

A Convex Formulation of Material Points and Rigid Bodies with GPU-Accelerated Async-Coupling for Interactive Simulation

Chang Yu^{*1}, Wenxin Du^{*1}, Zeshun Zong¹, Alejandro Castro², Chenfanfu Jiang¹, Xuchen Han²

Abstract—We present a novel convex formulation that weakly couples the Material Point Method (MPM) with rigid body dynamics through frictional contact, optimized for efficient GPU parallelization. Our approach features an asynchronous time-splitting scheme to integrate MPM and rigid body dynamics under different time step sizes. We develop a globally convergent quasi-Newton solver tailored for massive parallelization, achieving up to 500 \times speedup over previous convex formulations without sacrificing stability. Our method enables interactive-rate simulations of robotic manipulation tasks with diverse deformable objects including granular materials and cloth, with strong convergence guarantees. We detail key implementation strategies to maximize performance and validate our approach through rigorous experiments, demonstrating superior speed, accuracy, and stability compared to state-of-the-art MPM simulators for robotics. We make our method available in the open-source robotics toolkit, Drake.

I. INTRODUCTION

With the rise of robotics foundation models [1], simulation has become an indispensable tool for both policy training and evaluation [2], [3]. Simulations enable generating large-scale training data at a fraction of the cost of real-world collection and facilitate evaluating model checkpoints in controlled environments, ensuring repeatability. In these settings, modern physics simulators for robotics must satisfy several key requirements to be effective: (i) strong numerical stability guarantees, (ii) sufficient accuracy to capture real-world success and failure modes, (iii) computational efficiency to support data collection and policy evaluation at scale, and (iv) the ability to model diverse environments.

Among existing physics simulators for robotics, Drake [4] and MuJoCo [5] are well-regarded for their strong robustness guarantees and high accuracy required for manipulation tasks. Both employ a convex formulation of frictional contact, ensuring stability and global convergence [6], [7]. However, despite recent advancements in deformable body simulation within the convex framework [8], [9], [10], these simulators struggle to efficiently simulate deformable bodies with a large number of degrees of freedom (DoFs) at interactive rates. This limitation arises partly from the need to solve poorly-conditioned convex optimization problems that necessitates direct linear solvers with $O(n^3)$ complexity [7]. The inclusion of deformable bodies exacerbates this challenge, as it significantly increases the number of DoFs—often by orders of magnitude—making efficient solutions increasingly

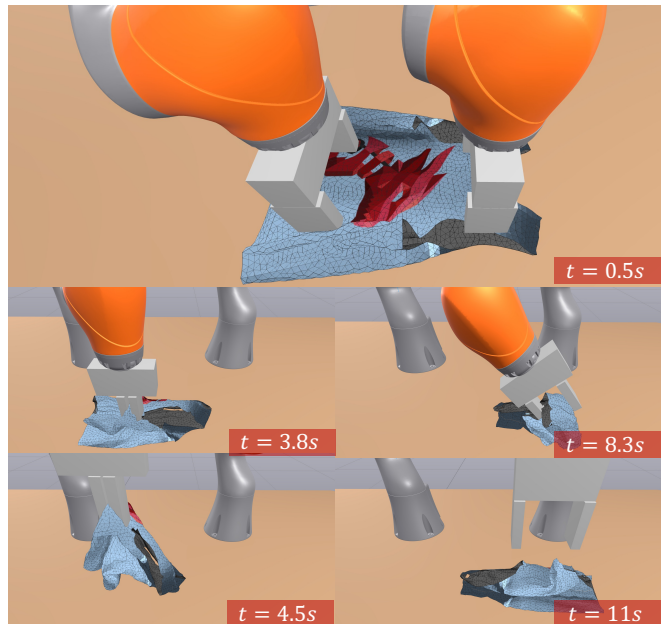


Fig. 1: A challenging T-shirt folding task demonstrating the accuracy, speed, and robustness of our method. Two robotic arms fold and unfold the T-shirt while our approach accurately captures frictional interactions, effectively handling MPM-based cloth self-collisions and rigid-MPM coupling without penetration. See the supplemental video for details.

difficult. Moreover, the inherently serial nature of direct linear solvers precludes efficient parallelization on GPUs, further restricting scalability.

Among various methods for simulating deformable materials, the Material Point Method (MPM) has gained traction due to its ability to handle large deformations and topology changes [11]. However, existing simulators struggle to achieve both robustness and efficiency [10]. In this paper, we propose a novel convex formulation for coupling MPM with rigid bodies through frictional contact, designed for efficient GPU parallelization. We prove the stability and global convergence of our frictional contact model. Furthermore, we demonstrate that our method efficiently simulates a wide range of materials while preserving the high accuracy required for robotic manipulation tasks.

II. PREVIOUS WORK

A. Physics-based Deformable Body Simulation

Physics-based deformable body simulation has been a fundamental topic in computer graphics, computational physics,

^{*} equal contribution.

¹{changyu1, wenxindu, zeshunzong, cffjiang}@ucla.edu, AIVC Laboratory, UCLA, USA.

²{alejandro.castro, xuchen.han}@tri.global, Toyota Research Institute, USA.

and engineering for decades, with applications ranging from material design to visual effects and virtual surgery. The Finite Element Method (FEM) has been widely used for simulating elastic and plastic deformations due to its accuracy in solving continuum mechanics problems [12], [13]. However, FEM often requires high-quality meshes [14] and struggles with large deformations and topological changes [15]. Additionally, handling contact among deformable bodies, including collision detection, requires significant computational effort [16], [17].

To overcome these limitations, MPM has emerged as a powerful alternative due to its ability to handle large deformations, topological changes, and naturally resolve collision responses [18], [19]. It has been widely applied in simulations of granular materials [20], volumetric and codimensional deformable bodies [21], [22], and fluid-solid coupling [23]. Explicit integration in MPM is well-suited for massive parallelization on GPUs [24], [25]. However, implicit integration remains computationally expensive [26], especially when coupling MPM with rigid bodies for frictional contact [10]. This coupling results in a large, often poorly-conditioned convex optimization problem, with no existing method efficiently solving it at interactive rates. To address this challenge, researchers have proposed asynchronous timestepping schemes, integrating MPM explicitly with smaller time steps to exploit massive parallelization while solving the remaining system at larger time steps [27], [11]. Our approach follows a similar asynchronous strategy but, to our knowledge, is the first to formulate the coupling between MPM and rigid bodies as an implicit convex optimization problem with a unique solution and global convergence guarantees, ensuring numerical stability.

