2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Jade: A Differentiable Physics Engine for Articulated Rigid Bodies with Intersection-Free Frictional Contact

Gang Yang[1], Siyuan Luo[2], Yunhai Feng[3], Zhixin Sun[4], Chenrui Tie[1], and Lin Shao[1]

***Abstract*— We present Jade, a differentiable physics engine for articulated rigid bodies. Jade models contacts as the Linear Complementarity Problem (LCP). Compared to existing differentiable simulations, Jade offers features including intersection-free collision simulation and stable LCP solutions for multiple frictional contacts. We use continuous collision detection to detect the time of impact and adopt the backtracking strategy to prevent intersection between bodies with complex geometry shapes. We derive the gradient calculation to ensure the whole simulation process is differentiable under the backtracking mechanism. We modify the popular Dantzig's algorithm to get valid solutions under multiple frictional contacts. We conduct extensive experiments to demonstrate the effectiveness of our differentiable physics simulation over a variety of contact-rich tasks. Supplemental materials and videos are available on our project webpage at https://sites.google.com/view/diffsim**

I. Introduction

With recent advances in automatic differentiation methods [1–6], a number of differentiable physics engines, including rigid bodies [7, 8], soft bodies [3, 9–12], cloth [13–16], articulated bodies [17–19], and fluids [20–23], have been developed for solving system identification and control problems. These differentiable physics simulations provide differentiation operations to perform end-to-end optimization, which have been demonstrated to be effective for a broad range of application in robotics [3, 7–9, 11, 11, 13, 14, 14, 24–30].

Frictional contact is ubiquitous in robotics. Small violations of contact constraints introduce the penetration between articulated/rigid bodies, resulting in a significant deficiency in simulation accuracy and stability, especially for articulated rigid bodies. The majority of existing differentiable physics simulation engines perform poorly and result in penetration for contact-rich manipulation tasks requiring high-precision quality. Chen et al. [31] proposed a robotic simulation called Midas for articulated rigid body under the IPC formulation [32] to provide penetration-free simulation. However, Midas is not a differentiable simulation. Howell et al. [33] developed a differentiable physics simulation called Dojo to present a primal-dual interior-point to enforce no penetration. Dojo only supports primitive shapes and does not handle meshes for collision. Our differentiable simulation Jade adopts a backtracking strategy to handle collision response, which prevents penetration and performs the gradient calculation.

Fig. 1: Given that Jade is an intersection-free differential simulator, a Franka Panda robot arm can learn a controlled sequence for opening a bottle using a two-finger gripper, all while avoiding any penetration of the delicate cap.

Jade models contacts as the Linear Complementarity Problem (LCP). Although different methods have been proposed for solving the LCP, including the popular Dantzig's algorithm and Projected Gauss–Seidel (PGS), we frequently observe the Dantzig or PGS fails to find a solution under friction constraints, even in simple cases (such as pushing a cube to slide along horizontal tables with friction). In this work, we revised Danzig's algorithms to calculate the valid solution under the friction contacts.

We provide an open-source implementation of our differentiable physics simulation, which we call Jade. Code and documentation will be released on our project webpage at https://sites.google.com/view/diffsim.

[1]Gang Yang, Chenrui Tie, and Lin Shao are with the Department of Computer Science, National University of Singapore, Singapore. yg.matinal@gmail.com, duzhuohanyue@gmail.com, and linshao@nus.edu.sg

[2]Siyuan Luo is with the Department of Computer Science and Technology, Xi'an Jiaotong University, China. 312700@stu.xjtu.edu.cn

[3]Yunhai Feng is with the Department of Computer Science and Engineering, University of California San Diego, USA. yuf020@ucsd.edu

[4]Zhixin Sun is with the Department of Computer Science and Technology, Nanjing University, China. 201502009@smail.nju.edu.cn

In summary, we make the following contributions:

- We adopt continuous collision detection to calculate the time of impact and use the backtracking strategy to prevent intersection between objects with complex shapes.
- Under the backtracking mechanism, we derive symbolic differentiation rules for collision response and make the whole simulation differentiable.
- We revise the LCP solver based on Dantzig's pivoting algorithm to stably and precisely handle multiple frictional contacts.

II. Related Work

A. Contact Simulation

Contact simulation for rigid bodies is one of the core components of physics simulation. Solving the contact impulses under frictions is formulated as a nonlinear complementarity problem (NCP). Dojo [33] developed a primal-dual interior-point solver for the NCP problem. One common approach is to approximate the NCP as the LCP by approximating the friction cone with a polyhedral cone. A number of popular physics simulation engines integrated the LCP, such as ODE [34], Bullet [35], DART [36], Drake [37] and PhysX [38]. NCP/LCP formulations are referred to as hard contacts indicating the contact surfaces are rigid. Unlike the hard contact formulations, Mujoco [39] formulated the contact impulses calculation as a convex optimization problem by minimizing the post-collision kinetic energy. Another line of contact simulations uses compliant models [11, 40–44], assuming the contact surfaces can deform. These soft contact models allow the penetration to derive contact forces, which is not physically realistic. System parameters, such as object stiffness, might be difficult to tune for contact-rich manipulation tasks. Our differentiable physics simulation adopts the LCP formulation.

