SoftMAC: Differentiable **Soft** Body Simulation with Forecast-based Contact **M**odel and Two-way Coupling with **A**rticulated Rigid Bodies and **C**lothes

Min Liu[1], Gang Yang[2], Siyuan Luo[2], and Lin Shao[2]

***Abstract*— Differentiable physics simulation provides an avenue to tackle previously intractable challenges through gradient-based optimization, thereby greatly improving the efficiency of solving robotics-related problems. To apply differentiable simulation in diverse robotic manipulation scenarios, a key challenge is to integrate various materials in a unified framework. We present SoftMAC, a differentiable simulation framework that couples soft bodies with articulated rigid bodies and clothes. SoftMAC simulates soft bodies with the continuum-mechanics-based Material Point Method (MPM). We provide a novel forecast-based contact model for MPM, which effectively reduces penetration without introducing other artifacts like unnatural rebound. To couple MPM particles with deformable and non-volumetric clothes meshes, we also propose a penetration tracing algorithm that reconstructs the signed distance field in local area. Diverging from previous works, SoftMAC simulates the complete dynamics of each modality and incorporates them into a cohesive system with an explicit and differentiable coupling mechanism. The feature empowers SoftMAC to handle a broader spectrum of interactions, such as soft bodies serving as manipulators and engaging with underactuated systems. We conducted comprehensive experiments to validate the effectiveness and accuracy of the proposed differentiable pipeline in downstream robotic manipulation applications. Supplementary materials are available on our project website at https://damianliumin.github.io/SoftMAC/.**

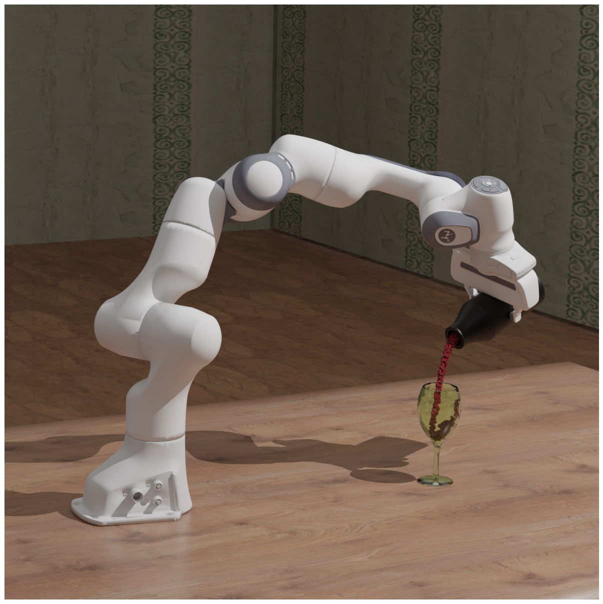


Fig. 1: To pour water into a glass in differentiable physics simulation, we need realistic contact model for soft-rigid coupling and correct gradient calculation. Then same applies to the interactions between soft bodies and clothes.

I. INTRODUCTION

Interactions between diverse materials are prevalent in the realm of robotic manipulation. For instance, the collision between glass and wine during a pouring task as in fig. 1, or the perpetual interplay between a tortilla and its fillings when crafting a taco. As a number of differentiable physics engines emerge to tackle learning and control problems [1], [2], [3], [4], [5], [6], it is tempting to unify different materials in a single simulator, thus supporting a wide range of manipulation tasks. Nevertheless, developing a general-purpose physics simulator is non-trivial, as different types of materials often need different physical models due to the dominant physical behaviors [7].

Depending on their dynamical behaviors, most daily objects can be simulated as: 1) soft bodies, like elastic, plastic, elasto-plastic objects and liquid; 2) rigid bodies, either articulated or not; 3) clothes, as well as other thin-shell objects with similar geometric and physical properties (*e.g.*, tortilla). The interactions between these three modalities encompass a wide spectrum of manipulation tasks. Rigid-clothes coupling has been extensively discussed in previous works [8], [9], [10]. Meanwhile, differentiable soft-cloth coupling remains a rarely explored topic in current literature, despite its prevalence in robotic manipulation scenarios such as making taco. A line of works propose methods for soft-rigid coupling. PlasticineLab [11], FluidLab [4] and DexDeform [12] deploys Material Point Method (MPM) to simulate soft bodies, but they only simulate the kinematics of rigid bodies, implying that force exerted on soft bodies cannot react on rigid bodies. Maniskill2 [13] introduces two-way dynamics coupling, but its pipeline is not differentiable. Besides, contact models for MPM in these works suffer from artifacts like penetration and unnatural rebound.

In this paper, we present a differentiable pipeline to couple soft bodies with articulated rigid bodies and clothes. Following previous works, we adopt MPM for soft bodies due to its ability to simulate a large variety of deformable materials and physical phenomena (e.g., Magnus effect and buoyancy) [14]. MPM struggles with delicate boundaries. Scaling down grid size and time step alleviates the problem, but is not always feasible in robotic simulation where computational efficiency should be balanced. Gu *et al.* [13] apply particle

[1]Min Liu is with School of Computer Science, Carnegie Mellon University, USA. `minliu2@cs.cmu.edu`

[2]Gang Yang, Siyuan Luo and Lin Shao are with Department of Computer Science, National University of Singapore, Singapore. `yg.matinal@gmail.com`, `luosiyuan2002@gmail.com` and `linshao@nus.edu.sg`

