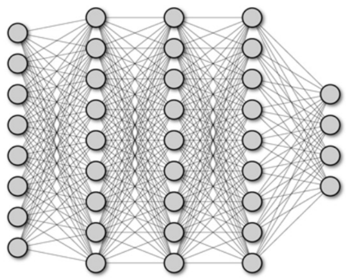
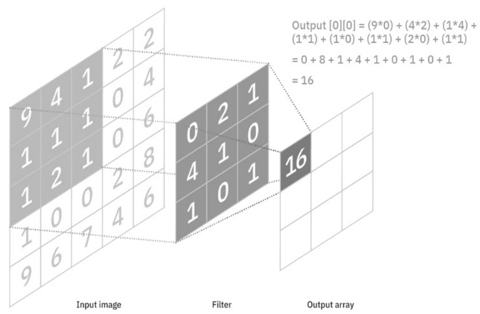
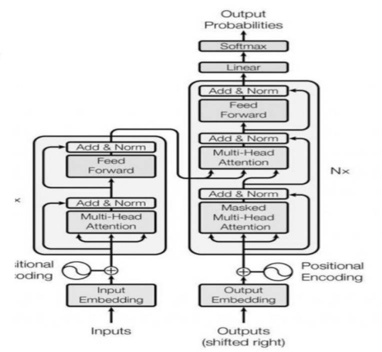
Learning The Physics World with
Differentiable Simulation & Geometry

- In Vision, Language, Robotics, etc…
- Fully connected networks, CNNs, Transformers, etc.

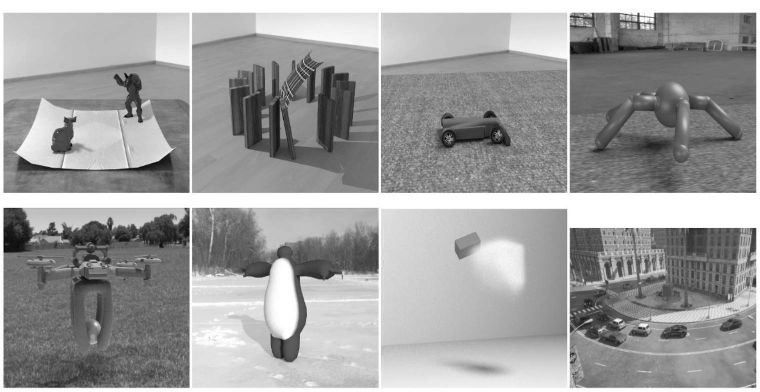


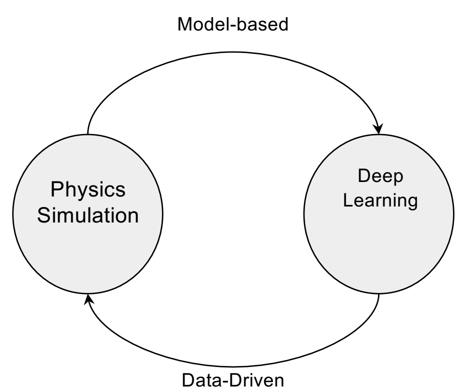




Physics Simulation

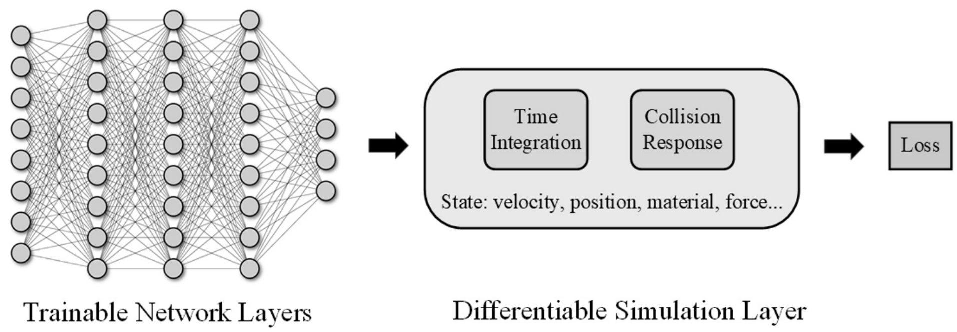
- Serve as an inductive bias for learning algorithms
- Utilize the power of deep learning to solve physics problems





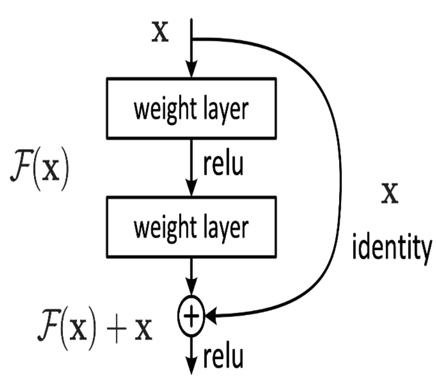
Differentiable Physics Simulation
as a Network Layer

- Physical property estimation (material, friction, etc)
- Control of physical systems
- Model-based RL

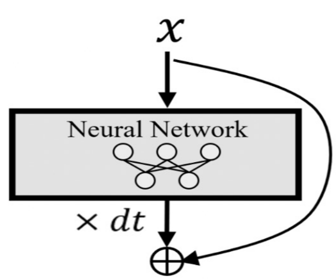


41

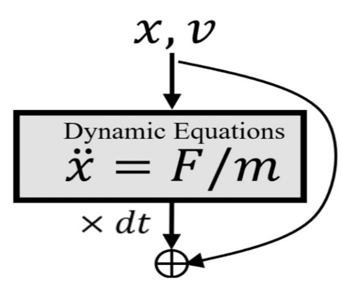
Differentiable Physics Simulation
as a Network Layer



ResNet

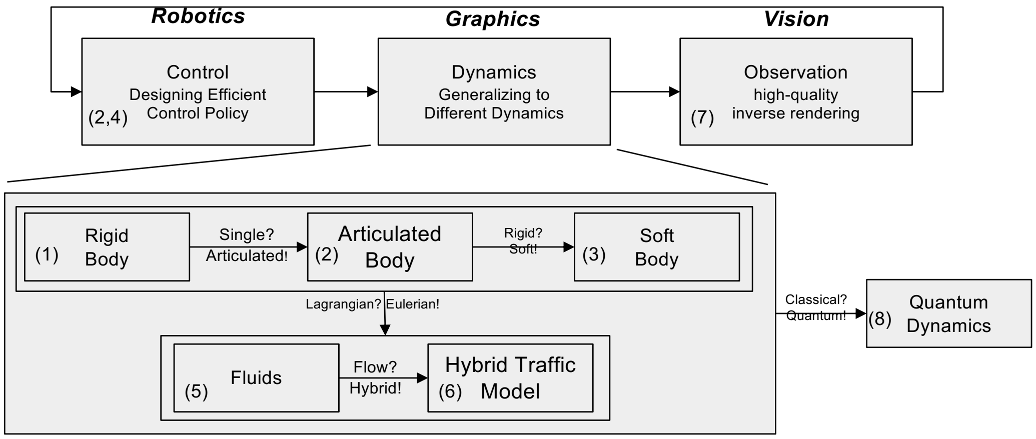


NeuralODE
(ResNet)



Differentiable Physics

42



(1) Scalable Differentiable Physics for Learning and Control. *(ICML 2020)*
(2) Efficient Differentiable Simulation of Articulated Bodies. *(ICML 2021)*
(3) Differentiable Simulation of Soft Multi-body Systems. *(NeurIPS 2021)*
(4) OF-VO: Efficient Navigation among Pedestrians Using Commodity Sensors. *(ICRA/RAL 2021)*
(5) Differentiable Fluids with Solid Coupling for Learning and Control. *(AAAI 2021)*
(6) Differentiable Hybrid Traffic Simulation. *(Siggraph Asia 2022, Journal Track)*
(7) NeuPhysics: Editable Neural Geometry and Physics from Monocular Videos. *(NeurIPS 2022)*
(8) Differentiable Analog Quantum Computing for Optimization and Control. *(NeurIPS 2022)*

43

Differentiable Cloth Simulation

Junbang Liang[1], Ming Lin[1], and Vladlen Koltun[2]

[1]University of Maryland at College Park

[2]Intel Labs

https://gamma.umd.edu/researchdirections/virtualtryon/differentiablecloth

UNIVERSITY OF MARYLAND

NeurIPS 2019

intel

44

44

Limitations with State-of-the-Art

- Differentiable rigid body simulation

✓ *Formulation not scalable to high dimensionality*

- Learning-based physics

✓ *Unable to guarantee physical correctness*

45

45

Key Contributions

- Dynamic collision detection to reduce collision dimensionality
- Gradient computation of collision response using implicit differentiation
- Optimized backpropagation using QR decomposition

46

46

Gradients of Physics Solve

- Formulation: $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$
- Input: $\hat{\mathbf{M}}$ and $\mathbf{f}$. Output: $\mathbf{a}$
- Back propagation: use $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$ to compute $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$, where $\mathcal{L}$ is the loss function.
- Solution: $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d_a}\mathbf{z}^\top \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d_a}^\top$,

  where $\mathbf{d_a}$ is computed from $\hat{\mathbf{M}}^\top \mathbf{d_a} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}}^\top$, and $\mathbf{z}$ is the solution of $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$.