B. Numerical Methods for Linear Complementarity Problems

Multiple numerical algorithms have been proposed for solving the LCP. The LCP can be formulated as a minimization problem of a constrained convex QP problem, for which PGS is a suitable solver. Cottle and Dantzig [45] presented a pivot algorithm for LCP called the Dantzig algorithm. Baraff [46] extended the Dantzig algorithm to deal with friction and Coutinho [47] provided a clear description. The pivoting methods can find an accurate solution to the LCP at a small computation cost for relatively small constraint sizes, whereas the iterative methods generally produce approximate solutions. But in practice, we observe that Dantzig's algorithm occasionally gives an invalid solution while solving frictional LCP. We find out the reason and present a modified solver based on Dantzig's algorithm.

C. Differentiable Physics Simulation

Here we review only differentiable simulations that support articulated rigid bodies. de Avila Belbute-Peres et al. [7] used the LCP formulation and derived gradients of an LCP solution with respect to input parameters based on implicit differentiation. A number of works [8, 19, 24] modeled contacts as the LCP with the PGS as the solver. Heiden et al. [24] and Degrave et al. [8] leveraged existing automatic differentiation frameworks to get gradients whereas Qiao et al. [19] proposed a reverse version of the PGS solver using the adjoint method. Nimble [17] computed analytical gradients through the LCP by exploiting the sparsity of the LCP solution. Qiao et al. [14] leveraged the structure of contacts and grouped contacts into localized impact zones, where a QP is solved for each impact zone and the contact dynamics together with the conservation laws are not considered. Geilinger et al. [11] proposed a differentiable physics engine with implicit forward integration and customized frictional contact model and developed a dynamics solver that is analytically differentiable. Howell et al. [33] adopted the NCP formulation and developed a primal-dual interior-point solver while obtaining the gradient based on the implicit-function theorem. Our differentiable physics simulation adopts the analytical gradients when the time of impact backtracking is exported. We propose a composite differentiation rule to calculate gradients when collision happens.

III. Preliminaries

A. Continuous Collision Detection

In physical simulation, the motion of an object is modelled integrally through time discretization. In general, there are two ways to detect collisions, discrete collision detection (DCD) and continuous collision detection (CCD). Discrete collision detection computes and deals with penetration at each discrete time step. Continuous collision detection checks the collision within the time period and returns the collision time.

In this work, we adopt the bisection method in Tight-Inclusion [48] for CCD implementation. While Tight-Inclusion loops over all triangle pairs to check the intersection between two arbitrary deformable meshes, we use GJK algorithm for convex shapes instead, which speeds up CCD for rigid objects. To deal with real-world objects with more intricate geometries, our pipeline requires convex decomposition before simulation.

B. Linear Complementarity Problem

LCP is a classic model for rigid-body contact constraints with problem parameters A and b, where A is symmetric and positive semidefinite and reflects the masses and contact geometries of the bodies and b is a vector in the column space of A and reflects the external and inertial forces in the system. From Newton's law, we have the dynamic function: $a = Af + b$, where f is the vector of normal forces on each contact, and a is the relative acceleration between two objects with contact. This equation has infinite solutions because a is unknown. However, from contact constraints and conservation laws, the contact force f_i and relative acceleration a_i of contact point i should satisfy some conditions, so that we could iteratively solve f. As for frictionless contact i, we have:

$$a_i \geq 0, f_i \geq 0 \text{ and } f_i a_i = 0 \tag{1}$$

The solution $f = \text{LCP}(A,b)$ can be solved iteratively by Dantzig's algorithm [46].

C. Differentiable Simulator with DCD

An articulated rigid body with discrete timestep can be thought of as a simple function $[q_{t+1}, \dot{q}_{t+1}] = P(q_t, \dot{q}_t, \mu_t, \tau_t, \Delta t)$, where $P(\cdot)$ is the collision-free forward dynamics function. The collision detections and responses are all dealt with at the end of discrete timesteps so that $P(\cdot)$ is continuous and linear on Δt. $P(\cdot)$ takes current position q_t, velocity $\dot{q}_t$, control forces τ_t, inertial properties μ_t and timestep Δt as input, and returns the next state, q_{t+1} and $\dot{q}_{t+1}$. In this work, we use the explicit Euler integral method to formulate forward dynamics:

