Research paper

Graph Neural Network-based surrogate model for granular flows

Yongjin Choi [*], Krishna Kumar

Department of Civil Architectural and Environmental Engineering Austin, The University of Texas at Austin, Austin, 78712, TX, USA

ARTICLE INFO

ABSTRACT

Accurate simulation of granular flow dynamics is crucial for assessing geotechnical risks, including landslides and debris flows. Traditional numerical methods are limited by their computational cost in simulating large-scale systems. Statistical or machine learning-based models offer alternatives. Still, they are largely empirical, based on limited parameters. Due to their permutation-dependent learning, traditional machine learning-based models require huge training data to generalize. To resolve these problems, we use a graph neural network (GNN), a state-of-the-art machine learning architecture that learns local interactions. Graphs represent the state of dynamically changing granular flows and their interactions. We implement a multi-Graphics Processing Units (GPU) GNN simulator (GNS) capable of handling different material types. We demonstrate the capability of GNS by modeling granular flow interactions with barriers. GNS takes the granular flow's current state and predicts the next state using Euler explicit integration by learning the local interaction laws. We train GNS on different granular trajectories. We then assess its performance by predicting granular column collapse and interaction with barriers. GNS accurately predicts flow dynamics for column collapses with different aspect ratios and interaction with barriers with configurations unseen during training. GNS is up to a few thousand times faster than high-fidelity numerical simulators.

1. Introduction

Landslides cause extensive material displacement and significant infrastructure damage. Accurate modeling of granular flow runout is crucial to understanding the impact of landslides. Numerical methods, such as particle-based and continuum approaches, are often employed to assess landslide runouts. Particle-based approaches, like the Discrete Element Method (DEM) (Staron and Hinch, 2005; Kermani et al., 2015; Kumar et al., 2017a), can model grain–grain interactions but are limited to representative elemental volumes. Traditional continuum approaches, such as the Finite Element Method, can predict the initiation of such failures but suffer from mesh distortions when capturing runout dynamics. Hybrid Eulerian–Lagrangian approaches like the Material Point Method (MPM) (Mast et al., 2014; Kumar et al., 2017b) can simulate large-deformation flows without undergoing mesh distortions. However, the hybrid nature of MPM requires tracking both the grid and the material points, which is computationally expensive. Multiple full-scale simulations are necessary for a comprehensive evaluation of runout hazard scenarios. Similarly, a back analysis to estimate material parameters requires a broad parametric sweep involving hundreds to thousands of simulations. However, current state-of-the-art numerical methods are restricted to, at most, a few full-scale simulations, limiting our ability in scenario testing or back analysis.

An alternative to numerical simulations is to develop statistical or machine learning models to evaluate landslide risks. These surrogate models build correlations between landslide risks and their influencing factors. However, they often oversimplify the granular flow process by relying on simple empirical correlations and statistical patterns mapped on a low-dimensional domain without considering the complex dynamics of granular flows. Several studies adopt probabilistic approaches, such as Monte Carlo simulation and Bayesian analysis, to evaluate the landslide runout distance based on factors including topology and geology (Gao et al., 2021; Zeng et al., 2021; Sun et al., 2021; Zhao et al., 2022). Machine learning models can predict the travel distance and potential path of granular flows based on the geometry and ground properties (Durante and Rathje, 2021; Ju et al., 2022; Yang and Hambleton, 2021). Although researchers have been able to correlate the runout of granular flow based on statistical or data-driven techniques, these techniques do not explicitly consider granular flow dynamics— the actual physics governing the flow behavior. Thus, due to a lack of physics, these models do not generalize outside their training range, such as, modeling other boundary conditions or geometry.

In order to develop a surrogate model for granular flows that generalize beyond the training datasets, it is essential to understand the interaction laws governing the behavior of granular masses. Traditional

* Corresponding author.
  E-mail address: yj.choi@utexas.edu (Y. Choi).

machine learning models like Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) face difficulties learning these interactions as granular systems continuously evolve and rearrange. CNNs, while adept at learning spatially invariant features through pooling operations, are limited by their reliance on fixed, mesh-based structures. This limitation is significant in modeling granular flows where particle interactions and arrangements are neither fixed nor regular. MLPs can theoretically model dynamic systems by mapping their physical states to dynamics. However, their lack of permutation invariance, i.e., their output depends on the order of inputs, means an exponential increase in the required datasets ($O(n!)$) to map the entire parameter space of particle arrangements (Battaglia et al., 2018; Haeri and Skonieczny, 2022).

To address these limitations, we use graph neural networks (GNNs), a state-of-the-art machine learning architecture that enables permutation invariant learning (Battaglia et al., 2016, 2018; Sanchez-Gonzalez et al., 2020). GNNs represent the physical state of granular systems as graphs, where nodes symbolize individual particles or groups of particles, and edges depict their interactions or relational dependencies. This graph-based approach allows GNNs to learn directly from particle interactions, regardless of their spatial arrangement or sequence in the input data. It effectively captures the complex and dynamic nature of granular flows in a permutation-invariant way. GNNs adapt to structural changes in the system over time, effectively learning the underlying interaction laws.

We implement a multi-Graphics Processing Units (GPU) GNN simulator (GNS) capable of handling different material types. We demonstrate the capability of GNS by replicating the collapse of a granular column and its interaction with barriers. Granular column collapse is a simple physical experiment that captures the overall dynamics of large-scale runouts. GNS, trained on granular flow trajectories, successfully predicts the runout dynamics of column collapse and the interaction with barriers outside its training range and generalizes to upscaled domain sizes.

2. Methods

This section describes the individual components of GNS: graphs, graph neural networks (GNNs), and message passing.

2.1. Graph neural networks and message passing

2.1.1. Graphs

Graphs can represent interactions in physical systems (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2020). We represent the granular media as a graph $G = (V, E)$ consisting of a set of vertices ($v_i \in V$) representing the soil grains or aggregation of grains and edges ($e_{i,j} \in E$) connecting a pair of vertices (v_i and v_j) representing the interaction between the grains. Consider an example involving interaction between grains in a box (see Fig. 1). We encode the state of the physical system, such as the kinematics of grains and their interaction (Fig. 1a and Fig. 1d), as a graph (Fig. 1b and Fig. 1c). The vertices describe the position and velocity of the grains, and the edges describe the directional interaction between them, shown as arrows in Fig. 1b and Fig. 1c. The state of the grain i is represented as a vertex feature vector v_i. The vertex feature vector includes velocities, mass, and distance to the boundary. The edge feature vector $e_{i,j}$ includes information about the interaction between grains i and j such as the relative distance between the grains. Thus, we can store and process the state of granular bodies and their interactions as graphs.

Graphs offer a permutation-invariant form of encoding data, where the interaction between vertices is independent of the order of vertices or their position in Euclidean space. As graphs represent the interactions between grains as edge connections, graphs are permutation invariants. For example, by storing the relative positional information in $e_{i,j}$, rather than the absolute position, machine learning models operating on these networks learn the interaction behavior of different relative distances between grains. Therefore, graphs can efficiently represent the physical state of granular flow involving multi-grain interactions.

2.1.2. Graph neural networks (GNNs)

GNN takes a graph $G = (V, E)$ as an input, computes properties and updates the graph, and outputs an updated graph $G' = (V', E')$ with an identical structure, where V' and E' are the set of updated vertex and edge features (v'_i and $e'_{i,j}$). Message passing is the process of updating the graph by propagating information through it.

In the grains-in-a-box example, the GNN first takes the original graph $G = (V, E)$ (Fig. 1b) that describes the current state of the physical system (X_t). The GNN then updates the state of the physical system through message passing, which models the exchange of energy and momentum between the grains, and returns an updated graph $G' = (V', E')$ (Fig. 1c). We decode G', the output of GNN, to extract information related to the future state of the physical system (X_{t+1}), such as the next position or acceleration of the grains (Fig. 1d).

2.1.3. Message passing

Message passing consists of three operations: message construction (Eq. (1)), message aggregation (Eq. (2)), and the vertex update function (Eq. (3)).

$$e'_{i,j} = \phi_{\Theta_\phi} \left(v_i, v_j, e_{i,j} \right) \tag{1}$$

$$\bar{v}_i = \sum_{j \in N(i)} e'_{i,j} \tag{2}$$

$$v'_i = \gamma_{\Theta_\gamma} \left(v_i, \bar{v}_i \right) \tag{3}$$

The subscript Θ_ϕ and Θ_γ represent a set of learnable parameters in each computation. The message construction function ϕ_{Θ_ϕ} (Eq. (1)) takes the feature vectors of the receiver and sender vertices (v_i and v_j) and the feature vector of the edge connecting them ($e_{i,j}$) and returns an updated edge feature vector $e'_{i,j}$ as the output. ϕ_{Θ_ϕ} is a matrix operation including the learnable parameter Θ_ϕ. The updated edge feature vector $e'_{i,j}$ is the message sent from vertex j to i. Fig. 2a shows an example of constructing messages on edges directed to vertex 0 originating from vertices 1, 2, and 3 ($e'_{0,1}, e'_{0,2}, e'_{0,3}$). Here, we define the message construction function ϕ_{Θ_ϕ} as $\left((v_i + v_j) \times e_{i,j} \right) \times \Theta_\phi$. The updated feature vector $e'_{0,1}$ is computed as $\left((v_0 + v_1) \times e_{0,1} \right) \times \Theta_\phi$, where v_0 and v_1 are the receiver and sender vertex feature vectors, and $e_{0,1}$ is their edge feature vector. Suppose we assume all values of Θ_ϕ are 1.0 for simplicity, we obtain $e'_{0,1} = (([1, 0, 2] + [1, 3, 2]) \times [2, 1, 0]^T) \times 1 = [4, 3, 0]$. Similarly, we can compute the messages $e'_{0,2} = [0, 3, 9]$ and $e'_{0,3} = [3, 4, 9]$.

The next step in message passing is the message aggregation $\Sigma_{j \in N(i)}$ (Eq. (2)), where $N(i)$ is the set of sender vertices j related to vertex i). It collects all the messages directing to vertex i and aggregates those into a single vector with the same dimension as the aggregated message ($\bar{v}_i$). The aggregation rule can be element-wise vector summation or averaging; hence it is a permutation invariant computation. In Fig. 2a, the aggregated message $\bar{v}_0 = [7, 10, 18]$ is the element-wise summation of the messages directing to vertex 0 as $\bar{v}_0 = e'_{0,1} + e'_{0,2} + e'_{0,3}$.

The final step of the message passing is updating vertex features using Eq. (3). It takes the aggregated message ($\bar{v}_i$) and the current vertex feature vector v_i, and returns an updated vertex feature vector v'_i, using predefined vector operations including the learnable parameter Θ_γ. Fig. 2b shows an example of the update at vertex 0. Here, we define the update function γ_{Θ_γ} as $\Theta_\gamma (v_i + \bar{v}_i)$. The updated feature vector v'_0 is computed as $\Theta_\gamma (v_0 + \bar{v}_0)$. Assuming all parameters in Θ_γ are 1.0 for simplicity, we obtain $v'_0 = v_0 + \bar{v}_0 = [1, 0, 2] + [7, 10, 18] = [8, 10, 20]$. Similarly, we update the other vertex features (v'_1, v'_2, v'_3).

