2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Real-time Model Predictive Control for Versatile Dynamic Motions in Quadrupedal Robots

Yanran Ding, Abhishek Pandala, and Hae-Won Park

Abstract— **This paper presents a new Model Predictive Control (MPC) framework for controlling various dynamic movements of a quadrupedal robot. System dynamics are represented by linearizing single rigid body dynamics in three-dimensional (3D) space. Our formulation linearizes rotation matrices without resorting to parameterizations like Euler angles and quaternions, avoiding issues of singularity and unwinding phenomenon, respectively. With a carefully chosen configuration error function, the MPC control law is transcribed into a Quadratic Program (QP) which can be solved efficiently in real-time. Our formulation can stabilize a wide range of periodic quadrupedal gaits and acrobatic maneuvers. We show various simulation as well as experimental results to validate our control strategy. Experiments prove the application of this framework with a custom QP solver could reach execution rates of 160 Hz on embedded platforms.**

I. Introduction

Legged animals have shown their versatile mobility to traverse challenging terrains via a variety of well-coordinated dynamic motions. This remarkable mobility of legged animals inspired the development of many legged robots and associated research works seeking for dynamic legged locomotion in robots. However, designing and controlling a legged robot to achieve similar mobility to that of legged animals remains a difficult problem. Enabling this versatile capability of animals in legged robot systems requires the control design to make good use of its inherent dynamics while dealing with constraints due to hardware limitations as well as interactions with the environment. In the field of legged robots, Model Predictive Control (MPC) recently became a widespread control method due to recent advancements in computing hardware and optimization algorithms, which enabled real-time execution of MPC controller in embedded systems. Based on a model prediction, the MPC framework easily incorporates system dynamics and constraints by transcribing the control law as a constrained optimization problem. Recent applications of MPC on humanoids [1], [2] and quadrupeds [3] have shown the capability of MPC in planning and controlling complex dynamic motions while embracing system dynamics and constraints arising from friction and motor saturation.

Despite the widespread adaptation of MPC, its direct implementation on a high degree-of-freedom (DoF) system requires heavy computational resources, hindering the application on embedded platforms. To tackle this problem, simpler models or templates [4] that capture the dominant system dynamics were used to predict the behavior of the system. Previous MPC schemes [1], [5] worked on simplified dynamics models like Linear Inverted Pendulum [6] (LIPM) which enabled online execution. The planar rigid body model is used in [7] to plan online jumping trajectories for MIT Cheetah 2 with different obstacle heights. The spring-mass model is used in [8] to achieve jumping and landing. Centroidal dynamics [9] model links the linear and angular momentum of the robot with the external wrench. This model is used in [10], [11] to capture the major dynamic effect of the complex full body dynamics model of the humanoid robot Atlas. Recent work such as [12] uses centroidal dynamics for the policy regularized nonlinear MPC (PR-MPC), which takes simple heuristics as reference trajectories for state and control to condition the optimization formulation.

Regarding the use of a simple dynamic model for MPC framework, there has been an increasing number of work utilizing a dynamic model based on a single rigid body in three-dimensional (3D) space. In this model, robot's body and legs are lumped into a single rigid body. Ground reaction forces (GRF) are applied as inputs to control the position and orientation of the body. Even in its simple form, this model efficiently captures the effect of net external wrench on the evolution of CoM motion and orientation. Utilizing this model, [13] could simultaneously optimize for gait and trajectory through phase-based parametrization; [14] achieved various quadrupedal gaits in experiment; [15] obtained high-speed bounding with MIT Cheetah 2.

Although the single rigid body model originally represents the orientation of the robot in 3D space with a rotation matrix, most of the works using this model handled 3D orientation by replacing the rotation matrix with its local coordinates such as Euler angles [13]. However, representation of dynamics using Euler angle is not properly invariant under the action of rigid transformations, so a control design based on such models will provide inconsistent behaviors across different operating configurations. Moreover, Euler angles suffer from singularities [16] that occur in certain configurations. Quaternions representation could cause unwinding issue [17] arising from ambiguities due to multiple spanning of configuration space. In previous works, these problems arising from using local coordinates were not a critical issue as they assume small deviations in roll and pitch angles [12] or only deal with planar motions [15] to represent robot's orientation in a simpler form.

This work presents a novel MPC formulation for controlling legged robots in 3D environment without use of local

Yanran Ding, Abhishek Pandala and Hae-Won Park are with the department of Mechanical Science and Engineering Department, University of Illinois at Urbana-Champaign, IL, 61801, USA. {yding35, pandala2, haewon}@illinois.edu.

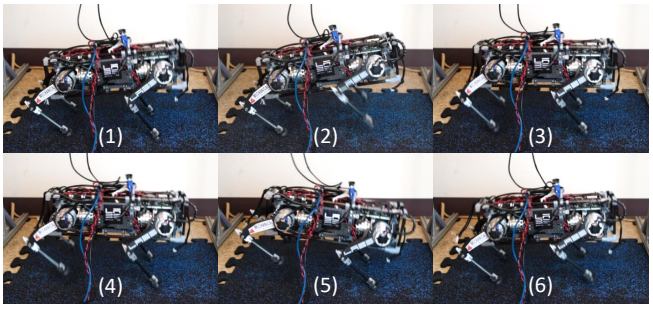

Fig. 1: Sequential snapshots of the robot in a bounding experiment

coordinates to represent rotation matrices. This MPC scheme is formulated into QP which can be solved efficiently in an embedded computer satisfying real-time constraints. To derive this formulation, a linearization technique in [18], [19] which directly works on equations of motion with rotation matrix are applied in this work. We also adopted a configuration error function on $SO(3)$ from [20] to obtain a cost function in a quadratic form. This application of the linearization technique and use of a configuration error function on $SO(3)$ leads to the a singularity-free MPC formulation with consistent performance even when executing motions that involve complex 3D rotations such as acrobatic motions in gymnastics.