$$P(\cdot): \begin{cases} \dot{q}_{t+1} = \dot{q}_t + M_t^{-1} z_t \\ q_{t+1} = q_t + \Delta t \dot{q}_t \\ z_t \equiv \Delta t(\tau_t - c_t) + J_t^T f_t \\ f_t = \text{LCP}(A_t, b_t) \\ A_t = J_t M_t^{-1} J_t^T \\ b_t = J_t(\dot{q}_t + \Delta t M_t^{-1}(\tau_t - c_t)) \end{cases} \tag{2}$$

where M is the mass matrix, Δt is the timestep, c is Coriolis and gravitational force, f is the contact force, J is the contact Jacobian matrix and z is the total impulse. In our notation, q, $\dot{q}$ and τ are all expressed in generalized coordinates, describing the relative motion of joints in an articulated rigid body system. Thus joint constraint conditions are automatically satisfied, and we mainly need to care about contact constraints, which is represented by a linear complementarity problem (LCP) that we will introduce later. We calculate the contact force f by solving LCP.

Now that we have the full analytical formula of $P(q_t, \dot{q}_t, \mu_t, \tau_t, \Delta t)$, it's able to directly differentiate it and calculate full gradients [17] and we leave the details to supplementary.

IV. Differentiable Simulator Design

When the objects in a simulation are moving fast or thin enough, they might suffer severe penetration in DCD setting. To solve this problem, this work exports the continuous collision detection (CCD) module and the time-of-impact (TOI) backtracking strategy into forward dynamics. We then design the corresponding differentiation rule for the new dynamics. Besides, we occasionally notice invalid contact solutions from the current LCP solver and provide a modified solver to prevent invalid solutions.

A. Differentiable Simulator with CCD

1) Time-of-impact Backtracking: In simulation design, we use the time-of-impact (TOI) to record the exact time when two objects collide. Continuous collision detection (CCD) is used to predict TOI. If TOI is less than the timestep Δt, we know there will be a collision within this time interval. Then we should track the motion back to TOI (the moment collision happens), deal with collision response, and then run over the remained time.

As mentioned in the previous section, the forward dynamics without collision can be described as $[q_{t+1}, \dot{q}_{t+1}] = P(q_t, \dot{q}_t, \mu_t, \tau_t, \Delta t)$, where timestep Δt is a constant. But when a collision is detected by CCD at TOI, the whole forward dynamics would be divided into three parts: two collision-free propagation before and after the collision, and one collision response at TOI, where both propagation timesteps are conditioned on the previous state. We use $[q^-, \dot{q}^-]$ and $[q^+, \dot{q}^+]$ to denote the state right before and after collision, and use Δt_c to denote TOI. The full forward dynamics $F(\cdot)$ over dt becomes:

$$F(\cdot): \begin{cases} [q^-, \dot{q}^-] = P(q_t, \dot{q}_t, \mu_t, \tau_t, \Delta t_c) \\ [q^+, \dot{q}^+] = C(q^-, \dot{q}^-, \mu_t) \\ [q_{t+1}, \dot{q}_{t+1}] = P(q^+, \dot{q}^+, \mu_t, \tau_t, \Delta t - \Delta t_c) \\ \Delta t_c = T(q_t, \dot{q}_t) \end{cases} \tag{3}$$

where $C(q, \dot{q}, \mu)$ is the collision response function and $T(q, \dot{q})$ is the collision detection function. After deriving $C(\cdot)$, $T(\cdot)$ and their differentiations in the following sections, we can combine with $P(\cdot)$ and get the full formula of forward and backward dynamics.

2) Collision Response: Although collision usually happens between one pair of objects, all objects linking to them through chains of constraints (joint and contact) should be considered when we calculate the response impulse f_c. In physics, the impulse conductive velocity of a rigid body is infinity, so the collision responses of all relevant objects are solved simultaneously. We use v^- and v^+ to denote the relative velocities of all contact points before and after the collision, where $v > 0$ means separating, $v < 0$ means approaching and $v = 0$ means static. From the simultaneous collision analysis in [49], the physical conditions are:

$$v_i^+ + \varepsilon_i v_i^- \geq 0, f_{c,i} \geq 0 \text{ and } f_{c,i}(v_i^+ + \varepsilon_i v_i^-) = 0 \tag{4}$$

where $\varepsilon_i \geq 0$ is the restitution coefficient of each contact. The Newton's law in contact space again gives $A_c f_c = v^+ - v^-$, which can be rewritten as $A_c f_c + b_c = v^+ + \varepsilon v^-$, where $b = (1+\varepsilon)v^-$. So the collision response problem is still an LCP. Since collision happens in an instant, all the impulses from external force and gravity are equal to zero. We can regard the collision response function as a variation of the propagation function by setting $\Delta t = 0$, i.e. $C(q, \dot{q}, \mu) = P(q, \dot{q}, \mu, \tau = 0, \Delta t = 0)$ with a modified LCP parameter $b_c = (1+\varepsilon)J\dot{q}$:

