Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Distributed Inverse Dynamics Control for Quadruped Robots using Geometric Optimization

Nimesh Khandelwal

nimesh20@iitk.ac.in

Indian Institute of Technology Kanpur

Amritanshu Manu

Indian Institute of Technology Kanpur

Shakti S. Gupta

Indian Institute of Technology Kanpur

Mangal Kothari

Prashanth Krishnamurthy

New York University

Farshad Khorrami

New York University

Research Article

Keywords: Quadruped robot, Reactive controller, Inverse dynamics, Friction cone constraint, Geometric optimization

Posted Date: May 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9731913/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Distributed Inverse Dynamics Control for Quadruped Robots using Geometric Optimization

Nimesh Khandelwal[1*], Amritanshu Manu[1], Shakti S. Gupta[1], Mangal Kothari[1], Prashanth Krishnamurthy[2], Farshad Khorrami[2]

[1*]Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur, 208016, Uttar Pradesh, India.
[2]Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, USA.

*Corresponding author(s). E-mail(s): nimesh20@iitk.ac.in;
Contributing authors: amritanshu@iitk.ac.in; ssgupta@iitk.ac.in; mangal@iitk.ac.in; prashanth.krishnamurthy@nyu.edu; khorrami@nyu.edu;

Abstract

This paper presents a distributed inverse dynamics controller (DIDC) for quadruped robots that addresses the limitations of existing reactive controllers: simplified dynamical models, the inability to handle exact friction cone constraints, and the high computational and energy requirements of whole-body controllers. Current methods either ignore friction constraints entirely or use linear approximations, which can lead to potential slip and instability, while comprehensive whole-body controllers require significant computational resources. Our approach uses full rigid-body dynamics and enforces exact friction cone constraints through a novel geometric optimization-based solver. DIDC combines the generalized forces required for the actuated and unactuated spaces by projecting them onto the actuated space, while satisfying physical constraints and maintaining orthogonality between the base and joint tracking objectives. Experimental validation shows that our approach reduces the required energy (and the cost-of-transport) and improves orientation tracking. The geometric solver is benchmarked against existing off-the-shelf QP solvers for the quadruped control optimization task. The controller enables stable omnidirectional trotting at various speeds and consumes less power than comparable methods, while running efficiently on embedded processors.

Keywords: Quadruped robot, Reactive controller, Inverse dynamics, Friction cone constraint, Geometric optimization

1 Introduction

Quadruped robots are being deployed in the real world for autonomous inspection, mapping, and surveillance operations in different environments. This has been realized by developing versatile locomotion control algorithms, suitable for fast execution on limited onboard computational resources. Locomotion control algorithms can be broadly classified into model-based, e.g., [3, 8, 11, 12, 14] and learning-based algorithms, e.g., [18, 21, 22]. A current trend in state-of-the-art model-based control implementations is to use a combination of predictive and reactive controllers, e.g., [23, 26, 32]. In this setup, the predictive controller utilizes a simplified model of the robot's dynamics and its interactions with the environment to provide the reference state trajectory and control inputs for a time horizon. The reactive controller uses a more accurate dynamics model, enforces state and control constraints, and prioritizes different tracking tasks. Thus, the reactive layer generates physically consistent joint torque commands at each time instant, allowing the robot to respond quickly to unexpected disturbances.

Reactive controllers for quadruped robots have been extensively explored in the literature over the last two decades. The initial approach was motivated by the inverse dynamics control (IDC) of robotic manipulators [31]. However, in the case of quadruped robots, the underactuated dynamics, dynamically switching contact states, and unknown contact forces proved to be a challenge. This was resolved by eliminating the contact force from the equation of motion of the system by using null-space projection and orthogonal decomposition of the contact jacobian, e.g. [6, 25, 29]. Although classic inverse dynamics control provides an analytical form for the joint torques, it does not ensure that the contact forces generated by these torques satisfy the unilaterality and friction cone constraints. This limitation was overcome by the introduction of quadratic program (QP) based methods to find the optimal distribution of contact forces [3], [17]. Using single rigid body dynamics (SRBD), a linear approximation of the friction cone (as a friction pyramid), and mature open-source QP solvers (like qpOASES [13], HPIPM [16], etc.), these methods are suitable for onboard implementations.

Subsequently, more comprehensive QP formulations have been developed that use the full rigid body dynamics (RBD), simultaneously optimize joint torques, joint accelerations, and contact forces, and achieve prioritized constraint satisfaction. These are collectively known as Whole Body Controllers (WBC), e.g. [7, 10, 28]. Prioritized constraint satisfaction in the WBC can be handled either by formulating a single QP with weighted objectives or by formulating multiple QPs that are solved hierarchically. However, solving the WBC problem at a fast enough rate does require a high-end onboard processor; for instance, the ANYmal robot [20] runs WBC at 400 Hz using an Intel Core i7-8850H processor [32]. This is primarily due to the increased size of the resulting optimization problem for handling multiple constraints. Additionally, to

keep the optimization problem as a QP, the WBC formulations need to use the friction pyramid approximation. In [1], the proposed controller uses the full-RBD model and the exact friction cone to form a QCQP, which is solved using an interior-point algorithm. However, an onboard implementation of the controller for a quadruped robot is not provided. Thus, it is of value to develop a reactive controller that uses accurate system dynamics and exact friction constraints while running comfortably on an embedded processor (e.g., ARM64 processors).

In this work, we propose a reactive controller, called the distributed inverse dynamics controller (DIDC), that uses the full-RBD model of a quadruped robot and enforces the exact friction cone constraint. We also propose a geometric-optimization-based custom solver that keeps the controller computationally lightweight. The optimization problem involved in our controller is inspired by [3], and our custom solver to handle the cone constraint leverages the projection operator proposed in [33] in the context of friction cones. We develop an approach to project the generalized forces required for unactuated degrees of freedom (DOFs) into the space of actuated DOFs while enforcing the cone constraints. The DIDC also maintains the orthogonality of the control actions for the unactuated and the actuated space so that the tracking of base motion is unaffected by joint tracking. This work also includes a control architecture using the DIDC along with a heuristic planner for the base-motion and footholds, and an extended Kalman filter (EKF) based state estimator. The proposed control architecture is validated for omnidirectional trotting at various speeds, in simulation, and on hardware.

In the following, we give an overview of the RBD and the IDC in Section 2. We describe the proposed reactive controller and the geometric-optimization-based solver in Section 3 and Section 4, respectively. The motion planner and the state estimator modules of the control architecture are discussed in Section 5, followed by the experimental results in Section 6.

2 Reactive Control of Quadruped Robots

2.1 Model formulation

The DOFs of the robot are represented by the generalized coordinates vector $\mathbf{q} \in \mathbb{R}^{18}$. Since the system is underactuated, the DOFs are partitioned into the 6-DOF unactuated space for the robot base, and the 12-DOF actuated space for the leg joints, as $\mathbf{q} = \begin{bmatrix}\mathbf{q}_b^T & \mathbf{q}_j^T\end{bmatrix}^T$. Here, $\mathbf{q}_b = \begin{bmatrix}\mathbf{r}_b^T & \mathbf{\Phi}_b^T\end{bmatrix}^T \in \mathbb{R}^6$, where $\mathbf{r}_b \in \mathbb{R}^3$ is the position vector of the robot center-of-mass (COM) and $\mathbf{\Phi}_b \in \mathbb{R}^3$ is the XYZ-Euler angle vector representing the orientation of the robot base. $\mathbf{q}_j \in \mathbb{R}^{12}$ is the vector of joint angles for all four legs, ordered as front-left (FL), front-right (FR), rear-left (RL), and rear-right (RR), with each leg numbered 0, 1, 2, and 3, respectively.

The general equation of motion (EOM) of a quadruped robot, modeled as a rigid-body system with contact constraints, is expressed as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_c^T\mathbf{F}_c, \tag{1}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{18\times18}$ is the joint space inertia matrix of the system, and $\boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{18}$ represents the nonlinear terms corresponding to Coriolis, centrifugal, and gravitational effects. $\mathbf{S} = \begin{bmatrix}\mathbf{0}_{12\times6} & \mathbf{I}_{12\times12}\end{bmatrix}$ is the selection matrix representing the underactuation since only the joint DOFs are actuated. $\boldsymbol{\tau} \in \mathbb{R}^{12}$ is the joint torque command, n_c is the number of feet in contact, $\mathbf{J}_c \in \mathbb{R}^{3n_c\times18}$ is the contact jacobian matrix, and $\mathbf{F}_c \in \mathbb{R}^{3n_c}$ are the contact forces on the feet of the robot. The contact constraint enforces the no-slip condition at the point of contact between the foot and the environment. It is expressed mathematically as

$$\begin{aligned}\mathbf{J}_{c,i}\dot{\mathbf{q}} &= \mathbf{0},\\ \mathbf{J}_{c,i}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{c,i}\dot{\mathbf{q}} &= \mathbf{0},\end{aligned} \tag{2}$$

where $\mathbf{J}_{c,i}$ and $\dot{\mathbf{J}}_{c,i}$ are the sub-matrix of the contact jacobian, $\mathbf{J}_c$, and its time derivative, $\dot{\mathbf{J}}_{c,i}$, corresponding to feet $i \in \mathcal{C}$, where $\mathcal{C}$ is the set of feet in contact with the ground. For physical correctness, the feet in contact should only push into the environment, and the contact forces must satisfy the friction constraints. These are together referred to as contact force constraints. Their mathematical expression is discussed in Equations (12) and (13) in Section 3.

The physical quantities used in the planner, estimator, and controller (PEC) computations are expressed either in the global (inertial) frame $\mathcal{G}$ or the local frame $\mathcal{B}$ that is attached to the COM and has the same orientation as the robot base. The axes of these frames are denoted with a superscript describing the frame, e.g. $X^{\mathcal{G}}$, whereas the notation ${}^{\mathcal{B}}\mathbf{r}_b$ implies that $\mathbf{r}_b$ is expressed in the frame $\mathcal{B}$.

2.2 QP-based balance control (BC)

A prevalent class of reactive controllers, as described in Section 1, uses simplified dynamics models and formulates a constrained QP to solve for desired contact forces acting at the stance legs, and uses only the leg dynamics to compute the desired joint torques for the swing legs. Although these controllers compensate for the total mass of the robot while satisfying the linearized friction constraints, they fail to account for the effect of leg dynamics on the base. Also, the optimal torques required for base motion are modified by the direct addition of torques required for swing leg motion, and base motion tracking.

