Proceedings of the 12th International and 52nd National Conference on Fluid Mechanics and Fluid Power (FMFP)
December 19-21, 2025, Nirma University, Gujarat, India

FMFP2025–MIS–060

Modelling Granular Avalanche Dynamics Using Graph Neural Networks

Pradyumn Singh Sikarwar[1], Vishal Sharma[2] and Gaurav Bhutani[1,*]

[1]School of Mechanical and Materials Engineering, Indian Institute of Technology Mandi, Himachal Pradesh 175075, India

Emails: s24047@students.iitmandi.ac.in; d22051@students.iitmandi.ac.in; gaurav@iitmandi.ac.in

ABSTRACT Modelling granular avalanches is essential for understanding its dynamics and mitigating the risks it poses in mountainous regions. In this work, a machine learning framework has been developed for simulating granular avalanche dynamics using a graph neural network (GNN) trained on material point method (MPM) simulations. The motion of granular particles moving down a slope, including interactions with various shapes and positions of obstacles, has been trained directly from MPM simulations. The GNN was evaluated on a range of scenarios and found to reproduce the results of the MPM solver with high accuracy, with reduced computational cost. The potential of GNN-based models for efficient and accurate avalanche simulation is demonstrated, supporting rapid hazard assessment and future research in avalanche modelling.

Keywords: Graph Neural Network, Material Point Method, Granular Flows, Data-Driven Simulation, Avalanche Dynamics

I. INTRODUCTION

Snow avalanches are fast-moving masses of snow, ice, and debris that descend steep slopes under the influence of gravity. Avalanches can be destructive, posing serious threats to life, infrastructure, and the ecosystem. Accurate simulation of avalanche dynamics is crucial for effective hazard assessment, risk mitigation, and early warning system design [6]. Physics-based models for avalanche simulations can be broadly classified into depth-averaged continuum models and high-fidelity particle-based solvers. Depth-averaged models simplify the governing equations to achieve fast computations, making them suitable for operational forecasting. On the other hand, high-resolution methods like the material point method (MPM) and Discrete Element Modelling(DEM) can provide more detailed and accurate representations of granular motion by explicitly resolving particle-scale interactions [7]. However, both categories of models suffer from a common drawback, i.e., their high computational cost.

There is growing interest in using machine learning and artificial intelligence methods to address these challenges to simulate physical systems. Physics-informed neural networks (PINNs) have been developed to incorporate physical laws directly into the training process, allowing neural networks to solve differential equations that govern complex phenomena [4]. Other neural network architectures, such as convolutional and recurrent networks, have also been used for modelling flows and material behaviours [3].

Graph neural networks (GNNs) have recently emerged as a promising approach for modelling particle-based systems [8]. In GNNs, particles are represented as nodes and the interactions between them as edges, allowing the model to learn both local and long-range physical relationships [5]. This makes GNNs particularly well-suited for simulating systems where the dynamics are driven by complex interactions among many particles, such as fluids and granular flows. Studies have shown that GNNs can predict the future states of these systems with high accuracy and significantly lower computational time compared to traditional solvers [5].

In this work, a GNN-based model is trained to simulate granular avalanche dynamics using data generated from the material point method. The model learns to predict the motion of particles over time, including their interactions with different shapes and positions of obstacles.

This paper is organised as follows. In Section II, the architecture of the GNN model is discussed, followed by a description of the MPM and the procedure by which the dataset was prepared. Section III is devoted to model evaluation, in which the flow of granular particles is visualised, the accuracy of the model is assessed, and the computational cost is compared. Finally, Section IV is dedicated to the conclusion of the paper, in which the principal findings are summarized.

II. METHODOLOGY

A. GNN Architecture

In GNN, particle-based physical systems are modelled as graphs. At each simulation timestep, the set of particles and their interactions are represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes $v_i \in \mathcal{V}$ correspond to particles and edges $e_{ij} \in \mathcal{E}$ encode pairwise interactions.