The paper is organized as follows: Section II presents the template dynamic model and introduces the linearization scheme together with the modification on the cost function that enables the formulation in QP; Section III shows the simulation and experimental results; Section IV provides the concluding remark with an outlook for future work.

II. Technical Approach

Our goal is to formulate the quadruped dynamic locomotion problem as a MPC scheme that is real-time executable on a mobile embedded computer and uses a global parameterization of orientation with rotation matrix. To meet the real-time constraint, single rigid body model is adopted to characterize the main dynamics of the robot, which is a close approximation of the robot model since the mass of all legs combined is less than 10% of the total mass. In addition, an MPC formulation is transcribed into a real-time conceivable QP, which directly works on representation of the orientation with rotation matrices.

A. 3D Single Rigid Body Dynamics

In this work, the dynamic model of the robot is approximated as a single rigid body with fixed moment of inertia. Then, the state of the robot is,

$$\boldsymbol{x} := [\boldsymbol{p} \quad \dot{\boldsymbol{p}} \quad \boldsymbol{R} \quad {}^{B}\boldsymbol{\omega}] \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the body Center of Mass (CoM); $\dot{\boldsymbol{p}}$ is the CoM velocity; $\boldsymbol{R} \in SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^T\boldsymbol{R} = \mathbb{I}, \det\boldsymbol{R} = 1\}$ is the rotation matrix of the body frame $\{\boldsymbol{B}\}$ expressed in the inertial frame $\{\boldsymbol{S}\}$; ${}^{B}\boldsymbol{\omega}$ indicates the angular velocity vector expressed in the body frame $\{\boldsymbol{B}\}$. Variables without superscript on the upper-left corner are assumed to be expressed in the inertial frame.

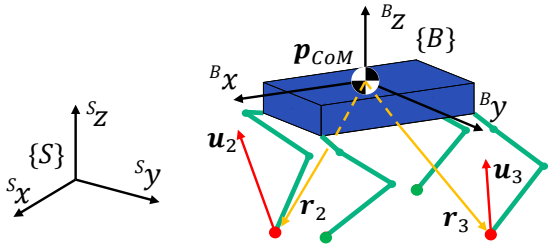

Fig. 2: Illustration of coordinate system and 3D rigid-body model. $\{\boldsymbol{S}\}$ is the inertia frame while $\{\boldsymbol{B}\}$ is the body attached frame. $\boldsymbol{r}_i$ is the position vector from center of mass to each foot in $\{\boldsymbol{S}\}$, while $\boldsymbol{u}_i$ is the ground reaction force of i^{th} contact foot in $\{\boldsymbol{S}\}$

The input to the system is the external wrench, which is created by the ground reaction force (GRF) $\boldsymbol{u}_i \in \mathbb{R}^3$ at contact foot locations $\boldsymbol{p}_i^f \in \mathbb{R}^3$. The foot positions $\boldsymbol{p}_i^f$ relative to CoM are denoted as $\boldsymbol{r}_i = \boldsymbol{p}_i^f - \boldsymbol{p}$. Therefore, the net external wrench exerted on the body is:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \sum_{i=1}^{4} \begin{bmatrix} \mathbb{I} \\ \hat{\boldsymbol{r}}_i \end{bmatrix} \boldsymbol{u}_i \tag{2}$$

where the hat map $\hat{(\cdot)} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined as $\hat{x}y = x \times y, \forall x, y \in \mathbb{R}^3$, and $\mathfrak{so}(3)$ is the space of skew-symmetric matrices; $\mathbb{I}$ is the 3-by-3 identity matrix. The full dynamics of the rigid body can be written as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \ddot{\boldsymbol{p}} \\ \dot{\boldsymbol{R}} \\ {}^{B}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{p}} \\ \frac{1}{M}\boldsymbol{f} - \boldsymbol{a}_g \\ \boldsymbol{R} \cdot {}^{B}\hat{\boldsymbol{\omega}} \\ {}^{B}\boldsymbol{I}^{-1}(\boldsymbol{R}^T\boldsymbol{\tau} - \hat{\boldsymbol{\omega}}{}^{B}\boldsymbol{I}\boldsymbol{\omega}) \end{bmatrix} \tag{3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ represents the state and $\mathbb{R}^m \ni \boldsymbol{u} = [\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, \boldsymbol{u}_3^T, \boldsymbol{u}_4^T]^T$ the control; M is the mass of the rigid body; $\boldsymbol{a}_g \in \mathbb{R}^3$ is the gravitational acceleration; ${}^{B}\boldsymbol{I} \in \mathbb{R}^{3\times3}$ is the fixed moment of inertia tensor in the body frame.

B. Nonlinear MPC Formulation

The equation of motion in (3) includes nonlinear dynamics in $\boldsymbol{R}$ and ${}^{B}\boldsymbol{\omega}$, thereby leading to a nonlinear MPC formulation. Utilizing a direct collocation technique [21], we formulated the nonlinear MPC as an optimization problem. The rigid-body dynamics is discretized and imposed on the sequences of predicted states $\{\boldsymbol{x}_k\}$ and control (GRF) $\{\boldsymbol{u}_k\}$ as equality constraints,

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{N-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) + \ell_T(\boldsymbol{x}_N) \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k), \boldsymbol{x}_1 = \boldsymbol{x}_{op} \\ & \boldsymbol{x}_k \in \mathbb{X}, k = 1, 2, \cdots, N \\ & \boldsymbol{u}_k \in \mathbb{U}, k = 1, 2, \cdots, N-1 \end{aligned} \tag{4}$$

where $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost function; $\ell_T : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost function; N is the prediction horizon; $\mathbb{X}, \mathbb{U}$ are the feasible sets for the state and control.