$$C(\cdot): \begin{cases} q^+ = q^- \\ \dot{q}^+ = \dot{q}^- + M^{-1} J_c^T f_c \\ f_c = \text{LCP}(A_c, b_c) \\ A_c = J_c M^{-1} J_c^T \\ b_c = (1+\varepsilon) J_c \dot{q}^- \end{cases} \tag{5}$$

where J_c is the Jacobian at collision. The gradients of $C(\cdot)$ are directly obtained by substituing $C(q,\dot{q},\mu) = P(q,\dot{q},\mu,\tau = 0,\Delta t = 0)$ and $f = \text{LCP}(A_c, b_c)$ and we leave the gradient details to supplementary.

3) Differentiate TOI: The time of impact is obtained from an iterative calculus in CCD, which is hard to directly differentiate step by step. So we derive another expression of TOI assuming we have already known the collision point (which is exactly solved by CCD), TOI is just the time for relative normal distance $\|n\|$ at the beginning of timestep to decrease to zero:

$$T : \Delta t_c = \frac{\|n\|}{(v_t^B - v_t^A)\cdot n} = \frac{\|n\|}{J_c^n \dot{q}_t} \tag{6}$$

The relation between $\|n\|$ and q_t is $\frac{\|n\|}{q_t} = -J_c^n$, where J_c^n is the Jacobian of collision point's normal direction component at TOI. With this relation, we can differentiate TOI as:

$$\partial T : \begin{cases} \dfrac{\partial \Delta t_c}{\partial q_t} = -\dfrac{J_c^n}{J_c^n \dot{q}_t} \\ \dfrac{\partial \Delta t_c}{\partial \dot{q}_t} = -\dfrac{J_c^n}{J_c^n \dot{q}_t}\Delta t_c \end{cases} \tag{7}$$

B. Modified LCP Solver

In this section, we first briefly introduce Dantzig's algorithm [46] for frictional LCP, then point out why its implementation in ODE is inaccurate and introduce our modification.

1) Dantzig's algorithm: We start from frictionless LCP, Dantzig's algorithm decomposes the nonlinear complementarity conditions in Eq.1 into a set of linear constraints and classifies all contacts into two classes $\mathscr{C} = \{C, N\}$, where $C : a = 0, f \le 0$ means clamping and $N : f = 0, a \le 0$ means separating.

Once we have classified all contacts, we can rearrange the contact indices and represent the dynamic equation in a block form:

$$\begin{bmatrix} a_C \\ a_N \end{bmatrix} = \begin{bmatrix} A_{CC} & A_{CN} \\ A_{NC} & A_{NN} \end{bmatrix} \begin{bmatrix} f_C \\ f_N \end{bmatrix} + \begin{bmatrix} b_C \\ b_N \end{bmatrix} \tag{8}$$

Applying the constraints of each class, the original LCP becomes the following linear equations, which are easy to solve:

$$\begin{cases} E_C : A_{CC} f_C + b_C = 0 \\ E_N : f_N = 0 \end{cases} \tag{9}$$

So the remaining problem is assigning contacts to the correct classes, Alg.1 describes an iterative method. Without losing generality, we let $\mathscr{C} = \{C_1, C_2, ..., C_m\}$ denote the contact classes and $\mathscr{E} = \{E_1, E_2, ..., E_m\}$ denote the linear equations of contact forces. Each class C_i corresponds to a linear constraint of (f,a) and has an available set S_i. The union $\bigcup_{i=1}^m S_i$ is assumed connected, so the transition between any pair of classes can be carried out by changing (f,a) continuously.

We start from an empty contact set and add new contacts one by one. Every iteration when we add a new contact $k+1$ of which the class is unknown, we move (f_{k+1}, a_{k+1}) towards $\bigcup_{i=1}^m S_i$ while maintaining previous k contacts under constraints. During this process, the class assignment of previous k contacts may change and finally $k+1$ contacts reach a new equilibrium. We repeat this iteration until all D contacts are considered.

