Variations of Augmented Lagrangian for Robotic Multi-Contact Simulation

Jeongmin Lee, Minji Lee, Sunkyung Park, Jinhee Yun, and Dongjun Lee

***Abstract*—The multi-contact nonlinear complementarity problem (NCP) is a naturally arising challenge in robotic simulations. Achieving high performance in terms of both accuracy and efficiency remains a significant challenge, particularly in scenarios involving intensive contacts and stiff interactions. In this article, we introduce a new class of multi-contact NCP solvers based on the theory of the Augmented Lagrangian (AL). We detail how the standard derivation of AL in convex optimization can be adapted to handle multi-contact NCP through the iteration of surrogate problem solutions and the subsequent update of primal-dual variables. Specifically, we present two tailored variations of AL for robotic simulations: the Cascaded Newton-based Augmented Lagrangian (CANAL) and the Subsystem-based Alternating Direction Method of Multipliers (SubADMM). We demonstrate how CANAL can manage multi-contact NCP in an accurate and robust manner, while SubADMM offers superior computational speed, scalability, and parallelizability for high degrees-of-freedom multibody systems with numerous contacts. Our results showcase the effectiveness of the proposed solver framework, illustrating its advantages in various robotic manipulation scenarios.**

***Index Terms*—Contact modeling, simulation and animation, dynamics, dexterous manipulation**

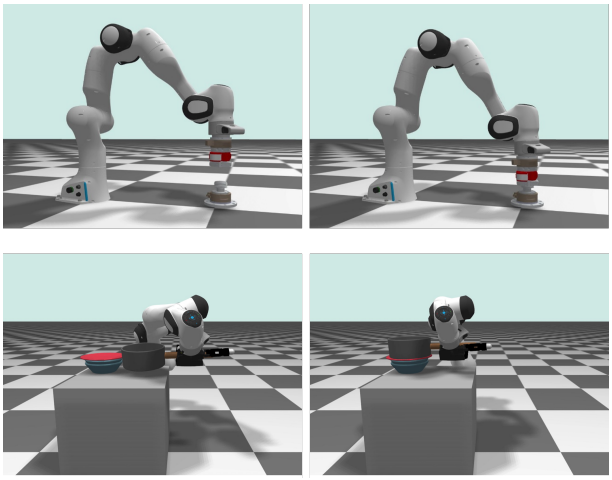

Fig. 1. Snapshots of a robotic simulation using our multi-contact solver. Top: bolt-nut assembly. Bottom: dish piling. Although intensive contact formation and stiff interactions make these scenarios challenging to simulate, our solvers successfully complete the simulations less than a ms of time budget per step.

I. Introduction

Physics simulation is a fundamental tool for the development of robotic intelligence, as it enables scalable data acquisition, training, and safe testing of various algorithms and designs. Moreover, simulations can be directly employed to solve modeled system dynamics, proving invaluable for a range of applications such as global planning, trajectory optimization, and parameter estimation. This significance has led to the development of diverse open-source platforms [1]–[6], which are increasingly being utilized in various research endeavors.

An essential focus in robotic simulation research revolves around achieving results that are both accurate and efficient in terms of memory and computation time. This presents a comprehensive and challenging problem, encompassing diverse considerations such as discrete-time integration, defining various geometric/physical constraints, incorporating friction, managing system-induced sparsity, and selecting numerical algorithms. Among these factors, multi-contact plays a crucial role in mimicking interactions between objects. A prevalent velocity-level modeling of such constraints [7] naturally induces a nonlinear complementarity problem (NCP), which is generally challenging to solve.

Typically, contact solvers for physics simulations must balance three crucial factors: efficiency, accuracy, and robustness. However, finding a universal solution remains challenging. Methods developed for graphics and game engines tend to prioritize efficiency and robustness, aiming to deliver visually plausible results, even if early termination occurs. However, they are known to converge slowly and may struggle with achieving highly accurate solutions. They frequently encounter difficulties in handling intensive contact interactions (i.e., where constraints are dense and numerous relative to the system degrees of freedom), which is common in robotic manipulation. Conversely, achieving a highly accurate solution for NCP often involves complex matrix operations and numerically sensitive processes, which generally lack efficiency and robustness for practical robotic applications. Moreover, some approaches aim to enhance efficiency and robustness by relaxing the contact constraints and exploiting them during the solving stage. However, such relaxations can be challenging to physically interpret, and the solutions they produce may exhibit undesirable physical behaviors.

In this article, we introduce a new series of multi-contact solvers for robotic simulation based on the theory of augmented Lagrangian (AL). We demonstrate how the variations of AL can address the multi-contact NCP for robotic sim-

This research was supported by the National Research Foundation (NRF) grant funded by the Ministry of Trade, Industry & Energy (MOTIE) of Korea (RS-2024-00419641), and the Ministry of Science and ICT (MSIT) of Korea (RS-2022-00144468). Corresponding author: Dongjun Lee.

The authors are with the Department of Mechanical Engineering, IAMD and IOER, Seoul National University, Seoul, Republic of Korea. {ljmlgh,mingg8,sunk1136,yjhs0932,djlee}@snu.ac.kr.

ulations, by iteratively solving surrogate problems, thereby enabling the proximal solution converges in a stable and robust manner. Specifically, we present two algorithms that are practically applicable to robotic simulation: the cascaded Newton-based Augmented Lagrangian method (CANAL) and the subsystem-based Alternating Direction Method of Multipliers (SubADMM). We explain how these two variations are advantageous in scenarios requiring precise management of high-density intensive contact and parallelized, scalable handling of high degree of freedom (DOF) multibody contact, respectively. Several robotic simulations, particularly those involving challenging multi-contact scenarios, are implemented and demonstrated to validate our framework.

The rest of the article is structured as follows. In Sec. II we review the development and utilization of multi-contact solvers in robotic applications and beyond. Sec. III provides essential background materials necessary to present our AL-based multi-contact solver. Then, Sec. IV presents our core theories and structures for the AL-based multi-contact solver. This leads to Sec. V, which outlines the first practical variation as the cascaded Newton-based AL, and Sec. VI, which introduces the other variation: subsystem-based ADMM. Sec. VII illustrates the implementation results of our solver in physics simulation and evaluates its performance under various robotic manipulation scenarios. Finally, Sec. VIII conclude the article with discussions and remarks.

II. Related Works

In this section, we summarize the multi-contact modeling and solver algorithms that have been utilized in robotic simulation. See also Table I for the comparison of widely used simulators in robotics.

TABLE I
Comparison of contact models and solvers used in popular robotic simulators.

	Bullet	MuJoCo	DART	PhysX	Drake	ODE
Model	LCP	Convex	LCP	NCP	Convex	LCP
Solver	Direct PGS	Newton CG PGS	Direct PGS	PGS TGS	Newton	Direct PGS

A. Direct Method

The conventional approach to handling dynamics equations with multi-contact constraints involves formulating the equations as a linear complementarity problem (LCP) [8] then applying Lemke's algorithms [9] or Dantzig's pivoting algorithms. While these direct methods can guarantee accuracy, they often suffer from high computational complexity. Moreover, the LCP-based formulation necessitates polygonal friction cone approximation, leading to undesirable error in friction behavior. In robotic simulation software, DART [10], ODE [11], and Bullet [1] provide implementations of Dantzig's method to solve the LCP problem.

B. Per-Contact Iteration

More widely used in recent years are iterative methods, which typically involve locally performing an impulse projection step to achieve global equilibrium. One of the most popular iteration schemes is projected Gauss-Seidel (PGS), which has been extensively developed and adopted in the game and graphics community [12], [13] as well as in robotics [14], [15]. These methods are known for being simple, robust, and advantageous in generating visually plausible results. However, they often experience slow convergence and limited efficiency, especially when the constraints are highly coupled. These weaknesses are particularly emphasized in robotic simulation, as the generalized coordinate representation (e.g., robot joint angles) is common, and over-specified contact (i.e., system DOF $<$ constraint DOF) is prevalent in manipulation tasks. Several research efforts have aimed to enhance the performance of impulse iteration methods. In [16], the bisection method is presented as a potential replacement for the local projection scheme in PGS, demonstrating its effectiveness in quadruped locomotion simulation. Additionally, a substepping variant of PGS, named temporal Gauss-Seidel (TGS), is introduced in [17], showing its better convergence in various situations. Unlike direct methods, iterative methods can be applied to various types of problem modeling, including LCP, cone complementarity problems (CCP), nonlinear complementarity problems (NCP), and also their position-based dynamics (PBD) variants [18]. As a result, they are employed in a wide range of simulation software, including Bullet [1], MuJoCo [2], RaiSim [5], and Isaac Sim [19].

C. Nonlinear Equation

Another approach to dealing with multi-contact simulation is to express all required relations in nonlinear equation form and solve them using gradient descent iteration. Implicit penalty-based contact, often referred to as regularized contact, exhibits the most natural connection to this approach, as demonstrated in [20], [21]. However, penalty methods have well-known weaknesses that they often necessitate parameter tuning to achieve plausible results, and high penalty gains can lead to numerical issues. For the other direction, in [22] construct and solve a nonlinear equation with complementarity smoothing, and [23] we derived a nonsmooth equation using the complementarity function (e.g., Fischer-Burmeister). While these methods typically exhibit superlinear convergence, the intricate nature of contact conditions frequently leads to lack of robustness or challenges in line search. Addressing this issue, the Newton-based techniques [24] and conjugate gradient (CG) algorithm for regularized convex contact models aim to ensure algorithmic robustness, albeit at the potential expense of physical accuracy. Among current simulation software, MuJoCo and Drake [25] are incorporating nonlinear equation-based solvers.

D. Augmented Lagrangian

Proximal algorithms, which were possibly pioneered by Moreau [26] comprise a class of methods designed to address constrained convex optimization problems by sequentially

solving a series of subproblems. The augmented Lagrangian (AL) method can be viewed as a class of proximal algorithm [27], as it formulates subproblems using the method of multipliers and a penalty term. Typically, the subproblems are addressed through simpler solutions or tailored designs, which has spurred the development of numerous open-source libraries that implement these strategies, thereby facilitating broader access to robust optimization tools. Notable examples include libraries for quadratic programming, such as OSQP [28] and QPALM [29], and for cone programming, such as SCS [30]. In robotics, proximal algorithms have been effectively utilized to address constraints within computational structures, notably in applications such as factor graph optimization [31] and differentiable dynamics programming [32]. The utility in solving robot dynamics with equality constraints is presented in [33]. Our previous work [34] presents a specific algorithm based on the Augmented Lagrangian (AL) method to achieve effective parallelization in contact simulations. Building upon this foundation, this article extends the general theory and variations of AL designed to handle robotic simulations involving contact.

