16th World Congress in Computational Mechanics (WCCM)
21-26 July 2024, Vancouver, Canada
A. Korobenko, M. Laforest, S. Prudhomme, R. Vaziri (Eds)

GRAPH NEURAL NETWORKS FOR ACCELERATING THE DISCRETE ELEMENT SIMULATION OF GRANULAR FLOW

PENG ZHI* AND YU-CHING WU

Department of Structural Engineering, College of Civil Engineering, Tongji University
1239 Siping Road, Shanghai 200092, China
*pengzhi@tongji.edu.cn

Key words: Granular Flow, DEM, GNN.

Abstract. *Granular flow is a phenomenon widely presented in both the natural and engineering fields. Here granular materials could be either solid particles, e.g. rocks, soil, and grains, or liquid particles, e.g. mud and fresh concrete mortar. Soil landslides, particle transport, and grain accumulation have been edge-cutting hot research topics. Discrete Element Method (DEM) has been regarded as one of the most important methods to simulate granular flows and to investigate discontinuous and large deformation problems. The basic principle of DEM was to view the simulated object as consisting of discrete particles, to define specific constitutive relationships for the particles, and to study the macroscopic properties of the simulated object from a microscopic perspective based on the interactions between particles. However, DEM simulations usually consume very high computational cost for particle contact searching and detection. To accelerate the computational process of discrete element simulation, the Graph Neural Network (GNN) based deep learning model was proposed in this paper. In GNNs, graph nodes and graph edges represent the particles and their interactions. The training and testing datasets were generated using an open-source software named YADE, while the neural network model was constructed using PyTorch and Deep Graph Library (DGL). Replacing the direct calculation of particle collisions in DEM with the trained neural network model, the state of the particles at the next moment could be predicted based on the current state of the particles. It significantly increased computational speed. The proposed technique was applied in various examples, such as drum rotation and hopper stacking, and its accuracy had been verified. This study established a solid foundation and provided robust support for further research and applications of granular flow simulation based on GNN.*

1 INTRODUCTION

Granular materials are widely used in various industries such as agriculture, soil engineering, petrochemicals, medicine and geotechnical engineering. In recent years, researchers have used numerical simulation methods to study complex granular flow phenomena. Among these methods, the Discrete Element Method (DEM) is a method for direct simulation of particles with high calculation accuracy[1]. However, since DEM requires calculating the forces on particles in very small time steps, it results in large computational loads and high costs.

With the improvement of computer technology, machine learning and artificial intelligence technologies have been widely used in various fields such as computational mechanics, image processing, data prediction, computer vision, and speech recognition, and have received great attention from scientific research institutions and the industrial community. Using neural networks as agents or accelerating numerical calculations has become a popular research direction.

On the one hand, neural networks can establish the relationship between microscopic parameters and macroscopic behaviors in DEM, so that microscopic parameters can be calibrated or DEM calculation results can be predicted. This aspect has been extensively studied by many researchers[2–5]. On the other hand, based on the current state information of the particles, neural networks can be used to predict the state of the particles in the next time step, thereby accelerating the calculation process of granular flows. This study will explore the latter aspect.

This study aims to further explore the possibility of using Graph Neural Networks (GNN) to accelerate DEM simulation in various granular flow scenarios. A particle-particle and particle-boundary GNN prediction model was established, and the boundaries in the DEM were sampled as discrete points and input into the GNN as nodes to consider the influence of irregular boundaries. By inputting the current state of the particle, the model can predict the state of the particle in the next time step. The prediction accuracy and stability of the GNN model were verified through the simulation cases of granular flow with horizontal drum and hopper stacking.

2 RELATED WORK

Neural network methods for predicting particle states generally include Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and GNNs.

MLPs can be used to establish prediction models for particle flows. Zhang et al.[6] created the FluidsNet model to predict the velocity field of fluid particles based on simulation data. Lai et al.[7] used MLPs to detect particle contact and obtain related contact information to simulate the physical behavior of irregularly shaped particles. Yao et al.[8] constructed a data-driven model to solve the pressure Poisson equation in fluid particles, which greatly accelerated the calculation speed.