Simulator	**Differentiable**	**MPM**			**Rigid**		**Cloth**	**Coupling**		
		Elastic	**Plastic**	**Liquid**	**Dynamics**	**Articulated**		**Contact**	**MPM-Rigid**	**MPM-Cloth**
PlasticineLab [11]	✓		✓					Grid	One-way	\
FluidLab [14]	✓	✓	✓	✓	*			Grid/Particle	One-way	\
DexDeform [12]	✓		✓			†		Grid	One-way	\
ManiSkill2 [13]		✓	✓	✓	✓	✓		Grid/Particle	Two-way	\
SoftMAC (Ours)	✓	✓	✓	✓	✓	✓	✓	Forecast	Two-way	Two-way

TABLE I: Comparison with other popular MPM-based simulators. *FluidLab supports modeling rigid bodies using kinematic skeletons or MPM. While the latter can simulate dynamics of rigid bodies, it is limited to simple shapes without articulation, which is not considered as general-purpose rigid body simulation. †DexDeform provides a simulator tailored for deformable object manipulation with the Shadow hand. The specialization limits its direct applicability to other articulated rigid bodies.

forces to reduce penetration, but the method introduces problems such as unnatural rebound. To achieve realistic collision effects, we introduce a novel method called *forecast-based contact model* to manage the boundary conditions for MPM. Specifically, the model takes a grid-to-particle transfer to look ahead in the grid operation stage, imposes constraints on particles within the contact region, and then adjusts the grid velocity accordingly. Forecast-based contact model requires signed distance fields (SDFs) to penalize penetration. While the definition of SDF is straightforward for volumetric objects, it is hard to determine the sign on nonvolumetric meshes. To solve the problem, we propose a *penetration tracing* algorithm that capitalizes on the localized motion of particles to reconstruct the SDF within confined zones. In this way, the contact model can be applied to both soft-rigid and soft-cloth coupling.

Previous differentiable MPM simulators [4], [11], [12] model external materials as kinematic skeletons, without explicitly considering forces, mass distribution, or other dynamic factors. While it simplifies the pipeline, the simulators cannot generalize to scenarios where dynamics of both materials is involved (e.g., underactuated systems). We overcome the problem by utilizing independent dynamics simulation for each modality and incorporating them into a cohesive system. As in [13], [15], we handle the contact between MPM and other materials explicitly, but make the whole process differentiable. Specifically, we copy motion of rigid bodies and clothes to the soft body simulator, and transfer force back in forward simulation. Then we apply chain rule in reverse direction along the computation graph to propagate gradients between different simulators. The gradient information allows us to control the motion between different modalities by optimizing the force acting on any point in the simulation. Code of the proposed simulator, which we call SoftMAC, is released at our project website.

Our main contributions can be summarized as follows:

- We propose a novel forecast-based contact model for MPM, which reduces penetration without introducing artifacts like unnatural rebound.
- We present a penetration tracing algorithm for the contact between MPM and non-volumetric cloth meshes.
- To the best of our knowledge, SoftMAC is the first differentiable robotic simulator to support two-way dynamics soft-rigid and soft-cloth coupling.

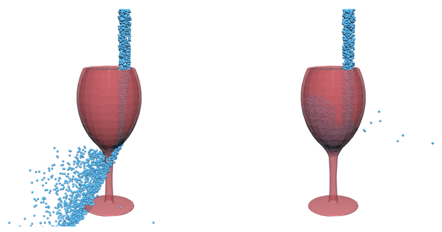
II. Related Work

A. Differentiable Physics-based Simulation

Physics simulation provides an avenue to avoid real-world damages, accelerate robotics-related problem solving, and establishes standardized benchmarks to evaluate different algorithms. A branch of works provide simulation for articulated rigid bodies [16], [17], [18], [19], which is fundamental in robotics. With advances in deformable object manipulation, an increasing number of physics engines are also proposed for clothes and soft bodies [20], [21]. Recent developments in automatic differentiation methods [22], [23], [24] boost the prosperity in differentiable physics simulators, which can convert challenging control tasks into gradient-based optimization problems. Several differentiable simulators are developed for rigid bodies [1], [2], clothes [6], [10], [25], thin shells [26], and soft bodies [27], [28]. A line of works [3], [4], [11] simulate soft bodies with material point method (MPM) to support various materials and physical processes. We also deploy MPM for soft bodies in this work.

B. Coupling Soft Bodies with Other Modalities

Unifying soft bodies with other modalities in a single framework requires the design of contact mechanisms between different materials. A number of methods have been proposed in robotics and computer graphics area. Du *et al.* [14] and Harada *et al.* [29] provide simulation methods to couple cloth and fluid computed by using smoothed particle hydrodynamics. MuJoCo [16] and Bullet [17] enable manipulating soft bodies simulated with finite element method. However, these methods only support a limited number of soft body materials. Recent works based on MPM provide the opportunity to couple various types of soft bodies with other modalities. PlasticineLab [11] and FluidLab [14] provide interaction with rigid bodies by penalizing grid velocity directly, but their contact models are one-way. Maniskill2 [13] transfers force back to rigid bodies to enable bidirectional contact, and penalize particles in their contact models to alleviate penetration. Nevertheless, their pipeline does not provide gradient information. Our work is based on a novel contact model better at reducing penetration. It also provides bidirectional contact for both soft-rigid and soft-cloth coupling, and make the process differentiable. We present a comparison in terms of differentiability, material types,


(a) Contact Models

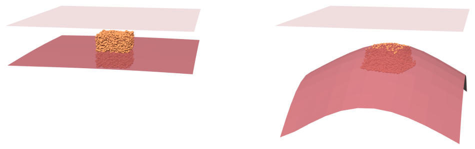

(b) Penetration Tracing

Fig. 2: *(a)* Pour water into a thin glass. Grid-based model (left) leads to severe penetration. Particle-based model (middle) also causes a few particles to penetrate the glass. Forecast-based model (right) achieves the most robust performance. *(b)* Drag four corners of a towel to squash a plasticine. Towel goes through the plasticine without penetration tracing (left). Both the plasticine and towel deform due to the contact after adding the algorithm (right).

and coupling methods with other MPM-based simulators in table I. SoftMAC covers a wider range of materials and their interactions, thereby facilitating a more extensive array of robotic manipulation scenarios.

