices, we first developed a tool (See Sec. 5.3) to visualize them and identify discrepancies. Future work will focus on investigating whether incorrect inertia matrices are indeed the cause of the issue.

5 Integrated Robot Development

5.1 ODRI: Open Dynamic Robot Initiative

ODRI is a collaborative project aimed at developing dynamic and interactive robots using an open-source approach. The main objectives are as follows:

- *Open-Source Development* : Providing plans, software, and performance data to the community.
- *Accessibility* : Making dynamic robotics technology more accessible to universities, research labs, and enthusiasts.
- *Modularity and Scalability* : Designing modular robots that can be easily adapted and improved for various applications.
- *Cost Reduction* : Utilizing affordable components while maintaining high performance.

Solo8 (See Fig. 19[6]) is a notable example of a robot developed under the ODRI framework. This quadrupedal robot is designed to exemplify the principles of the initiative.

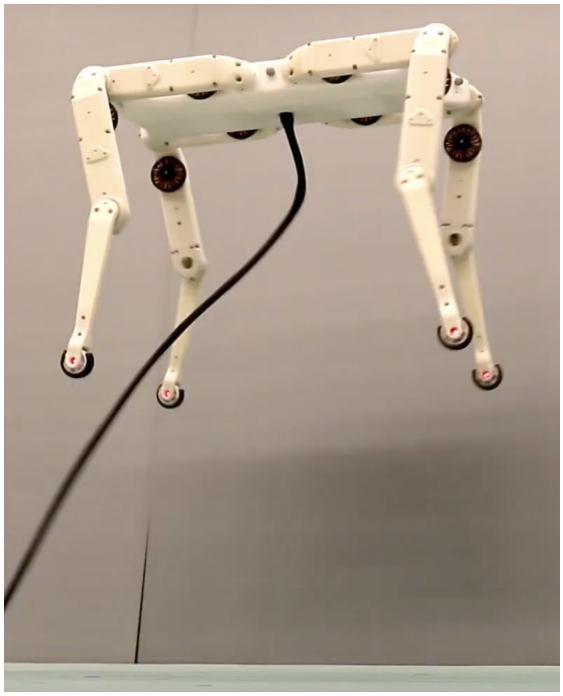


Fig. 19: Solo8 jumping.

In summary, ODRI and robots like Solo8 and Bolt play an important role in the advancement of dynamic robotics by promoting international collaboration, innovation, and education through an open-source approach.

[6] image from this https://www.youtube.com/watch?v=Kd8lOkEBcnc.

Fig. 20: Brushless motor used on Bolt.

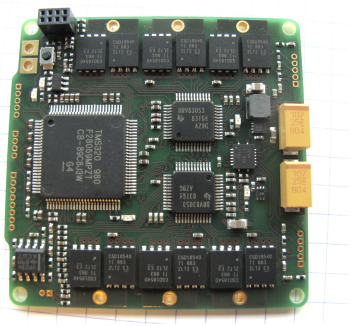


Fig. 21: Microdriver used on Bolt.

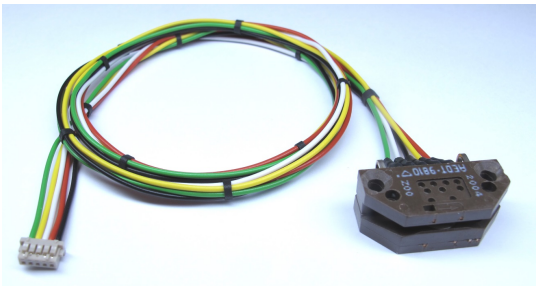


Fig. 22: Relative encoder used on Bolt.

Fig. 23: IMU used on Bolt.

5.2 Bolt's Architecture

5.2.1 General Overview

Bolt is a bipedal robot developed by ODRI. It features three active degrees of freedom per leg, which introduces significant challenges in terms of stability and control. This section provides an in-depth look at Bolt's architecture, highlighting its key components and their functionalities.

5.2.2 Electronic Components

Each actuator of the robot is driven by a brushless motor (See Fig. 20). The robot's electronic architecture is centered around three microdrivers (See Fig. 21), with each microdriver capable of controlling two motors. This configuration allows for precise motor control.

