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

Xuan Li
Department of Mathematics
University of California, Los Angeles
xuanli1@math.ucla.edu

Yadi Cao
Department of Computer Science
University of California, Los Angeles
yadicao95@cs.ucla.edu

Minchen Li
Department of Mathematics
University of California, Los Angeles
minchen@math.ucla.edu

Yin Yang
School of Computing
University of Utah
yin.yang@utah.edu

Craig Schroeder
Department of Computer Science and Engineering
University of California, Riverside
craigs@cs.ucr.edu

Chenfanfu Jiang
Department of Mathematics
University of California, Los Angeles
cffjiang@math.ucla.edu

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 [46, 41]. Harnessing the power of neural networks, this research has been successfully applied in computer animation [10], multibody systems [3, 6, 59, 12], human musculature simulation [20], computational fluid dynamics [4, 13], and non-linear continuum mechanics [5]. An alternative direction is represented by physics-informed neural networks (PINN) [44, 21], 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 [51]. Another noteworthy category is learning the physical modeling where the machine can either help increase the model resolution in a coarser grid [24], inject nonlinearity to a linear model [37], or apply a learnable model reduction to reduce the system degrees-of-freedom (DOF) for acceleration [47, 48].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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 [19]) cannot be directly applied to the mesh-based Finite Element Method (FEM) [49], 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 [42, 9, 23, 52] and metal [39], to snow [15, 57, 35] and glacier [58].

Many elastic materials, including those represented by mass-spring systems [2] and common *hyperelastic* solids [50], 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 projected Newton and line search can guarantee global convergence to a solution [28]. 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) [34], 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 [42, 9], and developed particle-based simulations of dry [23] and wet [52] sand. Extending a similar Cam-Clay plasticity model, snow avalanches [15, 35], glacier calving [58] and food fracturing [57] are also captured with high visual plausibility as well as physical accuracy. For metals and dough-like materials, the von-Mises plasticity model [39] is usually adopted, while [56, 18] presented its anisotropic extensions. Still, the implementation of these models in modern, optimization-based simulators is cumbersome due to the non-integrable forces. Recently, [34] 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 [40, 45]. A similar approach is applied in [53] 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, 54, 25], 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 [14], which formulated the backward Euler time integration with hyperelastic materials as a minimization problem [30] and [55] 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) [29], which proposed a variational form for frictional contacts. Their optimization-based frictional contact framework was also extended to codimensional objects [31], rigid bodies [11, 26], articulated multibodies [7], reduced elastic solids [27], embedded interfaces [60], and FEM-MPM coupled domains [33].

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 [28] and [14] 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 [19]. Robust simulation of elastodynamics can be achieved via implicit time integration, which updates the nodal positions x (or velocities 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

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

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

$$f_i^{n+1} = -\sum_q V_q^0 P(F_q^{n+1}) \nabla w_{iq}; \tag{2}$$

where q iterates the surrounding elements/particles of node i. In FEM/MPM, V_q^0 is the initial volume of the element/particle, $F = (I + \Delta t \nabla v)F^n$ (MPM) or $F = \nabla x^n + \Delta t \nabla v$ (FEM) is the deformation gradient, which measures deformation from the undeformed state to the deformed state, and 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 F is calculated as $F_q^{n>} \nabla w_{iq}^n$.

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

$$v^{n+1} = \operatorname{argmin}_v \frac{1}{2}\|v - (v^n + g\Delta t)\|_M^2 + \sum_q V_q^0 \Psi(F_q). \tag{3}$$

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 v^{n+1}, FEM directly updates mesh nodal positions as $x^{n+1} = x^n + \Delta t v^{n+1}$, 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