C. Robotic Manipulation with Differentiable Simulation

Differentiable physics simulation has been applied in several robotic manipulation methods [26], [30], [31], [32]. DexDeform [12] trains a skill model from human demonstrations, and uses a gradient optimizer to refine the trajectories planned by the skill model to generate more demonstrations. SAGCI [33] proposes a model-based learning method with differentiable simulation to online verify and modify the environment model during interaction. SAM-RL [34] combines differentiable physics simulation and rendering to propose a sensing-aware learning pipeline that selects an informative viewpoint to monitor the manipulation process.

III. Contact Models

A. Preliminary

We build our soft body simulator based on MLS-MPM [35]. For simplicity, we use *MPM* for *MLS-MPM* in this section. In a simulation loop, the matter is interpolated back and forth between particle and grid representations. Consider a system that consists of n_p particles and n_g grid nodes. Denote $x_p, v_p, m_p \in \mathbb{R}^{3n_p}$ as positions, velocities and masses of particles, and $v_g, p_g, m_g \in \mathbb{R}^{3n_g}$ as velocities, momentum and masses of grid nodes. We duplicate the masses to higher dimension for the convenience of vector operations. $W: \mathbb{R}^{3n_p} \to \mathbb{R}^{3n_g \times 3n_p}$ calculates a sparse matrix that contains weights for the 3^3 neighbouring grid nodes surrounding each particle. Superscript denotes the time step of a variable.

a) Particle-to-Grid (P2G): MPM first computes stress based on the constitutive model, and then transfer particles' momentum and masses to background grid using the interpolation matrix $W(x_p^n)$:

$$\begin{aligned} p_g^{n+1} &= W(x_p^n)(m_p^\top v_p^n + p_d^n), \\ m_g^{n+1} &= W(x_p^n) m_p^n. \end{aligned} \tag{1}$$

where p_d^n is a momentum term that reflects internal forces.

b) Grid Operation: The discrete equations of momentum are solved on grid nodes. Boundary conditions $BC(\cdot)$ can be enforced at this stage to set constraints on grid velocity:

$$\begin{aligned} \hat{v}_g^{n+1} &= (p_g^{n+1})^\top (1/m_g^{n+1}), \\ v_g^{n+1} &= BC(\hat{v}_g^{n+1}). \end{aligned} \tag{2}$$

c) Grid-to-Particle (G2P): The final stage transfers velocity back to particles, updates deformation information, and advects particles based on the new velocities.

$$\begin{aligned} v_p^{n+1} &= W(x_p^n)^\top v_g^{n+1}, \\ x_p^{n+1} &= x_p^n + v_p^{n+1}\Delta t. \end{aligned} \tag{3}$$

Contact between MPM and other modalities is typically handled on the background Eulerian grid. *Grid-based contact model* directly zeros out the tangent component of grid velocity through $BC(\cdot)$ in eq. (2). Reaction force can be computed via momentum change $F = \Delta p / \Delta t$. However, the grid nature makes it challenging to accurately represent and handle intricate geometries and moving boundaries. The method leads to severe penetration when coupling with delicate mesh-based objects. Although scaling up grid resolution can alleviate the problem, it is typically not feasible in robotic simulation due to efficiency constraints.

Gu *et al.* [13] find that *particle-based contact model* reduces penetration. Specifically, the method approximates the contact surface as a spring: as a particle crosses the contact boundary with distance $d > 0$, a force $F = -kd$ is exerted to push it back. Essentially, the penalty force also works as an indirect constraint for grid velocity by adding a momentum term $F\Delta t$ to eq. (1), but provides more fine-grained control. Particle-based contact model brings a problem on how to choose k. With a small k, penetration cannot be effectively alleviated. On the other hand, increasing k will lead particles to rebound unnaturally near the boundary. Alleviating one of the artifacts makes the other worse.

B. Forecast-based Contact Model

We formulate the goal as reducing the number of penetrations at the end of each simulation loop. Since particle positions x_p^{n+1} are advected based on velocities, we focus on how to set constraints on grid velocities to obtain v_p^{n+1} that can achieve the goal. The idea of looking ahead and then

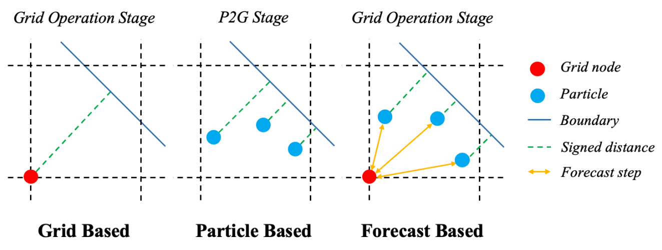


Fig. 3: Illustration of the contact models. Grid-based model directly computes SDF on grid nodes. Particle-based model computes SDF on particles and applies penalty on them. Forecast-based model takes a forecast step to compute SDF on particles and then adjust grid velocity accordingly.

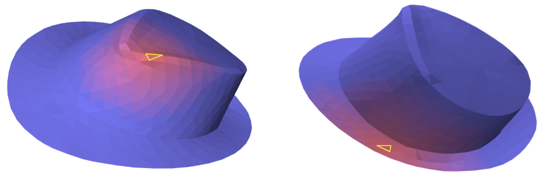


Fig. 4: Given a target face (yellow edge), we search for the triangles in its neighboring area (red). If a particle is still in contact after system state changes, its nearest face should be within the neighboring area due to the small time interval.

adjusting grid velocity gives our method the name *forecast-based contact model*. It also provides an intuition why the method has better performance in reducing penetration.