After message passing, the graph vertex and edge features (v_i and $e_{i,j}$) are updated to v'_i and $e'_{i,j}$. The GNN may include multiple message passing steps to propagate the information further through the network.
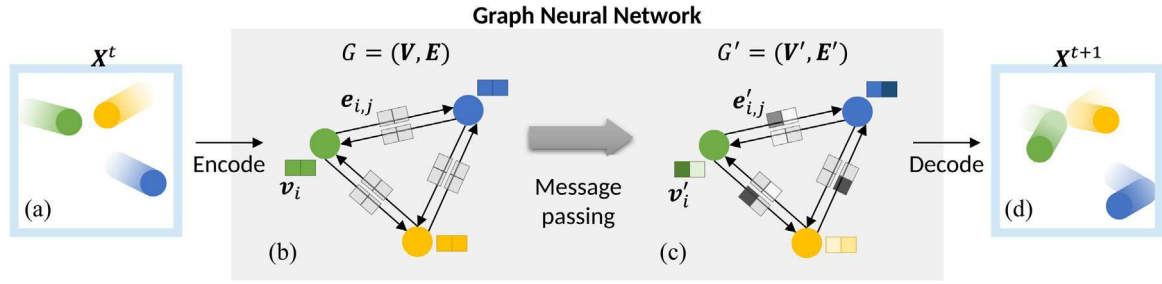
Fig. 1. An example of a graph and graph neural network (GNN) that process the graph (modified from (Battaglia et al., 2018)): (a) A state of the current physical system (X_t) where the grains are bouncing in a box boundary; (b) Graph representation of the physical system (G). There are three vertices representing grains and six edges representing their directional interaction shown as arrows; (c) The updated graph (G') that GNN outputs through message passing; (d) The predicted future state of the physical system (X_{t+1}) (i.e., the positions of the grains at the next timestep) decoded from the updated graph.
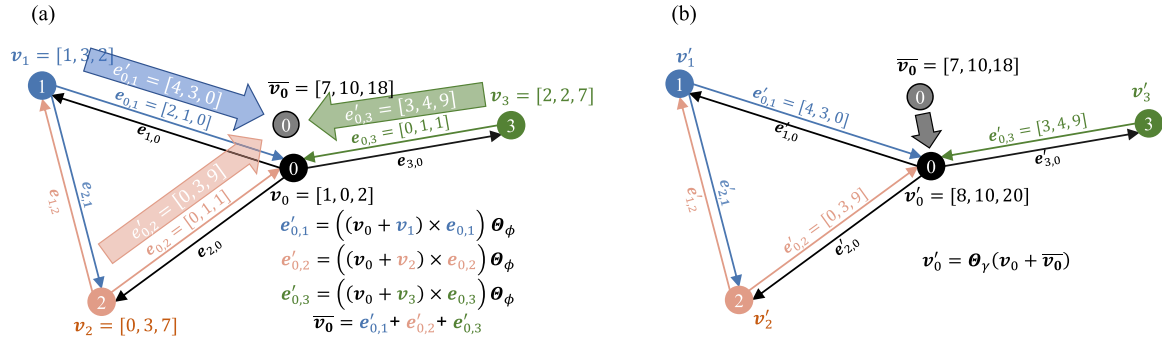

Fig. 2. An example of message passing on a graph: (a) message construction directing to receiver vertex 0 ($e'_{0,1}$, $e'_{0,2}$, $e'_{0,3}$) and the resultant aggregated message ($\bar{v}_0$); (b) feature update at vertex 0 using $\bar{v}_0$. Note that we assume Θ_ϕ and Θ_r are 1.0 for the convenience of calculation.

Unlike the example shown above, where we assume a constant value of 1.0 for the learnable parameters, in a supervised learning environment, the optimization algorithm will find a set of the best learnable parameters ($\Theta_\phi, \Theta_\gamma$) in the message passing operation.

2.2. Graph neural network-based simulator (GNS)

In this study, we use GNN as a surrogate simulator to model granular flow behavior. Fig. 3 shows an overview of the general concepts and structure of the GNN-based simulator (GNS) proposed by Sanchez-Gonzalez et al. (2020). Consider a granular flow domain represented as material points (Fig. 3a), which represent the collection of grains. In GNS, we represent the physical state of the granular domain at time t with a set of x_i^t describing the state and properties of each material point. The GNS takes the current state of the granular flow $x_t^i \in X_t$ and predicts its next state $x_{i+1}^t \in X_{t+1}$ (Fig. 3a). The GNS consists of two components: a parameterized function approximator d_Θ and an updater function (Fig. 3b). The function approximator d_Θ takes X_t as an input and outputs dynamics information $y_i^t \in Y_t$. The updater then computes X_{t+1} using Y_t and X_t. Fig. 3c shows the details of d_Θ which consists of an encoder, a processor, and a decoder. The encoder (Fig. 3-c1) takes the state of the system X_t and embeds it into a latent graph $G_0 = (V_0, E_0)$ to represent the relationships between material points. The vertices $v_i^t \in V_0$ contain latent information of the current state of the material point, and the edges $e_{i,j}^t \in E_0$ contain latent information of the pair-wise relationship between material points. Next, the processor (Fig. 3-c2) converts the input graph G_0 to the output graphs G_M through M stacks of message-passing GNN ($G_0 \rightarrow G_1 \rightarrow \cdots \rightarrow G_M$). The message passing computes the interaction between vertices. Finally, the decoder (Fig. 3-c3) extracts the dynamics of the points (Y^t) from G_M, such as the acceleration of the physical system. The entire simulation (Fig. 3a) involves running GNS surrogate model through K timesteps predicting from the initial state X_0 to X_K ($X_0, X_1, \ldots, X_K$), updating at each step ($X_t \rightarrow X_{t+1}$). We call this successive prediction from GNS the "rollout".

In the following sections, we explain the details of our input X^t (Fig. 3a), the encoder, processor, and decoder in d_Θ (Fig. 3c), and how we compute X^{t+1} from X^t using the GNS updater function (Fig. 3b).

2.2.1. Input

The input to the GNS, $x_i^t \in X^t$ (Eq. (4)), is a vector consisting of the current material point position p_i^t, the material point velocity context $\dot{p}_i^{\leq t}$, information on boundaries b_i^t, and material point type embedding f. The current state x_i^t will be used to construct vertex feature (v_i^t) (Eq. (6)).

$$x_i^t = [p_i^t, \ \dot{p}_i^{\leq t}, \ b_i^t, \ f, \ m] \tag{4}$$

The velocity context $\dot{p}_i^{\leq t}$ includes the current and previous material point velocities for n timesteps $[\dot{p}_i^{t-n}, \ldots, \dot{p}_i^t]$ with $n + 1$ velocities. We use $n = 4$ to include sufficient velocity context in the vertex feature v_i^t. Sanchez-Gonzalez et al. (2020) show that having $n > 1$ significantly improves the model performance. We compute the velocities using the finite difference of the position sequence (i.e., $\dot{p}_i^t = (p_i^t - p_i^{t-1}) / \Delta t$). b_i^t is boundary information. For a 2D problem, b_i^t has four components, each indicating the distance between material points and the four walls. We normalize b_i^t by the connectivity radius R which defines the interaction zone, explained in the next section, and restrict it between -1.0 to 1.0. b_i^t is used to evaluate the role of boundary friction on material points. f is a vector embedding distinguishing material point types such as deformable or rigid material. m is an optional feature that represents the material properties of the corresponding material points. If the training data includes different material properties for each material point, we can simply add relevant features here.

The GNS model accounts for the friction between vertices representing material points and boundaries by monitoring their distance using b_i^t as outlined in Eq. (4). GNS predicts the frictional acceleration when the nodal boundary distance is less than the connectivity radius R. When the boundary distance is larger than R, we clip the b_i^t to -1 or 1, signaling that boundary friction does not directly affect the
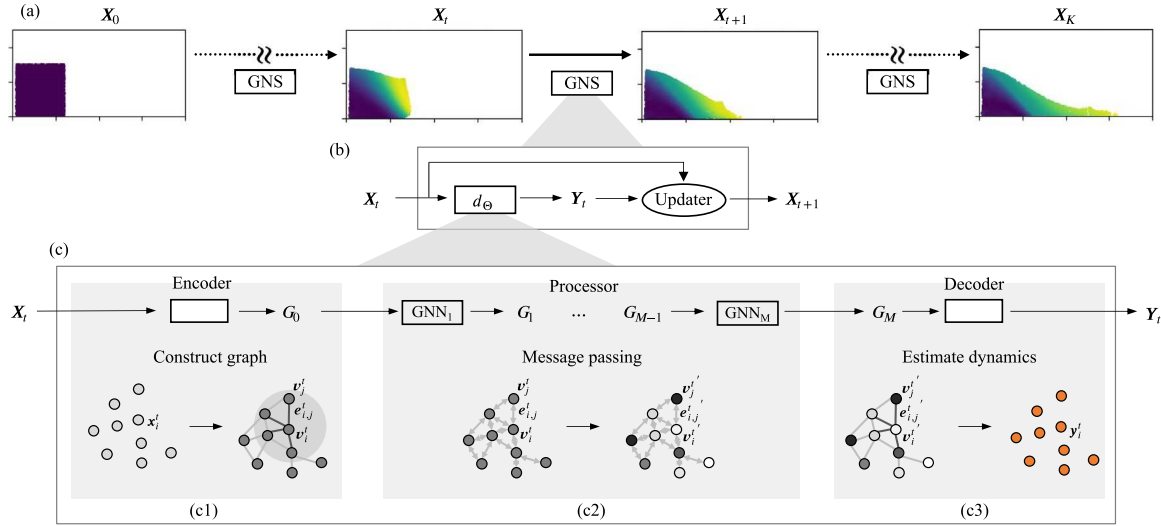
Fig. 3. The structure of the graph neural network (GNN)-based physics simulator (GNS) for granular flow (modified from (Sanchez-Gonzalez et al., 2020)): (a) The entire simulation procedure using the GNS, (b) The computation procedure of GNS and its composition, (c) The computation procedure of the parameterized function approximator d_Θ and its composition.

material points at this distance. During training, GNS learns the interaction between the material points' boundary distance and frictional acceleration from training data. The MPM training data models the tangential friction force as a function of the normal acceleration and the friction coefficient. With the mass of the material points known, we can compute the frictional forces from the accelerations predicted near the boundary.

We define the interaction between material points i and j as $r^t_{i,\ j}$ using the distance and displacement of the material points in the current timestep (see Eq. (5)). The former reflects the level of interaction, and the latter reflects its spatial direction. $r^t_{i,\ j}$ will be used to construct edge features ($e^t_{i,j}$). Although the distance provides an additional physical context about the interactions, we found that omitting it from Eq. (5) does not necessarily harm the model's performance since the distance can be calculated with the displacement. Specifically, the model with the distance feature shows the mean squared error of 0.0016 m to 0.0018 m, whereas the model without it shows 0.0014 m to 0.0017 m, when we validate the model's performance in predicting the evolution of material point positions.

$$r^t_{i,j} = \left[\left(p^t_i - p^t_j \right),\ \| p^t_i - p^t_j \| \right] \tag{5}$$

2.2.2. Encoder
The vertex and edge encoders (ε^v_Θ and ε^e_Θ) convert x^t_i and $r^t_{i,\ j}$ into the vertex and edge feature vectors (v^t_i and $e^t_{i,j}$) (Eq. (6)) and embed them into a latent graph $G_0 = (V_0,\ E_0)$, $v^t_i \in V_0$, $e^t_{i,j} \in E_0$.

$$v^t_i = \varepsilon^v_\Theta \left(x^t_i \right), \qquad e^t_{i,j} = \varepsilon^e_\Theta \left(r^t_{i,j} \right) \tag{6}$$

We use a two-layered 128-dimensional multi-layer perceptron (MLP) for the ε^v_Θ and ε^e_Θ. The MLP and optimization algorithm search for the best candidate for the parameter set Θ that estimates a proper way of representing the physical state of the material points and their relationship which will be embedded into G_0.

