Per-Contact Iteration Method for Solving Contact Dynamics

Author(s):
Hwangbo, Jemin; Lee, Joonho; Hutter, Marco

Publication date:
2018-04

Permanent link:
https://doi.org/10.3929/ethz-b-000224856

Originally published in:
IEEE Robotics and Automation Letters 3(2), https://doi.org/10.1109/LRA.2018.2792536

Funding acknowledgement:
166232 - Data-driven control approaches for advanced legged locomotion (SNF)

Per-Contact Iteration Method for Solving Contact Dynamics

Jemin Hwangbo, Joonho Lee and Marco Hutter

***Abstract*—This paper introduces a new iterative method for contact dynamics problems. The proposed method is based on an efficient bisection method which iterates over each contact. We compared our approach to two existing ones for the same model and found that it is about twice as fast as the existing ones. We also introduce four different robotic simulation experiments and compare the proposed method to the most common contact solver, the projected Gauss-Seidel (PGS) method. We show that, while both methods are very efficient in solving simple problems, the proposed method significantly outperforms the PGS method in more complicated contact scenarios. Simulating one time step of an 18-DOF quadruped robot with multiple contacts took less than 20 μs with a single core of a CPU. This is at least an order of magnitude faster than many other simulators which employ multiple relaxation methods to the major dynamic principles in order to boost their computational speed. The proposed simulation method is also stable at 50 Hz due to its strict adherence to the dynamical principles. Although the accuracy might be compromised at such a low update rate, this means that we can simulate an 18-DOF robot more than thousand times faster than the real time.**

***Index Terms*—simulation, contact dynamics, legged robot**

I. Introduction

THE objective of this work is to build a fast rigid-body dynamics simulator that can be used to generate realistic data for various robotic tasks. Generating realistic data can be difficult for two reasons: our lack of understanding of the true physics and our lack of abilities to solve what we believe to be an accurate model of the true physics. The first reason is critical in many dynamic simulations but we believe that contacts between rigid bodies that do not manifest bouncing behaviors can be fairly accurately simulated. Fortunately, this is the case for many robotic tasks that we are interested in, e.g. legged locomotion, manipulation of hard objects and building construction.

Developing very accurate models is an active research area [1], [2] but the focus of this paper is to develop a very fast contact solver which uses less approximations than the existing simulators. There are three fundamental principles that are commonly used to model rigid body contacts: the Signorini condition, the Coulomb's friction cone constraint, and the maximum dissipation principle. Most existing simulators relax all of them in order to make the problem tractable. Although these well-known principles are also approximations to the true physics, we believe that they are better than the models that are artificially modified just to make the problem easier.

*This work was funded by Swiss National Science Foundation (SNF) through the National Centre of Competence in Research Robotics and Project 200021-166232.

all authors are associated with RSL, ETHZ

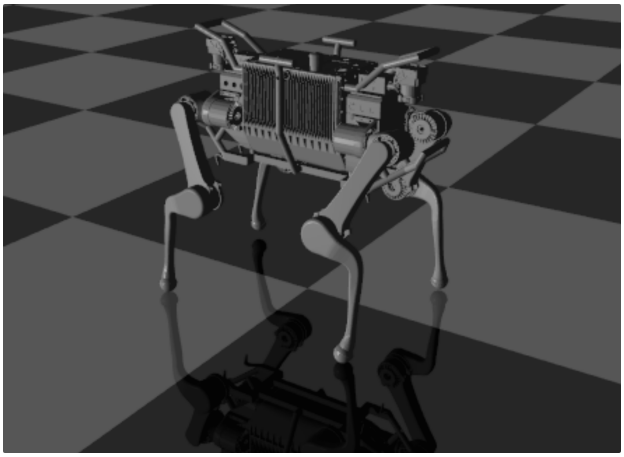

Figure 1: Rendering of ANYmal simulation.

A common approximation is a polytope approximation to the friction cone which effectively transforms the Nonlinear Complementarity Problem (NCP) to a Linear Complementarity Problem (LCP) [3]. Such approximation usually compromises the accuracy severely when simulating slipping condition and the solution is also highly affected by the polytope approximation scheme. In addition, the LCP based contact solvers are relatively slow and require a further approximation which usually results in relaxation of the Signorini condition. This results in even more unrealistic contact forces and the physical realism is sometimes lost in complicated contact scenarios.

Another common approximation that is employed in the existing simulators is a relaxation to the complementarity constraint. Chrono[1] and MuJoCo [4] are probably the most established simulators of this kind. The downside of this method is that the objects can penetrate each other. Some of the simulators are also unable to simulate different restitution characteristics of materials. Consequently, the simulation results might appear to be unrealistic.

The most common way to solve unapproximated Coulomb's friction model is the over-relaxed projected Gauss-Seidel method (PGS) [5], [6] which works very similarly to Gauss-Seidel algorithm for linear systems. It is a main solver for many of the aforementioned simulators. The idea is to exploit the fact that the Delassus operator is generally diagonal dominant. This is particularly true for granular physics simulation where there are many spherical particles. However, it does not hold true for articulated robotic models due to their

[1]https://projectchrono.org/

many bilateral constraints. In such case, the PGS method usually shows poor performance. This is due to the fact that individually updating the contact force components might negate the effect of the previous updates. Per-contact iteration methods, which have been well-studied in [7], might be more practical for robotic simulations. They solve a multi-contact problem by iteratively solving single contact problems. This leads to an algorithm that can fully account for the effect of the off-diagonal elements of an inverse apparent mass matrix.

In this paper, we propose a new per-contact iteration method for solving contact dynamics and evaluate it on robotic examples. We use a bisection method to find the optimal solution for a single contact problem and a successive over-relaxation to find the optimal solution to a multi-contact problem. We compare it to the PGS method in robotic simulations and analyze the computation costs of each algorithm.

