LEARNING 3D GRANULAR FLOW SIMULATIONS

Andreas Mayr[*] **Sebastian Lehner**[*] **Arno Mayrhofer**[†] **Christoph Kloss**[†]
Sepp Hochreiter[*,‡] **Johannes Brandstetter**[*,§]
[*]ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning,
Johannes Kepler University Linz, Austria
[†]DCS Computing GmbH, Linz, Austria
[‡]Institute of Advanced Research in Artificial Intelligence (IARAI)
[§]University of Amsterdam, Amsterdam, Netherlands

ABSTRACT

Recently, the application of machine learning models has gained momentum in natural sciences and engineering, which is a natural fit due to the abundance of data in these fields. However, the modeling of physical processes from simulation data without first principle solutions remains difficult. Here, we present a Graph Neural Networks approach towards accurate modeling of complex 3D granular flow simulation processes created by the discrete element method LIGGGHTS and concentrate on simulations of physical systems found in real world applications like rotating drums and hoppers. We discuss how to implement Graph Neural Networks that deal with 3D objects, boundary conditions, particle - particle, and particle - boundary interactions such that an accurate modeling of relevant physical quantities is made possible. Finally, we compare the machine learning based trajectories to LIGGGHTS trajectories in terms of particle flows and mixing entropies.

1 INTRODUCTION

Granular flows are ubiquitous in nature and in a large array of industrial processes. Pharmaceutical powders, plastic granulate and rocks obtained by mining are just some examples of granular media that are used in industries and which are processed in a multitude of different flow states. While attempts have been made to formulate governing equations for granular flows, like the Navier-Stokes equations for fluid flow, they have so far eluded a general framework (Faccanoni & Mangeney, 2013). More recently, research focus has been expanded towards applications of Deep Learning in the simulation of physical domains, such as fluid dynamics, deformable materials, or aerodynamics (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020). One key component of the recent progress in deep neural network simulation is the usage of graph neural networks (Scarselli et al., 2009, GNNs). GNNs have demonstrated effectiveness in settings that involve interactions between many entities via forward dynamics (Battaglia et al., 2018). In Sanchez-Gonzalez et al. (2020), GNNs are used to obtain models that generalize fluid dynamics and 2D interactions over many timesteps and different initial conditions. In Pfaff et al. (2020), mesh-based simulations are learned using GNNs to predict the dynamics of a wide range of physical systems, including aerodynamics, or structural mechanics.

Compared to previous work (i.e., Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020), we focus on learning 3D simulations of granular particle flow with nontrivial geometric boundary conditions. These simulations are highly relevant for the design of industrial processes and allow to understand and improve particle flow dynamics of various given materials. Since no underlying governing equation for general granular flows exists, we use simulation data originating from the Discrete Element Method (Cundall & Strack, 1979, DEM) as the ground truth (see Appendix A for further details on DEM). We generate granular flow simulation data with the open-source DEM software LIGGGHTS (Kloss et al., 2012). LIGGGHTS allows the simulation of particulate flows with a wide range of materials and complex mesh-based wall geometries, and therefore enables the simulation of relevant industrial processes. Such complex mesh-based wall geometries are in contrast to Sanchez-Gonzalez et al. (2020), where for the 3D simulations static cuboids have been assumed and a material point method (Sulsky et al., 1995, MPM) based simulator (Hu et al., 2018) has been applied.

Similar to Sanchez-Gonzalez et al. (2020), we learn a time-transition model, consisting of an encoder, processor, and, a decoder, to predict particle accelerations and build upon their relative positional encoding variant. The time transition model from time t_k to time t_{k+1} is given by

$$\dot{\boldsymbol{p}}^{t_{k+1}} = \dot{\boldsymbol{p}}^{t_k} + \Delta t \; \ddot{\boldsymbol{p}}^{t_{k+1}} \; , \tag{1}$$

$$\boldsymbol{p}^{t_{k+1}} = \boldsymbol{p}^{t_k} + \Delta t \; \dot{\boldsymbol{p}}^{t_{k+1}} \; , \tag{2}$$

where $\boldsymbol{p}$ is the particle location, and $\dot{\boldsymbol{p}}$ the particle velocity. The new particle acceleration $\ddot{\boldsymbol{p}}$ at time t_{k+1} needs to be predicted. The contributions of this work towards an accurate modeling of complex 3D granular flows are three-fold:

- Our model deals with triangularization of 3D objects, which is a very common representation in engineering applications. We achieve this by inserting virtual nodes if particles are close to boundaries and computing the proximity to the closest triangle.
- We include particle - boundary interactions by forcing the network to be invariant with respect to the orientation of the normal vectors.
- We compare and analyze relevant physical quantities between simulated processes and the output of our network.