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

based on the finite strain theory. Here $F^{P;n+1}$ encodes the permanent plastic deformation of the rest shape, and $F^{E;n+1}$ is the elastic deformation that results in effective elastic forces. In theory, $F^{E;n+1}$ is constrained within certain elastic regions. Computation-wise, an elastic predictor $F^{E;tr} = F^{n+1}(F^{P;n})^{-1}$ can be computed first by assuming $F^{P;n+1} = F^{P;n}$. If $F^{E;tr}$ is outside the elastic region, it will be projected back onto the boundary of the region to obtain $F^{E;n+1} = Z(F^{E;tr})$ (Figure 1). This projection Z is called a return mapping. Within this framework, the implicit elastoplastic nodal force can be computed as [34]

Figure 1: An illustration of a return mapping.

$$f_i^{n+1} = \sum_q V_q^0 \, \tau(Z(F_q^{E;tr})) F_q^{E;tr\,>} \, F^{P;n\,>} \, \nabla w_{iq} \tag{5}$$

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

$$\frac{\partial}{\partial F} = \tau(Z(F))F^{>} : \tag{6}$$

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 $Z(F)$ into an elastic potential does not form a potential energy for the elastoplastic forces defined in Equation 5.3 [4] 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 $\Psi(F)$, we search for an energy $\psi(F; F_0)$, parameterized by F_0, such that

$$\frac{\partial}{\partial F}(F; F_0)\big|_{F=F_0} = \tau(Z(F_0))F_0^{>} \; ; \quad \text{and} \quad \frac{\partial}{\partial F}(F; F_0) \quad \tau(Z(F))F^{>} : \tag{7}$$

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

$$\psi(F; F_0) = \mathrm{NN}(F; F_0) \quad (\nabla_F \mathrm{NN}(F_0; F_0) \quad \tau(Z(F_0))F_0^{>}) \quad F: \tag{8}$$

Here $A \quad B = A_{ij} B_{ij} = \mathrm{tr}(A^{>} B)$ is the matrix inner product. It can be verified that $\frac{\partial}{\partial F}(F; F_0)\big|_{F=F_0} = \tau(Z(F_0))F_0^{>} :$

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

$$L(\ ) = E_{F_0} E_F \left\| \frac{\partial}{\partial F}(F; F_0) \quad \tau(Z(F))F^{>} \right\|_F^2 : \tag{9}$$

During training, F is only sampled near 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 $F^{E,tr}$ falls in the plastic region. To account for hardening, the return mapping $Z(F; h)$ and the energy $(F; F_0; h)$ will depend on an extra hardening state h, which controls the shape of the elastic region. This hardening state is a function of F. However, to maintain integrability with respect to F, we approximately update h based on F_0, which is assumed to be close to F.

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

4.2 Optimization Time Integration with PlasticityNet

Fixed-Point Iteration The gradient of our learned elastoplastic potential energy $(F; F_0)$ only approximates the effective stresses locally near F_0. To approach the accurate solution of Equation 1 with elastoplastic forces, we apply a fixed-point iteration on F_0 to let it converge to F^{n+1}. Specifically, we solve a sequence of optimization problems

$$v^{n+1;j+1} = \operatorname{argmin}_v \frac{1}{2}\|v \quad (v^n + g \quad t)\|_M + \sum_q V_q^0 \quad (F_q; F_{0;q}^j; h_q^j); \quad \text{for } j = 0; 1; 2; :::; \quad (10)$$

treating the concatenated deformation gradients F_0^j and hardening states h as constants, which are only updated before each optimization as $F_0^j = F(v^{n+1;j})$ and $h = h(F_0)$. 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:

$$(F; F_0) = NN \quad (F; F_0) \quad (r_F NN \quad (F_0; F_0) \qquad (Z(F_0))F_0^{>} \quad) \quad F + \frac{1}{2} \quad \|F \quad F_0\|_F^2 : \quad (11)$$

Here is the shear modulus of the material that 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 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 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 $\det(Z(F)) = \det(F)$. For isotropic materials, the projection can be performed in the diagonal space, i.e., with $F = U\,\mathrm{Diag}( )V^{>}$ being the singular value decomposition of F; the projection is only needed for . In the diagonal space, a volume-preserving path is a straight line in the Hencky strain (defined as $= \log( )$) space, which is perpendicular to the diagonal line. The direction of the projection path is $\hat{} = \mathrm{sum}( )1$. The volume-preserving projection in the Hencky strain can be unified by $H = \frac{\hat{}}{k\hat{}k}$ for some , with $Z( ) = \exp(H)$ and $Z(F) = U\,\mathrm{Diag}(Z \quad)V^{>}$. An illustration is shown in Figure 3.