B. Frictional Contact

Contact modeling in robotics is a broad topic beyond the scope of this paper; we refer readers to [28] for a comprehensive overview. Here, we focus on frictional contact involving deformable bodies, which is central to our work.

Many simulators, such as SOFA [13] and PhysX [29], formulate frictional contact as a linear complementarity problem (LCP) and solve it using projected Gauss-Seidel (PGS). Others address the nonlinear complementarity problem (NCP) directly with methods such as PGS [30], non-smooth Newton solvers [31], interior point methods [32], or the alternating direction method of multipliers (ADMM) [33]. While these approaches can model hard contact, they lack convergence guarantees, which can lead to robustness issues in practice. Incremental Potential Contact (IPC) [16] offers strong non-penetration guarantees and demonstrates robust performance in practice [34]. However, IPC loses convergence guarantees when friction is present, and its computational cost makes it less suitable for interactive simulations.

A different approach embraces compliant contact, which was first popularized in robotics by MuJoCo [6]. With a focus on physical accuracy, Drake employs introduces convex approximations of engineering-grade contact models in [35], later extended to support frictional contact between FEM

and rigid bodies [9]. Moreover, Drake employs hydroelastic contact [36], [37], [7] to model continuous contact patches. More recently, Zong et al. [10] extended this approach to couple MPM with rigid bodies, ensuring robustness and global convergence. However, this method is inherently serial and remains far from real-time performance. In this paper, we build upon [10] and propose a GPU-accelerated approach that achieves interactive rates without compromising global convergence guarantees.

C. Hardware Acceleration

Researchers have explored two primary approaches to accelerating physics simulation using modern vectorized hardware like GPUs and TPUs. The first focuses on speeding up single large-scale simulations, enabling scenes with up to hundreds of millions of DoFs [38]. Significant progress has been made in accelerating FEM [39], MPM [24], [25], [40], and Discrete Element Method (DEM) [41]. However, some of these methods remain far from real-time execution, while others do not support two-way coupling with rigid bodies.

The second approach emphasizes increasing simulation throughput, particularly for reinforcement learning and robotics training. Frameworks like MJX [8], Isaac Gym [42], Maniskill [43], and Genesis [44] leverage hardware acceleration to run thousands of parallel simulations, improving data collection speed for policy learning.

Our GPU acceleration strategy aligns more with the first approach. However, rather than targeting massive simulations with millions of DoFs, we focus on interactive-rate simulations with thousands to tens of thousands of DoFs—expressive enough for robotic manipulation tasks. Unlike prior work prioritizing raw speed, our method emphasizes convergence and stability, ensuring reliable performance for robotics applications.

III. OUTLINE AND NOVEL CONTRIBUTIONS

In this work, we introduce several key contributions to advance the state of the art in efficient and robust deformable body simulation for robotics. First, we propose an asynchronous time-splitting scheme that enables integrating MPM and rigid body dynamics under significantly different time step sizes (Section IV-A). Second, we develop a convex formulation for the weak coupling of MPM and rigid bodies through frictional contact, ensuring global convergence (Section IV-B). Third, we design a quasi-Newton solver that is well-suited for massive parallelization, enabling efficient execution on GPUs (Section IV-C). We detail the key implementation strategies required for high-performance execution on GPUs in Section V. We thoroughly validate our approach through a suite of robotic manipulation simulation scenarios and demonstrate the speed, accuracy, and robustness of our method in Section VI. Finally, we release our implementation as part of the open-source robotics toolkit Drake [4], providing the robotics community with a scalable and reproducible solution for high-fidelity deformable body simulation.

IV. MATHEMATICAL FORMULATION

A. Asynchronous Time-Splitting Scheme

We discretize time into intervals of size Δt to advance the system dynamics from t_n to the next time step $t_{n+1} := t_n + \Delta t$ using the governing equation:

$$\mathbf{M}(\mathbf{q})(\mathbf{v}^{n+1} - \mathbf{v}^n) = \Delta t \mathbf{k}(\mathbf{q}, \mathbf{v}) + \hat{\mathbf{J}}^T(\mathbf{q})\hat{\gamma}(\mathbf{q}, \mathbf{v}), \quad (1)$$

where \mathbf{q} and \mathbf{v} are generalized positions and velocities, γ is the contact and friction impulses, \mathbf{k} represents all non-contact forces, and $\hat{\mathbf{J}}$ is the contact Jacobian. Using subscripts d for MPM DoFs and r for rigid body DoFs, we define:

$$\begin{aligned} \mathbf{q}^T &= [\mathbf{q}_r^T, \mathbf{q}_d^T], & \mathbf{v}^T &= [\mathbf{v}_r^T, \mathbf{v}_d^T], \\ \mathbf{k}^T &= [\mathbf{k}_r^T, \mathbf{k}_d^T], & \mathbf{M} &= \text{diag}(\mathbf{M}_r, \mathbf{M}_d). \end{aligned} \quad (2)$$

Specifically, \mathbf{q}_d and \mathbf{v}_d denote the MPM *particle* positions and *grid* velocities; \mathbf{k}_d includes elastic-plastic and external forces on MPM DoFs; \mathbf{q}_r and \mathbf{v}_r are the articulated rigid body joint positions and velocities; \mathbf{k}_r captures Coriolis terms and gravity on rigid body DoFs.

We further separate contact between MPM and rigid bodies from contact among rigid bodies as:

$$\hat{\gamma} = \begin{pmatrix} \gamma_d \\ \gamma_r \end{pmatrix}, \quad \hat{\mathbf{J}} = \begin{pmatrix} \mathbf{J}_r & \mathbf{J}_d \\ \mathbf{J}_{rr} & \mathbf{0} \end{pmatrix}.$$

Notably, we do not explicitly model contact forces among MPM DoFs, as these are naturally resolved through the hybrid Eulerian/Lagrangian representation with constitutive and plasticity modeling [22], [21].