47

47

Collision Response

- Collision Detection: $dist(\text{node}_i, \text{face}_j, t) < \delta$, where δ is the cloth thickness, and t is some time between two steps.
- Objective: introduce minimum energy to avoid collision:

$$dist(\text{node}_i, \text{face}_j, t) - \delta \geq 0$$

- Constraint formulation: $\mathbf{Gx} + \mathbf{h} \leq 0$
- Objective formulation: Quadratic Programming:

$$\begin{array}{ll} \underset{\mathbf{z}}{\text{minimize}} & \frac{1}{2}(\mathbf{z} - \mathbf{x})^\top \mathbf{W}(\mathbf{z} - \mathbf{x}) \\ \text{subject to} & \mathbf{Gz} + \mathbf{h} \leq \mathbf{0} \end{array}$$

48

48

Gradients of Collision Response

- Karush-Kuhn-Tucker (KKT) condition:

$$\begin{aligned} \mathbf{W}\mathbf{z}^* - \mathbf{W}\mathbf{x} + \mathbf{G}^\top \lambda^* &= 0 \\ D(\lambda^*)(\mathbf{G}\mathbf{z}^* + \mathbf{h}) &= 0 \end{aligned}$$

- Implicit differentiation:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top \\ D(\lambda^*)\mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{dz} \\ \mathbf{d}\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{dx} - \mathbf{dG}^\top \lambda^* \\ -D(\lambda^*)(\mathbf{dG}\mathbf{z}^* + \mathbf{dh}) \end{bmatrix}$$

49

49

Gradients of Collision Response

- Solution:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{x}} &= \mathbf{d}_\mathbf{z}^T \mathbf{W} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{G}} &= -D(\lambda^*)\mathbf{d}_\lambda \mathbf{z}^{*\top} - \lambda^* \mathbf{d}_\mathbf{z}^\top \\ \frac{\partial \mathcal{L}}{\partial \mathbf{h}} &= -\mathbf{d}_\lambda^T D(\lambda^*). \end{aligned}$$

where d_z and d_λ is provided by the linear equation:

$$\begin{bmatrix} \mathbf{W} & \mathbf{G}^\top D(\lambda^*) \\ \mathbf{G} & D(\mathbf{G}\mathbf{z}^* + \mathbf{h}) \end{bmatrix} \begin{bmatrix} \mathbf{d}_\mathbf{z} \\ \mathbf{d}_\lambda \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top \\ \mathbf{0} \end{bmatrix}$$

50

50

Acceleration of Gradient Computation

- Explicit solution of the linear equation:

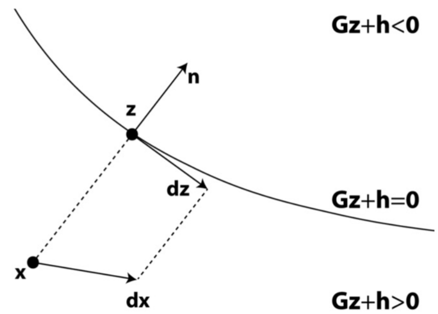
$$\mathbf{d}_{\mathbf{z}} = \sqrt{\mathbf{W}}^{-1}(\mathbf{I} - \mathbf{Q}\mathbf{Q}^\top)\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top$$

$$\mathbf{d}_{\lambda} = D(\lambda^*)^{-1}\mathbf{R}^{-1}\mathbf{Q}^\top\sqrt{\mathbf{W}}^{-1}\frac{\partial \mathcal{L}}{\partial \mathbf{z}}^\top$$

where Q and R is obtained from:

$$\sqrt{\mathbf{W}}^{-1}\mathbf{G}^\top = \mathbf{Q}\mathbf{R}$$

- Theoretical speedup: $O((n+m)^3) \rightarrow O(nm^2)$



51

51

Results

- Speed improvement in backpropagation.
- Scene setting: A large piece of cloth crumpled inside a pyramid.

Mesh	Baseline		Ours		Speedup	
Resolution	Matrix Size	Run Time (s)	Matrix Size	Run Time (s)	Matrix Size	Run Time
16x16	599 ± 76	0.33 ± 0.13	$\mathbf{66 \pm 26}$	$\mathbf{0.013 \pm 0.0019}$	8.9	25
32x32	1326 ± 23	1.2 ± 0.10	$\mathbf{97 \pm 24}$	$\mathbf{0.011 \pm 0.0023}$	13	112
64x64	2024 ± 274	4.6 ± 0.33	$\mathbf{242 \pm 47}$	$\mathbf{0.072 \pm 0.011}$	8.3	64

The runtime performance of gradient computation is significantly improved by up to *two orders of magnitude*

52

52

Results

- Application: Material estimation
- Scene setting: A piece of cloth hanging under gravity and a constant wind force.

Method	Runtime (sec/step/iter)	Density Error (%)	Non-Ln Streching Stiffness Error (%)	Ln Streching Stiffness Error (%)	Bending Stiffness Error (%)	Simulation Error (%)
Baseline	-	68 ± 46	74 ± 23	160 ± 119	**70 ± 42**	12 ± 3.0
L-BFGS [30]	2.89 ± 0.02	4.2 ± 5.6	64 ± 34	72 ± 90	**70 ± 43**	4.9 ± 3.3
Ours	**2.03 ± 0.06**	**1.8 ± 2.0**	**57 ± 29**	**45 ± 41**	77 ± 36	**1.6 ± 1.4**

Our method achieves the best runtime performance & the smallest error

53

53

Results

- Application: Motion control
- Scene setting: A piece of cloth being lifted and dropped to a basket.

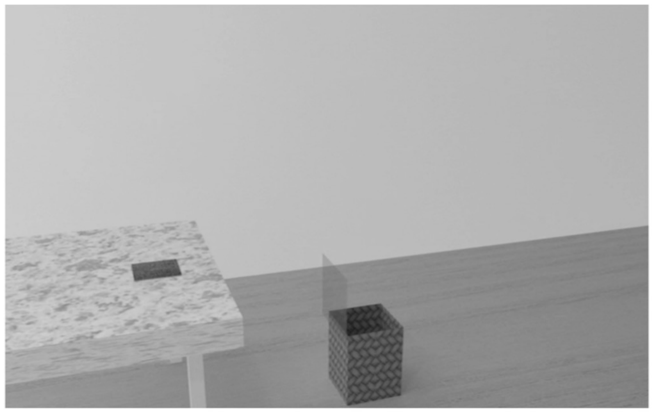
Method	Error (%)	Samples
Point Mass	111	–
PPO [18]	432	10,000
Ours	**17**	**53**
Ours+FC	39	108

Our method achieves the best performance with a much smaller number of simulations

54

54

Video Demos



Baseline - Treating as point mass

55

55

Summary

- A fully differentiable cloth simulation
  - Dynamic collision handling
  - Derivations of gradients using implicit differentiation
- Backpropagation acceleration by using QR decomposition to obtain the explicit solution
- Application examples: material estimation and motion control
  - Enabling ‘simulate-and-compare’ when embedding with deep network

56

56

Scalable Differentiable Physics for Learning and Control

Yi-Ling Qiao[1], Junbang Liang[1], Vladlen Koltun[2], and **Ming C. Lin[1]**

[1]University of Maryland at College Park

[2]Intel Labs

https://gamma.umd.edu/researchdirections/mlphysics/diffsim/

UNIVERSITY OF MARYLAND

ICML 2020

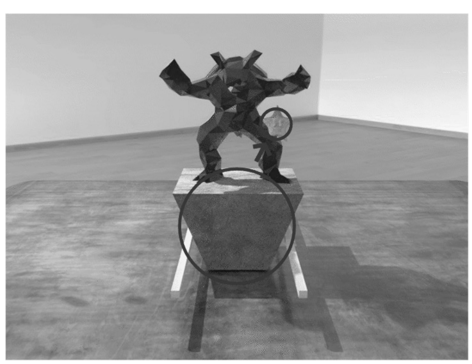
intel

57

57

Motivation

- Differentiable Physics Simulation as a Network Layer
  - Control of physical systems