In addition to the motor controllers, each motor is paired with relative encoders (See Fig. 22) that measure the position and angular velocity of the actuators. This feedback is necessary for the accurate control of the robot's movements. The robot is also equipped with an IMU (See Fig. 23), which provides real-time data on the angular velocity and linear acceleration of the robot's base.

The integration of these components is managed by a central interface board known as the masterboard. The masterboard serves as the communication hub between the user and the robot, facilitating the reading of sensor data and the transmission of motor commands. This design ensures that the user can effectively monitor and control the robot's actions.

Fig. 24: Opened lower leg actuator of Bolt.

5.2.3 Mechanical Components

The mechanical architecture of Bolt is designed for both performance and scalability. Each actuator (See Fig. 24) incorporates a motor system that is reduced by a timing belt and pinion assembly with a 1/9 reduction ratio. This setup provides the necessary torque and precision for the robot's movements.

The actuators are designed to be generic and modular, allowing for easy chaining and scalability. This modularity is a significant advantage, as it enables the customization and expansion of ODRI robots to meet various application requirements. The robust mechanical design, combined with the advanced electronic components, makes Bolt a versatile and powerful platform for research and development in robotics.

5.3 Inertia tensor viewer

To examine the dynamic parameters of URDFs, we propose a tool for visualizing the inertia matrices of each robot link. The chosen object for representing the inertia is a solid parallelepiped, which serves as a reasonable approximation for the inertia of a link. The inertia tensor of such an object is provided in Equation (32).

$$I_{\text{box}} = \begin{pmatrix} \frac{1}{12}m(h^2+d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2+h^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2+d^2) \end{pmatrix} \tag{32}$$

To determine the dimensions of the object corresponding to the inertia tensor $\hat{I}$ of our link, we must diagonalize $\hat{I}$ to express it in terms of its principal axes I_p. We then solve the system described by Equation (33), with the solution given by Equation (34).

$$\begin{cases} I_{p,xx} = \frac{1}{12}m(h^2+d^2) \\ I_{p,yy} = \frac{1}{12}m(w^2+h^2) \\ I_{p,zz} = \frac{1}{12}m(w^2+d^2) \end{cases} \tag{33}$$

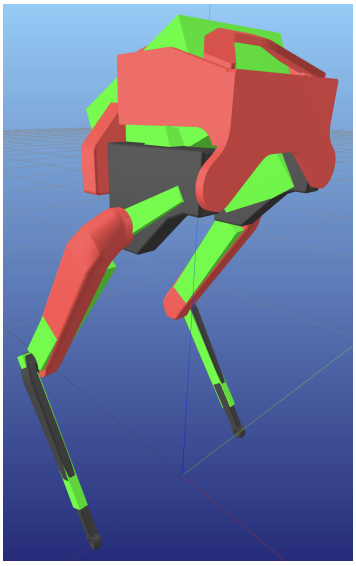


Fig. 25: Example of the inertia tensor visualization tool. The viewer used is Meshcat with a OnShape generated URDF of Bolt. Each link's inertia tensor is represented by a green box.

$$\begin{cases} h = \sqrt{6\frac{I_{p,xx}+I_{p,yy}-I_{p,zz}}{m}} \\ d = \sqrt{6\frac{I_{p,xx}-I_{p,yy}+I_{p,zz}}{m}} \\ w = \sqrt{6\frac{-I_{p,xx}+I_{p,yy}+I_{p,zz}}{m}} \end{cases} \tag{34}$$

The following algorithm summarizes the process for displaying a box representing the inertia tensor on each link:

Algorithm 2 Inertia Tensor Visualization

for all link **do**
  $\hat{I} \leftarrow$ link.inertia, $\hat{P} \leftarrow$ link.CoM
  $I_p, M \leftarrow$ diagonalize$(\hat{I})$
  $h, d, w \leftarrow$ solve (34) using I_p
  display$(h, d, w, \hat{P}, M)$
end for

where the *display* function creates a box with dimensions h, d, w and applies the rotation M to it as expressed in the link's frame. An example of visualization using Meshcat is provided in Figure 25.

