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

Yanran Ding[1] , Abhishek Pandala[2], and Hae-Won Park[3]

[1]University of Illinois at Urbana-Champaign (UIUC)
[2]Virginia Polytechnic Institute and State University
[3]Korea Advanced Institute of Science & Technology (KAIST)

Introduction

- The remarkable mobility of legged animals inspired the development of many legged robots with dynamic legged locomotion capabilities.
- *Model Predictive Control* (MPC) recently became a widespread control method due to advancements in computing hardware and optimization algorithms, which enabled real-time execution of MPC controller in embedded systems.
- 3D orientation is often represented using local coordinates such as *Euler angles* or *quaternion*. However, Euler angles suffer from singularities that occur in certain configurations. Quaternions representation could cause unwinding issue arising from ambiguities due to multiple spanning of configuration space.
- This work presents a novel MPC formulation for controlling legged robots in 3D environment where the orientation is represented using the ***rotation matrix***. This MPC scheme is formulated into Quadratic Program (QP) which can be solved efficiently in an embedded computer satisfying real-time constraints.

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. The state of the robot is: $\boldsymbol{x} := [\boldsymbol{p} \quad \dot{\boldsymbol{p}} \quad \boldsymbol{R} \quad {}^{B}\boldsymbol{\omega}]$

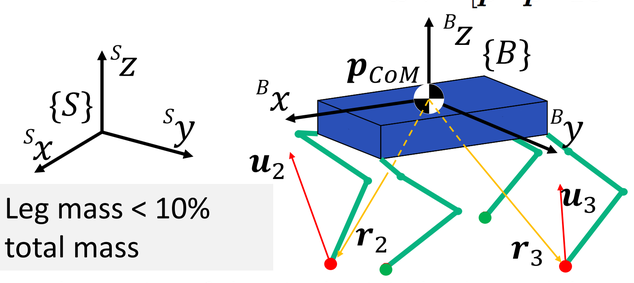


Fig. 1 Schematics of the single rigid body model

The input to the system is the external wrench, which is created by the ground reaction force (GRF) $\boldsymbol{u}_i$. The net external wrench exerted on the body is:

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

where the *hat map* is defined as $\hat{x}y = x \times y$. The full dynamics of the rigid body can be written as:

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

where M is the mass of the rigid body; $\boldsymbol{a}_g$ is the acceleration due to gravity; ${}^{B}\boldsymbol{I}$ is the body fixed inertia tensor; $\boldsymbol{u}$ is the control input.

MPC formulation

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,

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

where ℓ is the stage cost and ℓ_T is the terminal cost; N is the prediction horizon; $\mathbb{X}, \mathbb{U}$ are the feasible sets for state and control. The cost functions penalizes deviation of the predicted states and control from the corresponding reference trajectories.

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

where $\boldsymbol{e}_{\{\cdot\}}$ are the error terms; $\boldsymbol{R}_u$ and $\boldsymbol{Q}_{\{\cdot\}}$ are the positive definite weighting matrices. The error terms of angular velocity and rotation matrix are given as,

$$\begin{aligned} \boldsymbol{e}_{\omega_k} &= \boldsymbol{\omega}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{d,k} \boldsymbol{\omega}_{d,k} \\ \boldsymbol{e}_{R_k} &= \log(\boldsymbol{R}_{d,k}^T \cdot \boldsymbol{R}_k)^\vee \end{aligned}$$

where $\log(\cdot)$ is the *logarithm map* of rotation matrices; the *Vee map* $(\cdot)^\vee$ is the inverse of the hat map.

Variation-based Linearization

The rotational dynamics of the single ridged body evolves nonlinearly following the discrete equation,

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

Here we employed *successive linearization*, which is linearization performed around the current state and control (operating point). Linearization is performed by taking variation $\delta(\cdot)$ with respect to the operating point.

$$\begin{aligned} \delta \boldsymbol{R}_k &\approx \frac{1}{2}(\boldsymbol{R}_{op}^T \boldsymbol{R}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{op}) \\ \delta \boldsymbol{\omega}_k &= \boldsymbol{\omega}_k - \boldsymbol{R}_k^T \boldsymbol{R}_{op} \boldsymbol{\omega}_{op} \end{aligned}$$

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

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

The dynamics of rotation matrix $\dot{\boldsymbol{R}}_k = \boldsymbol{R}_k \hat{\boldsymbol{\omega}}_k$ is linearized

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

The dynamics of angular velocity $\dot{\omega}_k$ is linearized as

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

in which $\delta\tau_k$ is the variation of torque,

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

Vectorization

Even though the rotational part of the dynamics is linearized, there are *matrix variables* which are difficult to handle in conventional QP solvers. To tackle the problem, *Kronecker product* is used to transform matrix-matrix product into matrix-vector product.

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

Where $vec(\cdot)$ is the vectorization operator for a matrix; $\otimes$ is the Kronecker tensor operator; $\boldsymbol{N}$ is a constant matrix such that $\boldsymbol{N}\omega = vec(\hat{\omega})$; $\boldsymbol{P}$ is a constant permutation matrix such that $\boldsymbol{P} \cdot vec(\boldsymbol{A}) = vec(\boldsymbol{A}^T)$ for any matrix $\boldsymbol{A} \in R^{3\times3}$; $\boldsymbol{L} := \frac{1}{2}[(\mathbb{I} \otimes R_{op}^T) - (R_{op}^T \otimes \mathbb{I})P]$

The discrete dynamics could be expressed as,

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

where $\underline{\boldsymbol{x}}_k := [\boldsymbol{p}_k^T \quad \dot{\boldsymbol{p}}_k^T \quad vec(\boldsymbol{R}_k)^T \quad {}^{B}\boldsymbol{\omega}_k^T]^T$

Cost Function

The error term of rotation matrix involves logarithmic map, which is a nonlinear function. To closely approximate the error term, one adopts the *configuration error function*

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

where $\boldsymbol{G}_p := \mathrm{tr}(\boldsymbol{K}_p)\mathbb{I} - \boldsymbol{K}_p$, $tr(\cdot)$ gives the trace of a matrix; $\boldsymbol{K}_p$ is a symmetric positive definite matrix which could be calculated from

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

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

Inequality Constraints

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

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

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 the desired range,

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

GRF are clamped to zero for swing feet.

