Boundary Graph Neural Networks for 3D Simulations

Andreas Mayr[1,†], **Sebastian Lehner**[1], **Arno Mayrhofer**[2], **Christoph Kloss**[2], **Sepp Hochreiter**[1,3], **Johannes Brandstetter**[1,‡,*]

[1] ELLIS Unit Linz & LIT AI Lab, Johannes Kepler University Linz, Linz, Austria
[2] DCS Computing GmbH, Linz, Austria
[3] Institute of Advanced Research in Artificial Intelligence (IARAI), Vienna, Austria
[†] mayr@ml.jku.at, [‡] brandstetter@ml.jku.at

Abstract

The abundance of data has given machine learning considerable momentum in natural sciences and engineering, though modeling of physical processes is often difficult. A particularly tough problem is the efficient representation of geometric boundaries. Triangularized geometric boundaries are well understood and ubiquitous in engineering applications. However, it is notoriously difficult to integrate them into machine learning approaches due to their heterogeneity with respect to size and orientation. In this work, we introduce an effective theory to model particle-boundary interactions, which leads to our new Boundary Graph Neural Networks (BGNNs) that dynamically modify graph structures to obey boundary conditions. The new BGNNs are tested on complex 3D granular flow processes of hoppers, rotating drums and mixers, which are all standard components of modern industrial machinery but still have complicated geometry. BGNNs are evaluated in terms of computational efficiency as well as prediction accuracy of particle flows and mixing entropies. BGNNs are able to accurately reproduce 3D granular flows within simulation uncertainties over hundreds of thousands of simulation timesteps. Most notably, in our experiments, particles stay within the geometric objects without using handcrafted conditions or restrictions.

1 Introduction

The deep learning revolution (Krizhevsky, Sutskever, and Hinton 2012) has dramatically changed scientific fields such as computer vision, natural language processing, or medical sciences. More recently, deep learning research has been expanded towards physical simulations such as fluid dynamics, deformable materials, or aerodynamics (Li et al. 2019; Ummenhofer et al. 2020; Sanchez-Gonzalez et al. 2020; Pfaff et al. 2021). Currently, the progress of deep learning for physical simulations is driven by Graph Neural Networks (GNNs) (Scarselli et al. 2009; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017). GNNs are very effective when modeling interactions between many entities via forward dynamics (Battaglia et al. 2018), and as such are a strong building block when it comes to the replacement of slower numerical simulations in various engineering disciplines. We focus on granular flows, which are ubiquitous in nature and consequently in industrial processes. The accurate simulations of such versatile granular flows forms the backbone of designing and improving industrial machinery. Complex boundaries are present in every-day's machinery such as rotating drums, mixers or hoppers. In engineering, these complex boundaries are typically modelled by triangularizations, which are mathematical well founded and for which efficient construction and simulation tools are available. Therefore, triangular meshes are standard for representing and modelling industrial machinery.

Effective theory. In this work, we introduce an effective theory to model particle-boundary interactions, from which we derive a new approach to accurately and effectively model granular flow processes within triangularized boundary surfaces. In physics, effective theories allow the description of phenomena within much simpler frameworks without a significant loss of precision. The basic idea is to approximate a physical system by factoring out the degrees of freedom that are not relevant in the given setting and problem to solve (e.g. using Newton's equations instead of the much more complicated Einstein's equations, or, using simple algebraic equations instead of numerically solving differential equations for particle-particle interactions). Other examples are in the fields of gravitational wave theory (Goldberger and Rothstein 2006), particle physics (Leutwyler 1994), hydrodynamics (Endlich et al. 2013), and, even in deep learning theory (Roberts, Yaida, and Hanin 2022). Figure 1 illustrates the effective theory of gravitational forces for planetary movement modeling, which motivates the introduction of effective particle-boundary interactions in this work.

We introduce Boundary Graph Neural Networks (BGNNs) as an effective model for complex 3D granular flows. We test the effectiveness of BGNNs on flow simulations within different triangularized geometries. The data for BGNN training is obtained by precise but potentially time-consuming simulations. BGNNs are able to generalize granular flow dynamics over thousands of timesteps while potentially being considerably faster than state-of-the-art simulation methods. The contributions of this paper are:

• We describe particle-surface interactions as an effective theory and introduce Boundary Graph Neural Networks (BGNNs) which enable dynamic modifications of graph structures.

[*] now at Microsoft Research AI4Science

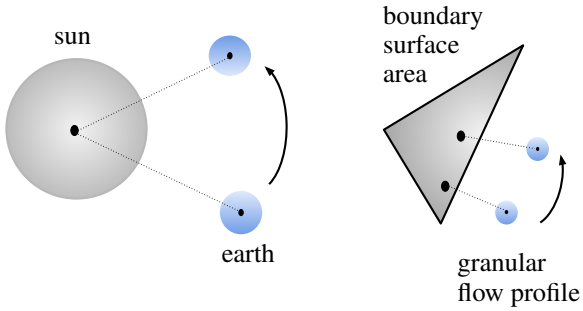

Figure 1: Effective theories of gravitational planetary movement (left), and particle-boundary interactions (right). Planetary movement is fully described by Einstein's field equations that relate mass and energy densities to the curvature of spacetime. A much simpler but in most cases sufficient description is to apply Newton's law of gravity to representative point masses. Black arrows indicate progress in time. Analogously, the interactions of granular flow particles and boundary surface areas is modeled by an effective two-point interaction.

• We implement BGNNs for 3D granular flow simulations of hoppers, rotating drums, and mixers as found in industrial machinery.

• We assess the performance of BGNNs via comparison of relevant physical quantities between model predictions and simulations.

