DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact

Yifei Li, Tao Du, Kui Wu, Jie Xu, Wojciech Matusik
MIT CSAIL
Cloth simulation has wide applications...

Film/Game

Garment Design

Robotics

Virtual Try-On
Slow & Tedious Manual Workflow

Edit

Iter 1  Iter 2  Iter 3  Iter 4  ...

Simulate

Final Design
Differentiable Cloth Simulation

Goal: Optimize $\theta$ (e.g. cloth material) to perform a task (e.g. garment design)
Differentiable Cloth Simulation

Goal: Optimize $\theta$ (e.g. cloth material) to perform a task (e.g. garment design)

1. **Forward Simulation through time to obtain $L(\theta)$**

2. **Gradient Back Propagation** $\frac{\partial L}{\partial \theta} \rightarrow L$ to obtain $\frac{\partial L}{\partial \theta}$

3. Use gradient-based optimizer to update $\theta$

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \frac{\partial L}{\partial \theta} \cdot k$$
Related Works

- Liang et al. 19 [NeurIPS]: Differentiable cloth simulation for inverse problems
- Ly et al. 20 [SIGGRAPH]: Projective Dynamics with dry frictional contact
- Du et al. 21 [TOG]: DiffPD: Differentiable Projective Dynamics
Contributions of DiffCloth

**Fast Simulation + Gradient Derivation**
- Projective-Dynamics-based forward simulation
- Novel gradient computation to speed up back-propagation

**Accurate Contact Modeling**
- Dry-frictional contact

**Effective in Inverse Tasks**
- Trajectory Optimization
- Closed-Loop Control
- Inverse Design
- System Identification
- Real-to-Sim
Simulating Cloth Dynamics

Implicit Euler integration is robust

Newton's 2nd Law with Implicit Euler:

\[ x_{t+1} - h^2 M^{-1} f(x_{t+1}) = x_t + h v_t \]

\( x_t, v_t \) position, velocity

\( h \) timestep

\( M \) mass matrix

\( f \) force
Simulating Cloth Dynamics

Implicit Euler integration is robust but expensive

Using Newton’s method requires costly Hessian matrix computation and factorization of \( A_t \) at every timestep \textit{(slow)}
Fast Simulation with Projective Dynamics

local/global iterative scheme [Bouaziz et al. 14]

**Global:** Same system matrix P at every timestep

\[ P^k x_t^{k+1} = b(p)^k_{t+1} \]

**Local:** parallel local projections \( p \)

enforce vertex-vertex frictional contact semi-implicitly to satisfy Signorine-Coulomb condition

\[ P v^{k+1} = b(p)^k \]
\[ := f(p)^k + r^k \]

impulse  contact impulse
PD with Dry Frictional Contact [Ly et al. 20]

enforce vertex-vertex frictional contact semi-implicitly to satisfy Signorine-Coulomb condition

\[ P \, v^{k+1} = b(p)^k \]
\[ := f(p)^k + r^k \]

**impulse**  **contact impulse**

**Take Off**  \( f_N \geq 0 \)
\( r = 0 \)

**Stick**  \( f_T \leq \mu f_N \)
\( r = -f \)

**Slide**  \( f_T > \mu f_N \)
\( r_N = -f_N \)
\( r_T = -\mu f_N \)

**Signorini Condition**  \( r = 0, v_N > 0 \)
\( ||r_T|| < \mu r_N, v = 0 \)
\( ||r_T|| = \mu r_N, v_N = 0, r_T || v_T, r_T \cdot u_t \leq 0 \)
\[ P v_{t+1}^{k+1} = b_{t+1}^k \]

Gradient computation via adjoint method

\[ \frac{\partial L}{\partial v_i} = M \left(P - \Delta P_i - \Delta R_i\right)^{-1} \frac{\partial L}{\partial v_{t+1}} \]

\( \Delta P \) Gradient for the projection vector \( p \)
\( \Delta R \) Gradient for the contact impulse response vector \( r \)
Slow Gradient Computation

\[ P \, v_{t+1}^{k+1} = b_{t+1}^k \]

\[ \frac{\partial L}{\partial v_t} = M \left( P - \Delta P_t - \Delta R_t \right)^{-1} \frac{\partial L}{\partial v_{t+1}} \]

\[ \Leftrightarrow Z_t \text{ adjoint vector} \]
Slow Gradient Computation

\[ P v_{t+1}^{k+1} = b_{t+1}^k \]

\[ \frac{\partial L}{\partial v_t} = M \]

\[ (P - \Delta P_t - \Delta R_t) z_t = \frac{\partial L}{\partial v_{t+1}} \]
Slow Gradient Computation

Can we exploit the source of efficiency in forward solve for backward solve?

\[ P v_{t+1}^{k+1} = b_{t+1}^{k} \]

\[ \frac{\partial L}{\partial v_t} = M \]

\[ (P - \Delta P_t - \Delta R_t) z_t = \frac{\partial L}{\partial v_{t+1}} \]

constant  
timestep-dependent  
slow to factorize
Fast Gradient Computation

\[(P - \Delta P_t - \Delta R_t) z_t = \frac{\partial L}{\partial v_{t+1}}\]

**Direct Solve**

\[P z_{t+1}^{k+1} = (\Delta P_t + \Delta R_t) z_{t+1}^{k} + \frac{\partial L}{\partial v_{t+1}}\]

**Iterative Solve: Good convergence in practice**

**Matrix Splitting**

**constant**  **timestep-dependent**

**constant: pre-factorize**
Fast Differentiable Cloth Simulation (Backward)
Iterative Solver Speedup (convergence $\epsilon = 1e-4$): 3x - 12x

<table>
<thead>
<tr>
<th></th>
<th>Cloth Grid Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12x12</td>
</tr>
<tr>
<td>Wind</td>
<td>2.9x</td>
</tr>
<tr>
<td>Slope</td>
<td>3.1x</td>
</tr>
</tbody>
</table>

Wind: minimal contact
Slope: maximal contact
Inverse Task Comparison with Gradient-Free Methods

Benchmark Test: Optimize force field on the cloth to reach the ring

![Graph showing comparison between gradient-based (LBFGS) and gradient-free (CMA-ES) methods. The graph illustrates the loss over optimization timesteps and speedup over optimization DoF.]

- Gradient-free (CMA-ES)
- Gradient-based (LBFGS)
Task: Identify wind model and material parameters to match target trajectory

4300 DoF | 250 Timesteps | $\Delta t = 1/90$s
6 Design Parameters: cloth stretching stiffness and sinusoidal wind model parameters
**Trajectory Optimization**

Task: Optimize manipulator end effector trajectories to pull a sock on the foot model

**Optimized Trajectory**

1700 DoF | 400 Timesteps | $\Delta t = 1/100s$

36 Design Parameters: Tangents and endpoints of the 4 Hermite Splines
**Inverse Design**

Task: Optimize dress material parameters so that the spinning angle of the dress is 50 degrees

19000 DoF | 125 Timesteps | $\Delta t = 1/120s$

2 Design Parameters: density and bending stiffness
Task: A generalizable NN controller that puts hat onto the head from any initial positions around the upper hemisphere

1700 DoF | 400 Timesteps | $\Delta t = 1/100$s
117000 Design Parameters: Network parameters of the 2-layer MLP
85x more sampling efficient compare with Reinforcement Learning baseline
A differentiable cloth simulator with dry frictional contact

Fast simulation with Projective Dynamics & fast back-propagation with iterative solver

More sampling efficient than gradient-free methods

Effective in a wide range of inverse tasks