The stage cost function penalizes deviation of the predicted states and control inputs from the corresponding reference trajectories of the states and control inputs. The definition of the stage cost is as follows

$$\begin{aligned}\ell(\boldsymbol{x}_k, \boldsymbol{u}_k) = \; & \boldsymbol{e}_{u_k}^T \boldsymbol{R}_u \boldsymbol{e}_{u_k} + \boldsymbol{e}_{p_k}^T \boldsymbol{Q}_p \boldsymbol{e}_{p_k} + \boldsymbol{e}_{\dot{p}_k}^T \boldsymbol{Q}_{\dot{p}} \boldsymbol{e}_{\dot{p}_k} + \\ & \boldsymbol{e}_{\omega_k}^T \boldsymbol{Q}_\omega \boldsymbol{e}_{\omega_k} + \boldsymbol{e}_{R_k}^T \boldsymbol{Q}_R \boldsymbol{e}_{R_k}\end{aligned} \tag{5}$$

where $\boldsymbol{R}_u, \boldsymbol{Q}_p, \boldsymbol{Q}_{\dot{p}}, \boldsymbol{Q}_\omega, \boldsymbol{Q}_R$ are positive definite weighting matrices; $\boldsymbol{e}_{u_k}, \boldsymbol{e}_{p_k}, \boldsymbol{e}_{\dot{p}_k}$ are the error terms for deviations from the corresponding desired trajectories $\boldsymbol{u}_k^d, \boldsymbol{p}_k^d, \dot{\boldsymbol{p}}_k^d$, which could be constructed from simple heuristics. The error terms of angular velocity and rotation matrix are given by [20] as

$$\boldsymbol{e}_{\omega_k} = \boldsymbol{\omega}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{d,k} \boldsymbol{\omega}_{d,k} \tag{6}$$

$$\boldsymbol{e}_{R_k} = \log(\boldsymbol{R}_{d,k}^T \cdot \boldsymbol{R}_k)^\vee \tag{7}$$

where $\log(\cdot) : SO(3) \to \mathfrak{so}(3)$ is the logarithm map [22] of rotation matrices; the Vee map $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the inverse of the hat map. The terminal cost function is similarly defined.

While the translational part in (3) is linear, the rotational part evolves nonlinearly following the discrete equation,

$$\begin{aligned}\boldsymbol{\omega}_{k+1} &= \boldsymbol{\omega}_k + {}^B\boldsymbol{I}^{-1}(\boldsymbol{R}_k^T \boldsymbol{\tau}_k - \widehat{\boldsymbol{\omega}}_k \, {}^B\boldsymbol{I} \boldsymbol{\omega}_k)\Delta t \\ \boldsymbol{R}_{k+1} &= \boldsymbol{R}_k \exp(\widehat{\boldsymbol{\omega}_k \Delta t})\end{aligned} \tag{8}$$

where Δt is the time step; $\exp(\cdot) : \mathfrak{so}(3) \to SO(3)$ is the matrix exponential map that makes sure that $\boldsymbol{R}_k$ stays on the $SO(3)$ manifold.

Although, in principle, nonlinear MPC (NMPC) can be solved to obtain control input, the nonlinearities complicates the solving process for the presence of local minima. Moreover, the severe restrictions on computational time hinder the application of NMPC on hardware. To obtain MPC formulation that can be solved with convex optimization, we linearize nonlinear dynamic constraints in (8) using a variation-based linearization scheme.

C. Variation-based Linearization

Although linearization around the reference trajectory gives provable local controllability [26] [18], the obtained linear dynamics differs from the robot's current dynamics when the operating point is not close to the reference trajectory. Hence, linearization is performed around the current state and control (operating point), which is known as successive linearizaiton [23] [24] [25]. To meet the real-time constraint, we employed a element-wise, forward-Euler linearization scheme for the rotation matrix. Linearization is performed by taking variation $\delta(\cdot)$ with respect to the operating point, which is treated here as a linear approximation of the distance of two points on a manifold [27]. Assuming that the predicted variables are close to the operating point, which could be guaranteed by having small time step, the variation of the rotation matrix on $\mathfrak{so}(3)$ will be approximated using the derivative of the error function on $SO(3)$ from [20], [28],

$$\delta \boldsymbol{R}_k \approx \frac{1}{2}(\boldsymbol{R}_{op}^T \boldsymbol{R}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{op}) \tag{9}$$

where the subscripts $(\cdot)_{op}$ and $(\cdot)_k$ indicate variables at the operating point and the k^{th} prediction step, respectively. The rotation matrix at the k^{th} prediction step is also approximated using the first-order Taylor expansion of matrix exponential map,

$$\boldsymbol{R}_k \approx \boldsymbol{R}_{op} \exp(\delta \boldsymbol{R}_k) \approx \boldsymbol{R}_{op}(\mathbb{I} + \delta \boldsymbol{R}_k). \tag{10}$$

Similarly, the variation of angular velocity $\delta \boldsymbol{\omega}_k$ is

$$\delta \boldsymbol{\omega}_k = \boldsymbol{\omega}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{op} \boldsymbol{\omega}_{op} \tag{11}$$

Using equations (10), (11), the dynamics of rotation matrix $\dot{\boldsymbol{R}}_k = \boldsymbol{R}_k \hat{\boldsymbol{\omega}}_k$ is linearized around the operating point $\boldsymbol{R}_{op}$ as

$$\dot{\boldsymbol{R}}_k = \boldsymbol{R}_{op} \hat{\boldsymbol{\omega}}_{op} + \boldsymbol{R}_{op} \hat{\boldsymbol{\omega}}_{op} \delta \boldsymbol{R}_k + \boldsymbol{R}_{op} \widehat{\delta \boldsymbol{\omega}_k} \tag{12}$$

where terms containing higher order variation are eliminated.