2 Background

Graph Neural Networks. We consider graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E}$, where N-dimensional node features $\boldsymbol{p}_{v_i} \in \mathbb{R}^N$ are attached to each of the nodes. We use nearest neighbor graphs, assuming local interactions allow us to build arbitrary global dynamics. Therefore, whether an edge between a pair of nodes (v_i, v_j) is contained in the graph $\mathcal{G}$ depends on the distance between the nodes:

$$e_{ij} \in \mathcal{E} \iff d(v_i, v_j) \leqslant r_{\text{cut-off}}, \tag{1}$$

where the cut-off radius $r_{\text{cut-off}}$ is usually a hyperparameter of the model. Edges have M-dimensional edge features $\boldsymbol{a}_{ij} \in \mathbb{R}^M$ attached to each edge e_{ij}. Message passing networks (Gilmer et al. 2017) are a specific type of graph neural networks (Battaglia et al. 2018) and usually consist of three different types of layers: i) node and edge feature embedding layers, ii) the core message passing layers, and iii) read-out layers. Message passing iteratively updates the embeddings of edges ($\boldsymbol{m}_{ij}$) and nodes ($\boldsymbol{h}_i$), i.e., the embeddings of $\boldsymbol{a}_{ij}$ and $\boldsymbol{p}_{v_i}$, at edge e_{ij} and node v_i via:

$$\boldsymbol{m}'_{ij} = \phi(\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{m}_{ij}), \tag{2}$$

$$\boldsymbol{h}'_i = \psi\Big(\boldsymbol{h}_i, \square_{e_{ij}\in\mathcal{E}}\, \boldsymbol{m}'_{ij}\Big), \tag{3}$$

where the aggregation $\square_{e_{ij}\in\mathcal{E}}$ at node v_i in Eq. (3) is across all nodes that are connected to node v_i via an edge e_{ij}. Typically, $\square$ represents a mean or max operation. The learnable functions ϕ and ψ are commonly presented by Multilayer Perceptrons (MLPs). Equation (3) describes the computation and aggregation of messages, and the subsequent update of node embeddings. The final node embeddings are used for predictions via read-out layers. It is worth noting, that the general concept of GNNs often needs to be adapted to the actual purpose in mind, as e.g. for molecular modeling (Atz, Grisoni, and Schneider 2021; Yang et al. 2022; Reiser et al. 2022).

Dataset. Granular flow simulations are obtained by an Discrete Element Method (DEM) (Cundall and Strack 1979) which is similar to molecular dynamics. For granular flows, governing equations like the Navier-Stokes equations for fluid flows (Faccanoni and Mangeney 2013) do not exist. DEM represents the granular media by discrete particles (e.g. spheres or polyhedra), which interact by exchanging momentum via contact models. For granular flow simulation with DEM, we resort to the open-source software LIGGGHTS (Kloss et al. 2012, see Sect. TApp. A). LIGGGHTS can simulate particle flow for a wide range of materials and complex mesh-based wall geometries, therefore is well suited to simulate various industrial processes. In this work, training, validation, and test data are generated by LIGGGHTS modeling particle trajectories within different machinery designs.

Time transition model. Our method is based on Sanchez-Gonzalez et al. (2020), where we use the semi-implicit Euler method to numerically integrate the equations of motion with model-predicted acceleration. The time-transition from time t to time $t+1$ is given by $\dot{\boldsymbol{x}}^{t+1} = \dot{\boldsymbol{x}}^t + \Delta t\ \ddot{\boldsymbol{x}}^t$ and $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \Delta t\ \dot{\boldsymbol{x}}^{t+1}$, where $\boldsymbol{x}$ is the particle location, and $\dot{\boldsymbol{x}}$ the particle velocity. The time-transition $\boldsymbol{x}^{t+1}$ is calculated from the predicted particle acceleration $\ddot{\boldsymbol{x}}^t$.

3 Boundary Graph Neural Networks

Modeling approach. Our goal is to model time transition dynamics of particles in complex geometries with GNNs. The focus is on developing a proper representation of the triangularized geometries. An obvious and straightforward approach is to sample individual points from the boundaries as in Sanchez-Gonzalez et al. (2020) and Ummenhofer et al. (2020). In our setting we have to sample points from the triangles and then include them as non-kinematic particles with fixed positions in the graph. However, sampling is not feasible for large and complex geometries with many triangles. Therefore, we resort to an effective theory to make boundary representations efficient.

Effective theory for particle-surface interactions. In order to apply an effective theory to particle-boundary interactions, we have to determine the most important interaction properties that should be conserved. For this purpose, we will define an effective and dynamic graph, which changes for every timepoint. The graph has to accurately model:

• *Time awareness*: particle-boundary interactions should be modeled for as many timesteps as necessary. Particle-boundary interactions are represented in the graph as connections between surface areas and the particles in their proximity. Thus, particle-boundary interactions have to stay within a predefined cutoff radius as long as possible.

• *Capture of strongest interaction*: similar to Newton's law of gravity, we target a point-like representation for both particles and surface areas. Knowingly, the physical interaction strength decreases with increasing distance. Thus, effective particle-boundary interactions should contain the smallest distances between particles and surface areas.

Given these considerations, we model particle-boundary interactions by point-like particle-particle interactions, where the *virtual particles* representing the boundary surface area are placed such that the distance to the *real particles* is minimized. Consequently, real particles "see" different virtual particles from the same surface area. However, for every granular flow particle, we effectively model only one particle-surface interaction. We give a roadmap of what follows: (i) we introduce an efficient way of calculating shortest distances between real particles and triangularized surface areas, and (ii) we construct a dynamic graph model which models the time transition dynamics.

Calculation of shortest distances. In order to obtain shortest distances between real particles and triangularized surface areas, the squared distance between the particle center and the closest point on the mesh triangles is calculated (adopted from Eberly (1999)). We outline this in the following. A location on a triangle $\boldsymbol{t}$ is parameterized by two scalar values $u_0, u_1 \in \mathbb{R}$ with $\boldsymbol{t}(u_0, u_1) = \boldsymbol{b} + u_0\,\boldsymbol{e}_0 + u_1\,\boldsymbol{e}_1$, where $u_0 \geq 0$, $u_1 \geq 0$, and $u_0 + u_1 \leqslant 1$, $\boldsymbol{b}$ represents one of the nodes of the triangle, and, $\boldsymbol{e}_0$ and $\boldsymbol{e}_1$ are vectors from $\boldsymbol{b}$ towards the other two nodes (see Fig. TApp. B.1). The minimal Euclidean squared distance d of the point $\boldsymbol{p}$ to the triangle is given by the optimization problem:

$$d = \min_{u_0,u_1} \; q(u_0,u_1) = \|\boldsymbol{t}(u_0,u_1) - \boldsymbol{p}\|^2 \qquad (4)$$
$$\text{s.t.} \quad u_0 \geq 0\,, \;\; u_1 \geq 0\,, \;\; u_0 + u_1 \leqslant 1\,.$$

The minimizing arguments u_0' and u_1' parameterize the closest point $\boldsymbol{t}(u_0', u_1')$ of the triangle to the point $\boldsymbol{p}$. The algorithmic computation of this minimization problem is more involved and comprises seven cases, that need to be distinguished (see Sect. TApp. B). Whether a virtual particle is inserted is determined by Eq. (6) and the particle-triangle distance d (multiple inserts for multiple boundaries).