2.3 Whole Body Control (WBC)

This class of controller poses the solution of Eq. (1) as a single constrained QP that include the full-body dynamics and system constraints. In this work, we use the formulation described in [7], called the Weighted WBC. In this, multiple cost terms are weighted and added together to form the final cost term. Both the cost and the constraints are formulated as tasks. In the cost, we use the swing leg tracking task, base acceleration task, and contact force regularization task with heuristically chosen weights. The friction pyramid task, EOM task, torque limit task, and contact constraint task are combined as linear constraints. These are combined into a single convex QP and solved using an off-the-shelf solver. In the current literature, WBC is the standard model-based reactive controller for quadruped robots.

The shortcomings of BC, WBC, and the related family of reactive controllers are overcome by the control formulation described in this work.

3 Distributed Inverse Dynamics Control

The DIDC uses the dynamics model in (1) to calculate joint torque command $\boldsymbol{\tau}$ while satisfying contact constraints and the friction cone constraint. The key ideas of the DIDC are summarized below:

1. considering the system as fully-actuated and computing the required generalized forces $\boldsymbol{\tau}_f = \begin{bmatrix}\boldsymbol{\tau}_b^T & \boldsymbol{\tau}_j^T\end{bmatrix}^T \in \mathbb{R}^{18}$,
2. mapping the generalized forces for the unactuated space $\boldsymbol{\tau}_b \in \mathbb{R}^6$ to the control input in the actuated space (expressed here as $\boldsymbol{\tau}_1 \in \mathbb{R}^{12}$), while enforcing the contact and friction constraints (14),
3. and adding the generalized forces for the actuated space $\boldsymbol{\tau}_j \in \mathbb{R}^{12}$ to $\boldsymbol{\tau}_1$, while maintaining the orthogonality of the control actions, to get $\boldsymbol{\tau}$ (17).

In the following, we describe in detail each of these steps in the computation of $\boldsymbol{\tau}$. We begin by expressing the general EOM in (1) with the introduction of $\boldsymbol{\tau_f}$ as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau_f}, \tag{3}$$

assuming that all the DOFs are actuated and that no external forces are acting at the feet. Given the commanded generalized acceleration $\ddot{\mathbf{q}}_{cmd}$, we can calculate $\boldsymbol{\tau}_f$ as

$$\boldsymbol{\tau}_f = \hat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_{cmd} + \hat{\boldsymbol{\eta}}(\mathbf{q}, \dot{\mathbf{q}}), \tag{4}$$

where $\hat{(.)}$ denotes the available estimates of the corresponding quantities of the system. Substituting the value of $\boldsymbol{\tau}_f$ in (3) using (4), we express $\ddot{\mathbf{q}}$ as

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\hat{\mathbf{M}}\ddot{\mathbf{q}}_{cmd} + \tilde{\boldsymbol{\eta}}), \tag{5}$$

where $\tilde{\boldsymbol{\eta}} = \hat{\boldsymbol{\eta}} - \boldsymbol{\eta}$, and we have dropped the explicit dependence on $\mathbf{q}$ and $\ddot{\mathbf{q}}$ for notational simplicity. With further manipulation of (5), we arrive at a partially linearized dynamics of the form

$$\begin{aligned}\ddot{\mathbf{q}} &= \ddot{\mathbf{q}}_{cmd} + \mathbf{M}^{-1}(\hat{\mathbf{M}}\ddot{\mathbf{q}}_{cmd} + \tilde{\boldsymbol{\eta}}) - \mathbf{M}^{-1}\mathbf{M}\ddot{\mathbf{q}}_{cmd}, \\ \ddot{\mathbf{q}} &= \ddot{\mathbf{q}}_{cmd} + \boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_{cmd}),\end{aligned} \tag{6}$$

where the quantity $\boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_{cmd})$ is the uncertainty in the modeling and is defined as

$$\boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_{cmd}) = \mathbf{M}^{-1}(\tilde{\mathbf{M}}\ddot{\mathbf{q}}_{cmd} + \tilde{\boldsymbol{\eta}}),$$

with $\tilde{\mathbf{M}} = \hat{\mathbf{M}} - \mathbf{M}$. If we assume no uncertainty in the modeling, the dynamics model in (6) reduces to a linear ODE

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{cmd}. \tag{7}$$

This is the decoupled linearized dynamics of the system since each DOF can be controlled individually by their corresponding commanded acceleration, $\ddot{\mathbf{q}}_{cmd}$, which is calculated as

$$\ddot{\mathbf{q}}_{cmd} = \ddot{\mathbf{q}}_{des} + \mathbf{K}_p(\mathbf{q}_{des} \ominus \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_{des} - \dot{\mathbf{q}}), \tag{8}$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ are the stiffness and damping coefficient matrices, chosen to be diagonal here, and $(.)_{des}$ denotes the desired quantities provided by the motion planner. The symbol $\ominus$ denotes the space-preserving subtraction operator. It differs from the algebraic subtraction operator $(-)$ for the orientation coordinates since they belong to a non-Euclidean space. The $\ominus$ operator is used here to calculate the error in the orientation space and represent it in $\mathbb{R}^3$.

Since there are multiple quantities sharing similar notation, differing only by their subscripts, we list them here for easier understanding hereafter.

- $\mathbf{J}_c$: jacobian for the feet in contact.
- $\mathbf{J}_{ab,c}$: sub-matrix of the contact jacobian matrix relating base wrench and contact forces for feet in contact with the ground.
- $\mathbf{J}_{ab}$: structure of $\mathbf{J}_{ab,c}$ extended for all feet irrespective of ground contact.
- $\mathbf{J}_{aa,c}$: sub-matrix of the contact jacobian matrix relating contact forces and joint torques for feet in contact with ground.
- $\mathbf{J}_{aa}$: structure of $\mathbf{J}_{aa,c}$ extended for all feet irrespective of ground contact.
- $\mathbf{J}_{a,b}$: jacobian matrix relating joint torques and base wrench. Combination of $\mathbf{J}_{ab}$ and $\mathbf{J}_{aa}$.
- $\boldsymbol{\tau}$: final joint torque commands sent to the actuator.
- $\boldsymbol{\tau}_1$: joint torques required for base motion tracking.

To control the 18-DOF system, $\boldsymbol{\tau}_f$ must be distributed over the actuated space to calculate the joint torque command. Since the robot interacts with the environment through its feet, it can produce the desired base wrench, $\boldsymbol{\tau}_b$, only by manipulating the contact forces, $\mathbf{F}_c$, at its feet. These are related by the submatrix $\mathbf{J}_{ab,c}$ of the contact jacobian $\mathbf{J}_c = \begin{bmatrix}\mathbf{J}_{ab,c} & \mathbf{J}_{aa,c}\end{bmatrix}$ as

$$\mathbf{J}_{ab,c}^T \mathbf{F}_c = \boldsymbol{\tau}_b. \tag{9}$$

The matrix $\mathbf{J}_{ab,c}^T \in \mathbb{R}^{6\times 3n_c}$ has the following structure

$$\mathbf{J}_{ab,c}^T = \begin{bmatrix} \mathbf{I}_{3\times 3} & \dots & \mathbf{I}_{3\times 3} \\ (\mathbf{r}_{p_i} - \mathbf{r}_b)^\times & \dots & (\mathbf{r}_{p_{n_c}} - \mathbf{r}_b)^\times \end{bmatrix}, \tag{10}$$

where $\mathbf{r}_{p_i} \forall\, i \in \{1, 2, \dots, n_c\}$ is the position of the feet in contact, $\mathbf{r}_b$ is the position of the COM. The superscript $(\times)$ in $\mathbf{r}^\times \in \mathfrak{so}(3)$ such that $(\mathbf{r}^\times)^T = -\mathbf{r}^\times$ represents the skew-symmetric matrix form of the vector $\mathbf{r} \in \mathbb{R}^3$. The matrix $\mathbf{J}_{aa,c} \in \mathbb{R}^{3n_c \times 3n_c}$ is the jacobian between the feet that are in contact with the ground and its corresponding joint space. Note that (9) is similar to existing QP-based controller formulations

in [3] and [17]. The main difference here is that the term $\boldsymbol{\tau}_b$ accounts for the complete dynamics of the system, including the interactions between different links of the quadruped robot, instead of the simplified dynamics (such as SRBD or centroidal dynamics). Specifically, the nonlinear effects $\boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}})$ have a non-negligible contribution to the desired base wrench.

To solve Equation (9), we formulate it as a quadratic optimization problem with regularization

$$\begin{aligned} \mathbf{F}_c^* = \arg\min_{\mathbf{F}_c} \quad & (\mathbf{J}_{ab,c}^T \mathbf{F}_c - \boldsymbol{\tau}_b)^T \mathbf{S}_1 (\mathbf{J}_{ab,c}^T \mathbf{F}_c - \boldsymbol{\tau}_b) \\ & + \mathbf{F}_c^T \mathbf{W} \mathbf{F}_c \\ & + (\mathbf{F}_c - \mathbf{F}_{c,prev}^*)^T \mathbf{V} (\mathbf{F}_c - \mathbf{F}_{c,prev}^*) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{F}_c) \leq \mathbf{0}, \end{aligned} \tag{11}$$

where $\mathbf{g}(\mathbf{F}_c)$ represents the contact constraints stacked as a single vector. $\mathbf{F}_{c,prev}^*$ is the solution from the previous time step. The second and third terms in the cost function penalize large contact forces and large contact impulses, respectively. The matrix, $\mathbf{S}_1$, is the relative weighting matrix of the resulting base wrench. Its values are chosen to prioritize base orientation stabilization over base translation motion. The matrices, $\mathbf{W}$ and $\mathbf{V}$, define the regularization of contact forces and impulses. The solution of this quadratic problem is subject to the unilaterality and maximum normal force constraint

$$F_{c,\max}^z \geq F_c^z \geq F_{c,\min}^z \geq 0, \tag{12}$$

along with the friction cone constraint

$$\sqrt{(F_c^x)^2 + (F_c^y)^2} \leq \mu |F_c^z|, \tag{13}$$

represented here as a single constraint vector $\mathbf{g}(\mathbf{F}_c)$. μ is the coefficient of friction. The optimization algorithm used to solve (11) is explained in Section 4.