The dynamics of angular velocity $\dot{\boldsymbol{\omega}}_k$ is linearized around the operating point as

$$\begin{aligned}{}^B\boldsymbol{I}\dot{\boldsymbol{\omega}}_k = & \boldsymbol{R}_{op}^T \boldsymbol{\tau}_{op} + \delta \boldsymbol{R}_k^T \boldsymbol{\tau}_{op} + \boldsymbol{R}_{op}^T \delta \boldsymbol{\tau}_k + \\ & - \hat{\boldsymbol{\omega}}_{op} \, {}^B\boldsymbol{I} \boldsymbol{\omega}_{op} - \delta \hat{\boldsymbol{\omega}}_k \, {}^B\boldsymbol{I} \boldsymbol{\omega}_{op} - \hat{\boldsymbol{\omega}}_{op} \, {}^B\boldsymbol{I} \delta \boldsymbol{\omega}_k,\end{aligned} \tag{13}$$

in which $\delta \boldsymbol{\tau}_k$ is the variation of torque,

$$\delta \boldsymbol{\tau}_k = (\sum_{i=1}^{4} \hat{\boldsymbol{u}}_{i,op}) \delta \boldsymbol{p}_k + (\sum_{i=1}^{4} \hat{\boldsymbol{r}}_{i,op} \cdot \delta \boldsymbol{u}_{i,k}), \tag{14}$$

where $\boldsymbol{u}_{op}$ is GRF applied at the current step, $\delta \boldsymbol{u}$ is the variation of GRF from $\boldsymbol{u}_{op}$.

D. Vectorization

Even though the dynamics of rotation matrix $\boldsymbol{R}$ and angular velocity $\boldsymbol{\omega}$ are linearized, there are matrix variables in (12) and (13) which are difficult to handle in conventional QP solvers. To tackle the problem, this section proposes a vectorization technique which uses Kronecker product [29] to transform matrix-matrix product into matrix-vector product. Each term of (12) is vectorized as

$$\begin{aligned}& vec(\boldsymbol{R}_{op} \delta \boldsymbol{R}_k \hat{\boldsymbol{\omega}}_{op}) = (\hat{\boldsymbol{\omega}}_{op}^T \otimes \boldsymbol{R}_{op}) \boldsymbol{L} vec(\boldsymbol{R}_k) \\ & vec(\boldsymbol{R}_{op} \delta \boldsymbol{R}_k^T \hat{\boldsymbol{\omega}}_{op}) = (\hat{\boldsymbol{\omega}}_{op}^T \otimes \boldsymbol{R}_{op}) \boldsymbol{P} \boldsymbol{L} vec(\boldsymbol{R}_k) \\ & vec(\boldsymbol{R}_{op} \hat{\boldsymbol{\omega}}_{op} \delta \boldsymbol{R}_k) = (\mathbb{I} \otimes \boldsymbol{R}_{op} \hat{\omega}_{op}) \boldsymbol{L} vec(\boldsymbol{R}_k) \\ & vec(\boldsymbol{R}_{op} \widehat{\delta \boldsymbol{\omega}_k}) = -(\mathbb{I} \otimes \boldsymbol{R}_{op}) \boldsymbol{N} (\mathbb{I} \otimes (\boldsymbol{R}_{op} \boldsymbol{\omega}_{op})^T) vec(\boldsymbol{R}_k) + \\ & \qquad (\mathbb{I} \otimes \boldsymbol{R}_{op}) \boldsymbol{N} \boldsymbol{\omega}_k\end{aligned} \tag{15}$$

where $vec(\cdot)$ is the vectorization operator for a matrix; $\otimes$ is the Kronecker tensor operator; $\boldsymbol{N}$ is a constant matrix such that $\boldsymbol{N}\boldsymbol{\omega} = vec(\hat{\boldsymbol{\omega}})$ for any vector $\boldsymbol{\omega} \in \mathbb{R}^3$; $\boldsymbol{P}$ is a constant permutation matrix such that $\boldsymbol{P} \cdot vec(\boldsymbol{A}) = vec(\boldsymbol{A}^T)$ for an arbitrary matrix $\boldsymbol{A} \in \mathbb{R}^{3\times 3}$; $\boldsymbol{L}$ is used to simplify the expression $\boldsymbol{L} := \frac{1}{2}[(\mathbb{I} \otimes \boldsymbol{R}_{op}^T) - (\boldsymbol{R}_{op}^T \otimes \mathbb{I})\boldsymbol{P}]$. The final expression for $vec(\dot{\boldsymbol{R}}_k)$ is

$$vec(\dot{\boldsymbol{R}}_k) = \boldsymbol{C}_R vec(\boldsymbol{R}_k) + \boldsymbol{C}_\omega \boldsymbol{\omega}_k + \boldsymbol{c}_0 \tag{16}$$

where $\boldsymbol{c}_0 = vec(\boldsymbol{R}_{op} \hat{\boldsymbol{\omega}}_{op})$, $\boldsymbol{C}_\omega = (\mathbb{I} \otimes \boldsymbol{R}_{op})\boldsymbol{N}$, and $\boldsymbol{C}_R$ is the sum of terms that precede $vec(\boldsymbol{R})$ in (15).

The final expression of $\dot{\boldsymbol{\omega}}_k$ is obtained by plugging (14) in (13).

$$\dot{\boldsymbol{\omega}}_k = \boldsymbol{D}_p(\boldsymbol{p}_k - \boldsymbol{p}_{op}) + \boldsymbol{D}_R vec(\boldsymbol{R}_k) + \boldsymbol{D}_\omega \boldsymbol{\omega}_k + \boldsymbol{D}_u \delta \boldsymbol{u}_k + \boldsymbol{d}_0 \tag{17}$$

where the coefficients are defined below

$$\begin{aligned}
\boldsymbol{d}_0 &= {}^B\boldsymbol{I}^{-1} vec(\boldsymbol{R}_{op}^T \boldsymbol{\tau}_{op} - \hat{\boldsymbol{\omega}}_{op}\, {}^B\boldsymbol{I}\boldsymbol{\omega}_{op}) \\
\boldsymbol{D}_u &= {}^B\boldsymbol{I}^{-1} \boldsymbol{R}_{op}^T (\sum \hat{\boldsymbol{r}}_{op}) \\
\boldsymbol{D}_\omega &= {}^B\boldsymbol{I}^{-1} (\widehat{{}^B\boldsymbol{I}\boldsymbol{\omega}_{op}} - \hat{\boldsymbol{\omega}}_{op}\, {}^B\boldsymbol{I}) \\
\boldsymbol{D}_R &= {}^B\boldsymbol{I}^{-1} [(\mathbb{I} \otimes \boldsymbol{\tau}_{op}^T) - {}^B\boldsymbol{I}\boldsymbol{D}_\omega (\mathbb{I} \otimes (\boldsymbol{R}_{op}\boldsymbol{\omega}_{op})^T)] \\
\boldsymbol{D}_p &= {}^B\boldsymbol{I}^{-1} \boldsymbol{R}_{op}^T (\sum \hat{\boldsymbol{u}}_{op})
\end{aligned} \tag{18}$$