III. Preliminary

A. Discretized Dynamics

We consider following continuous-time equations of motion:

$$M(q)\ddot{q} = f(q,\dot{q}) + J(q)^T\lambda \tag{1}$$

where $q \in \mathbb{R}^n$ is the generalized coordinate variable of system, $M(q) \in \mathbb{R}^{n\times n}$ is the system mass matrix, $f \in \mathbb{R}^n$ is the generalized force (including Coriolis/gravitational force, external input, etc.) and $\lambda \in \mathbb{R}^{n_c}, J(q) \in \mathbb{R}^{n_c\times n}$ are the constraint impulse and Jacobian with n, n_c being the system/constraint dimension. In typical robotic simulation, the discretized version of the equation (1) is employed:

$$\begin{aligned} M_k(v_{k+1} - v_k) &= f_k t_k + J_k^T\lambda_k \\ \hat{v}_k &= \theta v_k + (1-\theta)v_{k+1} \\ q_{k+1} &\leftarrow \text{update}(q_k, \hat{v}_k, t_k) \end{aligned} \tag{2}$$

where k denotes the time step index, t_k is the step size, and the v_k is the generalized velocity at the k-th step. In this work, we primarily integrate explicit and implicit schemes. Specifically, we utilize $M_k = M(q_k)$ and $f_k = f(q_k, v_k)$, while employing the representative mid-step velocity $\hat{v}_k \in \mathbb{R}^n$ for state updates and constraint handling. Here, $\theta \in [0,1]$ determines the precise integration rule, while its impact on physical behavior is discussed in [35]. From now on, time step index k will be omitted for simplicity but note that all components are still time(step)-varying.

B. Constraint Models

Throughout this article, we classify the constraints on the multibody system into three categories: soft, hard, and contact constraints. Similar to many other simulators [1], [2], [5], the constraint model can be formulated by relation between the velocity $\hat{v}$ to the impulse λ. Such velocity-impulse modeling has advantages in terms of the well-definedness of the problem (c.f., the Painleve paradox [36]) and can naturally express behaviors like friction or elastic collisions. However, it may exhibit position-level drift, as it is based on linearization on the constraints. Positional drift can be suppressed by adopting techniques such as multiple linearization, as in [37], or re-linearization [38], during the solution process. These methods may be considered for future implementation.

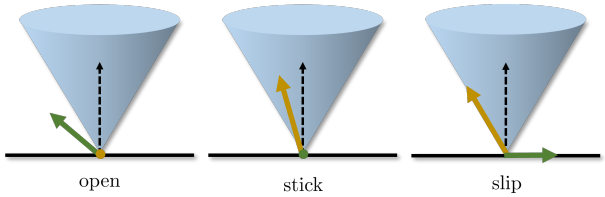

Fig. 2. Three cases resulting from the Signorini-Coulomb condition, ranging from open ($\lambda_{i,n} = 0$), stick ($\lambda_{i,n} > 0, \delta_i = 0$), to slip ($\lambda_{i,n} > 0, \delta_i > 0$), shown from left to right. The blue shape illustrates the friction cone, the green arrow indicates the contact frame velocity, and the yellow arrow represents the contact impulse.

1) Hard Constraint: Hard constraints ensure that equations and inequalities for the system are strictly satisfied (e.g., joint limit), including holonomic and non-holonomic types. If the i-th constraint is hard, the corresponding relation is

$$0 \le \lambda_i \perp J_i\hat{v} + e_i \ge 0 \tag{3}$$

where $e_i \in \mathbb{R}$ and $J_i \in \mathbb{R}^{1\times n}$ denote the error and Jacobian for hard constraint. Here, the error e_i is typically scaled and biased value, from the methods such as Baumgarte stabilization [39], to prevent the constraint drift effectively.

2) Soft Constraint: Soft constraints are typically originated from the elastic potential energy of the system (e.g., spring). If the i-th constraint is soft, constraint impulse can be written as

$$\lambda_i = -k_i e_i - b_i J_i\hat{v} \tag{4}$$

where $e_i \in \mathbb{R}$ and $J_i \in \mathbb{R}^{1\times n}$ are the error and Jacobian for soft constraint, $k_i, b_i > 0$ are the gain and damping parameter which are scaled and biased dependent on the time integration scheme, step size, and constraint-space damping. The values of k_i and b_i are associated with the system energy behavior, see [35], [40] for more details.

3) Contact Constraint: Contact condition is typically the most demanding type since it includes nonlinear complementarity relation between primal (i.e., velocity) and dual (i.e., impulse) variables. In this article, we assume that at each time step, set of contact features (i.e., gap, contact point, normal) are provided from collision detection module [41], [42]. Then for each contact point, we define Signorini-Coulomb condition (SCC), which is the most universal expression for dry frictional contact. If the i-th constraint is contact, the corresponding 3-DOF relation is

$$\begin{aligned} &0 \le \lambda_{i,n} \perp J_{i,n}\hat{v} + e_{i,n} \ge 0 \\ &0 \le \delta_i \perp \mu_i\lambda_{i,n} - \|\lambda_{i,t}\| \ge 0 \\ &\delta_i\lambda_{i,t} + \mu_i\lambda_{i,n}J_{i,t}\hat{v} = 0 \end{aligned} \tag{5}$$

where $\perp$ denotes complementarity, $e_{i,n} \in \mathbb{R}$ and $J_{i,n} \in \mathbb{R}^{1\times n}$ denote the error and Jacobian for contact normal, $J_{i,t} \in \mathbb{R}^{2\times n}$ is the Jacobian for contact tangential, and μ_i is the friction coefficient and δ_i is the auxiliary variable. The first condition, known as the velocity-level Signorini condition, captures the complementarity nature of the contact occurence and gap. The remaining conditions involve the complementarity between slipping velocity and the friction cone boundary, with the maximal dissipation law indicating that slip opposes the direction of impulse. There are three situations induced by the condition (5) - open, stick, and slip, as depicted in Fig. 2.

C. Augmented Lagrangian Method

By standard, augmented Lagrangian (AL) method is a class of algorithms to solve the following constrained optimization problem:

$$\min_{x,z} f(x) + g(z) \quad \text{s.t.} \quad Px + Qz = r.$$

Here, the augmented Lagrangian is defined as,

$$\mathcal{L} = f(x) + g(z) + u^T(Px + Qz - r) + \frac{\beta}{2}\|Px + Qz - r\|^2$$

where u is the Lagrange multiplier and $\beta > 0$ is the penalty weight. Then AL method takes the iteration step as

$$\begin{aligned}(x^{l+1}, z^{l+1}) &= \arg\min_{x,z} \mathcal{L}(x, z, u^l)\\ u^{l+1} &= u^l + \beta(Px^{l+1} + Qz^{l+1} - r)\end{aligned} \tag{6}$$

where l is the iteration index. In the above (6), (x, z) are coupled for minimization problem at each step. Meanwhile, Alternating direction method of multiplier (ADMM [43]) iteratively performs alternating minimization of $\mathcal{L}$ with respect to each variable. The iteration process of ADMM can be summarized as follow:

$$\begin{aligned}x^{l+1} &= \arg\min_{x} \mathcal{L}(x, z^l, u^l)\\ z^{l+1} &= \arg\min_{z} \mathcal{L}(x^{l+1}, z, u^l)\\ u^{l+1} &= u^l + \beta(Px^{l+1} + Qz^{l+1} - r)\end{aligned} \tag{7}$$

where l is the loop index. By independently resolving each variable, ADMM is often employed to enhance the efficiency and scalability of the application [43], [44]. In this work, we develop separate tailored algorithms based on the styles of (6) and (7).

IV. Multi-Contact Simulation via Augmented Lagrangian

A. Problem Formulation

The motivation for leveraging AL in contact simulation primarily stems from the insight to integrate tools from constrained optimization into the solving of constrained dynamics equation. The problem considered in this article, can be essentially formulated as

$$\begin{aligned}&\text{Solve } A\hat{v} = b + J^T\lambda\\ &\text{s.t. } (J\hat{v}, \lambda) \in \mathcal{S}_c\end{aligned} \tag{8}$$

where $A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ are the dynamics matrix/vector compressed from (2), and $\mathcal{S}_c$ represent the set that satisfies the relation between $J\hat{v}$ and λ described in (3), (4), and (5). In robotic manipulation scenarios, contact points are often generated numerously and densely, which can lead to the resulting problem being ill-conditioned or infeasibly defined. In such cases, performing per-contact iteration based on the dual conversion $JA^{-1}J^T$ (i.e., so-called Delassus operator) often proves inefficient and exhibits slow convergence. Meanwhile, AL in optimization is known for maintaining subproblem feasibility and demonstrating robust convergence, even converging to solutions with the least constraint violation in poorly defined problems [45].

Consequently, our primary objective is to investigate whether the advantages of the AL approach can be effectively applied to contact simulation. Although the problem (8) shares commonalities with optimization, it diverges due to the introduction of complementarity relations between primal and dual variables, particularly associated with contact conditions. Our aim is to establish a foundation for deriving AL techniques specifically tailored to multi-contact, thereby addressing the unique challenges posed in robotic simulation.

B. AL for Multi-Contact NCP

In this subsection, we will derive augmented Lagrangian (AL) based methods to address multi-contact NCP in simulation. We start by equivalently expressing (8) as follows:

$$\begin{aligned}&\text{Solve } A\hat{v} = b + J^T\lambda\\ &\text{s.t. } (z, \lambda) \in \mathcal{S}_c,\ J\hat{v} = z\end{aligned} \tag{9}$$

where $z \in \mathbb{R}^{n_c}$ serves as the slack variable for the constraint interface. The expression in (9) bears resemblance to the optimality condition of the following optimization problem:

$$\min_{\hat{v},z} \ \frac{1}{2}\hat{v}^T A\hat{v} - b^T\hat{v} + g(z) \ \text{ subject to } \ J\hat{v} = z \tag{10}$$

as the matrix A is always symmetric positive definite. In this context, g serves to enforce the constraint in dynamics, although $(z, \lambda) \in \mathcal{S}_c$ is not integrable into the function if the multi-contact condition included. Recalling the structure of (6) applicable to the optimization problem (10), we can similarly solve (9) as follows:

$$\begin{aligned}&\text{Solve } \begin{bmatrix} A + \beta J^T J & -\beta J^T \\ -\beta J & \beta I \end{bmatrix} \begin{bmatrix} \hat{v} \\ z \end{bmatrix} = \begin{bmatrix} b - J^T u \\ u + \lambda \end{bmatrix} \qquad (11)\\ &\quad \text{s.t. } (z, \lambda) \in \mathcal{S}_c\\ &u \leftarrow u + \beta(J\hat{v} - z). \qquad (12)\end{aligned}$$

The rationale of the above structure is that, at the fixed-point of the iteration (therefore, $J\hat{v} = z$), the result satisfies both dynamics equation and constraint relation. Similar to (6), the process can be interpreted as iterating between solving the problem relaxed via a penalty term and updating the Lagrange multipliers. We refer this relaxed problem (11) as the *surrogate* problem. However, unlike the minimization problem in (6), the solvability of the surrogate problem remains unclear, which may raise potential concerns. We address this issue in the following proposition.

Proposition 1: Surrogate problem (11) always has a feasible solution.