Boundary Graph Neural Networks (BGNNs). We associate each graph node v_i to a particle with location $\boldsymbol{x}_{v_i}$, velocity $\dot{\boldsymbol{x}}_{v_i}$ and acceleration $\ddot{\boldsymbol{x}}_{v_i}$, which is similar to Sanchez-Gonzalez et al. (2020). Additionally, we modify and enhance the graph structure to include boundaries (see Fig. 2). We dynamically add $\tilde{n}$ virtual nodes $\tilde{v}_j \in \tilde{\mathcal{V}}$ for boundary regions, iff the corresponding boundary region is within a cut-off radius to any other particle.

We augment the set of edges $e_{ij} \in \mathcal{E}$ by boundary edges $\tilde{e}_{ij} \in \tilde{\mathcal{E}}$ giving an enhanced edge set $\hat{\mathcal{E}} = \mathcal{E} \cup \tilde{\mathcal{E}}$. Analogously to Eq. (1), the existence of particle-particle edges e_{ij} and particle-boundary edges $\tilde{e}_{ij}$ is defined via:

$$e_{ij} \in \mathcal{E} \subseteq \hat{\mathcal{E}} \iff d(v_i, v_j) \leqslant r_{\text{cut-off}}\,, \qquad (5)$$
$$\tilde{e}_{ij} \in \tilde{\mathcal{E}} \subseteq \hat{\mathcal{E}} \iff \tilde{d}(v_i, \tilde{v}_j) \leqslant \tilde{r}_{\text{cut-off}}\,. \qquad (6)$$

The cut-off radii $r_{\text{cut-off}}$ and $\tilde{r}_{\text{cut-off}}$ need not necessarily be the same, and, $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$, while $\tilde{d} : \mathcal{V} \times \tilde{\mathcal{V}} \to \mathbb{R}$, i.e. bidirectional edges are used between real nodes and unidirectional edges are used between real and virtual nodes. To include information about boundary surfaces into particle-boundary interactions, $\tilde{N}$-dimensional node features that encode information about the inclination of triangles are concatenated with the existing node features $\boldsymbol{p}_{v_i} \in \mathbb{R}^N$. Additionally, coordinate information is used both for existing nodes ($\boldsymbol{X} = \{\boldsymbol{x}_{v_0}, \ldots, \boldsymbol{x}_{v_{n-1}}\}$) as well as for virtual nodes ($\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{x}}_{\tilde{v}_0}, \ldots, \tilde{\boldsymbol{x}}_{\tilde{v}_{\tilde{n}-1}}\}$). For virtual nodes, the additional coordinates $\tilde{\boldsymbol{x}}_{\tilde{v}_j}$ are chosen such that they minimize the distance between points from boundaries and real particles. The resulting set of node features $\hat{\boldsymbol{P}}$ and node coordinates $\hat{\boldsymbol{X}}$ are:

$$\hat{\boldsymbol{P}} = \{\boldsymbol{p}_{v_0}, \ldots, \boldsymbol{p}_{v_{n-1}}, \tilde{\boldsymbol{p}}_{\tilde{v}_0}, \ldots, \tilde{\boldsymbol{p}}_{\tilde{v}_{\tilde{n}-1}}\}, \qquad (7)$$
$$\hat{\boldsymbol{X}} = \{\boldsymbol{x}_{v_0}, \ldots, \boldsymbol{x}_{v_{n-1}}, \tilde{\boldsymbol{x}}_{\tilde{v}_0}, \ldots, \tilde{\boldsymbol{x}}_{\tilde{v}_{\tilde{n}-1}}\}\,, \qquad (8)$$

where $\hat{\boldsymbol{p}}_i \in \mathbb{R}^{N+\tilde{N}}$ and $\hat{\boldsymbol{x}}_i \in \mathbb{R}^3$ denote the elements of $\hat{\boldsymbol{P}}$ and $\hat{\boldsymbol{X}}$, respectively. Similarly to above, message passing updates the embeddings of edges ($\hat{\boldsymbol{m}}_{ij}$) and the embeddings of nodes ($\hat{\boldsymbol{h}}_i$) via

$$\hat{\boldsymbol{m}}'_{ij} = \hat{\phi}\Big(\hat{\boldsymbol{h}}_i, \hat{\boldsymbol{h}}_j, \hat{\boldsymbol{m}}_{ij}\Big)\,, \qquad (9)$$
$$\hat{\boldsymbol{h}}'_i = \hat{\psi}\left(\hat{\boldsymbol{h}}_i, \square_{\hat{e}_{ij} \in\, \hat{\mathcal{E}}}\, \hat{\boldsymbol{m}}'_{ij}\right)\,, \qquad (10)$$

where the aggregation $\square_{\hat{e}_{ij} \in\, \hat{\mathcal{E}}}$ at node v_i in Eq. (10) is across all real or virtual nodes that are connected to v_i via an edge $\hat{e}_{ij}$. Similar to Gilmer et al. (2017) and Satorras, Hoogeboom, and Welling (2021), we make use of pairwise distances ($\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|^2$ and $\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j$ and deterministic functions thereof). These are for BGNNs between real and between real and virtual particles and we pass this information to the graph network as edge attributes $\hat{\boldsymbol{a}}_{ij}$, for which an initial edge embedding $\hat{\boldsymbol{m}}_{ij}$ is determined via an edge embedding layer. The final node embeddings are used for the predictions via the read-out layers. For aggregation $\square$, we use the mean.