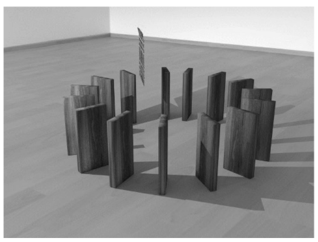


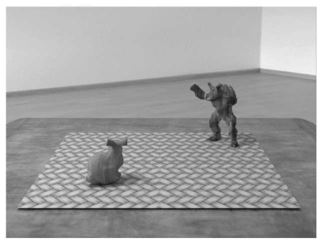
58

58

Motivation

- **Scalable** Differentiable Physics
  - Large **number** of interacting objects
  - Non-trivial **shapes**
  - Large variety of object **sizes**
  - Different physical properties/**material types**



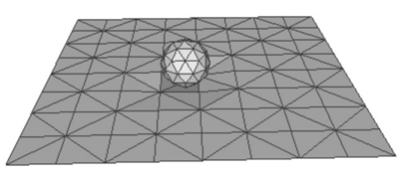


59

59

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth



63

63

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

64

64

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

65

65

Our Approach

1. Scalable
   - Localized collision handling - collisions are sparse
   - Fast differentiation - compute the gradients efficiently in large scenes
2. General
   - Modeling different objects - mesh scales well and can model complex objects
   - Interaction between different dynamics - coupling between rigid body and cloth

66

66

Mesh Simulation Flow

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v}_\mathbf{t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1} \mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$
5. $t = t + 1, \text{ goto } 2$

67

67

Mesh Simulation Flow: Backpropagation

Gradient computation available?

1. Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓ Handled by auto-differentiation
2. Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$?
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$ ✓ Handled by auto-differentiation
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$?
5. $t = t + 1, \text{ goto } 2$ ✓ Handled by auto-differentiation

68

68

Implicit Differentiation: Linear Solve

- Formulation: $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$
- Input: $\hat{\mathbf{M}}$ and $\mathbf{f}$. Output: $\mathbf{a}$
- Back propagation: use $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$ to compute $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$

  $\mathcal{L}$: the loss function.

69

69

Implicit Differentiation: Linear Solve

- Back propagation: use $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$ to compute $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{f}}$, where $\mathcal{L}$ is the loss function.
- Implicit differentiation form: $\partial \hat{\mathbf{M}}\mathbf{a} + \hat{\mathbf{M}}\partial \mathbf{a} = \partial \mathbf{f}$
- Solution: $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{M}}} = -\mathbf{d_a}\mathbf{z}^\top \quad \frac{\partial \mathcal{L}}{\partial \mathbf{f}} = \mathbf{d}_\mathbf{a}^\top$,

  where $\mathbf{d_a}$ is computed from $\hat{\mathbf{M}}^\top \mathbf{d_a} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}}^\top$, and $\mathbf{z}$ is the solution of $\hat{\mathbf{M}}\mathbf{a} = \mathbf{f}$.

70

70

Mesh Simulation Flow: Backpropagation

Gradient computation available?

1. $\text{Init } \mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓
2. $\text{Compute } \Delta \mathbf{v} \text{ from } \mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$ ✓
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$ ✓
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$ ✓ Using implicit differentiation! Algorithm-dependent
5. $t = t + 1, \text{ goto } 2$ ✓

71

71

Our Goal

- Scalability regarding resolution and shape
  - Mesh-based representation
- Scalability regarding material and quantity
  - Coupled physics between rigid body and deformable cloth
  - Localized collision handling

72

72

Key Contributions

- Adopting meshes as a general representation of objects.
- Leveraging the structure of contacts by grouping using localized impact zones
- An acceleration scheme that can handle the nonlinear constraints using implicit differentiation
- Demonstration examples on applications to learning and control scenarios, where the presented framework outperforms derivative-free and model-free base-lines by at least an order of magnitude.

73

73

Mesh Simulation Flow

1. $\text{Init } \mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$
2. $\text{Compute } \Delta \mathbf{v} \text{ from } \mathbf{x}_t, \mathbf{v_t}$
   - $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$
   - Newton's method
3. $\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$
4. $\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})$
5. $t = t + 1, \text{ goto } 2$

Dynamics Formulation

- Simulated objects: rigid body and deformable cloth
- Degree of freedom: 6 for rigid body, 3m for deformable cloth
- Stacked general coordinates: $\boldsymbol{q} = [\boldsymbol{q}_1^\top, \boldsymbol{q}_2^\top, ..., \boldsymbol{q}_n^\top]^\top$
   - $\boldsymbol{q}_k \in \mathbb{R}^6$ for rigid bodies
   - $\boldsymbol{q}_k \in \mathbb{R}^{3m_k}$ for clothes
- Dynamics:

$$\frac{d}{dt}\begin{pmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{pmatrix} = \begin{pmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{pmatrix}$$

k: object index, m: # of vertices of a cloth, n: # of objects

Collision Handling

- Global LCP solve for rigid bodies
  - Good at static contacts and static frictions
  - Difficult to couple with other materials
  - Slow
- Local constraint solver for clothes
  - Impulse-based solution: easy to couple between different materials
  - Solve within independent zones: faster computation
  - Unstable for large scale static contacts

76

76

Collision Handling

- Global LCP solve for rigid bodies
  - Good at static contacts and static frictions
  - Difficult to couple with other materials
  - Slow
- Local constraint solver for clothes
  - Impulse-based solution: easy to couple between different materials
  - Solve within independent zones: faster computation
  - Unstable for large scale static contacts

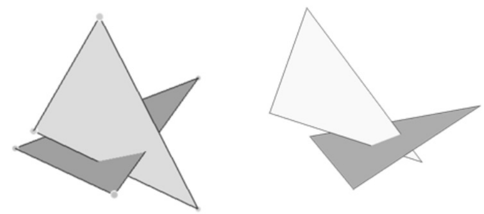
77

77

Local Collision Handling

Impact zone model (Harmon et al. 2008)

- Constraints built upon impacts
- Linear w.r.t. vertex positions
  - $C_{ee} = \boldsymbol{n} \cdot [(\alpha_3 \boldsymbol{x}_3 + \alpha_4 \boldsymbol{x}_4) - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2)]$
    $C_{vf} = \boldsymbol{n} \cdot [\boldsymbol{x}_4 - (\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \alpha_3 \boldsymbol{x}_3)]$



EE Impact VF Impact

α_i: barycentric coordinates of each vertex at collision

78

78

Local Collision Handling

Impact zone model (Harmon et al. 2008)

- Introduce minimum energy: QP formulation

$$\begin{aligned} \underset{\mathbf{x}'}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top \mathbf{M}(\mathbf{x} - \mathbf{x}') \\ \text{subject to} \quad & \mathbf{G}\mathbf{x}' + \mathbf{h} \leq \mathbf{0} \end{aligned}$$

composed of: $C_{vf} = n \cdot [x_4 - (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3)] < 0$

- Solve the optimization for each `connected component' of impacts

$\mathbf{M}$: mass matrix, $\mathbf{G}$ and $\mathbf{h}$: constraint matrix and vector

79

79

Local Collision Handling

Coupling with rigid bodies

- Treating one rigid body as one node
- Nonlinear constraint for optimization

$$\begin{aligned} \underset{\mathbf{q}'}{\text{minimize}} \quad & \frac{1}{2}(\mathbf{q}-\mathbf{q}')^\top \hat{\mathbf{M}}(\mathbf{q}-\mathbf{q}') \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{f}(\mathbf{q}') + \mathbf{h} \leq \mathbf{0} \end{aligned}$$

$\mathbf{f}(\cdot)$: a function mapping from general coordinates q to some endpoint $\mathbf{x}$

80

80

Mesh Simulation Flow: Backpropagation

Gradient computation available?

