Model Reduction for the Material Point Method on Nonlinear Manifolds Using Deep Learning

Peter Yichen Chen[1,2], **Maurizio Chiaramonte**[1], **Eitan Grinspun**[2,3], **Kevin Carlberg**[1]
[1]Facebook Reality Labs Research
[2]Columbia University
[3]Toronto University
cyc@cs.columbia.edu, mchiaram@fb.com, eitan@cs.toronto.edu, carlberg@fb.com

Abstract

This work proposes a projection-based model-reduction approach for the material point method (MPM), a popular hybrid Eulerian–Lagrangian method used in engineering and computer graphics for simulating solids, fluids, and multiphase phenomena. The proposed technique employs a *kinematic approximation* of the deformation map by enforcing deformation trajectories to reside on a low-dimensional manifold expressed in extrinsic form via a parameterization function corresponding to a decoder. By explicitly approximating the deformation map, the deformation gradient can be computed simply by differentiating the associated parameterization function. Further, this mapping can be inverted in order to generate new material points as needed allowing for adaptive refinement. The method generates *dynamics* via projection, i.e., at every time step, the method performs two steps: (1) compute the velocity on grid nodes of the Eulerian mesh, and (2) perform least-squares projection of the computed velocity onto the tangent space to this low-dimensional manifold. The approach supports hyper-reduction, which is essential for achieving computational complexity independent of the original number of material points. Numerical examples on large-scale problems illustrate the method's ability to generate orders-of-magnitude computational-cost savings with negligible error.

Introduction

The Material Point Method (MPM) (Sulsky, Zhou, and Schreyer 1995; Jiang et al. 2016) is a hybrid Eulerian–Lagrangian discretization method widely employed in the computer-graphics community for solid, fluid, and multiphase simulations. Due to its dual Eulerian and Lagrangian representations, MPM offers several advantages over the finite element method: large deformation, fracture, as well as contact and collision are handled relatively easily.

However, MPM's dual representations of the material make it especially computationally expensive. This precludes the ability of MPM to be used in time-critical and real-time settings such as fast-turnaround design, interactive applications, and control.

Reduced-order model

We introduce a strategy for constructing a projection-based reduced-order model (ROM) that introduces a *kinematic approximation* comprising constructing a low-dimensional nonlinear manifold approximation to the deformation map and generates *dynamics* by first computing the velocity on grid nodes of the Eulerian mesh, and then performing least-squares projection of the computed velocity onto the tangent space of the manifold.

Kinematics: low-dimensional manifold

In analogue to constructing low-dimensional nonlinear manifolds for finite-dimensional dynamical-system state spaces (Lee and Carlberg 2020), one can restrict any element of the reference domain $\Omega_{\text{ref}} \subseteq \mathbb{R}^d$ to evolve on a low-dimensional manifold. We first denote the approximated deformation map as $\tilde{\boldsymbol{\phi}} : \Omega_{\text{ref}} \times \mathcal{T} \times \mathcal{D} \to \mathbb{R}^d$ with $\tilde{\boldsymbol{\phi}}(\cdot;\cdot,\boldsymbol{\mu}) \in \mathcal{S}(\boldsymbol{\mu})$, $\forall \boldsymbol{\mu} \in \mathcal{D}$ and

$$\begin{aligned} \tilde{\boldsymbol{\phi}}(\cdot;t,\boldsymbol{\mu}) \ &: \boldsymbol{X} \mapsto \tilde{\boldsymbol{x}}(t,\boldsymbol{\mu}) && (1)\\ &: \Omega_{\text{ref}} \to \tilde{\Omega}(t,\boldsymbol{\mu}), \quad \forall t \in \mathcal{T},\ \boldsymbol{\mu} \in \mathcal{D}, && (2) \end{aligned}$$

where $\mathcal{S}(\boldsymbol{\mu})$ denotes the space of admissible solutions satisfying initial and essential boundary counditions, $\tilde{\Omega}(t,\boldsymbol{\mu}) \subseteq \mathbb{R}^d$ denotes the deformed domain corresponding to the approximated deformation map at time $t \in \mathcal{T}$ and parameter instance $\boldsymbol{\mu} \in \mathcal{D}$, and enforce the kinematic constraint

$$\tilde{\boldsymbol{\phi}}(\boldsymbol{X};\cdot,\cdot) \in \mathcal{M}(\boldsymbol{X}) := \{\boldsymbol{g}(\boldsymbol{X};\hat{\boldsymbol{y}}) \,|\, \hat{\boldsymbol{y}} \in \mathbb{R}^r\} \subseteq \mathbb{R}^d, \quad (3)$$