Algorithm 1 Dantzig's Algorithm

```
D denotes the dimension of the contact force
c(k) denotes the class label of contact i
C_1, C_2, ..., C_s ← ∅
f ← 0, a ← b, k ← 0
while k < D do
    while (f_{k+1}, a_{k+1}) ∉ ⋃_{i=1}^m S_i do
        Δf_{k+1} ← 1
        Calculate Δf_i|_{i=1}^k by differentiating 𝓔
        Δa ← AΔf
        Increase s from 0 until there exists
        i ≤ k and a neighboring C' ∈ 𝒞 s.t.
        (f_i + sΔf_i, a_i + sΔa_i) = C_{c(k+1)} ∩ C'
        f ← f + sΔf, a ← a + sΔa
        Move i from C_{c(k+1)} to C'
        Update 𝓔
    end while
    Assign k+1 to C_i that (f_{k+1}, a_{k+1}) ∈ S_i
    Update 𝓔
    k ← k+1
end while
Solve linear equations 𝓔
```

In frictional LCP, contact force components consist of normal forces and friction forces. As for normal forces, the classification is the same as frictionless case. As for friction forces, the classification is $\mathscr{C} = \{F, H, L\}$, where $F : a = 0, -\mu f_n < f < \mu f_n$ means static friction, $H : a < 0, f = \mu f_n$ means reaching higher Column friction bound and $L : a > 0, f = -\mu f_n$ means reaching lower Column friction bound. Here f_n denotes the corresponding normal force. After representing the dynamic equation in a block form, the total linear equation set is:

$$\begin{cases} E_C : (A_{CC_H} + \mu A_{CH}) f_{C_H} + (A_{CC_H} - \mu A_{CL}) f_{C_L} \\ \qquad + A_{CC_F} f_{C_F} + A_{CF} f_F + b_C = 0 \\ E_N : f_N = 0 \\ E_F : (A_{FC_H} + \mu A_{FH}) f_{C_H} + (A_{FC_H} - \mu A_{FL}) f_{C_L} \\ \qquad + A_{FC_F} f_{C_F} + A_{FF} f_F + b_F = 0 \\ E_H : f_H = \mu f_{C_H} \\ E_L : f_L = -\mu f_{C_L} \end{cases} \tag{10}$$

Dantzig's algorithm solves frictional LCP similarly following Alg.1 by substituting the frictional contact classes.

2) Failure Issues and Our Improvements: In practice, however, Dantzig's LCP solver implemented by ODE sometimes fails for two reasons:

(i). Ignoring the correlation between friction limit and normal force while updating forces. For example, when a frictional force f_i in class F is going to reach the boundary between F and H, current solver calculates the maximum step as $s = \frac{\mu f_{n_i} - f_i}{\Delta f_i}$. However, f_{n_i} is not a constant value as f varies and f_i's upper bound μf_{n_i} is not constant either. The true step should be $s = \frac{\mu f_{n_i} - f_i}{\Delta f_i - \mu \Delta f_{n_i}}$ and we fix this issue in our new solver.

(ii). No insurance of convergence. Every iteration k, while we continuously increase s, previous classes might be changed one by one. However, there are certain class conbinations that need simultaneously changing several classes, which can not be explored by the line search to s. Frictional LCP sometimes suffers infinite loops and the algorithm fails to converge. So our new solver takes the original line search as an initial try and applies ergodic search from the initial try when a loop is detected until the correct class assignment is found. Implement details and examples are put in supplementary.

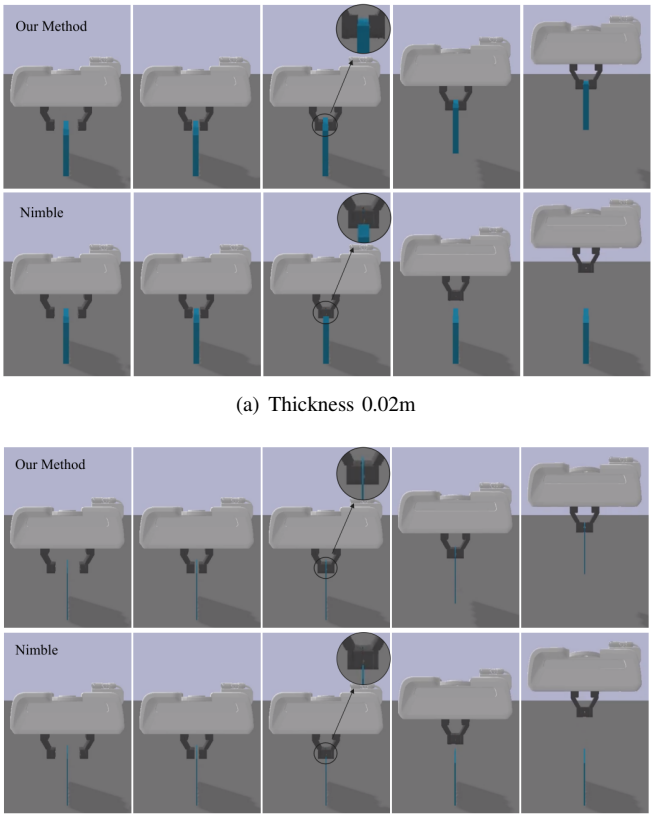
V. Experiments

Our experiments aim to address two key questions: 1) Does our differentiable simulator reliably generate intersection-free forward simulations? 2) Can it accurately compute gradients for control optimization? To showcase its capabilities, we illustrate its performance through various contact-rich robot manipulation tasks.

