PlasticityNet: Learning to Simulate Metal, Sand, and Snow for Optimization Time Integration

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we propose a neural network-based approach for learning to represent the behavior of plastic solid materials ranging from rubber and metal to sand and snow. Unlike elastic forces such as spring forces, these plastic forces do not result from the positional gradient of any potential energy, imposing great challenges on the stability and flexibility of their simulation. Our method effectively resolves this issue by learning a generalizable plastic energy whose derivative closely matches the analytical behavior of plastic forces. Our method, for the first time, enables the simulation of a wide range of arbitrary elasticity-plasticity combinations using time step-independent, unconditionally stable optimization-based time integrators. We demonstrate the efficacy of our method by learning and producing challenging 2D and 3D effects of metal, sand, and snow with complex dynamics.

1 Introduction

Combining machine learning with physical simulations has recently attracted a lot of attention. A vast amount of existing research adopts an end-to-end approach, where the specific underlying computational physics system is treated as a black box [39, 35]. Harnessing the power of neural networks, this research has been successfully applied in computer animation [10], multibody systems [3, 6, 52], human musculature simulation [17], computational fluid dynamics [4, 12], and non-linear continuum mechanics [5]. An alternative direction is represented by physics-informed neural networks (PINN) [37, 18], where in its original form, the residual of a partial differential equation is directly used as the loss function so that the network training is a physics-aware learning process. PINN becomes powerful when the design space of the input to the network can be parameterized, which accelerates both the roll-out and the inverse optimization process [44]. Another noteworthy category is learning the physical modeling where the machine can either help increase the model resolution in a coarser grid [21], inject nonlinearity to a linear model [32], or apply a learnable model reduction to reduce the system degrees-of-freedom (DOF) for acceleration [40, 41].

Despite its great success, training a neural network to replace a traditional simulator is not always the preferred choice. This is partially due to the challenges in the trained model's generality and portability. For example, a trained model on a particle-based deformable body solver (such as the Material Point Method (MPM [16]) cannot be directly applied to the mesh-based Finite Element Method (FEM) [42], while in traditional continuum mechanics, the constitutive model that describes the relationship between force and deformation is an independent module from the underlying geometric description or simulation scheme. Indeed, by simply switching the constitutive model and applying minor changes to the existing and general simulation pipeline, a wide range of materials can be simulated in the same framework, ranging from sand [36, 9, 20, 45] and metal [33], to snow [14, 50, 31] and glacier [51].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Many elastic materials, including those represented by mass-spring systems [2] and common *hyperelastic* solids [43], are usually governed by analytical elastic potential energy functions in terms of the deformation. These models are well fitted to experiments and proven to be simple, accurate, and predictive. Although most of these energy functions are highly nonlinear and non-convex, reformulating the dynamic simulation process as a numerical optimization problem and solving it using line searches can guarantee global convergence to a solution [25]. Beyond hyperelasticity, plasticity is much more challenging. The mechanical response of plastic materials imposes extra difficulties in the implementation as it is path-dependant and non-smooth. One common handling of plasticity is the return mapping algorithm, which applies the effects of plastic deformation to the elastic forces. However, this leads to asymmetrical force derivatives, which eliminate the possibility of integrating the plasticity into the energy function in a single optimization and complicates the pipeline. In the recent work of Energetic Consistent Inelasticity (ECI) [30], the plasticity is analytically modeled as an energy functional, and the simulation can be formulated as an optimization problem just like simulating pure elastic materials. However, their analytical derivation only works for St.Venant-Kirchhoff (StVK) elasticity with the von-Mises plasticity.

In this work, we propose PlasticityNet, a neural network-based approach for learning an energy-based force that locally approximates elastic forces with plasticity models and is compatible with optimization time integrators. PlasticityNet framework supports any combinations of elastic models and plastic models and works with both MPM and FEM discretizations. With optimization time integrators, we demonstrate that our framework can simulate vast types of plasticities, such as metal, sand, and snow, with large time step sizes.

2 Related work

Classic Plasticity Models The classic plastic models utilized the geometry information of the plasticity and are available for many applications. In the computer graphics community, researchers have followed mechanical literature on the Drucker-Prager elastoplasticity model [36, 9], and developed particle-based simulations of dry [20] and wet [45] sand. Extending a similar Cam-Clay plasticity model, snow avalanches [14, 31], glacier calving [51] and food fracturing [50] are also captured with high visual plausibility as well as physical accuracy. For metals and dough-like materials, the von-Mises plasticity model [33] is usually adopted, while [49, 15] presented its anisotropic extensions. Still, the implementation of these models in modern, optimization-based simulators is cumbersome due to the non-integrable forces. Recently, [30] proposed an elastoplastic energy of von-Mises plasticity under StVK elasticity for optimization time integrator, which can be viewed as a special-case analytical solution to our framework under the same combination of elasticity and plasticity. But our framework works for arbitrary combinations.

Data-Driven Plasticity Models The machine learning approach has been used to find new plastic models using large sets of measurements and parameters, outperforming many long-standing hand-crafted models. The macro-level constitutive relationship is learned from the results of the micro-level simulations [34, 38]. A similar approach is applied in [46] to learn anisotropic hyperelasticity, where additional geometrical information is included in the input. PINN can also be applied in plastic model finding from experimental measurements [1, 47, 22], where the loss includes the stress and Hessian, to infer stress with more accuracy in the implicit simulators. However, there does not exist any prior work, to the best of the author's knowledge, that tried to find variational form for arbitrary plasticity model.

Optimization Time Integration The optimization time integrators have advantages in terms of stability under large deformations and large time step sizes. Many of the nonlinear systems of equations that arise from implicit simulation can be integrated to get equivalent optimization problems, which allow robust optimization techniques to be applied. The MPM simulator in this work is based on [13], which formulated the backward Euler time integration with hyperelastic materials as a minimization problem. [27] and [48] also explored domain decompositions and hiearachical preconditioners to improve robustness and efficiency. The FEM simulator in this work is based on Incremental Potential Contact (IPC) [26], which proposed a variational form for frictional contacts. Their optimization-based frictional contact framework was also extended to codimensional objects [28], rigid bodies [11, 23], articulated multibodies [7], reduced elastic solids [24], embedded interfaces [53], and FEM-MPM coupled domains [29].

3 Background

3.1 Optimization Time Integration

In this section, we briefly introduce the optimization time integration for elastodynamics simulations with the Material Point Method (MPM) and the Finite Element Method (FEM). We refer the readers to [25] and [13] for more details.