The GNN operates through a structured three-step approach:

Encoder: The encoder transforms the state of each particle into a latent embedding. Each particle i is described by its position $\mathbf{x}_i$, velocity $\mathbf{v}_i$, and a category label c_i (indicating the type of material or boundary condition). For each particle i, the initial embedding of a node h_i^0 is calculated using a multilayer perceptron (MLP) ϕ_v:

$$h_i^0 = \phi_v(\mathbf{x}_i, \mathbf{v}_i, c_i). \tag{1}$$

Edges are constructed between neighbouring particles, where

two particles are considered neighbours if their distance is less than a specified connectivity radius ($R = 0.015\,\mathrm{m}$ in the present work). For each pair of neighbouring particles i and j, an initial edge embedding h_{ij}^0 is calculated as a function of the relative position $\mathbf{x}_i - \mathbf{x}_j$, using another MLP ϕ_e:

$$h_{ij}^0 = \phi_e(\mathbf{x}_i - \mathbf{x}_j). \tag{2}$$

Processor: The processor performs M rounds of message passes over the graph ($M = 10$ in the present work) to propagate information between particles. At each message passing step m, the edge embeddings and the node embeddings are updated as follows:

$$\begin{aligned} h_{ij}^{m+1} &= f_e(h_i^m, h_j^m, h_{ij}^m), \\ h_i^{m+1} &= f_v\left(h_i^m, \sum_{j \in \mathcal{N}(i)} h_{ij}^{m+1}\right). \end{aligned} \tag{3}$$

Here, h_i^m and h_{ij}^m are the node and edge embeddings at step m, f_e and f_v are MLPs, and $\mathcal{N}(i)$ denotes the set of neighbouring particles of node i.

Decoder: After message passing, a decoder network f_d (an MLP) predicts the acceleration $\mathbf{a}_i$ for each particle from its final node embedding:

$$\mathbf{a}_i = f_d(h_i^M), \tag{4}$$

where M denotes the total number of messages passing rounds. The velocity and position of the particles are updated using explicit Euler integration.

To improve robustness and generalisation, random walk noise with standard deviation $\sigma_v = 3 \times 10^{-4}\,\mathrm{m\,s^{-1}}$ was added to the input velocities during training, as proposed in [5]. The model was trained by minimising the mean squared error between the predicted acceleration by GNN and ground-truth accelerations obtained from MPM simulations.

B. Material Point Method Simulations

The material point method is a hybrid Eulerian-Lagrangian computational technique for simulating the behaviour of solids, fluids, and granular materials [7]. In MPM, the material body is discretized into collections of Lagrangian particles, called material points, which carry mass, velocity, and other state variables. At each timestep, these particle quantities are transferred to a fixed Eulerian background mesh, where the governing equations of motion are solved. The updated quantities are then mapped to the particles. This hybrid combination of both Lagrangian and Eulerian perspectives enables MPM to efficiently handle large deformations, collisions, and complex boundary interactions.

C. Dataset Preparation

To generate the data for the model, two-dimensional MPM simulations were conducted within a rectangular domain of length $1.65\,\mathrm{m}$ and width $1.19\,\mathrm{m}$. Within this domain, a fixed ramp was constructed as a base for the simulated granular flow as shown in the Figure 1. The shape of the ramp is defined by a fourth-order polynomial profile. The

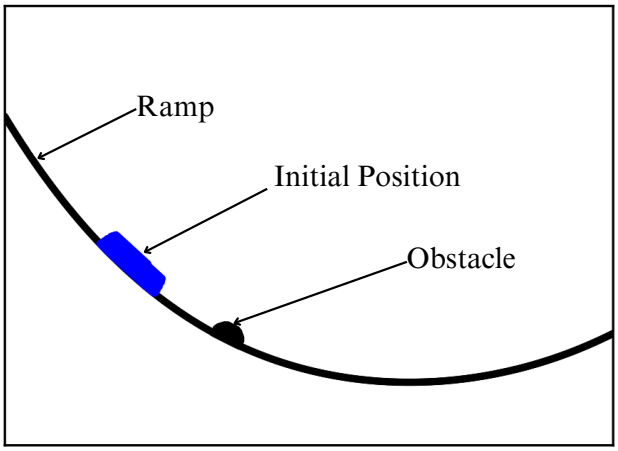

Figure 1: Initial configuration at $t = 0$ of MPM avalanche simulation. The initial granular particle positions, underlying ramp, and semi-circular obstacle are highlighted with arrows for clarity.

Table 1: Number of particles and obstacle types with their dimensions used in the training dataset.