Proof: The proof can be done by borrowing the existence proof from [36] based on the Brouwer fixed-point theorem, which states that a solution to the Coulomb friction problem always exists if the rows of the contact Jacobian are linearly independent. Due to the slack variable z, the contact Jacobian in (11) can be simply considered as an identity matrix, satisfying this condition. ■

Given that the surrogate problem (11) has a feasible solution, the numerical scheme used to find this solution becomes significant as its performance directly affects the overall efficiency and accuracy of the AL methods for multi-contact NCP.

C. Closed-Form Formulation of Slack Variables

Compared to the original problem (8), the surrogate problem (11) should be easier to solve in order to maintain the rationality of the framework. A crucial difference between (8) and (11) is that the constraint condition is defined on the slack variable z as shown below:

$$\beta z = \beta J\hat{v} + u + \lambda, \quad \text{s.t.} \quad (z, \lambda) \in \mathcal{S}_c. \tag{13}$$

This implies that the relationship between z and λ is matrix-free and involves only a simple scalar weight β. Based on this feature, we can derive the *closed-form* representation for λ (therefore, also for z) with respect to $\hat{v}$ for each hard, soft, and contact constraint. The derivations are listed as below.

If the i-th constraint is hard, we can determine λ_i by substituting (13) into the complementarity relation (3):

$$\lambda_i = \Pi_{\geq 0}\left(-\beta J_i\hat{v} - u_i - \beta e_i\right) \tag{14}$$

where $\Pi_{\geq 0}$ denotes the projection on positive set. Meanwhile if the i-th constraint is soft, we can determine λ_i by substituting (13) into the linear relation (4):

$$\lambda_i = -\frac{b_i(\beta J_i\hat{v} + u_i) + \beta k_i e_i}{b_i + \beta} \tag{15}$$

Finally, if the i-th constraint represents a contact, we can determine λ_i by substituting (13) into the contact condition described in (5):

$$\lambda_i = \Pi_{\mathcal{C}}^{\text{strict}}\left(-\beta J_i\hat{v} - u_i - \beta e_i\right) \tag{16}$$

where $\Pi_{\mathcal{C}}$ denotes the projection onto the friction cone $\mathcal{C}$. Specifically, the projection $\lambda_i = \Pi_{\mathcal{C}}^{\text{strict}}(\lambda_i^*)$ is carried out by the following steps:

$$\begin{aligned} \lambda_{i,n} &= \max(\lambda_{i,n}^*, 0) \\ \lambda_{i,t} &= \Pi_{\mathcal{C}(\lambda_{i,n})}(\lambda_{i,t}^*). \end{aligned} \tag{17}$$

Here, $\mathcal{C}(\lambda_{i,n})$ represents the cross-section of $\mathcal{C}$ where the plane at height $\lambda_{i,n}$ intersects the cone. This nested projection is distinct from the closest distance projection, commonly known as the proximal operator when applied to the indicator function of the friction cone [27]. See Fig. 3 for illustrations of each projection scheme. As in [46], we refer to (17) as the strict operator as the resulting (z_i, λ_i) strictly satisfies the Signorini-Coulomb condition. This property is demonstrated in the proposition below.

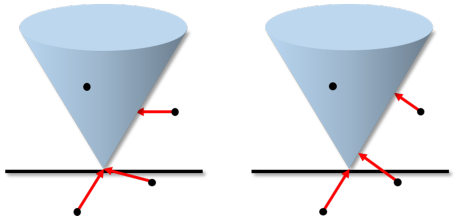

Fig. 3. Comparison of the strict operator (left) and the proximal operator (right) for the friction cone projection. Black dot: operator input; red arrow: projection direction.

Proposition 2: If the i-th constraint represents a contact, (16) with (17) gives the unique solution of (13).

Proof: As the normal component is completely decoupled from the tangential component in (13), it can be written as

$$\begin{aligned} \beta(z_{i,n} + e_{i,n}) &= \lambda_{i,n} + \beta J_{i,n}\hat{v} + u_{i,n} + \beta e_{i,n} \\ &= \lambda_{i,n} - \lambda_{i,n}^* \end{aligned}$$

Then, if $\lambda_{i,n}^* > 0$, $\lambda_{i,n} = \lambda_{i,n}^*$ is the only solution for which $z_{i,n} + e_{i,n} = 0$ is satisfied. Otherwise, if $\lambda_{i,n} = 0$, it is the only solution since $z_{i,n} + e_{i,n} > 0$. Therefore, the normal components are uniquely determined, satisfying the complementarity condition. For the tangential components, the relation can be written as

$$\beta z_{i,t} = \lambda_{i,t} - \lambda_{i,t}^*$$

Substituting the above equation into (5), we obtain

$$(\beta\delta_i + \mu_i\lambda_{i,n})\lambda_{i,t} = \mu_i\lambda_{i,n}\lambda_{i,t}^*$$

If $\lambda_{i,t}$ lies inside $\mathcal{C}(\lambda_{i,n})$, $\mu\lambda_{i,n} - \|\lambda_{i,t}\| > 0$ holds and δ_i should be 0. If $\lambda_{i,t}$ lies outside $\mathcal{C}(\lambda_{i,n})$, δ_i should be larger than 0, yet should satisfy $\mu\lambda_{i,n} = \|\lambda_{i,t}\|$, therefore it is uniquely determined. Finally, the resulting $\lambda_{t,i}$ is equivalent to the result of the strict operator, thus the statement holds. ■

The results (14), (15), and (16) derived above imply that λ_i can be expressed as a closed-form regardless of constraint type, allowing us to write it as:

$$\lambda_i = T(\lambda_i^*) \quad \text{where} \quad \lambda_i^* = -\beta J_i\hat{v} - u_i - \beta e_i \tag{18}$$

where T is a closed-form operator which is continuous yet may nonsmooth depending on the constraint type. Accordingly, by the linear relation (13), the slack variable z is also expressed in closed-form with respect to $\hat{v}$.

Based on the closed-form operation (18), solving (11) can be now expressed as solving following nonlinear equation:

$$\begin{aligned} r(\hat{v}) &= A\hat{v} - b - \sum_i J_i^T\lambda_i \\ &= A\hat{v} - b - \sum_i J_i^T T(-\beta J_i\hat{v} - u_i - \beta e_i) \end{aligned} \tag{19}$$

then computing $z = J\hat{v} + \frac{1}{\beta}(u + \lambda)$ accordingly. Due to the projection operator (17), $r : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous, yet semismooth equation. Therefore, one can handle the surrogate problem by solving this nonlinear equation (19) using the

Newton method, whose theories developed under semismooth case [47] by employing the generalized derivatives.

However, typical (semismooth) Newton methods are known to exhibit superlinear convergence near the solution but lack robustness. Although the Prop. 1 ensures the existence of solution and we attempt with various globalization techniques based on backtracking/edge-aware line-search and trust-region methods, we found that none provided sufficient robustness. This is critical considering that in physics simulation, as numerous iterations are required at each time step, and even a single failure can lead to significant consequences. Furthermore, the derivative of the closed-form operator (18) might become non-symmetric in contact cases, and cannot guarantee that $\frac{dr}{d\hat{v}}$ will always be non-singular. This issue makes the computation both expensive and unreliable. Consequently, we have developed two variations of the augmented Lagrangian tailored for the multi-contact NCP form (9): the cascaded Newton-based augmented Lagrangian method (CANAL) and the subsystem-based ADMM (SubADMM), which are presented in the following sections.

V. Cascaded Newton-Based Augmented Lagrangian

A. Cascaded Structure

A crucial issue of the Newton-based solution of (19) is that the landscape of the merit function $\frac{1}{2}\|r(\hat{v})\|^2$ is non-convex. Our core strategy to address this issue employs a cascaded method that relaxes each surrogate problem into a convex form, facilitating fast and stable solutions, while updating terms at each AL step to compensate for discrepancies between the convex problem and the original NCP. For the convex relation, we utilize the equivalence of $(z, \lambda) \in \mathcal{S}_c$ and (5) with the following condition:

$$\mathcal{C} \ni \lambda_i \perp z_i + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \mu_i \|z_{i,t}\| \end{bmatrix}}_{p_i} \in \mathcal{C}^* \tag{20}$$

where $\mathcal{C}^*$ denotes the dual cone of $\mathcal{C}$. This equivalence can be easily verified, as we refer [36] for details. The reformulated relation in (20) essentially constitutes a cone complementarity condition, if the perturbation term p_i is excluded.

A key idea of our cascaded Newton approach is to substitute the perturbation term p_i by borrowing z_i from the previous AL iteration. In other words, we treat p_i as a constant in every surrogate problem, and temporarily consider the relationship between z_i and λ_i as a cone complementarity condition. Consequently, in the $(l+1)$-th AL iteration, we solve the following nonlinear equation that replaces the strict operator (16) with the proximal operator:

$$\begin{aligned} r(\hat{v}^{l+1}) &= A\hat{v}^{l+1} - b - \sum_i J_i^T \lambda_i^{l+1} \\ \lambda_i^{l+1} &= \Pi_{\mathcal{C}}^{\text{prox}}(\underbrace{-\beta J_i \hat{v}^{l+1} - u_i^l - \beta \tilde{e}_i^l}_{\tilde{\lambda}_i^*}) \end{aligned} \tag{21}$$

where $\tilde{e}_i^l = e_i + p_i^l = e_i + \begin{bmatrix} 0 & 0 & \mu_i \|z_{i,t}^l\| \end{bmatrix}^T$. Even after this replacement, the nonlinear equation in (21) remains semismooth. However, we can demonstrate that it is integrable, as detailed in the following proposition. Note that to streamline the explanation, we will focus exclusively on the contact constraints below, as the other types (i.e., hard and soft) follow straightforwardly.

Proposition 3: The function $r(\hat{v})$ from (21) is the derivative of the following strongly-convex function:

$$h(\hat{v}) = \frac{1}{2}\hat{v}^T A\hat{v} - b^T\hat{v} + \sum_i \frac{1}{2\beta}\|\lambda_i\|^2 \tag{22}$$

Proof: The derivative of $h(\hat{v})$ can be expressed as:

$$\begin{aligned} \frac{dh(\hat{v})}{d\hat{v}} &= A\hat{v} - b - \sum_i J_i^T \frac{d\lambda_i}{d\lambda_i^*}^T \lambda_i \\ &= A\hat{v} - b - \sum_i J_i^T \lambda_i \end{aligned}$$

The latter equality holds due to the identity $\lambda_i^T(\lambda_i - \lambda_i^*) = 0$ in the proximal operator. The symmetric positive-definite property of A ensures that the quadratic term is strongly convex. Furthermore, since the squared distance to a convex set is convex, $\|\lambda_i\|^2$ is convex with respect to λ_i^*, and thus also for $\hat{v}$. Therefore, $h(\hat{v})$ is a strongly-convex function. ■

This result is closely related to those presented in [14], [24], although the objective function is defined differently based on our AL-based formulation. Given this property, we can apply the exact Newton method to the strongly-convex function (22) by computing the derivative of $r(\hat{v})$ (i.e., the Hessian), which is proven to exhibit global convergence [48].