FEM discretizes the simulation domain as unstructured meshes (e.g., triangle meshes in 2D), while in MPM, a point cloud composed of material particles is used to discretize the domain. While FEM directly uses the mesh nodes as the simulation degrees-of-freedom (DOF), MPM transfers its particle state to a uniform background grid, whose nodes are used as the DOFs for the integration of forces [16]. Robust simulation of elastodynamics can be achieved via implicit time integration, which updates the nodal positions ($\mathbf{x}$) or velocities ($\mathbf{v}$) step by step based on the previous physical states. To step from t^n to $t^{n+1} = t^n + \Delta t$ with time step size Δt, with implicit Euler time integration rule, one needs to solve a nonlinear system of equations

$$\mathbf{M}(\mathbf{v}^{n+1} - (\mathbf{v}^n + \mathbf{g}\Delta t)) = \Delta t \mathbf{f}^{n+1}. \tag{1}$$

Here $\mathbf{v}$ is the velocity DOF formed by concatenating all nodal velocity vectors, similarly concatenated, $\mathbf{M}$ is the mass matrix, $\mathbf{g}$ is the gravitational acceleration vector, and $\mathbf{f}$ is the internal force vector. Without plasticity, the internal force on a node i can be calculated as

$$\mathbf{f}_i^{n+1} = -\sum_q V_q^0 \mathbf{P}(\mathbf{F}_q^{n+1}) \nabla w_{iq}, \tag{2}$$

where q iterates the surrounding elements/particles of node i in FEM/MPM, V_q^0 is the initial volume of the element/particle, $\mathbf{F} = (\mathbf{I} + \Delta t \nabla \mathbf{v})\mathbf{F}^n$ (MPM) or $\mathbf{F} = \nabla \mathbf{x}^n + \Delta t \nabla \mathbf{v}$ (FEM) is the deformation gradient, which measures deformation from the undeformed state to the deformed state, and $\mathbf{P}$ is the first-Piola Kirchhoff stress, which describes the internal force per unit area within a material. ∇w_{iq} is the gradient of the weight function on node i evaluated on an element/particle center. The weight function is for transferring physical quantities between the elements/particles and the mesh/grid nodes. Unlike FEM, the last time step is used in MPM as the reference configuration, and so ∇w_{iq} is calculated as $\mathbf{F}_q^{n\top} \nabla w_{iq}^n$.

When there exists an energy density function Ψ such that $\mathbf{P}(\mathbf{F}) = \frac{\partial \Psi}{\partial \mathbf{F}}$, solving Equation 1 is equivalent to solving the following optimization problem

$$\mathbf{v}^{n+1} = \text{argmin}_{\mathbf{v}} \frac{1}{2} \|\mathbf{v} - (\mathbf{v}^n + \mathbf{g}\Delta t)\|_{\mathbf{M}}^2 + \sum_q V_q^0 \Psi(\mathbf{F}_q). \tag{3}$$

This formulation is more favored because with line search methods, convergence to a local minimum of Equation 3 can be guaranteed even when simulating challenging cases with stiff materials or large time step sizes. After solving for the velocity $\mathbf{v}^{n+1}$, FEM directly updates mesh nodal positions as $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}$, while for MPM, the velocity on the grid node is interpolated to particle locations for particle advection. The background grid is reset at the beginning of each time step, which allows MPM to benefit from the conveniences of a regular grid and a mesh-free formulation at the cost of some accuracy loss due to the transfers between the grid and particles.

3.2 Return Mapping for Plasticity

With plasticity, objects can undergo both plastic and elastic deformations, and the deformation gradient at the current time step can be decomposed as

$$\mathbf{F}^{n+1} = \mathbf{F}^{E,n+1} \mathbf{F}^{P,n+1} \tag{4}$$

based on the finite strain theory. Here $\mathbf{F}^{P,n+1}$ encodes the permanent plastic deformation of the rest shape, and $\mathbf{F}^{E,n+1}$ is the elastic deformation that results in effective elastic forces. In theory, $\mathbf{F}^{E,n+1}$ is constrained within certain elastic regions. Computation-wise, an elastic predictor $\mathbf{F}^{E,tr} = \mathbf{F}^{n+1}(\mathbf{F}^{P,n})^{-1}$ can be computed first by assuming

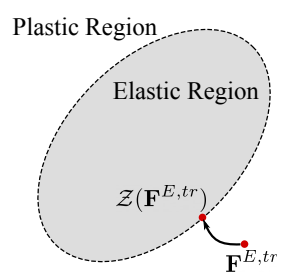


Figure 1: An illustration of a return mapping.

$\mathbf{F}^{P,n+1} = \mathbf{F}^{P,n}$. If $\mathbf{F}^{E,tr}$ is outside the elastic region, it will be projected back onto the boundary of the region to obtain $\mathbf{F}^{E,n+1} = \mathcal{Z}(\mathbf{F}^{E,tr})$ (Figure 1). This projection $\mathcal{Z}$ is called a *return mapping*. Within this framework, the implicit elastoplastic nodal force can be computed as [30]

$$\mathbf{f}_i^{n+1} = -\sum_q V_q^0 \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_q^{E,tr}))\mathbf{F}_q^{E,tr^{-\top}}\mathbf{F}^{P,n^{-\top}}\nabla w_{iq} \tag{5}$$

where $\boldsymbol{\tau}(\mathbf{F}) = \mathbf{P}(\mathbf{F})\mathbf{F}^\top$ is the Kirchoff stress. The above forces are integrable only if $\boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}$ can be represented as the gradient of some energy function:

$$\frac{\partial \Psi}{\partial \mathbf{F}} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}. \tag{6}$$

Most combinations of elastic constitutive models and plastic return mappings do not satisfy this integrability condition because the Jacobian field of the right-hand side is asymmetrical. Note that directly feeding $\mathcal{Z}(\mathbf{F})$ into an elastic potential does not form a potential energy for the elastoplastic forces defined in Equation 5. [30] only found one specific combination such that an elastoplastic potential energy exists. Thus, it remains challenging to simulate versatile plastic behaviors with optimization time integrators and achieve robust performance.