2 Accurate Modeling of Granular Flow Dynamics

Learning simulations that are governed by complex geometries. Triangularization of geometric boundaries is very common in engineering applications. One idea to tackle this challenge for 2D scenes is the insertion of stationary, virtual particles into the scene to describe boundaries (Sanchez-Gonzalez et al., 2020). However, for 3D scenes and thus for many practically relevant engineering applications, a computationally more efficient approach is required for the following reasons: Firstly, triangle surface areas in 3D would require significantly more stationary particles for the representation than curves in 2D scenes. Secondly, for certain time frames, only some parts of the mesh are relevant, e.g. as long as particles are in the state of a free fall in a container and the bottom of a container is still far away, the mesh part describing the bottom of the container is irrelevant for the next time step.

Modeling distances to triangular boundaries in 3D scenes. The challenge is to correctly incorporate the modeling of the proximity of a particle to a triangular 3D boundary and at the same time avoid large computational cost. Firstly, to avoid large computational cost, we dynamically insert boundary particles if a real particle is close to the corresponding boundary. The inserted virtual nodes in the graph describe whole surface areas in contrast to single 3D points. Therefore, we indicate the particle type, virtual or real, by introducing additional node features, such that the neural network is able to distinguish them. In other terms, this allows the model to 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 of the triangular surface (null vectors for real particles). The second challenge is to correctly model the proximity of a particle to the boundary. is computed as the minimum distance between the particle center and the closest point of the mesh triangles (adopted from Eberly (1999)). We assume a triangle to be represented by a function $\boldsymbol{T}$, which is parameterized by two scalar values u and $v \in \mathbb{R}$:

$$\boldsymbol{T}\,(u, v) = \boldsymbol{B} + u\;\boldsymbol{E}_0 + v\;\boldsymbol{E}_1, \tag{3}$$

where $u \geq 0$, $v \geq 0$, $u + v \leqslant 1$, $\boldsymbol{B}$ represents one of the corner points of the triangle, and, $\boldsymbol{E}_0$ and $\boldsymbol{E}_1$ are vectors from $\boldsymbol{B}$ towards the other two corner points (see Figure 1). We solve the optimization problem

$$\min_{u,v} \; \boldsymbol{Q}\,(u, v) = |\boldsymbol{T}\,(u, v) - \boldsymbol{P}|^2 \tag{4}$$

$$s.t. \;\; u \geq 0, \;\; v \geq 0, \;\; u + v \leqslant 1$$

to obtain the minimizing parameters u, v for retrieving the closest point $\boldsymbol{T}\,(u, v)$ of the triangle from the point $\boldsymbol{P}$ in an Euclidean sense. As indicated in Figure 1, we have to distinguish seven cases: the simplest one is *Case 0*, where $\boldsymbol{\nabla Q}\,(u, v) = \boldsymbol{0}$ is fulfilled in the inner of the triangle (i.e., $u > 0$, $v > 0$, $u + v < 1$). If $\boldsymbol{\nabla Q}\,(u, v) = \boldsymbol{0}$ is fulfilled outside from that triangle, cases where the

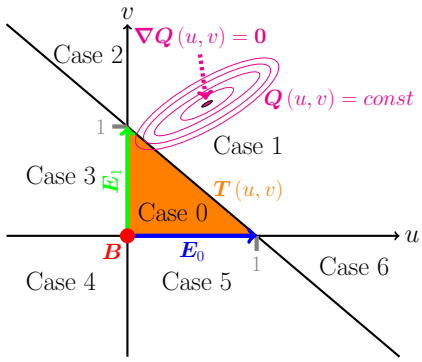

Figure 1: Visualization of point - triangle distance calculations in 3D. The triangle is represented by a parameterized function $\boldsymbol{T}(u,v) = \boldsymbol{B} + u\ \boldsymbol{E}_0 + v\ \boldsymbol{E}_1$ with $u \geq 0$, $v \geq 0$, $u + v \leqslant 1$ (indicated by the orange area). Level sets of $\boldsymbol{Q}(u,v)$ are indicated by ellipses and describe the squared Euclidean distance of a triangle point $\boldsymbol{T}(u,v)$ to the point $\boldsymbol{P}$, for which we compute the minimum distance.

minimizing point from the triangle is at the triangle border, i.e., the lines $u = 0, v = 0, u + v = 1$ have to be considered. In order to find the minimizing parameters in this case we have to look for stationary points of the one-parameter functions $\boldsymbol{Q}(u, 1-u)$, $\boldsymbol{Q}(u, 0)$ and $\boldsymbol{Q}(0, u)$.