The three-dimensional continuous convolution method was proposed by Ummenhofer et al.[9], which uses spatial convolution as the main differentiable operation to connect particles with their neighbors and has achieved good results in Lagrangian fluid simulation. Lu et al.[10] and Xu and Shen[11] applied this method on DEM and trained CNNs to replace the calculation process of particle-particle and particle-boundary contacts.

GNNs use less structured data (i.e., graph structure) as input. The lower structured level means that the input can have any shape and size and can show complex topological relationships. This method has a natural advantage in establishing connections between particles. Particles are represented as nodes and the interactions between particles are represented as edges. Sanchez-Gonzalez et al.[12] proposed a graph network-based simulator to represent the physical state of particles such as fluids and rigid solids. Klimesch et al.[13] established a GNN for learning fluid dynamics and improved the generalization ability and stability of the model by adding random noise. Li and Farimani[14] also used GNN to predict fluid particle flows, and they used two GNN structures, node-focused networks and edge-focused networks. Ma et al.[15] established a computational framework combining fluid dynamics with GNN, which can infer and predict the dynamics of particles in suspension.

Kumar and Vantassel[16] used GNN to predict the flow behavior of two-dimensional particles. Mayr et al.[17] established a boundary GNN model based on effective theory, which can effectively take into account the irregular triangular geometric boundaries in DEM.

3 METHODS

3.1 The basis of DEM

The basic equations of DEM include physical equations and motion equations. The physical equations are used to solve the forces on the particles, and the motion equations are used to calculate the acceleration, velocity and displacement of the particles.

In the linear contact stiffness model, the normal contact force vector $\boldsymbol{F}_n$ is determined by the normal contact stiffness K_n and the overlap between particles ξ_n. The tangential contact force vector is calculated in an incremental manner, that is, the increment of the tangential force $\Delta\boldsymbol{F}_T$ is related to the increment of the tangential relative displacement $\Delta\boldsymbol{\xi}_t$

$$\boldsymbol{F}_N = K_n \xi_n \boldsymbol{n} \tag{1}$$

$$\Delta\boldsymbol{F}_T = K_t \Delta\boldsymbol{\xi}_t \tag{2}$$

where $\boldsymbol{n}$ is the unit vector in the normal direction, K_t is tangential contact stiffness.

Newton’s second law is used to calculate the displacement, velocity and acceleration of particles. The movement of particles consists of two parts: translation and rotation. The translation equation and rotation equation are

$$F_i = m_i\left(\ddot{u}_i - g\right) \tag{3}$$

$$T_i = I\dot{\omega}_i \tag{4}$$

where F_i is the resultant force acting on the particle, m_i is the mass of the particle, $\ddot{u}_i$ is the acceleration of the particle, g is the gravitational acceleration, T_i is the total torque acting on the particle, I is the moment of inertia, $\dot{\omega}_i$ is the angular acceleration of the particle.

3.2 Encoder-processor-decoder GNN

GNN is a class of neural networks designed to work with graph-structured data, which learns the feature information of nodes and edges in the graph. GNNs can process data represented as graphs, which consist of nodes and edges.

Suppose a graph $G = (N, E)$, with nodes $n_i \in N$ and edges $e_{ij} \in E$, each node has node feature h_i, and each edge has edge feature m_{ij}. The traditional encoder-processor-decoder architecture is used. First, the node features and edge features are encoded

$$h_i' = \varepsilon^N\left(h_i\right) \tag{5}$$

$$m_{ij}' = \varepsilon^E\left(m_{ij}\right) \tag{6}$$

where ε^N and ε^E are the encoders for nodes and edges respectively, and the encoder is implemented by MLP.

In the processor, the message is transmitted for each edge to obtain the processed edge feature $\tilde{m}_{ij}$, and then the message is aggregated for each node to obtain the processed node feature $\tilde{h}_i$

$$\tilde{m}_{ij} = \phi\left(h_i', h_j', m_{ij}'\right) \tag{7}$$

$$\tilde{h}_i = \psi\left(h_i', \Lambda_{e_{ij} \in E}\left(\tilde{m}_{ij}\right)\right) \tag{8}$$

where ϕ and ψ are MLP, $\Lambda_{e_{ij} \in E}$ represents the aggregation of the edge e_{ij} of node n_i, Λ represents the aggregation method, generally there are sum, mean, and maximum methods, this study adopts the sum method.