The solution of this quadratic optimization is the required forces on the feet in contact, $\mathbf{F}_c^*$. These are combined with zero force vectors for the swing feet to get the final foot force vector $\mathbf{F}^* \in \mathbb{R}^{12}$ as the contact forces required for base tracking. We can get the required joint torque command $\boldsymbol{\tau}_1 \in \mathbb{R}^{12}$ for tracking the base motion from $\mathbf{F}^*$ using the combined jacobian for all four legs $\mathbf{J}_{aa} \in \mathbb{R}^{12\times 12}$

$$\boldsymbol{\tau}_1 = -\mathbf{J}_{aa}^T \mathbf{F}^*. \tag{14}$$

To ensure tracking for all the leg joints, $\boldsymbol{\tau}_j$ is the required additional joint torque. However, a direct addition of $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_j$ will modify the resulting wrench on the base as the joint torque commands for the stance legs will change. Here, we prioritize tracking of the base motion over joints by projecting $\boldsymbol{\tau}_j$ into the null space of the jacobian $\mathbf{J}_{a,b}$ mapping the base wrench $\boldsymbol{\tau}_b$ to joint torques. We use a composition of jacobians to define $\mathbf{J}_{a,b} = -\mathbf{J}_{aa}^T (\mathbf{J}_{ab}^T)^\dagger$. Here, $\mathbf{J}_{a,b} \in \mathbb{R}^{12\times 6}$. For a given total joint torque vector, $\boldsymbol{\tau}$,

the effective wrench on the base can be written as

$$\hat{\boldsymbol{\tau}}_b = (\mathbf{J}_{a,b})^{\dagger}\boldsymbol{\tau}. \tag{15}$$

To understand this mapping, it is helpful to consider that if we neglect the contact constraints, Equation (9) can be solved for $\mathbf{F}_c^*$ by taking the pseudo-inverse of $\mathbf{J}_{ab,c}^T$ and multiplying it with $\boldsymbol{\tau}_b$. The resulting contact forces, padded with zero forces for swing feet and multiplied by $-\mathbf{J}_{aa}^T$ would give us the base tracking torques. The solution we get from the optimization is an element of the subset of this solution space, defined by the contact constraints.

Since $\mathbf{J}_{a,b}$ defines the mapping from the body wrench to joint torques, the additional torques required for joint tracking should lie in the null space of $\mathbf{J}_{a,b}$ so that the base motion tracking is not affected. The corresponding null-space matrix is defined as

$$\mathbf{N}_{a,b} = \mathbf{I}_{12\times 12} - \mathbf{J}_{a,b}(\mathbf{J}_{a,b})^{\dagger} \tag{16}$$

The final joint torque command is calculated as follows

$$\boldsymbol{\tau} = -\mathbf{J}_{aa}^T\mathbf{F}^* + \mathbf{N}_{a,b}\boldsymbol{\tau}_j. \tag{17}$$

The null space projection ensures that the base and joint motion tracking components of $\boldsymbol{\tau}$ are always orthogonal. This can be verified by substituting the value of $\boldsymbol{\tau}$ from (17) in (15) to calculate the effective base wrench, neglecting the contact constraints, as shown

$$\begin{aligned}
\hat{\boldsymbol{\tau}}_b &= (\mathbf{J}_{a,b})^{\dagger}\boldsymbol{\tau} \\
&= (\mathbf{J}_{a,b})^{\dagger}(-\mathbf{J}_{aa,}^T\mathbf{F}^* + \mathbf{N}_{a,b}\boldsymbol{\tau}_j) \\
&= (\mathbf{J}_{a,b})^{\dagger}\mathbf{J}_{a,b}\boldsymbol{\tau}_b + (\mathbf{J}_{a,b})^{\dagger}\mathbf{N}_{a,b}\boldsymbol{\tau}_j,
\end{aligned}$$

where the first term is the projection of $\boldsymbol{\tau}_b$ into the column space of $\mathbf{J}_{a,b}$. The second term simplifies to zero using the definition of the generalized inverse [9] that for any matrix $\mathbf{A}$, a valid inverse $\mathbf{G}$ satisfies $\mathbf{AGA} = \mathbf{A} \Leftrightarrow \mathbf{GAG} = \mathbf{G}$. In our case, $(\mathbf{J}_{a,b})^{\dagger}\mathbf{J}_{a,b}(\mathbf{J}_{a,b})^{\dagger} = (\mathbf{J}_{a,b})^{\dagger}$.

$$\begin{aligned}
(\mathbf{J}_{a,b})^{\dagger}\mathbf{N}_{a,b} &= (\mathbf{J}_{a,b})^{\dagger}(\mathbf{I}_{12\times 12} - \mathbf{J}_{a,b}(\mathbf{J}_{a,b})^{\dagger}) \\
&= (\mathbf{J}_{a,b})^{\dagger} - (\mathbf{J}_{a,b})^{\dagger}\mathbf{J}_{a,b}(\mathbf{J}_{a,b})^{\dagger} \\
&= (\mathbf{J}_{a,b})^{\dagger} - (\mathbf{J}_{a,b})^{\dagger} \\
&= \mathbf{0}.
\end{aligned}$$

This shows that both the objectives of base and joint tracking are satisfied independently and simultaneously. This is similar to the hierarchical control formulation [24] if we consider the base control as the higher priority task and joint tracking as a lower priority task.

We must take note that the above holds only until the system is fully- (or over-) constrained, which in the case of quadruped robots implies that at least two feet are in contact with the ground. This ensures that the matrix $\mathbf{J}_{ab,c}^T$ is always of full row rank. If more than two feet are in contact with the ground ($n_c > 2; \mathbf{J}_{ab,c}^T \in \mathbb{R}^{6\times(6+m)}; m \in \{3, 6\}$), then there are infinite solutions to (9) and the final solution of the QP is the one that minimizes the cost function and satisfies the contact constraints. If two feet are in contact with the ground ($n_c = 2; \mathbf{J}_{ab,c}^T \in \mathbb{R}^{6\times6}$), there exists exactly one solution to (9). The QP, in that case, gives us an estimate of the solution that satisfies the contact constraints and is closest to the analytical solution of (9). If less than two feet are in contact with the ground ($n_c < 2; \mathbf{J}_{ab,c}^T \in \mathbb{R}^{6\times(6-m)}; m \in \{3, 6\}$), there exists no exact solution for (9), which implies the solution of the QP will satisfy the original equation in a least-squared error sense with a non-zero residual while satisfying the contact constraints. Note that for $n_c \geq 2$, $rank(\mathbf{J}_{ab,c}^T) = 6$, while for $n_c < 2$, $rank(\mathbf{J}_{ab,c}^T) \in \{3, 0\}$.

In this work, we assume that the jacobian matrices involved in the computations of the controller never become row-rank deficient or singular. The row rank sufficiency required for $\mathbf{J}_{ab,c}^T$ is ensured by the choice of gait, and non-singularity of $\mathbf{J}_{aa,c}^T$ by choosing the appropriate step length in the motion planner generating the base and foot trajectories, and is explained in Section 5.

4 Optimization Methodology

To solve the optimization problem (11), we reformulate the objective of the optimization problem in the standard quadratic form, as shown below

$$\begin{aligned} \arg\min_{\mathbf{f}} \ & \mathbf{f}^T\mathbf{Q}\mathbf{f} + \mathbf{P}\mathbf{f} + \mathbf{R} \\ \text{s.t. } & \mathbf{g}(\mathbf{f}) \leq \mathbf{0}, \end{aligned} \tag{18}$$

with the following definitions,

$$\begin{aligned} \mathbf{f} &= \mathbf{F}_c, \\ \mathbf{Q} &\triangleq \mathbf{J}_{ab,c}\mathbf{S}_1\mathbf{J}_{ab,c}^T + \mathbf{W} + \mathbf{V}, \\ \mathbf{P} &\triangleq -2(\boldsymbol{\tau}_b^T\mathbf{S}_1\mathbf{J}_{ab,c}^T + \mathbf{f}_{prev}^{*T}\mathbf{V}), \\ \mathbf{R} &\triangleq \boldsymbol{\tau_b}^T\mathbf{S}_1\boldsymbol{\tau_b} + \mathbf{f}_{prev}^{*T}\mathbf{V}\mathbf{f}_{prev}^*, \\ \mathcal{J} &\triangleq \mathbf{f}^T\mathbf{Q}\mathbf{f} + \mathbf{P}\mathbf{f} + \mathbf{R}, \end{aligned}$$

where $\mathbf{f}_{prev}^*$ is the solution of the problem from the previous solution. The constraint vector, $\mathbf{g}(\mathbf{f})$, consists of the unilateral normal force and the friction cone constraints. The friction cone constraint makes $\mathbf{g}(\mathbf{f})$ nonlinear. This makes the optimization problem in (18) difficult to solve. The general approach in the existing literature is to linearize this constraint by approximating the friction cone by a friction pyramid. This makes it possible to use off-the-shelf QP solvers like qpOASES, HPIPM, etc. Since we have a quadratic cost with a completely geometric constraint, i.e., the contact force

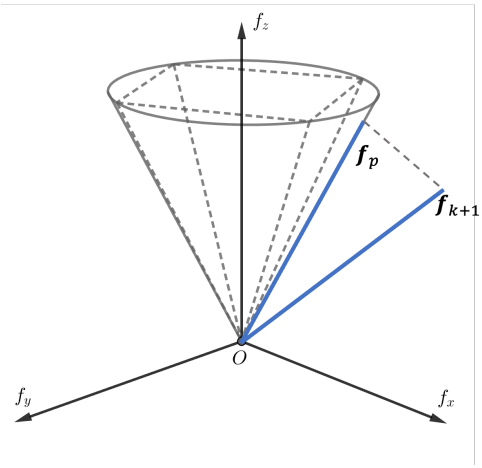


Fig. 1 Projection of the iterative solution on the friction cone.

should remain inside the friction cone, we propose an algorithm to solve this without linearizing the constraints, named Geometric Projected Gradient Descent (GPGD). In this approach, we use gradient descent to find the solution to the optimization problem and manually enforce the constraint by projecting the resulting forces on each leg that is in contact with the ground onto the friction cone. The proposed algorithm solves the problem with the original nonlinear constraints without any simplification.