The edge encoder ε^v_Θ uses x^t_i (Eq. (4)) without the current position of the material point (p^t_i), but with its velocities ($\dot{p}^{\le t}_i$), as velocity governs the momentum, and the interaction dynamics is independent of the absolute position of the material points. Rubanova et al. (2021) confirmed that including position causes poorer model performance. We only use p^t_i to predict the next position p^{t+1}_i based on the predicted velocity $\dot{p}^{t+1}_i$ using Explicit Euler integration.

We consider the interaction between two material points by constructing edges between them all pairs of vertices located within a certain distance called connectivity radius R (see the shaded circular area in Fig. 3b). The connectivity radius is a critical hyperparameter that governs how effectively the model learns the local interaction. R should be sufficiently large to include the local interaction between material points and capture the simulation domain's global dynamics.

2.2.3. Processor
The processor performs message passing (based on Eq. (1) to Eq. (3)) on the initial latent graph (G_0) from the encoder for M times ($G_0 \rightarrow G_1 \rightarrow \cdots \rightarrow G_M$) and returns a final updated graph G_M. We use two-layered 128-dimensional MLPs for both the message construction function ϕ_{Θ_ϕ} and vertex update function γ_{Θ_r}, and element-wise summation for the message aggregation function $\Sigma_{j \in N(i)}$ in Eq. (1) to Eq. (3). We set $M = 10$ to ensure sufficient message propagation through the network. These stacks of message passing model information propagation through the network of material points.

2.2.4. Decoder
The decoder δ^v_Θ extracts the dynamics $y^t_i \in Y^t$ of the material points from the vertices $v^{t\prime}_i$ (Eq. (7)) using the final graph G_M. We use a two-layered 128-dimensional MLP for δ^v_Θ, which learns to extract the relevant dynamics for material points from G_M.

$$y^t_i = \delta^v_\Theta \left(v^{t\prime}_i \right) \tag{7}$$

2.2.5. Updater
We use the dynamics y^t_i to predict the velocity and position of the material points at the next timestep ($\dot{p}^{t+1}_i$ and p^{t+1}_i) based on Euler integration (Eq. (8) and Eq. (9)), which makes y^t_i analogous to acceleration $\ddot{p}^t_i$.

$$\dot{p}^{t+1}_i = \dot{p}^t_i + y^t_i \Delta t \tag{8}$$

$$p^{t+1}_i = p^t_i + \dot{p}^{t+1}_i \Delta t \tag{9}$$

Based on the new position and velocity of the material points, we update $x^t_i \in X^t$ (Eq. (4)) to $x^{t+1}_i \in X^{t+1}$. The updated physical state X^{t+1} is then used to predict the position and velocity for the next timestep.

The updater imposes inductive biases, such as an inertial frame, on GNS to force it only to learn the interaction dynamics, improving learning efficiency. A traditional neural network learns both the update scheme and the interaction dynamics:

$$p^{t+1} = NN(p^t, v^t). \tag{10}$$

Table 1

Details of the Material Point Method (MPM) simulation geometries and properties used for generating the training datasets.

Property		Dataset		
		Granular column collapse	Multi-material granular column collapse	Granular flow with barriers
Simulation boundary		1.0×1.0 m	1.0×1.0 m	$0.8 \times 0.8 \times 0.8$ m
MPM element length		0.025×0.025 m	0.01×0.01 m	$0.03125 \times 0.03125 \times 0.03125$ m
Material point configuration		25,600 points/m^2	40,000 points/m^2	262,144 points/m^3
Granular mass geometry		0.2×0.2 and 0.3×0.3 m	0.2×0.2 to 0.4×0.4 m	$0.2 \times 0.2 \times 0.2$ to $0.4 \times 0.4 \times 0.4$ m
Max. number of particles		2.3K	6.4K	17K
Barrier geometry		None	None	$0.1 \times 0.3 \times 0.1$ m
Simulation duration (timesteps)		400 (dt=0.0025 s)	400 (dt=0.0025 s)	350 (dt=0.0025 s)
	Model	Mohr–Coulomb	Mohr–Coulomb	Mohr–Coulomb
	Density	1,800 kg/m^3	1,800 kg/m^3	1,800 kg/m^3
	Youngs modulus	2 GPa	2 GPa	2 GPa
Material property	Poisson ratio	0.3	0.3	0.3
	Friction angle	30°	15, 17.5, 22.5, 30, 37.5, 45°	35°
	Cohesion	0.1 kPa	0.1 kPa	None
	Tension cutoff	0.05 kPa	0.05 kPa	None

Whereas, using an inertial prior, we force the GNS only to learn the interaction dynamics, by hardcoding the update function:

$$p^{t+1} = p^t + v^t \cdot \Delta t + NN(p^t, v^t). \tag{11}$$

Nevertheless, GNS does not directly predict the next position from the current position and velocity (i.e., $p_i^{t+1} = GNS\left(p_i^t, \dot{p}_i^t\right)$) which has to learn the static motion and inertial motion. Instead, it uses (1) the inertial prior (Eq. (8)) where the prediction of the next velocity $\dot{p}_i^{t+1}$ should be based on the current velocity $\dot{p}_i^t$ and (2) the static prior (Eq. (9)) where the prediction of the next position p_i^{t+1} should be based on the current position p_i^t. These make GNS focus on learning unknown dynamics by hardcoding known physics. Since GNS learns the dynamics of material points through interactions independent of absolute position, GNS is generalizable to other geometric conditions.

3. Training and evaluation

We now train the GNS to predict granular column collapse and granular flow interacting with barriers. This section explains how we generate training data, details of the training process, and how we evaluate the performance of the GNS.

3.1. Material point method (MPM)

We utilize the Material Point Method (MPM) to generate the GNS training dataset of granular flow simulations. The MPM is a hybrid Eulerian–Lagrangian approach designed for modeling large-deformation flows (Soga et al., 2016). In the MPM, a continuum body is represented by individual material points that traverse a static background grid. The governing equation is solved at the nodes, and the updated velocity field is subsequently mapped back to the material points. We employ the position information stored in these material points to construct the current state X^t in the GNS. For more information on MPM refer to Soga et al. (2016) .

3.2. Datasets

3.2.1. Granular column collapse

We prepare three types of datasets (Table 1): granular column collapse with a single material property, granular column collapse with multiple material properties, and granular flow interacting with barriers in three-dimensional space.

For the granular column collapse case, the training datasets include 26 granular flow trajectories of square-shaped granular mass in a two-dimensional box boundary simulated by the MPM explicit time integration method using the CB-Geo MPM code (Kumar et al., 2019). Each simulation has a different initial configuration regarding the size

of the square granular mass, position, and velocity. Table 1 presents the details of the simulation configurations. The datasets are published on DesignSafe (Kumar and Choi, 2023). Appendix shows all the training trajectories with different initial configurations and initial velocities.

We also create the validation datasets to check if the model experiences overfitting. The datasets include seven trajectories of randomly picked rectangular-shaped granular mass with different initial configurations not included in the training datasets.

Additionally, we prepare another dataset that involves granular column collapse with multiple friction angles (ϕ) to evaluate the ability of GNS to learn diverging granular flow behaviors depending on the friction angle of the material. It includes MPM simulations with five different friction angles ($\phi = 15°$, $22.5°$, $30°$, $37.5°$, $45°$) where each case has 60 simulations. To include the material-dependent characteristic in GNS, we simply use the normalized friction angle ($\tan \phi$) as m in Eq. (4) for the additional vertex feature. Similar to the dataset with a single friction angle, the initial geometry of the granular mass is restricted to the square shape.

3.2.2. Granular flow with barriers

For the barrier interaction case, the training datasets include 890 granular flow trajectories of granular mass interaction with barriers in a three-dimensional box boundary. We use Taichi MPM code (Hu et al., 2019) with explicit time integration to generate the simulation data. The initial geometry of the granular mass is limited to a cube shape with varying lengths and initial velocities. For each simulation, the granular material interacts with one or two $0.1 \times 0.3 \times 0.1$ m (width, height, and length) shaped barriers in $0.8 \times 0.8 \times 0.8$ m domain. The barriers are modeled as a rigid body. The details of the simulation configurations are presented in Table 1.

3.3. Training

Our GNS has a learnable parameter set Θ. We train Θ to minimize the loss calculated as the mean squared error (MSE) between y_i^t (predicted proxy-acceleration) and the ground truth acceleration $\ddot{p}_i^t$ for all material points $i = 1, 2, \ldots, N$ as shown in Eq. (12) based on gradient ($\nabla loss_\Theta$)-based optimizer, Adam (Kingma and Ba, 2014).

$$loss_\Theta = \frac{1}{n} \sum_{i=1}^{N} \left(y_i^t - \ddot{p}_i^t\right)^2 \tag{12}$$

For training the GNS, we have to set hyperparameters to learn the flow behavior from the training trajectories properly. The first key hyperparameter is the connectivity radius R, which governs the model's capacity to learn the interactions of material points. For each model that we train (granular column collapse and barrier interaction model), we employ different values of R to consider a balance between the
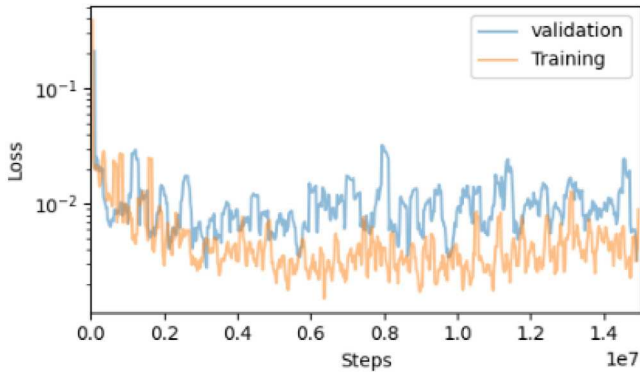
Fig. 4. Evolution of GNS loss in training and validation with training steps.

associated training costs and model capacity. For the granular column collapse model, we choose $R = 0.030$ m, and for the barrier interaction model, we choose $R = 0.025$, which includes about 9 to 10 and 3 to 4 material points along the diameter, respectively. The circular and spherical area defined by R for each model incorporates approximately 70 and 17 material points inside. For the barrier interaction model, we design to have a smaller R, and correspondingly, fewer material points inside it, to avoid excessive edge formation, since the model is defined in the three-dimensional domain. Another important hyperparameter is the Gaussian noise value for perturbing the ground truth position in the training trajectories. Since every predicted position for each timestep is based on the previous prediction, which includes a prediction error, the simulation over the large timesteps is subjected to an exponential error accumulation. To avoid this issue, we train the model on input positions with Gaussian noise that emulates the prediction error made by a one-step prediction ($X_t \rightarrow X_{t+1}$). The inclusion of noise in training leads to more rigorous long-rollout predictions.