We use W as short for $W(x_p^n)$ from this section. Our method first takes a G2P transfer $v_{init}^{n+1} = W\hat{v}_g^{n+1}$. Then we apply a boundary condition $BC_p(\cdot)$ on the particle velocities, which computes constrained velocities v_{tgt}^{n+1} that are supposed to avoid penetration. Given v_{tgt}^{n+1}, we transform our goal into an optimization problem:

$$v_g^{n+1} = \arg\min_{v_g} \|W^\top v_g - v_{tgt}^{n+1}\|^2. \tag{4}$$

The problem above can be tackled using iterative solvers like conjugate gradient method, whose computation overhead grows linearly with the number of iterations. However, we find that one-step gradient descent suffices in practice:

$$v_g^{n+1} = \hat{v}_g^{n+1} - \alpha W(W^\top v_g - v_{tgt}^{n+1}), \tag{5}$$

where α is a predefined step size. The extra computation cost is equivalent to interpolating a particle property to grid and transferring it back. Besides, the derivatives of eq. (5) is simple enough to be directly obtained through automatic differentiation methods.

Next, we introduce the details of $BC_p(\cdot)$. Given a particle with velocity v_{in}, we first check whether it is within the contact region by comparing signed distance d with a threshold $\hat{d}$. If $d < \hat{d}$, the contact point is approximated with the nearest point on the boundary, whose velocity is v_c. For rigid bodies, v_c is the linear velocity at the contact point. For clothes, we find three vertices on the Lagrangian mesh that make up the contact surface, and compute weighted average of their velocities based on the barycentric coordinate of the contact point. We then decompose the relative velocity $v_{rel} = v_{in} - v_c$ into a normal component v_n and a tangential component v_t. To penalize the velocity, we drop v_n and decay v_t with friction:

$$v_{out} = v_t \cdot \max\left(0, 1 - \mu\|v_n\|/\|v_t\|\right) + v_c, \tag{6}$$

where μ is the friction coefficient. We also utilize two other techniques: *1)* Blend the original and modified velocities with a smoothness factor $s = \min\{\exp(-\beta d), 1\}$, given by $v'_{out} = s v_{out} + (1-s)v_{in}$. β is a predefined smoothness factor. The method reduces drastic state changes and improves gradient quality. *2)* Advect the particle with v_{out}, and if penetration happens, add a component to v_{out} to ensure that the particles are moved to the nearest legal position. Given velocity change, we can obtain impulse exerted on the particle, thereby computing the reaction force F. The spatial force is accumulated on each rigid body link. For clothes, we distribute F to three nodes of the nearest triangle based on the barycentric coordinate of the contact point.

C. *Comparisons between Contact Models*

As shown in fig. 3, both grid-based and forecast-based contact models handle boundaries when solving grid velocities. The difference lies in that grid-based model computes SDF function on the static MPM grid nodes, while our method computes SDF function on Lagrangian particles that move with the material, allowing for more precise interaction with complex boundaries. Particle-based contact model also computes SDF on particles, and applies a penalty term during P2G transfer. However, the method is agnostic about how the penalty influences future particle states. In contrast, our method takes a *forecast* step to see whether the particles will penetrate the boundary and utilizes the information to direct the update of grid velocity.

D. *Penetration Tracing for Soft-Cloth Coupling*

The contact models require signed distance to couple MPM soft bodies with objects represented as meshes. For rigid bodies, we pre-compute the signed distance field on compact grid points within the bounding boxes, and look up signed distance by interpolating the grid values during simulation. However, the method is not applicable to soft-cloth coupling because meshes of clothes varies by time. Besides, such meshes are non-volumetric, which means that both sides can be considered as the outward surface. While it is feasible to manually define a positive side for some simple shapes like a square, the idea cannot generalize to special geometries such as a Möbius strip.

To fix the problem, we reconstruct SDF for cloth meshes by computing distance and sign separately. We first search the particle-face pairs to find the nearest face for each particle, and compute *absolute distances* accordingly. The process can be accelerated with spatial hashing. The *sign* indicates whether a particle penetrated the mesh. Therefore, we assign a binary penetration state z^n for each particle and trace it throughout the simulation process. Each time the position of particles or meshes change, we check whether

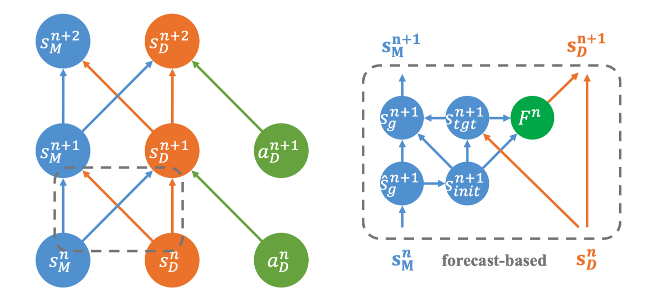


Fig. 5: Computation graph for soft-rigid coupled system (the same in soft-cloth coupling). *Blue nodes*: state of MPM simulation. *Orange nodes*: state of articulated rigid body simulation. *Green nodes*: actions and external forces. A line is connected between two nodes if and only if we directly use one state or action to compute the resulting state. The gray box illustrates the details of forecast-based contact model.

the state change will lead to penetration. If a particle moves to the other side of the mesh, we update the penetration state at the next time step, given by $z^{n+1} = 1 - z^n$. Otherwise, z^{n+1} inherits the value of z^n.