4 PlasticityNet

We propose PlasticityNet, a neural network-based elastoplastic model that finds a family of local potential energies whose negative gradients can approximate the elastoplastic forces within a small neighborhood so that plasticity can be conveniently simulated using optimization time integrators. The model architecture is illustrate in Figure 2. Specifically, instead of finding a global energy function $\Psi(\mathbf{F})$, we search for an energy $\Psi(\mathbf{F}, \mathbf{F}_0)$, parameterized by $\mathbf{F}_0$, such that

$$\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0)|_{\mathbf{F}=\mathbf{F}_0} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}, \quad \text{and} \quad \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0) \approx \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}. \tag{7}$$

To exactly enforce the first equality, we propose the following linear correction:

$$\Psi_\theta(\mathbf{F}, \mathbf{F}_0) = \mathcal{N}\mathcal{N}_\theta(\mathbf{F}, \mathbf{F}_0) - (\nabla_{\mathbf{F}}\mathcal{N}\mathcal{N}_\theta(\mathbf{F}_0, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}) \odot \mathbf{F}. \tag{8}$$

Here $\mathbf{A} \odot \mathbf{B} = A_{ij}B_{ij} = \text{tr}(\mathbf{A}^\top\mathbf{B})$ is the matrix inner product. It can be verified that $\frac{\partial \Psi_\theta}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0)|_{\mathbf{F}=\mathbf{F}_0} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}$.

Then we only need to focus on the approximation part in Equation 7. We design the training loss function for our neural network as

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{F}_0}\mathbb{E}_{\mathbf{F}}\left\|\frac{\partial \Psi_\theta}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}\right\|_F^2. \tag{9}$$

During training, $\mathbf{F}$ is only sampled near $\mathbf{F}_0$. Please refer to Section 5.1 for details.

4.1 Hardening of Plasticity

Hardening effects are widely observed in metals and snow. With hardening, the elastic region will expand by a certain amount whenever $\mathbf{F}^{E,tr}$ falls in the plastic region. To account for hardening, the return mapping $\mathcal{Z}(\mathbf{F}, h)$ and the energy $\Psi_\theta(\mathbf{F}, \mathbf{F}_0, h)$ will depend on an extra hardening state h, which controls the shape of the elastic region. This hardening state is a function of $\mathbf{F}$. However, to maintain integrability with respect to $\mathbf{F}$, we approximately update h based on $\mathbf{F}_0$, which is assumed to be close to $\mathbf{F}$.

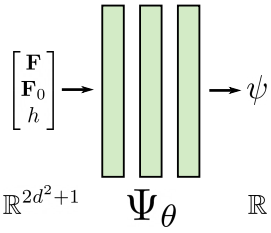


Figure 2: An overview of PlasticityNet. It is a map from $\mathbb{R}^{2d^2+1}$ to $\mathbb{R}$.

4.2 Optimization Time Integration with PlasticityNet

Fixed-Point Iteration The gradient of our learned elastoplastic potential energy $\Psi_\theta(\mathbf{F}, \mathbf{F}_0)$ only approximates the effective stresses locally near $\mathbf{F}_0$. To approach the accurate solution of Equation 1

with elastoplastic forces, we apply a fixed-point iteration on $\mathbf{F}_0$ to let it converge to $\mathbf{F}^{n+1}$. Specifically, we solve a sequence of optimization problems

$$\mathbf{v}^{n+1,j+1} = \operatorname{argmin}_{\mathbf{v}} \frac{1}{2}\|\mathbf{v} - (\mathbf{v}^n + \mathbf{g}\Delta t)\|_{\mathbf{M}} + \sum_q V_q^0 \Psi_\theta(\mathbf{F}_q, \mathbf{F}_{0,q}^j, h_q^j), \quad \text{for } j = 0, 1, 2, ..., \tag{10}$$

treating the concatenated deformation gradients $\mathbf{F}_0^j$ and hardening states $\mathbf{h}$ as constants, which are only updated before each optimization as $\mathbf{F}_0^j = \mathbf{F}(\mathbf{v}^{n+i,j})$ and $\mathbf{h} = \mathbf{h}(\mathbf{F}_0)$. At convergence, we will obtain the true solution of Equation 1. In practice, a few number of fixed-point iterations can already generate high-quality results.

Stability Regularizer We augment our learned potential with an extra quadratic regularizer to stabilize the simulation especially when the material is stiff or the time step size is large:

$$\Psi_\theta(\mathbf{F}, \mathbf{F}_0) = \mathcal{NN}_\theta(\mathbf{F}, \mathbf{F}_0) - (\nabla_{\mathbf{F}} \mathcal{NN}_\theta(\mathbf{F}_0, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}) \odot \mathbf{F} + \frac{1}{2}\mu\|\mathbf{F} - \mathbf{F}_0\|_F^2. \tag{11}$$

Here μ is the shear modulus of the material that Ψ_θ is learning. Note that this extra term is added after the model is trained instead of during the training. This extra term does not change the gradient at $\mathbf{F}_0$, so it will not change the fixed point of Procedure 10. Please see Section 5.3 for a comparison between simulations with and without this regularizer.

4.3 Learning Volume-Preserving Return Mapping

The return mapping $\mathcal{Z}$ required by PlasticityNet can be either given analytically or learned. Note that with different combinations of many practical elasticity and plasticity models, the return mapping may not have a closed-form solution, and the projection can only be performed by solving a nonlinear system of equations.

Here we provide a simple approach to learn a volume-preserving return mapping, which ensures that $\det(\mathcal{Z}(\mathbf{F})) = \det(\mathbf{F})$. For isotropic materials, the projection can be performed in the diagonal space, i.e., with $\mathbf{F} = \mathbf{U}\operatorname{Diag}(\boldsymbol{\Sigma})\mathbf{V}^\top$ being the singular value decomposition of $\mathbf{F}$; the projection is only needed for $\boldsymbol{\Sigma}$. In the diagonal space, a volume-preserving path is a straight line in the Hencky strain (defined as $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$) space, which is perpendicular to the diagonal line. The direction of the projection path is $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} - \operatorname{sum}(\boldsymbol{\epsilon})\mathbf{1}$. The volume-preserving projection in the Hencky strain can be unified by $\mathbf{H} = \boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}$ for some $\delta\gamma$, with $\mathcal{Z}^\Sigma(\boldsymbol{\Sigma}) = \exp(\mathbf{H})$ and $\mathcal{Z}(\mathbf{F}) = \mathbf{U}\operatorname{Diag}(\mathcal{Z}^\Sigma)\mathbf{V}^\top$. An illustration is shown in Figure 3.