6 Conclusion

To conclude, the objective of this internship was to enable the bipedal robot Bolt from ODRI to walk on flat ground.

To achieve this, we developed a control framework based on previous works [27] and simulated Bolt's control in PyBullet. The control pipeline combined a DCM MPC and a WB MPC for high-level control, along with a Riccati-based gain controller for low-level control. The sim-to-real transfer was not successful due to a lack of robustness.

Consequently, we shifted to a more robust control method, RL. We leveraged the CaT project [24], which involved RL with both soft and hard constraints and had been previously applied to the Solo12 robot, similar to Bolt. Despite promising simulations in Isaac Gym and PyBullet, we did not transition from sim to real. A modeling issue in the URDF was suspected, and tools were developed to debug this. Additionally, the robot encountered technical hardware issues, slowing progress.

In the future, during the PhD, the plan is to implement both RL and WB MPC controls on the real robot with the scientific goal of comparing the two approaches using performance metrics.

References

[1] Unitree. *Unitree's First Universal Humanoid Robot*. June 2024. URL: https://www.unitree.com/h1/.

[2] Fourier Intelligence. *General-purpose Humanoid Robot Fourier GR-1*. June 2024. URL: https://fourierintelligence.com/gr1/.

[3] LimX. *Bipedal Robot LimX P1*. 2024. URL: https://www.limxdynamics.com/en/biped-robot.

[4] Patrick M. Wensing et al. "Optimization-Based Control for Dynamic Legged Robots". In: *IEEE TRO* 40 (2024), pp. 43–63.

[5] Alessandro Assirelli et al. *Whole-Body MPC without Foot References for the Locomotion of an Impedance-Controlled Robot*. Technical report. 2022. URL: https://laas.hal.science/hal-03778738.

[6] He Li and Patrick M. Wensing. *Cafe-Mpc: A Cascaded-Fidelity Model Predictive Control Framework with Tuning-Free Whole-Body Control*. 2024. URL: http://arxiv.org/abs/2403.03995.

[7] Ewen Dantec et al. "Whole Body Model Predictive Control with a Memory of Motion: Experiments on a Torque-Controlled Talos". In: *ICRA*. May 2021, pp. 8202–8208.

[8] Debora Clever et al. "COCoMoPL: A Novel Approach for Humanoid Walking Generation Combining Optimal Control, Movement Primitives and Learning and its Transfer to the Real Robot HRP-2". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 977–984. DOI: 10.1109/LRA.2017.2657000.

[9] Teguh Santoso Lembono et al. "Learning how to walk: Warm-starting optimal control solver with memory of motion". In: *ICRA*. 2020, pp. 1357–1363.

[10] Rohan P. Singh et al. "Learning Bipedal Walking for Humanoids with Current Feedback". In: *IEEE Access* 11 (2023), pp. 82013–82023.

[11] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. *Humanoid Parkour Learning*. June 2024. URL: http://arxiv.org/abs/2406.10759.

[12] Ilija Radosavovic et al. "Real-world humanoid locomotion with reinforcement learning". In: *Science Robotics* 9.89 (2024).

[13] F. Jenelten et al. "DTC: Deep Tracking Control". In: *Science Robotics* 9.86 (2024).

[14] Angel Romero, Yunlong Song, and Davide Scaramuzza. *Actor-Critic Model Predictive Control*. 2024. arXiv: 2306.09852 [cs.RO]. URL: https://arxiv.org/abs/2306.09852.

[15] Johannes Englsberger, Christian Ott, and Alin Albu-Schaffer. "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion". In: *IEEE TRO* 31.2 (Apr. 2015), pp. 355–368.

[16] Zhao Zhang et al. "Robust Walking for Humanoid Robot Based on Divergent Component of Motion". In: *Micromachines* 13.7 (2022), p. 1095.

[17] Olivier Stasse et al. "Fast foot prints re-planning and motion generation during walking in physical human-humanoid interaction". In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009, pp. 284–289. DOI: 10.1109/ICHR.2009.5379563.

[18] Majid Khadiv et al. "Walking Control Based on Step Timing Adaptation". In: *IEEE TRO* 36.3 (2020), pp. 629–643.