Learning particle - boundary interactions. In granular flow simulations, usually particle - particle interaction data outweigh particle - boundary interaction data, which makes learning particle - boundary interactions difficult. We therefore put more emphasis on the type and normal vector features by introducing hyperparameters for the features, that allow discrimination between real particles and virtual boundary particles. The GNN predictions should be independent of the orientation of the normal vectors representing the planes of the triangle walls. Since there is a positive and a negative choice, we use both a positively and a negatively oriented version of the respective normal vectors as input features. However, as the prediction of a network should be invariant with respect to the orientation of the normal vectors, we define a partial ordering to be able to sort the normal vectors with respect to their orientations. For a given normal vector $\boldsymbol{n} = (n_1, n_2, n_3) \in \mathbb{R}^3$, we use the following partial order function

$$f_o(\boldsymbol{n}) = \sum_{i=1}^{3} 3^{i-1} \left(\text{sgn}\left(n_i\right) + 1\right) \tag{5}$$

to retrieve the scalar values $o_1 = f_o(\boldsymbol{n})$ and $o_2 = f_o(-\boldsymbol{n})$ and sort the two vectors according to their corresponding mapped values. We choose a base of 3 since zero entries of vectors are possible. Different sign combinations of the normal vectors are shown in Figure 2.

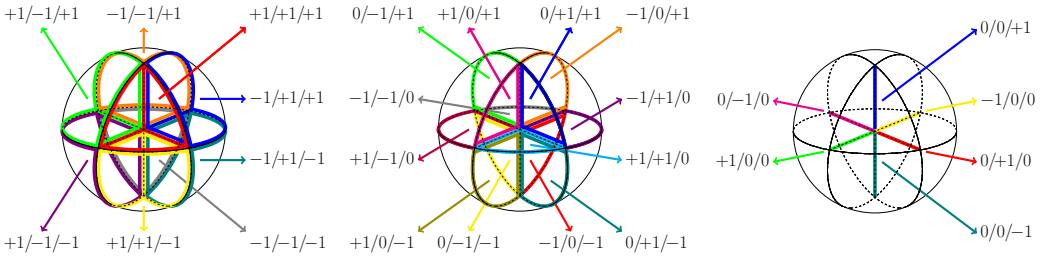

Figure 2: Partial ordering of normal vectors. The numbers indicate the signs of normal vector components, which are used in the partial order function. The figures from left to right visualize different (ordered) sign combinations. Sets of sign combinations without zero values form volumes (left), sets with one zero value form planar areas (middle), and sets with two zero values form line sections (right). The 0/0/0 combination forms a point at the origin.

To test the performance of our approach, we conduct a toy experiment as well as a simulation experiment with different representations of normal vectors. We describe both experiments in the appendix (Section B.1 and Section B.2).

3 RESULTS

We focus on two typical applications of granular flow simulations in the design of industrial machinery: the particle flow through a hopper and the particle dynamics in a rotating drum. Figure 3 visualizes the particle positions for both problem settings at different time steps in the ground truth simulation and in the prediction of our model. For both applications 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. 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. At the bottom of the hopper there is a hole with an adjustable radius, that is initially closed. Our generated training data consists of 30 simulation trajectories with different angles α and different hole sizes. Moreover, the initial filling distribution is varied. The rotation axis of the drum is the y-axis. The initial filling is obtained by rotating the filling of a resting drum around the x-axis.

In Figure 4 we compare ground truth trajectories to the learned trajectories. The machine learning model does not exactly reproduce the ground truth trajectories due to chaotic behavior in the long run, but aggregated quantities like particle-averaged positions (upper left plot) show good qualitative agreement. The upper right and the lower left plots of the figure show that also the time- and particle-averaged particle flows are well reproduced. In the lower right plot we analyse the machine learning model in terms of mixing behavior. We quantify the extend of particle mixing via the mixing entropy (Lai & Fan, 1975), which is computed by splitting particles into two classes +1, -1 at a certain time step $t_0 = 30$ and then summing over weighted local entropies $s(\boldsymbol{x}_{klm}, t)$ located at grid cells $\boldsymbol{x}_{klm}$ (see Appendix C for details). The lower right plot shows that the expected increase of the mixing entropy over time is well described by the GNN. Overall, we conclude that our GNN approach is able to model 3D granular flow simulations accurately.