II. Background

We consider a case where a single robot with a floating base making contacts with a static environment. The theory here is not limited to such case, but this will lead to a more concise description and be more relevant to the systems of our interest. We use **bold** symbols for vector/matrix quantities for clarity. The robot's generalized coordinates $\boldsymbol{q} \in \mathrm{SO}(3) \times \mathbb{R}^{n-3}$ and generalized velocity $\boldsymbol{u} \in \mathbb{R}^n$ are expressed by a minimal set of coordinates. At each instance of time, the robot makes N contacts with the environment. The resulting contact forces acting on the robot are denoted as $\boldsymbol{f} \in \mathbb{R}^{3N}$. The dynamics of the robot can be written as

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{u}} = \boldsymbol{\tau} - \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u}) + \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{f}, \tag{1}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the mass matrix, $\boldsymbol{\tau}$ is the actuation torque, $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{u})$ is the nonlinear forces and $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian mapping the generalized velocity to the contact space velocity $\boldsymbol{v}$ which is a concatenation of all contact velocities expressed in each contact frame. The contact frames are defined such that z-axes are always parallel to the respective normal vectors. The directions of x- and y-axes can be arbitrary since we limit our scope to isotropic friction. From here on, we abuse the notation and omit function arguments for brevity.

It is well-known that simulating impacts with Coulomb's friction law in force-acceleration context is inconsistent and results in no solution in some cases (i.e. Painlevé paradox). To avoid this problem, we use a simple discretization scheme

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{u}^+ - \boldsymbol{u}^-) &= \Delta t(\boldsymbol{\tau} - \boldsymbol{h}) + \boldsymbol{J}^T\boldsymbol{\lambda} \\ \boldsymbol{M}\boldsymbol{u}^+ &= \boldsymbol{M}\boldsymbol{u}^- + \Delta t(\boldsymbol{\tau} - \boldsymbol{h}) + \boldsymbol{J}^T\boldsymbol{\lambda} \\ \boldsymbol{u}^+ &= \boldsymbol{u}^- + \boldsymbol{M}^{-1}\{\Delta t(\boldsymbol{\tau} - \boldsymbol{h}) + \boldsymbol{J}^T\boldsymbol{\lambda}\}, \end{aligned}$$

where $\boldsymbol{\lambda}$ is the contact impulses and Δt is the simulation time step. The superscript $+$ and $-$ denote the next and the current time-step respectively.

Here we only deal with contact impulses instead of forces. It is implicitly assumed that the impacts occur over Δt under this discretization scheme. We convert the equation to express it in each contact frame by multiplying $\boldsymbol{J}_i$, which is the three rows of $\boldsymbol{J}$ related to i^{th} contact, to both sides as

$$\boldsymbol{J}_i\boldsymbol{u}^+ = \boldsymbol{J}_i\boldsymbol{u}^- + \boldsymbol{J}_i\boldsymbol{M}^{-1}\{\Delta t(\boldsymbol{\tau} - \boldsymbol{h}) + \sum_{k=0}^{N} \boldsymbol{J}_k^T\boldsymbol{\lambda}_k\} \tag{2}$$

$$\boldsymbol{v}_i^+ = \boldsymbol{\tau}_i^* + \boldsymbol{J}_i\boldsymbol{M}^{-1} \sum_{k=0, k\neq i}^{N} \boldsymbol{J}_k^T\boldsymbol{\lambda}_k + \boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{J}_i^T\boldsymbol{\lambda}_i, \tag{3}$$

where $\boldsymbol{\tau}^* \equiv \boldsymbol{J}_i\boldsymbol{u}^- + \boldsymbol{J}_i\boldsymbol{M}^{-1}\Delta t(\boldsymbol{\tau} - \boldsymbol{h})$ is the sum of all constant terms. The index i is for the contact of interest and the index k is for all other contacts. Introducing a new symbol $\boldsymbol{M}_{i,k}^{-1} \equiv \boldsymbol{J}_i\boldsymbol{M}^{-1}\boldsymbol{J}_k^T$, which describes the linear relationship between kth contact impulse and ith contact velocity, we can rewrite the equation as

$$\boldsymbol{v}_i^+ = \boldsymbol{\tau}_i^* + \sum_{k=0, k\neq i}^{N} \boldsymbol{M}_{i,k}^{-1}\boldsymbol{\lambda}_k + \boldsymbol{M}_{i,i}^{-1}\boldsymbol{\lambda}_i. \tag{4}$$

When we are solving a single contact problem assuming that all other contact impulses are constant, we can further simplify Equ. (4) to

$$\boldsymbol{v}_i^+ = \boldsymbol{c}_i + \boldsymbol{M}_{i,i}^{-1}\boldsymbol{\lambda}_i \tag{5}$$

by combining the constant terms and the other contact impulse terms to $\boldsymbol{c}_i$. As a result, we get the contact velocity as a affine function of $\boldsymbol{\lambda}_i$.

The contact impulse solution is based on the following conditions and principles.

- Signorini condition:
  $\lambda_z \geq 0$, $g_i \geq 0$ and $\lambda_z g_i = 0$
- Coulomb's friction cone constraint:
  $\lambda_x^2 + \lambda_y^2 \leq \mu^2\lambda_z^2$ or $|\boldsymbol{\lambda}_{i,T}| \leq \mu\lambda_{i,z}$
- the maximum dissipation principle

The symbol g_i denotes a gap between ith contact bodies, and the subscript $_{i,T}$ represents tangential components (i.e. x-y components) of the ith contact. In this work, we make an additional assumption of pure inelastic contact since elastic contact behaviors are not common in robotics. However, it can be easily extended to different values of coefficient of restitution. We also express the Signorini condition in velocity space for computational simplicity (i.e. $\lambda_z \geq 0$, $v_{i,N} \geq 0$ and $\lambda_z v_{i,N} = 0$). To the best of our knowledge, this simplification is employed in all major simulators. A drift in the position level constraint is possible, but due to the accuracy of our solvers, it was not visible at all in our simulation. The physical realism of this simplification might be questionable in some special cases and interested readers are referred to [8].

The most common friction model is Coulomb's friction law which can be expressed as

$$\text{if } |\boldsymbol{v}_{i,T}^+| > 0 \text{ then } \boldsymbol{\lambda}_{i,T} = -\mu_i\lambda_{i,z}\boldsymbol{v}_{i,T}^+/|\boldsymbol{v}_{i,T}^+|, \tag{6}$$

which simply states that the tangential contact force always opposes the tangential velocity. Many approaches were developed to solve the Coulomb's friction problem, e.g. the projected Gauss-Seidel method [5], bipotential method [9], analytical solution [10], Newton type algorithms, etc. This model is rather difficult to solve due to the non-convexity of the problem [11]. A possible convexification was introduced

by Anitescu et. al. [12] which relaxes the Signorini condition as

$$\boldsymbol{\lambda}_i = \underset{\boldsymbol{\lambda}_i \in \mathcal{F}_i}{\arg\min}\, \boldsymbol{v}_i^{+T} \boldsymbol{M}_{i,i} \boldsymbol{v}_i^{+}. \tag{7}$$

where $\mathcal{F}_i$ is a set of points satisfying the friction cone constraint of the ith contact. By relaxing the Signorini condition, the contact shows bouncing behavior.