[19] Mohammad Hasan Yeganegi et al. “Robust Walking Based on MPC With Viability Guarantees”. In: *IEEE TRO* 38.4 (Aug. 2022), pp. 2389–2404.

[20] Yuval Tassa, Tom Erez, and Emanuel Todorov. “Synthesis and stabilization of complex behaviors through online trajectory optimization”. In: *IROS*. 2012.

[21] J. Koenemann et al. “Whole-body model-predictive control applied to the HRP-2 humanoid”. In: *IROS*. 2015.

[22] Carlos Mastalli et al. “Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control”. In: *ICRA*. 2020.

[23] Anthony V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Vol. Volume 165. Academic Press, 1983.

[24] Elliot Chane-Sane et al. *CaT: Constraints as Terminations for Legged Locomotion Reinforcement Learning*. 2024. DOI: 10.48550/ARXIV.2403.18765. URL: https://arxiv.org/abs/2403.18765.

[25] Felix Grimminger et al. “An Open Torque-Controlled Modular Robot Architecture for Legged Locomotion Research”. In: *IEEE RA-L* 5.2 (2020), pp. 3650–3657.

[26] Constant Roux, Côme Perrot, and Olivier Stasse. “Whole-body MPC and sensitivity analysis of a real time foot step sequencer for a biped robot Bolt”. In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Oct. 2024.

[27] Elham Daneshmand et al. “Variable Horizon MPC With Swing Foot Dynamics for Bipedal Walking Control”. In: *IEEE RA-L* 6.2 (2021), pp. 2349–2356.

[28] B. Stellato et al. “OSQP: an operator splitting solver for quadratic programs”. In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672.

[29] Ewen Dantec, Michel Taix, and Nicolas Mansard. “First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback”. In: *IEEE RA-L* 7.2 (2022), pp. 4448–4455.

[30] *Differential dynamic programming*. 2024. URL: https://en.wikipedia.org/wiki/Differential_dynamic_programming.

[31] Ignat Georgiev. *Deriving Differential Dynamic Programming*. 2023. URL: http://www.imgeorgiev.com/2023-02-01-ddp/.

A DCM expression

Several assumptions are made to simplify the analysis of the LIPM:

- The contact points are only on the floor, so the Center of Pressure (CoP) p is only on the ground:
$$p_z = 0 \tag{35}$$
- The sum of forces can be expressed as a unique force λ expressed in the center of mass c of the robot, with m the total mass of the robot and $g = 9.81\, m \cdot s^{-2}$ the gravity constant:
$$\begin{cases} m(\ddot{c} - g) = \lambda \\ mc \times (\ddot{c} - g) + \dot{L} = p \times \lambda \end{cases} \tag{36}$$
- The angular momentum $\dot{L}$ is neglected, in front of the linear momentum:
$$\dot{L} = 0 \tag{37}$$
- The pendulum is virtually constrained in a horizontal plane, at height z_c:
$$c_z = z_c, \ddot{c}_z = 0 \tag{38}$$

By applying the hypothesis (37) on the system (36), we obtain the equation (39).

$$\begin{cases} m(\ddot{c} - g) = \lambda & \text{(39a)} \\ mc \times (\ddot{c} - g) = p \times \lambda & \text{(39b)} \end{cases}$$

Next, we inject the equation (39a) into the equation (39b) and use the skew matrix operator to develop the cross-product (equation (40) and (41)).

$$m \begin{pmatrix} 0 & -c_z & c_y \\ c_z & 0 & -c_x \\ -c_y & c_x & 0 \end{pmatrix} \begin{pmatrix} \ddot{c}_x \\ \ddot{c}_y \\ \ddot{c}_z + g \end{pmatrix} = m \begin{pmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{pmatrix} \begin{pmatrix} \ddot{c}_x \\ \ddot{c}_y \\ \ddot{c}_z + g \end{pmatrix} \tag{40}$$

$$\begin{cases} -c_z\ddot{c}_y + c_y(\ddot{c}_z + g) = p_y(\ddot{c}_z + g) - p_z\ddot{c}_y & \text{(41a)} \\ c_z\ddot{c}_x - c_x(\ddot{c}_z + g) = -p_x(\ddot{c}_z + g) + p_z\ddot{c}_x & \text{(41b)} \\ c_x\ddot{c}_y - c_y\ddot{c}_x = p_x\ddot{c}_y - p_y\ddot{c}_x & \text{(41c)} \end{cases}$$