$\forall \boldsymbol{X} \in \Omega_{\text{ref}}$, where $\boldsymbol{g} : \Omega_{\text{ref}} \times \mathbb{R}^r \to \mathbb{R}^d$ denotes the parameterization of an r-dimensional manifold.

The kinematic restriction (3) implies that there exists generalized coordinates $\hat{\boldsymbol{x}} : \mathcal{T} \times \mathcal{D} \to \mathbb{R}^r$ such that

$$\begin{aligned} &\tilde{\boldsymbol{\phi}}(\boldsymbol{X};t,\boldsymbol{\mu}) = \boldsymbol{g}(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu})),\\ &\forall \boldsymbol{X} \in \Omega_{\text{ref}},\ \forall t \in \mathcal{T},\ \boldsymbol{\mu} \in \mathcal{D}. \end{aligned} \quad (4)$$

Assuming the parameterization function is continuously differentiable in its first argument, Eq. (4) implies that the deformation gradient of the approximate deformation map can be calculated by differentiating the parameterization function as

$$\begin{aligned} \tilde{\boldsymbol{F}} \ &: (\boldsymbol{X},t,\boldsymbol{\mu}) \mapsto \nabla\tilde{\boldsymbol{\phi}}(\boldsymbol{X};t,\boldsymbol{\mu}) \equiv \nabla\boldsymbol{g}(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu}))\\ &: \Omega_{\text{ref}} \times \mathcal{T} \times \mathcal{D} \to \mathbb{R}^{d\times d}. \end{aligned} \quad (5)$$

, where ∇ denotes differentiation with respect to the first argument. Further, assuming the parameterization function is continuously differentiable in its second argument, Eq. (4) implies that the velocity of the approximate solution can be calculated via the chain rule as

$$\dot{\tilde{\boldsymbol{\phi}}}(\boldsymbol{X};t,\boldsymbol{\mu}) \equiv \frac{\partial \boldsymbol{g}}{\partial \hat{\boldsymbol{x}}}(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu}))\dot{\hat{\boldsymbol{x}}}(t,\boldsymbol{\mu}). \tag{6}$$

Remark (Manifold construction via deep learning)**.** *While the manifold-parameterization function $\boldsymbol{g} : \Omega_{ref} \times \mathbb{R}^r \to \mathbb{R}^d$ can be constructed in a variety of ways, we employ a fully-connected, 5-layer, deep-learning architecture (i.e., a multi-layer perceptron) with ELU activation functions due to their continuous differentiability. ELU can be viewed as a smooth extension of the more popular ReLU activation function. We set $\boldsymbol{g} \leftarrow \boldsymbol{g}_{\theta^\star}$ where $\theta^\star$, the network weights, are the (approximate) solutions to the problem*

$$\begin{aligned}&\underset{\theta,\,\{\hat{\boldsymbol{x}}(t,\boldsymbol{\mu})\}_{t\in\mathcal{T}_{\text{train}},\,\boldsymbol{\mu}\in\mathcal{D}_{\text{train}}}}{\text{minimize}} \\ &\sum_{\boldsymbol{\mu}\in\mathcal{D}_{\text{train}},\, t\in\mathcal{T}_{\text{train}},\, \boldsymbol{X}\in\Omega_{\mathit{ref,train}}} (\|\boldsymbol{g}_\theta(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu})) - \boldsymbol{\phi}(\boldsymbol{X};t,\boldsymbol{\mu})\|_2^2 \\ &+ \lambda_1\|\nabla\boldsymbol{g}_\theta(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu})) - \nabla\boldsymbol{\phi}(\boldsymbol{X};t,\boldsymbol{\mu})\|_F^2 \\ &+ \lambda_2\|\frac{\partial \boldsymbol{g}_\theta}{\partial \hat{\boldsymbol{x}}}(\boldsymbol{X};\hat{\boldsymbol{x}}(t,\boldsymbol{\mu}))\dot{\hat{\boldsymbol{x}}}(t,\boldsymbol{\mu}) - \dot{\boldsymbol{\phi}}(\boldsymbol{X};t,\boldsymbol{\mu})\|_2^2),\end{aligned} \tag{7}$$