The complication lies in how to check whether penetration happens. We provide a method by utilizing the locality of motion in simulation. The small time interval in physics simulation suggests that system state only changes a bit at each time. If a particle is in contact with face i at time n and face j at time $n+1$, then we assume that faces i and j are within the neighboring area of each other. Based on this idea, we pre-compute the neighboring triangles around each face using breadth first search, and define a consistent orientation for faces in this local area. Then we can directly check whether a particle resides on the same side of the area during simulation. An illustration can be found in fig. 4. We discuss limitations of the algorithm in section VI.

IV. Differentiable Simulation Design

A. Forward Simulation

Denote s_M^n as state of MPM object at time n. The forward simulation for MPM is defined as $\mathscr{F}_M(s_M^n)$. s_D^n and a_D^n represent the state and action of the manipulator (articulated rigid bodies or clothes). The forward simulation for manipulator is $\mathscr{F}_D(s_D^n, a_D^n)$. While both simulations can be conducted independently, interactions between the two modalities cannot be directly simulated. Therefore, we provide explicit coupling between these simulators as in [13], [15].

Specifically, we modify the simulations as follows. The MPM simulator takes poses and velocities of the manipulators as inputs. Based on the contact models, it updates MPM state and computes the reaction force on the manipulator. Manipulator simulation takes external force as an additional input, which is involved in the computation of the next state:

$$\begin{aligned} s_M^{n+1}, F^{n+1} &= \mathscr{F}'_M(s_M^n, s_D^n), \\ s_D^{n+1} &= \mathscr{F}'_D(s_D^n, a_D^n, F^{n+1}). \end{aligned} \tag{7}$$

In a simulation loop, we execute $\mathscr{F}'_M$ first, with the manipulators as boundary conditions. Then we transfer the force back and execute $\mathscr{F}'_D$. The computation graph is displayed in fig. 5. We define state of the coupled system as $s^n = (s_M^n, s_D^n)$, and action as $a^n = a_D^n$. The forward simulation functions as:

$$s^{n+1} = (s_M^{n+1}, s_D^{n+1}) = \mathscr{F}(s^n, a^n). \tag{8}$$

B. Backward Gradients

The differentiable operations inside MPM simulation $\mathscr{F}'_M$ help us gather the Jacobian matrices $\frac{\partial s_M^{n+1}}{\partial s_M^n}$, $\frac{\partial s_M^{n+1}}{\partial s_D^n}$, $\frac{\partial F^{n+1}}{\partial s_M^n}$, $\frac{\partial F^{n+1}}{\partial s_D^n}$. The manipulator simulation $\mathscr{F}'_D$ computes $\frac{\partial s_D^{n+1}}{\partial s_D^n}$, $\frac{\partial s_D^{n+1}}{\partial a^n}$, $\frac{\partial s_D^{n+1}}{\partial F^{n+1}}$. Based on these Jacobian matrices, we apply the chain rule to obtain gradients for every system state at each time step.

s_M^n is directly involved in the computation of s_M^{n+1}, and affects s_D^{n+1} through the contact force F^{n+1}. Consequently, the gradient of s_M^n has two components:

$$\frac{\partial \mathscr{L}}{\partial s_M^n} = \frac{\partial \mathscr{L}}{\partial s_M^{n+1}} \frac{\partial s_M^{n+1}}{\partial s_M^n} + \frac{\partial \mathscr{L}}{\partial s_D^{n+1}} \frac{\partial s_D^{n+1}}{\partial F^{n+1}} \frac{\partial F^{n+1}}{\partial s_M^n}. \tag{9}$$

s_D^n influences s_D^{n+1} in two ways. In addition to the one inherent in $\mathscr{F}_D(s_D^n, a_D^n)$, s_D^n affects the contact force F^n between MPM and the manipulator, and the force in turn impacts on s_D^{n+1}. Another part of the gradient comes directly from s_M^{n+1}, which takes s_D^n as the boundary:

$$\frac{\partial \mathscr{L}}{\partial s_D^n} = \frac{\partial \mathscr{L}}{\partial s_D^{n+1}} \left(\frac{\partial s_D^{n+1}}{\partial s_D^n} + \frac{\partial s_D^{n+1}}{\partial F^{n+1}} \frac{\partial F^{n+1}}{\partial s_D^n} \right) + \frac{\partial \mathscr{L}}{\partial s_M^{n+1}} \frac{\partial s_M^{n+1}}{\partial s_D^n}. \tag{10}$$

Finally, a_D^n can be directly computed from $\frac{\partial \mathscr{L}}{\partial s_D^{n+1}}$ inside the manipulator simulation:

$$\frac{\partial \mathscr{L}}{\partial a_D^n} = \frac{\partial \mathscr{L}}{\partial s_D^{n+1}} \frac{\partial s_D^{n+1}}{\partial a_D^n}. \tag{11}$$

It is worth noting that $\frac{\partial s_M^{n+2}}{\partial a_D^n} = \frac{\partial s_M^{n+2}}{\partial s_D^{n+1}} \frac{\partial s_D^{n+1}}{\partial a_D^n}$. The gradients propagated from s_M^{n+2} to a_D^n explain why we can define losses on soft bodies to optimize the actions on the manipulators.

V. Experiments

In this section, we introduce our system implementation, and conduct quantitative and qualitative evaluations to answer the following questions: 1) Does forecast-based contact model produce more realistic simulation results? 2) Can penetration tracing effectively support soft-cloth coupling? 3) How accurate is the gradient information provided by the proposed two-way differentiable dynamics coupling? In all the experiments, we maintain a CFL number below the critical threshold of 0.3 to ensure that the time integration scheme remains stable.