By applying the assumption (35) on the equation (41a) (resp. equation (41b)), we obtain the equation (42a) (resp. equation 42b).

$$\begin{cases} p_x = c_x - \dfrac{c_z}{\ddot{c}_z + g}\ddot{c}_x & \text{(42a)} \\ p_y = c_y - \dfrac{c_z}{\ddot{c}_z + g}\ddot{c}_y & \text{(42b)} \end{cases}$$

By applying the assumption (38) on the equation (42a) (resp. equation (42b)), we obtain the equation (43a) (resp. equation (43b)).

$$\begin{cases} p_x = c_x - \dfrac{z_c}{g}\ddot{c}_x & \text{(43a)} \\ p_y = c_y - \dfrac{c_z}{g}\ddot{c}_y & \text{(43b)} \end{cases}$$

With $w_0 = \sqrt{\frac{g}{z_c}}$, equation (43) becomes equation (44), or in a vector shape the equation (45) with $p = \begin{pmatrix} p_x & p_y \end{pmatrix}^T$ and $c = \begin{pmatrix} c_x & c_y \end{pmatrix}^T$.

$$\begin{cases} p_x = c_x - \dfrac{\ddot{c}_x}{w_0^2} & \text{(44a)} \\ p_y = c_y - \dfrac{\ddot{c}_y}{w_0^2} & \text{(44b)} \end{cases}$$

$$p = c - \frac{\ddot{c}}{w_0^2} \tag{45}$$

By rearranging the equation (45), we obtain the equation (46).

$$\ddot{c} = w_0^2(c - p) \tag{46}$$

We then introduce the DCM, expressed in [18] (resp. the derivative of the DCM) as the equation (47) (resp. (48)).

$$\zeta = c + \frac{\dot{c}}{w_0} \tag{47}$$

$$\dot{\zeta} = \dot{c} + \frac{\ddot{c}}{w_0} \tag{48}$$

By injecting the DCM definition in (46), we obtain the system (49), expressed as a state representation in (50).

$$\begin{cases} \dot{c} = w_0(\zeta - c) & \text{(49a)} \\ \dot{\zeta} = w_0(\zeta - p) & \text{(49b)} \end{cases}$$

$$\begin{pmatrix} \dot{c} \\ \dot{\zeta} \end{pmatrix} = \begin{pmatrix} -w_0 & w_0 \\ 0 & w_0 \end{pmatrix} \begin{pmatrix} c \\ \zeta \end{pmatrix} + \begin{pmatrix} 0 \\ -w_0 \end{pmatrix} p \tag{50}$$

The state matrix eigen value of the DCM part $\lambda_2 = w_0$ is positive, which describes the instability of the linear inverted pendulum.

B DCM nominal offset

Khadiv et al. [18] computes analytically the DCM offset following the equations 51 and 52, with $b_i = \zeta(T_i) - p_i$ and $b_{i-1} = \zeta(T_{i-1}) - p_{i-1}$ where i denotes the next footstep and $i-1$ the current stance foot.

$$\zeta(T_i) = (\zeta(T_{i-1}) - p_{i-1})e^{w_0(T_i - T_{i-1})} + p_{i-1} \tag{51}$$

$$b_i + p_i = b_{i-1}e^{w_0(T_i - T_{i-1})} + p_{i-1} \tag{52}$$

We then develop the DCM offset along the x-axis, with the assumption that the x-component of the DCM offset is constant over the sequence, ie $b_{i,x} = b_{i-1,x}$ (equations 53 to 55) and that $L = p_{i,x} - p_{i-1,x}$. The nominal x-DCM offset is given by the equation 56 when L and T_i are nominal.