B. Newton Step

Computing the derivative of $r(\hat{v})$ in (21) with respect to $\hat{v}$ is straightforward, except for the part involving T. As the operator T is a proximal operator on a friction cone (see Fig. 3), it involves a continuous concatenation of three formulaic forms, yet the function is semismooth at the connection points. Below, we provide derivative of each form which can be obtained from a few algebraic calculation:

$$\frac{d\lambda_i}{d\tilde{\lambda}_i^*} = \begin{cases} 0_{3\times 3}, & \text{open} \\ I_{3\times 3}, & \text{stick} \\ \frac{1}{\mu^2+1}\begin{bmatrix} \mu_i^2 I_{2\times 2} + \frac{\mu_i \tilde{\lambda}_{i,n}^*}{\|\tilde{\lambda}_{i,t}^*\|} P(\bar{\lambda}_{i,t}) & \mu_i \bar{\lambda}_{i,t}^T \\ \mu_i \bar{\lambda}_{i,t} & 1 \end{bmatrix}, & \text{slip} \end{cases} \tag{23}$$

where $\bar{\lambda}_{i,t}^*$ is the normalized vector of $\tilde{\lambda}_{i,t}^*$ and $P(\bar{\lambda}_{i,t}) = I - \bar{\lambda}_{i,t}\bar{\lambda}_{i,t}^T$ is the tangential projection matrix. Then the derivative can be written as

$$\frac{dr(\hat{v})}{d\hat{v}} = A + \sum_i \beta J_i^T \frac{d\lambda_i}{d\tilde{\lambda}_i^*} J_i. \tag{24}$$

Due to the structure given in (23), and consequently the matrix (24), is guaranteed to be symmetric positive definite, therefore

always invertible. Followingly, the direction of the Newton step is computed as

$$d(\hat{v}) = -\left(\frac{dr(\hat{v})}{d\hat{v}}\right)^{-1} r(\hat{v}) \tag{25}$$

where the $d(\hat{v})$ denotes the direction of $\hat{v}$ update.

Computation of the step (25) requires the linear solving of (24), therefore assemble and factorization of the matrix is necessary. For better efficiency, we can exploit sparsity pattern of the inertia matrix and the constraint Jacobian during the process.

C. Exact Line-Search

Drawing from well-known convex optimization theory [48], we can guarantee that (25) provides a descent direction. However, we still need to integrate a suitable line-search scheme to ensure global convergence. Here, the line-search problem can be described as following one-dimensional, strictly convex optimization problem:

$$\min_{\alpha>0} f(\hat{v} + \alpha d(\hat{v})). \tag{26}$$

Similar to [24], we can find a globally optimal solution of the problem (26) using the `rtsafe` algorithm, which effectively combines the one-dimensional Newton-Raphson method and a bisection scheme. In practice, we find that the Newton step, when combined with the aforementioned exact line-search, performs robustly even with large values of β. This approach significantly enhances the robustness of the simulator, as the standard semismooth Newton method on (19) often leads to failures, especially for large β. The effectiveness is attributable to the integration of our cascaded scheme and the well-established theories in convex optimization literature.

D. Warm-Start and Penalty Parameter Update

At each Newton loop, we can warm-start $\hat{v}$ from the value of the previous CANAL loop. This effectively reduces the number of necessary Newton steps in practice, as the optimal solution of the inner convex optimization should be similar as the AL iteration converges. Typically, with warm starting, we find that one or two Newton iterations often suffice after some progress has been made in the CANAL iteration. Therefore, the computational cost per iteration step tends to decrease. For the first iteration, we can warm-start $\hat{v}$ and u (and therefore also z) by using the values from the previous time step.

The penalty parameter β plays a crucial role in the CANAL algorithm. Typically, a high value of β improves the convergence of the residual $\|J\hat{v} - z\|$ to zero. However, it also makes the convex problem numerically stiff, thus requiring additional Newton iterations to solve. Consequently, we begin with a moderate value of β (10^4 in our cases) and increase it if the residual value is not sufficiently reduced. The update rule for increasing β is defined as follows:

$$\beta \leftarrow \min(\kappa\beta, \beta^{\max}), \tag{27}$$

where $\kappa > 1$ is the hyperparameter. This rule includes restriction of β from becoming unnecessarily large, thus bounding the stiffness of (27) to circumvent numerical instability. Note that, if we perform only a single iteration on CANAL, it is equivalent to solving a soft convex formulation of contact as in [2], [25]. In this regard, CANAL can be considered as their extension, refining approximations and converging to near-rigid behavior through the update of primal-dual variables. In practice, we find that it takes only a few iterations to achieve plausible behavior in simulations, and after several iterations, it tends to converge to very high accuracy. The overall CANAL algorithm is summarized in Alg. 1.

Algorithm 1: Multi-Contact Simulation via CANAL

while *simulation* **do**
  initialize $l = 0, \hat{v}^0, z^0, \beta > 0, \kappa > 1, 0 < \eta < 1$
  while *CANAL loop* **do**
    initialize $\hat{v}^{l+1} \leftarrow \hat{v}^l$
    compute $\tilde{e}$ based on z^l
    while *Newton loop* **do**
      compute $r(\hat{v}^{l+1})$ (21)
      if $\|r(\hat{v}^{l+1})\| < \theta_{th}^N$ **then**
        break
      end
      compute Newton step $d(\hat{v}^{l+1})$ (25)
      compute α via exact line-search (26)
      $\hat{v}^{l+1} \leftarrow \hat{v}^{l+1} + \alpha d(\hat{v}^{l+1})$
    end
    update z^{l+1} and multiplier u^{l+1} (12)
    if $\|J\hat{v}^{l+1} - z^{l+1}\| < \theta_{th}^{AL}$ **then**
      break
    else
      if $\|J\hat{v}^{l+1} - z^{l+1}\| > \zeta\|J\hat{v}^l - z^l\|$ **then**
        update β (27)
      end
    end
    $l \leftarrow l + 1$
  end
  update system state using $\hat{v}^{l+1}$
end

VI. Subsystem-Based Alternating Direction Method of Multiplier

While the CANAL-based multi-contact simulation described in Sec. V exhibits fast convergence and stable constraint handling in practice, its scalability may be limited by the need to compute at least one Newton step for each AL iteration. Although we fully exploit the sparsity pattern, the Hessian matrix may become fully dense in the worst-case scenarios (e.g., long kinematic chains, dense coupling), thereby significantly increasing the complexity of the factorization process. Consequently, this approach can become computationally expensive when dealing with large-DOF multibody systems that include numerous objects.

One reasonable option in this regard is to adopt the methodology of ADMM, which, instead of solving the coupled problem of $(\hat{v}, z)$, performs alternating computation for each $\hat{v}$ and z. By employing this alternating approach, the problem

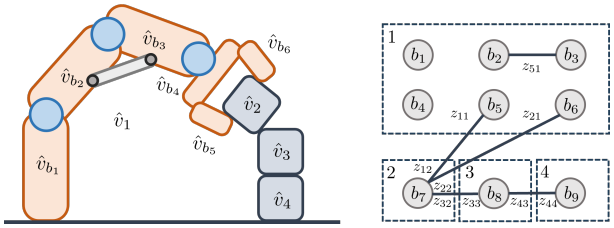

Fig. 4. Illustrative example demonstrating division and slack variable definition in SubADMM. Left: A multibody system with contacts comprising 4 subsystems, including 1 articulated body (robot) and 3 rigid bodies. Right: Corresponding graphical representation.

can be decomposed into a closed-form operator for z from (18) and a linear problem for $\hat{v}$ from (11):

$$(A + \beta J^T J)\hat{v} = b + J^T(\beta z - u) \tag{28}$$

Although this vanilla ADMM allows matrix factorization to be performed only once for each time step, its computational efficiency diminishes with increasing system size, and also the sparsity pattern of the matrix in (28) is same with the Hessian matrices used in CANAL. Additionally, in practice, ADMM often necessitates tuning of the parameter β during iterations to achieve optimal performance, which may require re-factorization of the matrix. Hence, we introduce a novel algorithm termed subsystem-based ADMM (SubADMM), designed to offer enhanced scalability with parallelization capabilities.

A. Subsystem-Based Reformulation

In our robotic simulation, we assume that the multibody system is composed of a kinematic chain [49], where each body is connected to its parent body via various types of joints (fixed, floating, revolute, prismatic, etc.). We then define the notion of a *subsystem* as a single subtree rooted at the ground. For example, a single floating rigid body or a single robot (each of robotic arm, humanoid, etc.) is regarded as a subsystem.

To better leverage the subsystem structure, we adopt a variations on the definition of augmented Lagrangian compared to the one described in Sec. IV. We first rewrite the multi-contact simulation problem (8) as follows:

$$\begin{aligned} &A_j\hat{v}_j = b_j + \sum_i J_{ij}^T\lambda_i \quad \forall j \in \{1,\cdots,N\} \\ &\text{s.t.} \quad (\hat{V}_i, \lambda_i) \in \mathcal{S}_{c,i} \quad \forall i \in \{1,\cdots,M\} \end{aligned} \tag{29}$$

where M is the number of constraint, N is the number of subsystem, and $\hat{V}_i$ is the subset of $\{\hat{v}_1,\cdots,\hat{v}_N\}$ that contributes to the i-th constraint. Here, J_{ij} are defined only for (i,j) such that $\hat{v}_j \in \hat{V}_i$. Note that our reformulation does not rely on any assumptions about the system. In the unconstrained case, the dynamics of each subsystem are readily decoupled (thus $A_j\hat{v} = b_j \ \forall j$). The coupling between subsystems is modeled by constraint forces as described in (29), which include both intra- and inter-subsystem interactions.

In robotic systems, constraints are applied either intra- or inter-bodies (e.g., contacts, tendons) or to joints (e.g., limits, controls). Based on this insight, we also reformulate the structure of the Jacobian. To illustrate the idea, let us consider the example in Fig. 4. In this example, the Jacobian for the first constraint (contact between the 5-th and 7-th bodies) can be written as follows:

$$J_1 = \begin{bmatrix} J_{1,b_5}J_{b_5,1} & J_{1,b_7}J_{b_7,2} \end{bmatrix}$$

where $J_{b_\star,\star}$ maps the joint space to the body space, and $J_{\star,b_\star}$ maps the body space to the constraint space. In this case, Jacobian for each subsystem is naturally defined as follows:

$$J_{11} = J_{1,b_5}J_{b_5,1} \quad J_{12} = J_{1,b_7}J_{b_7,2} \tag{30}$$

Meanwhile, in the case of an inter-subsystem constraint, such as the 5-th constraint in Fig. 4, acting internally on the first subsystem, we express the Jacobian as:

$$J_{51} = \begin{bmatrix} J_{5,b_2}J_{b_2,1} \\ J_{5,b_3}J_{b_3,1} \end{bmatrix} \tag{31}$$

which splits the original Jacobian $J_{5,b_2}J_{b_2,1} + J_{5,b_3}J_{b_3,1}$ and stack it row-wise[1]. This reformulation is similarly applied for the constraints to joints. Below, we describe how this reformulation can lead to an efficient ADMM process.