Letting $\underline{\boldsymbol{x}}_k := [\boldsymbol{p}_k^T \quad \dot{\boldsymbol{p}}_k^T \quad vec(\boldsymbol{R}_k)^T \quad {}^B\boldsymbol{\omega}_k^T]^T \in \mathbb{R}^n$ be the state vector, and $\boldsymbol{u}_k := [\delta \boldsymbol{u}_{1,k}^T \quad \delta \boldsymbol{u}_{2,k}^T \quad \delta \boldsymbol{u}_{3,k}^T \quad \delta \boldsymbol{u}_{4,k}^T]^T \in \mathbb{R}^m$ be the control vector, the discrete dynamics could be expressed in the state-space form

$$\underline{\boldsymbol{x}}_{k+1} = \boldsymbol{A}\underline{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{d} \tag{19}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \boldsymbol{R}^{n\times m}$, and $\boldsymbol{d} \in \boldsymbol{R}^n$ are constant matrices which could be constructed from (16), (17).

E. Cost Function and Inequality Constraints

In the expression of the stage cost function (5), all the terms are in a quadratic form of the state vector $\boldsymbol{x}_k$ and control vector $\boldsymbol{u}_k$ except the cost on orientation error $\boldsymbol{e}_{R_k}^T \boldsymbol{Q}_R \boldsymbol{e}_{R_k}$ due to the expression of $\boldsymbol{e}_{R_k}$ in (7) as the logarithm map of the rotation matrix $\boldsymbol{R}_k$. To closely approximate the error term, here we adopt the configuration error function of Proposition 11.31 in [20],

$$\Psi(\boldsymbol{R}_d^T \boldsymbol{R}_k) = \frac{1}{2}\text{tr}[\boldsymbol{G}_p(\mathbb{I} - \boldsymbol{R}_d^T \boldsymbol{R}_k)] \tag{20}$$

where $\boldsymbol{G}_p := \text{tr}(\boldsymbol{K}_p)\mathbb{I} - \boldsymbol{K}_p$; $\text{tr}(\cdot)$ gives the trace of a matrix; $\boldsymbol{K}_p$ is a symmetric positive definite matrix which will be selected to incorporate the weight matrix $\boldsymbol{Q}_R$. Because this configuration error function is linear with respect to $\boldsymbol{R}_k$, the stage cost function is quadratic with the introduction of this approximation, penalizing orientation error.

This configuration error function $\Psi(\boldsymbol{X})$ for $\boldsymbol{X} \in SO(3)$ is smooth with $\Psi(\mathbb{I}) = 0$. This function is also locally positive definite about $\mathbb{I}$ within the region where the rotation angle of $\boldsymbol{X}$ from $\mathbb{I}$ is less than 180° which almost covers $SO(3)$.

In order to obtain $\boldsymbol{K}_p$, we use $\boldsymbol{e}_R = \log(\boldsymbol{R}_d^T \boldsymbol{R}_k)^\vee$ and Rodirigues' Formula [20] to rewrite the configuration error function (20) as,

$$\begin{aligned}
\Psi\{\exp[\log(\boldsymbol{R}_d^T \boldsymbol{R}_k)^\vee]\} &= \Psi\{\exp(\boldsymbol{e}_R)\} \\
&= \frac{1 - \cos||\boldsymbol{e}_R||}{2||\boldsymbol{e}_R||^2} \boldsymbol{e}_R^T [\text{tr}(\boldsymbol{K}_p)\mathbb{I} + \boldsymbol{K}_p]\boldsymbol{e}_R
\end{aligned} \tag{21}$$

where the coefficient term $\frac{1-\cos||\boldsymbol{e}_R||}{2||\boldsymbol{e}_R||^2}$ approaches $\frac{1}{4}$ as $||\boldsymbol{e}_R|| \to 0$. Comparing (21) and the term $\boldsymbol{e}_{R_k}^T \boldsymbol{Q}_R \boldsymbol{e}_{R_k}$ in (5), the expression for $\boldsymbol{K}_p$ could be calculated from

$$\frac{1}{4}[\text{tr}(\boldsymbol{K}_p) \cdot \mathbb{I} + \boldsymbol{K}_p] = \boldsymbol{Q}_R \tag{22}$$

given weighting matrix $\boldsymbol{Q}_R$.

TABLE I: System Parameters of the Robot

Parameter	Value	Unit
M	5.5	kg
I_{xx}	0.026	kg m^2
I_{yy}	0.112	kg m^2
I_{zz}	0.075	kg m^2
Body length	0.3	m
Body width	0.2	m
link length	0.14	m

Friction cone is approximated as a linearized friction pyramid, and the following inequality constraint is imposed

$$\left\|\boldsymbol{u}_{op}^t + \delta \boldsymbol{u}^t\right\| \leq \mu(\boldsymbol{u}_{op}^n + \delta \boldsymbol{u}^n) \tag{23}$$

where μ is the coefficient of friction; superscript $(\cdot)^t$ indicates tangential force, and $(\cdot)^n$ normal force.