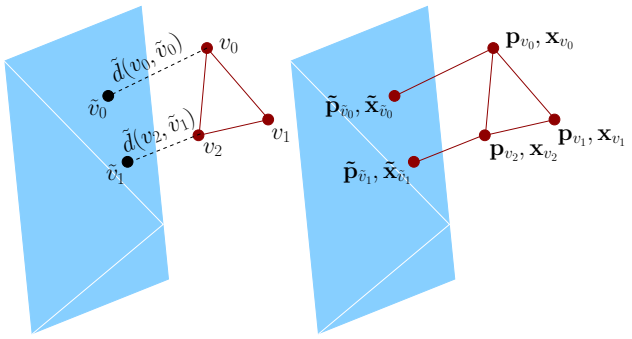

Figure 2: Dynamic modification of the graph edges (red lines) and nodes (red points). Left: Calculation of the distances $\tilde{d}(v_0, \tilde{v}_0)$, $\tilde{d}(v_2, \tilde{v}_1)$ between real particle at nodes v_0, v_2 and the triangles corresponding to virtual particle nodes $\tilde{v}_0$, $\tilde{v}_1$. Right: Insertion of an additional edge between $\tilde{v}_0$ and v_0 and between $\tilde{v}_1$ and v_2 and representation of the nodes in terms of the corresponding node features $\mathbf{p}_{v_i}$, $\mathbf{x}_{v_i}$ and $\tilde{\mathbf{p}}_{\tilde{v}_j}$, $\tilde{\mathbf{x}}_{\tilde{v}_j}$ for real and virtual nodes.

Dynamical graph model. At each time point a graph of the current scene is built up, containing the minimum distances between particles and walls as well as distances between particles within certain neighborhoods. The definition of the graph and computations on it make up our effective theory. Especially, every particle "sees" at most one virtual particle representing the boundary surface area, namely that virtual particle which has the shortest distance. Table 1 shows average numbers of nodes $\left|\mathcal{V}\right|$, as well as average numbers of boundary edges $\left|\tilde{\mathcal{E}}\right|$ and the relative increase in edges (ratio of the number of added wall edges to the total number of particle edges). The scalability of BGNNs would suffer if more than one particle per particle-boundary interaction surface was considered. Our approach is summarized in Algorithm 1.

Algorithm 1: BGNN: Dynamic Graph Message Passing

1: $\text{fr}_{s,t} \leftarrow$ random frame from trajectory s at time t
2: Assign node v_i to each real particle at $\boldsymbol{x}_{v_i}$ with features $\boldsymbol{p}_{v_i}$ and calculate pairwise distances $d_{ij} = d(v_i, v_j)$ between nodes v_i and v_j. Assign edges e_{ij} with edge features to those fulfilling Eq. (5).
3: Calculate $\tilde{d}_{ij} = q\left(u'_{0;(v_i,\triangle_j)}, u'_{1;(v_i,\triangle_j)}\right)$ for all particles (v_i) and triangles $(\triangle_j)$ according to Eq. (4). In case Eq. (6) is fulfilled, insert (a) a virtual node $\tilde{v}_j$ at $\tilde{\boldsymbol{x}}_{\tilde{v}_j} = \boldsymbol{t}\left(u'_{0;(v_i,\triangle_j)}, u'_{1;(v_i,\triangle_j)}\right)$ with associated triangle-specific features according to Sect. TApp. C, and (b) an edge between the real and the inserted virtual node together with associated edge features.
4: Fill up empty triangle-specific real-node features $\boldsymbol{p}_{v_i}$ and particle-specific virtual-node features $\tilde{\boldsymbol{p}}_{\tilde{v}_j}$ with null values and add indicator features for node types (real/virtual) to obtain $\hat{\boldsymbol{p}}_i$ and $\hat{\boldsymbol{p}}_j$.
5: Apply BGNN message passing according to Eqs. (9) and (10).
6: Update positions via semi-implicit Euler method.

Experiment	$\left\|\mathcal{V}\right\|$	$\left\|\tilde{\mathcal{E}}\right\|$	% increase
Hopper	1113 ± 738	5475 ± 3547	72.2
Drum	3283 ± 282	1678 ± 188	54.8

Table 1: Growth of the number of edges due to boundaries in the graph. The table shows statistics across the training trajectories of non-cohesive particles in a standard setting in hopper and drum experiments. For each trajectory the frame with maximum relative increase in the number of edges due to virtual particles $\left|\tilde{\mathcal{E}}\right|/\left|\mathcal{E}\right|$ has been selected as a representative frame. This is done since we are interested in the maximum effect additional virtual particles have on the memory requirements. Number of particles $\left|\mathcal{V}\right|$, number of additional virtual edges $\left|\tilde{\mathcal{E}}\right|$, and % increase are listed.

Boundary normal directions. Typical granular flow simulations comprise substantially more particle-particle interactions than particle-boundary interactions, which may impede the learning of particle-boundary interactions. In Kipf et al. (2018) the problem of qualitatively different interactions is addressed by introducing a dedicated message generating network for each interaction type. We avoid such extensions of our model by means of the following two approaches. First, we introduce additional node features, such that the neural network is able to distinguish the different types of nodes. Second, we adapt the weight initialization of the node feature embedding $\hat{\psi}$, such that the embedding network can be trained with larger values for the additional features. Consequently, the network can learn different dynamics for particle-particle and particle-boundary interactions. The additional node features are: (i) type feature, i.e., a binary indicator of whether a node represents a particle that is real or virtual, and, in the latter case, (ii) the components of the normal vector (see Sect. TApp. C for more information on an orientation-independent representation of the normal vectors) of the triangular surface areas (null vectors for real particles).

4 Related Work

There is a rich body of literature on applications of Deep Learning in the context of physics simulations. Most notably related to BGNNs are the works of Sanchez-Gonzalez et al. (2020), Ummenhofer et al. (2020), and, Li et al. (2019), all of which propose methods of learning particle simulations without enforcing constraints. These approaches can be contrasted to works like Ladickỳ et al. (2015) or Schenck and Fox (2018) that utilize strong inductive biases. Ladickỳ et al. (2015) construct features for Random Forest Regression that are influenced by Smooth Particle Hydrodynamics (Gingold and Monaghan 1977; Lucy 1977). Schenck and Fox (2018) construct a differentiable fluid dynamics network that is closely related to the Position Based Fluids method (Macklin and Müller 2013). Importantly, both methods are built on the assumption that the governing equations of the system are known, which is, as mentioned in Sect. 2, not necessarily the case for granular flow dynamics.