A promising alternative to this model was recently introduced in [7]. The idea is to separate the problem into two cases. If the unconstrained motion results in an opening contact, the contact force is trivially zero. If the contact is closing, the author assumes

$$\boldsymbol{\lambda}_i = \underset{\boldsymbol{\lambda}_i \in \mathcal{S}_i}{\arg\min}\, \boldsymbol{v}_i^{+T} \boldsymbol{M}_{i,i} \boldsymbol{v}_i^{+}, \tag{8}$$

where $\mathcal{S}_i$ is the feasible set formed by the velocity space Signorini condition and the Coulomb's friction cone constraint of the ith contact. This model assumes that the normal impulse is also a part of the optimization variable that minimizes the kinetic energy of the contact point. For a single contact, this model leads to convex optimization with a unique solution, under an assumption that the inverse apparent mass matrix $\boldsymbol{M}_{i,i}^{-1}$ is full rank[2].

Equation (6) is generally the better accepted model. It was reported that the results from Equ. (8) is very similar to Equ. (6). in granular simulation [7]. As it will be shown in the discussion section, the solutions of the two models are very similar for a quadrupedal model as well. In this paper, We evaluate both Equ. (6) and Equ. (8) with a different numerical scheme.

III. Bisection Method

We first go through how we can solve a single contact problem described in the Equ. (8) and then extend it to a multi-contact problem with nonlinear block Gauss-Seidel method.

A. Solving for a Contact

In this section, the index i is dropped since we only talk about a single contact. There are three cases for a single contact: an opening contact, a slip and a stick. In the case of an opening contact, the optimal solution is $\boldsymbol{\lambda} = \mathbf{0}_{3\times1}$, where $\mathbf{0}_{3\times1}$ is a 3×1 zero vector. In the case of a stick contact, the optimal solution is $\boldsymbol{\lambda} = -\boldsymbol{M}\boldsymbol{c}$. Since these two cases are trivial, we only discuss a slip case in details.

Assuming a slip on i^{th} contact, we have an optimization problem with the two equality constraints, $h_1(\boldsymbol{\lambda}) = \boldsymbol{M}_{(r3)}\boldsymbol{\lambda} + c_z = 0$ and $h_2(\boldsymbol{\lambda}) = \lambda_x^2 + \lambda_y^2 - \mu^2\lambda_z^2 = 0$, which are the zero normal velocity constraint and the Coulomb's friction cone constraint. Note that the Signorini condition is reduced to the zero normal velocity constraint given that a slip occurs. The subscript $(r3)$ means the third row of the matrix. The gradients of the constraints are $\nabla h_1(\boldsymbol{\lambda}) = \boldsymbol{M}_{(r3)}$ and $\nabla h_2(\boldsymbol{\lambda}) = [2\lambda_x, 2\lambda_y, -2\mu^2\lambda_z]$.

[2]degenerate $\boldsymbol{M}^{-1}$ means that the point is locked in a certain direction. For example, a contact between ground and a link which is connected to the ground by a rotational joint.

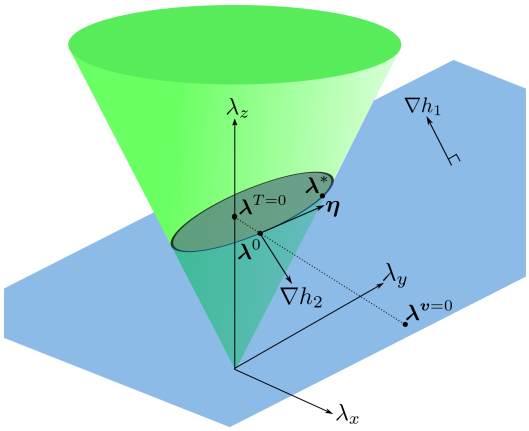

Figure 2: Friction cone (green) and zero normal velocity constraint (blue plane) are shown. The slip condition occurs on the boundary of the intersection.

First order Karush-Kuhn-Tucker (KKT) condition states that, if $\boldsymbol{x}$ is optimal respect to the cost function L, $\partial L/\partial \boldsymbol{x}$ lives only in the constrained space (i.e. span of ∇h's). This simply means that the gradient projected to the unconstrained space is zero and can be mathematically written as

$$\frac{\partial L}{\partial \boldsymbol{x}} \boldsymbol{\eta} = \mathbf{0}, \tag{9}$$

where $\boldsymbol{\eta}$ represents a matrix whose columns are basis vectors pointing to the unconstrained directions. In a single contact dynamics, we have 3 variables with 2 constraints, so $\boldsymbol{\eta}$ becomes a vector as shown in Fig 2. Using the orthogonality to ∇h's, $\boldsymbol{\eta}$ can be computed as

$$\boldsymbol{\eta} = \frac{\nabla h_1(\boldsymbol{\lambda}) \times \nabla h_2(\boldsymbol{\lambda})}{|\nabla h_1(\boldsymbol{\lambda}) \times \nabla h_2(\boldsymbol{\lambda})|}. \tag{10}$$

We assume that this problem has a unique solution since the degenerate $\boldsymbol{M}^{-1}$ leads to a 1D or 2D problem where the solution can be trivially obtained.

In some cases, there might be two vectors satisfying the stationarity condition so it alone is not a sufficient condition for optimality. Fortunately, [7] introduces a simple "line of sight" check that guarantees the global optimality. If a minimum is in line of sight of the zero velocity solution, the solution is globally minimal. Informal proof can be given as following. Let's denote the line connecting any feasible point and the zero-velocity solution as l. If the point is out of sight, it cannot be a global minimal since the intersection between l and the boundary of $\mathcal{S}_i$ has lower cost. In addition, due to the strict convexity of both $\mathcal{S}_i$ and the objective function, the line of sight region is unimodal. So a local minimum in the line of sight region must be a global minimum. A formal proof can be found in [7].