Simulation

(1) The bounding simulation is presented here to verify that the proposed MPC formulation is capable of leveraging full body dynamics

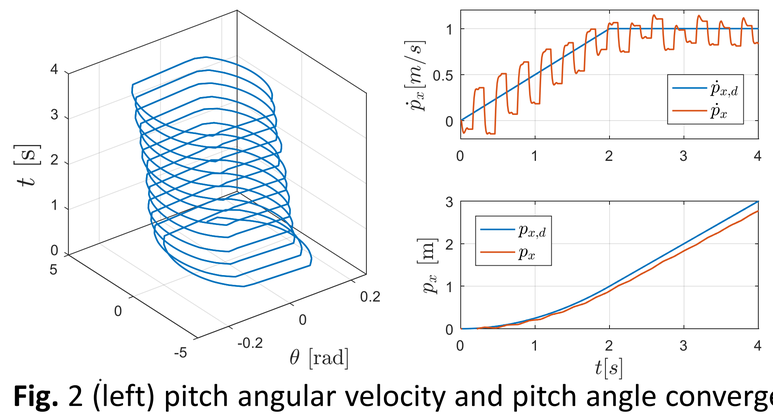


Fig. 2 (left) pitch angular velocity and pitch angle converge to a periodic orbit (right) position and velocity of CoM tracks the desired trajectory

(2) The simulation of the *aperiodic complex dynamic maneuver* is presented. The robot jumps off an inclined surface of a slope of 45° while performing a twist jump.

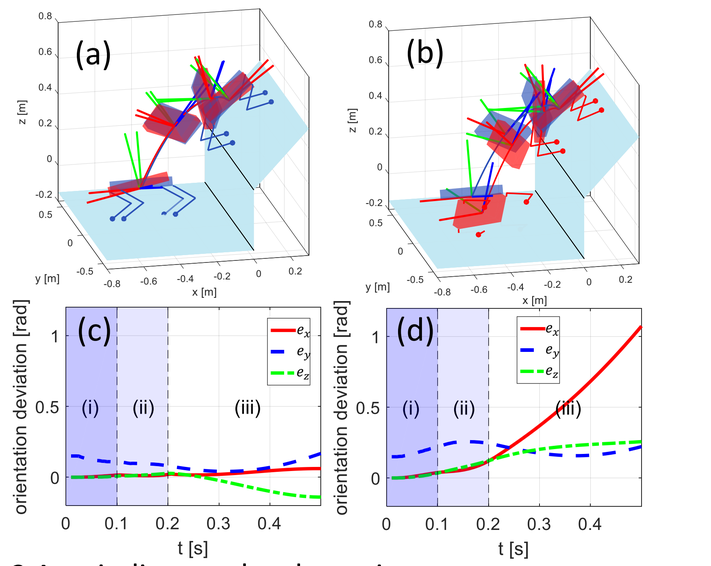


Fig. 3 Aperiodic complex dynamic maneuver

Experiment

The same MPC framework could be used to achieve various motion tasks such as **walking trot**, **pose control**, **balancing** and **bounding**.

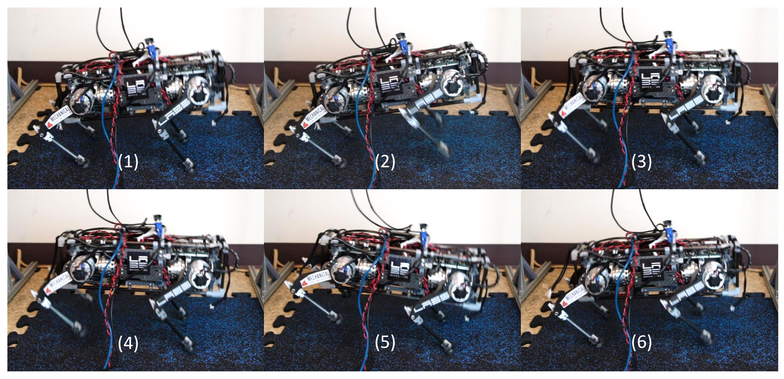
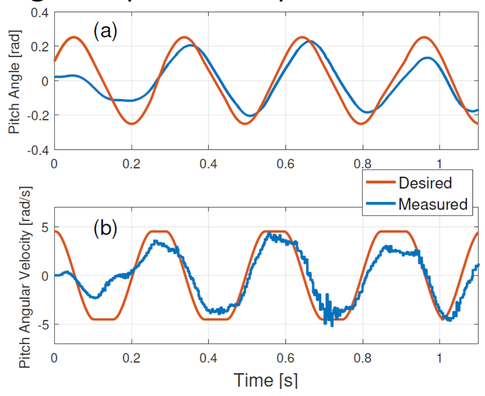


Fig. 4 Sequential snapshots of the robot bounding



In the bounding experiment, the robot starts from static pose and initiate the bounding gait trying to track a prescribed trajectory.

Fig. 5 Experiment results of (a) pitch angle and (b) pitch angular velocity in a bounding experiment.

Acknowledgement

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.