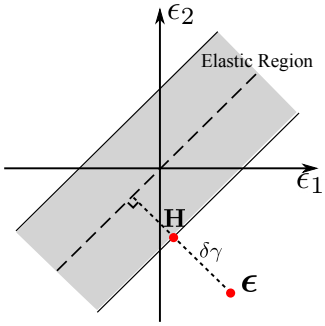


Figure 3: Volume-preserving projection.

The elastic region is usually represented by an implicit function $y(\boldsymbol{\Sigma}) \leq 0$. We can use a neural network to predict $\delta\gamma$, where the training leverages the differentiability of the implicit representation for the elastic region boundary. The volume-preserving path usually has two intersections with the elastic region boundary. To eliminate this ambiguity, we clamp the output of the neural network with a maximum $\|\hat{\boldsymbol{\epsilon}}\|$. We define our neural-network-based return mapping on the diagonal space as:

$$\delta\gamma_\theta(\boldsymbol{\Sigma}) = \min\{\mathcal{NN}_\theta(\Sigma), \|\hat{\boldsymbol{\epsilon}}\|\}, \quad \mathcal{Z}_\theta^\Sigma(\boldsymbol{\Sigma}) = \begin{cases} \exp(\boldsymbol{\epsilon} - \delta\gamma_\theta\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}), & y(\boldsymbol{\Sigma}) > 0, \\ \boldsymbol{\Sigma}, & y(\boldsymbol{\Sigma}) \leq 0. \end{cases} \tag{12}$$

The training loss function for a single $\boldsymbol{\Sigma}$ is defined as

$$\mathcal{L}(\boldsymbol{\Sigma}; \theta) = \begin{cases} y(\mathcal{Z}_\theta^\Sigma(\boldsymbol{\Sigma}))^2 + \max\{\delta\gamma_\theta(\boldsymbol{\Sigma}) - \|\hat{\boldsymbol{\epsilon}}\|, 0\}, & y(\boldsymbol{\Sigma}) > 0 \\ 0, & y(\boldsymbol{\Sigma}) \leq 0 \end{cases} \tag{13}$$

Here, the first term is to pull the points outside the elastic region back onto the boundary. The second term is to avoid these points to be always projected onto the diagonal due to the clamping in $\delta\gamma_\theta$. To account for hardening, we only need to let the $\delta\gamma$ network accept an extra hardening state variable h: $\delta\gamma_\theta(\boldsymbol{\Sigma}, h) = \min\{\mathcal{NN}_\theta(\Sigma, h), \|\hat{\boldsymbol{\epsilon}}\|\}$. The learned return mapping is then ready to be used by our PlasticityNet.

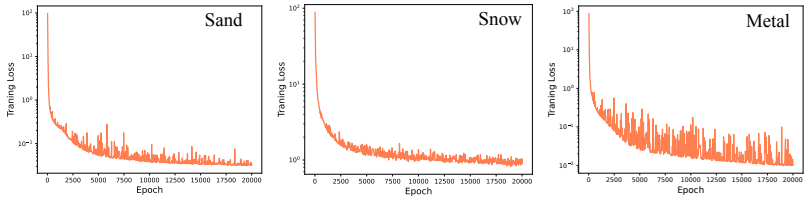


Figure 4: Training losses of our 2D models.

5 Experiments

We show examples to demonstrate the capability of our PlasticityNet in learning versatile plasticity models. Our physical simulators are implemented using C++, and we applied PyTorch to learn the potential energies, which are then loaded into our simulators with TorchScript. All our potential energies are trained as multilayer perceptrons using the Adam optimizer [19] on a single Nvidia RTX 3090 GPU. Please see Appendix A.1 for more training details. All ground-truth data are generated using standard explicit time integration with analytical plasticity returning mapping under small time step sizes for stability. With our PlasticityNet, we can robustly simulate elastoplastic behaviors with much larger time step sizes using optimization time integrators.

5.1 Training

The training of PlasticityNet only requires the return mapping (either given analytically or pre-trained) for the plasticity model and the Kirchhoff stress for the underlying elasticity model. There is no need for extra labeled data. At each epoch, we will sample a new batch of $(\mathbf{F}, \mathbf{F}_0, \mathbf{h})$. The sampling of deformation gradients is based on its singular value decomposition $\mathbf{F} = \mathbf{U}\,\mathrm{Diag}(\boldsymbol{\Sigma})\mathbf{V}^\top$, with $\mathbf{U}, \mathbf{V}$ being two rotation matrices. To sample $\mathbf{F}$ and $\mathbf{F}_0$ so that their singular values are close to each other, we set $\mathbf{F}_0 = \mathbf{R}_1\,\mathrm{Diag}(e^{\boldsymbol{\epsilon}})\mathbf{R}_2$ and $\mathbf{F} = \mathbf{R}_3\,\mathrm{Diag}(e^{\boldsymbol{\epsilon}+\delta\boldsymbol{\epsilon}})\mathbf{R}_4$, where $\boldsymbol{\epsilon}$ is a randomly sampled vector, $\delta\boldsymbol{\epsilon}$ is a random perturbation, and $\mathbf{R}_i$'s are randomly sampled rotation matrices. The hardening state is sampled uniformly from an appropriate range depending on the plasticity model. Please see Appendix A.2 for definitions of hardening states and their range selections. In this work, we uniformly sample $\boldsymbol{\epsilon}$ from $[-1, 1]^d$, $\delta\boldsymbol{\epsilon}$ from $[-0.1, 0.1]^d$ for sand plasticity and metal plasticity, and $[-0.2, 0.2]^d$ for the snow plasticity. The training loss curves of our 2D models are shown in Figure 4.

5.2 Testing on 2D Simulations

Sand Plasticity We start by learning the elastoplastic model of dry sand (Figure 5). The model combination is St. Venant-Kirchhoff (StVK) elasticity, and the closed-form Drucker Prager plasticity return mapping [20] (See Appendix A.2.1). In this example, we simulate a column of sand falling onto the ground under gravity with MPM. Our method generates visually identical results compared to the ground truth, both with the same time step size and a $100\times$ larger time step size. The quantitative comparison of our results and the ground truth is shown in Figure 9a. Note that there is no hardening mechanism in this plasticity model, so our PlasticityNet does not need the hardening state in its input.

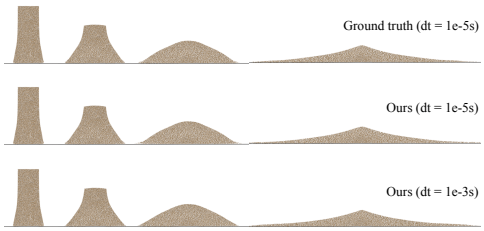


Figure 5: Sand plasticity.