Note that the plane and the boundary of the cone forms a conic section which is the feasible set. Thus the contact problem in Equ. (8) can be transformed into a problem of finding the closest point on the conic section to a point with

a constant metric tensor. We denote this conic section as $\mathcal{C}$. Then, we can rewrite the problem as

$$\boldsymbol{\lambda} = \underset{\boldsymbol{\lambda}\in\mathcal{C}}{\arg\min}\, \frac{1}{2}(\boldsymbol{c} + \boldsymbol{M}^{-1}\boldsymbol{\lambda})^T \boldsymbol{M}(\boldsymbol{c} + \boldsymbol{M}^{-1}\boldsymbol{\lambda}). \tag{11}$$

We change the equation to a more intuitive form

$$\boldsymbol{\lambda} = \underset{\boldsymbol{\lambda}\in\mathcal{C}}{\arg\min}\, \underbrace{\frac{1}{2}(\boldsymbol{M}\boldsymbol{c} + \boldsymbol{\lambda})^T \boldsymbol{M}^{-1}(\boldsymbol{M}\boldsymbol{c} + \boldsymbol{\lambda})}_{E}. \tag{12}$$

$-\boldsymbol{M}\boldsymbol{c}$ is the unconstrained optimum (i.e. zero velocity solution) which is denoted as $\boldsymbol{\lambda}^{\boldsymbol{v}=0}$ in Fig. 2. The minimization is essentially finding the closest point to it in $\mathcal{C}$ with the given metric tensor $\boldsymbol{M}^{-1}$.

Let's denote the intersection between λ_z axis and the zero velocity plane as $\boldsymbol{\lambda}^{T=0}$. We define a projection $\mathcal{E}_{\mathcal{C}}(\cdot)$, which maps any point $\boldsymbol{\lambda}$ on the zero velocity plane to the intersection between the conic section and the line connecting $\boldsymbol{\lambda}^{T=0}$ and $\boldsymbol{\lambda}$. This mapping is illustrated in Fig. 2.

We use $\boldsymbol{\lambda}^0 = \mathcal{E}_{\mathcal{C}}(\boldsymbol{\lambda}^{\boldsymbol{v}=\boldsymbol{0}})$ as an initial guess of our numerical solver. Note that the projected point is always dissipative since it is closer to the zero velocity point than the center point $\boldsymbol{\lambda}^{T=0}$. Strict dissipativity of the center point can be proven by substituting the center point into the energy formula $(\boldsymbol{M}\boldsymbol{c} + \boldsymbol{\lambda})^T \boldsymbol{M}^{-1}(\boldsymbol{M}\boldsymbol{c} + \boldsymbol{\lambda})$ but we do not show this here due to its simplicity. This initial guess is globally optimal only in some special cases (e.g. if $\boldsymbol{M}$ can be written in a form $c\mathbf{I}$ where c is a positive real number and $\mathbf{I}$ is an identity matrix).

We work in polar coordinates for computational benefits. The tangential impulse can be represented as $[r\cos(\theta), r\sin(\theta)]$. Then the projection $\mathcal{E}_{\mathcal{C}}(\boldsymbol{\lambda}^{\boldsymbol{v}=\boldsymbol{0}})$ is reduced to $\theta^0 = \text{atan2}(\lambda_y^{\boldsymbol{v}=\boldsymbol{0}}, \lambda_x^{\boldsymbol{v}=\boldsymbol{0}})$.

The bisection method requires the sign of the gradient at every mid point. We tested three different methods to compute the sign of the gradient. The most naive way is computing $\frac{dE}{d\boldsymbol{\lambda}}\boldsymbol{\eta}$. It has the same sign as $\frac{dE}{d\theta}$. Another way is changing to polar coordinates and compute $\frac{dE}{d\theta}$ directly as described in [7]. Although this method is probably the best way to find the energy directly, it actually leads to more computation cost than the following hybrid form.

The z-component of the impulse can be obtained from our zero normal velocity constraint $\boldsymbol{M}^{-1}_{(r3)}\boldsymbol{\lambda} + c_{(z)} = 0$. After a few algebraic manipulation, we get

$$\lambda_z = \frac{-c_{i(z)} - M^{-1}_{(zx)} r\cos(\theta) - M^{-1}_{(zy)} r\sin(\theta)}{M^{-1}_{(zz)}}. \tag{13}$$

Now we substitute the above equation to the slipping impulse condition $\mu\lambda_z = \sqrt{\lambda_x^2 + \lambda_y^2} = r$, we can express r as a function of θ as,

$$r = \frac{-c_{(z)}}{\frac{M^{-1}_{(zz)}}{\mu} + M^{-1}_{(zx)}\cos(\theta) + M^{-1}_{(zy)}\sin(\theta)}. \tag{14}$$

In the bisection method, we only compute θ and r without computing the full impulse. The sign of the gradient along $\boldsymbol{\eta}$ can be computed only with the tangential impulse as

$$\frac{dE}{d\boldsymbol{\lambda}}\breve{\boldsymbol{\eta}} = (\boldsymbol{M}^{-1}\boldsymbol{\lambda} + c)\breve{\boldsymbol{\eta}} = (\bar{\boldsymbol{M}}^{-1}\boldsymbol{\lambda}_T + \bar{c})\breve{\boldsymbol{\eta}}, \tag{15}$$

where $\breve{\boldsymbol{\eta}}$ any vector with the same direction as $\boldsymbol{\eta}$. $\breve{\boldsymbol{\eta}}$ results from skipping the normalization steps of $\boldsymbol{\eta}$.

The hybrid properties $\bar{\boldsymbol{M}}^{-1} \in \mathbb{R}^{3\times 2}$ and $\bar{c} \in \mathbb{R}^3$ can be found by substituting Equ. (13) to the above equation as

$$\begin{aligned} \bar{\boldsymbol{M}}^{-1} &= \boldsymbol{M}^{-1}_{(c1,c2)} - \boldsymbol{M}^{-1}_{(c3)}\Big[\frac{M^{-1}_{(zx)}}{M^{-1}_{(zz)}}, \frac{M^{-1}_{(zy)}}{M^{-1}_{(zz)}}\Big] \\ \bar{c} &= c - \boldsymbol{M}^{-1}_{(c3)}(c_{(z)}/M^{-1}_{(zz)}). \end{aligned} \tag{16}$$

We observed that this hybrid method is about 30 % more efficient than computation of $\frac{dE}{d\boldsymbol{\lambda}}\boldsymbol{\eta}$ and about 10 % more efficient than full polar coordinate formulation.