Parameter	Range / Dimensions (in meters)
Number of particles	700–1000
Number of obstacles	0–2
Obstacle position	0.5–1.3 (x coordinate)
Semi-circular obstacle	Radius: 0.02–0.05 m
Rectangular obstacle	Width: 0.02–0.06 m; Height: 0.03–0.07 m

elevation of the ramp at each horizontal position x is given by:

$$\begin{aligned} y(x) = 0.1086 - 0.5179x + 1.3434x^2 \\ - 1.6513x^3 + 0.8923x^4, \end{aligned} \tag{5}$$

where x and y are in metres. The polynomial profile is adapted from the experimental configuration described by Daniel et al. [1]. The initial configuration of the MPM avalanche simulation at $t = 0$ is shown in Figure 1. Particles were assigned a density of $1538\,\mathrm{kg\ m^{-3}}$ (similar to that of silica sand) and a diameter of $1.82\,\mathrm{mm}$.

To keep a good representative sample of each parameter type as mentioned in Table 1, the dataset developed in this work consists of 1100 simulations, each saved as a sequence of 600 frames with a timestep of $0.0025\,\mathrm{s}$. Of these, 1000 records were used for training and 100 for testing. The number of particles, obstacle type, and obstacle position and dimensions were varied to produce each simulation for the dataset, as shown in Table 1. In the dataset, each simulation is stored in a three-dimensional tensor capturing the particle positions in the (x, y) plane across 600 timesteps.

III. RESULTS AND DISCUSSION

A. Flow Analysis

To qualitatively assess the performance of the trained GNN-based model, the time evolution of particle positions for a simulation from the test data is visualised. Figure 2 captures key moments throughout the avalanche movement and its interaction with a spherical obstacle. The black colour indicates the basal surface and the spherical obstacle, while the blue colour represents the granular material. The granular material is released from its initial position on a curved surface. As it moves downslope, the flow accelerates and collides with the spherical obstacle, resulting in some of the granular mass spilling over the obstacle and creating a jet. This flow–obstacle interaction leads to the formation of a dead zone, which enlarges over time and is characterized by the complete stoppage of granules near the base and upstream of the obstacle. Additionally, the granules that spill over the obstacle travel further as a ballistic projection before landing back on the bed. This behaviour has also been quantitatively observed in experiments [2]. This approach enables us to assess the model's capacity to propagate system dynamics and maintain physical consistency over extended time horizons.

Figure 2 also includes comparison of MPM solver simulation (left) with GNN prediction (right). It can be seen that the model can accurately predict not only the overall movement of the particles but also important details, such as the advancement of the avalanche front, the interaction of particles with obstacles, and the spreading of the particles over time, similar to what is observed in the true simulation.

B. Model Performance with Training

In this section, model training and its performance during training is reported. The model was trained for a total of 2 million steps. Model checkpoints were saved at 0.5 million, 1 million, 1.5 million, and 2 million steps to enable analysis of the model's progress. These checkpoints were labeled as V1, V2, V3, and V4, and are used throughout this section to compare training error, training time, and test error to find the most effective point to stop training.

Figure 3 shows how the training error changes during training. The training error is defined as the mean squared error between the normalised particle accelerations computed from the MPM simulation and those predicted by the GNN model. Particle accelerations in the MPM simulation were obtained by double differentiating particle positions with respect to time and were normalised before error computation. Figure 3 shows that training error drops quickly at the beginning and then levels off, reaching a low and stable value as training went on.

The relationship between the number of training steps and the training time is approximately linear, as shown in Figure 4. However, as the number of steps increases beyond 1.5 million, a slight deviation from linearity is observed, with the time required for 2 million steps rising to approximately 134 hours, which is higher than the earlier rate would suggest. This indicates that, beyond 1.5 million steps, each additional step incurs more time, which increases the computation overhead.