The following box constraints on the GRF are added to clamp the force within desired ranges.

$$\boldsymbol{u}_{i,min} \leq \boldsymbol{u}_{i,op} + \delta \boldsymbol{u}_{i,k} \leq \boldsymbol{u}_{i,max} \tag{24}$$

This clamping is used to set the value of GRF to zero for legs in swing phase by clamping $\boldsymbol{u}_{i,min}$ and $\boldsymbol{u}_{i,max}$ to zero.

III. RESULTS

This section evaluates the MPC controller through simulation and hardware experiments.

A. Simulation

The simulation experiments are setup in the following way. In each sample time, control input is obtained by solving the QP transcribed by MPC using $quadprog$ in MATLAB, and then applied to the continuous model (3) through $ode45$ to simulate the evolution of nonlinear dynamics. Table I provides the system parameters used in the simulation.

*1) **Walking Trot**:* The walking trot simulation serves as a baseline test for verifying the tracking capability of the proposed MPC formulation. The robot is commanded a forward walking speed of 0.3 m/s. Fig. 3 shows the position and orientation deviation from the reference point. The gait pattern is time-based, with the stance time set as 0.4 sec and swing time 0.2 sec.

*2) **Bounding**:* The bounding simulation is presented here to verify that the proposed MPC formulation is capable of leveraging full body dynamics. To achieve bounding motion, a dynamically feasible reference trajectory that preserves periodicity is designed from the impulse-scaling analysis [15], where the gain on the pitch motion is set as $\alpha_\theta = 15$. In this experiment, the robot is commanded to follow a forward trajectory starting from the static nominal pose and accelerate to 1.0 m/s. Fig. 4 (a) shows that the pitch motion converges to a periodic orbit; Fig. 4 (b) demonstrates that the robot is able to closely track the forward reference trajectory. The gait pattern is also time-based with a stance time of 0.1 sec and swing time of 0.2 sec.

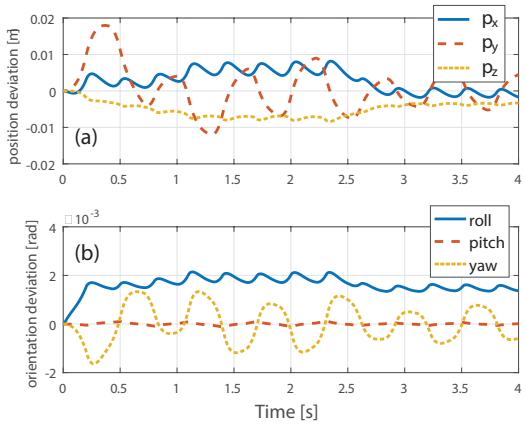

Fig. 3: Simulation results of walking trot (a) CoM position deviation (b) orientation deviation. The reference is 0 for both position and orientation.

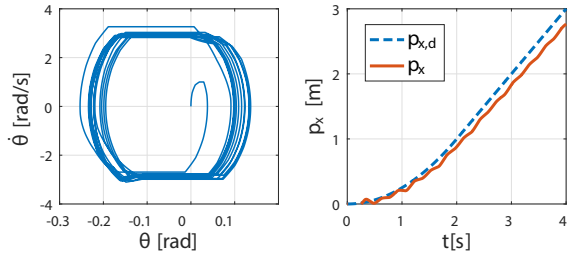

Fig. 4: Simulation results of bounding are generated with the robot start from static nominal pose and track a forward accelerating trajectory (a) pitch angular velocity and pitch angle converge to a periodic orbit (b) position tracking performance of the controller

*3) **Aperiodic Complex Dynamic Maneuver**:* In addition to simulation of periodic gait motions, we demonstrate the performance of our MPC to control a complex acrobatic 3D maneuver. Fig. 5 shows the reference trajectory of back flip with twist that are tracked using the MPC. Initially, the robot stands on an inclined surface with a slope of $45°$. Then, the robot initiates jump with all the four legs in contact with the surface until 0.1 sec. When time reaches 0.1 sec, the front pair of legs are airborne while the hind pair of legs are still in contact with the ground applying forces to the surface. At 0.2 sec, the hind pair of legs also lift off from the surface. With the selection of a feed-forward trajectory of ground reaction forces obtained from separate trajectory optimization, this jumping creates a back flip with twist, which makes the robot land at 0.5 sec after an airborne phase in an upright configuration that faces opposite to the inclined surface. Fig. 5(c)(d) show a comparison between open-loop uncontrolled back flip with twist and closed-loop back flip with twist controlled by our MPC controller when the slope of surface is perturbed to $53.6°$ from $45°$. It can be observed that up to 0.2 sec during the stance phase while the robot has control authority, orientation errors are better regulated in the MPC controller compared to the open-loop controller, placing the landing configuration close to the desired configuration.

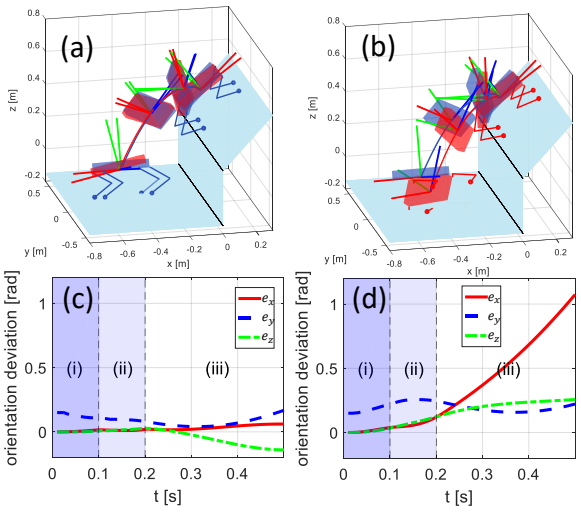

Fig. 5: Simulation results of a complex aperiodic 3D maneuver where the robot performs a twist jump off a sloped surface. The slope of the surface is deviated from the nominal $45°$ to $53.6°$. (a) reference (blue) and MPC controlled (red) trajectories, (b) reference (blue) and open-loop (red) trajectories. (c) orientation deviation of MPC controlled trajectory from the reference, (d) orientation deviation of open-loop trajectory from the reference. The orientation deviation is calculated using log map $\boldsymbol{e}_R = \log(\boldsymbol{R}_d^T \boldsymbol{R})^\vee$. (i) is when four legs are all in contact with the surface; (ii) is when only hind legs are in contact; (iii) is aerial phase