Starting from the initial guess, we perform the bisection method, which finds a root of (9). The bisection method first performs an incremental stepping which moves the estimated minimum in the opposite direction of the projected gradient. This is to find an interval in where the global minimum lies. We always check the line of sight and positiveness of r to make sure that the global minimum actually lies in that interval.

The bisection part of the algorithm is simple. We bisect the angle and compute the projected gradient at the mid point. The new interval is set such that the global minimum lies within in. We stop when the desired precision is met.

The pseudocode is shown in Algorithm 1. For clarity, it shows a simplified version of the contact solver. In the actual solver, many normalization steps are skipped and the aforementioned hybrid method of computing the gradient is employed.

B. Expansion to multi-contact

Once the per-contact solution is obtained, it is easy to expand to multi-contact cases. Inspired by the existing contact solvers, we use nonlinear block Successive Over-Relaxation (SOR) method [13] to handle a multi-contact problem. Nonlinear block SOR method only takes a partial step toward the solution in coupled problems so that the solver converges more reliably. We use a successive relaxation of a form $\boldsymbol{X} \leftarrow \alpha\boldsymbol{X}^* + (1-\alpha)\boldsymbol{X}$, where $\boldsymbol{X}$ is the current estimated solution, $\boldsymbol{X}^*$ is the optimal solution assuming that the other coupled variables are fixed and α is an over-relaxation parameter. Algorithm 2 summarizes the overall algorithm. The algorithm essentially 1) finds an optimal solution for a single contact 2) update the current solution as a weighted average of the current solution and the optimal solution and 3) decays the relaxation factor.

Since we compute the numerically accurate solution for every contact, we can compute how much the current solution violates the physical constraints as well. Therefore, we can have a reliable terminal condition.

IV. Projected Gauss-Seidel

The bisection method introduced in the previous section iterates over contacts. Another possible scheme is an iteration over each force component as in the PGS method [6]. The PGS method for purely inelastic collisions can be written as,

$$\begin{aligned} \lambda_{i,z} &\leftarrow \text{prox}_z(\lambda_{i,z} - r_{i,z}v_{i,z}) \\ \boldsymbol{\lambda_{i,T}} &\leftarrow \text{prox}_T(\boldsymbol{\lambda}_{i,T} - r_{i,T}\boldsymbol{v}_{i,T}). \end{aligned} \tag{17}$$

Algorithm 1 Bisection method for a single slip contact

Given hyperparameters β_1, β_2, β_3 and γ
$\theta \leftarrow \angle \boldsymbol{\lambda}_T^0$
Find r and λ_z as per Equ. (14) and (13)
$\boldsymbol{\lambda} \leftarrow [r\cos(\theta), r\sin(\theta), \lambda_z]^T$
$D_0 \leftarrow \frac{dE}{d\boldsymbol{\lambda}}\boldsymbol{\eta}$
$\alpha \leftarrow -\beta_1 \text{sgn}(D_0)$
loop ▷ Initial stepping
  $\theta^p \leftarrow \theta$ ▷ Store old angle
  $\boldsymbol{\lambda}^p \leftarrow \boldsymbol{\lambda}$
  $\theta \leftarrow \theta + \alpha$
  Find r and λ_z as per Equ. (14) and (13)
  $\boldsymbol{\lambda} \leftarrow [r\cos(\theta), r\sin(\theta), \lambda_z]^T$
  if $\nabla h_2(\boldsymbol{\lambda}) \cdot (\boldsymbol{\lambda} - \boldsymbol{\lambda}^{\boldsymbol{v}=0}) < 0$ or $r < 0$ **then**
    $\alpha \leftarrow \beta_2 \alpha$ and $\theta \leftarrow \theta^p$ ▷ back to old param
  else
    Find $\frac{dE}{d\boldsymbol{\lambda}}\boldsymbol{\eta}$ as per Equ. (15)
    if $(\frac{dE}{d\boldsymbol{\lambda}}\boldsymbol{\eta})D_0 > 0$ **then** ▷ Zero-crossing
      $\alpha \leftarrow \beta_3 \alpha$
    else
      break
    end if
  end if
end loop
repeat ▷ Bisection
  $\theta^{\triangle} \leftarrow \frac{1}{2}(\theta + \theta^p)$ ▷ Compute mid-point
  Find $r^{\triangle}$ and $\lambda_z^{\triangle}$ as per Equ. (14) and (13)
  $\boldsymbol{\lambda}^{\triangle} \leftarrow [r^{\triangle}\cos(\theta^{\triangle}), r^{\triangle}\sin(\theta^{\triangle}), \lambda_z^{\triangle}]^T$
  if $(\frac{dE}{d\boldsymbol{\lambda}}|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^{\triangle}}\boldsymbol{\eta}^{\triangle})D_0 > 0$ **then**
    $\theta^p \leftarrow \theta^{\triangle}$
  else
    $\theta \leftarrow \theta^{\triangle}$
  end if
until $|\boldsymbol{\lambda}^p - \boldsymbol{\lambda}| < \gamma$