To achieve efficient GPU parallelization without sacrificing stability, we integrate elastic forces explicitly using symplectic Euler, while treating stiff frictional contact forces implicitly via backward Euler. However, symplectic Euler requires smaller time step sizes than backward Euler for stability, motivating our asynchronous time-splitting approach:

$$\begin{aligned} \mathbf{M}_r(\mathbf{q}_r^n)(\mathbf{v}_r^{n+1} - \mathbf{v}_r^n) &= \Delta t \mathbf{k}_r(\mathbf{q}_r^n, \mathbf{v}_r^n) + \mathbf{J}_{rr}^T \gamma_r(\mathbf{v}_r^{n+1}) \\ &+ \mathbf{J}_r^T \sum_{k=0}^{N-1} \gamma_d^{n,k}, \end{aligned} \quad (3)$$

$$\mathbf{M}_d(\mathbf{q}_d^{n,k})(\mathbf{v}_d^{n,k+1} - \mathbf{v}_d^{n,k}) = \frac{\Delta t}{N} \mathbf{k}_d(\mathbf{q}_d^n) + \mathbf{J}_d^T \gamma_d^{n,k}, \quad (4)$$

for $k = 0, \dots, N-1$, with $\mathbf{v}_d^{n,0} = \mathbf{v}_d^n$, and $\mathbf{v}_d^{n,N} = \mathbf{v}_d^{n+1}$.

Here, $\gamma_d^{n,k} = \gamma_d(\mathbf{v}_r^n, \mathbf{v}_d^{n,k+1})$ is the frictional impulse between rigid bodies and MPM at substep k . Notice that in Eq. (4), we decompose a single time step into N substeps. Elastic forces \mathbf{k}_d are integrated explicitly to facilitate parallelization, whereas contact forces γ_d are integrated implicitly with respect to $\mathbf{v}_d^{n,k+1}$ for stability. Throughout all substeps in a single timestep, the rigid body velocity is fixed at \mathbf{v}_r^n . The contact impulses at each substep are accumulated and applied to rigid body DoFs in Eq. (3). This approach results in a weak coupling between rigid bodies and MPM DoFs, in contrast to the strong coupling scheme proposed by [10]. We evaluate the accuracy of our weak coupling scheme in

Section VI.

B. Convex Formulation

The integration of rigid DoFs in Eq. (3) follows the same methodology as described in [35]. We refer readers to that work for detailed explanations and implementation practices. Here, we focus on the integration of Eq. (4) for MPM DoFs. For notational simplicity, we drop the subscript d and the superscript n in Eq. (4), reducing it to an algebraic difference equation that advances a substep of MPM:

$$\mathbf{M}(\mathbf{v}^{k+1} - \mathbf{v}^k) = \frac{\Delta t}{N} \mathbf{k} + \mathbf{J}^T \gamma(\mathbf{v}_c). \quad (5)$$

Here, $\mathbf{v}_c = \mathbf{J}\mathbf{v}^{k+1} + \mathbf{b}_r$ denotes the relative contact velocity between particles and the rigid body they are in contact with, expressed in the contact frame, where the contact frame is a local frame with the z -axis aligned with the contact normal. The term $\mathbf{b}_r = \mathbf{J}_r \mathbf{v}_r^n$ represents the bias velocity from the rigid body in the contact frame.

We solve Eq. (5) in two stages by performing another time-splitting. First, we compute the *free motion velocity*, \mathbf{v}^* , which is the velocity the MPM DoFs would attain in the absence of contact forces:

$$\mathbf{M}(\mathbf{v}^* - \mathbf{v}^k) = \frac{\Delta t}{N} \mathbf{k}. \quad (6)$$

This step is equivalent to a standard explicit MPM step. We adopt the moving least-squares formulation described in [45] for this computation.

In the second stage, we compute the *post-contact velocity* \mathbf{v}^{k+1} by solving:

$$\mathbf{M}(\mathbf{v}^{k+1} - \mathbf{v}^*) = \mathbf{J}^T \gamma(\mathbf{v}_c). \quad (7)$$

To ensure global convergence, we reformulate Eq. (7) as a convex optimization problem [35]:

$$\mathbf{v}^{k+1} = \arg \min_{\mathbf{v}} \ell_p = \arg \min_{\mathbf{v}} \frac{1}{2} \|\mathbf{v} - \mathbf{v}^*\|_{\mathbf{M}}^2 + \ell_c(\mathbf{v}_c). \quad (8)$$

The term $\ell_c(\mathbf{v}_c)$ is the contact potential energy, defined such that $\gamma(\mathbf{v}_c) = -\partial \ell_c / \partial \mathbf{v}_c$. With mass lumping, \mathbf{M} is diagonal and positive definite, ensuring that problem (8) is strongly convex as long as the frictional contact model is designed with a convex ℓ_c . We adopt the *lagged contact model* in [35].

C. Quasi-Newton Solver

A key advantage of our weak coupling scheme is that, instead of requiring the Schur complement of the Jacobian of the *momentum residual* as in [10]—which is unfriendly to GPU parallelization—our formulation replaces it with the diagonal mass matrix, which is trivial to parallelize. However, the Hessian $\hat{\mathbf{H}}$ of ℓ_p still contains off-diagonal entries from the Hessian of ℓ_c . Given the massive parallelization capabilities of GPUs, it is more efficient to trade additional solver iterations for better parallelization (see Section VI-B). Therefore, we adopt a quasi-Newton strategy by approximating $\hat{\mathbf{H}}$ with its 3×3 block diagonal counterpart \mathbf{H} to avoid the inherently serial Cholesky factorization of $\hat{\mathbf{H}}$. Algorithm 1 summarizes our quasi-Newton solver, leveraging

Algorithm 1 Quasi-Newton solver for problem (8)

```
1: Initialize  $\mathbf{v} \leftarrow \mathbf{v}^k$ 
2: repeat until  $\|\tilde{\nabla} \ell_p\| < \varepsilon_a + \varepsilon_r \max(\|\tilde{\mathbf{p}}\|, \|\tilde{\mathbf{J}}_c\|)$ ,
3:    $\Delta \mathbf{v} = -\mathbf{H}^{-1}(\mathbf{v}) \nabla_{\mathbf{v}} \ell_p(\mathbf{v})$ 
4:    $\alpha^* = \arg \min_{\alpha > 0} \ell_p(\mathbf{v} + \alpha \Delta \mathbf{v})$ 
5:    $\mathbf{v} = \mathbf{v} + \alpha^* \Delta \mathbf{v}$ 
6: return  $\{\mathbf{v}, \gamma = \frac{\partial \ell_c}{\partial \mathbf{v}_c} \big|_{\mathbf{v}_c(\mathbf{v})}\}$ 
```

the convergence criteria from [7] and [10] to ensure robustness against large mass ratios and enable fair comparison in Section VI-B. The resulting γ is accumulated into Eq. (3).