Snow Plasticity with Hardening Snow is an elastoplastic material that can become stiffer under compression. Essentially, this is the effect of hardening where its elastic region get expanded. The variation in the stiffness across the snow body makes it easily fracture. Here we simulate a snowball hitting

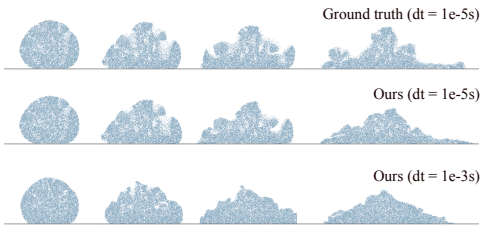
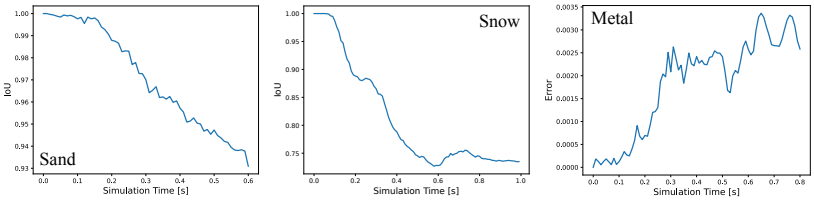


Figure 6: Snow plasticity with hardening.



(a) (b) (c)

Figure 9: **(a) (b)** The intersection-over-union (IOU) measure between the ground truth and our results. The IOUs are computed using the mass distributions on the MPM grid. **(c)** The average FEM nodal position difference. Note that the bounding box of the 2D metal frame is $0.1m \times 0.18m$.

the ground in the MPM simulator (Figure 6). We use Neo-Hookean elasticity with the closed-form non-associative Cam-Clay plasticity return mapping [14] (See Appendix A.2.2). Our method generates similar results compared to the ground truth when using the same time step size. The quantitative comparison of our results and the ground truth is shown in Figure 9b. Our framework remains stable even under much larger time step sizes. However, more numerical damping artifacts are introduced as the time step size increases, which results in slightly different behaviors compared to the ground truth.

Metal Plasticity with Hardening Metal is another common plastic material with hardening. In this example, we train PlasticityNet to learn metal plasticity with the StVK elasticity and the closed-form von-Mises plasticity return mapping [33]. (See Appendix A.2.3) We simulate a metal frame compressed by a rigid plate in the FEM simulator (Figure 7), where the Incremental Potential Contact (IPC) [26] is used to handle the frictional contact between the solids. When we run the explicit time integration to generate ground truth, we have to decrease Young's modulus to enable using large enough time step sizes so that the simulation can be finished in practical time. Our method with the original setting generates visually identical results using a much large time step size. The quantitative comparison of our result and the ground truth is shown in Figure 9c.

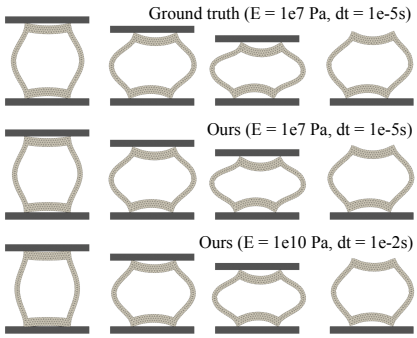


Figure 7: Metal plasticity with hardening.

Metal Plasticity Return Mapping Here we show an example simulated using PlasticityNet with a learned von-Mises plasticity return mapping. The underlying elasticity is neo-Hookean, instead of the StVK model in the last example (See Appendix A.2.4). Note that for Neo-Hookean material, there is no closed-form solution available for the von-Mises return mapping. In this case, a nonlinear optimization problem will need to be solved to perform the return mapping for every element/particle in every time step, which could severely slow down the standard explicit time integration. Using the same parameters as the metal compression experiments above, we show that PlasticityNet with learned plasticity return mapping under neo-Hookean elasticity can generate qualitatively similar results (Figure 8) to those from PlasticityNet with closed-form return mapping under the StVK elasticity.

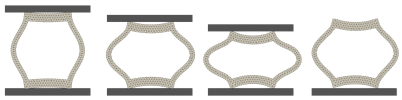


Figure 8: Learned metal plasticity return mapping with neo-Hookean elasticity.

5.3 An Ablation Study on Stability Regularizer

As an ablation study for the stability regularizer in Equation 11, we compare the simulations with and without the regularizer on two 2D examples (Figure 10). Without the regularizer, the metal frame can not even stay in its original rest configuration after the first time step. In the sand

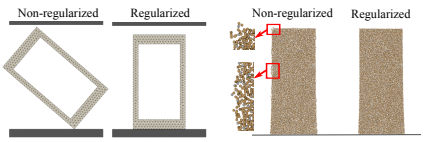


Figure 10: The regularizer significantly improves the stability of the simulation.

Figure 11: 3D sand collumn collapse simulaiton.

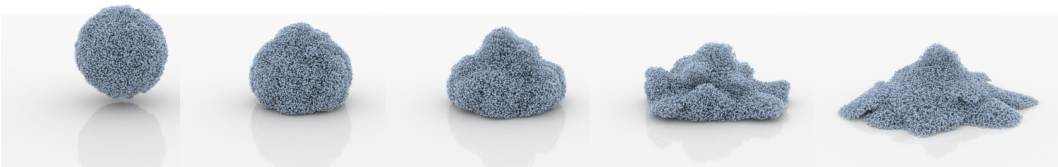


Figure 12: 3D snow simulation.

Figure 13: 3D metal compression simulation.

example, particles in the highlighted regions tend to separate from the sand column in a non-physical manner. These demonstrate that our regularizer significantly improves the stability of the simulation.

5.4 Testing on 3D Simulations

Extending PlasticityNet to support 3D simulation is straightforward. We only need to increase the dimension of the inputs to the PlasticityNet. To improve the expressiveness of the network, we also increase the dimension of hidden variables. Here we demonstrate the 3D versions of our 2D examples with similar physical parameters: 3D sand plasticity (Figure 11), 3D snow plasticity (Figure 12), and 3D metal plasticity (Figure 13). The 3D metal is simulated with $\Delta t = 10^{-2}$, and for sand and snow, we use $\Delta t = 10^{-3}s$ to satisfy the CFL condition [8] in MPM, preventing the particles from traveling farther than the grid cell spacing in a single time step.