We use the learning rate (lr) decay with the initial value of 10^{-4} and decay rate of 0.1 ($lr = 10^{-4} \times 0.1^{step/5 \times 10^6}$) for more stable convergence. We use the batch size of two, i.e., X_t from two different trajectories are used simultaneously in updating the learnable parameters. We enable a multi-GPU training algorithm based on data-distributed parallelism. It distributes GNS models and datasets across GPUs, and the training is conducted in parallel for each GPU. In this way, the training process can observe as many datasets as the number of GPUs at each training step. For information on the scalability of the GNS algorithm, refer to Kumar and Vantassel (2022).

For the granular column collapse model training, we investigate if the model experiences overfitting by plotting the loss history (Fig. 4) for the training and validation datasets evaluated for every 10 K training steps. The training and validation losses show a drastic decrease until 2M steps. After that, the validation loss tends to remain slightly larger than the training loss. Fig. 4 shows no overfitting during the training.

3.4. Granular column collapse

We trained the GNS to predict the collapse of a granular column (as studied by Lajeunesse et al. (2004), Lube et al. (2005)). Fig. 5 shows the granular column collapse experiments to evaluate its ability to replicate granular flow dynamics. Granular column collapse is a simple physical experiment that captures the transient response of granular flow dynamics. The experiment involves the collapse of a granular column of initial height H_0 and length L_0 on a flat surface due to gravity. As the gate holding the column is removed, the granular material destabilizes, resulting in a runout. We measure the final runout deposit with the final height H_f and runout L_f.

The runout of the granular column is governed by the initial aspect ratio ($a = H_0/L_0$) (Staron and Hinch, 2005; Kumar, 2015). For short

columns ($a \lesssim 2$) (Fig. 5a), the soil mass fails along the flanks of the column above a well-defined failure surface (dashed line). The soil mass beneath the failure surface remains in static equilibrium throughout the collapse forming a truncated conical shape. With the increase in aspect ratio, the portion of the sliding mass above the failure surface increase, and the static part becomes smaller, forming a conical shape. For tall columns ($a \gtrsim 2$) (Fig. 5b), the majority of the soil mass is involved in the collapse, and it initially experiences a free fall due to gravitational acceleration. As the falling mass reaches the failure surface, the vertical kinetic energy is converted to horizontal acceleration, resulting in a longer runout distance than the short column (Fig. 5a).

In addition, researchers (Kumar, 2015; Staron and Hinch, 2005; Kermani et al., 2015; Utili et al., 2015) observed a transition zone where the flow dynamics change from short to tall columns. The normalized runout ($(L_f - L_0)/L_0$) of a granular column is largely governed by the aspect ratio (a). The normalized runout represents how far the granular mass runs out before reaching the final deposit state compared to the initial length of the column. Short columns show a linear relationship with the initial aspect ratio. In contrast, tall columns have a power-law relationship with the initial aspect ratio.

Another factor that affects the runout is the friction angle. Nguyen et al. (2020) found that the increase in the friction angle tends to reduce runout since more shear stress is required to mobilize the granular mass.

The GNS was trained only on the aspect ratio of 1.0. However, we evaluate its performance in predicting the runout dynamics of other aspect ratios by comparing the GNS predictions with the MPM simulations. Table 2 presents the test cases for evaluating GNS performance trained on the single friction angle dataset.

3.5. Granular flow interacting with barriers

We test the GNS trained on granular flows interacting with barriers to predict its behaviors for debris-resisting baffles (Choi et al., 2014, 2015; Yang et al., 2020). Debris-resisting baffles are rigid, flow-impeding structures installed perpendicular to potential landslide debris flow paths to mitigate excessive runout and related hazards downstream.

Baffles work by slowing down the runout as it impacts each baffle, causing the flow to lose energy. As the debris diverges and passes through openings between baffles and hits the subsequent baffles, the energy of the flow dissipates. The energy dissipation caused by the baffles is quantified by the difference between the sum of kinetic and potential energy of the flow upstream and downstream of the baffles. It measures the general performance of the baffles for regulating energy from the flow.

Baffles can promote the deposition of material behind them and cause backwater effects upstream. The backwater effect refers to the increase in flow depth upstream of an obstruction like the baffles. A high rise of the upstream flow can cause hazardous overflow if it exceeds the baffle height, which can be uncontrollable. Therefore, upstream depth is an important criterion for baffle performance.

We evaluate the performance of GNS in terms of predicting overall runout dynamics, energy evolution, and upstream depth. Our training data only includes the cube-shaped granular mass geometry interacting with one or two barriers with the size of $0.1 \times 0.3 \times 0.1$ m in the $0.8 \times 0.8 \times 0.8$ m simulation domain. However, the prediction (i.e., flow with debris-resisting baffles) is performed with a cuboid-shaped granular mass colliding three barriers with the size of $0.15 \times 0.30 \times 0.15$ m in the $1.8 \times 0.8 \times 1.8$ m simulation domain, which is not seen during the training.

4. Results and discussions

In this section, we show the prediction performance of GNS on granular column collapse and granular flow interacting with barriers.
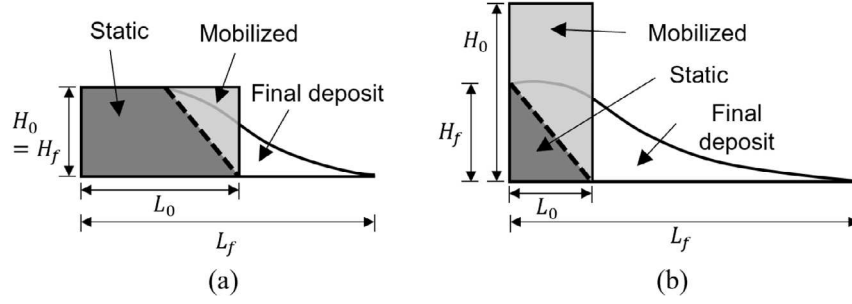
Fig. 5. Schematics for the granular column collapse configuration and its behavior on the aspect ratio.

Table 2
Granular column collapse simulation cases for testing GNS.

Test case		$H_0 \times L_0$ (m)	Duration (timesteps)	Simulation boundary (m)	Number of material points
Short columns	$a = 0.5$	0.2×0.4	400	X: 0 to 1.0 Y: 0 to 0.5	1956
	$a = 0.8$	0.24×0.30	400	X: 0 to 1.0 Y: 0 to 0.5	1824
	$a = 1.0$	0.30×0.30	400	X: 0 to 1.0 Y: 0 to 0.5	2304
Tall columns	$a = 2.0$	0.30×0.15	400	X: 0 to 1.0 Y: 0 to 0.5	1152
	$a = 3.0$	0.36×0.12	400	X: 0 to 1.0 Y: 0 to 0.5	1106
	$a = 4.0$	0.35×0.075	400	X: 0 to 1.0 Y: 0 to 0.5	576
Up-scaled	$a = 0.8$	0.36×0.45	500	X: 0 to 1.5 Y: 0 to 1.0	5120

Table 3
Normalized runout from MPM and GNS depending on aspect ratios and corresponding prediction error.

Aspect ratio, a	Normalized runout		Runout error (%)
	MPM	GNS	
0.5	0.831	0.811	2.48
0.8	1.444	1.445	0.06
1.0	2.071	2.152	3.78
2.0	3.892	3.682	5.70
3.0	5.620	5.341	5.23
4.0	5.753	6.070	5.21

Fig. 6. Normalized runout distance $((L_f - L_0)/L_0)$ with different aspect ratios (a).

4.1. Granular column collapse

We evaluate the GNS predictions of granular column collapse trained on a single friction described in Table 1 against the MPM simulations in terms of the (1) geometry of sliding mass, (2) evolution of runout and height with time, and (3) energy evolution during the collapse. Fig. 6 shows the normalized runout $((L_f - L_0)/L_0)$ predictions of GNS for different aspect ratios in comparison with MPM. L_f is the distance from the left wall to the material point that runs out the farthest, as shown in Fig. 5. Previous research observed a transition zone for the relationship between the normalized runout and aspect ratio that distinguishes short-column from tall-column dynamics. For both GNS and MPM, we observe the transition around an initial aspect ratio $a = 1.2$ (Fig. 6). Table 3 summarizes the errors between GNS predictions and MPM simulations for different aspect ratios. In general, the GNS runout prediction is within 5% of the MPM runout estimate. Fig. 6 suggests that the GNS successfully captures the dependence of the final runout with the initial aspect ratio, including the transition from the short to the tall column.

4.1.1. Short column

We now evaluate the GNS rollout (prediction) of the granular flow dynamics with time for a short column ($a = 0.8$). Fig. 7 shows the
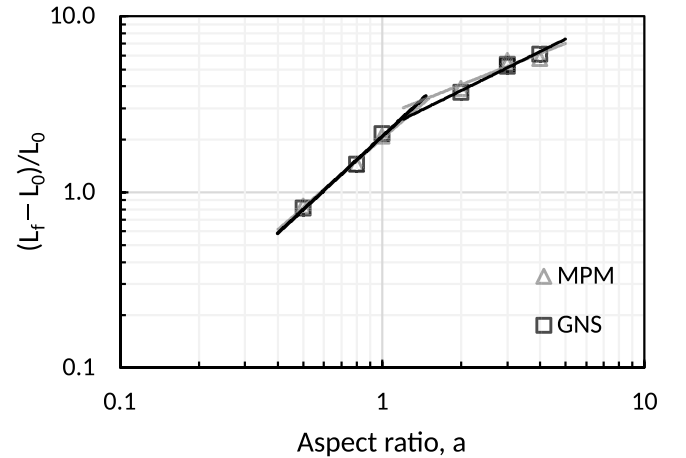
time evolution of granular flow for the short column collapse. We use a normalized time (t/τ_c) to compare the flow evolution, where t is physical time, and τ_c is the critical time defined as the time required for the flow to fully mobilize. τ_c is defined as $\sqrt{H_0/g}$, where g is the gravitational acceleration. In Fig. 7, the collapse shows three stages. First, the flow is mobilized by the failure of the flank and reaches full mobilization around $t/\tau_c = 1.0$. The majority of the runout occurs until $t/\tau_c = 2.5$. Beyond $t/\tau_c > 2.5$, the spreading decelerates due to the basal friction and finally stops at around $t/\tau_c = 4.0$ for both MPM and GNS rollout (prediction). As seen in Fig. 7, although the GNS has only seen an aspect ratio $a = 1.0$ during training, GNS successfully captures the overall time-evolution of granular flows for a short column ($a = 0.8$).