Algorithm 2 Solving multi-contact dynamics

while $\boldsymbol{\lambda}$ not converged **do**
  for each active contacts **do**
    if $c_{i(z)} > 0$ **then** ▷ if opening
      $\boldsymbol{\lambda}_i \leftarrow (1-\alpha)\boldsymbol{\lambda}_i$
    else if $\mu\lambda_{i,z}^{\boldsymbol{v}=0} \geq |\boldsymbol{\lambda}_{i,T}^{\boldsymbol{v}=0}|$ **then** ▷ if sticking
      $\boldsymbol{\lambda}_i \leftarrow \alpha\boldsymbol{\lambda}_i^{\boldsymbol{v}=0} + (1-\alpha)\boldsymbol{\lambda}_i$
    else ▷ if slipping
      Compute $\boldsymbol{\lambda}_i^*$ as per Algorithm 1
      $\boldsymbol{\lambda}_i \leftarrow \alpha\boldsymbol{\lambda}_i^* + (1-\alpha)\boldsymbol{\lambda}_i$
    end if
  end for
  $\alpha \leftarrow \alpha_{min} + \gamma(\alpha - \alpha_{min})$
end while

where $\text{prox}_z(\cdot)$ is equivalent to $\max(0, \cdot)$ and $\text{prox}_T(\cdot)$ is an Euclidean projection onto the Coulomb's friction disc given the current estimate of λ_z. The constants r_z and r_T can be tuned using the apparent inertia matrix as

$$r_{i,z} = \frac{\alpha}{M^{-1}_{i,i(zz)}} \text{ and } r_{i,T} = \frac{\alpha}{\max(M^{-1}_{i,i(xx)}, M^{-1}_{i,i(yy)})}, \quad (18)$$

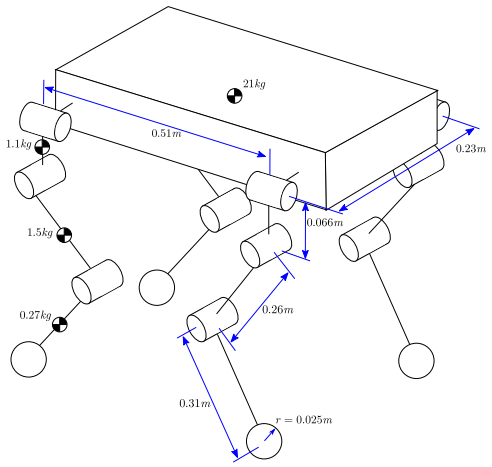

Figure 3: ANYmal has 4 legs, 12 joints and 4 feet. The schematic is simplified for clarity.

where the subscript (xx) represent the row and column indices.

The PGS method has been proven to be effective in granular matter simulations in where thousands or sometimes millions of spherical objects interact with each other. We also believe that the PGS method is the best option when we only deal with spherical objects since their apparent inertia matrix is diagonal assuming homogeneity of the density. However, the focus of this paper is on robotic systems. Convergence of the PGS method can be extremely slow when the off-diagonal elements in the Delassus matrix become large [14].

V. Numerical Experiments

Here we present numerical experiments to demonstrate the performance of the proposed bisection method.

A. Experimental Setup

Our interest lies in the field of robotics, especially in articulated robots. Therefore, we chose a model of ANYmal [15], illustrated in Fig. 3, to validate the proposed bisection method and compare it to the PGS method. ANYmal is a quadrupedal robot weighing about 35 kg. The collision geometry is defined only at the four feet and the torso for simplicity. The feet are modeled as spheres and the torso is modeled as a box. We test the following four scenarios:

- Trotting: The model of ANYmal runs forward at 1.5 m/s with its legs moving in diagonal pairs. The maximum possible number of contacts is 4.
- Hanging: The model of ANYmal is thrown in the air with zero initial velocity and 1 m height. The unilateral constraints keep only the feet (not the torso) above the ground, emulating upside down hanging behavior. The maximum possible number of contacts is 4.
- Drop: The model of ANYmal is thrown into the air with 1 m/s initial tangential velocity and 1 m initial height and the unilateral constraints keep both the feet and the torso above the ground. The simulation resets when there is

	Bisection	Golden Section	Analytical
Costs (μs)	0.58 μs	1.2 μs	1.6 μs

Table I: Computation cost of a single contact optimization.

no significant motion observed. The maximum possible number of contacts is 8.

- Random: The model of ANYmal is thrown into the air. The joints are controlled by a PD controller and the reference inputs are given randomly with 1 rad^2 variance. The simulation resets every 5 s. The maximum possible number of contacts is 8. Note that this is a process of collecting data for reinforcement learning, which is implemented in RAI framework [16].

All experimental results are from a single framework and only the relevant parts of the code are modified to implement different algorithms. All numerical computations are based on Eigen3 library [17] except symmetric positive definite matrix inversion using Cholesky factorization. We used our custom implementation which seems to be faster by about 30 %. One core of Intel Xeon E5-1620 (3.6 GHz), which is not a high-end cpu in today's standard, is used for all experiments. For time measurement, `CLOCK_MONOTONIC` of the posix-timers under Linux was used. For computing articulated-body dynamic properties, we use RBDL library[3] which is based on the algorithms described in [18]. We use Δt =1 ms for all experiments. For unactuated tasks, the simulation is stable even when a very large time step (e.g. Δt =20 ms) is employed but the accuracy is highly compromised with such a large time step.

B. Single-Contact Solver Performance

Preclik et. al. suggested two solvers of solving the same contact model [7], [19], namely the golden section method and the analytical method. We compare the proposed bisection method against these two methods.

The golden section uses the values of a function whereas the bisection method uses the sign of the gradient. Using the hybrid gradient computation method described in Sec. III-A, the computation of the sign of the gradient is just 10 % more expensive than the value. For the analytical method, we tested four free quartic solvers available online and picked the fastest one. We further had to modify the code so that it is seamlessly integrated to our code and inlined in the main solver loop. In addition to the quartic solver, we also integrated coefficient computation and all necessary checks described in [19].

We ran all four tasks and collected 100,000 samples from each task. We terminated the optimization when the impulse interval goes below 10^{-6} Ns for both the bisection and the golden section method. The computation costs of each solver is shown in Tab.I.

C. Multi-Contact Solver Performance

For the bisection method, we set a relaxation factor $\alpha = 1$ with a decay factor $\gamma = 0.99$ and a minimum relaxation factor $\alpha_{min} = 0.7$. For the PGS method, we set $\alpha = 0.6$ and keep it constant. The parameters here are fairly tuned to show good performance in all tasks. We could not use the same relaxation factor scheduling on the PGS method since recomputing r_z and r_T resulted in a significant increase in the computation cost without much performance gain.

[3]https://rbdl.bitbucket.io/

	Trot	Hang	Drop	Random
Bisection (s)	1.98	2.57	2.62	1.90
PGS (s)	1.81	5.92	4.35	2.88

Table II: Computation cost per 100k time steps

The two algorithms have a different termination condition so it is tricky to compare them fairly. The PGS method computes the error by summing the magnitude of the impulse updates and the norms of the penetration velocities. The bisection method computes the error by adding the exact violations of the physical constraints. Therefore, the termination condition for the bisection method is much more strict and reliable. We terminate when the error is below 1 μNs or μm/s unless otherwise specified.