A. Precise Collision Detection and Response

Picking up thin-shell objects like plates and boards presents a challenge in robotic simulation due to the potential intersection between grippers and objects. In this experiment, we use force control to lift thin plates with a Franka robot's gripper, measuring 0.02m and 0.002m in thickness. Our task involves the gripper closing its fingers to grasp the object and then moving upward.

Trajectories are shown in Fig. 2. Comparing our results with Nimble [17], which uses DCD and an inaccurate LCP solver, highlights the superior performance of our simulator. While Nimble's gripper penetrates both plates, ours successfully holds and lifts them, demonstrating our high-precision CCD, accurate collision response, and correct friction calculation with LCP.



(a) Thickness 0.02m

(b) Thickness 0.002m

Fig. 2: Comparison of gripper performance in picking up a thin plate with thicknesses of 0.02m (a) and 0.002m (b) using our simulator and Nimble.

B. Gradient-based Optimization for Robot Manipulation

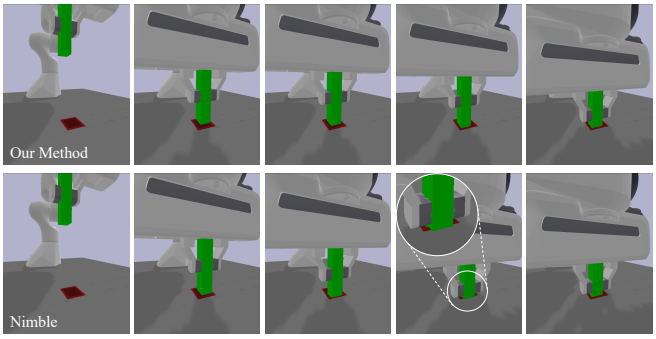
1) Peg-in-Hole: In this experiment, force control guides a Franka robot in inserting a long cube into a slightly wider hole. Control force sequence optimization through gradient descent, employing a differentiable simulator, aligns and stabilizes the cube within the hole. Results are compared with Nimble [17].

A time step of 0.01s with no gravity or friction is adopted. The green cube (0.017m×0.017m width, 0.08m length) is gripped, while the red hole (0.022m×0.022m wide, 0.04m deep) is placed randomly. The simulation comprises 40 time steps, divided into two stages, minimizing the position and orientation difference between the cube and a target pose. Trajectories are optimized until convergence (loss $< 5 \times 10^{-5}$). The learning curve of our method is shown in Fig. 4, showing that both two stages of optimization can convergent quickly.

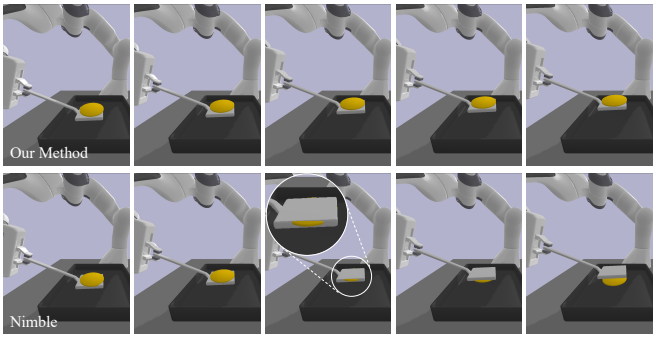
Fig. 3(a) depicts optimized trajectories. While Nimble achieves zero loss quickly, its trajectory proves impractical due to intersection issues. In contrast, our method, Jade, prioritizes intersection-free movements, resulting in longer optimization times but ultimately finding feasible trajectories. Jade's approach involves initial downward insertion, followed by horizontal adjustments, achieving stable positioning within the hole without table intersection.

2) Lift a Pancake: This experiment uses force control to elevate a thin pancake with a Franka robot using a spatula, overcoming gravity to reach a specified height while ensuring horizontal stability. The loss function incorporates alignment and velocity considerations for maintaining a position at the target point. The simulation uses a time step of 0.01s, gravitational acceleration of 10m/s², and friction coefficient of 1.0. The pancake is a disk (0.048m radius, 0.01m thickness), while the spatula (0.09m×0.12m width, 0.01m thick) is fixed to the gripper.