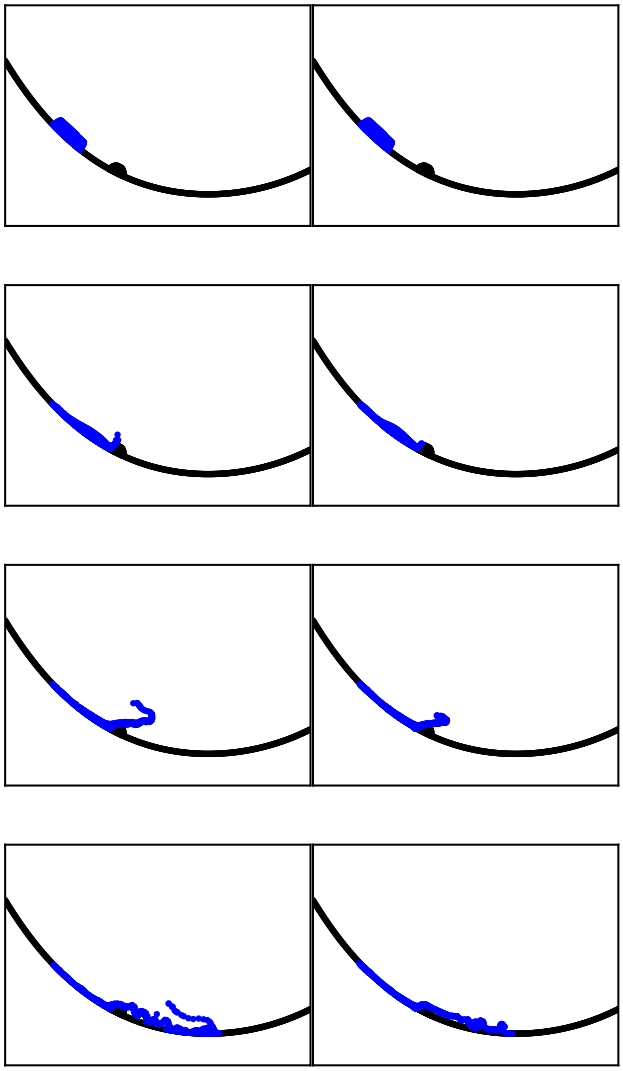

Figure 2: **Visualization of the simulation for selected timesteps, arranged in chronological order from top to bottom (timesteps 1, 80, 120, and 180). Particle count: 900; obstacle: spherical. For each row, the left plot shows the MPM simulation particle positions and the right plot shows the GNN prediction.**

After analysing the training loss and runtime, the model's performance on unseen data was examined. Figure 5 shows the test error, which is calculated as the mean squared error in position averaged across all simulations, particles, and timesteps in the test set. This positional error metric differs from the normalised acceleration error used during training. The test error drops quickly between 0.5 million and 1.5 million training steps, and then levels off. Beyond about 1.5 million steps, further training leads to only minor improvements, which suggests that the model has more or less reached its best possible performance for this setup.

Taken together, these results show that the GNN-based

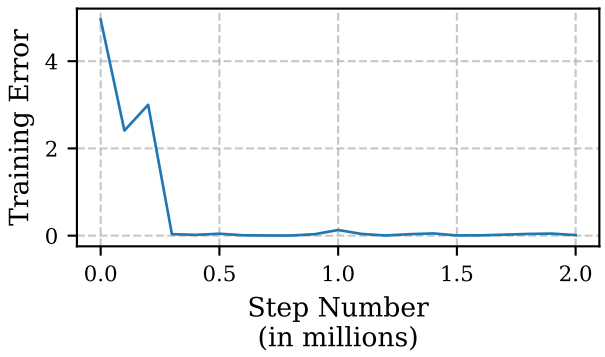

Figure 3: **Training error (sampled every 100k steps) versus number of training steps.**

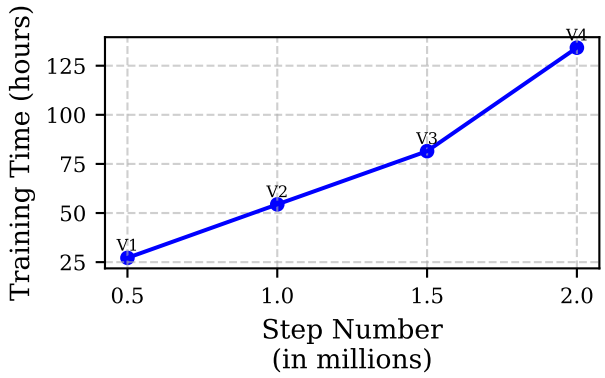

Figure 4: **Training time (hours) versus number of training steps for model variants V1–V4.**

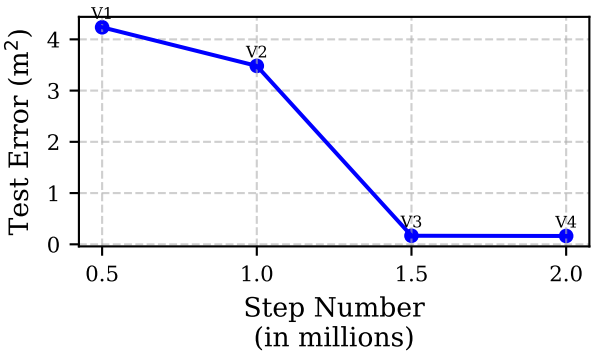

Figure 5: **Test error versus number of training steps for model variants V1–V4.**