6 Conclusion

We proposed PlasticityNet, a neural network-based elastoplastic model learning framework that is agnostic to spatial discretizations. PlasticityNet represents the elastoplastic forces as the positional gradients of learned potential energies, so that optimization time integration could be applied to achieve robust and efficient simulation at large time step sizes. We demonstrated that low-level components in traditional physical simulation frameworks can be substituted with neural networks to obtain desired numerical properties that benefit the computation. Notably, this also avoids tedious analytical derivations or expensive nonlinear root-findings without significantly sacrificing the accuracy. We believe our work can inspire more research that applies machine learning to physical simulation in the bottom-up style, maintaining fundamental physical properties and applicability to general scenarios.

Limitations and Future Work There are several limitations of our framework. First, we can not guarantee our fixed-point iteration procedure will converge for arbitrary scenes. It is theoretically valuable to explore under what conditions the fixed-point can converge and what kinds of training loss functions can accelerate the convergence. Second, although the regularizer added during the simulation improves the stability of the simulation without changing the solution at convergence, it may introduce some artificial viscosity because the regularized energy penalizes deformations away from $\mathbf{F}_0$. Running more fixed-point iterations can alleviate this issue. It will also be interesting to

explore adaptive weighting mechanisms for the regularizer. Third, we do not consider the Hessian of the learned plastic energy in our training. Since we use second-order methods to perform optimization time integration, the properties of the Hessian matrices may have an impact on the convergence of the optimization method. Although the Jacobian matrices of the target gradients are asymmetric, it may be helpful if the Hessian of our learned elastoplastic energy can approximate them so that the stiffness of the material can be more accurately resolved.

References

[1] F. As' ad, P. Avery, and C. Farhat. A mechanics-informed artificial neural network approach in data-driven constitutive modeling. In *AIAA Scitech 2022 Forum*, page 0100, 2022.

[2] A. W. Bargteil, T. Shinar, and P. G. Kry. An introduction to physics-based animation. In *SIGGRAPH Asia 2020 Courses*, pages 1–57. 2020.

[3] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[4] F. D. A. Belbute-Peres, T. Economon, and Z. Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning*, pages 2402–2411. PMLR, 2020.

[5] F. E. Bock, R. C. Aydin, C. J. Cyron, N. Huber, S. R. Kalidindi, and B. Klusemann. A review of the application of machine learning and data mining approaches in continuum materials mechanics. *Frontiers in Materials*, page 110, 2019.

[6] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[7] Y. Chen, M. Li, L. Lan, H. Su, Y. Yang, and C. Jiang. A unified newton barrier method for multibody dynamics. *ACM Trans. Graph. (SIGGRAPH)*, 41(4), 2022.

[8] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[9] D. C. Drucker. Some implications of work hardening and ideal plasticity. *Quarterly of Applied Mathematics*, 7(4):411–418, 1950.

[10] M.-L. Eckert, K. Um, and N. Thuerey. Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[11] Z. Ferguson, M. Li, T. Schneider, F. Gil-Ureta, T. Langlois, C. Jiang, D. Zorin, D. M. Kaufman, and D. Panozzo. Intersection-free rigid body dynamics. *ACM Transactions on Graphics*, 40(4), 2021.

[12] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem. A review on deep reinforcement learning for fluid mechanics. *Computers & Fluids*, 225:104973, 2021.

[13] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics*, 21(10):1103–1115, 2015.

[14] J. Gaume, T. Gast, J. Teran, A. Van Herwijnen, and C. Jiang. Dynamic anticrack propagation in snow. *Nature communications*, 9(1):1–10, 2018.

[15] C. Jiang, T. Gast, and J. Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[16] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pages 1–52. 2016.

[17] Y. Jin, Y. Han, Z. Geng, J. Teran, and R. Fedkiw. Analytically integratable zero-restlength springs for capturing dynamic modes unrepresented by quasistatic neural networks. *arXiv preprint arXiv:2201.10122*, 2022.

[18] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

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

[20] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[21] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

[22] A. Koeppe, F. Bamer, M. Selzer, B. Nestler, and B. Markert. Explainable artificial intelligence for mechanics: physics-explaining neural networks for constitutive models. *Front. Mater. 8: 824958. doi: 10.3389/fmats*, 2022.

[23] L. Lan, D. M. Kaufman, M. Li, C. Jiang, and Y. Yang. Affine body dynamics: Fast, stable & intersection-free simulation of stiff materials. *ACM Trans. Graph. (SIGGRAPH)*, 41(4), 2022.

[24] L. Lan, Y. Yang, D. Kaufman, J. Yao, M. Li, and C. Jiang. Medial ipc: accelerated incremental potential contact with medial elastics. *ACM Trans. on Graph.*, 40(4):1–16, 2021.

[25] M. Li. *Robust and Accurate Simulation of Elastodynamics and Contact*. PhD thesis, University of Pennsylvania, 2020.

[26] 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 transactions on graphics*, 2020.

[27] M. Li, M. Gao, T. Langlois, C. Jiang, and D. M. Kaufman. Decomposed optimization time integrator for large-step elastodynamics. *ACM Trans. Gr.*, 38(4), 2019.

[28] M. Li, D. M. Kaufman, and C. Jiang. Codimensional incremental potential contact. *ACM Transactions on Graphics*, 40(4), 2021.

[29] X. Li, Y. Fang, M. Li, and C. Jiang. Bfemp: Interpenetration-free mpm–fem coupling with barrier contact. *Comp. meth. applied mech. eng.*, 2021.

[30] X. Li, M. Li, and C. Jiang. Energetically consistent inelasticity for optimization time integration. *ACM Trans. Gr.*, 41(4), 2022.

[31] X. Li, B. Sovilla, C. Jiang, and J. Gaume. Three-dimensional and real-scale modeling of flow regimes in dense snow avalanches. *Landslides*, 18(10):3393–3406, 2021.

[32] R. Luo, T. Shao, H. Wang, W. Xu, X. Chen, K. Zhou, and Y. Yang. Nnwarp: Neural network-based nonlinear deformation. *IEEE transactions on visualization and computer graphics*, 26(4):1745–1759, 2018.

[33] R. v. Mises. Mechanik der festen körper im plastisch-deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592, 1913.

[34] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. Bessa. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420, 2019.