The rendering of the simulation is shown in Fig. 1 to aid understanding of the experiment. Fig. 5 and 4 show the computation cost and the number of iterations taken by each algorithm. Since all the code include logging, the total computation time does not necessarily reflect the true computation time taken by the simulation. We perform the same test without logging, and the total computation time is shown in Table. II. Since we do not have the ground truth, it is hard to measure the validity of the simulation. However, we provide two statistics here. First, we measure the difference between the two solvers. While we simulate with the bisection method, we ran the PGS solver as well and measure the relative difference as $\frac{|\lambda_b - \lambda_p|}{|\lambda_p|}$. The histogram of the samples of the differences is shown in Fig. 6. Here we set the termination threshold of PGS to 10^{-20} to make sure that we are measuring the correct difference.

Another meaningful measure of accuracy is the penetration depth of the feet, which is shown in Fig. 7. We get the meaningful penetration depth only in task 2 since the other tasks are actuated and the feet often leaves the ground. In task 2, the robot still swings around due to the conservation of energy but the feet stays on the ground all the time except the first two second when the robot falls.

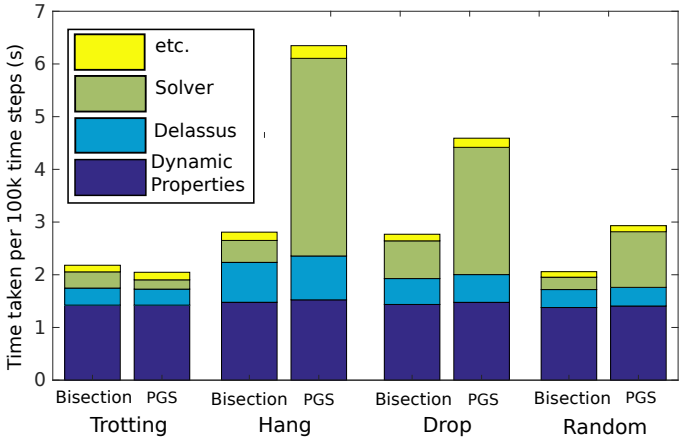

Figure 5: Computational costs on 4 different tasks averaged over 5 runs are shown.

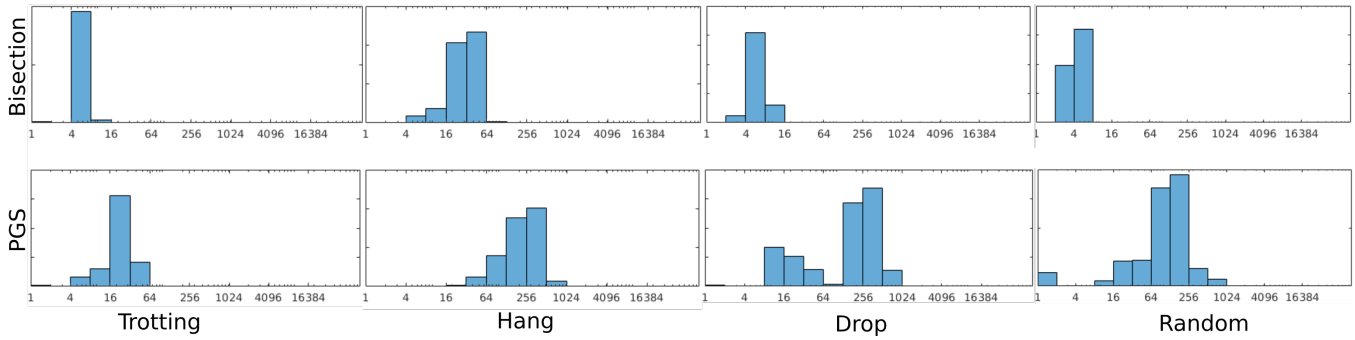

Figure 4: Number of iterations taken by the bisection method and the PGS method are shown. The bin edges increase by a factor of two (log scale). The y-axis represents the percentage. In average, the bisection method takes significantly less iterations to solve a contact problem.

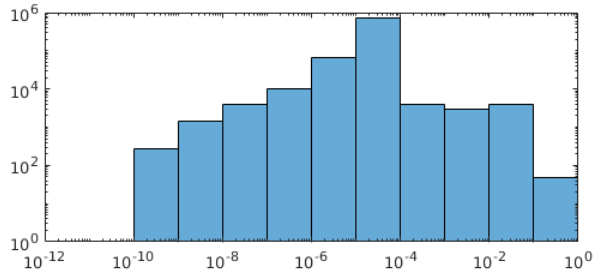

Figure 6: Error between Coulomb's friction model and the maximum dissipation model.

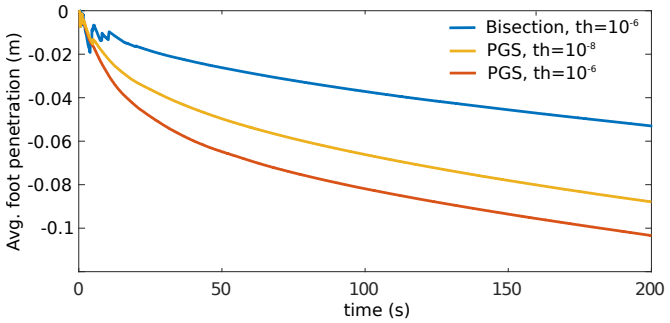

Figure 7: Average penetration depth of the feet in task 2 is shown. PGS method is tested with two different termination threshold levels.

VI. Discussion

We found a significant performance improvement with the proposed bisection method in a single contact problem. There are a few reasons why it outperforms the golden section method. First, the interval size decreases by 50 % for the bisection and 38 % for the golden section method. Second, the proposed bisection method starts with a good guess which is a Euclidean projection of the zero-velocity condition, whereas the golden section method proposed in [7] searches within the two end points of the line-of-sight region. In most cases, we found that the Euclidean projection to be near the optimal solution and the bisection method converges in just two steps. Lastly, following the method presented in [7], the golden section method has an initial overhead to find the interval which is quite significant. The proposed bisection method just requires a single `atan2` call for initialization.