We conclude this section by noting that Algorithm 1 is globally convergent with at least a linear rate, as proved in the Appendix. In practice, the algorithm can be efficiently parallelized on the GPU, as detailed in Section V-B.

V. GPU IMPLEMENTATION

A. GPU Optimization for the Free Motion Solver

The explicit MPM time integration procedure for the *free motion* solve in Eq. (6) involves three main operations: (1) transferring particle attributes to the Eulerian grid, (2) updating grid values at each grid node, and (3) transferring updated grid values back to particles. For a detailed overview of the MPM algorithm, we refer readers to [19].

The key to high-performance MPM lies in optimizing particle-to-grid transfer operators, where we use one GPU thread to perform the computation for a single particle, and addressing write conflicts that occur during parallelization when multiple particles write to the same grid node is very important for performance. Conflicts within a GPU warp are particularly detrimental to performance. There are two known strategies to mitigate such conflicts:

- **Randomization:** Proposed by [38], [40], this approach randomly shuffles particles. By introducing randomness into particle order, the chances of particles from different warps writing to the same grid location are reduced, as the large number of particles and grid nodes makes such conflicts statistically rare.
- **Warp-Level Reduction:** Introduced by [25], [24], this approach uses warp-level reduction before atomically writing data to grid nodes. It requires sorting particles into cells to maintain the order of reduction, offering more controlled conflict management.

We adopt the warp-level reduction strategy combined with a partial sorting approach. When particles are fully sorted such that particles transferring to the same grid nodes are consecutive and grouped together, reductions are first independently performed within each particle group. This operation is efficiently implemented using warp-level CUDA intrinsic functions [25]. The thread associated with the first particle in each group then writes the reduced result back to global memory. This ensures that each group performs only one atomic operation, significantly reducing atomic operation overhead. This method is particularly effective for

interactive-rate simulations with small problem sizes, where the GPU’s computational throughput cannot be fully utilized, and memory access becomes the bottleneck. However, sorting at each substep is time-consuming due to kernel launches and implicit host-device synchronizations. To address this, we leverage the spatial continuity of simulations to reuse sorting results from the previous time step. In practice, sorting is only needed once per time step, allowing all substeps within that time step to reuse the sorted data, forming our partial sorting strategy. We use a radix sort with 10-bit keys instead of the full bit-width to enhance efficiency. While this approach may lead to suboptimal particle ordering and cause particles in the same cell being distributed across multiple warps, resulting in more atomic operations, the performance gains from reduced sorting time outweigh this drawback.

B. Parallel Quasi-Newton Solver Implementation

At the beginning of each substep, we register a contact point for each particle that is inside a rigid geometry, creating one contact point per particle-geometry pair. The contact velocity at the p -th contact point is given by

$$\mathbf{v}_{c,p} = \mathbf{J}_p \mathbf{v}_d + \mathbf{b}_p.$$

The first term represents the velocity of particle p expressed in the contact frame, which can be efficiently computed in parallel using a grid-to-particle kernel. The second term, the bias velocity of the rigid body evaluated at the contact point, is constant throughout all substeps within a single time step, allowing it to be precomputed and reused across all substeps.

Similarly, the gradient and Hessian of the objective function in problem (8) can be efficiently computed in parallel:

$$\nabla_{\mathbf{v}} \ell_p = \mathbf{M}(\mathbf{v} - \mathbf{v}^*) + \mathbf{J}^T \frac{\partial \ell_c}{\partial \mathbf{v}_c}, \quad (9)$$

$$\hat{\mathbf{H}} = \mathbf{M} + \mathbf{J}^T \frac{\partial^2 \ell_c}{\partial \mathbf{v}_c^2} \mathbf{J}. \quad (10)$$

The first terms involving the mass matrix \mathbf{M} is trivially parallelized because, with mass lumping, they reduce to a simple sweep over all grid nodes. For the second terms, we first compute the local 3-dimensional gradient vector $\frac{\partial \ell_c}{\partial \mathbf{v}_c}$ and the 3×3 Hessian matrix $\frac{\partial^2 \ell_c}{\partial \mathbf{v}_c^2}$ for each contact point. We then execute a single particle-to-grid kernel for all particles in contact to accumulate their contributions, leveraging the fact that \mathbf{J} is the MPM transfer operator that maps grid velocity in the world frame to the particle velocities in the contact frame. By only keeping the diagonal terms when computing $\hat{\mathbf{H}}$, we obtain the approximate Hessian \mathbf{H} used in step 3 of Algorithm 1. Since \mathbf{H} is block diagonal, the linear system in step 3 decouples into independent 3×3 subsystems, which can be efficiently solved in parallel using Cholesky factorizations. After obtaining the search direction, we perform the line search in step 4 using a one-dimensional Newton’s method that falls back to bisection when convergence stalls, as described in [7]. Similar to the computations in Eq. (9) and Eq. (10), the objective function and its gradient and Hessian required by the one-dimensional Newton solver are efficiently computed through a single

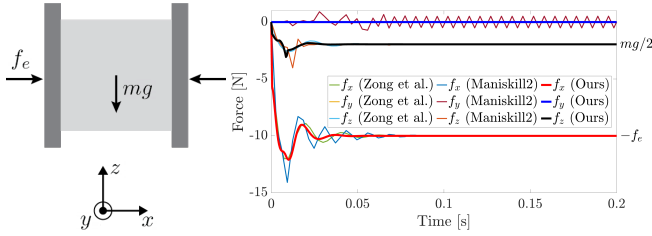


Fig. 2: **(Left)** Two rigid panels apply a normal force of $f_e = 10$ N in the x -direction to hold an elastic MPM box in place against gravity. **(Right)** The contact forces exerted by the elastic box on one of the rigid panels is compared using our method, ManiSkill2, and Zong et al. The corresponding analytical steady-state values are indicated on the right.

particle-to-grid transfer for the ℓ_c term and a simple parallel reduction over the grid for the M term. With this strategy, the entire substep, including solving Eq. (6) and Eq. (7), is efficiently parallelized and entirely executed on the GPU, with our optimization practice in Section V-A.