where $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}$, $\mathcal{T}_{\text{train}} \subseteq \mathcal{T}$, and $\Omega_{\mathit{ref,train}} \subseteq \Omega_{\mathit{ref}}$ denote parameter, time, and reference-domain instances at which the original (full-order) MPM has been solved and solutions are available, $\dot{\hat{\boldsymbol{x}}}(t,\boldsymbol{\mu}) = \frac{\hat{\boldsymbol{x}}(t+\Delta t,\boldsymbol{\mu})-\hat{\boldsymbol{x}}(t,\boldsymbol{\mu})}{\Delta t}$ is computed via explicit Euler with Δt being the time step size of MPM, and $\lambda_1,\ \lambda_2 \in \mathbb{R}_+$ denote penalty parameters for the deformation gradient and velocity, respectively.

Dynamics: velocity projection

To generate projection-based dynamics at each time instance, the proposed ROM (1) computes the velocity on grid nodes of the Eulerian mesh, and (2) performs least-squares projection of the computed velocity onto the tangent space to the low-dimensional manifold. However, to ensure the computational complexity is independent of the number of original material points n^p, we must restrict these steps to execute on only a subset of domain. Toward this end, we propose two separate hyper-reduction approaches.

Hyper-reduction approach #1: material-point projection The first approach we propose is *material-point centric*. We begin by identifying *a priori* a set $\mathcal{M} \subseteq \{1,\ldots,n^p\}$ of sample material via stochastic sampling. We also consider the neighbors of these material points $\mathcal{N} \subseteq \{1,\ldots,n^p\}$, defined as the set of material points needed for the computation of the dynamics of the sample material points for all $t \in \mathcal{T}$ and $\boldsymbol{\mu} \in \mathcal{D}$; note that $\mathcal{M} \cap \mathcal{N} = \emptyset$. With these material points identified, Algorithm 1 provides the associated steps taken by the resulting (online) reduced-order-model simulation.

While this approach can incur an operation count independent of the original number of material points n^p and Eulerian basis functions n^b, it requires computing (and tracking) the set of neighboring material points, which is difficult

Algorithm 1: ROM dynamics approach #1

```
Input: Generalized coordinates $\hat{x}(t_n)$ and velocities $\dot{\hat{x}}(t_n)$.
Output: Generalized coordinates $\hat{x}(t_{n+1})$ and velocities $\dot{\hat{x}}(t_{n+1})$.
1 Identify the grid nodes $\mathcal{I} \subseteq \{1,\ldots,n^b\}$ needed to
    compute dynamics for the sample material points,
    i.e., $\mathcal{I} = \{i \mid N_i(x_n^p) \neq 0$ for any $p \in \mathcal{M}\}$, where
    $N_i$ is the Eulerian basis function.
2 Compute the deformation gradient $F_n^p$, velocity $v_n^p$,
    and position $x_n^p$ for each sample and neighbor
    material point $p \in \mathcal{M} \cup \mathcal{N}$ at time instance $t_n$ by
    evaluating (4)–(6) for $X = X^p$, $p \in \mathcal{M} \cup \mathcal{N}$ and
    $t = t_n$.
3 Perform the 'particle to grid' transfer by computing
    for $i \in \mathcal{I}$
      $f_{i,n}^{\sigma} = -\sum_{p\in\mathcal{M}\cup\mathcal{N}} \frac{J(F_n^p)}{\rho_0}\sigma(F_n^p)\nabla_x N_i(x_n^p)\, m^p$
      $f_{i,n}^{e} = \sum_{p\in\mathcal{M}\cup\mathcal{N}} \frac{J(F_n^p)}{\rho_0} b(x_n^p) N_i(x_n^p)\, m^p,$
    where $J := \det(F)$, $\sigma$ denotes the Cauchy stress, $b$
    denotes body forces, $m^p$ is the material point mass,
    and $\rho_0$ is the initial material density.
4 Perform the update step by computing for $i \in \mathcal{I}$
      $\dot{v}_{i,n+1} = \frac{1}{m_i}(f_{i,n}^{\sigma} + f_{i,n}^{e})$
      $\Delta v_{i,n+1} = \dot{v}_{i,n+1}\Delta t_n$
      $v_{i,n+1} = v_{i,n} + \Delta v_{i,n+1}.$
5 Perform the 'grid to particle' transfer by computing
    for $p \in \mathcal{M} \cup \mathcal{N}$
      $v_{n+1}^p = \sum_{i\in\mathcal{I}} v_{i,n+1} N_i(x_n^p).$
6 Update the generalized coordinates
    $\hat{x}(t_{n+1}) = \hat{x}(t_n) + \Delta t_n \dot{\hat{x}}(t_{n+1})$, where $\dot{\hat{x}}(t_{n+1})$
    satisfies the minimization problem
      $\min_{\hat{x}(t_{n+1})} \sum_{p\in\mathcal{M}} \|\frac{\partial g}{\partial \hat{x}}(X^p;\hat{x}(t_n))\dot{\hat{x}}(t_{n+1}) - v_{n+1}^p\|_2^2.$
```