Finally, the node features are decoded to obtain the output result

$$o_i = \gamma^N\left(\tilde{h}_i\right) \tag{9}$$

where γ^N is the decoder of the node and is also implemented by MLP.

3.3 GNN framework

The boundaries in DEM are generally composed of walls. The movement of particles is affected by both particle-particle contact and particle-boundary contact. In order to consider the influence of boundaries in GNN, the walls are discretized into uniformly distributed boundary points[9,10], which can be input into the graph as nodes. When the distance between particles is less than the cutoff distance for particles r_{cutoff}, edges are established between nodes. Similarly, the cutoff distance for particles to contact the boundary r^w_{cutoff} determines the edge between particles and boundary points.

The GNN computation framework is shown in Figure 1. They are two GNN models, GNN for particle-particle contact and GNN for particle-boundary contact. The positions of particles and boundary points are input, and then node features and edge features are constructed. The particle node feature contains the position $\boldsymbol{x}^n$ and velocity vector $\boldsymbol{v}^n$; the boundary point feature contains the position and normal vector $\boldsymbol{nv}^n$; the edge feature refers to the idea of Mayr et al.[17] and contains many formulas with physical information calculated based on the node position.

The two GNNs output the particle-particle velocity $\hat{\boldsymbol{v}}^{n+1}_{\text{pp}}$ and the particle-boundary velocity $\hat{\boldsymbol{v}}^{n+1}_{\text{pw}}$ at the next time step ($t+\Delta t$), respectively, and calculate the particle velocity and the particle position according to the difference method.

$$\hat{\boldsymbol{v}}^{n+1} = \hat{\boldsymbol{v}}^{n+1}_{\text{pp}} + \hat{\boldsymbol{v}}^{n+1}_{\text{pw}} \tag{10}$$

$$\hat{\boldsymbol{x}}^{n+1} = \boldsymbol{x}^n + \hat{\boldsymbol{v}}^{n+1} \cdot \Delta t \tag{11}$$

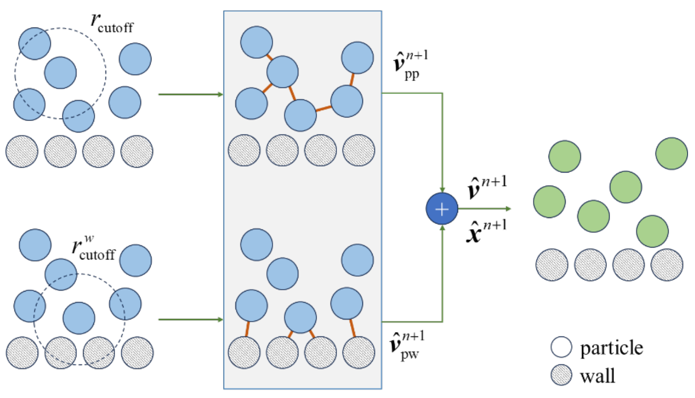
r_{cutoff}
r^{w}_{cutoff}
$\hat{\boldsymbol{v}}^{n+1}_{\text{pp}}$
$\hat{\boldsymbol{v}}^{n+1}_{\text{pw}}$
$\hat{\boldsymbol{v}}^{n+1}$
$\hat{\boldsymbol{x}}^{n+1}$
particle
wall

Figure 1: GNN framework

4 EXPERIMENTS

This study used the open-source software for DEM numerical simulations, Yade[18], which runs on Linux system, is written in C++. It has a variety of common particle models, and can customize particle material and contact constitutive relations. The GNN waws built by Deep Graph Library (DGL), which is an open-source library for deep learning GNN in Python. It provides an efficient architecture, easy-to-use API, and support for various GNN models and algorithms. The computing framework was implemented and trained by PyTorch, and the graphics card device used was NVIDIA RTX 4090 24GB.