Init $\mathbf{x}_0, \mathbf{v}_0, \Delta t, t = 0$ ✓

Compute $\Delta \mathbf{v}$ from $\mathbf{x}_t, \mathbf{v_t}$

- $\Delta \mathbf{v} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}_{t+1}, \mathbf{v}_{t+1}) * \Delta t$ ✓
- Newton's method

$\tilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{v}}_{t+1} * \Delta t, \tilde{\mathbf{v}}_{t+1} = \mathbf{v}_t + \Delta \mathbf{v}$ ✓

$\boxed{\mathbf{x}_{t+1}, \mathbf{v}_{t+1} = \text{resolve\_collision}(\tilde{\mathbf{x}}_{t+1}, \tilde{\mathbf{v}}_{t+1})}$ ✓

1. $t = t + 1, \ \text{goto } 2$ ✓

81

81

Gradient Computation for Collision

- Optimization problem:

$$\begin{aligned} &\underset{\mathbf{q}'}{\text{minimize}} && \frac{1}{2}(\mathbf{q}-\mathbf{q}')^\top \hat{\mathbf{M}}(\mathbf{q}-\mathbf{q}') \\ &\text{subject to} && \mathbf{G}\boldsymbol{f}(\mathbf{q}') + \mathbf{h} \leq \mathbf{0} \end{aligned}$$

- Karush-Kuhn-Tucker (KKT) condition at optimal point:
  - Stationarity (gradient=0), and complementary slackness

$$\hat{\mathbf{M}}\mathbf{z}^* - \hat{\mathbf{M}}\mathbf{q} + \nabla \boldsymbol{f}^\top \mathbf{G}^\top \lambda^* = 0$$

$$D(\lambda^*)(\mathbf{G}\boldsymbol{f}(\mathbf{z}^*) + \mathbf{h}) = 0$$

82

82

Results

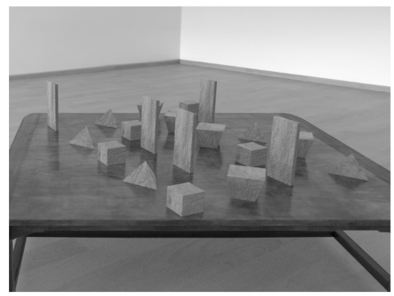
1. Scalable
   - Large number of objects - Linear w.r.t. number of objects
   - High resolution
2. General
   - Complex objects
   - Two-way Coupling - Constant
3. Applications
   - Inverse Problem - Faster than derivative-free methods
   - Control - Faster than RL

86

86

Results - Scalable

- Scale the number of objects
- Scene setting: A bunch of (20 - 1000) objects collide with the ground.
  - Methods: Ours vs. ChainQueen[8] (on CPU, for 2 second)
  - Scale the number of objects, while keeping the density of collisions and objects
  - When the number of object scales from 20 to 200, the grid size of ChainQueen[8] scales from 64 to 640

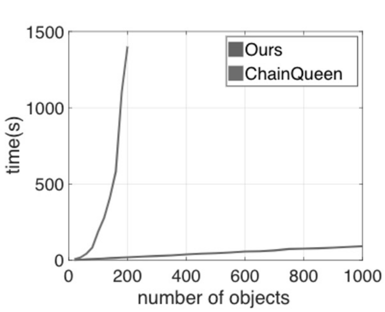


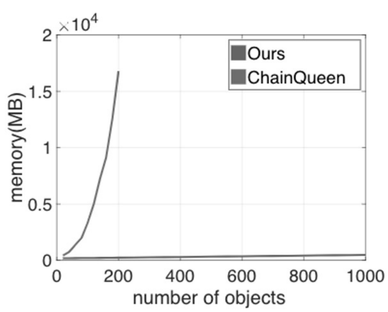
87

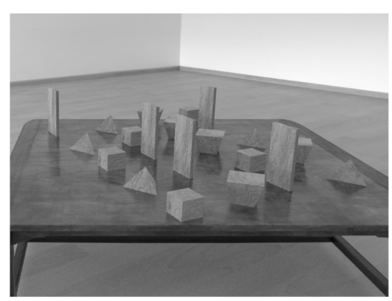
87

Results - Scalable

- Scale the number of objects
- Scene setting: A bunch of (20 - 1000) objects collide with the ground.
- **Our method scales well (linearly) in large scenes with big number of objects.**





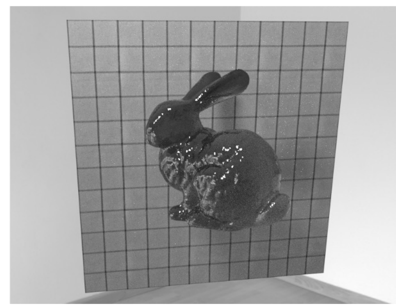


88

88

Results - Scalable

- Scale the resolution
- Scene setting: A bunny and a piece of cloth. Vary the relative sizes of cloth.
  - Methods: Ours vs. ChainQueen[8] (on CPU, for 2 second)
  - The relative size of two cloths: n:1.
  - n scales from 1 to 10.
  - The grid size of ChainQueen[8] scales from 64 to 640

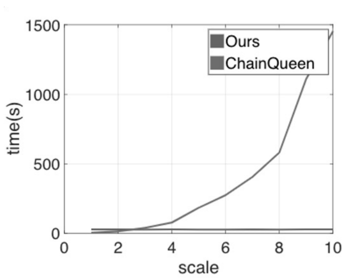


89

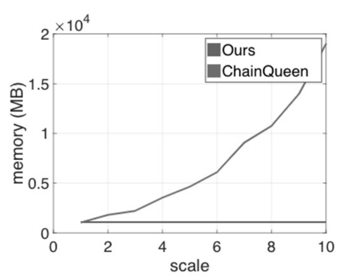
89

Results - Scalable

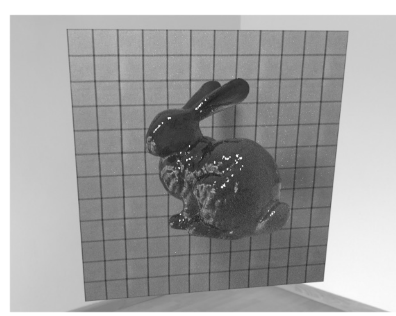
- Scale the resolution
- Scene setting: A bunny and two piece of cloths. Vary the relative sizes of clothes.
- **Our method runs in constant time in different resolutions.**



(b) Running time



(c) Memory consumption



90

90

Results - Inverse Problem

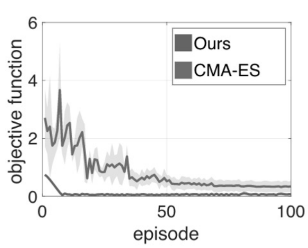
- Learn the trajectory
- Scene setting: Compute the force on the marble in each step to drive it to a target point.
  - Methods: Ours vs. CMA-ES
  - The combined force vector has 100 dimensions
  - Object function: the distance to target + norm of the force vector

91

91

Results - Inverse Problem

- Learn the trajectory
- Scene setting: Compute the force on the marble in each step to drive it to a target point.
- **Our method converges more than 10x faster than CMA-ES.**



(b) Objective function

92

92

Results - Control

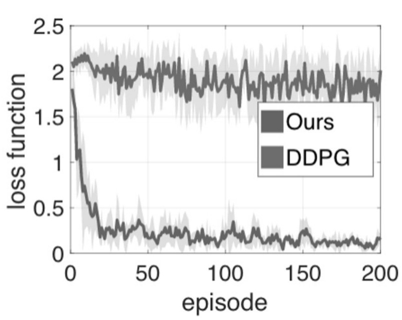
- Manipulation
- Scene setting: Control the motion of a pair of parallel grippers, to move an object towards a random target in 2 second
    - Methods: Ours vs. DDPG[6]
    - Fixed initial position and random target.
    - Loss is the L2 distance from the target to the current position. Reward = -1 * Loss
    - Observation: [x_now - x_target, v_now, time, reward]
    - Action: [v_next]

93

93

Results - Control

- Manipulation
- Scene setting: Control the motion of a pair of parallel grippers, to move an object towards a random target
- **Our method converges much faster than RL**

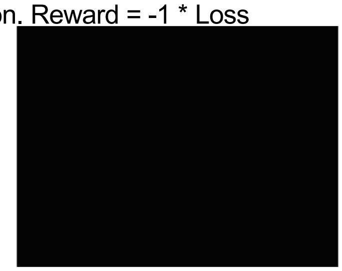


94

94

Results - Control

- Motion control
- Scene setting: Control the <u>motion of four handles</u> on a cloth, to move the cube towards a random target
  - Methods: Ours, DDPG[6]
  - Fixed initial position and random target.
  - Loss is the L2 distance from the target to the current position. Reward = -1 * Loss
  - Observation: [x_now - x_target, v_now, time, reward]
  - Action: [v_next]