In addition to the visual comparison of profiles, we quantitatively investigate the flow dynamics of the GNS rollout of the short column by
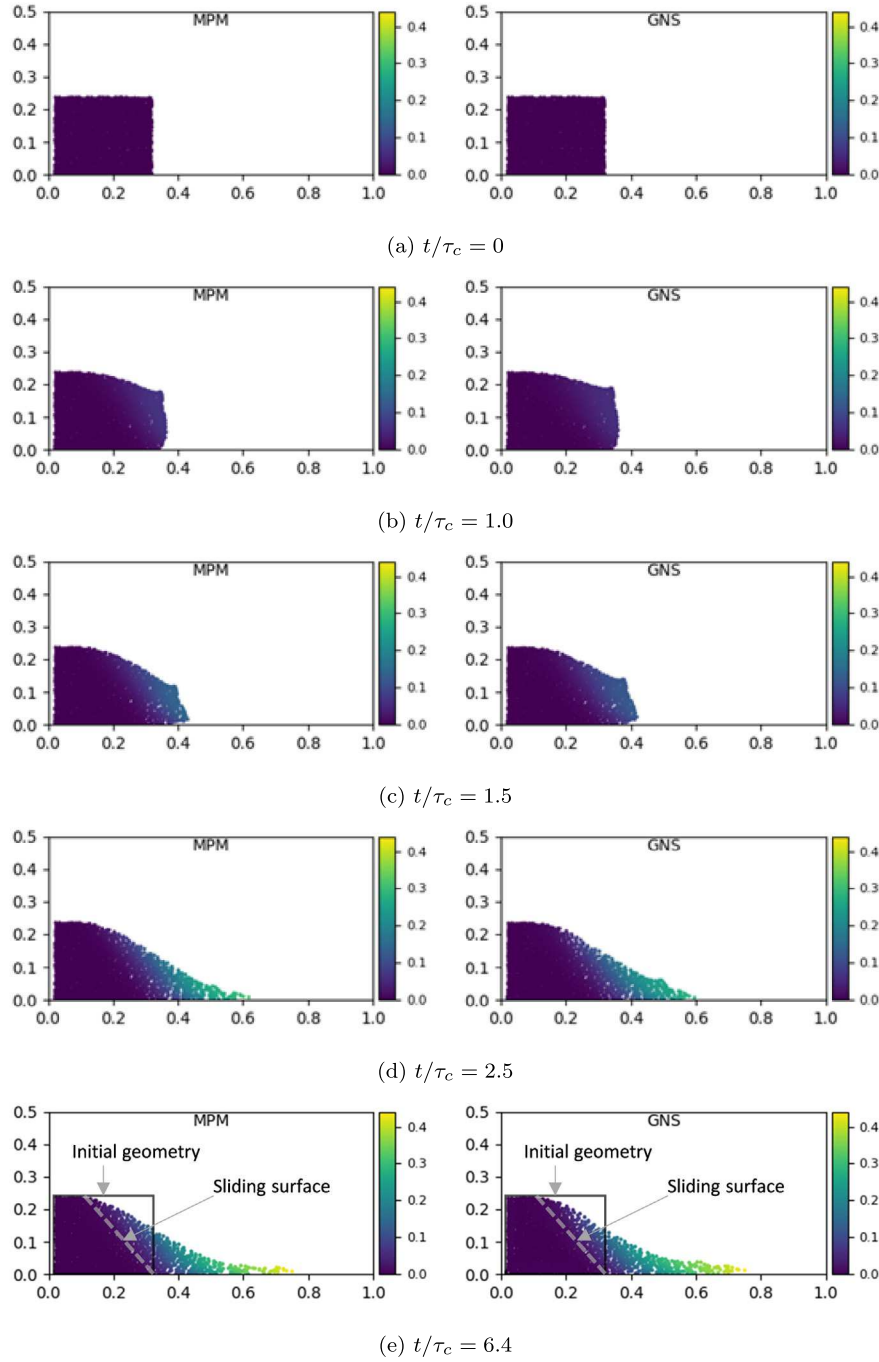
Fig. 7. Evolution of flow with normalized time for GNS and MPM for the short column with $a = 0.8$. Units are in *m*. The color represents the magnitude of the displacement. Subfigure (e) shows the final deposit at the last timestep.

comparing the normalized runout and height evolution with the MPM. Fig. 8a shows the evolution of normalized runout and height with time. The normalized runout of the MPM (see the gray line corresponding to the left axis in Fig. 8a) shows the three stages of collapse. The collapse of the granular column starts with the failure of the flank and evolves slowly until the runout is fully mobilized by $t/\tau_c = 1.0$. As the collapse proceeds, the runout acceleration increases ($t/\tau_c = 1.0$ to 2.5). After this time, the runout deaccelerates due to basal friction, and finally stops at $t/\tau_c \approx 4.0$. Both GNS and MPM show a constant normalized height (see the gray line corresponding to the right axis in Fig. 8a) as only the flank of the column collapse, leaving a static truncated-conical core. GNS predicts an almost identical evolution of runout as the MPM simulation, which is noteworthy as only a small portion of the training

trajectories (5 out of 26) includes the deacceleration behavior leading to the flow coming to rest due to the basal friction before hitting the walls. Overall, the quantitative comparison shown in Fig. 8a confirms that the GNS can accurately model the granular flow dynamics for the short column.

Fig. 8b shows the energy evolutions from GNS rollout and MPM simulation. Based on the principle of energy conservation, the granular flow must satisfy $E_0 = E_p + E_k + E_d$, where E_0 is the potential energy of the column before material points start to mobilize, E_p is the potential energy, E_k is the kinetic energy, and E_d is the dissipation energy due to friction along the boundary and material. In Fig. 8b, both GNS rollout and MPM simulation show identical energy evolutions. A significant fraction of the stored potential energy is converted to kinetic energy in
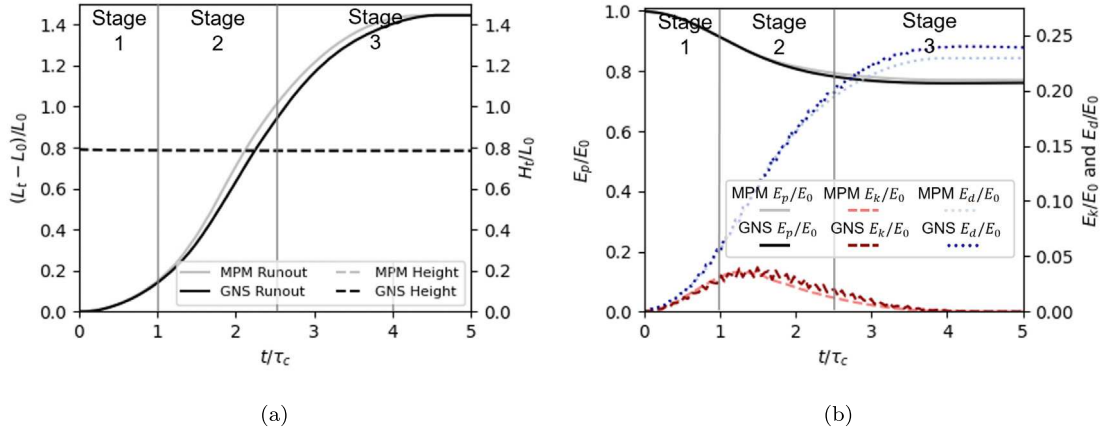
(a)   (b)

Fig. 8. (a) Normalized runout and height evolution with normalized time and (b) normalized energy evolution with normalized time for the short column $a = 0.8$. H_t is the height from the bottom corner of the boundary to the highest part of the column at t. $E_p = \sum_{i=1}^{n} m_i g h_i$ is the potential energy of the column, and $E_k = \frac{1}{2} \sum_i^n m_i v_i^2$ is the kinetic energy of the column, where m_i, h_i, and v_i is the mass, height, and velocity of the material point i, and n is the total number of material points. $E_d = E_0 - E_p - E_k$ is the dissipation energy where E_0 is the potential energy before material points start to move.

the initial stages of the failure, reaching a peak value of kinetic energy at $t/\tau_c = 1$. The kinetic energy dissipates due to the basal friction and flow ceases at $t/\tau_c = 4.0$ when E_k is fully dissipated.

4.1.2. Tall column

Tall columns exhibit different runout dynamics than the short column. GNS was only trained on granular mass with an aspect ratio of 1.0 and has not seen the dynamics of a tall column during training. As an example, we demonstrate the GNS prediction for a tall column with $a = 2.0$. Fig. 9 shows the GNS rollout and MPM simulation of the runout evolution for this case. GNS rollout predicts an identical runout profile with normalized time as the MPM simulation. Similar to the short column, the tall column also shows the three stages: the initial mobilization of the flow (t/τ_c to 1.0), runout ($t/\tau_c = 1.0$ to 2.5) along the failure surface, deacceleration ($t/\tau_c = 2.5$ to 4.0). In the tall column, however, a larger volume of sliding mass above the failure plane is mobilized. During the initial stages of the collapse, the granular mass experiences free fall due to gravity dominated by collisional dissipation. As the granular mass reaches the failure surface, the vertical kinetic energy is converted to horizontal acceleration, resulting in longer runout distances. GNS rollout shows similar behavior to the MPM runout simulation.

Fig. 10a shows the normalized runout and height evolution for the tall column. Although the runout evolution remains identical in the initial phase of the collapse, MPM shows a slightly larger normalized runout compared to the GNS. The final height in both GNS and MPM remains the same.

Fig. 10b presents the normalized energy evolution of the GNS rollout and the MPM simulation. During the initial stages of the collapse (t/τ_c to 1.0), a large amount of initial potential energy is converted to kinetic energy due to the free fall of mass under gravity. Both GNS and MPM show almost identical energy profiles. GNS shows a larger potential energy loss as the flow accelerates with an almost similar gain in kinetic energy. It indicates that GNS predicts larger frictional dissipation in tall columns, which could be from the training data focused only on short columns having higher frictional dissipation than tall columns. At the final stage, MPM shows less dissipation due to the basal boundary friction, resulting in a slightly longer runout than GNS rollout. Generally, energy dissipation behavior in GNS is consistent with MPM showing a more significant potential drop and increased dissipation energy accumulation.

Overall, the GNS rollout is consistent with the MPM simulation with a runout error of 5.7% for the tall column with $a = 2.0$, implying that the GNS can capture the dynamics of granular flows in collision-dominated tall columns despite only being trained on $a = 1.0$.

4.1.3. Upscaled domain

GNS is generalizable to different initial configurations of the flow simulation owing to the strong inductive bias of the GNN (Battaglia et al., 2018). The strengths of GNS surrogate models would be to train them on small-scale experiments and then predict large-scale dynamic scenarios with complex boundary conditions. We now evaluate the scalability of GNS to a larger domain, including more material points than the training dataset. Fig. 11 shows the GNS rollout of a short column $a = 0.8$ with 5120 material points (up to 5× more material points than the training dataset) for a larger simulation domain and longer rollout duration than the training dataset.

GNS successfully predicts the flow dynamics for an upscaled domain size showing a similar runout profile with the MPM simulation. The GNS rollout predicts a normalized runout of 1.74 while the MPM simulation shows 1.76, showing an error of 1.30%. Fig. 12 shows that GNS rollout successfully replicates energy evolution observed in an upscaled domain compared to the MPM simulation. Hence, GNS can reproduce the flow dynamics even for the upscaled geometries beyond the training dataset.

The primary source of GNS rollout error is not from the simulation scale but from the portion of material points that shows a large amount of displacement during column collapse. Fig. 13 shows the evolution of mean squared error (MSE) of displacement over all material points (N) with time t computed as $\frac{1}{n} \sum_i^N \left(p_i^t - p_{MPM_i}^t \right)^2$, where $p_{MPM_i}^t$ is material point position from MPM. When we compare the MSE for $a = 0.8$ with 1,824 of material points and its upscaled domain (2.22x material points), upscaling does not alter the MSE significantly. Fig. 14 shows the evolution of the squared error of displacement of individual material points for the upscaled domain ($a = 0.8$). The squared error shows larger values for those material points which run out further, i.e., the larger final displacements, but the proportion of the error with respect to the final runout is small so that GNS could simulate the upscaled case without significant error.