The robot arm holds the spatula above the table initially, parallel to it, with the pancake slightly above. The simulation

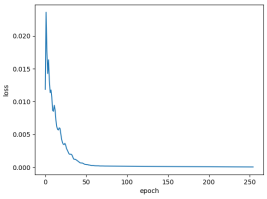


(a) Peg-in-Hole

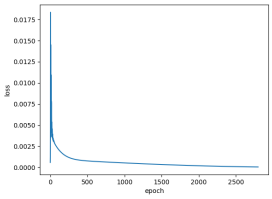


(b) Lift a Pancake

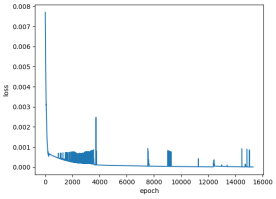
Fig. 3: Optimized trajectories for inserting a peg into a hole (a) and lifting a pancake (b) by our method and Nimble.



(a) Peg Insertion Stage 1



(b) Peg Insertion Stage 2



(c) Lift a Pancake

Fig. 4: Learning Curves of Gradient-based Optimizations in Jade

spans 30 time steps, minimizing the position and orientation difference between the pancake and the target pose, set approximately 0.04m higher. Trajectories are optimized until convergence (loss $< 1 \times 10^{-5}$). The learning curve of our method is shown in Fig. 4. The fluctuation in the learning curve comes from the discrete gain and loss of contacts because lifting the pancake contains acceleration and deceleration.

Comparison with Nimble [17] demonstrates Jade's success in preventing penetration and maintaining pancake stability on the spatula. While Nimble fails due to pancake penetration under gravity, Jade handles contact and collision, ensuring the pancake remains on the spatula without sliding, even with potential spatula tilting during optimization. Optimized trajectories are in Fig. 3(b).

C. Frictional Sliding

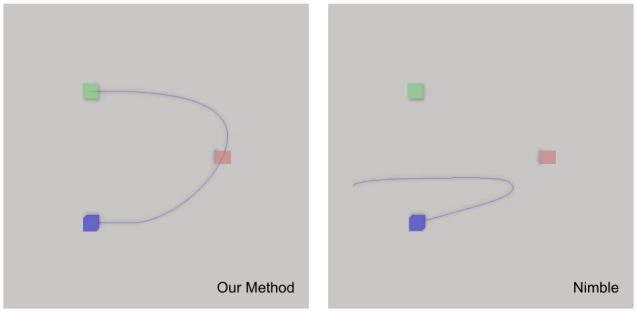


Fig. 5: Push a cube on a frictional table, the blue cube represents the start point A, the red cube represents the middle target point B and the green cube represents the final target point C.

This experiment optimizes the planar force sequence for a cube on a frictional table to move from point A to point C and remain there, passing through point B. Control sequences learned from simulators are tested in a ground truth environment where friction is analytically solved. Both Jade and Nimble are employed, with their resulting trajectories in the test environment depicted in Fig. 5.

While Jade's precise handling of frictional contact forces ensures the achievement of targets, Nimble struggles due to multiple contact friction, resulting in optimization failure. This suggests that Jade may have a smaller sim-to-real gap, making it more suitable for real-world tasks.

VI. Conclusion

This paper introduces Jade, a differentiable physics engine for articulated rigid bodies. We use continuous collision detection to detect the time of impact for collision checking and adopt the backtracking strategy to prevent intersection between bodies with complex geometric shapes. We derive the gradient calculation to ensure the whole simulation process is differentiable under the backtracking mechanism. We also modify the popular Dantzig's algorithm to get a valid solution under multiple frictional contacts. Experiments on a variety of contact-rich robot manipulation tasks demonstrate the superior performance of our method.

References

[1] 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, pp. 8026–8037, 2019.

[2] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[3] 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, p. 201, 2019.

[4] B. Bell, "Cppad: a package for c++ algorithmic differentiation," http://www.coin-or.org/CppAD, 2020.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[6] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[7] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf

[8] J. Degrave, M. Hermans, J. Dambre *et al.*, "A differentiable physics engine for deep learning in robotics," *Frontiers in neurorobotics*, p. 6, 2019.

[9] 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.

[10] K. M. Jatavallabhula, 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," *arXiv preprint arXiv:2104.02646*, 2021.

[11] M. Geilinger, D. Hahn, J. Zehnder, M. Bächer, B. Thomaszewski, and S. Coros, "Add: analytically differentiable dynamics for multi-body systems with frictional contact," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[12] 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. [Online]. Available: https://doi.org/10.1145/3490168

[13] J. Liang, M. Lin, and V. Koltun, "Differentiable cloth simulation for inverse problems," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/28f0b864598a1291557bed248a998d4e-Paper.pdf

[14] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Scalable differentiable physics for learning and control," *arXiv preprint arXiv:2007.02168*, 2020.

[15] 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.

[16] 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.

[17] K. Werling, D. Omens, J. Lee, I. Exarchos, and C. K. Liu, "Fast and feature-complete differentiable physics for articulated rigid bodies with contact," in *Proceedings of Robotics: Science and Systems (RSS)*, July 2021.