VI. RESULTS

We present several test cases to showcase the efficiency, accuracy, and robustness of our method. All simulations are run on a system with an Intel(R) Xeon(R) CPU E5-2690 v4 processor (56-core) and 128 GB of RAM, and an RTX 4090 with 24 GB device memory. We report scene statistics, including the Eulerian grid spacing h , MPM particles count, the time step size Δt , the number of substeps N , runtime performance, and number of contacts throughout the simulation in Table I. All simulations are solved to convergence with $\varepsilon_r = 5 \times 10^{-2}$ and ε_a set to machine epsilon unless otherwise specified.

A. Comparison with Gu et al. [11] and Zong et al. [10]

We validate the accuracy of our convex formulation and parallel quasi-Newton solver in a comparison against the method adopted in ManiSkill2 [11], a state-of-the-art embodied AI simulation environment with MPM support, and Zong et al. [10], which tightly couples MPM and rigid bodies using a convex formulation.

We replicate the experimental setup from [10], where an elastic cube is compressed by two rigid panels and held in place by friction, as shown in Fig. 2. Our method is simulated with $\Delta t = 0.1$ ms and $N = 1$ substep, while Gu et al. uses $\Delta t = 0.01$ ms with 25 substeps for its explicit MPM integration, and Zong et al. employs $\Delta t = 10$ ms due to its fully implicit formulation.

The contact forces exerted by the cube on the left panel are shown in Fig. 2. The normal and z -direction friction forces converge to the analytical solutions after the initial transient period for all methods. Consistent with the observations in [10], the method by Gu et al. exhibits high-frequency oscillations in the y -direction friction force, as its explicit rigid-MPM coupling struggles with numerical stability in stiction.

Although our method also employs explicit integration for MPM elasticity forces, the contact coupling between rigid bodies and material points is as stable as in Zong et al., demonstrating a comparable level of accuracy to the fully implicit treatment. This stability stems from our convex formulation, which solves for contact implicitly. However, due to the weak coupling nature of our approach, our time step size is more limited compared to Zong et al., highlighting a limitation of our method. Despite this, our approach still achieves physically correct results, whereas explicit scheme by Gu et al. fails to produce the correct outcome, even with 10 times smaller time steps and 250 times more substeps.

To further stress test the robustness of our method, we replicate the challenging *close-lift-shake* experiment from [10], where two rigid panels are controlled to grasp a rigid red cube positioned between two elastic cubes modeled with MPM (see Fig. 3). This scenario serves as a stress test with pathological mass ratios using extreme parameters. The mass density of the MPM cube is 100 kg/m^3 , with a Young’s modulus of $5 \times 10^5 \text{ Pa}$ and a Poisson’s ratio of 0.4, while the rigid cube has a mass density of 15000 kg/m^3 . Even in this extreme case, our method successfully completes the task with a time step size of $\Delta t = 0.1$ ms and a moderate tolerance of $\varepsilon_r = 10^{-2}$. Here, we increase the number of substeps to $N = 10$ to maintain stability in the explicit MPM free motion solve, given the high material stiffness relative to its mass density.

B. Rolling an Elastoplastic Dough

We reproduce the complex dough rolling example from [10] with our asynchronous time-splitting scheme, as shown in Fig. 4. The friction coefficient is 1.0 between the dough and the rolling pin. The material parameters and prescribed motion trajectories are identical to those in [10]. We refer readers to the supplemental video for the full task trajectory.

We compare the runtime performance of our method with [10] under various tolerance criteria in Fig. 4. For both methods, we set the absolute tolerance ε_a in Algorithm 1 to machine epsilon and vary the relative tolerance ε_r from 10^{-5} to 10^{-1} . Both methods are simulated with a time step size $\Delta t = 10$ ms, and our method uses $N = 10$ substeps. Across all tolerance levels, our method consistently outperforms [10], achieving at least a 10 \times speed-up and demonstrating strong scalability as the convergence criteria are relaxed, making it well-suited for interactive-rate simulations. With $\varepsilon_r = 10^{-1}$, our method attains a 500 \times speed-up compared



Fig. 3: Our method successfully completes the stress test proposed in [10], which involves pathological mass ratios. The rigid panels securely hold a stack of three boxes with vastly different mass densities in place with friction, maintaining a stable grasp even under vigorous shaking.

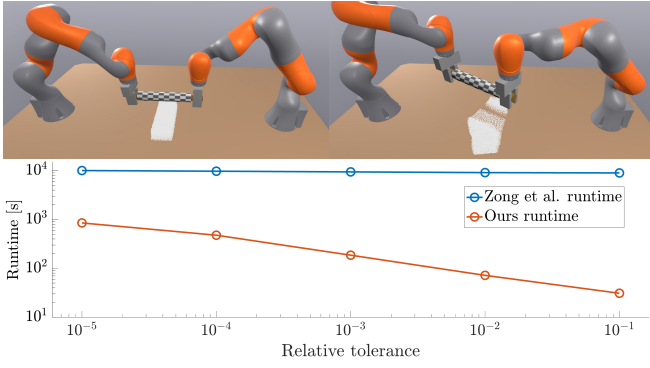


Fig. 4: **(Top)** Flattening a piece of dough with a rolling pin. Our coupling scheme captures the rolling pin’s friction-driven rotation and the dough’s deformation. **(Bottom)** We compare the runtime between ours and Zong et al. [10] under various relative tolerances.

to [10], reaching a 59% real-time rate (defined as simulation time divided by wall-clock time). In practice, we observe that the simulation dynamics are insensitive to the relative tolerance ε_r . We provide a side-by-side comparison of the simulation runs with $\varepsilon_r = 10^{-1}$ and $\varepsilon_r = 10^{-5}$ in the supplementary video. Based on these observations, we generally use a loose relative tolerance of $\varepsilon_r = 5 \times 10^{-2}$ to maintain interactive-rate performance.