The gradient and the Hessian used in a single step of the optimization algorithm are

$$\begin{aligned}\nabla \mathcal{J} &= (\mathbf{Q} + \mathbf{Q}^T)\mathbf{f} + \mathbf{P}^T \\ &= 2\mathbf{Q}\mathbf{f} + \mathbf{P}^T \quad (\text{since } \mathbf{Q} = \mathbf{Q}^T), \\ \nabla^2 \mathcal{J} &= 2\mathbf{Q}.\end{aligned}$$

The projection step in the proposed algorithm is inspired by the algorithm given in [33]. It is simplified and shown in Fig. 1. The proposed algorithm is given in the block marked **Algorithm 1**. In the algorithm, $\mathbf{f}_{n,i}$ denotes the contact force at iteration n for i-th foot in contact with the ground.

5 Planning and Estimation

5.1 Planning

The planner in the planning-estimation-control (PEC) loop computes the desired positions of the robot base and feet. The input to the planner is the commanded velocity, ${}^{\mathcal{B}}\mathbf{v}_{cmd} = \begin{bmatrix} v^x & v^y & \omega^z \end{bmatrix}^T$, generated by scaling the joystick commands with appropriate velocity limits. This is used to compute the desired translational base motion along $X^{\mathcal{G}}$ and $Y^{\mathcal{G}}$ axes, rotational base motion about $Z^{\mathcal{G}}$ axis, and the stepping location for the swing legs at each time-step k of the PEC loop.

Algorithm 1 Geometric Projected Gradient Descent

function GPGD( )
  Initialize: $n \leftarrow 0$, $\mathbf{f}_n \leftarrow \mathbf{F}^*_{c_k}$, $\mathbf{f}_p \leftarrow \mathbf{0}$
  while $n <$ max_iters **do**
    $\nabla\mathcal{J} \leftarrow \mathbf{Q}\mathbf{f}_n + \mathbf{P}^T$ ▷ update gradient
    $\alpha \leftarrow$ line-search()
    $\mathbf{f}_{n+1} \leftarrow \mathbf{f}_n - \alpha\nabla\mathcal{J}$ ▷ GD step
    for $i \in \{1, \ldots, n_c\}$ **do**
      $f^z_{n+1,i} \leftarrow \max\{f^z_{n+1,i}, f^z_{\min}\}$
      $f^z_{n+1,i} \leftarrow \min\{f^z_{n+1,i}, f^z_{\max}\}$ ▷ unilaterality
      $f_t \leftarrow \|\mathbf{f}^{x,y}_{n+1,i}\|$
      $f_z \leftarrow f^z_{n+1,i}$
      if $f_t \leq \mu f_z$ **then** ▷ inside friction cone
        $\mathbf{f}_{p,i} \leftarrow \mathbf{f}_{n+1,i}$
      else if $f_t \leq -\frac{f_z}{\mu}$ **then** ▷ inside dual cone
        $\mathbf{f}_{p,i} \leftarrow \mathbf{0}$
      else ▷ project on friction cone surface
        $f^z_{p,i} \leftarrow \frac{\mu f_t + f_z}{\mu^2+1}$
        $f^z_{p,i} \leftarrow \min\{\max\{f^z_{p,i}, f^z_{\min}\}, f^z_{\max}\}$
        $\mathbf{f}^{x,y}_{p,i} \leftarrow \mu f^z_{p,i} \frac{\mathbf{f}^{x,y}_{n+1,i}}{f_t}$
      end if
    end for
    if $\|\mathbf{f}_p - \mathbf{f}_n\|_2 < 10^{-2}$ **then** ▷ check convergence
      return $\mathbf{f}_p$ ▷ return converged solution
    end if
    $n \leftarrow n + 1$ ▷ update iteration counter
  end while
  return $\mathbf{f}_p$ ▷ return sub-optimal solution
end function

5.1.1 Base motion planning

The required linear acceleration of the base in frame $\mathcal{B}$ is given by

$$ {}^{\mathcal{B}}\ddot{\mathbf{r}}^{x,y}_{b_k} = K_b({}^{\mathcal{B}}\mathbf{v}^{x,y}_{cmd,k} - {}^{\mathcal{B}}\dot{\mathbf{r}}^{x,y}_{b_{d,k-1}}), \tag{19} $$

where $\dot{\mathbf{r}}^{x,y}_{b_{d,k-1}}$ is the desired base velocity at the previous time-step. The required base linear acceleration is saturated using the maximum allowed acceleration $\mathbf{a}_{\max}$ to get the desired acceleration and then used to update the desired base velocity $\dot{\mathbf{r}}^{x,y}_{b_{d,k}}$ and

position $\mathbf{r}^{x,y}_{b_{d,k}}$ as shown in (20).

$$
\begin{aligned}
{}^{\mathcal{B}}\ddot{\mathbf{r}}^{x,y}_{b_{d,k}} &= \min\{\max\{{}^{\mathcal{B}}\ddot{\mathbf{r}}^{x,y}_{b_k}, -{}^{\mathcal{B}}\mathbf{a}^{x,y}_{\max}\},\ {}^{\mathcal{B}}\mathbf{a}^{x,y}_{\max}\}, \\
{}^{\mathcal{G}}\dot{\mathbf{r}}^{x,y}_{b_{d,k}} &= {}^{\mathcal{G}}\dot{\mathbf{r}}^{x,y}_{b_{d,k-1}} + {}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}\ {}^{\mathcal{B}}\ddot{\mathbf{r}}^{x,y}_{b_{d,k}}\Delta t, \\
{}^{\mathcal{G}}\mathbf{r}^{x,y}_{b_{d,k}} &= {}^{\mathcal{G}}\mathbf{r}^{x,y}_{b_{d,k-1}} + {}^{\mathcal{G}}\dot{\mathbf{r}}^{x,y}_{b_{d,k}}\Delta t + \frac{1}{2}\,{}^{\mathcal{G}}\ddot{\mathbf{r}}^{x,y}_{b_{d,k}}\Delta t^2,
\end{aligned}
\tag{20}
$$

where ${}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}$ is the rotation matrix representing the transformation from the local to the global frame, and Δt is the difference between the current time and the time at the end of the previous iteration. The maximum velocity values are chosen based on the base height and the gait frequency. The base height imposes a kinematic constraint on the maximum step length of the robot. For acceleration limits, we multiply $F^z_{c,\max}$ by the friction coefficient to get the maximum horizontal force, then divide it by the total mass of the robot to get the maximum possible acceleration. K_b is the gain chosen based on the desired first-order response of the velocity. Similarly, the desired yaw motion is calculated as

$$
\begin{aligned}
\ddot{\theta}^z_{d,k} &= \min\{\max\{K_\theta(\omega^z_k - \dot{\theta}^z_{d,k-1}), -\alpha^z_{\max}\}, \alpha^z_{\max}\}, \\
\dot{\theta}^z_{d,k} &= \min\{\max\{\dot{\theta}^z_{d,k-1} + \ddot{\theta}^z_{d,k}\Delta t, -\omega^z_{\max}\}, \omega^z_{\max}\}, \\
\theta^z_{d,k} &= \theta^z_{d,k-1} + \dot{\theta}^z_{d,k}\Delta t + \frac{1}{2}\ddot{\theta}^z_{d,k}\Delta t^2,
\end{aligned}
$$

where θ^z is the rotation angle about the $Z^{\mathcal{G}}$ axis, $\alpha^z_{\max}$ and $\omega^z_{\max}$ are the fixed maximum angular acceleration and velocity about the $Z^{\mathcal{G}}$ axis, respectively. This desired angle is converted to the specific orientation representation being used before being passed to the controller for tracking.

5.1.2 Foothold planning

To plan the footholds, a desired gait is predefined that specifies when each leg will be in the swing phase or the stance phase. Inspired by the Hildebrand gait parameters [19], we define a gait using four variables:

a) gait period t_g,
b) duty factor $\phi_{st,i}$,
c) phase offset $\psi_{o,i}$, and
d) step height h_z.

The variables with subscript i are defined for each leg individually. The duty factor $\phi_{st,i}$ is the fraction of the normalized gait period after which the contact state of the i-th leg changes from stance to swing. The phase offset $\psi_{o,i}$ is the normalized timing offset of each leg. $\psi_{o,i}$ along with $\phi_{st,i}$ define the type of gait the robot will execute (walk, trot, bound, etc.). The desired contact state $s_{d_{i,k}}$ is calculated as

$$
\phi_{ph,i} = \mod(t/t_g + \psi_{o,i}, 1),
$$

$$s_{d_{i,k}} = \begin{cases} 1 & \text{if } \frac{\phi_{ph,i}}{\phi_{st}} < 1 \\ 0 & \text{otherwise} \end{cases},$$

where $s_{d_{i,k}} = 1$ corresponds to stance and $s_{d_{i,k}} = 0$ to swing. For each foot in swing, its landing position $\mathbf{r}^{x,y}_{p_{i,d,k}}$ is calculated using the following foot placement heuristic

$$\begin{aligned} {}^{\mathcal{G}}\mathbf{r}^{x,y}_{p_{i,d,k}} &= {}^{\mathcal{G}}\mathbf{r}^{x,y}_{h_{i,d,k}} + {}^{\mathcal{G}}\mathbf{d}^{x,y}_{i}, \\ {}^{\mathcal{G}}\mathbf{d}^{x,y}_{i} &= \frac{1}{2}({}^{\mathcal{G}}\dot{\mathbf{r}}^{x,y}_{h_{i,d,k}} t_{st,i} + {}^{\mathcal{G}}\dot{\mathbf{r}}^{x,y}_{h_{i,d,k}} (1-\phi_{ph,i}) t_{sw,i}), \end{aligned} \tag{21}$$

where $\mathbf{r}^{x,y}_{h_{i,d,k}}$ is the position of the i-th hip, and $\mathbf{d}^{x,y}_{i}$ is the position of the desired foothold from the current hip position, calculated using the Raibert heuristic [27]. $t_{st,i} = \phi_{st,i} t_g$ and $t_{sw,i} = (1-\phi_{st,i}) t_g$ are the stance and swing durations for one gait cycle. The norm of $\mathbf{d}^{x,y}_{i}$ is the step length for the i-th leg in the horizontal plane.