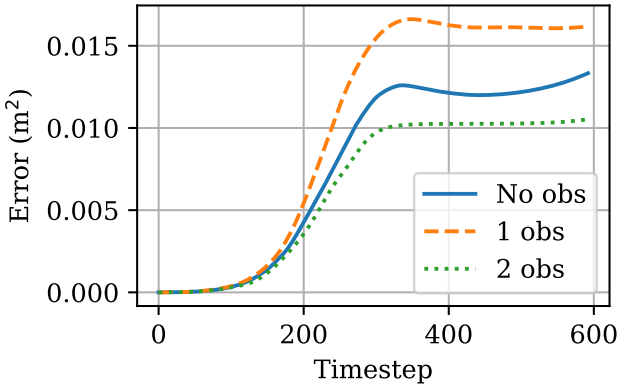

Figure 6: **Positional MSE for simulations with no obstacle, one obstacle, and two obstacles.**

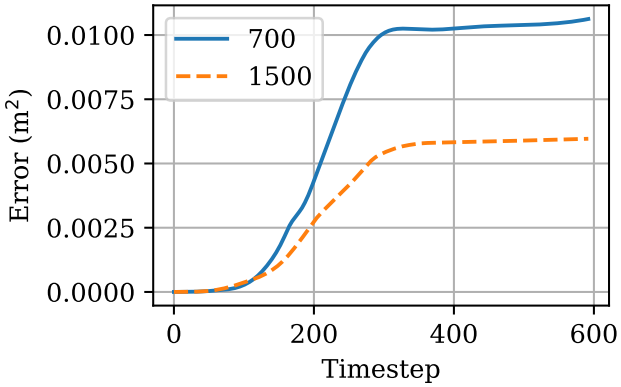

Figure 7: **Positional MSE for simulation with 700 and 1500 particles.**

model converged quickly and maintained stable performance, with most improvements happening before 1.5 million training steps. Since the test error changed very little after this point, and further training required much more computation time for only minor gains, the model saved at 1.5 million steps (V3) offers the best balance between accuracy and efficiency for this problem. For the rest of this work, the V3 model was used.

C. Model Accuracy

In this section, the accuracy of the V3 model is evaluated using the mean squared error, which is computed at each timestep across different simulations from the test set. The error is obtained by comparing the predicted and true particle positions at each timestep, and then averaged over all particles.

Figure 6 shows the model's MSE when evaluated with zero, one, and two obstacles. The presence of obstacles increased the prediction error, with the highest error for one obstacle, indicating that complex obstacle interactions present a challenge to the model.

Figure 7 illustrates the effect of particle count on the

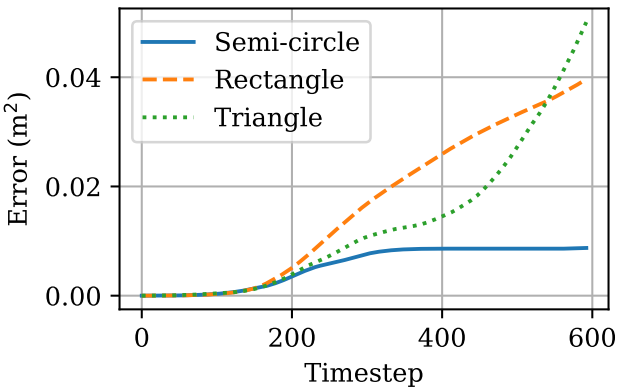

Figure 8: **Positional MSE for simulation with semi-circular, rectangular, and triangular obstacles.**

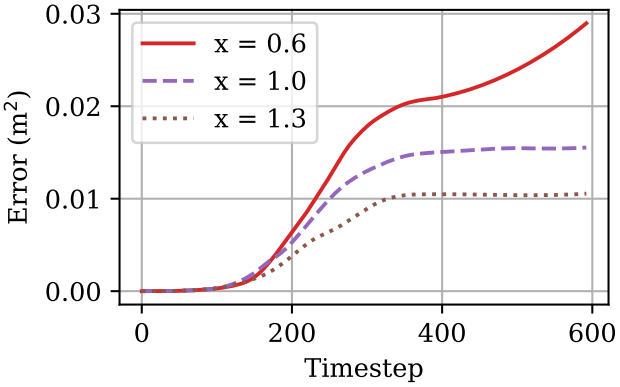

Figure 9: **Positional MSE for simulation with rectangular obstacles placed at different positions along the ramp ($x = 0.6$, $x = 1.0$, and $x = 1.3$).**

prediction error. Higher particle count yielded lower error, suggesting that the model performs better with denser particle representations.