C. Cloth and rigid bodies

Another key advantage of our weak coupling scheme is that it eliminates the requirement for a convex energy density in the constitutive model of MPM, as mandated by [10] to maintain convexity in the optimization problem (8). This flexibility unlocks the potential of MPM to model a broader range of materials.

In this experiment, we implement the method from [46] to model cloth with MPM. The frictional contact and self-collision of the cloth are automatically handled by the MPM grid. The simulation setup involves a 42 cm^2 cloth modeled with a Young’s modulus of $E_{\text{cloth}} = 3.2 \times 10^6 \text{ Pa}$, a Poisson’s ratio of $\nu = 0.4$, and a density of $\rho_{\text{cloth}} = 1.5 \times 10^3 \text{ kg/m}^3$. The four corners of the cloth are fixed with boundary conditions to form a cradle. A KUKA LBR iiwa 7 arm,

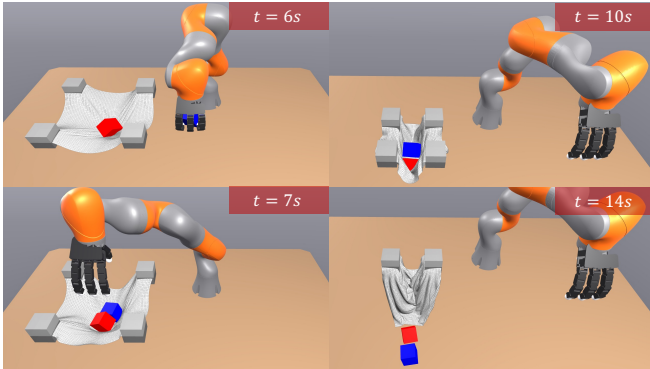


Fig. 5: Pick and place rigid boxes into a deformable cradle.

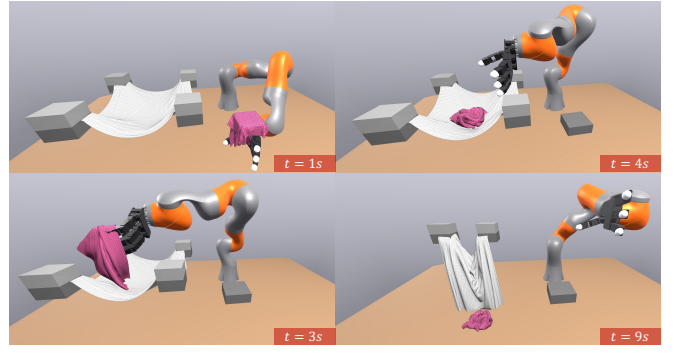


Fig. 6: A robot picks up a piece of cloth and throws it in a “laundry bag”. Our method resolves the complex cloth-on-cloth collisions and cloth self-collisions at interactive rate.

equipped with an anthropomorphic Allegro hand, picks up two rigid boxes and places them on the cloth. Each box has a side length of 8 cm and a density of $\rho_{\text{box}} = 10^3 \text{ kg/m}^3$. The friction coefficient is 0.2 between the boxes and the cloth. The cloth corners are then moved inward to wrap around the rigid boxes, and finally, two corners are released, allowing the boxes to fall out. We refer readers to the supplemental video for the full trajectory.

D. Laundry

Similar to the setup in Section VI-C, we replace the two boxes with another piece of cloth to mimic a common household scenario of moving clothes into a laundry bag. The robot follows a prescribed trajectory to grasp the cloth and throw it into the laundry bag. We then release the grip on two corners of the laundry bag, allowing the cloth to fall out naturally. The simulation achieves a real-time rate of 26.5%, demonstrating that our method efficiently handles intense cloth-on-cloth collisions and self-collision scenarios.

E. Folding and Unfolding a T-Shirt

We demonstrate the accuracy and robustness of our method with a challenging T-shirt folding task (Fig. 1). The T-shirt mesh consists of 4,171 vertices and 7,987 faces, and is simulated with a Young’s modulus of $E_{\text{cloth}} = 10^5 \text{ Pa}$, a Poisson’s ratio of $\nu = 0.3$, and a density of $\rho_{\text{cloth}} = 10^3 \text{ kg/m}^3$. The robot setup features two PD-controlled KUKA LBR iiwa 7 arms equipped with custom parallel grippers. The friction coefficient between the grippers and the cloth is set as 0.6. The simulation sequence proceeds as follows:

- The shirt free falls onto the table ($t = 0.5 \text{ s}$).
- Both grippers perform the first fold along the long edge of the shirt ($t = 3.0 \text{ s}$), followed by a second fold along the short edge ($t = 3.8 \text{ s}$, $t = 4.5 \text{ s}$).
- One gripper grasps a corner of the shirt ($t = 8.3 \text{ s}$), lifts it up vertically ($t = 10 \text{ s}$), and unfolds it ($t = 11 \text{ s}$).
- The grippers then fine-tune the shirt to fully flatten it.

For the full trajectory of this complex task, we refer readers to the supplemental video. Our method accurately captures frictional interactions between the gripper and the T-shirt, enabling smooth task execution. The T-shirt undergoes two

TABLE I: **Simulation statistics.** Runtime is measured as the simulation time in milliseconds per time step. Since the total number of DoFs in the system, n_v , varies as particles move through space, activating different grid nodes, we report the average n_v over the entire simulation. We also provide the average and maximum number of contact points registered throughout the simulation as well as the total particle count. For the cloth simulation, we sample a particle at each mesh vertex and at the centroid of each triangle face, following the approach of [21].