4.1 Simulation details

(1) Horizontal drum simulation

As shown in Figure 2, a horizontally placed drum with a diameter of 300 mm and a length of 600 mm was used to generate dataset. Densely packed particles were generated in random regions of the drum, and the initial velocity of the particles was set to a random velocity, with a total of 20 different cases. The number of particles was 2530. The DEM simulation parameters are shown in Table 1. The global damping coefficient was 0, the time step was 1×10^{-5} s, and the drum rotation speed was 3 rad/s. For each case, the run time was 3 s.

Table 1: DEM simulation parameters

Parameter	Value
gravity acceleration (m/s^2)	9.81
particle diameter (m)	0.02
particle density (kg/m^3)	1000
particle elastic modulus (Pa)	1×10^6
particle friction angle (rad)	0.5
particle Poisson's ratio	0.25
wall friction angle (rad)	0.5

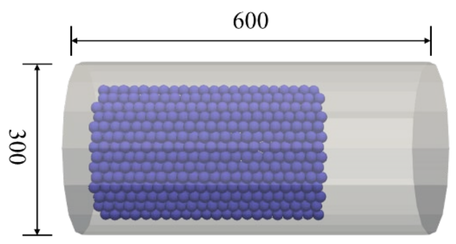

Figure 2: Horizontal rotating roller simulation (unit: mm)

(2) Hopper simulation

As shown in Figure 3, the diameter of the upper and lower parts of the hopper is 500 mm, the diameter of the middle hopper mouth is 200 mm, and the total height is 900 mm. Particles were generated at random positions in the cylindrical area of the upper part of the hopper, and the initial velocity of the particles was set to a random velocity, with a total of 20 different cases. The number of particles was 2608. The DEM simulation parameters were shown in Table 1. The global damping coefficient was 0.2, and the running time was 2 s.

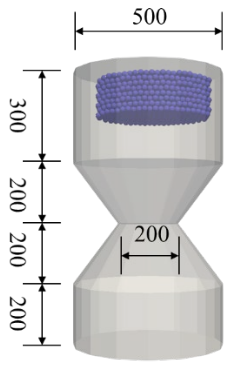

Figure 3: Hopper simulation settings (unit: mm)

4.2 Training

The DEM simulation sampled data at a frequency of 100 Hz for the two cases. For both the horizontal drum simulation and the hopper simulation, 16 cases were used as training set, and 4 cases were used as test set.

Based on the Poisson distribution sampling method[19], the geometric boundary of the horizontal drum was discretized into 3000 boundary points, and the geometric boundary of the hopper was sampled to 5000 boundary points, see Figure 4. The cutoff distance of particle-particle contact was $r_{\text{cutoff}} = 2.5r$ and the critical distance of particle-boundary contact was $r^{w}_{\text{cutoff}} = 2.5r$, where r is the radius of the particle. The input features were the position and velocity of the particles, the position and the normal vector of the boundary points. The encoder and decoder in the GNN both used three-layer MLP, the hidden layer dimension was 128, and the ReLU activation function was used. The number of layers of the GNN was 3.

The training of the GNN network adopted the one-step prediction mode, inputting the state information of the particles at a certain time to predict the position of the granular flow at the next time. The loss function was the Mean Square Error (*MSE*) between the predicted value $\hat{\boldsymbol{x}}^{n+1}$ and the true value $\boldsymbol{x}^{n+1}$ of the position of all particles. The Adam optimizer was used to optimize the model parameters. We performed a maximum of 20000 gradient update steps, with the learning rate decay from 1×10^{-4} to 3.125×10^{-6}.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{x}^{n+1} - \hat{\boldsymbol{x}}^{n+1}\right)^2 \tag{12}$$

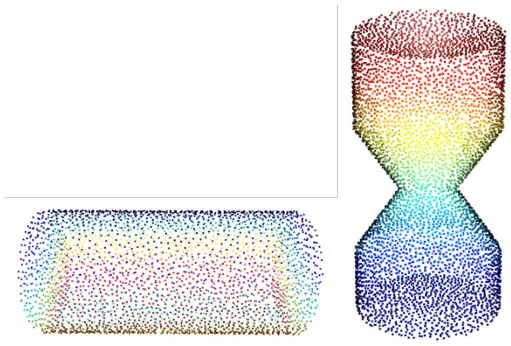
Figure 4: Boundary points for the horizontal drum and hopper