For flat ground, ${}^{\mathcal{G}}\mathbf{r}^{z}_{p_{i,d,k}} = 0$ since the terrain height is uniform. For planning on uneven terrain, a 3D plane is fitted using the estimated positions of the stance legs using the algorithm described in [3]. The Z-coordinate on the plane is assumed to be zero at the current horizontal estimated position of the base. This fitted 3D plane is used as an approximate height map of the terrain. The target foot height at the landing location for swing legs is then calculated as:

$${}^{\mathcal{G}}r^{z}_{p_i,d,k} = h({}^{\mathcal{G}}\mathbf{r}^{x,y}_{p_i,d,k}) \tag{22}$$

This is used during the uneven terrain tests in Section 6.

The position and velocity of the i-th hip are given by

$$\begin{aligned} {}^{\mathcal{G}}\mathbf{r}_{h_i} &= {}^{\mathcal{G}}\mathbf{r}_{b} + {}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}\, {}^{\mathcal{B}}\mathbf{r}_{h_i/b}, \\ {}^{\mathcal{G}}\dot{\mathbf{r}}_{h_i} &= {}^{\mathcal{G}}\dot{\mathbf{r}}_{b} + {}^{\mathcal{G}}\boldsymbol{\omega}_{b} \times {}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}\, {}^{\mathcal{B}}\mathbf{r}_{h_i/b}, \end{aligned} \tag{23}$$

where $\mathbf{r}_{h_i/b}$ is the vector from the COM to the i-th hip and $\boldsymbol{\omega}_b$ is the angular velocity of the base. This swing foot trajectory is generated using second-order Bezier curves fitted using the takeoff and landing positions of the feet calculated above.

5.2 State estimation

The full state estimation pipeline consists of base pose and contact state estimators that are interdependent.

5.2.1 Base pose estimation

The robot base pose (position and orientation) and its derivative (linear and angular velocity) are estimated using the IMU and joint encoder data. The framework used here is based on the formulation proposed in [5] and [3]. The acceleration and gyroscope data from the IMU are used for the prediction model of the Kalman filter. The

orientation data is taken directly from the IMU. For the correction step of the estimator, a no-slip condition is assumed on the stance feet to update the robot base linear position and velocity.

5.2.2 Contact estimation

For contact estimation, the measured torques from the motors are used to first estimate the contact forces on the feet. The EOM described in (1) is used for estimating foot forces as

$$\hat{\mathbf{F}}_c = -(\mathbf{J}_c^T(\hat{\mathbf{q}}))^{\dagger}(\mathbf{S}^T\hat{\boldsymbol{\tau}} - \mathbf{M}(\hat{\mathbf{q}})\hat{\dot{\mathbf{q}}} - \boldsymbol{\eta}(\hat{\mathbf{q}}, \hat{\dot{\mathbf{q}}})),$$

where $\hat{(\cdot)}$ represents the estimated quantities or filtered sensor data. The estimated generalized coordinate vector $\hat{\mathbf{q}}$ and generalized velocity vector $\hat{\dot{\mathbf{q}}}$ are constructed by combining the output of the base pose estimator and filtered sensor values. The estimated generalized acceleration vector, $\hat{\ddot{\mathbf{q}}}$, is calculated by using a filtered finite-difference [30] of the estimated generalized velocity vector $\hat{\dot{\mathbf{q}}}$. The estimated contact force, $\hat{\mathbf{F}}_c$, coupled with the estimated feet height and reference contact phase as per the gait scheduler, is used to estimate the contact probabilities $P_{c_{i,k}}$ using a Kalman filter-based approach described in [4]. The estimated contact state $s_{i,k}$ is computed using hysteresis thresholding on $P_{c_{i,k}}$ to make the contact state prediction more stable.

$$s_{i,k} = \begin{cases} 1 & \text{if } P_{c_{i,k}} \geq 0.6 \\ 0 & \text{if } P_{c_{i,k}} \leq 0.4 \\ s_{i,k-1} & \text{else} \end{cases},$$

where $P_{c_{i,k}}$ is the contact probability and $s_{i,k}$ is the contact state for the leg i for the current time-step k.

6 Experiments and Results

The proposed controller is tested and validated on the Go2 quadruped robot from Unitree Robotics. The simulation experiments were conducted in the MuJoCo [34] simulator running on a desktop system with an AMD Ryzen 9 3900XT CPU and an NVIDIA RTX 3060 GPU. The software architecture used for the experiments is shown in Fig. 2. For the communication interface, CycloneDDS [15] was chosen as the primary backend. The controller block in Fig. 2 is expanded in Fig. 3 to show further details about how joint commands being sent to the motor controller are generated.

6.1 Solver performance evaluation

To evaluate the proposed GPGD solver for the optimization problem 11, we perform Monte Carlo simulations for a representative problem and compare the results between different standard off-the-shelf solvers: qpOASES [13], HPIPM [16], Proxsuite/ProxQP [2], and Quadprog [2]. The contact force solver benchmark task is defined around the

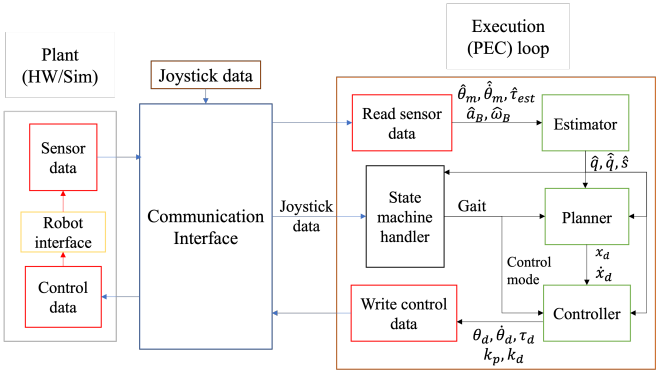


Fig. 2 The architecture of the locomotion software and its interface with the simulation/hardware.

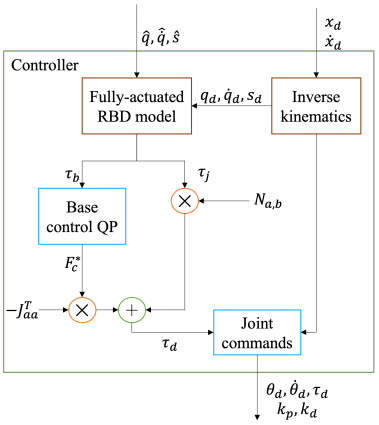


Fig. 3 The architecture of the controller block of the locomotion software shown in Figure 2.

nominal stance configuration of the robot. In the nominal configuration, the positions of the feet with respect to the base are chosen to be

$$\mathbf{r}_{p_i/b} = \mathbf{r}_{p_i/b_{nominal}} + \boldsymbol{\eta}_p. \tag{24}$$

where $\eta_{p,l} \sim \mathcal{N}(0, 0.1) \forall l \in \{x, y, z\}$ is the random variation of foot position for the experiment. $\mathbf{r}_{p_i/b_{nominal}} = \begin{bmatrix} s_{x,i}0.25 & s_{y,i}0.15 & -0.3 \end{bmatrix}^T$ is the nominal position of the i-th foot in stance configuration with respect to the base. Here, $s_x = [1, 1, -1, -1]$ and $s_y = [1, -1, 1, -1]$ are the axes sign indicator for the respective legs. This is used to construct the matrix $\mathbf{J}_{ab,c}^T$ as described in 10. The desired base wrench for this experiment is chosen as:

$$\boldsymbol{\tau}_b = \begin{bmatrix} 0 & 0 & mg & 0 & 0 & 0 \end{bmatrix}^T + \boldsymbol{\eta}_b, \tag{25}$$

where $\boldsymbol{\eta}_b \in \mathbb{R}^6$ with each $\eta_{b,l} \sim \mathcal{U}(-20, 20), \forall l \in 1, 2, ...6$, is the random force/torque to emulate the different wrench requirements on the robot base in varying scenarios.

The residual is calculated as follows:

$$r_b = ||\mathbf{J}_{ab,c}^T \mathbf{F}_c^* - \boldsymbol{\tau}_b||_2, \tag{26}$$

where $\mathbf{F}_c^*$ is the vector of optimized contact forces on the feet of the robot. The relative weighing and the regularization matrices used in this benchmark are kept constant:

$$\begin{aligned} \mathbf{S}_1 &= diag(1, 2, 1, 120, 120, 1), \\ \mathbf{W}_{i,i} &= 0.0001; \mathbf{W}_{i,j} = 0 \forall i \neq j, \\ \mathbf{V}_{i,i} &= 0.001; \mathbf{V}_{i,j} = 0 \forall i \neq j. \end{aligned} \tag{27}$$

Figure 4 shows the comparison of the residual and solve time statistics between the different solvers tested here.

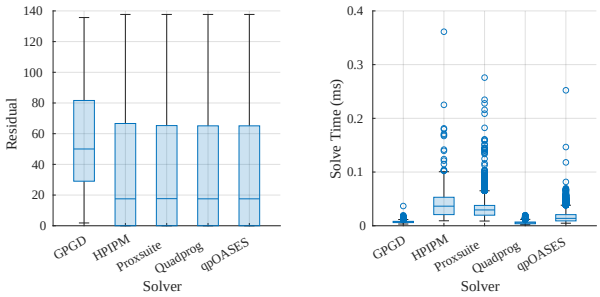


Fig. 4 Residual and solve time statistics for different solvers for the contact force optimization benchmark task.

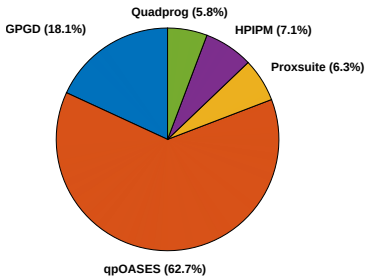


Fig. 6 Distribution of the solvers with the least residual out of the 50K random problems from the contact force optimization benchmark task.

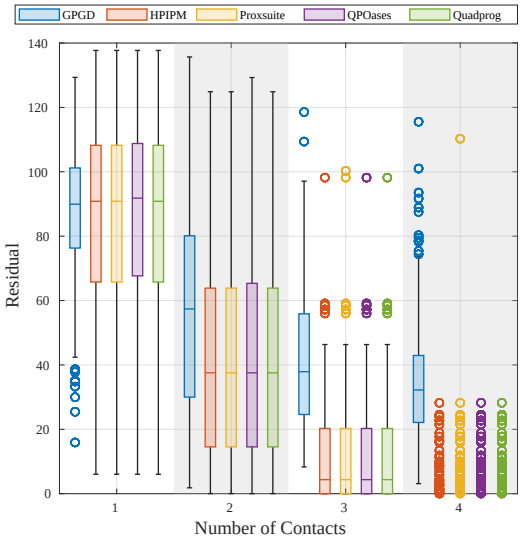


Fig. 5 Residual statistics for different solvers for the different number of feet in contact.