Integrating finite element methods and therefore triangularized boundaries into deep learning architectures has started to gain interest (Longo et al. 2022). Complex mesh-based wall geometries have been employed to compute updates for nodes of the mesh itself (Pfaff et al. 2021). In contrast to Pfaff et al. (2021) in our scenario, the mesh is static, i.e. the descriptive representation of machine parts. We share the opinion of Sanchez-Gonzalez et al. (2020) that the network architecture with continuous convolutions as suggested by Ummenhofer et al. (2020) can be interpreted as GNNs. In doing so, a difference to Sanchez-Gonzalez et al. (2020) and our work is that Ummenhofer et al. (2020) use static particles as special nodes in the first message passing step only. Consequently, the framework of Sanchez-Gonzalez et al. (2020), which is based on Battaglia et al. (2018), appears to be the most general to us, performing well even without explicit hierarchical clustering as suggested in

DPI-Net (Li et al. 2019). Experiments of Sanchez-Gonzalez et al. (2020) further suggest that their simulation of sand particles are superior to the implementation of Ummenhofer et al. (2020). However, Sanchez-Gonzalez et al. (2020) only consider simple cuboid boundaries for their 3D simulations, leaving more realistic complex geometries as an open and yet untouched challenge. Furthermore, they use sampled, static particles to represent boundaries for 2D simulations, which in general does not scale well for 3D simulations due to the quadratic increase of boundary particles (square areas instead of lines).

5 Experiments

We test the effectiveness of BGNNs on complex 3D granular flow simulations. The development, design, and construction of many mechanical devices is based on granular flow simulations. These devices can have very different geometries and must be designed for a wide range of materials with highly varying properties. For example, cohesion properties can range from dry, wet, to oily. In the simulations, we consider very common device geometries and different cohesion properties, as well as static and moving geometries, to cover a wide range of situations with our available computational resources. The two common geometries are hoppers and rotating drums (see Figs. 3 and 4 and TApp. A.1). The two different cohesion properties are non-cohesive describing liquid-like, oily materials and cohesive describing dry, sand-like materials. We compare the BGNN predictions to the simulations in two aspects: speed and accuracy.

Simulation Details. For all experiments, gravitation acts along the z-direction. The upper part of the hopper is delimited along the y-axis by two planes, which are parallel to the x-z plane (see Fig. 3). The x-axis is delimited by two planes, that are inclined at certain angles α, $180° - \alpha$ to the x-y plane and at corresponding angles $\alpha - 90°, 90° - \alpha$ to the y-z plane. The hopper has an initially closed hole at the bottom, which has an adjustable radius. The rotation axis of the drum is the y-axis (see Fig. 4). The initial filling of the hopper and drum is done by randomly inserting particles into a predefined region, see Sect. TApp. D. We use around 1000 and around 3000 particles for hopper and rotating drum simulations, respectively. In order to have trajectories with non-cohesive and cohesive particles, we use the simplified JKR model (Roessler and Katterfeld 2019) with a cohesion energy density of 0 J/m^3 and 10^5 J/m^3 for non-cohesive and cohesive particles. The training data consists of 30 simulation trajectories, where each trajectory consists of 100.000 (250.000) simulation timesteps for hopper (rotating drum). For BGNN training every 40 (100)-th timestep is used. Trajectories have different angles α and different hole radii (hopper) and different initial particle placement (drum). Moreover, the number of particles is varied by $\pm 25\%$.

Implementation Details. We use 3 to 10 message passing layers, with 128 and 512 nodes for intermediate node and edge representation. The cut-off radii strongly depend on the particle size. We use cut-off radii of 0.02 and 0.008 for rotating drum and hopper, respectively. Cut-off radii have been treated as hyperparameters of our model. More details can be found in Sect. TApp. D.

Assessment Of Physical Quantities. Granular flow simulations should correctly describe systems on macroscopic scales in terms of *particle-averaged positions* $\bar{\mathbf{x}}(t)$ and *particle flows* $\bar{\mathbf{v}}(t)$ for n particles as a function of time: $\bar{\mathbf{x}}(t) = \frac{1}{n}\sum_i \mathbf{x}_i(t)$ and $\bar{\mathbf{v}}(t) = \frac{1}{n}\sum_i \mathbf{v}_i(t)$. Hoppers are devices that aim at adjusting the flow of particles along the direction of gravity, which coincides with the z-axis in our experiments. Rotating drums are commonly utilized as mixing devices for various applications in e.g. industry, research, and agriculture. They are essentially rotating cylinders that are partially filled with a granular material. The mixing property of these devices is a result of numerous particle interactions under time-varying boundary conditions. For rotating drum experiments, we quantify the extend of particle mixing via the *mixing entropy* (Lai and Fan 1975). If the z-coordinate of a particle's initial position $\mathbf{x}_i(0)$ is above (below) the median z-coordinate of all particles in the initial state, we assign it to class $c = +1$ (-1). Based on this assignment local entropies $s(\boldsymbol{g}_{klm}, t)$ at grid cells $\boldsymbol{g}_{klm}$ are calculated, where the indices klm identify an individual grid cell. The local entropies $s(\boldsymbol{g}_{klm}, t)$ are computed from particle counts $n_c(\boldsymbol{g}_{klm}, t)$, of the respective classes $c = \pm 1$. The total number of particles in a grid cell is obtained by $n(\boldsymbol{g}_{klm}, t) = n_{+1}(\boldsymbol{g}_{klm}, t) + n_{-1}(\boldsymbol{g}_{klm}, t)$. Calculating the particle-number weighted average of the local mixing entropies yields the mixing entropy $\mathrm{S}(t)$ of the entire system:

$$\mathrm{S}(t) = \frac{-\sum_{klm}\sum_{c=\pm 1} n(\boldsymbol{g}_{klm}, t)\left(f_c(\boldsymbol{g}_{klm}, t)\log f_c(\boldsymbol{g}_{klm}, t)\right)}{\sum_{klm} n(\boldsymbol{g}_{klm}, t)},$$

where $f_c(\boldsymbol{g}_{klm}, t)$ denotes the relative fraction of class c particles in cell $\boldsymbol{g}_{klm}$ at time t.