[35] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[36] W. Prager. The theory of plasticity: a survey of recent achievements. *Proceedings of the Institution of Mechanical Engineers*, 169(1):41–57, 1955.

[37] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[38] D. Reimann, K. Nidadavolu, H. ul Hassan, N. Vajragupta, T. Glasmachers, P. Junker, and A. Hartmaier. Modeling macroscopic material behavior with machine learning algorithms trained by micromechanical simulations. *Frontiers in Materials*, 6:181, 2019.

[39] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[40] S. Shen, T. Shao, K. Zhou, C. Jiang, F. Luo, and Y. Yang. Hod-net: High-order differentiable deep neural networks and applications. 2022.

[41] S. Shen, Y. Yin, T. Shao, H. Wang, C. Jiang, L. Lan, and K. Zhou. High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026*, 2021.

[42] E. Sifakis and J. Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. 2012.

[43] A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran. Energetically consistent invertible elasticity. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, pages 25–32, 2012.

[44] L. Sun, H. Gao, S. Pan, and J.-X. Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[45] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[46] N. N. Vlassis, R. Ma, and W. Sun. Geometric deep learning for computational mechanics part i: Anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 371:113299, 2020.

[47] N. N. Vlassis and W. Sun. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695, 2021.

[48] X. Wang, M. Li, Y. Fang, X. Zhang, M. Gao, M. Tang, D. M. Kaufman, and C. Jiang. Hierarchical optimization time integration for cfl-rate mpm stepping. *ACM Trans. on Graph.*, 39(3):1–16, 2020.

[49] J. Wolper, Y. Chen, M. Li, Y. Fang, Z. Qu, J. Lu, M. Cheng, and C. Jiang. Anisompm: Animating anisotropic damage mechanics: Supplemental document. *ACM Trans. Graph*, 39(4), 2020.

[50] J. Wolper, Y. Fang, M. Li, J. Lu, M. Gao, and C. Jiang. Cd-mpm: continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[51] J. Wolper, M. Gao, M. P. Lüthi, V. Heller, A. Vieli, C. Jiang, and J. Gaume. A glacier–ocean interaction model for tsunami genesis due to iceberg calving. *Communications Earth & Environment*, 2(1):1–10, 2021.

[52] S. Yang, X. He, and B. Zhu. Learning physical constraints with neural projections. *Advances in Neural Information Processing Systems*, 33:5178–5189, 2020.

[53] Y. Zhao, J. Choo, Y. Jiang, M. Li, C. Jiang, and K. Soga. A barrier method for frictional contact on embedded interfaces. *Computer Methods in Applied Mechanics and Engineering*, 393:114820, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Network Architecture

All our models are using the Multilayer Perceptron (MLP) architecture with Swiss activation functions ($x\,\mathrm{sigmoid}(x)$) except the output layer. They are trained using ADAM optimizer with the same parameters: initial learning rate $\alpha = 0.01$, decay rate $\gamma = 0.95$, decay step 1000. The dataset is generated during the training process with random sampling, as discussed in Section 5.1, and the batch size is 2^{16} for all cases. The models are all trained with 20000 epochs. The detailed architectures for each model is listed in Table 1.

A.2 Technical Details on Plasticity Models

We focus on isotropic materials, where the elasticity and plasticity can both be described in the diagonal space without loss of generality. Given the polar singular value decomposition of the

Table 1: Network Architectures and Training Details

Model	**MLP layers**
Sand (StVK+Drucker-Prager)	[8,32,32,32,1]
Snow (Neohooken+NACC)	[9,32,32,32,1]
Metal (StVK+von-Mises)	[9,32,32,32,1]
Sand 3D	[18,64,64,64,1]
Metal 3D (StVK+von-Mises)	[19,64,64,64,1]
Snow 3D	[19,64,64,64,1]

deformation gradient $\mathbf{F} = \mathbf{U}\,\mathrm{Diag}(\mathbf{\Sigma})\mathbf{V}^\top$, the Kirchhoff stress $\boldsymbol{\tau}$ and the return mapping $\mathcal{Z}$ can both be computed solely by $\mathbf{\Sigma}$ as $\boldsymbol{\tau}^\Sigma$ and $\mathcal{Z}^\Sigma$, and then restored to the full space via $\boldsymbol{\tau} = \mathbf{U}\,\mathrm{Diag}(\boldsymbol{\tau}^\Sigma)\mathbf{V}^\top$ and $\mathcal{Z} = \mathbf{U}\,\mathrm{Diag}(\mathcal{Z}^\Sigma)\mathbf{V}^\top$. In the following sections, we will omit the superscript Σ and only discuss the models in the diagonal space.

Here is a list of the material parameters that will be mentioned in the following sections:

Notation	Meaning	Relation to (E, ν)
E	Young's modulus	/
ν	Poisson's ratio	/
μ	Shear modulus	$\mu = \frac{E}{2(1+\nu)}$
λ	Lamé modulus	$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$
K	Bulk modulus	$K = \frac{E}{3(1-2\nu)}$

A.2.1 Sand Plasticity

We use StVK elasticity and Drucker-Prager plasticity for sand simulations [20].

The Kirchhoff stress $\boldsymbol{\tau}$ of StVK elasticity is defined as

$$\begin{aligned} \boldsymbol{\epsilon} &= \log(\mathbf{\Sigma}), \\ \boldsymbol{\tau} &= 2\mu\boldsymbol{\epsilon} + \lambda\,\mathrm{sum}(\boldsymbol{\epsilon})\mathbf{1}. \end{aligned} \tag{14}$$

The elastic region is characterised in the stress space as:

$$\alpha\,\mathrm{sum}(\tau) + \|\,\mathrm{sum}(\tau) - \frac{\mathrm{sum}(\tau)}{d}\mathbf{1}\| \leq 0, \tag{15}$$

where $\alpha = \sqrt{\frac{2}{3}}\frac{2\sin\phi_f}{3-\sin\phi_f}$ and ϕ_f is the friction angle. In our sand examples, ϕ_f is set to $\frac{\pi}{6}$. The elastic region in the stress space is shown in Figure 14.

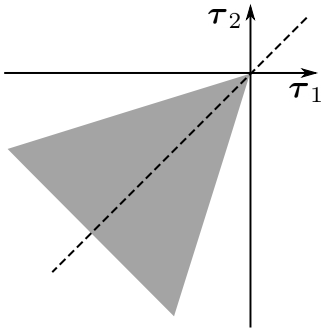


Figure 14: Drucker-Prager plasticity's elastic region in the stress space.