B. ADMM for the Reformulation

Based on the reformulation described in Sec. VI-A, we define slack variable to equivalently express the problem similar to (9):

$$\begin{aligned} &A_j\hat{v}_j = b_j + \sum_i J_{ij}^T\lambda_i \quad \forall j \in \{1,\cdots,N\} \\ &\text{s.t.} \quad (Z_i, \lambda_i) \in \mathcal{S}_{c,i}, \quad z_{ij} = J_{ij}\hat{v}_j \quad \forall i \in \{1,\cdots,M\} \end{aligned} \tag{32}$$

where Z_i is the set of slack variables z_{ij} such that (i,j) satisfies $\hat{v}_j \in \hat{V}_i$. Fig. 4 gives an example of how z_{ij} are defined under the subsystem-based structure. Then from the augmented Lagrangian structure described in Sec. IV, surrogate problem for the reformulation (32) can be formulated as follows:

$$\begin{aligned} (A_j + \sum_i \beta J_{ij}^T J_{ij})\hat{v}_j - \beta\sum_i J_{ij}^T z_{ij} &= b_j - \sum_i J_{ij}^T u_{ij} \quad \forall j \\ -\beta J_{ij}\hat{v}_j + \beta z_{ij} &= u_{ij} + \lambda_i \quad \forall i,j \\ \text{s.t.} \quad (Z_i,\lambda_i) &\in \mathcal{S}_{c,i} \quad \forall i. \end{aligned} \tag{33}$$

The problem described in (33) still involves coupling between all $\hat{v}_j$ and z_{ij}. Therefore, solving it all at once would negate the benefits of using a subsystem-based representation.

However, by leveraging the ADMM structure (7), an alternate resolution for each variable is performed, allowing us to capitalize on the subsystem-based partitioning described above. By solving (33) with respect to $\hat{v}$ first, it reduces to following linear problem for all j:

$$(A_j + \sum_i \beta J_{ij}^T J_{ij})\hat{v}_j^{l+1} = b_j + \sum_i J_{ij}^T(\beta z_{ij}^l - u_{ij}^l) \tag{34}$$

where l denotes the iteration index. The process described in (34) essentially involves obtaining a linear solution of size

[1]For consistency, λ_i in (29) should technically be stacked row-wise for this case, but we maintain the current notation for simplicity.

$\dim(\hat{v}_j)$ for each subsystem, and always solvable from the positive definite property of the left-most matrix. A crucial difference between (28) is that the linear problem is much smaller, while each of them can be solved in parallel. Each $A_j + \sum_i \beta J_{ij}^T J_{ij}$ can be pre-factorized before iteration, as they remain invariant unless β is modified.

Then, solving (33) with respect to z_{ij} can be reduced to following problem for all i:

$$\beta z_{ij}^{l+1} = \underbrace{\beta J_{ij}\hat{v}_j^{l+1} + u_{ij}^l}_{y_{ij}^{l+1}} + \lambda_i^{l+1}, \quad \text{s.t.} \quad (Z_i, \lambda_i) \in \mathcal{S}_{c,i}. \tag{35}$$

The solution of (35) can be performed independently for each i, allowing for parallelization across all constraints. Additionally, as described in Sec. IV-C, the relation between z and λ is matrix-free, thus allowing for a closed-form solution process. However, the formulation needs to be slightly adjusted since multiple slack variables in Z_i are involved in each constraint. For the case of hard constraint, (14) can be adjusted for the subsystem-based reformulation as follows:

$$\lambda_i^{l+1} = \Pi_{\geq 0}\left(-\frac{\sum_j y_{ij}^{l+1} + \beta e_i}{|Z_i|}\right) \tag{36}$$

where $|Z_i|$ denotes the cardinality of Z_i. Meanwhile, for the soft constraints, (15) can be adjusted as follows:

$$\lambda_i^{l+1} = -\frac{b_i \sum_j y_{ij}^{l+1} + \beta k_i e_i}{b_i |Z_i| + \beta}. \tag{37}$$

Finally, for the contact constraints, we can use

$$\lambda_i^{l+1} = \Pi_{\mathcal{C}}^{\text{strict}}\left(-\frac{\sum_j y_{ij}^{l+1} + \beta e_i}{|Z_i|}\right) \tag{38}$$

instead of (16). The resulting equations (36), (37), and (38) still consist of simple algebraic operations, making them easy to compute. Note that we directly employ the strict operator in (38), without adopting the cascaded approach with the proximal operator as used in CANAL. This is because ADMM strategically employs a single alternating solution without fully solving the surrogate problem, which provides conservative updates and maintains stability without requiring a specific convexification and globalization process.

After both alternating steps, the Lagrange multiplier is updated as follows:

$$\begin{aligned} u_{ij}^{l+1} &= u_{ij}^l + \beta(J_{ij}\hat{v}_j^{l+1} - z_{ij}^{l+1}) \\ &= -\lambda_i^{l+1}. \end{aligned} \tag{39}$$

Hence, there is no necessity to store u_{ij} separately; only λ_i needs to be retained. In summary, our SubADMM iterates between (34), (35), and (39), with each step being naturally parallelizable per subsystem or per constraint as illustrated in Fig. 5. This property ensures the scalability of the algorithm with respect to the number of subsystems and constraints.

C. Factorization of Subsystem Matrices

In the SubADMM process described above, the size of each linear problem in (34) is determined by $\dim(\hat{v}_j)$, which equals

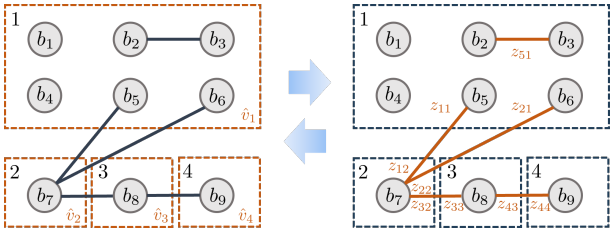

Fig. 5. Iteration structure of SubADMM. $\hat{v}$ is updated independently for each subsystem block, while z is independently updated for each constraint factor.

6 for a subsystem consisting of a single rigid body. However for subsystems with articulated body structures, it corresponds to the total DOF of joints. Generally, the factorization of a matrix has a complexity of $\mathcal{O}(n^3)$. Consequently, there may be concerns that our algorithm could become susceptible to an increase in the degrees of freedom of the subsystem, such as in a tree with extensive length, unless $A_j + \sum_i \beta J_{ij}^T J_{ij}$ possesses a special structure.

However, the reformulation technique (31) we established earlier for the Jacobian matrix, becomes crucial in addressing this concern. To illustrate, let us derive the equation as follows:

$$\begin{aligned} A_j + \sum_i \beta J_{ij}^T J_{ij} &= A_j + \sum_i \sum_k \beta J_{b_k,j}^T J_{i,b_k}^T J_{i,b_k} J_{b_k,j} \\ &= J_{\mathcal{B}_j,j}^T \underbrace{\begin{bmatrix} H_{b_{j,1}} & & \\ & \ddots & \\ & & H_{b_{j,|\mathcal{B}_j|}} \end{bmatrix}}_{H_{\mathcal{B}_j}} J_{\mathcal{B}_j,j} \end{aligned} \tag{40}$$

where $\mathcal{B}_j$ represents the set containing every body index in j-th subsystem, $J_{\mathcal{B}_j,j}$ denotes the Jacobian mapping from subsystem joints to all child body spaces, and $H_{b_k} \in \mathbb{R}^{6\times 6}$ can be interpreted as the virtual effective matrix defined for each body, expressed as

$$H_{b_k} = M_{b_k} + \sum_i \beta J_{i,b_k}^T J_{i,b_k} \tag{41}$$

where M_{b_k} is the inertia matrix for the body originated from A_j. One significant advantage of the structure (40) is that $H_{\mathcal{B}_j}$ forms a block-diagonal matrix, ensuring that the entire matrix always maintains the same sparsity structure as the inertia matrix of the articulated body. Hence, we can implement an efficient construction and factorization algorithm based on the kinematic-tree structure, which is well-known as Featherstone's algorithm [49]. Specifically, we use composite rigid body algorithm [49] that capitalizes on branch-induced sparsity to streamline computations, recursively navigating to the parent node to perform efficient fill-ins. In Appendix A, we details how the algorithm on (41) can be efficiently performed. It is worth noting that, without the Jacobian reformulation described in (31), $H_{\mathcal{B}_j}$ does not exhibit block diagonal characteristics. As a result, the sparsity pattern becomes more complex and cannot be determined solely by the kinematic structure, but rather changes with each time step.

Algorithm 2: Simulation using SubADMM

subsystem-based reformulation (Sec. VI-A)
while *simulation* **do**
  initialize β
  $\forall j$ construct A_j, b_j in parallel
  $\forall i$ construct e_i, J_i in parallel
  $\forall j$ factorize $A_j + \sum_i \beta J_{ij}^T J_{ij}$ in parallel (Sec. VI-C)
  initialize $l = 0, z^0, u^0$
  while *SubADMM loop* **do**
    $\forall j$ update $\hat{v}_j^{l+1}$ in parallel (34)
    $\forall j$ update Z_i^{l+1} in parallel (35)
    $\forall i$ store u_i^{l+1} (39)
    compute residual θ_p, θ_d (42)
    if $\theta_p + \theta_d < \theta_{th}$ *or* $l = l_{max}$ **then**
      break
    else
      update β (43) (optional)
      $\forall j$ refactorize $A_j + \sum_i \beta J_{ij}^T J_{ij}$ in parallel
    end
    $l \leftarrow l + 1$
  end
  update each subsystem state using $\hat{v}_j^{l+1}$
end

Remark 1: Based on the aforementioned matrix structure, rather than constructing and factorizing (40), we can solve the linear equation (34) using the articulated body algorithm [49] at each SubADMM iteration step. While this approach strictly guarantees $\mathcal{O}(n)$ complexity for the body count, with no additional overhead for changes in β, we have observed that using factorization is often more efficient in practice, given that SubADMM typically requires a few tens of iteration. Nonetheless, this alternative remains as a viable option.

D. Convergence and Adaptive Penalty Parameter

For strictly convex optimization, ADMM is known to guarantee convergence at a linear rate [43]. Although our formulation shares similarities with the convex optimization (10), the multi-contact condition is generally not integrable, making theoretical convergence not well established in general. However, we empirically find that the convergence properties in our SubADMM-based simulations are comparable to those observed in convex optimization.