Example	h [m]	# of Particles	n_v	Δt [ms]	N	Runtime [ms]	Contacts Avg. (Max)
Dough Rolling	0.02	5,920	2,229.1	10	10	21.7	17.8 (101)
Box Bagging	1/64	2,582	2,974.7	5	10	45.3	19.6 (45)
Laundry	1/128	14,904	17,296.1	5	5	18.9	163.4 (617)
T-shirt Folding	1/128	12,160	8,299.6	2	4	21.5	533.8 (1,307)
Shake	0.01	3,456	1,576.0	0.1	10	231.1	571.1 (576)

folds, resulting in up to eight stacked layers. Our approach effectively handles cloth self-collisions and rigid-MPM interactions, preventing cloth self-penetration throughout the simulation. This ensures the T-shirt is successfully unfolded without artifacts, showing the robustness of our method.

VII. LIMITATIONS AND FUTURE WORK

Weak coupling: Our weak coupling approach enables broader material choices and efficient GPU parallelization, but it requires smaller time steps when there is a large mass ratio between rigid bodies and MPM bodies, as observed in Section VI-A. Developing a strong coupling formulation that maintains parallelization efficiency for applications like high-frequency vibro-tactile feedback remains an active research direction for the authors.

Explicit integration: As a consequence of explicitly integrating the MPM free motion solve (Eq. (6)), our method requires sufficient substeps to meet the stability criteria of explicit integration for MPM elastic forces. For example, in the stress test scenario described in Section VI-A, we use substeps as small as 0.01 ms to maintain stability. This contrasts with [10], where the scheme is unconditionally stable regardless of the time step size. However, we believe this trade-off of introducing an additional parameter to control substeps is justified by the significant performance gains, enabling interactive-rate simulations.

Linear convergence: As shown in Fig. 4, our quasi-Newton solver exhibits linear convergence. Consequently, the speedup compared to [10] is less pronounced when problem (8) is solved to very tight tolerances, as [10] uses Newton’s method, which achieves quadratic convergence. However, as demonstrated in Section VI-B, the dynamics of our simulation are largely insensitive to the tolerance ε_r , and we generally find that a loose tolerance of $\varepsilon_r = 5 \times 10^{-2}$ is sufficient for most manipulation tasks.

Discrete contact detection: Similar to [11] and [10], we employ discrete collision detection at each substep, where the rigid body’s position is fixed at the beginning of the

time step. This approach introduces the possibility that a rigid body may pass through a thin layer of MPM particles within a single time step without registering a contact. This effect is particularly pronounced in cloth simulations, where penetration can significantly alter subsequent dynamics. To mitigate this issue, we use a smaller time step size of 2 ms in Section VI-E. Developing a more principled solution to this problem, such as incorporating continuous collision detection (CCD), remains an avenue for future work.

APPENDIX

We show that Algorithm 1 converges globally with at least linear rate. This follows immediately from the next lemma.

Lemma 1: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be μ -strongly convex, and consider the quasi-Newton iterations with Hessian approximations $\mathbf{H}(\mathbf{v})$ performed using exact line search from an initial point \mathbf{v}_0 . Define the sublevel set

$$S(\mathbf{v}_0) = \{\mathbf{v} \in \mathbb{R}^n : f(\mathbf{v}) \leq f(\mathbf{v}_0)\}.$$

Assume that for all $\mathbf{v} \in S(\mathbf{v}_0)$ the approximation $\mathbf{H}(\mathbf{v})$ is symmetric positive definite (SPD) with condition number bounded above by σ , and that ∇f is locally Lipschitz with Lipschitz constant L . Then the iterates converge to the unique minimizer \mathbf{v}_* of f , with the error bound

$$f(\mathbf{v}_m) - f(\mathbf{v}_*) \leq \left(1 - \frac{\mu}{\sigma^2 L}\right)^m [f(\mathbf{v}_0) - f(\mathbf{v}_*)] \quad (11)$$

for all iterations $m \geq 0$. Furthermore,

$$\|\mathbf{v}_m - \mathbf{v}_*\| \leq C\rho^m \quad (12)$$

for some $C > 0$ and $0 < \rho < 1$.

Proof: The proof for (11) can be found in Appendix E of [7], where the result is derived using the Polyak inequality. Using μ -strong convexity of f around \mathbf{v}_* , we get

$$\|\mathbf{v}_m - \mathbf{v}_*\|^2 \leq \frac{\mu}{2} [f(\mathbf{v}_m) - f(\mathbf{v}_*)].$$

Taking square roots and applying (11) completes the proof. ■

Theorem 1: Algorithm 1 is globally convergent with at least a linear rate.

Proof: Recall the objective function in problem (8)

$$\ell_p(\mathbf{v}) = \frac{1}{2} \|\mathbf{v} - \mathbf{v}^*\|_{\mathbf{M}}^2 + \ell_c(\mathbf{v}_c).$$

$\nabla \ell_p$ is locally Lipschitz since ℓ_p is continuously differentiable. In addition, Since \mathbf{M} is diagonal and SPD and ℓ_c is convex, the Hessian $\hat{\mathbf{H}}$ of ℓ_p is SPD. Therefore, its block-diagonal approximation \mathbf{H} is also SPD. Thus, by applying Lemma 1, the result follows. ■

REFERENCES

- [1] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, “Foundation models in robotics: Applications, challenges, and the future,” *The International Journal of Robotics Research*, p. 02783649241281508, 2023.
- [2] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve *et al.*, “On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward,” *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, p. e1907856118, 2021.