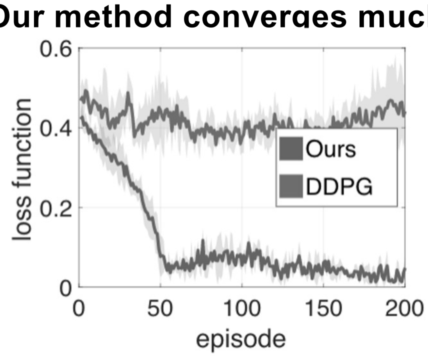


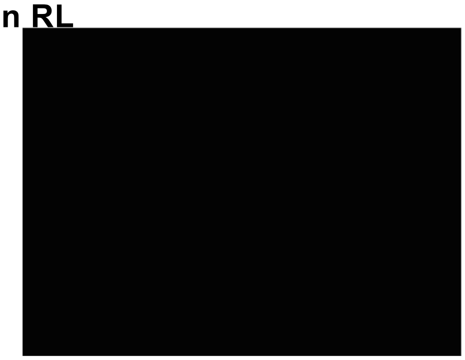
95

95

Results - Control

- Motion control
- Scene setting: Control the <u>motion of four handles</u> on a cloth, to move the cube towards a random target
- **Our method converges much faster than RL**



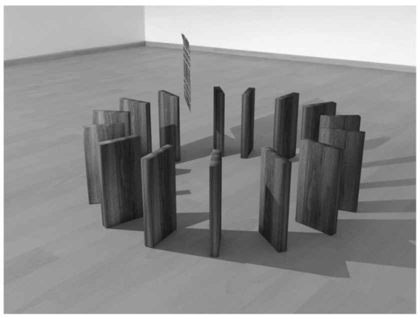


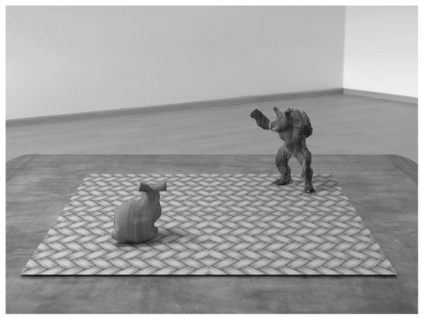
96

96

Results - General

- Two-way coupling between cloth and rigid body
- Scene setting: Cloth & dominos



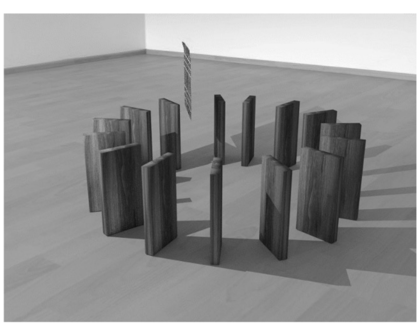


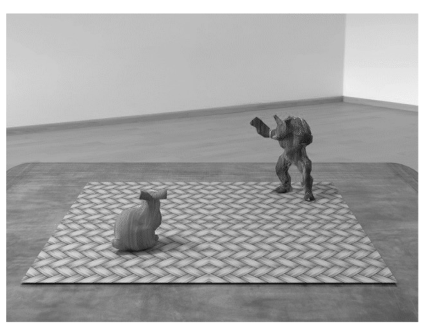
97

97

Conclusion

- A method for scalable and general differentiable physics
- Future work
  - More general dynamics
  - More Application





98

98

Video Demonstration

Scalable Differentiable Physics for
Learning and Control

Submission ID: 15

99

99

Efficient Differentiable Articulated Dynamics

Yi-Ling Qiao*, Junbang Liang*, Vladlen Koltun, and Ming Lin*

* UNIVERSITY OF MARYLAND

intel

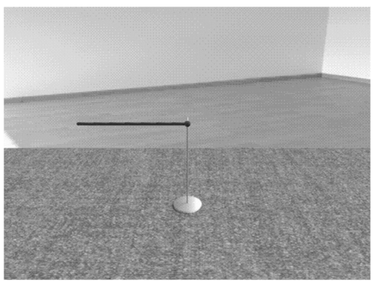
Code & data: https://github.com/YilingQiao/diffarticulated ICML 2021

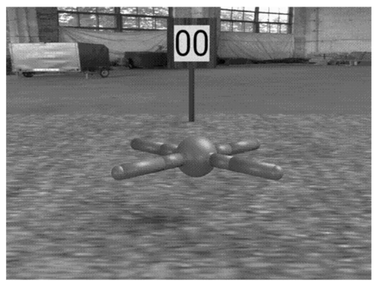
100

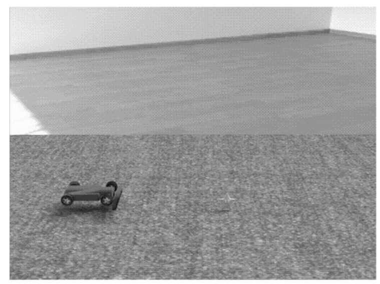
100

Motivation

- Differentiable articulated body simulation as a network layer
  - Control physical systems
  - Enhance reinforcement learning
  - Estimate physics parameters



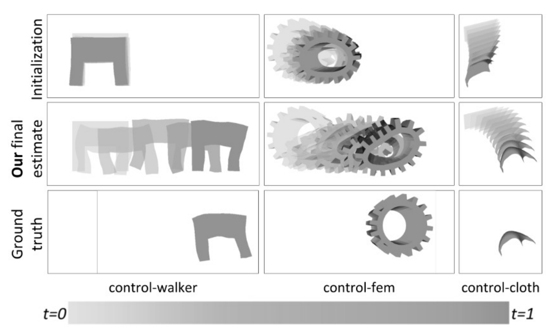




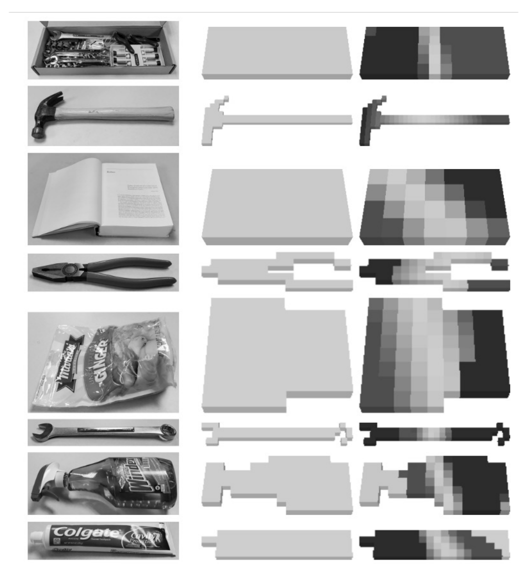
101

101

Applications



[2] Murthy et al. (2021)

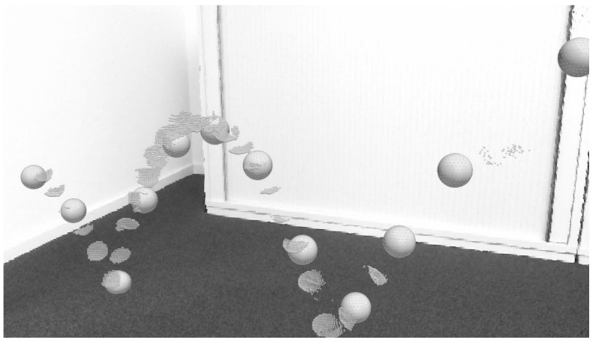


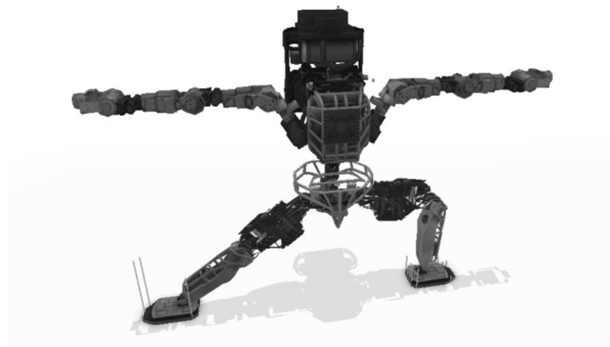
[5] Song et al. (2020)

102

102

Related/concurrent work





[1] Geilinger et al. (2020)

[7] Werling et al. (2021)

103

103

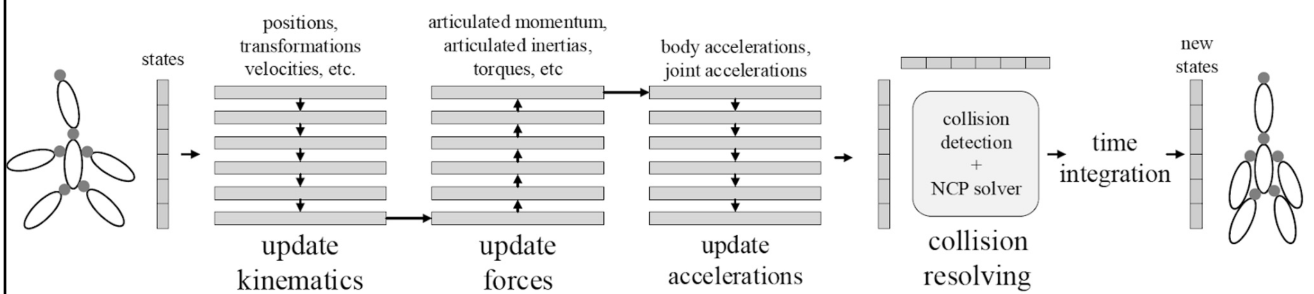
Key Contributions

- Derive the adjoint formulations for the entire articulated body simulation workflow, enabling a 10x acceleration over autodiff tools
- Adapt the checkpointing method to the structure of articulated body simulation to reduce memory consumption by 100x, making stable collection of extended experiences feasible
- Introduce two general schemes for accelerating reinforcement learning using differentiable physics
- Demonstrate the utility of differentiable simulation of articulated bodies in motion control and parameter estimation, enhancing performance in these scenarios by more than an order of magnitude