to do in practice. The next method mitigates this issue by adopting a sample-node-centric perspective.

Hyper-reduction approach #2: sample-node projection This section presents an alternative that leverages the fact that—if constructed to be injective—the manifold-parameterization function $\boldsymbol{g}$ can be inverted for positions in the reference domain given the position in the deformed configuration and values of the generalized coordinates. Thus, the key elements of this approach are: (1) no need to track individual material points, (2) the ability to generate effective material points as needed, and (3) performing classical finite-element quadrature on the Eulerian grid to assemble the governing equations.

Regarding quadrature, we first note that

$$\int_{\Omega} f\rho dV = \sum_{e=1}^{n_e} \int_{\Omega^e} f\rho dV \approx \sum_{e=1}^{n_e}\sum_{q=1}^{n_q} f(\boldsymbol{x}_q)\rho(\boldsymbol{x}_q^e)w_q.$$

where n_e is the number of elements, n_q is the number of quadrature points, and w_q is the quadrature weight. Consider approximating the integrals from the weak-form of the equation of motion using classical techniques as

$$\begin{aligned}&\sum_{e=1}^{n_e}\sum_{q=1}^{n_q}(\sum_j \rho\dot{\boldsymbol{v}}_j N_j\, N_i)|_{\boldsymbol{x}_q}\, w_q \\ &= -\sum_{e=1}^{n_e}\sum_{q=1}^{n_q}(\boldsymbol{\sigma}(\boldsymbol{F})\nabla N_i - \boldsymbol{b}N_i)|_{\boldsymbol{x}_q} w_q \\ &+ \int_{\partial\Omega} \bar{\boldsymbol{t}}N_i, \quad i = 1,\dots,n^b.\end{aligned} \tag{8}$$

In contrast to the first approach, this approach (Algorithm 2) does not require tracking any fixed material points nor their neighbors. Rather, by selecting in step 1 a set of sample nodes $\mathcal{I}$ and leveraging invertibility of the deformation map, the approach can incur an operation count independent of the original number of material points n^p and Eulerian basis functions n^b without any special tracking.