Figure 8 presents the model accuracy for different obstacle shapes. The semi-circular obstacle shows the lowest error, while rectangular and triangular obstacles result in higher errors, especially at later timesteps, indicating that sharp edges or vertices in obstacles pose greater difficulty for the model. Notably, even though the triangular obstacle was not included in the training data, the model still maintains a low error throughout the simulation. This suggests that the model has learned the underlying physics and can generalise its predictions to unseen obstacle shapes, rather than relying solely on the specific geometries encountered during training. An upward trend in the MSE is observed in the later simulation timesteps as the error becomes compounded with each timestep.

Figure 9 shows the MSE for rectangular obstacles placed in three different positions. The error is highest when the obstacle is closest to the initial heap, and lowest when it is farthest downstream. This indicates that the position of the obstacle relative to the initial heap has an impact on the accuracy of the prediction of the GNN model.

Table 2: **Average simulation time comparison between MPM and GNN (NVIDIA Tesla V100 GPU) across different simulations of test data.**

Section	Case	Avg. Time (MPM) [s]	Avg. Time (GNN, GPU) [s]
Shape of Obstacle	Semi-circular	19.90	0.043
	Rectangular	18.64	0.043
	Triangular	17.40	0.045
Position of Obstacle	$x = 0.6$	19.55	0.045
	$x = 1.0$	17.74	0.046
	$x = 1.3$	17.34	0.044
Number of Obstacles	None	19.85	0.033
	One	21.48	0.040
	Two	23.01	0.049
Number of Particles	700	16.79	0.032
	1000	17.27	0.034
	1500	17.72	0.053

D. Computation Time

In this study, computation times for the MPM solver and the GNN model were compared across various simulations of test data. The GNN model was trained for a total of 81.5 hours, which should be regarded as a one-time investment. After the model was trained and loaded, the inference was performed to predict the outcomes of new simulations on test data. Here, inference time is defined as the duration required by the model to produce results for a given set of inputs. As shown in Table 2 and Figure 10, a significant reduction in simulation time was observed when the GNN model was utilized. Each simulation was completed by the GNN model over 400–600 times faster than by the MPM solver.

Although the absolute times for the GNN model varied slightly with scenario complexity, these differences remained negligible in practice. In contrast, the computational cost of the MPM solver increased more noticeably with particle count and obstacle complexity. As a result, the inference time for the GNN model remained nearly constant across all scenarios, while the MPM solver was observed to slow down as obstacle shape, position, and particle count became more complex.

IV. CONCLUSIONS

In this work, a graph neural network was trained to emulate the results of high-fidelity MPM simulations for granular avalanches at a fraction of the computational cost. Test errors below 0.17 were achieved within 1.5 million training steps, and inference times of 0.03–0.05 seconds per simulation were observed, compared to 17–23 seconds for the MPM solver. Speed-ups of over $400\times$ were obtained while maintaining accuracy across a range of obstacle shapes, positions, and particle densities. The model was found to converge