A. System Implementation

We implement a differentiable soft body simulator using DiffTaichi [24]. Corotated constitutive models are used to simulate fluid, elastic and plastic materials. We refer readers to FluidLab [14] for more details about simulating different materials using MPM. We adopt two differentiable physics

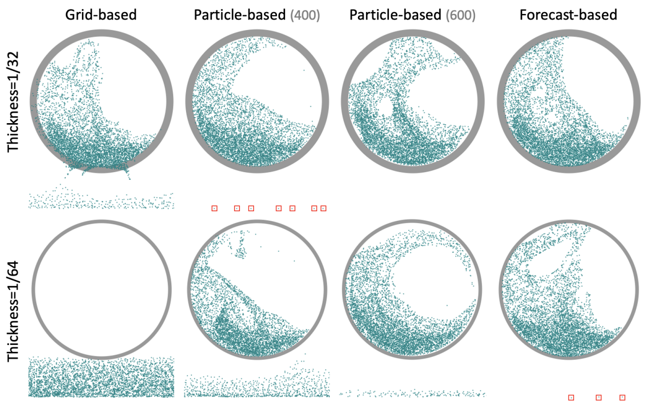


Fig. 6: Shake a container with liquid. Grid-based model causes severe penetration. Particle-based model also faces the problem if the penalty coefficient k is not high enough. Increasing k alleviates penetration, but makes particles rebound more unnaturally near boundary. Forecast-based model results in least penetrations without bringing obvious artifacts.

engines, Jade [36] and DiffClothAI [10], for the simulation of articulated rigid bodies and clothes, respectively. In each simulation loop of the coupled system, we take actions, manipulator states, and MPM states as input, calculate contact in the soft body simulator, and transfer external force to the rigid body / clothes simulator. MPM typically requires a smaller time interval than the other two simulators. In this case, we take several MPM substeps and average the force on the manipulator. Gradients are calculated accordingly.

B. Checking Forecast-based Contact Model

We first conduct a quantitative experiment to compare forecast-based contact model with the baselines (grid-based and particle-based models). Specifically, we initialize the 2D scene with fluid inside a circular rigid container, which has radius r and thickness d. The container shakes left and right at a constant speed, and we record the number of particles penetrating the container.

We choose step size $\alpha = 0.2$ for the one-step gradient descent in forecast-based contact model, and the objective in eq. (4) decreases by 83.1% on average. As shown in table II, forecast-based model achieves the best performance in reducing penetration. The computation efficiency lags behind baselines as we introduce additional P2G and G2P transfers. Another empirical observation is that while increasing the penalty coefficient k in particle-based contact model can alleviate penetration, it also exacerbates unnatural rebound near the boundary. Meanwhile, our method does not have this problem. Visualization for the last frame is provided in fig. 6. We also conduct a qualitative experiment: pour liquid into a thin glass with high velocity. Results shown in fig. 2a are consistent with our quantitative experiment.

C. Checking Penetration Tracing

We drag the four corners of a towel to squash a plasticine. The control points are moved down from their initial positions at a constant speed. We add and remove the penetration tracing algorithm to demonstrate the effectiveness of our method in soft-cloth coupling. The results are displayed in fig. 2b. We find that lacking the penetration information causes the towel to go though the plasticine directly. However, with penetration tracing, the plasticine is squashed and the towel deforms due to the reaction force. We conduct the comparison on particle-based contact model. With forecast-based model, the towel cannot pass the plasticine even after removing the tracing algorithm in this case.

Contact Model	**Thickness**	**#Penetration↓**	**Unnatural Rebound**	**Time (s)**
Grid	1/32	449	✗	4.90
	1/64	5998	✗	4.81
Particle (k=400)	1/32	12	✗	4.45
	1/64	1512	✗	4.71
Particle (k=600)	1/32	0	✓	4.33
	1/64	217	✓	4.68
Forecast (Ours)	1/32	0	✗	5.24
	1/64	3	✗	5.15

TABLE II: Experiment results of shaking containers of different thickness d with liquid for 1100 frames. #Penetration denotes the number of particles outside the container in the final frame. Unnatural rebound occurrences are recorded. Simulation time is compared on an NVIDIA Tesla T4 GPU.

D. Checking Two-way Differentiable Dynamics Coupling

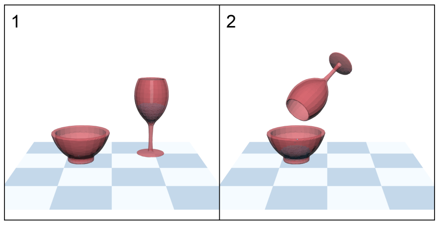
We check the quality of two-way differentiable dynamics coupling under 6 robotic manipulation tasks. The first three tasks can also be accomplished within current simulations that employ kinematic skeletons as controllers. However, *pull door*, *make taco*, and *push towel* necessitates the utilization of SoftMAC as the dynamics of both modalities must be incorporated to successfully complete these tasks.

We formulate our problem as trajectory optimization: Given an action sequence $a = (a_1, a_2, \cdots, a_T)$ and the resulting state sequence $s = (s_1, s_2, \cdots, s_T)$, we define an objective function $f(a,s)$ and find the optimal actions through

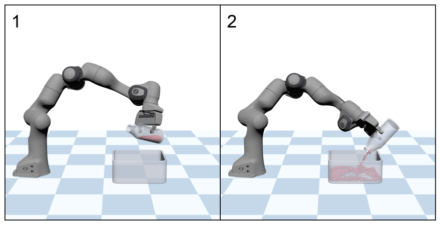
$$a^* = \arg\min_a f(a,s). \tag{12}$$

We use the gradients provided by SoftMAC and first-order optimizers (SGD for *pouring water (franka)* and Adam [37] for other tasks) to solve eq. (12). The number of variables to optimize ranges from 200 to 1150. It is worth noting that for more complicated tasks, the trajectories are prone to get stuck in local optima. Initialization from teleportation or learned policies helps overcome the problem [12], but is beyond the scope of our experiments.