- [3] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y.-G. Jiang *et al.*, “Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks,” *arXiv preprint arXiv:2412.18194*, 2024.
- [4] R. Tedrake and the Drake Development Team, “Drake: Model-based design and verification for robotics,” <https://drake.mit.edu>, 2019.
- [5] E. Todorov, “MuJoCo,” <http://www.mujoco.org>.
- [6] —, “Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6054–6061.
- [7] A. M. Castro, F. N. Permenter, and X. Han, “An unconstrained convex formulation of compliant contact,” *IEEE Transactions on Robotics*, 2022.
- [8] K. Bayes, M. Bennice, T. Erez, E. Frey, C. K. Fu, N. Gileadi, A. Quaglino, B. Tabanpour, Y. Tassa, and S. Tunyasuvunakool, “Mujoco 3,” 2023. [Online]. Available: <https://github.com/google-deepmind/mujoco>
- [9] X. Han, J. Masterjohn, and A. Castro, “A convex formulation of frictional contact between rigid and deformable bodies,” *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6219–6226, 2023.
- [10] Z. Zong, C. Jiang, and X. Han, “A convex formulation of frictional contact for the material point method and rigid bodies,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 1831–1838.
- [11] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao *et al.*, “Maniskill2: A unified benchmark for generalizable manipulation skills,” *arXiv preprint arXiv:2302.04659*, 2023.
- [12] E. Sifakis and J. Barbic, “FEM simulation of 3d deformable solids: a practitioner’s guide to theory, discretization and model reduction,” in *ACM SIGGRAPH 2012 courses*, 2012, pp. 1–50.
- [13] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, “SOFA: A Multi-Model Framework for Interactive Physical Simulation,” in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321. [Online]. Available: <https://hal.inria.fr/hal-00681539>
- [14] D. Madier, “An introduction to the fundamentals of mesh generation in finite element analysis,” *FEA Academy*, 2023.
- [15] J. F. O’Brien and J. K. Hodgins, “Graphical modeling and animation of brittle fracture,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp. 137–146.
- [16] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, “Incremental potential contact: Intersection- and inversion-free large deformation dynamics,” *ACM Trans. Graph. (SIGGRAPH)*, vol. 39, no. 4, 2020.
- [17] M. Li, D. M. Kaufman, and C. Jiang, “Codimensional incremental potential contact,” *ACM Trans. Graph. (SIGGRAPH)*, vol. 40, no. 4, 2021.
- [18] D. Sulsky, Z. Chen, and H. L. Schreyer, “A particle method for history-dependent materials,” *Computer methods in applied mechanics and engineering*, vol. 118, no. 1-2, pp. 179–196, 1994.
- [19] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, “The material point method for simulating continuum materials,” in *ACM SIGGRAPH 2016 courses*, 2016, pp. 1–52.
- [20] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran, “Drucker-prager elastoplasticity for sand animation,” *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–12, 2016.
- [21] C. Jiang, T. Gast, and J. Teran, “Anisotropic elastoplasticity for cloth, knit and hair frictional contact,” *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.
- [22] X. Han, T. F. Gast, Q. Guo, S. Wang, C. Jiang, and J. Teran, “A hybrid material point method for frictional contact with diverse materials,” *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 1–24, 2019.
- [23] M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, and C. Jiang, “Animating fluid sediment mixture in particle-laden flows,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.
- [24] Y. Fei, Y. Huang, and M. Gao, “Principles towards real-time simulation of material point method on modern gpus,” *arXiv preprint arXiv:2111.00699*, 2021.
- [25] M. Gao, X. Wang, K. Wu, A. Pradhana, E. Sifakis, C. Yuksel, and C. Jiang, “Gpu optimization of material point methods,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–12, 2018.
- [26] X. Wang, M. Li, Y. Fang, X. Zhang, M. Gao, M. Tang, D. M. Kaufman, and C. Jiang, “Hierarchical optimization time integration for cfl-rate mpm stepping,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 3, pp. 1–16, 2020.
- [27] X. Li, M. Li, X. Han, H. Wang, Y. Yang, and C. Jiang, “A dynamic duo of finite elements and material points,” in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.
- [28] Q. Le Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, “Contact models in robotics: a comparative analysis,” *IEEE Transactions on Robotics*, 2024.
- [29] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller, “Small steps in physics simulation,” in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2019, pp. 1–7.
- [30] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2020.
- [31] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and V. Makoviyshuk, “Non-smooth newton methods for deformable multi-body dynamics,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, pp. 1–20, 2019.
- [32] T. Howell, S. Cleac’h, J. Kolter, M. Schwager, and Z. Manchester, “Dojo: A differentiable physics engine for robotics,” *arXiv preprint arXiv:2203.00806*, 2022.
- [33] E. Ménager, L. Montaut, Q. L. Lidec, and J. Carpentier, “Differentiable simulation of soft robots with frictional contacts,” *arXiv preprint arXiv:2501.18956*, 2025.
- [34] I. Huang, Y. Narang, C. Eppner, B. Sundaralingam, M. Macklin, T. Hermans, and D. Fox, “Defgraspsim: Simulation-based grasping of 3d deformable objects,” *arXiv preprint arXiv:2107.05778*, 2021.
- [35] A. Castro, X. Han, and J. Masterjohn, “A theory of irrotational contact fields,” *arXiv preprint arXiv:2312.03908*, 2023.
- [36] R. Elandt, E. Drumwright, M. Sherman, and A. Ruina, “A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 8238–8245.
- [37] J. Masterjohn, D. Guoy, J. Shepherd, and A. Castro, “Velocity level approximation of pressure field contact patches,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 593–11 600, 2022.
- [38] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand, “Taichi: a language for high-performance computation on spatially sparse data structures,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–16, 2019.
- [39] Z. Xian, X. Tong, and T. Liu, “A scalable galerkin multigrid method for real-time simulation of deformable objects,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–13, 2019.
- [40] X. Wang, Y. Qiu, S. R. Slattey, Y. Fang, M. Li, S.-C. Zhu, Y. Zhu, M. Tang, D. Manocha, and C. Jiang, “A massively parallel and scalable multi-gpu material point method,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 30–1, 2020.
- [41] L. Fang, R. Zhang, C. Vanden Heuvel, R. Serban, and D. Negrut, “Chrono:: Gpu: An open-source simulation package for granular dynamics using the discrete element method,” *Processes*, vol. 9, no. 10, p. 1813, 2021.
- [42] V. Makoviyshuk, 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.
- [43] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-k. Chan *et al.*, “Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai,” *arXiv preprint arXiv:2410.00425*, 2024.
- [44] G. Authors, “Genesis: A universal and generative physics engine for robotics and beyond,” December 2024. [Online]. Available: <https://github.com/Genesis-Embodied-AI/Genesis>
- [45] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang, “A moving least squares material point method with displacement discontinuity and two-way rigid body coupling,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.
- [46] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, “The affine particle-in-cell method,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–10, 2015.