5 RESULTS AND DISCUSSIONS

5.1 Results

The evaluation of the results adopts the rollout prediction mode. The state information of the particle flow at a certain time is input, and the neural network outputs the state of the particle flow at the next time step, and then the output result is re-input, and the state of the particles at the next time step is continued to be output, and the cycle repeats.

The evaluation error of the GNN network adopted the average value of the Euclidean distance between the true value and the predicted value of all particle positions, and the calculation formula is

$$E = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{x}^{n+1} - \hat{\boldsymbol{x}}^{n+1}\right\|_2 \tag{13}$$

The GNN model inputted the state of the particles at time t = 0.1 s. Figure 5 shows the average error of the horizontal drum test set over time. At t = 0.11 s, the average error of the test set for one-step prediction of all particles was 0.00281 m; at t = 3 s, the average error of all particles in the test set was 0.0790 m. Figure 6 shows the comparison of the average angle of repose (AoR) of the test set. It can be seen that under DEM calculation, the AoR of the particle flow was basically stable at t = 3 s, with an average of about 34.46°. The average AoR predicted by GNN was about 35.03°. Figure 7 show the comparison between the predicted particle velocity and the DEM calculated value at t = 3 s. The velocity predicted by GNN was lower than the value calculated by DEM. This is because the value calculated by DEM was the instantaneous velocity (calculated with a time step of 1×10^{-5} s), while the velocity predicted by GNN is calculated by the difference method (calculated with a time step of 1×10^{-2} s). Therefore, there is a certain difference between the two. Lu et al.[10] believed that the prediction model underestimated the velocity of the particles, which is due to the fact that the prediction model ignored the fluctuations of the particles and smoothed the collision of the particles.

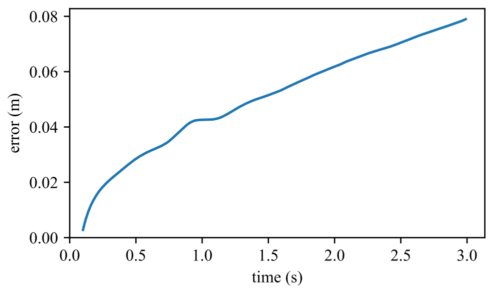

Figure 5: Evaluation error curves for the test set under horizontal drum simulation

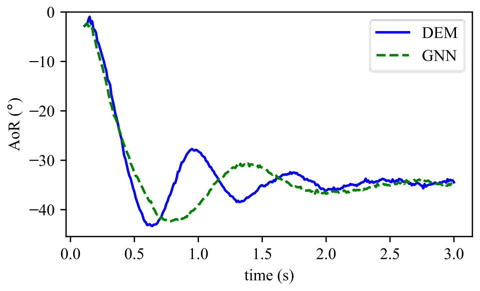

Figure 6: Comparison of AoR for the test set between GNN and DEM

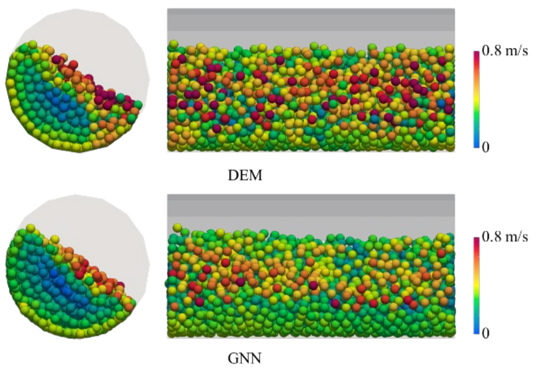

Figure 7: Comparison of particle velocity between GNN and DEM under horizontal drum simulation ($t = 3$ s)

In the hopper stacking prediction, Figure 8 shows the average error of the hopper test set. At $t = 0.11$ s, the average error of all particles in the test set single-step prediction was 0.000874 m; at $t = 2$ s, the average error of all particles in the test set was 0.0939 m. Figure 9 shows the change of the average weight center of all particles over time. It can be seen that the two curves are very close, indicating that GNN can accurately predict the change of the center of gravity of the particles and evaluate the stacking state of the granular flow as a whole. Figure 10 shows the comparison of the particle velocity values predicted by GNN and the DEM calculated values at $t = 2$ s.