The return mapping for the Drucker-Prager plasticity is

$$\mathcal{Z}(\mathbf{\Sigma}) = \begin{cases} \mathbf{1} & \mathrm{sum}(\boldsymbol{\epsilon}) > 0 \\ \mathbf{\Sigma} & \delta\gamma \leq 0, \text{and}\, \mathrm{sum}(\boldsymbol{\epsilon}) \leq 0 \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right) & \text{otherwise} \end{cases}, \tag{16}$$

where $\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| + \alpha\frac{(d\lambda+2\mu)\,\mathrm{sum}(\boldsymbol{\epsilon})}{2\mu}$.

A.2.2 Snow Plasticity

We use neo-Hookean elasticity and non-associative Cam-Clay (NACC) plasticity for snow simulations [50].

The Kirchhoff stress of neo-Hookean elasticity is

$$\begin{aligned} J &= \det \mathrm{Diag}(\boldsymbol{\Sigma}), \\ \mathbf{b} &= \boldsymbol{\Sigma}^2, \\ \widehat{\mathbf{b}} &= \mathrm{dev}(\mathbf{b}) = \mathbf{b} - \frac{\mathrm{sum}(\mathbf{b})}{d}\mathbf{1}, \\ \boldsymbol{\tau} &= \mu J^{-\frac{2}{d}}\widehat{\mathbf{b}} + \frac{K}{2}(J^2-1)\mathbf{1}. \end{aligned} \tag{17}$$

The elastic region of NACC is characterized by

$$y(p,q) = q^2(1+2\beta) + M^2(p+\beta p_0)(p-p_0) \leq 0, \tag{18}$$

where

$$\begin{aligned} M &= d\sqrt{\frac{6-d}{2}}\sqrt{\frac{2}{3}}\frac{2\sin\phi_f}{3-\sin\phi_f} \\ p &= \frac{K}{2}(J^2-1), \\ q &= \mu J^{-\frac{2}{d}}\sqrt{\frac{6-d}{2}}\|\widehat{\mathbf{b}}\|, \\ p_0 &= K\sinh(\xi\max\{-\alpha,0\}). \end{aligned} \tag{19}$$

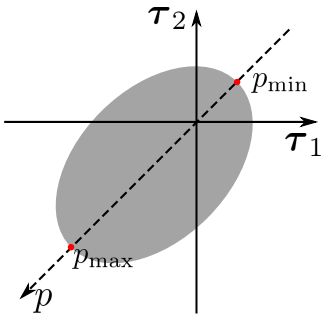


Figure 15: NACC plasticity's elastic region in the stress space.

ξ, β, ϕ_f are the parameters of plasticity and α is the hardening state. The elastic region in the stress space is shown in Figure 15. In our snow examples, $\xi = 0.5$, $\beta = 0.3$ and $\phi_f = \frac{\pi}{4}$.

The return mapping is defined as

$$\mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} (1-\frac{2p_{\max}}{K})^{-\frac{1}{2d}}\mathbf{1}, & p > p_{\max} = p_0, \\ (1+\frac{2p_{\min}}{K})^{-\frac{1}{2d}}\mathbf{1}, & p < p_{\min} = -\beta p_0, \\ \boldsymbol{\Sigma}, & y(p,q) \leq 0, \\ \frac{J^{-\frac{2}{d}}}{\mu}\sqrt{\frac{-2M^2(p+\beta p_0)(p-p_0)}{(6-d)(1+2\beta)}}\frac{\widehat{\mathbf{b}}}{\|\widehat{\mathbf{b}}\|} + \frac{1}{d}\mathrm{sum}(\mathbf{b})\mathbf{1} & \text{Otherwise} \end{cases} \tag{20}$$

Please refer to [50] for the hardening state update procedure, which is controlled by the simulator. For PlasticityNet, we set $h = \min\{\alpha, 0\}$ as the hardening state input. During the training, we sample $h \in [-0.5, 0]$ for 2D snow and $h \in [-1, 0]$ for 3D snow.

A.2.3 Metal Plasticity under StVK Elasticity

We use StVK elasticity and von-Mises plasticity for metal simulations [20]. This combination provides a closed-form return mapping projection.

The elastic region is characterized by

$$\|\boldsymbol{\tau} - \frac{1}{d}\mathrm{sum}(\boldsymbol{\tau})\| - \tau_y \leq 0, \tag{21}$$

where τ_y controls the radius of the yield surface in the stress space (Figure 16).

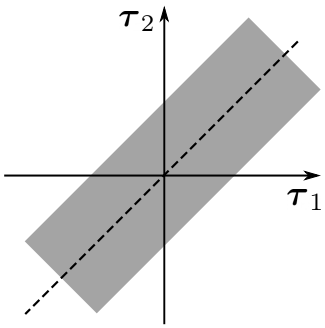


Figure 16: Von-Mises plasticity's elastic region in the stress space.

The return mapping for the von-Mises plasticity is defined as

$$\mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} \boldsymbol{\Sigma}, & \|\boldsymbol{\tau} - \frac{1}{d}\mathrm{sum}(\boldsymbol{\tau})\| - \tau_y \leq 0 \\ \exp(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}), & \text{Otherwise} \end{cases}, \tag{22}$$

where $\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\|_F - \frac{\tau_y}{2\mu}$.

Under hardening, τ_Y is updated with

$$\tau_Y^{n+1} = \tau_Y^n + 2\mu\xi\delta\gamma, \tag{23}$$

where ξ is the hardening coefficient.

We use $h = \frac{\tau_Y}{2\mu}$ as the hardening state input to our PlasticityNet. During the training, h is sampled from $[0, 1]$.

A.2.4 Metal Plasticity under Neo-Hookean Elasticity

The combination of neo-Hookean elasticity and von-Mises plasticity does not have a closed-form return mapping, we thereby use this combination for the task of learning metal plasticity return mapping. The Kirchhoff stress of neo-Hookean elasticity is given by

$$\boldsymbol{\tau} = \mu(\boldsymbol{\Sigma}^2 - \mathbf{1}) + \lambda \log J \mathbf{1}. \tag{24}$$

The implicit representation of the elastic region we used in the training of the return mapping is given by

$$y(\boldsymbol{\Sigma}, h) = \|\boldsymbol{\tau} - \frac{1}{d}\operatorname{sum}(\boldsymbol{\tau})\|^2 - (2\mu h)^2. \tag{25}$$

During training, h is sampled from $[0, 1]$.