In Figure 4 and Figure 5, the GPGD solver has a higher mean residual than the other solvers. However, it is important to understand that comparing the mean residuals of the solvers does not give a complete understanding of the actual performance of the solvers. The reader may be tempted to conclude that the GPGD solver is the least preferred if residual minimization is a priority, but a more accurate test would be to individually count the number of times when each solver gives the minimum residual out of all the solvers, out of all the problems in this randomized test. The results of such a test are shown in Figure 6. The data in this shows that for residual minimization, GPGD (18.1%) is the second-best solver after qpOASES (62.7%) in the entire problem space. So, even when the GPGD solver seems to have relatively worse performance when aggregate statistics are compared, it is actually the second-best solver in the entire problem space for residual minimization. Since the benchmark optimization problem is solved for configurations around the nominal stance configuration 24, the

solutions found are for configurations that the robot is very likely to achieve during motion. Also, in the authors' experience, qpOASES has issues with handling equality constraints and has inconsistent solve times.

The comparison of the solve time between solvers for the same set of problems is shown in Figure 4. In this, GPGD and Quadprog solvers have consistent solve times, while the other solvers have long tails.

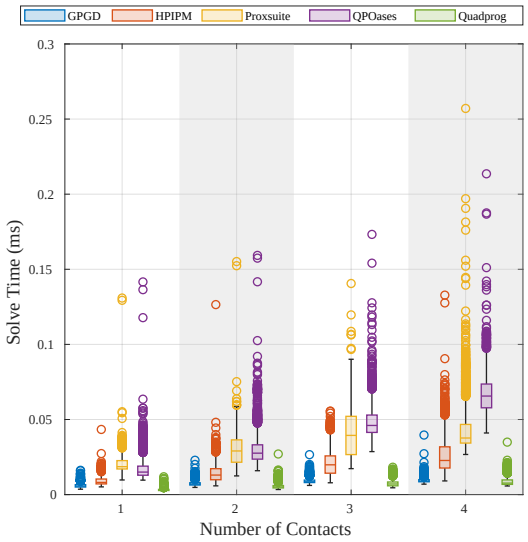


Fig. 7 Solve time statistics for different solvers for different numbers of feet in contact.

This can also be seen across different sizes of the problems (due to varying numbers of feet in contact) in Figure 7. Here also, GPGD and Quadprog seem to have the most consistent timings compared to the other solvers.

For effective control performance for resource constrained systems, a fast solver with good enough accuracy is required. Given the results presented above, the GPGD solver emerges as a strong choice for the contact force optimization problem, with consistent performance in both residual minimization and solve times.

6.2 Controller performance evaluation

The proposed controller, DIDC, is compared against the standard Balance Controller (BC) and the Whole Body Controller (WBC) described in Section 1.

6.2.1 Simulation results

In the simulation environment, rate limiting and latency are added to make it similar to the actual hardware system. The original update rate of the simulation is kept at 1000 Hz. We update the control and sensor data objects at 500 Hz. To introduce

latency in the simulation, we first make a probabilistic map of the latency times in bins of 5ms. While updating the control commands, we first push the incoming data at the start of a queue. We then use the probability map of the system latency to calculate the index for the control command to be used from the queue at the current time step. The gains chosen during the simulation experiment are:

$$\begin{aligned}
\mathbf{S}_1 &= diag(1, 2, 5, 120, 120, 1), \\
\mathbf{W}_{i,i} &= 0.00001; \mathbf{W}_{i,j} = 0 \forall i \neq j, \\
\mathbf{V}_{i,i} &= 0.001; \mathbf{V}_{i,j} = 0 \forall i \neq j, \\
\mathbf{K}_{p_{i,i}} &= \mathbf{M}_{i,i} \omega_{n,i}^2 \\
\mathbf{K}_{d_{i,i}} &= 2\gamma \sqrt{\mathbf{K}_{p_{i,i}}}; \gamma = 1.1 \\
\mathbf{K}_{d_{i,i}} &= diag(25, 25, 2, 0.4, 0.2, 20) \forall i \in \{1, 2, ..., 6\}
\end{aligned} \tag{28}$$

where $\omega_{n,i}$ is the desired natural response frequency of i-th DoF. Additional joint-level gains used during this experiment were:

$$\begin{aligned}
\mathbf{K}_{p,j,st} &= 3; \mathbf{K}_{d,j,st} = 1.8, \\
\mathbf{K}_{p,j,sw} &= 8; \mathbf{K}_{d,j,sw} = 0.8,
\end{aligned} \tag{29}$$

where $\mathbf{K}_{p,j,st}$ and $\mathbf{K}_{p,j,sw}$ represent gains for the stance and swing leg joints, respectively.

6.2.2 Straight line locomotion performance on flat ground

Each controller is given a fixed set of commands for the following scenarios:

1. longitudinal (along $X^{\mathcal{B}}$ axis): Increasing velocity from 0.2 m/s to 1.0 m/s in steps of 0.2 m/s, and
2. lateral (along $Y^{\mathcal{B}}$ axis): Increasing the velocity from 0.1 m/s to 0.6 m/s in steps of 0.1 m/s.

Each velocity step is held for 5 seconds, and each experiment is repeated 10 times. During the entire duration of the experiment, the velocity error, the orientation error ($\Delta\boldsymbol{\phi}$), the energy (E), and the cost-of-transport (CoT) are recorded. These values are calculated as shown below:

$$\begin{aligned}
\Delta\mathbf{v} &= \sum_{\forall k} (\dot{\mathbf{r}}_{b,d,k} - \hat{\dot{\mathbf{r}}}_{b,k}), \\
\Delta\boldsymbol{\Phi} &= \sum_{\forall k} \log(\, {}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}_{d,k}\, {}^{\mathcal{G}}_{\mathcal{B}}\hat{\mathbf{R}}_k^T)^{\vee}, \\
E &= \sum_{\forall k} \hat{\tau}_{i,k} \hat{\dot{\theta}}_{i,k} \qquad \forall i \in 1, 2, 3..., 12,
\end{aligned}$$

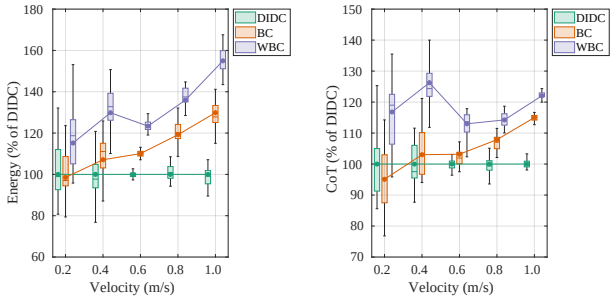


Fig. 8 Normalized energy and CoT for increasing velocity in the longitudinal direction in simulation.

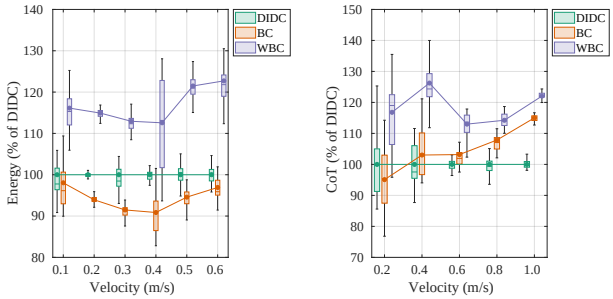


Fig. 9 Normalized energy and CoT for increasing velocity in the lateral direction in simulation.

$$CoT = \frac{E}{mgd}.$$

Here, $d = \sum_{\forall k} \hat{\mathbf{r}}_{b,k}$ is the total distance traveled by the robot during the experiment. It is calculated by integrating the estimated velocity of the robot. $\forall k$ denotes all the indices of the data recorded during one run of a particular experiment.

Figures 8 and 9 show the variation of energy and CoT for each controller with increasing velocities. It can be seen in the Figure 8 that the CoT is lower when the DIDC controller is used for higher velocities. This is due to the increasing effect of leg movement on the net feedback wrench required on the torso, which is better compensated, and hence, lesser, in DIDC. Even with a lower average torque for BC, as seen in Figure 10, the energy consumption and CoT are higher as compared to DIDC for straight line motion at higher velocities (> 0.6 m/s).

In the sideways motion, the energy consumption seems significantly lower for BC as compared to DIDC, but the overall CoT is similar for both. This is because of the larger distance covered by the robot while using the DIDC controller, thus producing higher torques and leading to higher energy consumption. This is due to better velocity tracking in the lateral direction, as can be seen in Figure 11. If the BC gains were tweaked to achieve similar velocity tracking performance, the energy consumption and hence the CoT would be significantly higher. WBC has the highest energy consumption in all cases out of all the controllers due to higher torque requirements for satisfying

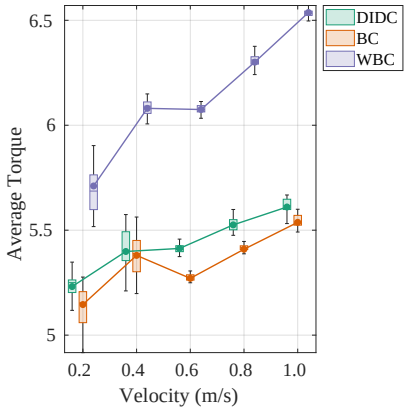


Fig. 10 RMS torque for straight direction experiments in simulation.