Figure 3: Volume-preserving projection.

The elastic region is usually represented by an implicit function $(\ ) \quad 0$. We can use a neural network to predict , 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

Figure 4: Training losses of our 2D models.

elastic region boundary. To eliminate this ambiguity, we clamp the output of the neural network with a maximum $\hat{k}k$. We de ne our neural-network-based return mapping on the diagonal space as:

$$(\quad) = \min\{N\,N\quad(\,)\;;\hat{k}k\}; \quad Z\quad(\quad) = \begin{cases} \exp\left(\quad\frac{\hat{}}{\hat{k}k}\right); & y(\quad) > 0; \\ ; & y(\quad)\quad 0: \end{cases} \tag{12}$$

The training loss function for a single is de ned as

$$L(\quad;\quad) = \begin{cases} y(Z\quad(\quad))^2 + \max\{\quad(\quad)\quad\hat{k}k; 0\}; & y(\quad) > 0 \\ 0; & y(\quad)\quad 0 \end{cases} \tag{13}$$

Here, the rst 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. To account for hardening, we only need to let the network accept an extra hardening state variable h: $(\quad;h) = \min\{N\,N\quad(\quad;h);\hat{k}k\}$. The learned return mapping is then ready to be used by our PlasticityNet.

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 [22] 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 $(F; F_0; h)$. The sampling of deformation gradients is based on its singular value decomposition $F = U\,\mathrm{Diag}(\quad)V^{>}$, with $U; V$ being two rotation matrices. To sample F and F_0 so that their singular values are close to each other, we set $F_0 = R_1\,\mathrm{Diag}(e\quad)R_2$ and $F = R_3\,\mathrm{Diag}(e^{+}\quad)R_4$, where is a randomly sampled vector, is a random perturbation, and 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 de nitions of hardening states and their range selections. In this work, we uniformly sample from $[\quad 1; 1]^d$, from $[\quad 0{:}1; 0{:}1]^d$ 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

In this section, explicit time integrators are used to generate the ground-truth data for the validation of the optimization time integrators with PlasticityNet on multiple 2D experiments. The quantitative comparisons are plotted in Figure 5. We additionally include the computational costs in Table 1. We

(a) (b) (c)

Figure 5: (a) (b) The intersection-over-union (IOU) [17] 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 0.18 m.

Table 1: Computational costs of 2D experiments.

Experiment	Ours		Explicit	
	Time step (s)	s/frame	Time step (s)	s/frame
Sand	1e-3	12.58	1e-5	6.20
Snow	1e-3	35.56	1e-5	6.78
Von-Mises Metal	1e-2	1.08	1e-5	5.39
Neohookean Metal	1e-2	1.03	1e-5	7.88
MPM-FEM Coupling	1e-3	38.90	1e-6	184.58

remark that the main objective of our work is not to surpass the performance of the existing simulation of every constitutive model, but to provide a methodology that enables the usage of implicit plasticity in an optimization time integration framework.

Sand Plasticity We start by learning the elastoplastic model of dry sand (Figure 6). The model combination is St. Venant-Kirchhoff (StVK) elasticity, and the closed-form Drucker Prager plasticity return mapping [23] (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 10× larger time step size. The quantitative comparison between our results and the ground truth is shown in Figure 5a. Note that there is no hardening mechanism in this plasticity model, so our PlasticityNet does not need the hardening state in its input.

Figure 6: 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 the ground in the MPM simulator (Figure 7). We use Neo-Hookean elasticity with the closed-form non-associative Cam-Clay plasticity return mapping [15] (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 5b. 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.

Figure 7: Snow plasticity with hardening.