4 OUTLOOK

Although, we have already used comparably large time steps in our machine learning model (i.e. multiples of the ground truth simulation training data), we have not fully investigated the potential of machine learning models for granular simulation data yet: Firstly, the simulation time of the ground truth simulation is strongly material-dependent. As a downside, the true Young‘s Modulus, which describes a stress-strain relationship in the linear elastic region of a material, is usually not used in the computation of simulations, but a much smaller value. Machine learning models therefore could allow the simulation of granular flows with more realistic material parameters. Secondly, larger time steps would allow to model physical phenomena occurring on larger time scales, which cannot be modeled accurately with current DEM approaches. Further, we plan to leverage the multiple symmetries occurring in the geometry of granular flow problems. Another research direction might be the usage of global parameters such as the angles of the side walls of the hopper as additional input features to the neural network.

ACKNOWLEDGMENTS

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

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), DeepToxGen (LIT-2017-3-YOU-003), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), PRIMAL (FFG-873979), S3AI (FFG-872172), ELISE (H2020-ICT-2019-3 ID: 951847), AIDD (MSCA-ITN-2020 ID: 956832). We thank Janssen Pharmaceutica (MaDeSMart, HBC.2018.2287), 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, Software Competence Center Hagenberg GmbH, TÜV Austria, and the NVIDIA Corporation.

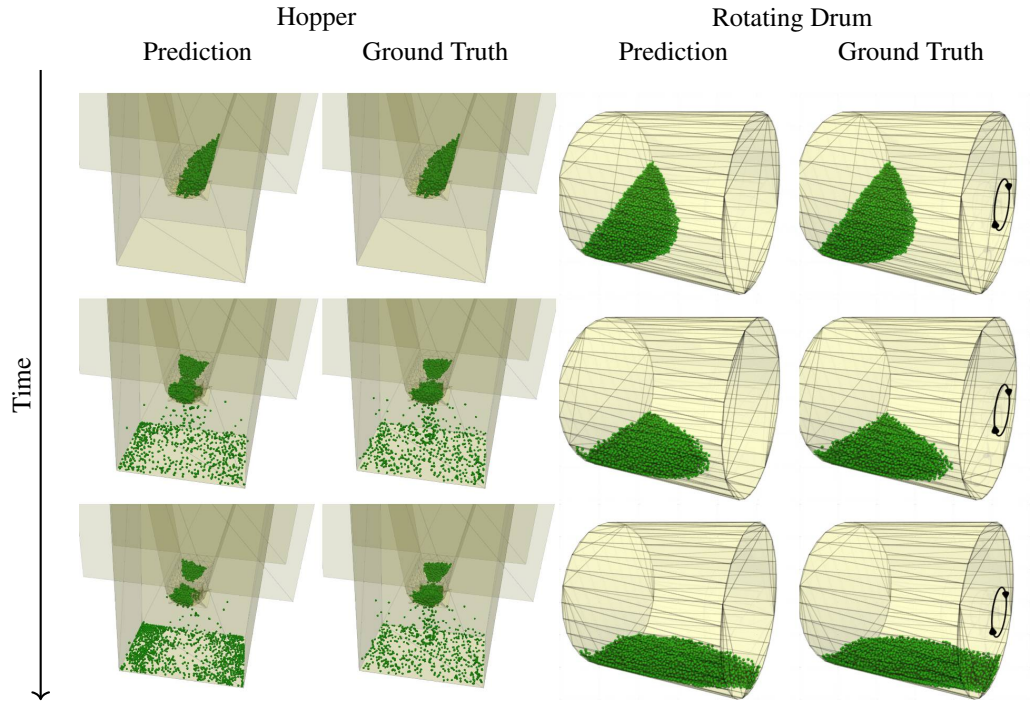

Figure 3: Particle distributions for Hopper and Drum dynamics. Data obtained by the particle simulator LIGGGHTS (Ground Truth) and our trained graph neural network (Prediction) are compared. Particles are indicated by green spheres, triangular wall areas are yellow, the edges of these triangles are indicated by grey lines. The circular arrow indicates the rotation direction of the Drum.