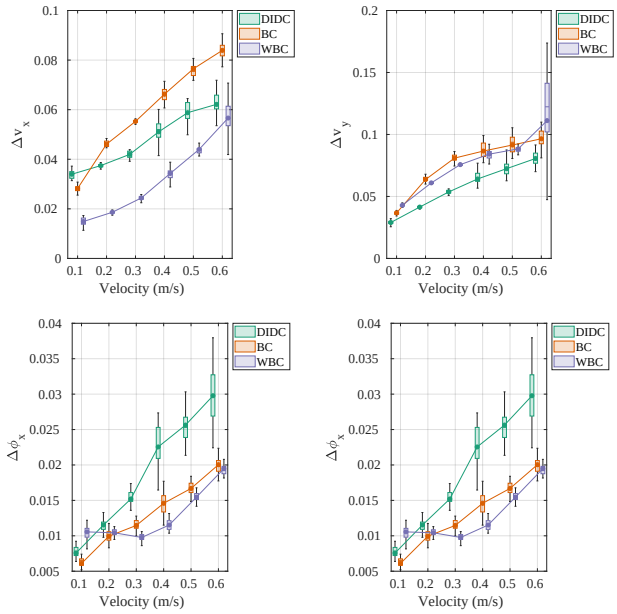


Fig. 11 Velocity and orientation tracking error for sideways motion experiments in simulation.

multiple tasks. The CoT for WBC is lower for a certain range of velocities in the longitudinal direction because of better velocity tracking. Better velocity tracking leads to a larger total distance covered, hence lowering the CoT. The velocity tracking in the longitudinal direction is better with WBC as can be seen in Figure 12.

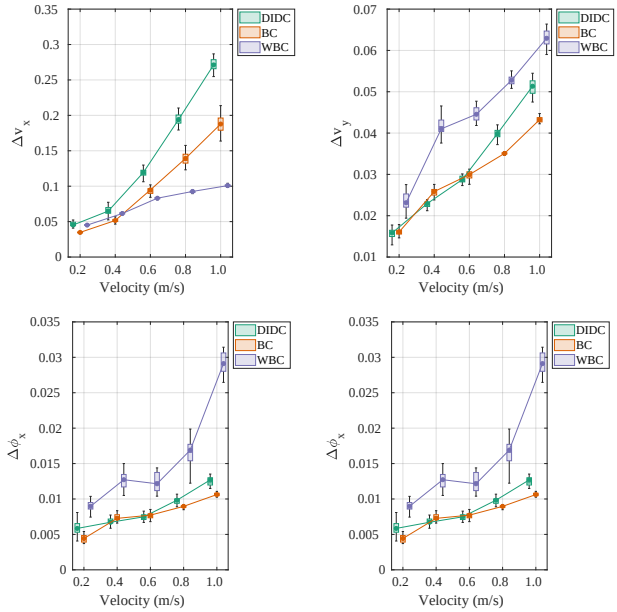


Fig. 12 Velocity and orientation tracking error for straight motion experiments in simulation.

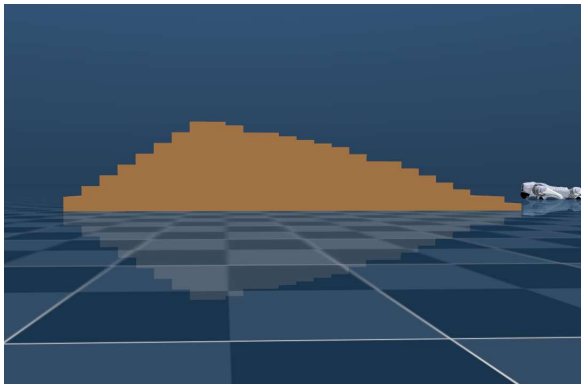


Fig. 13 Uneven terrain used for testing controllers in a simulation environment.

6.2.3 Crossing uneven terrain with stairs

To test performance on uneven terrain and stairs, a simulation environment was created with a staircase-like obstacle as shown in Figure 13. The height of steps in the terrain varies from 2-5 cm along the incline (to the right of the figure) to 6-8 cm along the decline (to the left of the figure). To ensure consistent testing across all three controllers, the robot was spawned in front of this terrain on flat ground and given a commanded velocity of 0.2 m/s in the forward direction along the robot base for 40 seconds. This was repeated 10 times for each controller. The average CoT results are

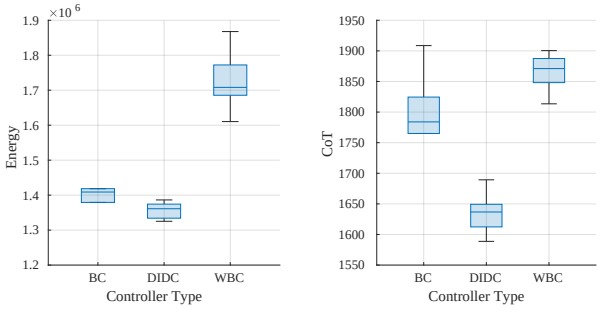


Fig. 14 Energy and CoT results for uneven terrain testing in simulation.

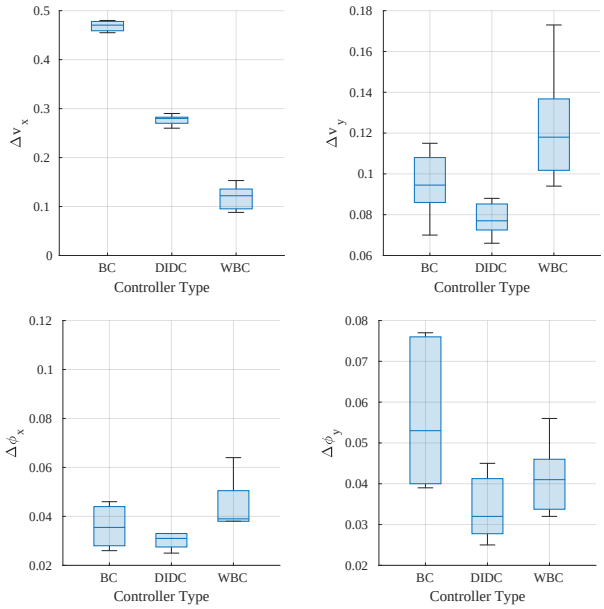


Fig. 15 Velocity and orientation tracking error for uneven terrain testing in simulation.

shown in Figure 14. Similar to earlier results on flat ground, DIDC is the most energy-efficient controller on uneven terrain, while WBC consumes the most energy and has the highest CoT.

6.3 Hardware results

The robot used for hardware testing was the Unitree Go2 robot with an integrated NVIDIA Orin NX compute module on which we run the control pipeline. For hardware testing, only linear motion was tested for all three controllers. The experiment routine was as follows:

1. Move along $+X^{\mathcal{G}}$ axis at 1 m/s for 4 seconds

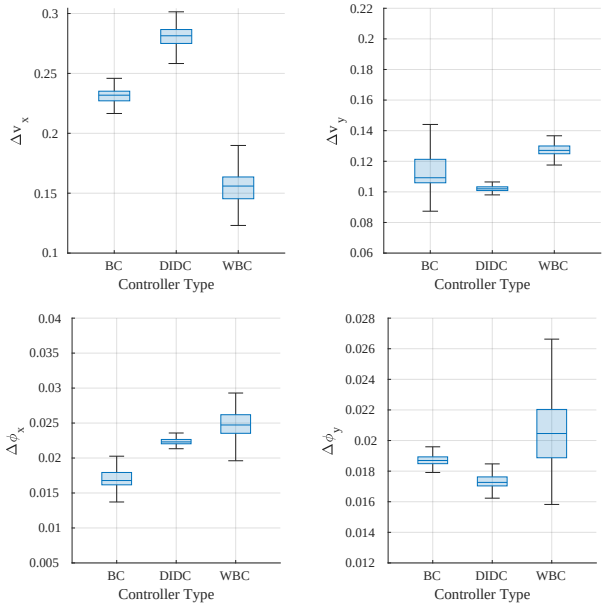


Fig. 16 Velocity and orientation tracking results for straight line experiment on hardware.

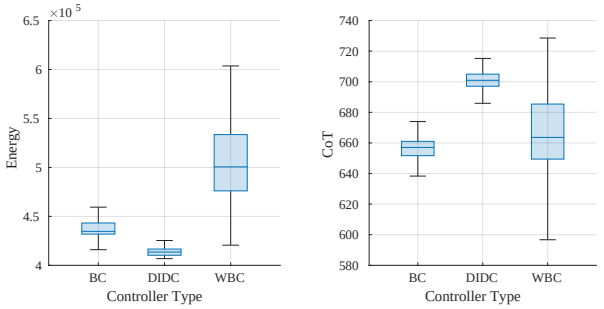


Fig. 17 Energy and CoT for straight line experiment on hardware.

2. Stop and trot on the spot for 2 seconds
3. Move along $-X^{\mathcal{G}}$ direction at 1 m/s for 4 seconds
4. Stop and trot on the spot for 2 seconds

The joint-level gains were changed for the hardware experiments, keeping all other gains similar to the simulation experiments. The final tuned joint gains used during the experiment were:

$$\begin{aligned} \mathbf{K}_{p,j,st} &= 10; \mathbf{K}_{d,j,st} = 2.8, \\ \mathbf{K}_{p,j,sw} &= 30; \mathbf{K}_{d,j,sw} = 1.8. \end{aligned} \tag{30}$$

The velocity and orientation tracking results of this can be seen in Figure 16. The energy consumption and CoT are shown in Figure 17. The DIDC controller still

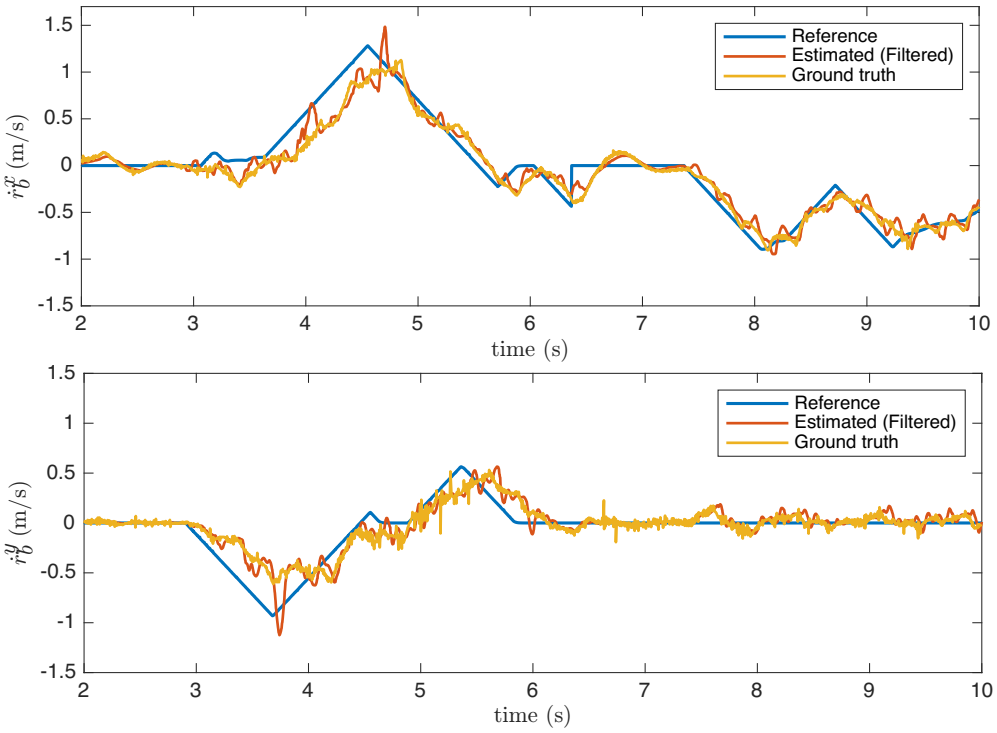


Fig. 18 Comparison of estimated base velocity against ground truth data on hardware.

has lower energy consumption for the experiment on hardware, but the overall CoT was higher compared to DIDC. This is again attributed to better velocity tracking in the case of the BC controller. The WBC controller has the highest energy usage in hardware experiments and has better longitudinal velocity tracking, in line with the simulation results.