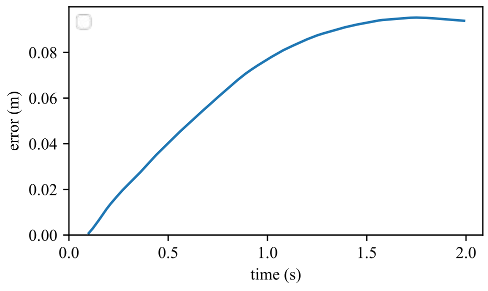

Figure 8: Evaluation error curves for the test set under hopper simulation

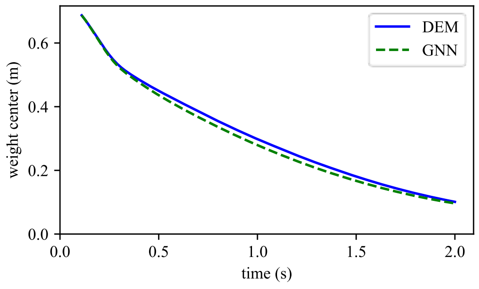

Figure 9: Comparison of weight center for the test set between GNN and DEM

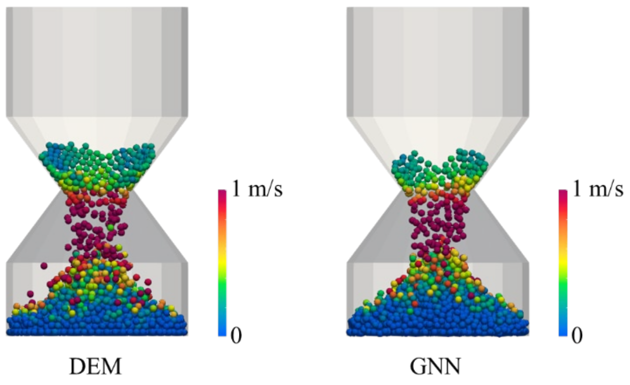

Figure 10: Comparison of particle velocity between GNN and DEM under hopper simulation ($t = 2$ s)

5.2 Limitations

In this study, we adopted the GNN framework to accelerate the computational process of DEM by predicting the state information of each time step of the granular flow, which is capable of performing both speedy and accurate simulations. The time step of GNN was 1000 times that of DEM, and the average speed of GNN prediction was about 30 times that of DEM calculation. However, there are still several limitations as follows.

First, GNN training not only takes time, but also the larger the graph (more nodes and edges), the more hardware resources are occupied by training, which limits the size of GNN graph.

Second, there will always be an irreconcilable error between the prediction results of the GNN model and the calculated value, and as the number of prediction steps increases, the difference with the calculated value will gradually be amplified.

Finally, the GNN model will smooth the collision of particles, resulting in the particle speed being lower than the DEM calculated value. In addition, there may be overlap between particles in the prediction results.

6 CONCLUSIONS

This study adopted a GNN-based accelerated DEM calculation model, including GNN for particle-particle contact and GNN for particle-boundary contact, to simulate the granular flow under horizontal drum and hopper. The main conclusions are as follows.

1) The single-step prediction errors of the GNN prediction model for the test set under the cases of horizontal drum and hopper were 0.00281 m and 0.000874 m. This indicates that the GNN can achieve high accuracy in the prediction of different granular flow scenarios. Compared with traditional DEM, GNN had high prediction accuracy and fast calculation speed, achieving about 30 times acceleration.

2) The GNN can predict macroscopic features such as the angle of repose and weight center better, but the GNN smoothed out the collision of the particles and might lead to the overlapping of the particles.