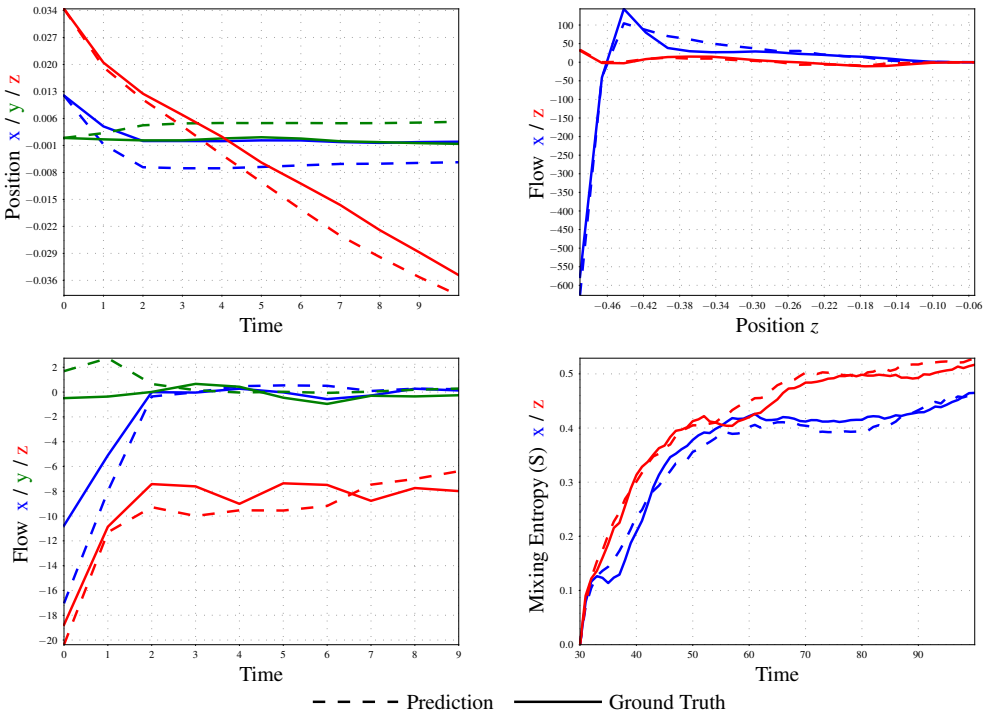

Figure 4: Position (upper left) and flow profile (lower left) plots for the Hopper, and, flow profile (upper right) and entropy plot (lower right) for the Drum. The plots visualize ground truth (solid line) vs. predictions (dashed line) in dependence of the time step or a coordinate. The flow profiles are average velocities of of particles at a given time step or z coordinate. The mixing entropies are obtained by splitting particles into two partitions according to a threshold on the respective x or z coordinate at time step 30.

REFERENCES

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

N. Bonneel, M. van de Panne, S. Paris, and W. Heidrich. Displacement interpolation using lagrangian mass transport. 30(6):1–12, December 2011. ISSN 0730-0301. doi: 10.1145/2070781.2024192.

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

D. Eberly. Distance between point and triangle in 3d. *Retrieved from http://www.magic-software.com/Documentation/pt3tri3.pdf*, 1999.

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

R. Flamary and N. Courty. Pot python optimal transport library, 2017. URL `https://pythonot.github.io/`.

Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics*, 37(4), July 2018. ISSN 0730-0301.

C. Kloss, C. Goniva, A. Hager, S. Amberger, and S. Pirker. Models, algorithms and validation for opensource dem and cfd-dem. *Progress in Computational Fluid Dynamics, An International Journal*, 12(2/3):140, 2012. ISSN 1468-4349.

C. Kloss, A. Aigner, A. Mayrhofer, and C. Goniva. fastdem: A method for faster dem simulations of granular media. In *Particles 2017*, 2017.

F.S. Lai and L.T. Fan. 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, 1975. doi: 10.1021/i260056a009.

J. Mellmann, K.L. Iroba, T. Metzger, E. Tsotsas, C. Mészáros, and I. Farkas. Moisture content and residence time distributions in mixed-flow grain dryers. *Biosystems Engineering*, 109(4):297–307, 2011.

M. Obermayr, C. Vrettos, P. Eberhard, and T. Däuwel. A discrete element model and its experimental validation for the prediction of draft forces in cohesive soil. *Journal of Terramechanics*, 53:93–104, 2014.

T. Pfaff, M. Fortunato, A. Alvaro Sanchez-Gonzalez, and P.W. Battaglia. Learning mesh-based simulation with graph networks, 2020.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 8459–8468, 2020.

F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

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

A DISCRETE ELEMENT METHOD (DEM)

The key idea of of the Discrete element Method (DEM) is that the granular medium is represented by discrete objects, commonly referred to as particles (e.g. spheres or polyhedra) and that they interact by exchanging momentum at a particle - particle contact level using a so-called contact model. The most basic contact model is a spring-dashpot model, in which the interaction force F_{ij} between two particles i and j is given as

$$F_{ij} = k\delta_{ij} - \gamma v_{ij} \ , \tag{A.1}$$

where k is the spring stiffness, δ_{ij} is the overlap of the two particles, γ is the damping constant and v_{ij} is the relative velocity between the two particles. Such contact models can become prohibitively complex in order to model phenomena such as cohesion (Obermayr et al., 2014), surface roughness and others. One significant downside of the DEM approach is that the time integration of the Lagrangian particles requires small time steps to properly resolve the particle contacts. In industrial applications there often is the need to study physical phenomena which occur on different time scales, e.g. particle collisions ($\mathcal{O}(10^{-5}s)$) vs. moisture content in particles ($\mathcal{O}(1s)$, Mellmann et al. (2011)), which can lead to weeks of simulation time. While advances have been made to overcome such issues (e.g. Kloss et al. (2017)), they remain limited in their application, due to the fact that they rely on prior simulation of the exact setup and cannot be used for interpolation of quantities directly related to the flow behavior.