To validate the accuracy of the EKF-based state-estimator used here, the estimates were compared against the ground-truth data from a motion capture setup. The state estimator is able to estimate the speed of the robot with a mean absolute error of ~ 0.066m/s and a mean error of ~ 0.00044 m/s, the comparison of which is shown in Figure 18. This shows that the estimator has an accurate mean estimate and can be used to obtain reliable velocity data for the controller.

7 Conclusion

In this work, we have proposed a reactive controller that uses the full-RBD model of a quadruped robot and exact friction cone constraints to compute the required joint torque command, while being computationally lightweight. We achieve this by avoiding the dependence on general-purpose QP solvers and proposing a custom solver that exploits the geometry of the friction constraints. We compare the performance of our controller with QP-based balance controller and Weighted Whole-Body Controller

in omnidirectional mobility routines on flat and uneven terrain, in simulations, and on flat terrain on hardware. The results reveal that the proposed controller consumes less energy and improves the base orientation tracking, especially at higher velocities. In future work, we plan to test our controller in the two-layer predictive-reactive control architecture. Also, further studies will address the investigation of the nature of constraints, in addition to the cone constraint, that can be handled by the geometric projection-based approach used by our custom solver.

Acknowledgements. We thank the MRL Lab at IIT Kanpur and the CRRL Lab at NYU for providing the resources to conduct the research and experiments for this paper.

Declarations

- **Funding**: Not applicable.
- **Competing interests**: The authors declare no competing interests.
- **Ethics approval and consent to participate**: Not applicable.
- **Consent for publication**: Not applicable.
- **Data availability**: Not applicable.
- **Materials availability**: Not applicable.
- **Code availability**: Not applicable.
- **Author contributions**: N. Khandelwal led the development and implementation of the DIDC and GPGD algorithms, implemented baseline controllers, designed the software architecture, conducted experiments, and wrote the primary manuscript draft. A. Manu implemented the planning and estimation modules, the balance controller, contributed to GPGD debugging and optimization, and made substantial contributions to both the manuscript and control software implementation, including the safety state machine design. P. Krishnamurthy provided critical technical feedback on the manuscript and experimental methodology. F. Khorrami, M. Kothari, and S. S. Gupta provided editorial oversight and final manuscript revisions.

References

[1] Aghili F (2017) Quadratically constrained quadratic-programming based control of legged robots subject to nonlinear friction cone and switching constraints. IEEE/ASME Transactions on Mechatronics 22(6):2469–2479. https://doi.org/10.1109/TMECH.2017.2755859

[2] Bambade A, El-Kazdadi S, Taylor AB, et al (2022) Prox-qp: Yet another quadratic programming solver for robotics and beyond. In: Proceedings of Robotics: Science and Systems (RSS), New York, United States, p Paper #040, URL https://www.roboticsproceedings.org/rss18/p040.pdf, ffhal-03683733f

[3] Bledt G, Powell MJ, Katz B, et al (2018) Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In: Proceedings of the 2018 IEEE/RSJ

International Conference on Intelligent Robots and Systems (IROS), pp 2245–2252, https://doi.org/10.1109/IROS.2018.8593885

[4] Bledt G, Wensing PM, Ingersoll S, et al (2018) Contact model fusion for event-based locomotion in unstructured terrains. In: Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), pp 4399–4406, https://doi.org/10.1109/ICRA.2018.8460904

[5] Bloesch M, Hutter M, Hoepflinger MA, et al (2013) State estimation for legged robots: Consistent fusion of leg kinematics and imu. In: Robotics: Science and Systems VIII. The MIT Press

[6] Buchli J, Kalakrishnan M, Mistry M, et al (2009) Compliant quadruped locomotion over rough terrain. In: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 814–820, https://doi.org/10.1109/IROS.2009.5354681

[7] Dario Bellicoso C, Gehring C, Hwangbo J, et al (2016) Perception-less terrain adaptation through whole body control and hierarchical optimization. In: Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp 558–564, https://doi.org/10.1109/HUMANOIDS.2016.7803330

[8] Di Carlo J, Wensing PM, Katz B, et al (2018) Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In: Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1–9, https://doi.org/10.1109/IROS.2018.8594448

[9] Doty KL, Melchiorri C, Bonivento C (1993) A theory of generalized inverses applied to robotics. The International Journal of Robotics Research 12(1):1–19

[10] Fahmi S, Mastalli C, Focchi M, et al (2019) Passive whole-body control for quadruped robots: Experimental validation over challenging terrain. IEEE Robotics and Automation Letters 4(3):2553–2560. https://doi.org/10.1109/LRA.2019.2908502

[11] Farshidian F, Jelavić E, Winkler AW, et al (2017) Robust whole-body motion control of legged robots. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4589–4596, https://doi.org/10.1109/IROS.2017.8206328

[12] Farshidian F, Neunert M, Winkler AW, et al (2017) An efficient optimal planning and control framework for quadrupedal locomotion. In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), pp 93–100, https://doi.org/10.1109/ICRA.2017.7989016

[13] Ferreau HJ, Kirches C, Potschka A, et al (2014) qpoases: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation 6:327–363

[14] Focchi M, Del Prete A, Havoutis I, et al (2017) High-slope terrain locomotion for torque-controlled quadruped robots. Autonomous Robots 41:259–272

[15] Foundation E (2024) Eclipse Cyclone DDS. URL https://github.com/eclipse-cyclonedds/cyclonedds, accessed on [20-07-2024]

[16] Frison G, Diehl M (2020) Hpipm: a high-performance quadratic programming framework for model predictive control**this research was supported by the german federal ministry for economic affairs and energy (bmwi) via eco4wind (0324125b) and dyconpv (0324166b), and by dfg via research unit for 2401. IFAC-PapersOnLine 53(2):6563–6569. https://doi.org/https://doi.org/10.1016/j.ifacol.2020.12.073, URL https://www.sciencedirect.com/science/article/pii/S2405896320303293, 21st IFAC World Congress

[17] Gehring C, Coros S, Hutter M, et al (2013) Control of dynamic gaits for a quadrupedal robot. In: Proceedings of the 2013 IEEE International Conference on Robotics and Automation, pp 3287–3292, https://doi.org/10.1109/ICRA.2013.6631035

[18] Hafner D, Lillicrap T, Ba J, et al (2019) Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:191201603

[19] Hildebrand M (1989) The quadrupedal gaits of vertebrates. BioScience 39(11):766

[20] Hutter M, Gehring C, Jud D, et al (2016) Anymal - a highly mobile and dynamic quadrupedal robot. In: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 38–44, https://doi.org/10.1109/IROS.2016.7758092

[21] Jenelten F, He J, Farshidian F, et al (2024) Dtc: Deep tracking control. Science Robotics 9(86):eadh5401

[22] Kang D, Cheng J, Zamora M, et al (2023) Rl + model-based control: Using on-demand optimal control to learn versatile legged locomotion. IEEE Robotics and Automation Letters 8(10):6619–6626. https://doi.org/10.1109/LRA.2023.3307008

[23] Kim D, Di Carlo J, Katz B, et al (2019) Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. arXiv preprint arXiv:190906586

[24] Liu M, Tan Y, Padois V (2016) Generalized hierarchical control. Autonomous Robots 40:17–31

[25] Mistry M, Buchli J, Schaal S (2010) Inverse dynamics control of floating base systems using orthogonal decomposition. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation, pp 3406–3412, https://doi.org/10.1109/ROBOT.2010.5509646

[26] Neunert M, Stäuble M, Giftthaler M, et al (2018) Whole-body nonlinear model predictive control through contacts for quadrupeds. IEEE Robotics and Automation Letters 3(3):1458–1465. https://doi.org/10.1109/LRA.2018.2800124

[27] Raibert MH (1986) Legged robots that balance. MIT press

[28] Raiola G, Mingo Hoffman E, Focchi M, et al (2020) A simple yet effective whole-body locomotion framework for quadruped robots. Frontiers in Robotics and AI 7:528473

[29] Righetti L, Buchli J, Mistry M, et al (2011) Inverse dynamics control of floating-base robots with external constraints: A unified view. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, pp 1085–1090, https://doi.org/10.1109/ICRA.2011.5980156

[30] Savitzky A, Golay MJ (1964) Smoothing and differentiation of data by simplified least squares procedures. Analytical chemistry 36(8):1627–1639

[31] Siciliano B, Sciavicco L, Villani L, et al (2009) Robotics: Modelling, Planning, and Control, 1st edn. Springer London

[32] Sleiman JP, Farshidian F, Minniti MV, et al (2021) A unified mpc framework for whole-body dynamic locomotion and manipulation. IEEE Robotics and Automation Letters 6(3):4688–4695. https://doi.org/10.1109/LRA.2021.3068908

[33] Tasora A, Anitescu M (2011) A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics. Computer Methods in Applied Mechanics and Engineering 200(5-8):439–453

[34] Todorov E, Erez T, Tassa Y (2012) Mujoco: A physics engine for model-based control. In: Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 5026–5033, https://doi.org/10.1109/IROS.2012.6386109