The analytical method was the slowest. In addition to its quartic solver, the computation of the coefficients and the post checking processes combined took as much as the quartic solver.

We observed that the inverse apparent inertia matrix at the feet is almost always diagonal-dominant when ANYmal is trotting. It is not very surprising to see the PGS method performing the best in such cases since it only involves a few elementary operations per iteration. Considering the computation cost of the solver only, the PGS was about twice faster. However, the computation cost of the overall algorithm was not significantly different since the total time was dominated by the computation cost of the dynamic properties (updating the Jacobians, the mass matrix and the non-linearities).

The performance of PGS algorithm quickly drops when the kinematics is not favorable. It is outperformed by the bisection method by a significant margin for hanging and drop tests. As shown in Fig. 4, the iteration taken by the PGS method is about a magnitude higher than that by the bisection method. We also noticed that the PGS method becomes extremely slow at the impacts. In the hang and drop task, the PGS took more than 20,000 iterations to solve the contact problem at the impact. This was the main reason that the performance gap between the two methods became very large in the two tasks.

In the random motion task, the bisection method again outperformed the PGS method. Considering the solver time only, the bisection method was about 4 times faster. Considering the total simulation time, the bisection method took about 35 % less time. We suspect that the big performance gap is due to the frequent slippages at the feet which are not observed in the trotting task.

In terms of the penetration depth, the bisection method seems much more accurate than the PGS method at the same termination threshold as shown in Fig 7. Even when the threshold is lowered to 10^{-8}, the PGS was outperformed by the bisection method. The tighter termination condition was translated to about 50 % increase in solver time for the PGS. Note that since the system is nonlinear and discrete, the penetration in unavoidable. Practically, this rate of penetration can be adjusted by introducing a small spring term.

The difference between the two models is shown in Fig. 6. More than 99.6 % of the time, the difference remained below 1 %. The resulting motions were visibly indistinguishable as well. This shows that the two models are highly similar, supporting the claims made in [19].

We believe there is enough room to improve the computation of the dynamic properties and the Delassus matrix using a faster implementation and vectorization. This will make the performance gap between the two solvers more significant. Although it is not within the scope of this paper, we found that a proper implementation of Featherstone's algorithms outperforms the RBDL by a factor of 4.

It was unexpected that the computation cost of the Delassus operator became significant. It was as expensive as the contact solvers. In a four-contact case, computing the Delassus operator costs 6,480 double precision floating point multiplications and a similar number of addition operations. The high number of instructions is due to its quadratic relationship to both the DOF (18) and the number of contacts.

We do not believe that it is easy to prove the convergence of our method just like most of the methods that are based on the nonlinear Gauss-Seidel method. The solver is extensively tested ($>$100 years in simulation time) through our reinforcement learning frameworks [16] with two different robotic models, ANYmal and a humanoid model. No convergence issue was observed yet. However, the robustness of the solver has to be tested in more broad varieties of robotic systems in the future.

The solution of the impulse might not be unique when there can be a residual force. This is a well-known problem in contact dynamics. In real life, the residual force is highly dependent on the geometry and the rigidity of the material. The proposed solver is just computing one of all feasible solutions. However, the velocity solution is unique if the solver fully converges.

VII. Conclusion

We introduced the bisection method for solving contact dynamics which accounts for the non-diagonal dominant nature of the inverse apparent mass matrix. We found that it outperforms the two existing methods for a single contact problem at least by a factor of two. In addition, we fairly compared it against the PGS method on four robotic simulation tasks. The proposed method showed fast and consistent performance on all tasks whereas the PGS became significantly slow on complicated tasks. Our result shows that the proposed method can be a promising alternative to the existing contact solvers which employ many relaxations of the physical principles.

References

[1] N. Chakraborty, S. Berard, S. Akella, and J. C. Trinkle, "A geometrically implicit time-stepping method for multibody systems with intermittent contact," *The International Journal of Robotics Research*, vol. 33, no. 3, pp. 426–445, 2014.

[2] S. Berard, B. Nguyen, K. Anderson, and J. Trinkle, "Sources of error in a simulation of rigid parts on a vibrating rigid plate," *Journal of Computational and Nonlinear Dynamics*, vol. 5, no. 4, p. 041003, 2010.

[3] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[4] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[5] T. Liu and M. Y. Wang, "Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods," *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 1, pp. 19–31, 2005.

[6] C. Studer and C. Glocker, "Solving normal cone inclusion problems in contact mechanics by iterative methods," *Journal of System Design and Dynamics*, vol. 1, no. 3, pp. 458–467, 2007.

[7] T. Preclik, "Models and algorithms for ultrascale simulations of non-smooth granular dynamics," Ph.D. dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2014.

[8] A. Chatterjee, "On the realism of complementarity conditions in rigid body collisions," *Nonlinear Dynamics*, vol. 20, no. 2, pp. 159–168, 1999.

[9] G. De Saxcé and Z.-Q. Feng, "The bipotential method: a constructive approach to design the complete contact law with friction and improved numerical algorithms," *Mathematical and computer modelling*, vol. 28, no. 4-8, pp. 225–245, 1998.

[10] O. Bonnefon and G. Daviet, "Quartic formulation of coulomb 3d frictional contact," Ph.D. dissertation, INRIA, 2011.

[11] M. Anitescu and G. D. Hart, "A fixed-point iteration approach for multibody dynamics with contact and small friction," *Mathematical Programming*, vol. 101, no. 1, pp. 3–32, 2004.

[12] M. Anitescu and G. Hart, "A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction," *International Journal for Numerical Methods in Engineering*, vol. 60, no. 14, pp. 2335–2371, 2004.

[13] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[14] T. Heyn, M. Anitescu, A. Tasora, and D. Negrut, "Using krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation," *International Journal for Numerical Methods in Engineering*, vol. 95, no. 7, pp. 541–561, 2013.

[15] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 38–44.

[16] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[17] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[18] R. Featherstone, "Rigid body dynamics," 2008.

[19] T. Preclik, S. Eibl, and U. Rüde, "The maximum dissipation principle in rigid-body dynamics with purely inelastic impacts," *arXiv preprint arXiv:1706.00221*, 2017.