$$b_{i,x} + p_{i,x} = b_{i-1,x}e^{w_0(T_i - T_{i-1})} + p_{i-1,x} \tag{53}$$

$$L = b_{i,x}(e^{w_0(T_i - T_{i-1})} - 1) \tag{54}$$

$$b_{i,x} = \frac{L}{e^{w_0(T_i - T_{i-1})} - 1} \tag{55}$$

$$b_{i,x,\text{nom}} = \frac{L_{\text{nom}}}{e^{w_0(T_{\text{nom}} - T_{i-1})} - 1} \tag{56}$$

Along the y-axis, when the right foot is stance (resp. left foot is stance) and with $b_i = b_{y,l}$, $b_{i-1} = b_{y,r}$, $p_i - p_{i-1} = w_r$ (resp. $b_i = b_{y,r}$, $b_{i-1} = b_{y,l}$, $p_i - p_{i-1} = w_l$) we obtain the equations 57 and 58 (resp. 57 and 59). By solving the system 60, we obtain solution 61. The nominal y-DCM offsets are given by the solution 62 when w_l, w_r and T_i are nominal.

$$b_{i,y} + p_{i,y} = b_{i-1,y}e^{w_0(T_i - T_{i-1})} + p_{i-1,y} \tag{57}$$

$$w_r = b_{y,r}e^{w_0(T_i - T_{i-1})} - b_{y,l} \tag{58}$$

$$w_l = b_{y,l}e^{w_0(T_i - T_{i-1})} - b_{y,r} \tag{59}$$

$$\begin{cases} w_r = b_{y,r}e^{w_0(T_i - T_{i-1})} - b_{y,l} \\ w_l = b_{y,l}e^{w_0(T_i - T_{i-1})} - b_{y,r} \end{cases} \tag{60}$$

$$\begin{cases} b_{y,r} = \frac{w_l + w_r e^{w_0(T_i - T_{i-1})}}{e^{w_0(T_i - T_{i-1})^2} - 1} \\ b_{y,l} = \frac{w_r + w_l e^{w_0(T_i - T_{i-1})}}{e^{w_0(T_i - T_{i-1})^2} - 1} \end{cases} \tag{61}$$

$$\begin{cases} b_{y,r,\text{nom}} = \frac{w_{l,\text{nom}} + w_{r,\text{nom}} e^{w_0(T_{\text{nom}} - T_{i-1})}}{e^{w_0(T_{\text{nom}} - T_{i-1})^2} - 1} \\ b_{y,l,\text{nom}} = \frac{w_{r,\text{nom}} + w_{l,\text{nom}} e^{w_0(T_{\text{nom}} - T_{i-1})}}{e^{w_0(T_{\text{nom}} - T_{i-1})^2} - 1} \end{cases} \tag{62}$$

C Differential Dynamic Programming

This appendix provides a streamlined overview of the DDP method, derived from the sources [30] and [31].

C.1 Finite-Horizon Discrete-Time Problems

In a discrete-time context, the OCP is formulated over a finite time horizon comprising N time steps. The objective is to minimize the cumulative cost while adhering to the system's dynamics.

The system dynamics are described by the following equation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{63}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_\mathbf{x}}$ denotes the state vector at time step k, and $\mathbf{u}_k \in \mathbb{R}^{n_\mathbf{u}}$ represents the control vector.

The cost function to be minimized is given by:

$$J_0 = \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + \Phi(\mathbf{x}_N) \tag{64}$$

where $l(\mathbf{x}_k, \mathbf{u}_k)$ is the running cost at time step k, and $\Phi(\mathbf{x}_N)$ represents the terminal cost.

C.2 Dynamic Programming

Define $\mathbf{U}_i = \{\mathbf{u}_i, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_{N-1}\}$, and let the *cost-to-go* function J_i be expressed as:

$$J_i = \sum_{k=i}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + \Phi(\mathbf{x}_N) \tag{65}$$

The value function is then defined as the optimal *cost-to-go*, achieved by minimizing over the control sequence:

$$V_i = \min_{\mathbf{U}_i} J_i \tag{66}$$

With $V_N = \Phi(\mathbf{x}_N)$, the Bellman equation is formulated as:

$$\begin{aligned} V_i &= \min_{\mathbf{U}_i} \left(\sum_{k=i}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + \Phi(\mathbf{x}_N) \right) \\ &= \min_{\mathbf{U}_i} \left(l(\mathbf{x}_i, \mathbf{u}_i) + \sum_{k=i+1}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + \Phi(\mathbf{x}_N) \right) \\ &= \min_{\mathbf{U}_i} \left(l(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1} \right) \end{aligned} \tag{67}$$

C.3 Differential Dynamic Programming

The DDP algorithm iterates between a backward pass and a forward pass to progressively refine the control policy until convergence. We will first address the backward pass.