B. Experiment

*1) **Platform Description**:* The hardware platform used for the experiments is a fully torque controllable, electrical quadruped that weighs 5.5 kg. Each leg module of the quadruped robot is equipped with three custom made brushless direct current (BLDC) motor units [30]. The body of the robot is assembled from carbon fiber tubes and plates, connected by carbon fiber reinforced 3D printed parts. The foot is cushioned with sorbothane because of its capability to absorb shock. The electronic system consists of an on-board computer PC104 with Intel i7-3517UE at 1.70 GHz, Elmo Gold Twitter amplifiers, RLS-RMB20 magnetic encoders on each joint, and an inertial measuring unit (IMU).

*2) **Computation Setup**:* The controller is implemented in Simulink Real-Time (SLRT), which runs on the embedded system at a base rate of 4 kHz, while the IMU signals are received at 1 kHz. A custom QP solver based on the interior-point method with algorithmic details outlined in [31] is developed to embed the QP solver into SLRT. It exploits the sparsity structure of the KKT matrix induced by the MPC formulation to solve QP efficiently, enabling faster executions. For a prediction horizon of 7, this entails solving a QP with 210 variables, 168 inequality and 126 equality constraints. Written in ANSI C, the solver is library-free while and it interfaces with SLRT through a gateway s-function. Even with reduced computational resources on embedded system, the proposed MPC achieves execution rates of 160 Hz for prediction horizon of 7. Further details

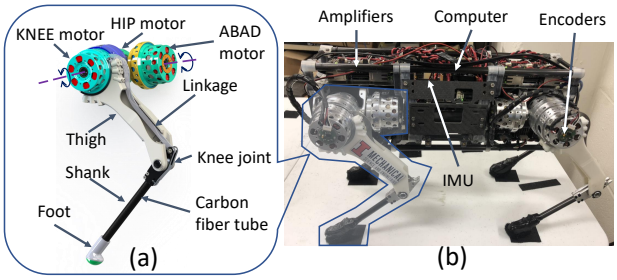

Fig. 6: An illustration of the hardware Platform (a) CAD model of the mechanical components of the leg module, which includes ABAD, HIP and KNEE modules and linkages of the leg (b) A picture of the assembled quadruped platform, which integrates a computer, sensors including an IMU and 12 encoders

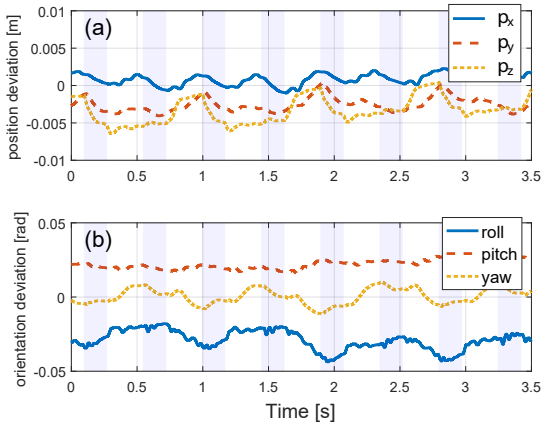

Fig. 7: Results of a trotting experiment on the hardware. (a) CoM position deviation from the reference (b) orientation deviation from the reference. The shaded area indicates when all four legs are in stance phase. The reference is 0 for both position and orientation.

about the custom QP solver are out of the scope of this paper.

*3) **Walking Trot**:* The walking trot gait is tested to evaluate the performance of the proposed MPC formulation. Fig. 7 shows that the deviations in both position and orientation are kept small during the trotting experiment. Stance and swing phases are switched in a fixed time schedule. The swing foot follows a smooth spline using the PD position control. Specifically, the foot retracts and re-initiate contact at the desired foothold, which is designed based on the capture-point [32]. The gain values in Table II are first tuned in simulation and then adjusted for experiment as the model only approximates real physical systems.

*4) **Bounding**:* Bounding experiments are conducted to show that the proposed method could leverage the full system dynamics. Fig. 1 shows a sequence of snapshots of the bounding in place experiment. Fig. 8 shows the pitch angle and pitch angular velocity plots of a bounding experiment where the robot starts from a static pose. Transition to stance phase occurs upon a touchdown event detected using momentum observer [33]. This experiment is preliminary result where the robot could bound up to 4 steps. Further experimental test will be conducted.

TABLE II: Cost function weights for simulation and experiments. The values in parenthesis represent weights on terminal cost.

	Sim. Bounding	Sim. Trotting	Acro. Maneuver	Exp. Bouding	Exp. Trotting
Q_{p_x}	4e5 (5e5)	5e4	5e6	5e4	2e5 (12e4)
Q_{p_y}	4e5 (5e5)	5e4	5e6	5e4	4e5 (4e5)
Q_{p_z}	4e5 (5e5)	5e4	5e6	7e4	1.5e5 (2e5)
$Q_{\dot{p}_x}$	1e3 (4e3)	2e3 (4e3)	5e3	5e2	50
$Q_{\dot{p}_y}$	1e3 (4e3)	2e3 (4e3)	5e3	5e2	200 (150)
$Q_{\dot{p}_z}$	1e3 (4e3)	2e3 (4e3)	5e3	5e2	30
Q_{R_x}	3e4 (5e4)	1e4 (5e4)	1e6	1e3	3e3 (1e3)
Q_{R_y}	3e4 (5e4)	1e4 (5e4)	1e6	0	4e3 (8e3)
Q_{R_z}	3e4 (5e4)	1e4 (5e4)	1e6	0	1e3 (3e3)
Q_{ω_x}	5e2	5e2	5e3	1e2	3 (2)
Q_{ω_y}	5e2	5e2	5e3	0	6 (2)
Q_{ω_z}	5e2	5e2	5e3	2e2	5 (8)
R_{u_x}	0.1	0.1	0.1	0.1	0.1
R_{u_y}	0.1	0.1	0.1	0.1	0.18
R_{u_z}	0.1	0.1	0.1	0.1	0.2