Results. In Fig. 3 and Fig. 4 results for the hopper and the rotating drum simulations are presented. The upper parts visualize granular flow snapshots at different time steps, both for cohesive and non-cohesive materials. The lower parts of the figures include average position and particle flow plots for hopper, as well as particle flow and mixing entropy plots for rotating drum simulations. The simulation uncertainties arise due to the different distributions of the initial filling and due to a $\pm 25\%$ variation in the number of particles across simulations. The difference between cohesive and non-cohesive particles is evident. BGNNs have learned to model granular flow simulations over thousands of time steps. Most notably, hardly any particle leaves the geometric boundaries. This is achieved without using handcrafted conditions or restrictions on the positions of the particles. Furthermore, BGNNs have learned to model particle-boundary interactions and in doing so correctly represent the dynamics within the system. The predicted quantities are within uncertainties of the simulations.

Out-of-distribution generalization. Therefore, we consider the BGNN predictions as sufficiently precise to substitute the simulations. Figure 5 shows out-of-distribution (OOD) scenarios, where the devices are changed with respect to the training data. The hole size of the hopper is

decreased in mean by $\sim 50\%$, while side wall inclination angles have been increased by $\sim 15°$. For the drum the

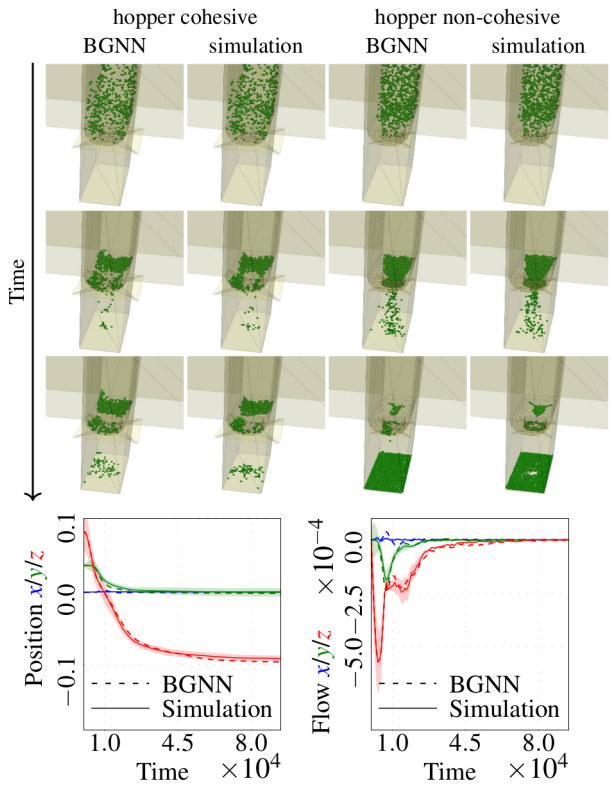

Figure 3: Hopper dynamics. Top: Distributions for cohesive and non-cohesive particles. Simulation data and BGNN predictions are compared. Particles are indicated by green spheres, triangular wall areas are yellow, the edges of these triangles are indicated by grey lines. In contrast to liquid-like non-cohesive particles, cohesive particles lead to congestion of the hopper. Bottom: Position (left) and flow profile (right) for non-cohesive particles. Corresponding plots for cohesive particles can be found in Sect. TApp. D. Simulation data (solid lines) and BGNN predictions (dashed lines) are compared. Simulation uncertainties are due to a change of the particle numbers ($\pm 25\%$) and to different initial conditions. We provide simulation predictions for a hopper with more timesteps in animations at https://ml-jku.github.io/bgnn/.

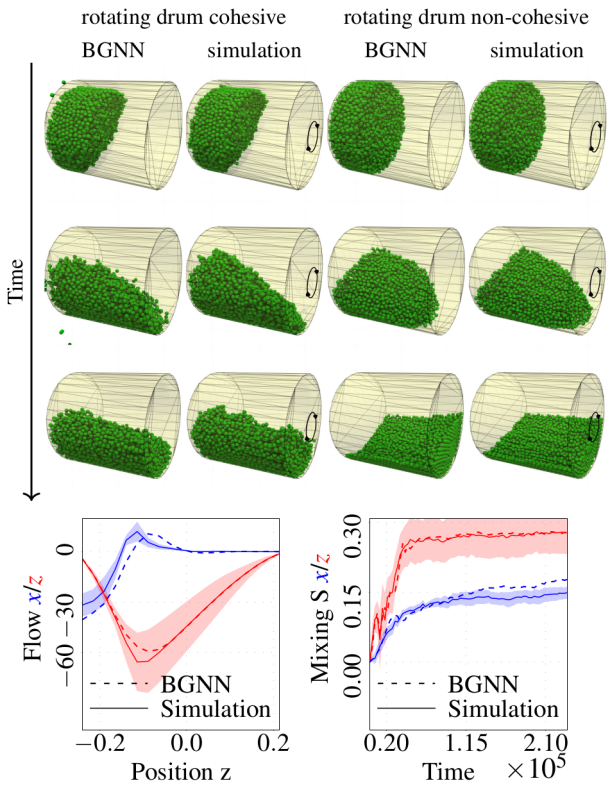

Figure 4: Rotating drum dynamics. Top: Particle distributions for cohesive and non-cohesive particles. Simulation data and BGNN predictions are compared. Particles are indicated by green spheres, triangular wall areas are yellow, the edges of these triangles are indicated by grey lines. The circular arrow indicates the rotation direction of the drum. In contrast to liquid-like non-cohesive particles, cohesive particles stick together much stronger. Bottom: Flow profile (left) and entropy plot (right) for non-cohesive particles. The entropy is shown for particle class assignment according to the x (blue) and z (red) position. Corresponding plots for cohesive particles can be found in Sect. TApp. D. Simulation data (solid lines) and BGNN predictions (dashed lines) are compared. Simulation uncertainties are due to a change of the particle numbers ($\pm 25\%$) and to different initial conditions. We provide simulation predictions for a rotating drum with more timesteps in animations at https://ml-jku.github.io/bgnn/.

length of the corresponding cylinder was increased in mean by $\sim 50\%$. Our experiments show that our model generalizes well across variations in the geometry. This finding demonstrates that trained BGNNs could be used for designing and studying different geometries without retraining the model.