Typically, residuals in ADMM are defined in two kinds: primal and dual. In SubADMM, these definitions similarly hold as follows:

$$\begin{aligned} \theta_p &= \max_{i,j} \|J_{ij}\hat{v}_j - z_{ij}\| \\ \theta_d &= \max_{j} \|A_j\hat{v}_j - b_j - \sum_i J_{ij}^T \lambda_i\| \end{aligned} \tag{42}$$

where θ_p and θ_d represent the primal and dual residuals, respectively. Here, the primal residual can be interpreted as the satisfaction of constraints, while the dual residual reflects the satisfaction of the dynamics equations. Although we observe stable convergence of the residuals (42) in SubADMM, the method still inherits well-known drawbacks associated with ADMM-style iterations. Specifically, the performance of the algorithm is heavily dependent on the strategy used to select the penalty parameter β. A popular strategy is to adaptively tune β based on the residual. Generally, a large β reduces the primal residual, while a small β reduces the dual residual. Therefore, we can adopt the strategy of adjusting β based on the following rule:

$$\beta = \beta\sqrt{\frac{\theta_p}{\theta_d}} \quad \text{if} \quad \theta_p > \gamma\theta_d \text{ or } \theta_d > \gamma\theta_p \tag{43}$$

where $\gamma > 1$ is a hyperparameter. This adjustment should accompany the refactorization of $A_j + \sum_i \beta J_{ij}^T J_{ij}$. However, thanks to our subsystem-based division structure, this refactorization can also be performed in a parallelized and scalable manner, allowing us to reduce overhead and enable more frequent feedback adjustments.

Moreover, another crucial aspect we observe to address this issue is that the presence of large number of inactive constraints (the open case for contact) can impede convergence. Therefore, conducting thorough broad-phase collision tests to cull reasonable contact pairs is a significant step in enhancing convergence speed. For the initialization of β, we consider the structure of the terms in (40). A practical strategy involves balancing the weighting between the dynamics-related term A_j and the constraint-related term $\sum_i J_{ij}^T J_{ij}$, as suggested in general theoretical analysis [50], [51]. From this perspective, β in (41) can be interpreted as reflecting a pseudo-density (i.e., body mass divided by the number of contact points on it), given that the Jacobian J_{i,b_k} maps the motion of the body frame to the motion of a specific point on the body (see also derivation in Appendix A). Based on this insight, we use the geometric mean of the pseudo-densities of all bodies as the initial value for β.

E. Comparison with Previous Work

Portions of the algorithm outlined in this section were presented in our previous work [34], applying the idea of leveraging ADMM in physics simulations and the division of the entire multibody system into smaller parts. However, the scope of this article extends to more general theories and diverse variations of the augmented Lagrangian for robotic multi-contact simulations, as outlined in Sec. IV and V. For this section specifically, this article provides a complete subsystem division and parameter adaptation rule for subsystem-based ADMM. More importantly, we introduce novel reformulation techniques (31), enabling efficient factorization of submatrices described in Sec. VI-C. This improvement enhances scalability, not only for the number of subsystems but also for the DOF of each subsystem. Finally, a variety of new manipulation examples are implemented in simulations for the experiments.

VII. Examples and Evaluations

In this section, we present several implementation examples to illustrate the advantages of the proposed solver algorithms. The key question we address here is whether our AL-based

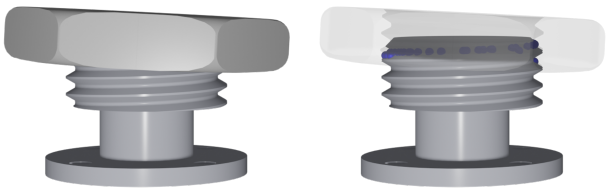

Fig. 6. Left: Visualization of an M48 bolt and nut used in our bolt-nut assembly simulation test. Right: Contact points visualized as blue spheres in the test configuration.

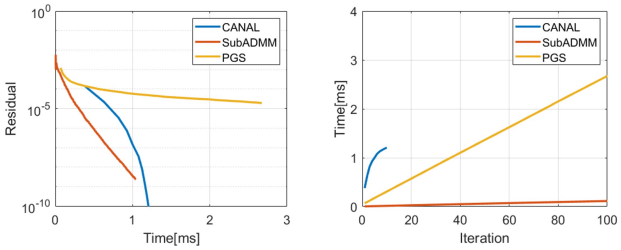

Fig. 7. Comparison of CANAL, SubADMM, and PGS for the bolt-nut assembly task simulation. Left: Residual decrease over computation time. Right: Computation time over iteration.

multi-contact solver (CANAL and SubADMM) can address the limitations of existing per-contact schemes in solving NCP posed in various robotic scenarios. As a universal baseline, we have implemented the Projected Gauss-Seidel (PGS) solver, which is widely utilized in numerous software applications (see Table I) and can handle NCP without the need for model relaxation. Note that we do not consider methods that depend on specific model relaxations, such as direct pivoting schemes based on LCP formulation.

Quantifying the accuracy of different multi-contact solvers is non-trivial. For each time step, it is essential to assess how well the solution $(\hat{v}, \lambda)$ satisfies two conditions: dynamics and contact constraints. For consistent comparison, we first project $\hat{v}$ using the relationship $\hat{v} = A^{-1}(b + J^T\lambda)$ based on the λ result of solver, making the dynamics residual become zero. Subsequently, we compute z using the following equation:

$$z_i = J_i\hat{v} - \lambda_i + T(-J_i\hat{v} + \lambda_i - e_i)$$

which is derived from the process described in Sec. IV-C, and we measure the contact residual as $\|J\hat{v} - z\|$ divided by the number of contacts. This residual becomes zero only when the contact conditions are exactly satisfied. Using this strategy, we can fairly compare the accuracy of different solvers: dual-based (PGS) and primal-dual based (CANAL and SubADMM). For all examples, we utilize a time step of $1/240$ s.

For the code implementation, we utilize the C++ language, employing the Eigen matrix library [52] for linear algebra operations and the OpenGL library for rendering. Computation time is measured on an Intel Core i5-13600KF CPU at 3.50 GHz.

A. Bolt-Nut Assembly

The first example we consider is the simulation of a bolt-nut assembly. This scenario is characterized by the intensive formation of contacts and stiff interactions due to the complexity of the geometry. As a result, the DOF for the constraints (typically hundreds) far exceed those of the system itself. In this context, two significant issues arise with the per-contact solver: 1) the Delassus operator becomes large and dense (i.e., many contacts are coupled), which slows down the iteration scheme, and 2) the intensive contact results in a limited feasible region or even leads to infeasibility, thereby slowing convergence.

The scenario is configured via a pair of M48 bolt and nut, visualized in Fig. 6. Here, collision detection process between the bolt and nut might be challenging. Therefore, we adopt a neural network-based signed distance function model presented in [53]. Based on this model, we can precisely represent the surface of the bolt, while the nut is represented through multiple triangulated face to perform collision detection using Frank-Wolfe algorithm [54]. Additionally, if the number of detected contact points exceeds 120, which is empirically found to be impractical, we perform contact clustering [53], [55] to effectively reduce them.

1) Single Step Test: To precisely evaluate the quantitative performance, we first measure the results of running different solvers single step at the same state and inputs. For test case generation, we sample 10 configurations of a nut that form numerous contacts with the bolt (see Fig. 6 for an example). We then apply 10 random external wrenches to the nut, generating a total of 100 test cases. The performance of the solvers in such scenarios is depicted Fig. 7. As demonstrated, CANAL and SubADMM solve the problems with higher accuracy compared to the PGS algorithm. Both algorithms can achieve residuals of 10^{-8} or less, whereas PGS struggles to converge quickly past 10^{-4}. In particular, CANAL exhibits superlinear convergence, achieving complete convergence in about 10 iterations. While SubADMM quickly converges to a certain accuracy, due to its first-order nature, it shows lower accuracy than CANAL after a certain period. For the computation time per iteration, SubADMM requires significantly less compared to the other two algorithms as it requires negligible preparation phase cost and each iteration can be performed very quickly. CANAL takes more time per iteration compared to PGS, however, its rapid convergence leads to fewer iterations overall, resulting in shorter total computation time. While SubADMM and PGS incur uniform costs for each iteration, leading to a linear increase in computation time, computation time for CANAL grows sublinearly as iterations progress. This is due to the effect of warm-starting explained in Sec. V-D, as the outer AL iteration approaches convergence, the number of required Newton iterations to solve the inner convex problem decreases. Overall, the results suggest that if we aim to achieve reasonable accuracy in a very short amount of time, SubADMM is a good option. Conversely, if very high accuracy is desired and more computational budget is available, CANAL is the preferable option.

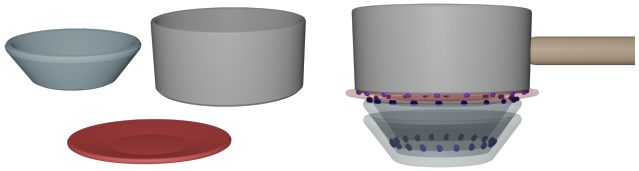

Fig. 8. Left: Visualization of a bowl, plate and pot used in our dish piling simulation test. Right: Contact points visualized as blue spheres in the test configuration.

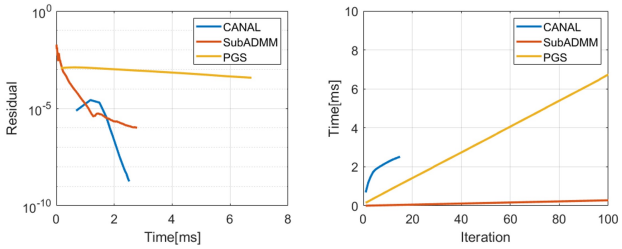

Fig. 9. Comparison of CANAL, SubADMM, and PGS for the dish piling simulation. Left: Residual decrease over computation time. Right: Computation time over iteration.

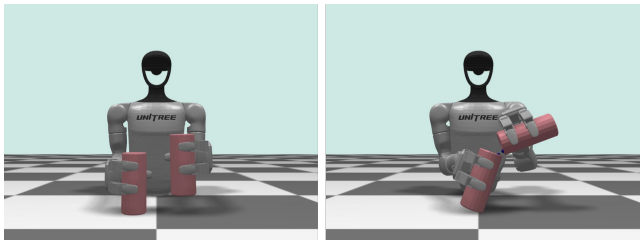

Fig. 10. Snapshots of a particle pouring task simulation. SubADMM excels in computation speed and scalability; CANAL in accuracy.

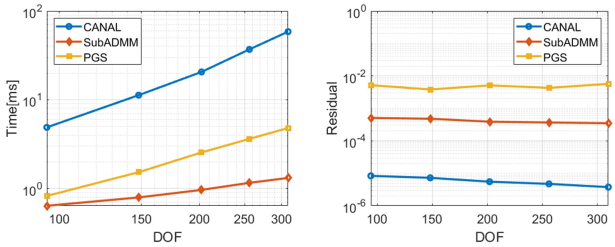

Fig. 11. Comparison of CANAL, SubADMM, and PGS for the pouring simulation. Left: Average computation time over system DOF. Right: Average residual over system DOF.