B NORMAL VECTOR REPRESENTATIONS

B.1 REFLECTION TOY EXAMPLE

We conduct a toy experiment to showcase that a canonical representation of normal vectors is helpful for learning 3D simulations. In detail, we consider reflection at a plane as given by $Ref_{\boldsymbol{n}}(\boldsymbol{v}) = \boldsymbol{v} - 2\frac{\boldsymbol{v}\cdot\boldsymbol{n}}{\boldsymbol{n}\cdot\boldsymbol{n}}\boldsymbol{n}$ and try to learn the reflection formula by a simple ReLU network, which takes the 3 components of $\boldsymbol{n}$ and $\boldsymbol{v}$ as input features and predicts the 3 components of $Ref_{\boldsymbol{n}}(\boldsymbol{v})$. The training data thereby consists of reflections at four fixed walls: the top, the bottom, the left, and, the right side of a simple cube.

We use normal vectors of these wall, that point towards the outer of the cube. When evaluating the performance of the trained models, we observe decent predictions, if the orientation of the normal vectors describing the inclination of the walls was equal to the training data (see R1-R4 in Figure B.1). However, for inverted normal vectors in the test set, only networks which are trained by taking a canonical orientation into account work well (see R3, R4 in Figure B.2).

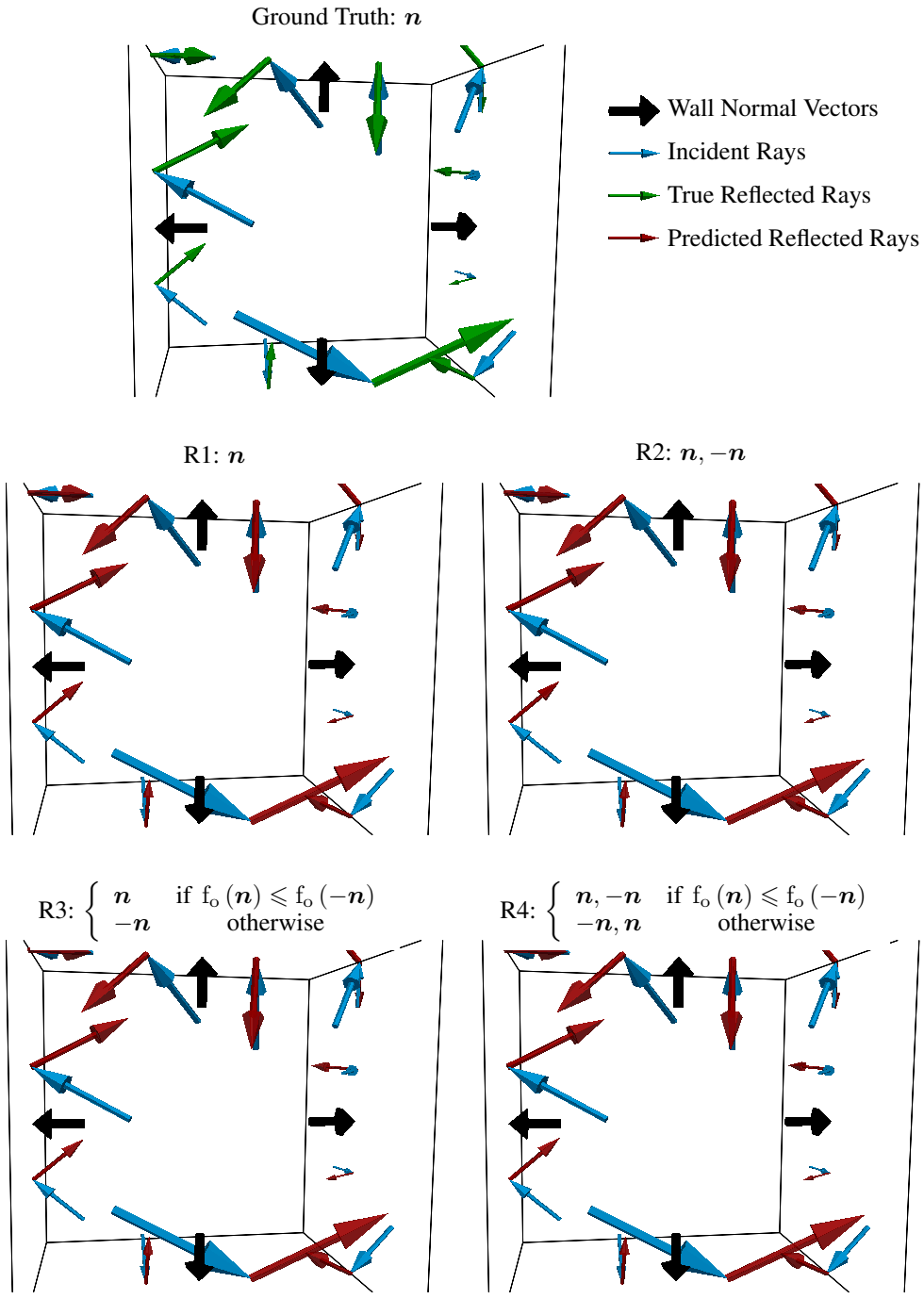

Figure B.1: Reflection of rays at four different walls (left, right, bottom, top). Wall normal vectors are visualized by black arrows. The incident rays are visualized by blue arrows, reflected rays are indicated by green arrows in the ground truth plot. Red arrows in plots R1-R4 visualize neural network predictions. Neural network predictions are based on wall representations **that are oriented the same way as in the training phase**. The caption above each plot indicates the wall input features used by each of the trained networks.