104

104

Workflow of one simulation step

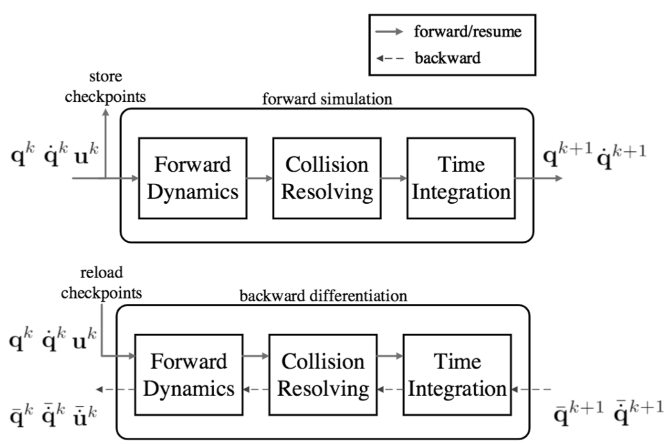


105

105

Checkpointing

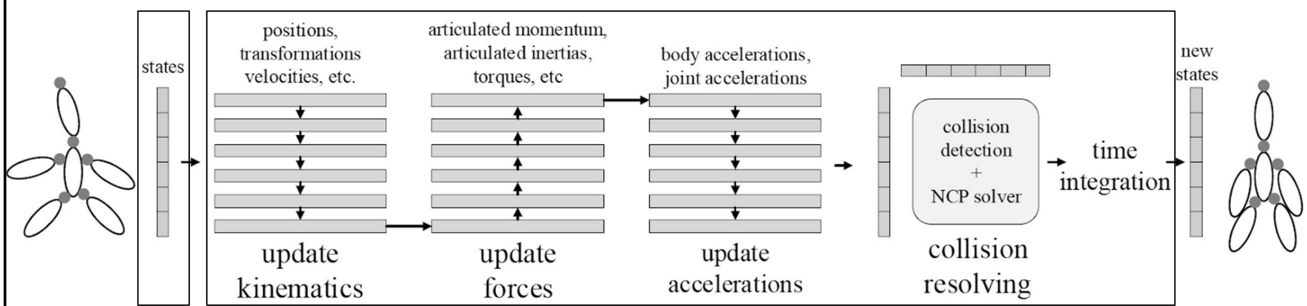
Forward and backward workflow with checkpointing scheme



106

106

Checkpointing



checkpoints

Intermediate variables

107

107

Application with Reinforcement Learning

- Sample enhancement
  - Increase sample efficiency
  - Faster convergence
- Policy enhancement
  - Update the policy using analytic gradients
  - Better scalability in high dimensionality

108

108

Sample Enhancement

- Idea: Use simulation gradients to generate extra nearby examples
- Point sample → patch sample
  - Faster convergence

$$a_k = a_0 + \Delta a_k$$
$$s'_k = s'_0 + \boxed{\frac{\partial s'_0}{\partial a_0}} \Delta a_k$$
$$r_k = r_0 + \boxed{\frac{\partial r_0}{\partial a_0}} \Delta a_k$$

Enabled by differentiable simulation!

a: action
s: observation
s': next-step observation
r: reward

109

109

Policy Enhancement

- Idea: Use simulation gradients to compute better policy gradients
- Use one-step rollout to approximate the action gradients

$$\mathcal{L}_\mu = -Q(s, \mu(s)) + Z$$

Soft Actor-Critic

$$\frac{\partial Q(s,a)}{\partial a} = \boxed{\frac{\partial r}{\partial a}} + \gamma \frac{\partial Q(s', \mu(s'))}{\partial s'} \boxed{\frac{\partial s'}{\partial a}}$$
$$\mathcal{L}'_\mu = -\frac{\partial Q(s,a)}{\partial a} \mu(s) + Z$$

Ours

Enabled by differentiable simulation!

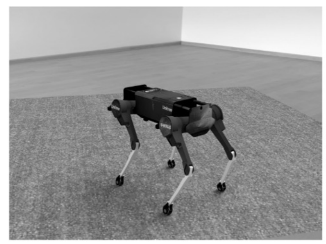
a: action
s: observation
s': next-step observation
r: reward
Q: critic network
μ: policy network
Z: regularization term

110

110

Results - Performance

- Compare the runtime and memory usage.
- Scene: One Laikago released from the air and hitting the ground
  - Scale the simulation length: 50, 100, 500, 1000, 5000 steps
- Comparisons:
  - Use autodiff tools in the same simulation pipeline



111

111

Results - Performance

- Compare the runtime and memory usage.
- Scene: A Laikago released from the air and hitting the ground
- Our method has the highest speed and the lowest memory usage
  - x10 faster than autodiff tools with 1% of memory usage

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

Forward simulation time (ms) per step

steps	50	100	500	1000	5000
ADF	25.7	25.5	25.1	32.1	58.4
Ceres	27.2	27.5	27.2	34.0	58.2
CppAD	2.4	2.4	2.3	2.3	4.5
JAX	53.3	46.1	43.1	42.7	42.3
PyTorch	195.6	192.2	199.2	192.8	N/A
Ours	**0.3**	**0.3**	**0.2**	**0.2**	**0.2**

Peak Memory (MB)

112

112

Policy Enhancement

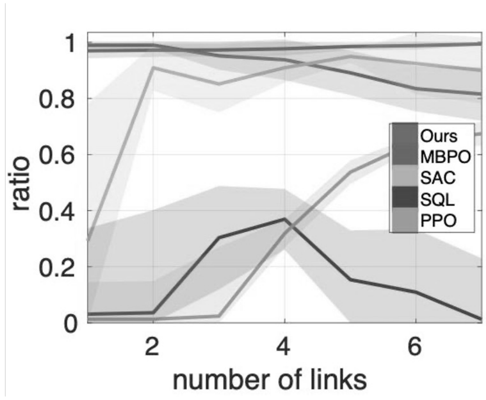
- Scenario: N-link pendulum
- Objective: reaching the highest point within 100 frames
- Reward
  - -dist_to_target^2
- Baseline: MBPO, SAC, SQL, PPO
- Number of links: 1-7
- Number of training epochs: 100 * n_links
  - Samples per epoch: 100

113

113

Policy Enhancement

- Test metric: Best relative reward
  - Absolute reward / maximum possible reward (reaching exactly the target)



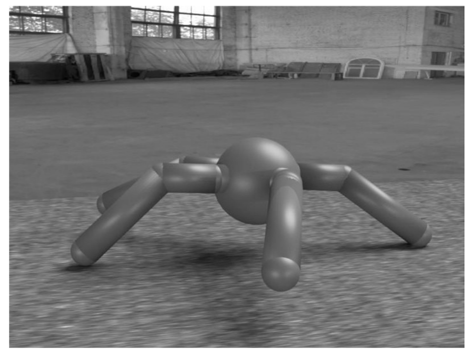
Our method scales with increasing system complexity

114

114

Sample Enhancement

- Scenario: Mujoco Ant
- Objective: walking towards +x axis
- Reward
  - v_x - sum(action^2)
- Baseline: MBPO, SAC, SQL, PPO
- Number of training epochs: 100
  - Samples per epoch: 1000