2) Assembly using Robot Manipulator: We also perform a full assembly task simulation using a robotic manipulator. The manipulator comprises a Franka Panda arm equipped with a Hebi X5 gripper, with the nut attached to the gripper. We control the robot using joint-level impedance control to follow the desired assembly trajectory, and the snapshots are depicted in Fig. 1. For performance validation, we limit the computation budget for the solver to 0.5 ms for each time step. As a result, simulations using CANAL and SubADMM successfully complete the assembly task. In contrast, PGS fails, as significant penetrations are generated due to its lack of convergence.

B. Dish Piling

The next scenario we implement involves piling dishes, a common situation in household environments. To compose the environment, we generate various types of dishes, including bowls, plates, and pots, as depicted in Fig. 8. As the shapes of the dishes are concave, this scenario is characterized by a large number of contacts distributed across the surfaces, leading to issues similar to those described in Sec. VII-A. Moreover, we model light bowls and plates (0.1 kg) beneath a heavy pot (5 kg), resulting in a challenging mass ratio for the stable simulation.

We develop a specially designed class of signed distance functions to represent the geometry of dishes, which is detailed in Appendix B. This approach enables us to compute the signed distance using simple algebraic operations, such as rounding and revolution. Subsequently, we generate the corresponding triangulated faces to perform collision detection between dishes, employing the same Frank-Wolfe algorithm used in the bolt-nut assembly scenario.

1) Single Step Test: For a single-step test, we first obtain the stacked pose of four dishes (bowl-bowl-plate-pot), as depicted in Fig. 8. We then apply random external wrenches to each dish, generating a total of 100 test cases. The performance of the solvers in these cases is depicted in Fig. 7. As shown, CANAL and SubADMM achieve significantly better accuracy in less time compared to PGS. CANAL achieves the highest accuracy, with residuals under 10^{-8}, and exhibits over linear convergence. SubADMM, demonstrating first-order convergence, struggles to achieve residuals under 10^{-6}. Due to the odd mass ratio present in the environment, the differences in achievable residuals between the solvers are larger compared to those in the bolt-nut assembly test. This suggests that CANAL may be the more preferable option in this case, although SubADMM remains a viable choice for achieving moderate results in a very short time. The trend in computation time per iteration is similar to that observed in bolt-nut assembly scenarios; per-iteration cost ranks as follows: CANAL $>$ PGS $>$ SubADMM, and the cost for each iteration in CANAL tend to decreases as the iterations proceed.

2) Piling using Robot Manipulator: We also perform a piling task simulation using a robotic manipulator composed of a Franka Panda arm equipped with an Allegro hand, bringing the total system dimension to 47. We employ a joint-level impedance controller to enable the robot to follow the desired grasp-and-place trajectory, as depicted in the snapshots in Fig. 1. Similar to the bolt-nut assembly simulation test, we limit the computation budget to 1 ms to compare the performance of different solvers. In this test, SubADMM and CANAL can successfully simulate the piling, while PGS fails, generating jittery movements due to a lack of convergence.

C. Pouring

We then simulate the pouring of particles contained in a bottle, employing a 28 DOF dual-arm manipulator with the upper body of a Unitree G1 with gripper. Also the particles

are modeled as spheres with a radius of 5 mm and a density of 10 $\mathrm{g/cm^3}$, contained within a bottle whose geometry is defined in the same way to the dishes in the previous subsection. We program the robot to reach for, grasp the bottle, and then pour the particles into another bottle, as depicted in the simulation snapshots in Fig. 10.

This scenario is characterized by a large number of bodies; therefore, both the system DOF and the constraint DOF are large, yet their relationship is relatively sparse compared to previous examples. Moreover, the results are expected to maintain a stable grasp and precisely emulate the coupling between a high-gain controlled robot and lightweight particles. Our primary objective for this scenario is to evaluate the scalability of the solvers. Therefore, we utilize a fixed number of iterations for each solver—5 for CANAL and 100 for SubADMM and PGS, then observe the average residual and computation time over the task execution while varying the number of particles in the bottle (9, 18, 27, 36, 45).

The results are depicted in Fig. 11. As shown in the plots, the increase in computation time with respect to the particle number (therefore, system DOF) is ordered as SubADMM $<$ PGS $<$ CANAL. This result aligns with the theoretical properties, as the alternating steps of SubADMM scale at least linearly with the number of subsystems and constraints. Note that this scalability could potentially be reduced with the adoption of more advanced parallelization hardware architectures, although we remain this as a work for future implementation. In the case of PGS, while the computation time for each Gauss-Seidel iteration step increases near linearly, the computation required to establish a contact relation in the dual space (related to the Delassus operator) increases superlinearly, making the overall exponent over SubADMM. CANAL scales superlinearly mainly due to the factorization of the Hessian matrix (24). However, the exponent is still significantly lower than 3, which is typical for dense matrix factorization, because we leverage the sparse structure of the contact Jacobian.

In terms of accuracy, CANAL outperforms both SubADMM and PGS, aligning with theoretical expectations and previous results. Within 5 iterations, the solver achieves a residual under 10^{-5}. Additionally, SubADMM consistently demonstrates better accuracy than PGS, achieving approximately 0.1 times the residual of PGS. We find that using PGS, attempting a strong grasp on the bottle can lead to significant instability in the simulation. For all solvers, the residual is not significantly affected by the number of particles.

D. Ablation Studies

Lastly, we conduct ablation studies to validate the effectiveness of the technical components presented in this article.

1) CANAL: First, we compare a cascaded Newton structure in CANAL with performing augmented Lagrangian by directly solving (19) using standard semismooth Newton methods. For baselines, we implement two algorithms: trust-region dogleg algorithm [48], which is a widely standard method for nonlinear equation solver, and damped semismooth Newton algorithms based on backtracking line search on merit function [56]. For both methods, Jacobian of $r(\hat{v})$ is derived similarly with (23) and (24). However, since the operator T is a strict operator here, the Jacobian is not guaranteed to be symmetric positive definite. Therefore, solving the linear problem may require additional computational effort in practice, compared to the Newton step computation in CANAL (25).

TABLE II
COMPARISON RESULTS OF LEVERAGING A CASCADED NEWTON STRUCTURE VERSUS DIRECTLY APPLYING STANDARD SEMISMOOTH NEWTON SOLVERS IN AN AL-BASED MULTI-CONTACT SOLVER.

	CANAL	TRDogleg	DampedSN
Inner iter.	30.55	118.8	246.9
Failure[%]	0	1.2	1.1
Residual[-log]	8.2231	8.5693	8.5644

TABLE III
COMPARISON RESULTS OF ADMM WITH SUBSYSTEM-BASED SPLITTING VERSUS MORE STANDARD SPLITTING STRATEGIES.

	SubADMM	No Sub	No Body
Time[ms]	0.435	1.21	0.638
Residual[-log]	4.8802	4.8817	4.8562

For the test, we utilize the same environment and cases described in the single-step test for the dish piling. The results are shown in Table II. Here, we perform 10 AL loops for each test case, and measure the number of inner iterations (i.e., Newton steps) throughout the entire AL loop, the failure rate (if the inner loop fails to find the surrogate problem solution within the desired tolerance), and the residuals of the resulting solutions.

As indicated in the table, the number of inner iterations is significantly lower for CANAL. This reduction is due to its cascaded structure, which can transform solving the surrogate problem into solving a convex optimization problem. Consequently, this structure allows for a convergence guarantee to the global minimum with exact line search. However, both TRDogleg and DampedSN often gets stuck in a zone where it cannot effectively reduce the merit function, leading to requirement of more inner iteration steps and occasional failure. TRDogleg relatively performs better than DampedSN, with half the required inner iterations. Comparing the residuals after 10 iterations, CANAL exhibits similarly low residuals compared to the two baselines. This indicates that updating $\tilde{e}_i$ in every AL iteration within the cascaded structure does not substantially degrade the convergence properties of the entire algorithm.

2) SubADMM: Next, we compare our SubADMM with ADMM implementations based on more standard splitting strategies. For baselines, we implement the following methods: 1) ADMM without subsystem-based splitting (`No Sub`), as described earlier in Sec. VI, which requires solving a full system size linear equation (28), and 2) ADMM that utilizes subsystem-based splitting but does not employ body-based splitting (30) (`No Body`), which is equivalent to the algorithm in our prior work [34].

For the test, we set up an environment where multiple quadrupedal robots Laikago, are dropped onto the floor, as

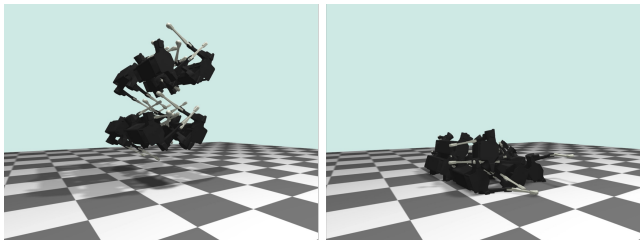

Fig. 12. Snapshots of a multiple Laikago dropping simulation. The system is divided into 8 subsystems, each interpreted as a kinematic tree.

depicted in Fig. 12. In this scenario, each robot possesses 18 DOF, resulting in a total system DOF of 144. For the collision geometry, we approximate the trunk, legs, and feet using simple primitives such as boxes and spheres. Typically, $80-100$ contacts are generated during the simulation. We use fixed 100 iterations for SubADMM and other baselines.

The comparison results are shown in Table III. As demonstrated, the ablation studies highlight the advantages of the proposed techniques. SubADMM achieves the shortest computation time due to its ability to utilize efficient matrix assembly, parallelized matrix factorization and solving. `No Sub` takes the longest computation time, requiring about 2.78 times more than SubADMM primarily due to its need for whole system size matrix factorization. Compared to `No Body`, SubADMM is about 1.46 times faster, as `No Body` cannot exploit efficient submatrix factorization based on the structure given in (40). Such differences could become more pronounced as the overall system dimension increases, or through the employment of advanced code parallelization techniques in SubADMM. For example, in 27 Laikago dropping simulation, we find SubADMM is 3.63 times faster than `No Sub`. The residuals for all three solvers are similar, as they share similar theoretical convergence properties of ADMM.

VIII. Discussions and Concluding Remarks

In this article, we introduce two multi-contact solver algorithms, CANAL and SubADMM, based on variations of the augmented Lagrangian method. Our formulation extends the theory of AL to handle multi-contact NCP by iteratively solving surrogate problems and subsequently updating primal and dual variables. In CANAL, we variate this AL-based structure into a cascaded form of convex optimization, which can be solved by exact Newton steps, thereby ensuring accurate and robust simulation results. In SubADMM, we employ the concept of ADMM to enable an alternating solution approach to the surrogate problem. Here, we propose a novel subsystem-based variable splitting method, which not only achieves a parallelizable structure but also preserves the sparsity pattern of the submatrix, significantly improving efficiency. The examples demonstrate their effectiveness in various robotic simulations characterized by intensive contact formation and stiff interactions, and also illustrate the trade-offs between CANAL, SubADMM, and other existing methods.