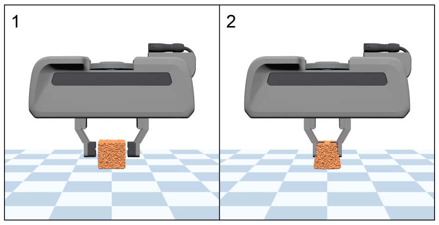
a) Rigid2MPM: We evaluate soft-rigid coupling on three tasks. In *pour water*, we conduct 6 DoFs control over a glass to pour liquid into a bowl. In *pour water with Franka*, we conduct 7 DoFs control over the Franka arm to pour liquid from a bottle into a tank. In *squeeze plasticine*, we conduct 2 DoFs control over a gripper to squeeze the 16200 DoFs plasticine into target shape. Losses are computed as the Chamfer distance between current and target positions of the



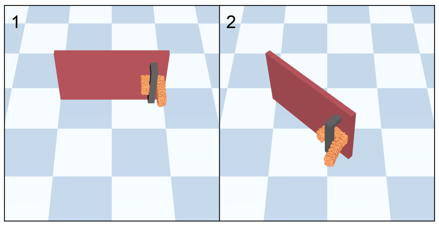
(a) Pour water



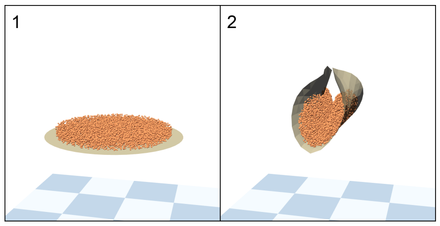
(b) Pour water (Franka)



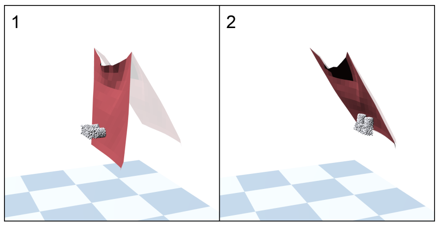
(c) Squeeze plasticine



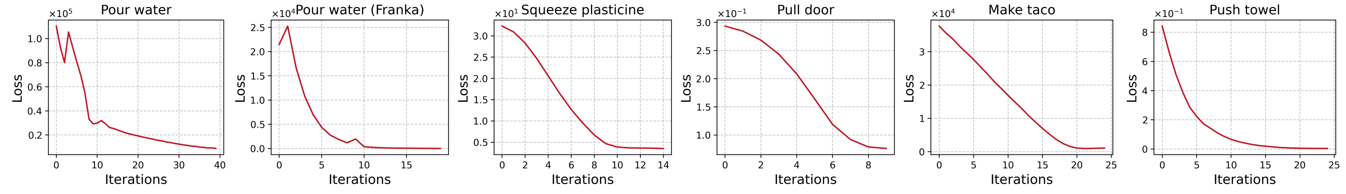
(d) Pull door



(e) Make taco



(f) Push towel



(g) Loss curves

Fig. 7: Experiment results of coupled differentiable simulation. *(a)* Control glass to pour the liquid into a bowl. *(b)* Control Franka arm to pour the liquid into a tank from a bottle. *(c)* Control Franka gripper to squeeze the plasticine into target shape. *(d)* Control 2 selected points on the tortilla to fold the taco into target shape. *(e)* Control soft gripper to pull the door into target angle. *(f)* Control soft gripper to push the towel into target pose. In all tasks, their respective training loss curves are plotted on the right. Full trajectories are displayed in our video.

particles. In the pouring tasks, two additional loss terms are calculated on poses and velocities of the source container to penalize it from colliding with the target container.

b) Cloth2MPM: We verify soft-cloth coupling with *make taco*. The tortilla and taco fillings are simulated as clothes and MPM particles, respectively. We conduct 4 DoFs control over two selected points of the tortilla and fold the taco into target shape. The state space is combination of the 30000 DoFs taco fillings and the 651 DoFs tortilla. Loss is computed on taco fillings as the Chamfer distance between current and target configurations. To avoid overly stretching the tortilla, we control the particles in the perpendicular plane through circle center and project the control points back to a confined region after each epoch.

c) MPM2Rigid/Cloth: We check two-way differentiable coupling by controlling soft grippers to manipulate articulated rigid bodies or clothes. We control MPM by exerting an impulse on selected particles in P2G transfer. In *pull door*, we control the elastic gripper to pull the door into target angle. In *push towel*, we control the elastic gripper to push the 432 DoFs towel into target pose. We computed losses on the state of rigid bodies / clothes and optimize the control over soft grippers.

The pour water and make taco tasks are optimized for 40 and 25 iterations respectively, costing 21.67s and 20.45s for each iteration on average on an NVIDIA Tesla T4 GPU. Experiment results reported in fig. 7 demonstrate the correctness and effectiveness of our differentiable simulation.

VI. Conclusion and Future Work

In this work, we introduce SoftMAC, a differentiable simulation that couples soft bodies with articulated rigid bodies and clothes. We simulate soft bodies with material point method, and provide a novel contact model that reduces artifacts like penetration and unnatural rebound. To couple soft body particles with deformable and non-volumetric clothes meshes, we propose a method to reconstruct signed distance field in local area. We also utilize a two-way differentiable dynamics coupling mechanism to unify the simulation of different materials. Comprehensive experiments are conducted in robotic manipulation scenarios and demonstrate the correctness and effectiveness of the differentiable simulation.