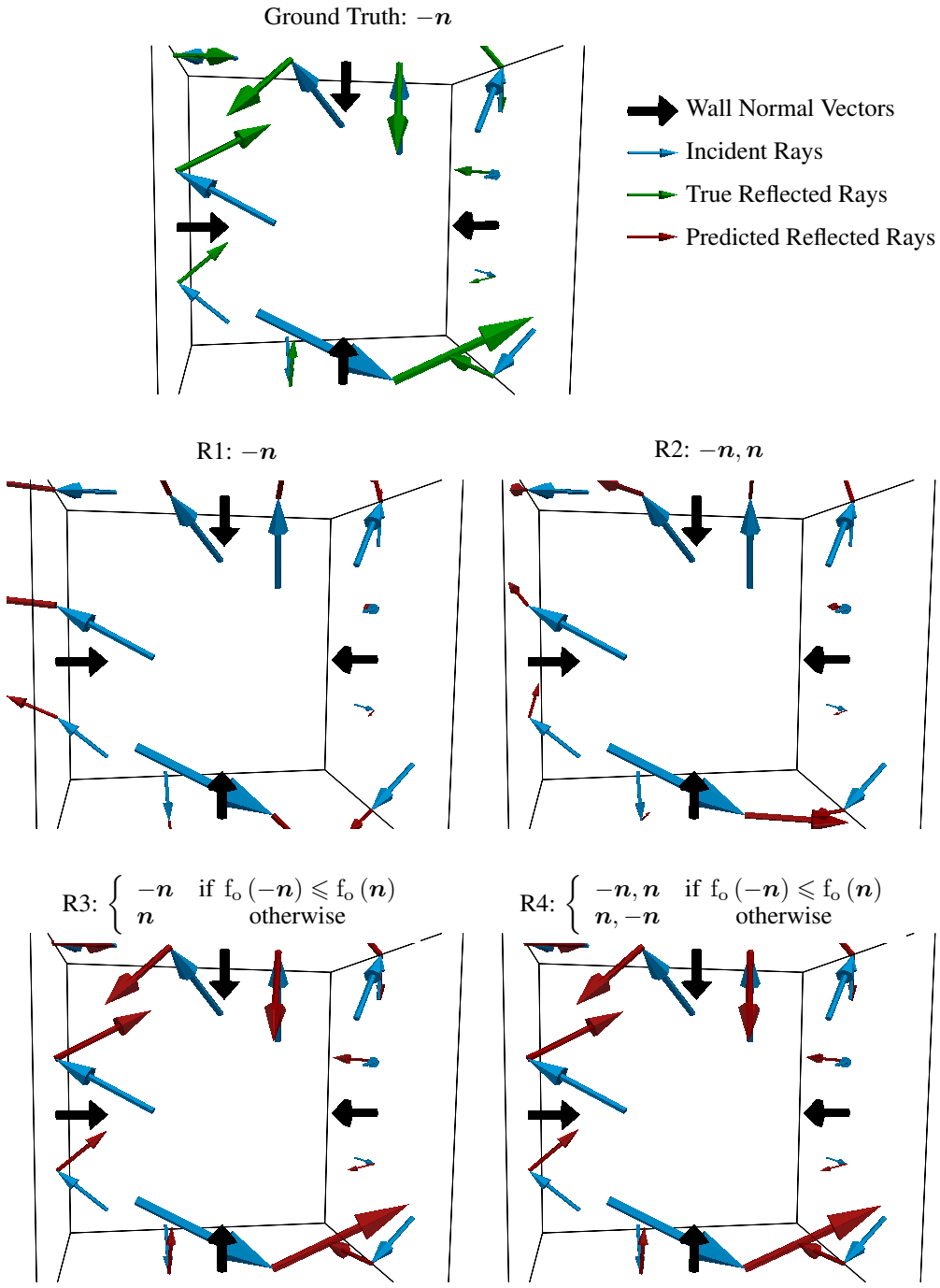

Figure B.2: Reflection of rays at four different walls (left, right, bottom, top). Wall normal vectors are visualized by black arrows. The incident rays are visualized by blue arrows, reflected rays are indicated by green arrows in the ground truth plot. Red arrows in plots R1-R4 visualize neural network predictions. Neural network predictions are based on wall representations **that are inversely oriented compared to the training phase**. The caption above each plot indicates the wall input features used by each of the trained networks.

B.2 Simulation Experiment

We compare three versions for including normal vectors as boundary node features for the hopper particle flow:

- not including normal vector information, filling six node features up with zero entries instead (V1)
- including single normal vector orientation, which is given by the triangle corner point order of the mesh (V2)
- including both normal vector orientations (six features) (V3)

From an information perspective, it should be noted that (i) distance information (scalar distance and relative distance vectors) to the walls is present in the edge features of the graph and (ii) in most cases the used normal vectors were oriented towards the outside of relevant border walls.

The different particle distribution trajectories obtained by the three versions are compared by computing the Earth Movers distances (Bonneel et al., 2011; Flamary & Courty, 2017, EMD) between the trajectories from the machine learning model and the simulator. We use Euclidean distances for the cost matrix, which we compute at time steps $2^0, 2^1, \cdots, 2^{16}$ for 5 training trajectories and 5 test trajectories. Table B.1 shows the means (μ) and standard deviations (σ) of EMD values at different time steps and from 5 different training and test trajectories. A paired Wilcoxon test on the concatenated trajectories, shows that V3 significantly outperforms V1 (p-value 2.42e-04) and V2 (p-value 1.50e-03) on the test data.