Algorithm 2: ROM dynamics approach #2

```
Input: Generalized coordinates $\hat{\boldsymbol{x}}(t_n)$ and velocities $\dot{\hat{\boldsymbol{x}}}(t_n)$.
Output: Generalized coordinates $\hat{\boldsymbol{x}}(t_{n+1})$ and velocities $\dot{\hat{\boldsymbol{x}}}(t_{n+1})$.
1 Select (any) sample nodes $\mathcal{I} \subseteq \{1,\dots,n^b\}$ of
    interest that satisfy $N_i(\boldsymbol{x}) \neq 0$ for some $\boldsymbol{x} \in \Omega$.
2 Define a quadrature rule comprising quadrature
    points and weights $\boldsymbol{x}_n^q \in \Omega$, $\boldsymbol{w}_n^q \in \mathbb{R}_+$,
    $q = 1,\dots,n^q$ used to assemble the governing
    equations at the sample nodes $\mathcal{I}$.
3 Compute the deformation gradient $\boldsymbol{F}_n^q$ and velocity
    $\boldsymbol{v}_n^q$ for each quadrature point $q = 1,\dots,n^q$ at time
    instance $t_n$ by evaluating (4)–(6) for $\boldsymbol{X} = \boldsymbol{X}_n^q$,
    $q = 1,\dots,n^q$ and $t = t_n$, where $\boldsymbol{X}_n^q$ satisfy
    $\boldsymbol{x}_n^q = \boldsymbol{g}(\boldsymbol{X}_n^q;\hat{\boldsymbol{x}}(t_n))$, $q = 1,\dots,n^q$.
4 Perform the 'particle to grid' transfer by computing
    for $i \in \mathcal{I}$
      $$\boldsymbol{f}_{i,n}^{\boldsymbol{\sigma}} = -\sum_{q=1}^{n^q}\frac{J(\boldsymbol{F}_n^q)}{\rho_0}\boldsymbol{\sigma}(\boldsymbol{F}_n^q)\nabla_{\boldsymbol{x}}N_i(\boldsymbol{x}_n^q)\, w^q$$
      $$\boldsymbol{f}_{i,n}^{e} = \sum_{q=1}^{n^q}\frac{J(\boldsymbol{F}_n^q)}{\rho_0}\boldsymbol{b}(\boldsymbol{x}_n^q)N_i(\boldsymbol{x}_n^q)\, w^q.$$
5 Perform the update step by computing for $i \in \mathcal{I}$
      $$\dot{\boldsymbol{v}}_{i,n+1} = \frac{1}{m_i}(\boldsymbol{f}_{i,n}^{\boldsymbol{\sigma}} + \boldsymbol{f}_{i,n}^{e})$$
      $$\Delta\boldsymbol{v}_{i,n+1} = \dot{\boldsymbol{v}}_{i,n+1}\Delta t_n$$
      $$\boldsymbol{v}_{i,n+1} = \boldsymbol{v}_{i,n} + \Delta\boldsymbol{v}_{i,n+1}.$$
6 Select (any) sample material points $\boldsymbol{x}_n^s$,
    $s = 1,\dots,n^s$ that satisfy the condition: if
    $N_i(\boldsymbol{x}_n^s) \neq 0$, then $i \in \mathcal{I}$.
7 If needed, perform the 'grid to particle' transfer by
    computing for $s = 1,\dots,n^s$
      $$\boldsymbol{v}_{n+1}^s = \sum_{i\in\mathcal{I}}\boldsymbol{v}_{i,n+1}N_i(\boldsymbol{x}_n^s),$$
    where $\boldsymbol{v}_n^s$ can be computed by evaluating Eq. (6) for
    $\boldsymbol{X} = \boldsymbol{X}_n^s$, $s = 1,\dots,n^s$, where $\boldsymbol{X}_n^s$ satisfies
    $\boldsymbol{x}_n^s = \hat{\boldsymbol{x}}(\boldsymbol{X}_n^s;\hat{\boldsymbol{x}}(t_n))$, $s = 1,\dots,n^s$.
8 Update the generalized coordinates
    $\hat{\boldsymbol{x}}(t_{n+1}) = \hat{\boldsymbol{x}}(t_n) + \Delta t_n\dot{\hat{\boldsymbol{x}}}(t_{n+1})$, where $\dot{\hat{\boldsymbol{x}}}(t_{n+1})$
    satisfies the minimization problem
      $$\min_{\dot{\hat{\boldsymbol{x}}}(t_{n+1})}\sum_{s=1}^{n^s}\|\frac{\partial\boldsymbol{g}}{\partial\hat{\boldsymbol{x}}}(\boldsymbol{X}^s;\hat{\boldsymbol{x}}(t_n))\dot{\hat{\boldsymbol{x}}}(t_{n+1}) - \boldsymbol{v}_{n+1}^s\|_2^2.$$
```

Numerical experiments

Unit tests

For unit testing, we shear the top of a cylinder made of elastic material for one timestep.