4.1.4. Different friction angles

To make GNS learn the different frictional behaviors, we provided the GNS with the granular flow trajectories with five different friction angles as described in Section 3.2. Fig. 15 is the summary of the prediction result. It shows the normalized runout predicted by the GNS compared to MPM on the aspect ratios and friction angles not shown during the training. As we have already observed in the single material case Fig. 6, the increasing trend of normalized runout with the aspect ratio is well captured through multiple friction angles as well as the relationship transitions between short columns and tall columns. Additionally, GNS generalizes well beyond the geometry (aspect ratio of 1.0)
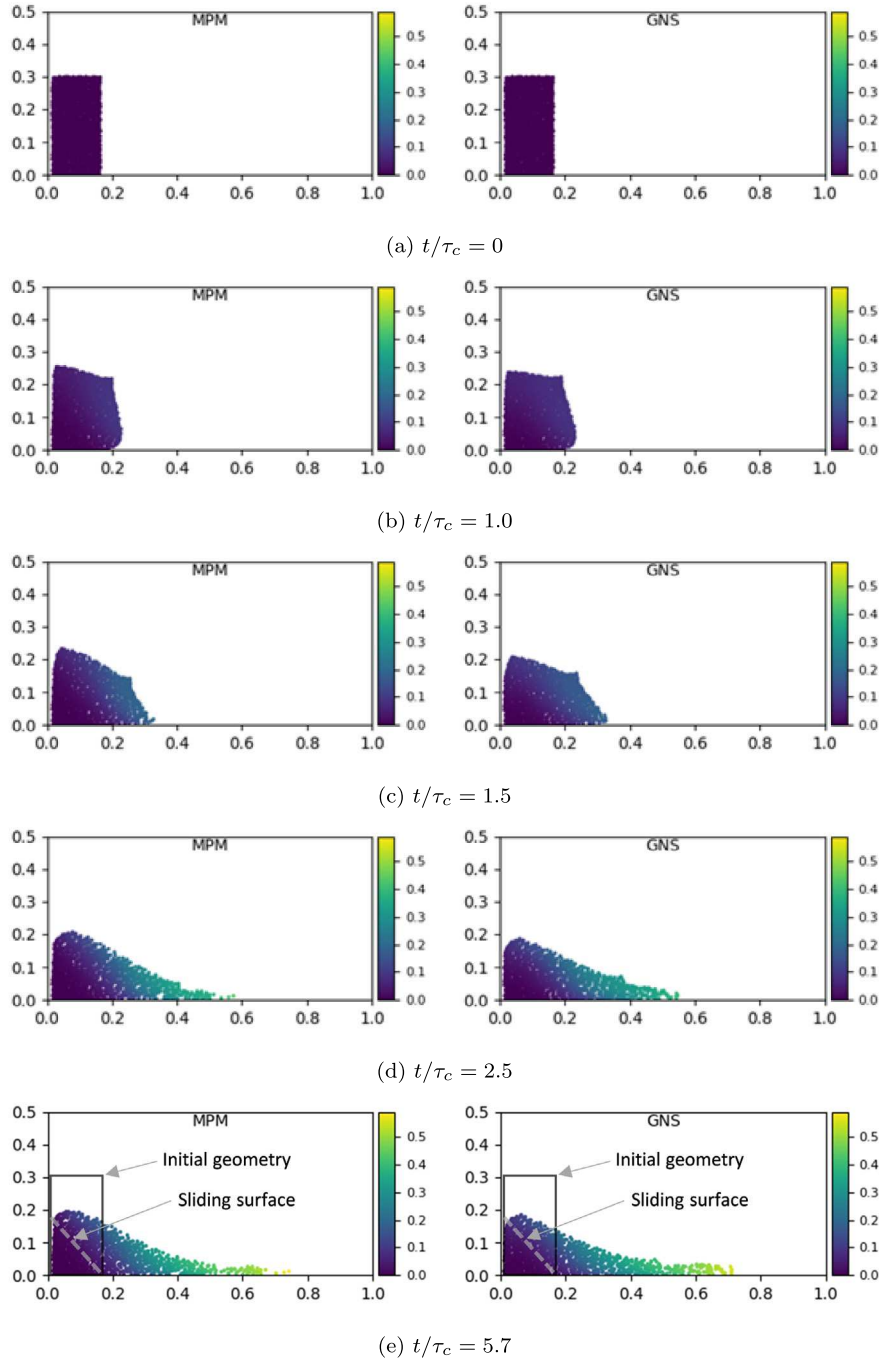
(a) $t/\tau_c = 0$

(b) $t/\tau_c = 1.0$

(c) $t/\tau_c = 1.5$

(d) $t/\tau_c = 2.5$

(e) $t/\tau_c = 5.7$

Fig. 9. Evolution of flow with normalized time for GNS and MPM for the tall column with $a = 2.0$. Units are in m. The color represents the magnitude of the displacement. Subfigure (e) shows the final deposit at the last timestep.

on which it was trained both for short and tall columns. These results suggest that the GNS successfully predicts the different granular flow dynamics depending on various friction angles although an exhaustive range of friction angles is not provided during the training.

4.2. Granular flow with barriers

To evaluate the generalizability of GNS for predicting interactions with barriers, we set the simulation configuration of the debris-resisting baffles to be outside the training distribution described in Section 3.2. Fig. 16a shows the initial configuration of the simulation. The simulation boundary is from 0.1 to 1.9 for the x-axis, 0.1 to 0.9 for the y-axis, and 0.1 to 1,9 for the x-axis. The granular mass of $0.35 \times 0.25 \times 1.4$ m

for each dimension approaches three baffles, whose geometry is $0.15 \times 0.30 \times 0.15$ m, with the initial x-velocity of 2.0 m/s. The center location of the two baffles in the first row is at $(x = 0.775, \ z = 0.375)$ m and $(x = 0.775, \ z = 1.625)$ m, and that of the final baffle is at $(x = 1.275, \ z = 1.0)$ m. Compared to training data, the simulation domain is four times larger with 2.2 times more material points. While the initial granular mass geometry takes on a cuboid shape, the training data exclusively features cubes. Additionally, we test the GNS with three baffles, in contrast to the training data which only contains one or two.

From the initial state to $t = 0.15$ s (Fig. 16), the granular debris propagates downstream uniformly and impacts the first baffle row. Upon hitting the first row, the side of the flow is obstructed by the baffles and the rest of the flow between the baffles proceeds towards the
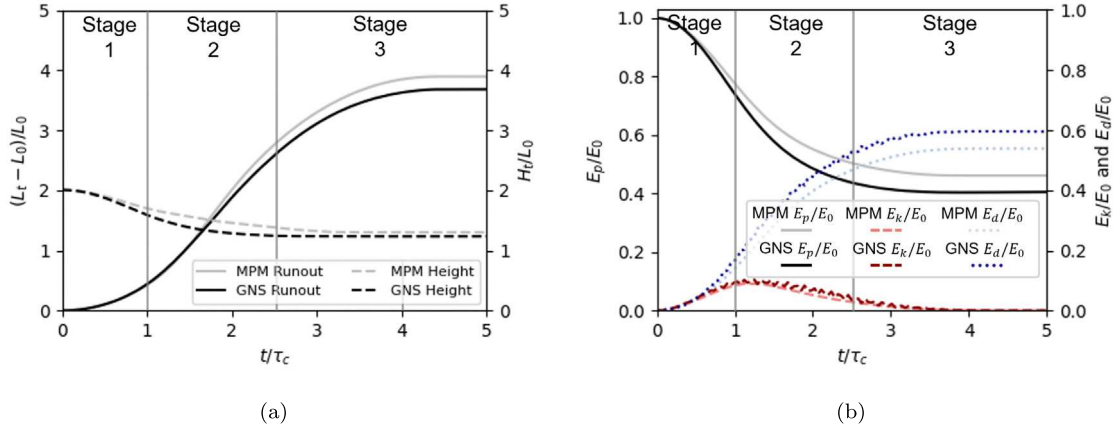
Fig. 10. (a) Normalized runout and height evolution with normalized time and (b) normalized energy evolution with normalized time for the tall column with $a = 2.0$.

Table 4
Computation time of MPM and GNS rollout per timestep for varying number of material points. Note the 2D model is the one used for simulating granular column collapse, and the 3D model is for the flow with barriers.

Model	# material points	MPM	GNS				
		Time (s/timestep)	# edges (Approx.)	Time-CPU (s/timestep)	Speed up	Time-GPU (s/timestep)	Speed up
2D	3.6K ("small")	37	0.153M	5.9	6.3	0.050	740
	40K ("intermediate")	227	1.8M	65.3	3.5	0.616	369
	90K ("large")	596	4.0M	148	4.0	1.515	393
	102K ("larger")	818	4.6M	170	4.8	Exceeds memory	
3D	6.9K ("small")	129	0.125M	4.59	28.1	0.045	2867
	250K ("intermediate")	2,767	4.8M	174.12	15.9	1.521	1819
	389K ("large")	4,149	7.2M	271.56	15.3	1.996	2079
	439K ("larger")	4,628	8.2M	306.92	15.1	Exceeds memory	

next baffle. Simultaneously, material is deposited and built up upstream of the baffles due to the obstruction and almost reaches the height of the baffle. From $t = 0.15$ s to $t = 0.34$ s, the flow impacts the next baffle and diverges into two granular jets through the open area with roughly symmetric shapes. A smaller amount of material deposition forms upstream of the last baffle compared to the first row. Beyond $t = 0.34$ s, the spreading of the grains decelerates due to the basal friction and finally reaches static equilibrium at around $t = 0.63$ s. GNS rollout successfully replicates the overall kinematics of the flow including the complex interaction with the obstacles.

To quantitatively compare the results from GNS and MPM, we measure the runout evolution (Fig. 17(a)), similar to the granular column collapse case. In this study, we define runout as the longest travel distance of the flow along the path compared to the initial forefront of the granular mass. With the initiation of the flow, runout rapidly propagates until $t = 0.34$ s experiencing the complex interaction between structures. Subsequently, the runout gradually decelerates and eventually halts as the rest of the flow energy dissipates. GNS perfectly predicts the runout evolution simulated by MPM although the simulation configuration includes more obstacles and a larger domain size compared to the training data.

We also measure the upstream depth evolution of the granular mass deposition right behind the baffles (Fig. 17(a)). The upstream flow depth refers to the depth of the granular flow measured upstream of the baffle array when the flow impacts the baffles. As runout proceeds, the flow faces the baffles at the first row and, subsequently, the second baffle. Upon impact with the first baffles, the upstream depth spikes

almost to the height of the baffles (0.3 m) due to the deposition of the grains and sudden backwater effects. After peaking, it shows a small amount of decrease over time as the granular flow around the baffles gradually reaches static equilibrium. As the flow impacts the baffle in the next row, the upstream depth surges, and it gradually decreases again but at a lower rate. The overall trend of the upstream depth change is well characterized by GNS, although GNS predicts a slightly less decrease in depth compared to the MPM. However, the difference is not significant.

Fig. 17(b) shows the energy evolution. Initially, the approaching granular flow has high kinetic energy due to the initial potential energy being mobilized. The material accumulation behind the baffles contributes to the steep drop in the kinetic energy, as the flow passes and interacts with the baffles. GNS captures these energy evolution trends accurately. The results from Figs. 16 and 17 suggest that the GNS successfully learns the granular flow dynamics interacting with obstacles.