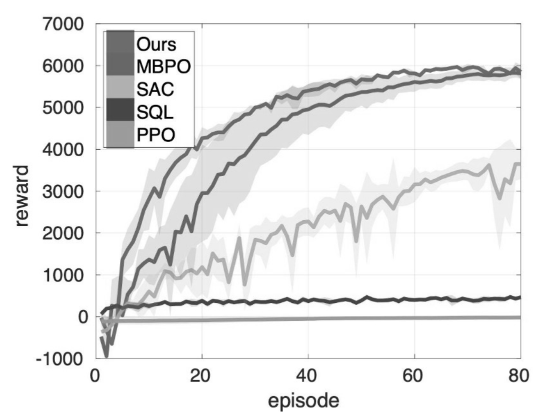


115

115

Sample Enhancement

- Test metric:
  - Maximum (absolute) reward



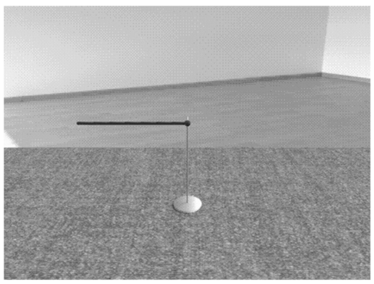
Our method achieves the same best reward and converges faster

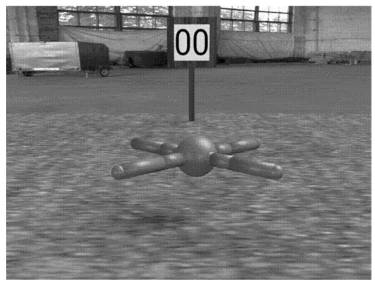
116

116

Motivation

- Differentiable articulated body simulation as a network layer
  - Control physical systems
  - Enhance reinforcement learning
  - Estimate physics parameters





117

117

Video Demonstration

Efficient Differentiable Simulation of Articulated Bodies

Submission ID: 118

118

118

Differentiable Simulation of Soft Multi-body Systems

Yi-Ling Qiao*, Junbang Liang*, Vladlen Koltun, and Ming Lin*

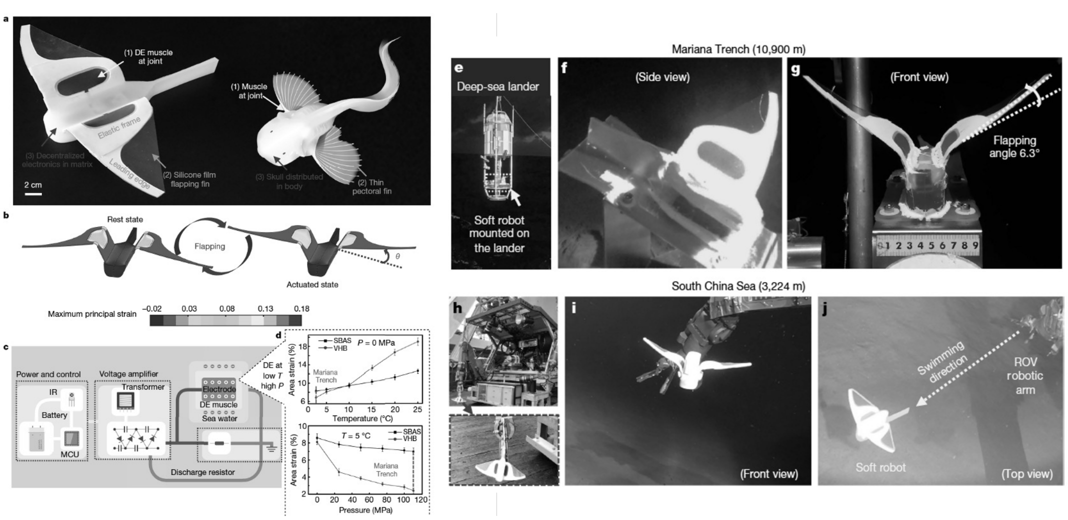
*UNIVERSITY OF MARYLAND intel

119

119

Motivation

- Self-powered soft robot in the Mariana Trench

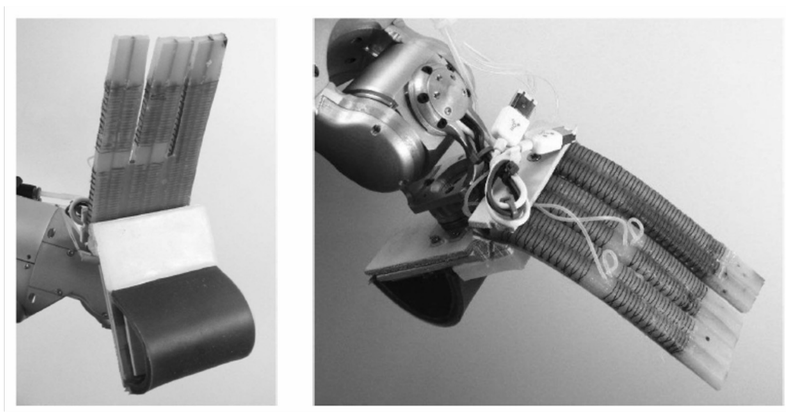


120

120

Motivation

- A Compliant Hand Based on a Novel Pneumatic Actuator.

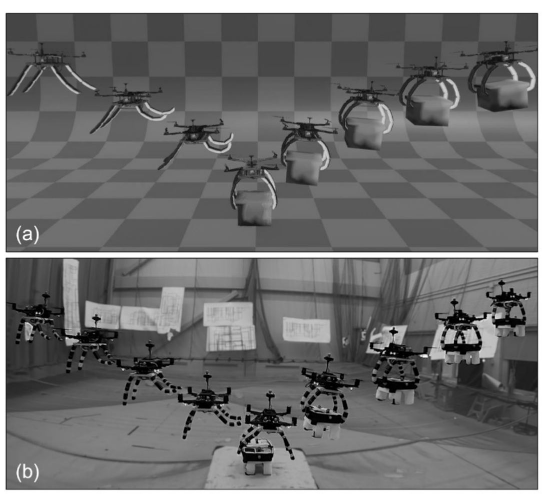


121

121

Motivation

- Dynamic Grasping with a “Soft” Drone



122

122

OBJECTIVE

- ***Differentiable Physics Simulator*** to support different scenarios
  - Complex Contact
  - Embedded Skeleton
  - Joint, muscle, and pneumatic actuators

123

123

Key Contributions

- A ***top-down matrix assembly*** algorithm within **Projective Dynamics** to make soft-body dynamics compatible with reduced-coordinate articulated system
- An **extended and generalized dry friction model** for soft solids with a new matrix splitting strategy to stabilize the solver
- **Analytical models of muscles, joint torques, and pneumatic actuators** to enable more realistic and stable simulation results
- A **unified differentiable framework** that incorporates skeletons, contact, and actuators to enable gradient computation for learning and optimization
- **Experimental validation** demonstrating that differentiable physics accelerates system identification and motion control with soft articulated bodies up to orders of magnitude

124

124

Background

Projective dynamics

Implicit Euler : $$\mathbf{M}(\mathbf{q}_{n+1} - \mathbf{q}_n - h\mathbf{v}_n) = h^2(\nabla E(\mathbf{q}_{n+1}) + \mathbf{f}_{ext})$$

Solve: $$\mathbf{q}_{n+1} = \arg\min_{\mathbf{q}} \frac{1}{2h^2}(\mathbf{q} - \mathbf{s}_n)^\top \mathbf{M}(\mathbf{q} - \mathbf{s}_n) + E(\mathbf{q})$$

Local step: $$E(\mathbf{q}) = \sum_i \frac{\omega_i}{2} \|\mathbf{G}_i \mathbf{q} - \mathbf{p}_i\|_F^2$$

Global step: $$\mathbf{q}_{n+1} = \arg\min_{\mathbf{q}} \frac{1}{2}\mathbf{q}^\top \left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{q} + \mathbf{q}^\top \left(\frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}\right)$$

130

130

Method - rigid bodies

Vertices on rigid bodies : $$\mathbf{q}_k = \mathbf{Q}\mathbf{T}_k^r \mathbf{V}_k$$

Linearize: $$\mathbf{q}_k^{i+1} = \mathbf{q}_k^i + \Delta\mathbf{q}_k^i = \mathbf{q}_k^i + \frac{\partial \mathbf{q}_k^i}{\partial \mathbf{z}_k}\Delta\mathbf{z}_k^i \qquad \mathbf{B} = \frac{\partial \mathbf{q}^i}{\partial \mathbf{z}}$$

New global step: $$\Delta\mathbf{z}^i = \arg\min_{\Delta\mathbf{z}} \frac{1}{2}\Delta\mathbf{z}^\top \mathbf{B}^\top \left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{B}\Delta\mathbf{z} + \Delta\mathbf{z}^\top \mathbf{B}^\top \left(\left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{q}^i - \left(\frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}\right)\right)$$