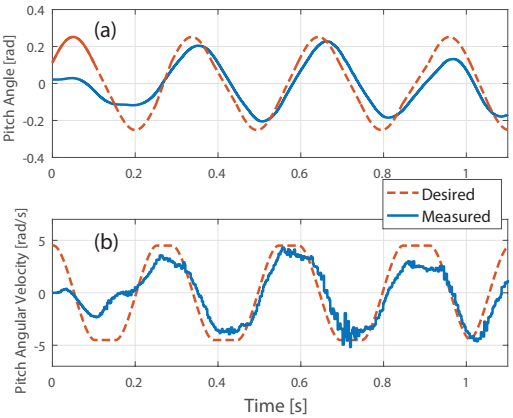

Fig. 8: Experiment results of (a) pitch angle and (b) pitch angular velocity in a bounding experiment

IV. Conclusion and Future Work

In this work, we present a Model Predictive Control framework to control various kinds of dynamic maneuvers in 3D space. By directly using linearized models with rotation matrices, our method could avoid issues arising from use of local coordinates including singularities (Euler angles) and unwinding issues (quaternion). Along with this linearization, the choice of error function on body orientation enables a QP formulation, which could be efficiently solved to achieve real-time execution. With the proposed control framework, a wide range of dynamic gaits both in simulation and experiments are demonstrated. In the future, we plan to apply this MPC controller to control more complex dynamic maneuvers including fast galloping and more acrobatic gymnastic motions.

V. Acknowledgment

This project is supported by NAVER LABS Corp. under grant 087387, Air Force Office of Scientific Research under grant FA2386-17-1-4665, and National Science Foundation under grant 1752262.

REFERENCES

[1] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[2] B. Henze, C. Ott, and M. A. Roa, "Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3253–3258.

[3] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[4] R. J. Full and D. E. Koditschek, "Templates and anchors: neuromechanical hypotheses of legged locomotion on land," *Journal of experimental biology*, vol. 202, no. 23, pp. 3325–3332, 1999.

[5] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *IROS 2010-IEEE-RSJ International Conference on Intelligent Robots & Systems*. IEEE, 2010, pp. 190–195.

[6] S. KAJITA, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Proc. IEEE Int. Conf. on Robotics and Automation, Sacramento, CA, 1991*, 1991, pp. 1405–1411.

[7] H.-W. Park, P. M. Wensing, S. Kim *et al.*, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems, Rome, Italy, July 13-17*, 2015.

[8] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3821–3828.

[9] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[10] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[11] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with simple dynamics and full kinematics," in *Proceedings of the IEEE-RAS international conference on humanoid robots*, 2014.

[12] G. Bledt, P. M. Wensing, and S. Kim, "Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 4102–4109.

[13] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[14] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[15] H.-W. Park, P. M. Wensing, and S. Kim, "High-speed bounding with the MIT cheetah 2: Control design and experiments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 167–192, 2017.

[16] M. D. Shuster, "A survey of attitude representations," *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[17] S. P. Bhat and D. S. Bernstein, "A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon," in *American Control Conference, 1998. Proceedings of the 1998*, vol. 5. IEEE, 1998, pp. 2785–2789.

[18] G. Wu and K. Sreenath, "Variation-based linearization of nonlinear systems evolving on SO(3)and S2,," *IEEE Access*, vol. 3, pp. 1592–1604, 2015.

[19] T. Lee, M. Leok, and N. H. McClamroch, "Stable manifolds of saddle equilibria for pendulum dynamics on S2 and SO(3)," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 2011, pp. 3915–3921.

[20] F. Bullo and A. D. Lewis, *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*. Springer Science & Business Media, 2004, vol. 49.

[21] O. von Stryk, *Numerical Solution of Optimal Control Problems by Direct Collocation*. Basel: Birkhäuser Basel, 1993, pp. 129–143. [Online]. Available: https://doi.org/10.1007/978-3-0348-7539-4_10

[22] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Journal of Mechanical Design*, vol. 117, no. 1, pp. 48–54, 1995.

[23] M. A. Henson, "Nonlinear model predictive control: current status and future directions," *Computers and chemical engineering*, vol. 23, pp. 187–202, 1998.

[24] F. Kuhne, W. F. Lages, and J. G. da Silva Jr, "Model predictive control of a mobile robot using linearization," in *Proceedings of mechatronics and robotics*, 2004, pp. 525–530.

[25] M. Cannon, D. Ng, and B. Kouvaritakis, "Successive linearization nmpc for a class of stochastic nonlinear systems," in *Nonlinear Model Predictive Control*. Springer, 2009, pp. 249–262.

[26] J. P. Hespanha, *Linear systems theory*. Princeton university press, 2018.

[27] J. E. Marsden and T. S. Ratiu, "Introduction to mechanics and symmetry," *Physics Today*, vol. 48, no. 12, p. 65, 1995.

[28] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on SE(3)," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5420–5425.

[29] A. Graham, *Kronecker products and matrix calculus with applications*. Courier Dover Publications, 2018.

[30] Y. Ding and H.-W. Park, "Design and experimental implementation of a quasi-direct-drive leg for optimized jumping," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference*. IEEE, 2017, pp. 300–305.

[31] L. Vandenberghe, "The cvxopt linear and quadratic cone program solvers," *Online: http://cvxopt.org/documentation/coneprog. pdf*, 2010.

[32] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 200–207.

[33] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 634–639.