[18] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem," in *Robotics*, ser. Robotics: Science and Systems, S. Srinivasa, N. Ayanian, N. Amato, and S. Kuindersma, Eds. United States: MIT Press Journals, 2017, publisher Copyright: © 2017 MIT Press Journals. All rights reserved.; 2017 Robotics: Science and Systems, RSS 2017 ; Conference date: 12-07-2017 Through 16-07-2017.

[19] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Efficient differentiable simulation of articulated bodies," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8661–8671.

[20] K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey, "Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6111–6122, 2020.

[21] N. Wandel, M. Weinmann, and R. Klein, "Learning incompressible fluid dynamics from scratch–towards fast, differentiable fluid models that generalize," *arXiv preprint arXiv:2006.08762*, 2020.

[22] P. Holl, V. Koltun, and N. Thuerey, "Learning to control pdes with differentiable physics," *arXiv preprint arXiv:2001.07457*, 2020.

[23] T. Takahashi, J. Liang, Y.-L. Qiao, and M. C. Lin, "Differentiable fluids with solid coupling for learning and control," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 7, pp. 6138–6146, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16764

[24] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "NeuralSim: Augmenting differentiable simulators with neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://github.com/google-research/tiny-differentiable-simulator

[25] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning –extended abstract," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*. California: International Joint Conferences on Artificial Intelligence, 2019, pp. 6231–6235. [Online]. Available: https://www.ijcai.org/Proceedings/2019/

[26] C. Schenck and D. Fox, "Spnets: Differentiable fluid dynamics for deep neural networks," in *Conference on Robot Learning*. PMLR, 2018, pp. 317–335.

[27] P. Holl, N. Thuerey, and V. Koltun, "Learning to control pdes with differentiable physics," in *International Conference on Learning Representations*, 2019.

[28] X. Zhu, J. Ke, Z. Xu, Z. Sun, B. Bai, J. Lv, Q. Liu, Y. Zeng, Q. Ye, C. Lu, M. Tomizuka, and L. Shao, "Diff-lfd: Contact-aware model-based learning from visual demonstration for robotic manipulation via differentiable physics-based simulation and rendering," in *Conference on Robot Learning (CoRL)*, 2023.

[29] 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," 2023.

[30] 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 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[31] Y. Chen, M. Li, W. Lu, C. Fu, and C. Jiang, "Midas: A multi-joint robotics simulator with intersection-free frictional contact," *arXiv preprint arXiv:2210.00130*, 2022.
[32] 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.
[33] T. A. Howell, S. L. Cleac'h, J. Z. Kolter, M. Schwager, and Z. Manchester, "Dojo: A differentiable simulator for robotics," *arXiv preprint arXiv:2203.00806*, 2022.
[34] R. Smith, "Open dynamics engine," 2008, http://www.ode.org/. [Online]. Available: http://www.ode.org/
[35] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.
[36] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. Karen Liu, "Dart: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.
[37] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu
[38]
[39] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control." in *IROS*. IEEE, 2012, pp. 5026–5033. [Online]. Available: http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12
[40] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli, "Automatic differentiation of rigid body dynamics for optimal control and estimation," *Advanced Robotics*, vol. 31, no. 22, pp. 1225–1237, 2017.
[41] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," *Robotics: Science and Systems XIV*, 2018.
[42] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "Neuralsim: Augmenting differentiable simulators with neural networks," *arXiv preprint arXiv:2011.04217*, 2020.
[43] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax - a differentiable physics engine for large scale rigid body simulation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. [Online]. Available: https://openreview.net/forum?id=VdvDlnnjzIN
[44] M. Macklin, "Warp: A high-performance python framework for gpu simulation and graphics," https://github.com/nvidia/warp, March 2022, nVIDIA GPU Technology Conference (GTC).
[45] R. W. Cottle and G. B. Dantzig, "Complementary pivot theory of mathematical programming," *Linear Algebra and its Applications*, vol. 1, no. 1, pp. 103–125, 1968. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0024379568900529
[46] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, 1994, p. 23–34. [Online]. Available: https://doi.org/10.1145/192161.192168
[47] M. G. Coutinho, *Guide to Dynamic Simulations of Rigid Bodies and Particle Systems*. Springer Publishing Company, Incorporated, 2012.
[48] B. Wang, Z. Ferguson, T. Schneider, X. Jiang, M. Attene, and D. Panozzo, "A large scale benchmark and an inclusion-based algorithm for continuous collision detection," *ACM Transactions on Graphics*, vol. 40, no. 5, Oct. 2021.
[49] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," vol. 23, no. 3, p. 223–232, jul 1989. [Online]. Available: https://doi.org/10.1145/74334.74356