C.3.1 Taylor Series Expansion

We approximate $(l(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1})$ using a second-order Taylor expansion around $\begin{pmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{u}}\end{pmatrix}$:

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = \frac{1}{2}\begin{pmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{pmatrix}^T \begin{pmatrix}0 & Q_{\mathbf{x}}^T & Q_{\mathbf{u}}^T\\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}}\\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}}\end{pmatrix}\begin{pmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{pmatrix} \tag{68}$$

where $\delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\delta\mathbf{u} = \mathbf{u} - \bar{\mathbf{u}}$. The matrices are defined as:

- $Q_{\mathbf{x}} = l_{\mathbf{x}} + f_{\mathbf{x}}^T V_{\mathbf{x},i+1}$
- $Q_{\mathbf{u}} = l_{\mathbf{u}} + f_{\mathbf{u}}^T V_{\mathbf{x},i+1}$
- $Q_{\mathbf{xx}} = l_{\mathbf{xx}} + f_{\mathbf{x}}^T V_{\mathbf{xx},i+1} f_{\mathbf{x}}$
- $Q_{\mathbf{uu}} = l_{\mathbf{uu}} + f_{\mathbf{u}}^T V_{\mathbf{xx},i+1} f_{\mathbf{u}}$
- $Q_{\mathbf{xu}} = l_{\mathbf{xu}} + f_{\mathbf{x}}^T V_{\mathbf{xx},i+1} f_{\mathbf{u}}$

C.3.2 Backward Pass

During the backward pass, we seek to determine the optimal adjustment to the control sequence. The optimal change $\delta\mathbf{u}^*$ is found by minimizing $Q(\delta\mathbf{x}, \delta\mathbf{u})$:

$$\delta\mathbf{u}^* = \arg\min_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = -Q_{\mathbf{uu}}^{-1}(Q_{\mathbf{u}} + Q_{\mathbf{ux}}\delta\mathbf{x}) = k + K\delta\mathbf{x} \tag{69}$$

where $k = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}}$ represents the open-loop term and $K = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}$ is the feedback gain term.

C.3.3 Forward Pass

In the forward pass, we compute the optimal control sequence based on the results from the backward pass:

$$\delta\mathbf{u}^* = \mathbf{u}^* - \bar{\mathbf{u}} \Rightarrow \mathbf{u}^* = \delta\mathbf{u}^* + \bar{\mathbf{u}} = k + K\delta\mathbf{x} + \bar{\mathbf{u}} \tag{70}$$

The optimal state sequence is then determined using the system dynamics equation.

C.4 Simplified DDP Algorithm

The following algorithm outlines the procedure for performing the backward and forward passes iteratively until convergence criteria are met:

Algorithm 3 Simplified DDP

Input: $\mathbf{x}_0, \bar{\mathbf{U}}_0$
$V_N \leftarrow \Phi(\mathbf{x}_N)$
repeat
  for $k = [N-1:0]$ **do**
    $\delta\mathbf{u}_k^* \leftarrow k_k + K_k \delta\mathbf{x}_k$
  end for

  $\mathbf{x}_0^* \leftarrow \mathbf{x}_0$
  for $k = [0:N-1]$ **do**
    $\mathbf{u}_k^* \leftarrow \delta\mathbf{u}_k^* + \bar{\mathbf{u}}_k$
    $\mathbf{x}_{k+1}^* \leftarrow f(\mathbf{x}_k^*, \mathbf{u}_k^*)$
  end for
  $\bar{\mathbf{U}}_0 \leftarrow \mathbf{U}_0$
until Convergence criteria are met