Interestingly, there is less significance on the training data, which might indicate that the usage of orientation-independent features to represent walls, helps to improve generalization performance, while it might not be that helpful for optimization purposes alone.

Table B.1: Usage of normal vector as node feature for the particle flow through a hopper. The table summarizes means (μ) and standard deviations (σ) of the EMD for the different versions and shows the results of a paired Wilcoxon test.

Version		**Train**			**Test**		
		μ	σ	p-value Row < V3	μ	σ	p-value Row < V3
V1	No normal vector	5.06e-05	1.17e-04	2.36e-02	6.80e-05	1.59e-04	2.42e-04
V2	Single normal vector	1.15e-04	3.84e-04	3.40e-03	1.21e-04	4.33e-04	1.50e-03
V3	Both orientations	5.99e-05	1.77e-04		6.36e-05	2.06e-04	

C Mixing Entropy

At time t, the local entropies $s(\boldsymbol{x}_{klm}, t)$ at grid cells, represented by a point $\boldsymbol{x}_{klm}$, are computed from particle counts $n_{+1}(\boldsymbol{x}_{klm}, t), n_{-1}(\boldsymbol{x}_{klm}, t)$ of the respective classes at the grid cells, where $n(\boldsymbol{x}_{klm}, t) = n_{+1}(\boldsymbol{x}_{klm}, t) + n_{-1}(\boldsymbol{x}_{klm}, t)$. The mixing entropy $\mathrm{S}(t)$ is then computed by the formulas given by eqs C.1.

$$
\begin{aligned}
f_{\pm 1}(\boldsymbol{x}_{klm}, t) &= \frac{n_{\pm 1}(\boldsymbol{x}_{klm}, t)}{n_{+1}(\boldsymbol{x}_{klm}, t) + n_{-1}(\boldsymbol{x}_{klm}, t)} \\
s(\boldsymbol{x}_{klm}, t) &= -f_{+1}(\boldsymbol{x}_{klm}, t) \log f_{+1}(\boldsymbol{x}_{klm}, t) - f_{-1}(\boldsymbol{x}_{klm}, t) \log f_{-1}(\boldsymbol{x}_{klm}, t) \\
\mathrm{S}(t) &= \frac{1}{\sum\limits_{k,l,m} n(\boldsymbol{x}_{klm}, t)} \sum_{k,l,m} n(\boldsymbol{x}_{klm}, t) s(\boldsymbol{x}_{klm}, t)
\end{aligned}
\tag{C.1}
$$