A variety of future research areas remain open within the presented framework. From an algorithmic perspective, the CANAL algorithm could be tested with other convex optimization methods, such as the conjugate gradient [57] or accelerated projected gradient [46], for example. SubADMM could be enhanced with various strategies to improve its convergence, including the acceleration of fixed-point iterations [58] and advanced penalty parameter update schemes. Although our initial trials observed that the application of these schemes could degrade the robustness of the simulation, the development of a solid methodology still remains an open question. From an implementation perspective, several component of the current framework can be improved. For instance, tailored factorization based on the branch-induced sparsity structure could be adopted for CANAL. Additionally, a GPU implementation for SubADMM could fully exploit its parallelizable nature.

We believe that the algorithms presented in this article can be further employed in the development of other model-based solvers for robotic applications. For example, our problem described in (8), when formulated without friction, becomes equivalent to a quadratic programming problem, which is commonly encountered in motion primitives, planning, and model predictive control of robotic systems. Moreover, a forward simulation solver can be coupled with the differentiation of the results, utilized to address diverse inverse problems involving contact [22]. Our highly accurate solutions are particularly beneficial from this perspective.

Finally, while the focus of this article is on contact solvers, we also believe that contact modeling is a crucial aspect of robotic simulation. This includes the representation of geometry, definition of contact features, time stepping, friction modeling, and often linked with the contact solver. In this regard, investigating how our AL-based solver can be effectively integrated with various aspects, including continuous collision detection [59], contact fields [60], temporal position updates [17], anisotropic friction, and lubrication [61], will be a valuable topic to explore.

Appendix A
Composite Rigid Body Algorithm for Subsystem Matrices

Contact Jacobian with respect to spatial velocity of the body can be written as follows:

$$J_{i,b_k} = R_i \begin{bmatrix} I_{3\times 3} & -[p_i] \end{bmatrix} \tag{44}$$

where $R_i \in \mathrm{SO}(3)$ is the contact frame generated through the contact normal and p_i is the global position of the contact point. Based on (44), (41) can be written as follows:

$$\begin{aligned} H_{b_k} &= M_{b_k} + \sum_i \beta J_{i,b_k}^T J_{i,b_k} \\ &= \begin{bmatrix} m_{b_k} I_{3\times 3} & -m_{b_k}[c_{b_k}] \\ m_{b_k}[c_{b_k}] & I_{b_k} - m_{b_k}[c_{b_k}]^2 \end{bmatrix} + \sum_i \beta \begin{bmatrix} I_{3\times 3} & -[p_i] \\ [p_i] & -[p_i]^2 \end{bmatrix} \end{aligned}$$

where m_{b_k} is the mass, I_{b_k} is the moment of inertia, c_{b_k} is the global center of mass position of the body. It can be easily verified that H_{b_k} shares the same fill-in structure as M_{b_k}, with only 10 elements required for matrix storage. Therefore, as in the original composite rigid body algorithm, addition and multiplication (of spatial transformation matrix) operations can be performed with equal efficiency.

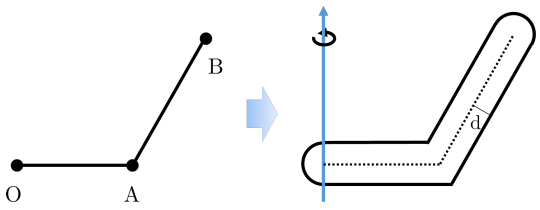

Fig. 13. Illustrations for dish signed distance function module generation.

APPENDIX B
SIGNED DISTANCE FUNCTION OF DISH MODULE

In our experiments, signed distance function of dishes are defined as follows:

$$\mathrm{SDF}_{3\mathrm{d}}(p) = \mathrm{SDF}_{2\mathrm{d}}\left(\begin{bmatrix}\sqrt{p_1^2+p_2^2} & p_3\end{bmatrix}\right)$$

which is indeed revolution of following 2D signed distance function (see also Fig. 13):

$$\mathrm{SDF}_{2\mathrm{d}}(p) = \min\left(\mathrm{dist}(p, \overline{OA}), \mathrm{dist}(p, \overline{AB})\right) - d \qquad (45)$$

where A, B are points on the plane, and d is a thickness for padding. By adjusting A, B and d in (45), we can generate diverse range of dishes. Additionally, the derivative of (45) can be computed analytically, and thus it can be utilized in the collision detection process.

REFERENCES

[1] Bullet physics engine. https://pybullet.org/.
[2] Mujoco physics engine. http://www.mujoco.org/.
[3] Physx physics engine. https://developer.nvidia.com/physx-sdk.
[4] Brax - a differentiable physics engine for large scale rigid body simulation. http://github.com/google/brax.
[5] Raisim physics engine. https://raisim.com/.
[6] Sofa physics engine. https://www.sofa-framework.org/.
[7] 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*, 39(15):2673–2691, 1996.
[8] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.
[9] J. E. Lloyd. Fast implementation of lemke's algorithm for rigid body contact simulation. In *IEEE International Conference on Robotics and Automation*, 2005.
[10] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500, 2018.
[11] Open dynamics engine. http://ode.org.
[12] M. Macklin, M. Müller, N. Chentanez, and T. Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics*, 33(4), 2014.
[13] M. Macklin, M. Müller, and N. Chentanez. Xpbd: Position-based simulation of compliant constrained dynamics. In *International Conference on Motion in Games*, pages 49–54, 2016.
[14] E. Todorov. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *IEEE International Conference on Robotics and Automation*, pages 6054–6061, 2014.
[15] P. C. Horak and J. C. Trinkle. On the similarities and differences among contact models in robot simulation. *IEEE Robotics and Automation Letters*, 4(2):493–499, 2019.
[16] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3(2):895–902, 2018.
[17] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller. Small steps in physics simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2019.
[18] M. Müller, M. Macklin, N. Chentanez, S. Jeschke, and T. Kim. Detailed rigid body simulation with extended position based dynamics. In *Computer Graphics Forum*, volume 39, pages 101–112. Wiley Online Library, 2020.
[19] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.
[20] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics*, 39(6):1–15, 2020.
[21] A. M. Castro, A. Qu, N. Kuppuswamy, A. Alspach, and M. Sherman. A transition-aware method for the simulation of compliant contact with regularized friction. *IEEE Robotics and Automation Letters*, 5(2):1859–1866, 2020.
[22] T. A. Howell, S. Le Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 9, 2022.
[23] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviychuk. Non-smooth newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics*, 38(5):1–20, 2019.
[24] A. M. Castro, F. N Permenter, and X. Han. An unconstrained convex formulation of compliant contact. *IEEE Transactions on Robotics*, 39(2):1301–1320, 2022.
[25] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.
[26] J. J. Moreau. Fonctions convexes duales et points proximaux dans un espace hilbertien. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 255:2897–2899, 1962.
[27] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.
[28] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12:637–672, 2020.
[29] B. Hermans, A. Themelis, and P. Patrinos. Qpalm: a newton-type proximal augmented lagrangian method for quadratic programs. In *IEEE Conference on Decision and Control*, pages 4325–4330. IEEE, 2019.
[30] P. Sopasakis, K. Menounou, and P. Patrinos. Superscs: fast and accurate large-scale conic optimization. In *European Control Conference*, pages 1500–1505, 2019.
[31] B. Bazzana, H. Andreasson, and G. Grisetti. How-to augmented lagrangian on factor graphs. *IEEE Robotics and Automation Letters*, 2024.
[32] T. A. Howell, B. E. Jackson, and Z. Manchester. Altro: A fast solver for constrained trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7674–7679, 2019.
[33] J. Carpentier, R. Budhiraja, and N. Mansard. Proximal and sparse resolution of constrained dynamic equations. In *Robotics: Science and Systems*, 2021.
[34] J. Lee, M. Lee, and D. Lee. Modular and parallelizable multibody physics simulation via subsystem-based admm. In *IEEE International Conference on Robotics and Automation*, pages 10132–10138, 2023.
[35] M. Kim, Y. Lee, Y. Lee, and D. J. Lee. Haptic rendering and interactive simulation using passive midpoint integration. *International Journal of Robotics Research*, 36(12):1341–1362, 2017.
[36] V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick. A formulation of the linear discrete coulomb friction problem via convex optimization. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 91(2):155–175, 2011.
[37] G. Daviet. Simple and scalable frictional contacts for thin nodal objects. *ACM Transactions on Graphics*, 39(4), 2020.
[38] M. Verschoor and A. C. Jalba. Efficient and accurate collision response for elastically deformable models. *ACM Transactions on Graphics*, 38(2), 2019.
[39] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 1(1):1–16, 1972.
[40] S. Andrews, M. Teichmann, and P. G. Kry. Geometric stiffness for real-time constrained multibody dynamics. *Computer Graphics Forum*, 36(2):235–246, 2017.
[41] J. Pan, S. Chitta, and D. Manocha. Fcl: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation*, pages 3859–3866, 2012.
[42] M. Lee, J. Lee, and D. Lee. Differentiable dynamics simulation using invariant contact mapping and damped contact force. In *IEEE*

International Conference on Robotics and Automation, pages 11683–11689, 2023.

[43] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning, 2011.

[44] J. Wang, F. Yu, X. Chen, and L. Zhao. Admm for efficient deep learning with global convergence. In *ACM International Conference on Knowledge Discovery & Data Mining*, pages 111–119, 2019.

[45] Y. Dai and L. Zhang. The augmented lagrangian method can approximately solve convex optimization with least constraint violation. *Mathematical Programming*, 200(2):633–667, 2023.

[46] J. Lee, M. Lee, and D. Lee. Large-dimensional multibody dynamics simulation using contact nodalization and diagonalization. *IEEE Transactions on Robotics*, 39(2):1419–1438, 2022.

[47] M. Hintermüller. Semismooth newton methods and applications. *Department of Mathematics, Humboldt-University of Berlin*, 2010.

[48] J. Nocedal and S. J Wright. *Numerical optimization*. Springer, 1999.

[49] R. Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[50] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[51] P. Giselsson and S. Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2016.

[52] Gaël G., Benoît J., et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[53] J. Yoon, M. Lee, D. Son, and D. Lee. Fast and accurate data-driven simulation framework for contact-intensive tight-tolerance robotic assembly tasks. *arXiv preprint arXiv:2202.13098*, 2022.

[54] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse. Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(1):1–17, 2020.

[55] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[56] T. De Luca, F. Facchinei, and C. Kanzow. A semismooth equation approach to the solution of nonlinear complementarity problems. *Mathematical programming*, 75:407–439, 1996.

[57] R. Fletcher and C. M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[58] A. Themelis and P. Patrinos. Supermann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. *IEEE Transactions on Automatic Control*, 64(12):4875–4890, 2019.

[59] M. Li, Z. Ferguson, T. Schneider, T. R. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M Kaufman. Incremental potential contact: intersection- and inversion-free, large-deformation dynamics. *ACM Transactions on Graphics*, 39(4):49, 2020.

[60] A. Castro, X. Han, and J. Masterjohn. A theory of irrotational contact fields. *arXiv preprint arXiv:2312.03908*, 2023.

[61] K. C. Ludema and L. Ajayi. *Friction, wear, lubrication: a textbook in tribology*. CRC press, 2018.