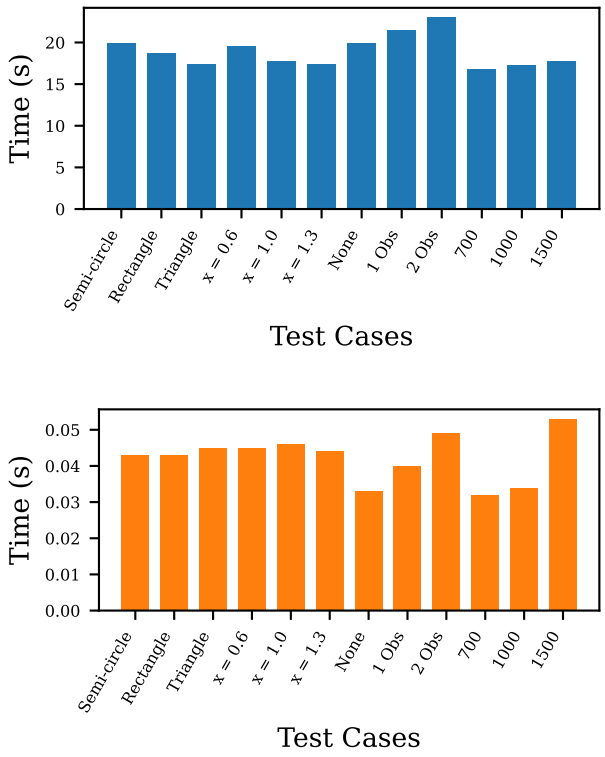

Figure 10: **Comparison of average simulation times between the MPM solver and the GNN model (NVIDIA Tesla V100 GPU) for various obstacle shapes, positions, counts, and particle numbers. The top panel corresponds to the MPM solver, and the bottom panel to the GNN.**

quickly, remain stable over long predictions, and generalise well to different scenarios.

However, some limitations should be noted. First, a large and diverse training dataset is required, and significant computational effort may be involved in preparing data for complex geometries and three-dimensional cases. Second, real-world applicability remains unverified, as calibration against experimental measurements has not yet been performed.

In future studies, the training dataset will be expanded to include more complex geometries and slope profiles, and the approach will be extended to three dimensions to further improve robustness and real-time applicability.

ACKNOWLEDGEMENTS

PSS gratefully acknowledges the Ministry of Education (MoE), Government of India, for the scholarship support during his M.Tech (Research) studies. Authors also acknowledge National Supercomputing Mission (NSM) for providing computing resources of 'PARAM Himalaya' at IIT Mandi, which is implemented by C-DAC and supported by the Ministry of Electronics and Information Technology (MeitY) and Department of Science and Technology (DST), Government of India.

NOMENCLATURE

$\mathcal{G}$	Graph of particles	–
$\mathcal{V}$	Set of nodes (particles)	–
$\mathcal{E}$	Set of edges (interactions)	–
$\mathbf{x}_i$	Position vector of particle i	m
$\mathbf{v}_i$	Velocity vector of particle i	$\mathrm{m\,s^{-1}}$
$\mathbf{a}_i$	Predicted acceleration of particle i	$\mathrm{m\,s^{-2}}$
c_i	Category label for particle i	–
h_i^m	Node embedding of particle i after m passes	–
h_{ij}^m	Edge embedding between particles i and j at step m	–
M	Total number of message-passing iterations	–
m	Message-passing iteration index	–
$\mathcal{N}(i)$	Neighbour set of node i	–
R	Connectivity radius	m
Δt	Simulation timestep size	s
σ_v	Std. dev. of noise	$\mathrm{m\,s^{-1}}$
ϕ_v	Encoder MLP for nodes	–
ϕ_e	Encoder MLP for edges	–
f_e	Edge-update MLP	–
f_v	Node-update MLP	–
f_d	Decoder MLP	–
x, y	Horizontal and vertical coordinates	m
t	Time	s

REFERENCES

[1] Daniel Caviedes-Voullième, Carmelo Juez, Javier Murillo, and Pilar García-Navarro, *2D dry granular free-surface flow over complex topography with obstacles. part i: Experimental study using a consumer-grade RGB-D sensor*, Computers and Geosciences **73** (2014), 177–197.

[2] Thierry Faug, *Depth-averaged analytic solutions for free-surface granular flows impacting rigid walls down inclines*, Physical Review E **92** (2015), no. 6, 062310.

[3] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang, *Physics-informed machine learning*, Nature Reviews Physics **3** (2021), no. 6, 422–440.

[4] Maziar Raissi, Paris Perdikaris, and George E Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics **378** (2019), 686–707.

[5] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia, *Learning to simulate complex physics with graph networks*, International conference on machine learning, PMLR, 2020, pp. 8459–8468.

[6] Jürg Schweizer, J Bruce Jamieson, and Martin Schneebeli, *Snow avalanche formation*, Reviews of Geophysics **41** (2003), no. 4.

[7] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle, *A material point method for snow simulation*, ACM Transactions on Graphics (TOG) **32** (2013), no. 4, 1–10.

[8] Yingxue Zhao, Haoran Li, Haosu Zhou, Hamid Reza Attar, Tobias Pfaff, and Nan Li, *A review of graph neural network applications in mechanics-related domains*, Artificial Intelligence Review **57** (2024), no. 11, 315.