Moving geometries. As an additional challenge we consider moving geometries, as shown in Fig. 6, where additional difficulty is imposed due to a rotating blade inside a particle mixer. Consequently, not only the geometry but also the blade itself are triangularized and particle-surface interactions are extended by particle-blade interactions. Experiments show that our BGNN approach is well suited to model such scenarios of increased difficulty.

Runtime Table 2 gives a run-time comparison of the LIGGGHTS simulation versus a forward pass of BGNNs, which only predict every 100 time step. The highly optimized CPU algorithm (LIGGGHTS) and a non-optimized GPU compatible algorithm (BGNNs) are compared via their wall-clock times since the hardware settings are quite different. Nevertheless, Table 2 shows that the wall-clock time of BGNNs is shorter than the wall-clock time of the simulation. The usage of more particles, would further increase the lead of BGNNs over the simulation in terms of wall-clock time. For the time comparison, we use a typical simulation tra-

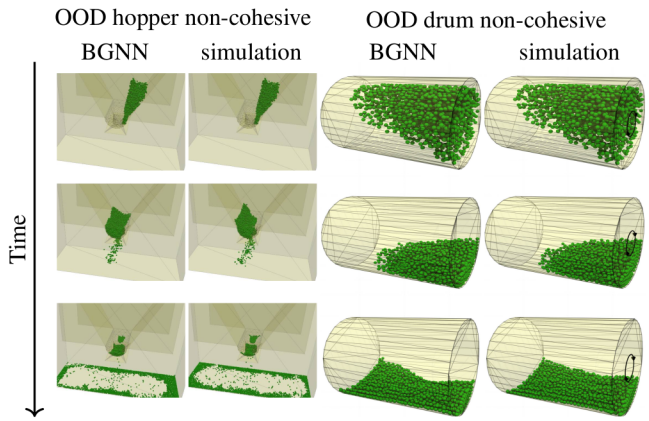

Figure 5: OOD generalization behavior for the hopper (left) and the rotating drum (right). In contrast to the training and validation data the outlet size of the hopper was decreased, the inclination angles of the hopper side walls are enlarged, and, the length of the rotating drum is increased.

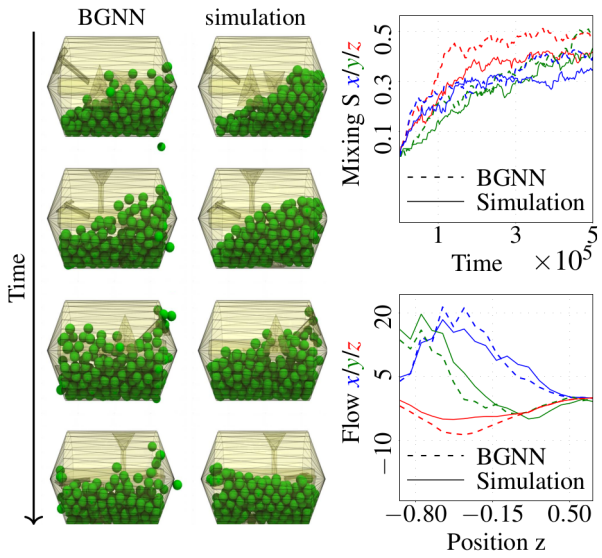

Figure 6: Mixer dynamics. A mixer can be seen as a moving geometry where additional difficulty is imposed due to rotating blades inside. BGNNs are also well suited to model such scenarios of increased difficulty.

jectory from our datasets with 3,408 particles, which needs approximately 2 GB GPU memory for one forward pass. An essential reason for speedup in our (simple) setups are GPU parallelization capabilities. There is potentially even more space for improvement of the BGNN predictions over simulations due to the so called *Young's modulus*. For simulations, it is often assumed that energy is purely transmitted through Rayleigh waves. Thus the time step of DEM simulations is targeted to be a fraction of the propagation time through a single, solid particle. As such the propagation time depends on material parameters, most notably the Young's modulus. However, for several materials the Young's moduli that reflect the true material properties, would lead to extremely small propagation times, which in turn means much more simulation steps. Consequently, much smaller Young's moduli are considered as an approximation, which is valid for gravity driven flows (Coetzee 2017). However, for many cases, e.g. the penetration of a particle bed by an object, this approximation breaks down (Lommen, Schott, and Lodewijks 2014). BGNNs have the potential to be trained on very small time steps reflecting the true Young's moduli and consequently generalize over much more than "just" 40 or 100 time steps.

method	time steps	real world time	wall-clock time [s]
LIGGGHTS	250.000	$12.5s$	356
BGNNs	2.500	$12.5s$	158

Table 2: Runtime comparison for one granular flow process (rotating drum) consisting of 250.000 simulation timesteps, which are 2500 BGNN predictions. For both, simulation and BGNN trajectories, this corresponds to a real world time of $12.5s$. LIGGGGHTS simulation is run on a CPU AMD EPYC 7H12, BGNN forward pass is run on a GPU NVIDIA A100.

6 Conclusion and Future Directions

We have introduced an effective theory to model complex particle-boundary interactions, resulting in Boundary Graph Neural Networks (BGNNs). BGNNs dynamically modify graph structures via modifying edges, augmenting node features, and dynamically inserting virtual nodes. BGNNs achieve an accurate neural network modeling of simulated physical processes within complex geometries. We have tested BGNNs on complex 3D granular flow processes of hoppers, rotating drums, and mixers, where BGNNs are able to accurately reproduce these flows within simulation uncertainties over hundreds of thousands of timesteps. Most notably particles stay within the geometric objects without using handcrafted conditions or restrictions. A possible extension of our work is towards a wide range of different materials, e.g. materials with high Young's moduli as described in Sect. 5. Another interesting extension is to introduce a velocity dependent cut-off radius, and in doing so considering also those particle-boundary interactions which are about to happen within the next timesteps although the spatial distances are still large. Finally, leveraging the symmetries and geometries of granular flow problems (Brandstetter et al. 2022a,b) is appealing.