5. Comparative analysis of computational efficiency

5.1. Rollout

Table 4 shows the computation times for the MPM, GNS with CPU, and GNS with GPU. We investigate four different cases of the number of material points ("small", "intermediate", "large" number of material points, and the "larger" case which slightly exceeds GPU memory
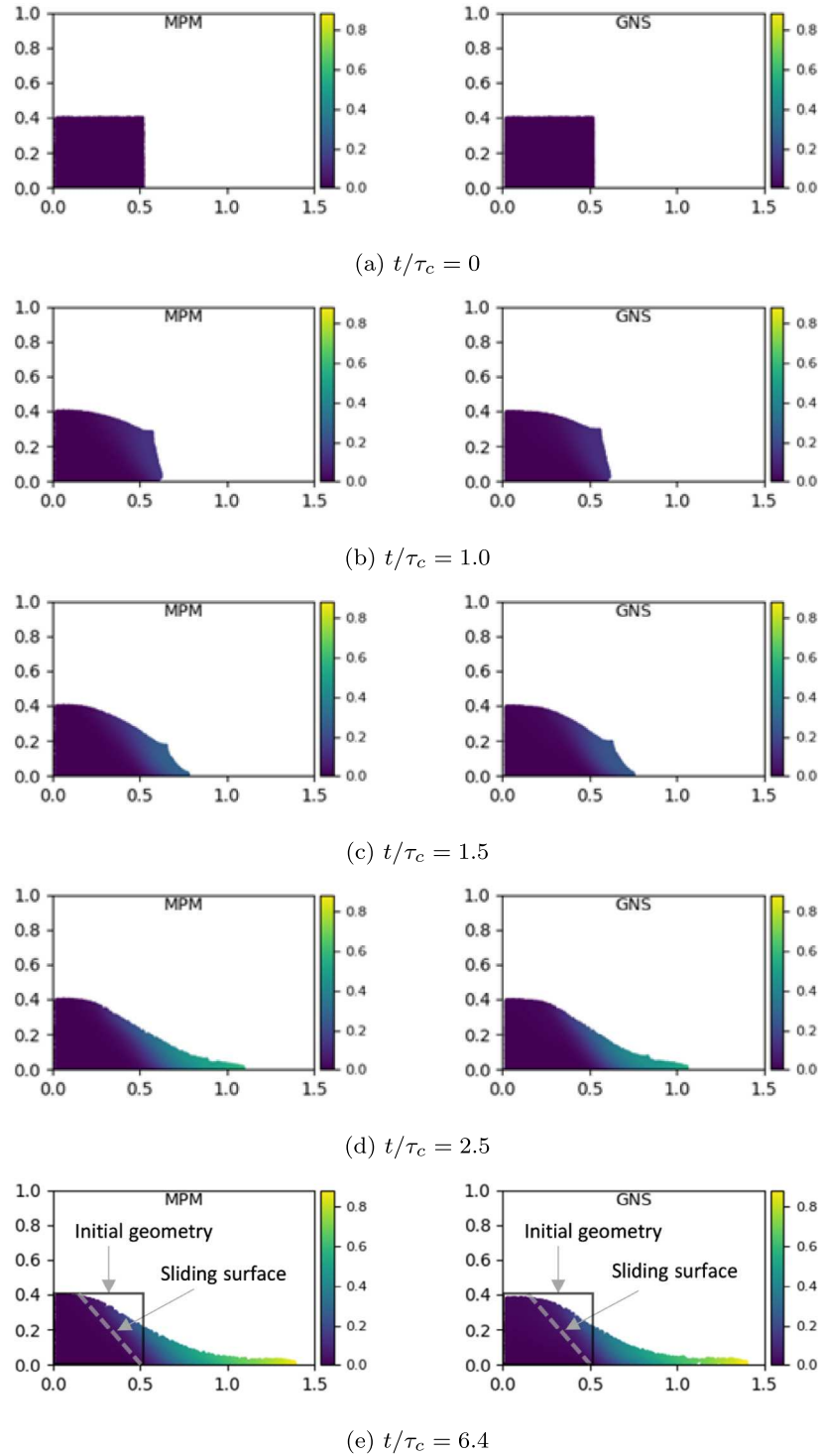
(a) $t/\tau_c = 0$

(b) $t/\tau_c = 1.0$

(c) $t/\tau_c = 1.5$

(d) $t/\tau_c = 2.5$

(e) $t/\tau_c = 6.4$

Fig. 11. Evolution of flow with normalized time for GNS and MPM for the upscaled case of short column with $a = 0.8$. Units are in m. The color represents the magnitude of the displacement. Subfigure (e) shows the final deposit at the last timestep.

capacity) for each GNS model (2D and 3D). The MPM simulations were run on 56 cores of Intel Cascade Lake processors with 128 GB memory on TACC Frontera using CB-Geo MPM code, while the GNS rollouts were computed on a single A100 GPU with 40 GB memory on TACC Lonestar 6.

As the number of material points increases, MPM computation time increases with $O(n \log n)$ at best and $O(n^2)$ at worst (Kumar et al., 2019). For GNS, as the number of material points increases, the edge connections linearly increase, which governs the message passing and

computation volume of GNS. Accordingly, it leads to the increase of GNS computation time with linear scaling ($O(n)$). When we compare the computation time of GNS with MPM in 2D model, GNS on CPU shows about 3.5 to 6.3×speed up, and GNS on GPU shows about 369 to 740×speed up. For the 3D model, GNS on GPU shows about 15 to 28×speed up, and GNS on GPU shows about 1819 to 2867×speed up. However, a limitation of the GPU-based GNS in its current form is its dependency on the available GPU memory. When the number of material points increases to 102 K for 2D and 439 K for 3D, the
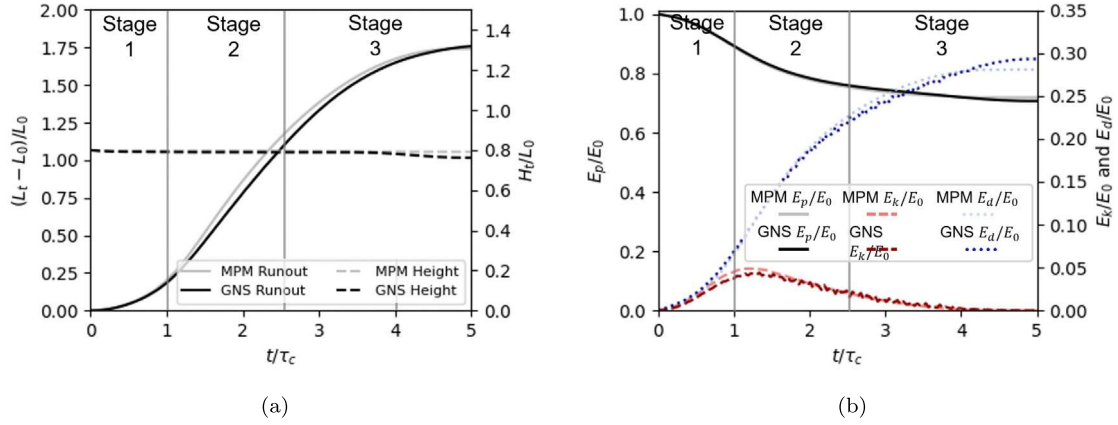
(a)   (b)

Fig. 12. Normalized runout and height evolution with normalized time and (b) normalized energy evolution with normalized time for the upscaled case of the short column with $a = 0.8$. Note that the data after $t/\tau_c > 5.0$ is abbreviated since the flow reaches a static state.
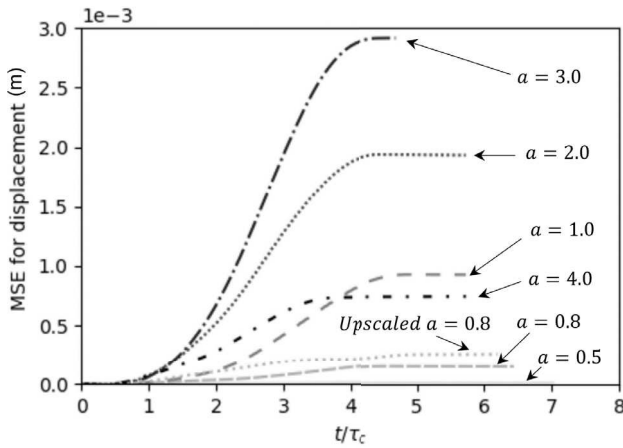

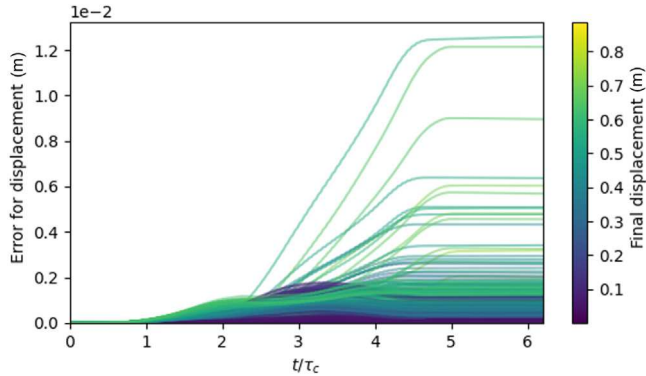**Fig. 13.** Evolution of mean squared displacement error over all material points with time.

Fig. 15. Normalized runout distance $((L_f - L_0)/L_0)$ with different aspect ratios (a) and friction angles (ϕ).

Fig. 14. Evolution of squared displacement error for each material point with normalized time in upscaled case of $a = 0.8$. The line color represents the final displacement of each material point.

this, GNS still provides a great deal of performance boost over GPU versions of numerical simulations, which are also restricted to smaller domains due to GPU memory bandwidths.

GNS can run on multiple GPUs, which increases computational efficiency and the number of simulation scenarios that can be run. Surrogate models, such as GNS, are computationally efficient compared to full-scale numerical simulations with MPM. Therefore, GNS allows for evaluating thousands of scenarios rather than just a handful of cases with full-scale numerical modeling.

5.2. Training

The training cost of GNS is significant, as illustrated by the metrics in 5 which include the number of parameters, edge connections, memory usage, and time taken for each training step. The 2D model, which has approximately 20 K edge connections, takes 0.26 s for each training step. In comparison, the more intricate 3D model, which has about 32 K edge connections, takes 0.38 s per step. Given the current hyperparameters, roughly 5 million training steps are required to achieve optimal rollout performance. This translates to an estimated training duration of 15 days for the 2D model and 22 days for the 3D model.

We consider using training examples with fewer edge connections to decrease training costs. However, this approach risks compromising our learning capacity. Reducing the connectivity radius decreases edge connections, limiting our model's ability to grasp vital local interactions.

model cannot proceed with the rollout due to memory constraints. Despite this, the CPU version of GNS can handle computations for a larger number of material points, surpassing the capability of its GPU counterpart. It is worth noting that the 3D model can simulate a higher number of material points compared to the 2D model because it has fewer edge connections within the connectivity radius.