The effects of different hyperparameters on the prediction results of GNN should be further investigated subsequently. In addition, how to make the GNN learn the effects of factors such as the size and shape of the particles is also worth considering.

REFERENCES

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

[2] Qu, T., Di, S., Feng, Y.T., Wang, M. and Zhao, T. Towards data-driven constitutive modelling for granular materials via micromechanics-informed deep learning. *Int. J. Plast*. (2021) **144**:103046.

[3] Nasato, D.S., Albuquerque, R.Q. and Briesen, H. Predicting the behavior of granules of complex shapes using coarse-grained particles and artificial neural networks. *Powder Technol*. (2021) **383**:328-335.

[4] Hesse, R., Krull, F. and Antonyuk, S. Prediction of random packing density and flowability for non-spherical particles by deep convolutional neural networks and discrete element method simulations. *Powder Technol*. (2021) **393**:559-581.

[5] Tejada, I.G. and Antolin, P. Use of machine learning for unraveling hidden correlations between particle size distributions and the mechanical behavior of granular materials. *Acta Geotechnica*. (2022) **17**(4):1443-1461.

[6] Zhang, Y., Ban, X., Du, F. and Di, W. FluidsNet: End-to-end learning for Lagrangian fluid simulation. *Expert Syst. Appl*. (2020) **152**:113410.

[7] Lai, Z., Chen, Q. and Huang, L. Machine-learning-enabled discrete element method: Contact detection and resolution of irregular-shaped particles. *Int. J. Numer. Anal. Methods Geomechanics*. (2022) **46**(1):113-140.

[8] Yao, Q., Wang, Z., ZHANG, Y., Li, Z. and Jiang, J.. Towards real-time fluid dynamics simulation: a data-driven NN-MPS method and its implementation. *Math. Comput. Modell. Dynamical Syst*. (2023) **29**(1):95-115.

[9] Ummenhofer, B., Prantl, L., Thuerey, N., and Koltun, V. Lagrangian fuid simulation with continuous convolutions. In: Proceedings of The International Conference on Learning Representations 2020. *ICLR*. (2020).

[10] Lu, L., Gao, X., Dietiker, J.F., Shahnam, M. and Rogers, W.A. Machine learning accelerated discrete element modeling of granular flows. *Chem. Eng. Sci*. (2021) **245**:116832.

[11] Xu, D. and Shen, Y. An improved machine learning approach for predicting granular flows. *Chem. Eng. J*. (2022) **450**:138036.

[12] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R. Leskovec, J. and Battaglia, P.W. Learning to simulate complex physics with graph networks. In: Proceedings of the 37th International Conference on Machine Learning. *PMLR*. (2020) **119**.

[13] Klimesch, J., Holl, P. and Thuerey, N. Simulating liquids with graph networks. *arXiv*. (2022) 2203.07895

[14] Li, Z. and Farimani, A.B. Graph neural network-accelerated Lagrangian fluid simulation. *Comput. Graphics*. (2022) **103**:201-211.

[15] Ma, Z., Ye, Z. and Pan, W. Fast simulation of particulate suspensions enabled by graph neural network. *Comput. Methods Appl. Mech. Eng*. (2022) **400**:115496.

[16] Kumar, K. and Vantassel, J. GNS: A generalizable Graph Neural Network-based simulator for particulate and fluid modeling. *arXiv*. (2022) 2211.10228.

[17] Mayr, A., Lehner, S., Mayrhofer, A., Kloss, C., Hochreiter, S. and Brandstetter, J. Boundary Graph Neural Networks for 3D Simulations. In: Proceedings of the AAAI Conference on Artificial Intelligence. *Washington DC: AAAI Press*. (2023) **37**(8):9099-9107.

[18] Smilauer, V., Angelidakis, V., Catalano, E., et al. *Yade Documentation 3rd ed.* (2021) http://dx.doi.org/10.5281/zenodo.5705394.

[19] Yuksel, C. Sample Elimination for generating Poisson disk sample sets. *Comput. Graphics Forum*. (2015) **34**(2):25-32.