Further Information

Code is available at https://ml-jku.github.io/bgnn/, technical appendix at https://arxiv.org/abs/2106.11299.

Acknowledgments

This research was supported by FFG grant 871302 (DL for GranularFlow).

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, TÜV Austria, Frauscher Sensonic and the NVIDIA Corporation.

We thank Angela Bitto-Nemling, Markus Holzleitner, and, Günter Klambauer for helpful discussions and comments on this work.

References

Atz, K.; Grisoni, F.; and Schneider, G. 2021. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 3(12): 1023–1032.

Battaglia, P.; Hamrick, J.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C.; Song, F.; Ballard, A.; Gilmer, J.; Dahl, G.; Vaswani, A.; Allen, K.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Brandstetter, J.; Hesselink, R.; van der Pol, E.; Bekkers, E.; and Welling, M. 2022a. Geometric and Physical Quantities improve E(3) Equivariant Message Passing. In *10th International Conference on Learning Representations*. OpenReview.net.

Brandstetter, J.; van den Berg, R.; Welling, M.; and Gupta, J. 2022b. Clifford neural layers for pde modeling. *arXiv preprint arXiv:2209.04934*.

Coetzee, C. 2017. Calibration of the discrete element method. *Powder Technology*, 310: 104–142.

Cundall, P.; and Strack, O. 1979. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1): 47–65.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Eberly, D. 1999. Distance Between Point and Triangle in 3D. Technical report, Geometric Tools.

Endlich, S.; Nicolis, A.; Porto, R.; and Wang, J. 2013. Dissipation in the effective field theory for hydrodynamics: First-order effects. *Physical Review D*, 88: 105001.

Faccanoni, G.; and Mangeney, A. 2013. Exact solution for granular flows. *International Journal for Numerical and Analytical Methods in Geomechanics*, 37(10): 1408–1433.

Gilmer, J.; Schoenholz, S.; Riley, P.; Vinyals, O.; and Dahl, G. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1263–1272. PMLR.

Gingold, R.; and Monaghan, J. 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3): 375–389.

Goldberger, W.; and Rothstein, I. 2006. Effective field theory of gravity for extended objects. *Physical Review D*, 73: 104029.

Kipf, T.; Fetaya, E.; Wang, K.; Welling, M.; and Zemel, R. 2018. Neural Relational Inference for Interacting Systems. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80, 2688–2697. PMLR.

Kipf, T.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*. OpenReview.net.

Kloss, C.; Goniva, C.; Hager, A.; Amberger, S.; and Pirker, S. 2012. Models, algorithms and validation for opensource DEM and CFD–DEM. *Progress in Computational Fluid Dynamics, an International Journal*, 12: 140–152.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Ladickỳ, L.; Jeong, S.; Solenthaler, B.; Pollefeys, M.; and Gross, M. 2015. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics*, 34(6): 1–9.

Lai, F.; and Fan, L. 1975. Application of a Discrete Mixing Model to the Study of Mixing of Multicomponent Solid Particles. *Industrial & Engineering Chemistry Process Design and Development*, 14(4): 403–411.

Leutwyler, H. 1994. On the Foundations of Chiral Perturbation Theory. *Annals of Physics*, 235(1): 165–203.

Li, Y.; Wu, J.; Tedrake, R.; Tenenbaum, J.; and Torralba, A. 2019. Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids. In *7th International Conference on Learning Representations*. OpenReview.net.

Lommen, S.; Schott, D.; and Lodewijks, G. 2014. DEM speedup: Stiffness effects on behavior of bulk material. *Particuology*, 12: 107–112.

Longo, M.; Opschoor, J.; Disch, N.; Schwab, C.; and Zech, J. 2022. De Rham compatible Deep Neural Networks. *arXiv preprint arXiv:2201.05395*.

Lucy, L. 1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82: 1013–1024.

Macklin, M.; and Müller, M. 2013. Position Based Fluids. *ACM Transactions on Graphics*, 32(4): 1–12.

Pfaff, T.; Fortunato, M.; Sanchez-Gonzalez, A.; and Battaglia, P. 2021. Learning Mesh-Based Simulation with Graph Networks. In *9th International Conference on Learning Representations*. OpenReview.net.

Reiser, P.; Neubert, M.; Eberhard, A.; Torresi, L.; Zhou, C.; Shao, C.; Metni, H.; van Hoesel, C.; Schopmans, H.; Sommer, T.; and Friederich, P. 2022. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1): 1–18.

Roberts, D.; Yaida, S.; and Hanin, B. 2022. *The Principles of Deep Learning Theory*. Cambridge University Press.

Roessler, T.; and Katterfeld, A. 2019. DEM parameter calibration of cohesive bulk materials using a simple angle of repose test. *Particuology*, 45: 105–115.

Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. 2020. Learning to Simulate Complex Physics with Graph Networks. In III, H.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119, 8459–8468. PMLR.

Satorras, V.; Hoogeboom, E.; and Welling, M. 2021. E(n) Equivariant Graph Neural Networks. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139, 9323–9332. PMLR.

Scarselli, F.; Gori, M.; Tsoi, A.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Schenck, C.; and Fox, D. 2018. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. In Billard, A.; Dragan, A.; Peters, J.; and Morimoto, J., eds., *Proceedings of The 2nd Conference on Robot Learning*, volume 87, 317–335. PMLR.

Ummenhofer, B.; Prantl, L.; Thuerey, N.; and Koltun, V. 2020. Lagrangian Fluid Simulation with Continuous Convolutions. In *8th International Conference on Learning Representations*. OpenReview.net.

Yang, R.; McCandler, C.; Andriuc, O.; Siron, M.; Woods-Robinson, R.; Horton, M.; and Persson, K. 2022. Big Data in a Nano World: A Review on Computational, Data-Driven Design of Nanomaterials Structures, Properties, and Synthesis. *ACS Nano*, 16(12): 19873–19891.