Training with gradient penalty In Table 1, we compare different training setups. With both λ_1 and λ_2 set to be zero, the network is trained only with the position information without first-order gradient information. While the position error and deformation gradient error of the reduced-order simulations are relatively small, a large velocity error is observed. Training with a positive λ_2 while keeping λ_1 zero, the network is now trained with additional temporal gradient information (velocity). Consequently, the velocity error is significantly improved, which also leads to significant improvement in the position error. However, a slight increase in the deformation gradient is observed. We thereby train with both the spatial (deformation gradient) and the temporal (velocity) gradient information by setting both λ_1 and λ_2 set to

Training Parameters	Position Error	Velocity Error	Deformation Gradient Error
$\lambda_1 = 0, \lambda_2 = 0$	0.19%	13%	0.2%
$\lambda_1 = 0, \lambda_2 = 0.01$	0.004%	0.3%	0.3%
$\lambda_1 > 10000, \lambda_2 > 0.01$	0.004%	0.2%	0.2%

Table 1: Training parameters' influence on the accuracy of the reduced-order simulations. Training with both spatial (λ_1) and temporal (λ_2) gradient penalty yields the best result. The material is discretized with 3 material points.

Projection Scheme	Position Error	Velocity Error	Deformation Gradient Error
Material-Point -Centric	0.01%	0.06%	0.11%
Sample-Node -Centric	0.01%	0.06%	0.11%
Material-Point -Centric, Hyper-reduction (6)	0.01%	0.16%	0.11%
Sample-Node -Centric, Hyper-reduction (6)	0.01%	0.13%	0.11%

Table 2: Projection approaches and hyper-reduction. Both projection approaches achieve the same level of accuracy with or without hyper-reduction. The material is discretized with 17 material points.

be positive. As expected, optimal errors across position, velocity, and deformation gradient are observed.

Material-point-centric projection vs. sample-node-centric projection In Table 2, we compare the reduced-order simulation's accuracy when using different projection methods. All the simulations use the same network with the same weights. Both the material-point-centric approach and the sample-node-centric approach achieve relatively small error.

Hyper-reduction Table 2 also demonstrates the effectiveness of hyper-reduction where we use only a small subset of the original material points for projection, which leads to cheaper computational cost. While the original number of material points is $n^p = 17$, using 6 for projection yields the same level of position error.

Poking test

An elastic cylinder, whose information is listed in Table 3, is poked at the top (Figure 1). The poking force is characterized by its spherical coordinate, $\boldsymbol{f}(\rho, \theta, \phi)$. We generate training and testing data by varying the three parameters characterizing the poking force, where $\rho \in [1.0, 1.2]$, $\theta \in [0°, 9°]$, and $\phi \in [10°, 15°]$. A total number of 27 simulations are generated via full factorial sampling, 22 of which for training while 5 for testing. Each simulation consists of 36 timesteps or 0.25s. Therefore, a total of 945 simulation snapshots is used for training and testing.

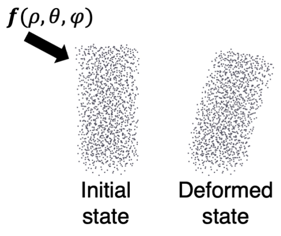

Figure 1: The material is poked at the top by different forces, resulting in different deformed state.

Specimen information	Value	Unit
Radius	10	cm
Height	40	cm
E	12500	Pa
ν	0.3	

Table 3: Poking test specimen information

There are $n^p = 1,368$ material points in the full-order simulation. The dimension of the generalized coordinate is chosen to be 5, effectively reducing the dimension of the simulation by a factor of 821. The number of material points for hyper-reduction is 15; these points are chosen from the portion where the poking force is applied, the bottom where the cylinder is fixed, as well as the middle section where the material is allowed to deform freely.

The reduced-order simulation is able to reproduce the training cases with a position error of 0.50% and is able to predict the testing cases with a 0.55% error.

Future work

In the future, we would like to extend our work to support plasticity, fracture, contact, and collision.

References

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

Lee, K.; and Carlberg, K. T. 2020. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics* 404: 108973.

Sulsky, D.; Zhou, S.-J.; and Schreyer, H. L. 1995. Application of a particle-in-cell method to solid mechanics. *Computer physics communications* 87(1-2): 236–252.