One limitation of this work lies in the assumption that cloth meshes are manifold and not pressed together for penetration tracing. The assumption is grounded in the underlying cloth simulator [10] that uses Incremental Potential Contact [38] to guarantee a gap between meshes. However, addressing penetration tracing challenges in complex fabric arrangements remains an open problem and requires further exploration for a comprehensive solution. Another drawback is rooted in computational efficiency. The MPM simulator in SoftMAC is developed using Taichi to support massive parallelism on GPUs. However, the rigid body and cloth simulators run on CPUs. The rate of state and gradient transfer between CPUs and GPUs is restricted by bandwidth and impacts overall performance. We leave the development of GPU-based differentiable rigid body and cloth simulators with necessary interfaces as future work.

References

[1] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," *arXiv preprint arXiv:2103.16021*, 2021.

[2] 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*, vol. 9, 2022.

[3] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6265–6271.

[4] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan, "Fluidlab: A differentiable environment for benchmarking complex fluid manipulation," *arXiv preprint arXiv:2303.02346*, 2023.

[5] J. Liang, M. Lin, and V. Koltun, "Differentiable cloth simulation for inverse problems," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[6] Y. Li, T. Du, K. Wu, J. Xu, and W. Matusik, "Diffcloth: Differentiable cloth simulation with dry frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 1, pp. 1–20, 2022.

[7] J. Liang and M. C. Lin, "Differentiable physics simulation," in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

[8] Y. Bai and C. K. Liu, "Coupling cloth and rigid bodies for dexterous manipulation," in *Proceedings of the 7th International Conference on Motion in Games*, 2014, pp. 139–145.

[9] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[10] X. Yu, S. Zhao, S. Luo, G. Yang, and L. Shao, "Diffclothai: Differentiable cloth simulation with intersection-free frictional contact and differentiable two-way coupling with articulated rigid bodies," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.

[11] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan, "Plasticinelab: A soft-body manipulation benchmark with differentiable physics," *arXiv preprint arXiv:2104.03311*, 2021.

[12] S. Li, Z. Huang, T. Chen, T. Du, H. Su, J. B. Tenenbaum, and C. Gan, "Dexdeform: Dexterous deformable object manipulation with human demonstrations and differentiable physics," *arXiv preprint arXiv:2304.03223*, 2023.

[13] 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.

[14] P. Du, M. Tang, C. Meng, R. Tong, and L. Lin, "A fluid/cloth coupling method for high velocity collision simulation," in *Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, 2012, pp. 309–314.

[15] 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.

[16] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[17] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[18] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[19] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 097–11 107.

[20] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 432–448.

[21] C. Gan, S. Zhou, J. Schwartz, S. Alter, A. Bhandwaldar, D. Gutfreund, D. L. Yamins, J. J. DiCarlo, J. McDermott, A. Torralba *et al.*, "The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai," *arXiv preprint arXiv:2103.14025*, 2021.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[23] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne *et al.*, "Jax: composable transformations of python+ numpy programs," 2018.

[24] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand, "Difftaichi: Differentiable programming for physical simulation," *arXiv preprint arXiv:1910.00935*, 2019.

[25] T. Stuyck and H.-y. Chen, "Diffxpbd: Differentiable position-based simulation of compliant constraint dynamics," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 6, no. 3, pp. 1–14, 2023.

[26] Y. Wang, J. Zheng, Z. Chen, Z. Xian, G. Zhang, C. Liu, and C. Gan, "Thin-shell object manipulations with differentiable physics simulations," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=KsUh8MMFKQ

[27] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, and W. Matusik, "Diffpd: Differentiable projective dynamics," *ACM Trans. Graph.*, vol. 41, no. 2, nov 2021.

[28] Z. Huang, D. C. Tozoni, A. Gjoka, Z. Ferguson, T. Schneider, D. Panozzo, and D. Zorin, "Differentiable solver for time-dependent deformation problems with contact," *arXiv preprint arXiv:2205.13643*, 2022.

[29] T. Harada, S. Koshizuka, and Y. Kawaguchi, "Real-time fluid simulation coupled with cloth." in *TPCG*, 2007, pp. 13–20.

[30] Y. Xiang, F. Chen, Q. Wang, Y. Gang, X. Zhang, X. Zhu, X. Liu, and L. Shao, "Diff-transfer: Model-based robotic manipulation skill transfer via differentiable physics simulation," *arXiv preprint arXiv:2310.04930*, 2023.

[31] J. K. Murthy, M. Macklin, F. Golemo, V. Voleti, L. Petrini, M. Weiss, B. Considine, J. Parent-Lévesque, K. Xie, K. Erleben *et al.*, "gradsim: Differentiable simulation for system identification and visuomotor control," in *International conference on learning representations*, 2020.

[32] X. Lin, Z. Huang, Y. Li, J. B. Tenenbaum, D. Held, and C. Gan, "Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools," *arXiv preprint arXiv:2203.17275*, 2022.

[33] J. Lv, Q. Yu, L. Shao, W. Liu, W. Xu, and C. Lu, "Sagci-system: Towards sample-efficient, generalizable, compositional, and incremental robot learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 98–105.

[34] J. Lv, Y. Feng, C. Zhang, S. Zhao, L. Shao, and C. Lu, "Sam-rl: Sensing-aware model-based reinforcement learning via differentiable physics-based simulation and rendering," *arXiv preprint arXiv:2210.15185*, 2022.

[35] 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.

[36] G. Yang, S. Luo, and L. Shao, "Jade: A differentiable physics engine for articulated rigid bodies with intersection-free frictional contact," *arXiv preprint arXiv:2309.04710*, 2023.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] 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 Trans. Graph.*, vol. 39, no. 4, p. 49, 2020.