Local step:
$$\mathbf{T}_k = \begin{bmatrix} \mathbf{I} + \omega_k^{i*} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{T}_k^r + \begin{bmatrix} \mathbf{0} & \mathbf{l}_k^i \\ \mathbf{0} & 0 \end{bmatrix} = \mathbf{U}\Sigma\mathbf{V}^\top$$

$$\mathbf{T}_k^{r\prime} = \mathbf{U}\mathbf{V}^\top$$

131

131

Method - Articulated body

Skeleton tree: $\mathbf{T}_k^r = \prod_u \mathbf{A}_u$ A is the local transformation matrix

Jacobian:

$$\mathbf{B}_{u,v} = \frac{\partial \mathbf{T}_u^r \mathbf{V}_u}{\partial \mathbf{z}_v} = \mathbf{Q}\mathbf{P}_v \frac{\partial \mathbf{A}_v}{\partial \mathbf{z}_v} \mathbf{S}_{v,u} \mathbf{V}_u$$

Compute recursively:

$$\mathbf{P}_v = \mathbf{P}_{v'} \mathbf{A}_{v'}$$
$$\mathbf{S}_{v',u} = \mathbf{A}_v \mathbf{S}_{v,u}$$

P is the prefix product
S is the suffix product

132

Method - Articulated body

Algorithm 2 Matrix Assembly for the Articulated System

Input: tree link u
Compute $\mathbf{P}_u$ using Eq. 16
$v \leftarrow u$
while v is not root **do**
    Compute $\mathbf{S}_{v,u}$ using Eq. 17
    Compute $\mathbf{B}_{u,v}$ using Eq. 13
    $v \leftarrow$ parent(v)
end while
Compute $\mathbf{B}_{u,root}$ using Eq. 15
for s **in** descendants(u) **do**
    Solve link s recursively
end for

133

Method - Articulated body

Rotational joint. This joint is characterized by a rotation axis $\mathbf{n}$ and the angle θ. Its transformation matrix and the Jacobian are:

$$\mathbf{A}^r = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \frac{\partial \mathbf{A}^r}{\partial \theta} = \begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \theta} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \tag{18}$$

$$\mathbf{R} = \cos\theta \cdot \mathbf{I} + \sin\theta [\mathbf{n}]_\times + (1 - \cos\theta)\mathbf{n}\mathbf{n}^\top \tag{19}$$

$$\frac{\partial \mathbf{R}}{\partial \theta} = -\sin\theta \cdot \mathbf{I} + \cos\theta [\mathbf{n}]_\times + \sin\theta \mathbf{n}\mathbf{n}^\top \tag{20}$$

The local update of the rotational joint is given by:

$$\theta^{i+1} = \arctan(\sin\theta^i + \cos\theta^i \Delta\theta^i, \cos\theta^i - \sin\theta^i \Delta\theta^i) \tag{21}$$

134

134

Method - Articulated body

Prismatic joint. This joint is characterized by a prismatic axis $\mathbf{u}$ and the scale l. Its transformation matrix and the Jacobian are:

$$\mathbf{A}^p = \begin{bmatrix} \mathbf{I} & l\mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \frac{\partial \mathbf{A}^p}{\partial l} = \begin{bmatrix} \mathbf{0} & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix} \tag{22}$$

(23)

The local update of the prismatic joint is simply addition:

$$l^{i+1} = l^i + \Delta l^i \tag{24}$$

135

135

Method - Actuation - Joint Torque

$$\Delta \mathbf{z}^i = \arg\min_{\Delta \mathbf{z}} \frac{1}{2} \Delta \mathbf{z}^\top \mathbf{B}^\top \left(\frac{\mathbf{M}}{h^2} + \mathbf{L} \right) \mathbf{B} \Delta \mathbf{z} + \Delta \mathbf{z}^\top \mathbf{B}^\top \left(\left(\frac{\mathbf{M}}{h^2} + \mathbf{L} \right) \mathbf{q}^i - \left(\frac{\mathbf{M}}{h^2} \mathbf{s}_n + \mathbf{J}\mathbf{p} \right) \right) \quad (8)$$

Solve a linear system:

$$\begin{bmatrix} \mathbf{H}_d & \mathbf{H}_c^\top \\ \mathbf{H}_c & \mathbf{H}_r \end{bmatrix} \begin{bmatrix} \Delta \mathbf{z}_d^i \\ \Delta \mathbf{z}_r^i \end{bmatrix} = \begin{bmatrix} \mathbf{k}_d \\ \mathbf{k}_r \end{bmatrix}$$

Torques can be added to K_r directly

136

Method - Actuation - Pneumatic

Pneumatic actuator. We use co-rotational elastic strain energy model for tetrahedral cells. For a pneumatic cell with activation level a, the energy is computed as

$$\Psi_{pneumatic}(\mathbf{F}, a) = \frac{k_p}{2} \|\mathbf{F} - \mathbf{R}(a)\|^2 \quad (27)$$

where the SVD decomposition of the deformation gradient is $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{R}(a) = \mathbf{U}\Sigma^*\mathbf{V}^T$, $\Sigma^* = \mathbf{D} + \Sigma$, and $\mathbf{D}$ is computed by

$$\arg\min_{\mathbf{D}} \|\mathbf{D}\|_2^2, s.t. \prod_i (\Sigma_i i + \mathbf{D}_i) = a \quad (28)$$

137

Method - Actuation - Muscle

Muscle actuator. We use the muscle actuators described in [49]. Muscles are modeled as fibers in the soft bodies, and the forces are computed as $\mathbf{f}_{muscle}(a) = -f_{muscle}(a)\mathbf{m}$, where $a \in [0, 1]$ is the activation level, $\mathbf{m}$ is the direction of fiber. To achieve this force, a strain energy model [32] is used, $E_{muscle} = \mathbf{V}_{muscle}\Psi_{muscle}(\mathbf{F}, e)$, where $\Psi_{muscle}(\mathbf{F}, a) = \frac{k_m}{2}\|(1-r)\mathbf{F}\mathbf{m}\|$, k_m is the stiffness, $r = \frac{1-a}{l}$ is the projection of the cord segment, $l = \|\mathbf{F}\mathbf{m}\|$ is the stretch factor.

138

138

Method - Contact

Original global step:

$$\left(\frac{\mathbf{M}}{h^2} + \mathbf{L}\right)\mathbf{q}_{n+1} = \frac{\mathbf{M}}{h^2}\mathbf{s}_n + \mathbf{J}\mathbf{p}$$

Convert to velocity space:

$$\overbrace{\mathbf{M}\mathbf{v}^{i+1}}^{\text{Adjusted momentum}} = \overbrace{\mathbf{f} - h^2\mathbf{L}\mathbf{v}^i}^{\text{Current momentum}} + \xi^i$$

$$\mathbf{f} = \mathbf{M}\mathbf{s}_n - (\mathbf{M} + h^2\mathbf{L})\mathbf{q}_n + h^2\mathbf{J}\mathbf{p}$$

Contact handling:

ξ^i Depends on the relative velocities/momentums of collided vertices

140

140

Method - Contact

- Friction law enforcement
  - The new impulse is added to the individual vertex
  - Iteratively resolved until converged
- Convergence
  - Not guaranteed
  - Depends on **M** and **L** if **f** and ξ are fixed
- Applicability to soft bodies
  - **L** too large compared to **M**
  - Unstable solve

$$\overbrace{\mathbf{M}\mathbf{v}^{i+1}}^{\text{Adjusted momentum}} = \overbrace{\mathbf{f} - h^2\mathbf{L}\mathbf{v}^{i}}^{\text{Current momentum}} + \xi^{i}$$

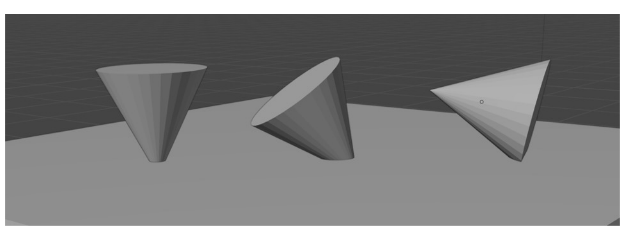
$$\mathbf{f} = \mathbf{M}\mathbf{s}_n - (\mathbf{M} + h^2\mathbf{L})\mathbf{q}_n + h^2\mathbf{J}\mathbf{p}$$

141

141

Method - Contact

- Improvement
  - Move the diagonals of **L** to the left!
  - $(\mathbf{M} + h^2\mathbf{D})\mathbf{v}^{i+1} = \mathbf{f} - h^2(\mathbf{L} - \mathbf{D})\mathbf{v}^{i} + \xi^{i}$
  - When **f** and ξ are fixed, the improved method is guaranteed to converge
- Contact detection
  - Continuous collision detection
  - Grouped vertex-face collision handling
    - Contact forces need to be computed jointly



142

142