Arduino et al. (2021) demonstrated that GPU MPM simulations speed up 5 to 10× compared to the CPU CB-Geo MPM version. Despite
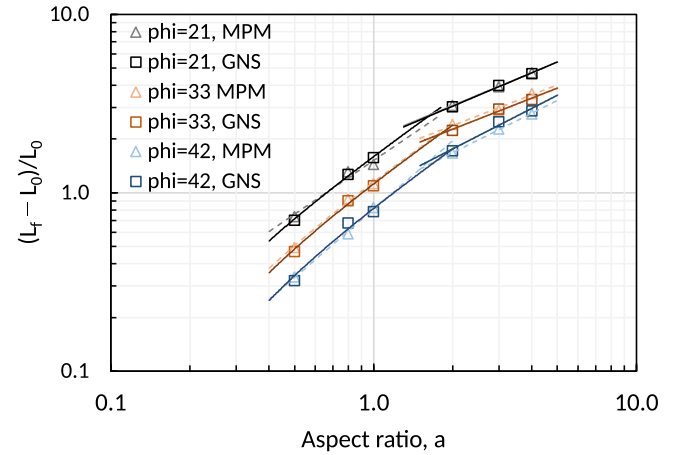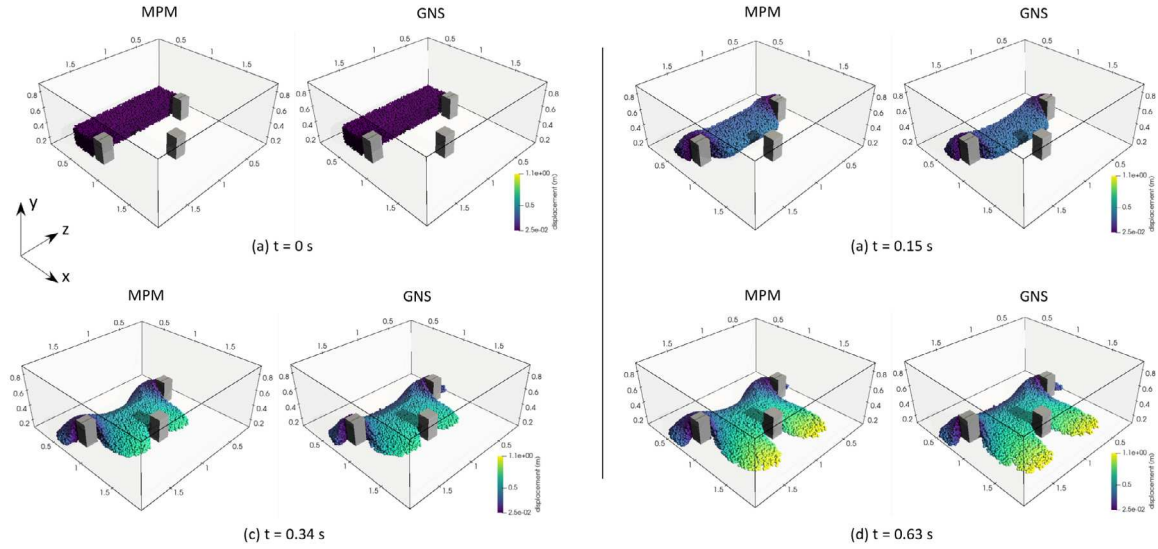
Fig. 16. Evolution of flow interacting with baffles for GNS and MPM from initial condition to the final deposit at the last timestep. The simulation domain is $1.8 \times 0.8 \times 1.8$ m and the initial geometry of the granular mass is $0.35 \times 0.25 \times 1.4$ m. The barrier size is $0.15 \times 0.30 \times 0.15$ m. The center location of the two baffles in the first low are at $(x = 0.775, \ z = 0.375)$ m and $(x = 0.775, \ z = 1.625)$ m, and the final baffle is at $(x = 1.275, \ z = 1.0)$ m.
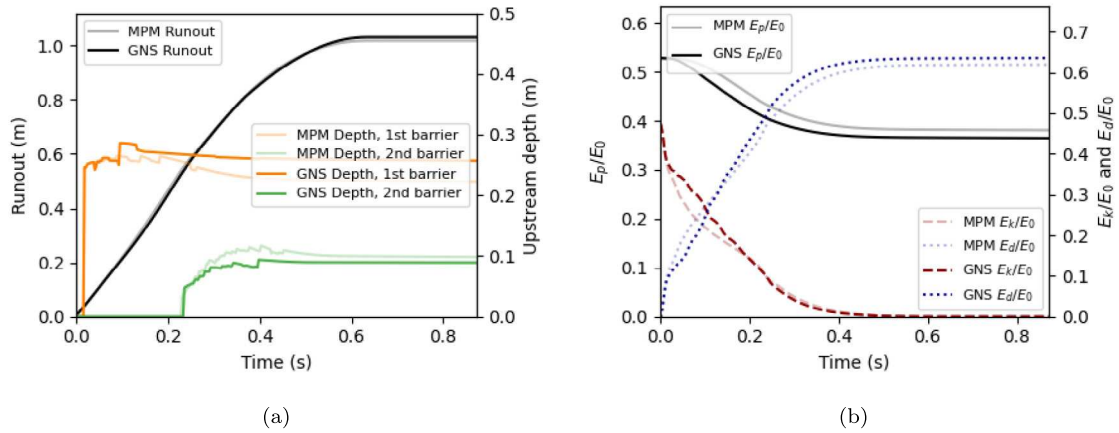

Fig. 17. (a) Runout and upstream depth evolution and (b) normalized energy evolution with time.

Table 5
Training cost of GNS.

Model	# parameters	# edges (approx.)	GPU memory usage (approx.) (GB)	Training speed (s/step)
2D	1 591 826	20K	20	0.26
3D	1 592 979	32K	30	0.38

Alternatively, when we reduce the number of vertices in the training example, we restrict the range of interactions our model captures. The challenge lies in finding a balance between training efficiency and learning depth. We must ensure the connectivity and number of vertices remain high enough to capture complex interactions yet not so high as to be computationally inefficient. There is a need to balance training costs with optimal learning outcomes.

6. Limitations of GNS

The GNS surrogate model demonstrates accuracy, generalizability, and improved computation efficiency. However, the GNS suffers from the following limitations.

As discussed earlier in Table 4, the GPU memory limits the current implementation of the GNS surrogate model. A GPU with 40 GB memory can simulate up to around 439 K material points (approximately 8.2M edge connections). However, this shortcoming can be improved by optimizing the size of the connectivity radius R. For example, in Table 4, the 3D model is able to simulate 4× more material points with 2× more edge connections than the 2D model, since R of the 3D model uses a smaller R ($= 0.025$) and includes less number of material points inside it, while the 2D model uses a larger R ($= 0.030$) and includes more number of material points than the 3D model. Nevertheless, the 3D model still shows an accurate rollout as shown in Section 4.2. This implies that, although a larger R can consider more interaction between neighbors, excessively large R is not necessary and can aggravate the computation efficiency of GNS. Multi-GPU GNS rollouts will enable the scalability of GNS to larger and more complex domains.

Another limitation is that GNS shows error accumulation with time, see Fig. 13. GNS uses the explicit Euler time integration to predict the next state based on the current state ($X_t \rightarrow X_{t+1}$) as described in Section 2.2.5. GNS uses acceleration to predict the runout of the next time step. However, long duration rollouts result in error accumulation. To overcome this issue, where the GNS predictions could
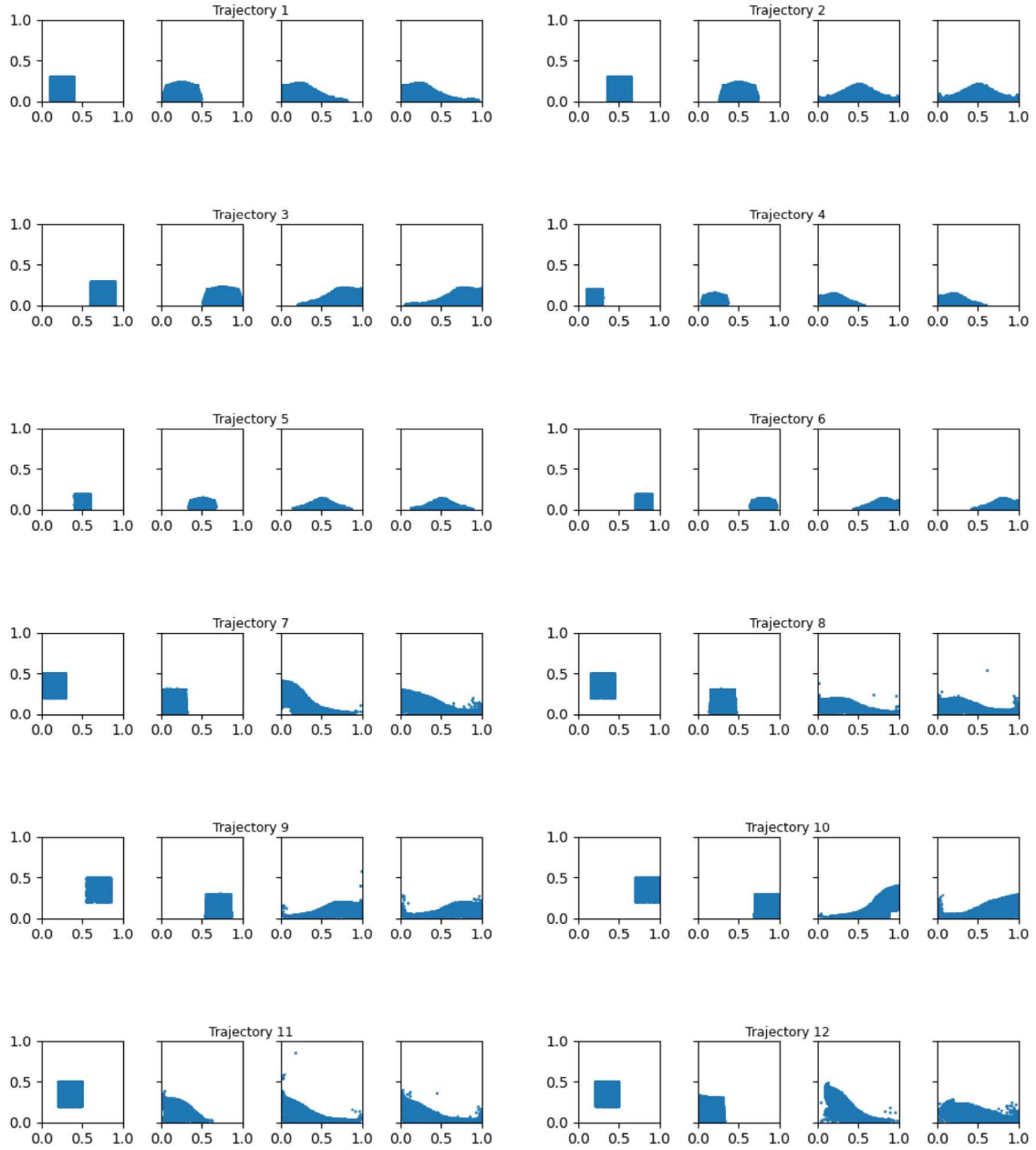
Fig. A.1. MPM granular trajectories of training data generated using CB-Geo MPM.

deviate from the reality, we introduce Gaussian noise at each step of the training data to minimize error accumulation. Although including noise effectively reduces error accumulation (Sanchez-Gonzalez et al., 2020), GNS cannot be free from the error accumulation as timestep increases. The use of purely data-driven architecture, i.e., MLP, for the encoder, processor, and decoder can be a potential source of error as GNS learns to predict acceleration. Substituting MLPs in the encoder and decoder with non-data-driven architectures such as simple linear layers or principal component analysis might help reduce the error of GNS using less training data. Sanchez-Gonzalez et al. (2020) investigated the validity of the former method, but confirmed no significant changes in performance. Imposing known physics-based constraints to the message passing in the processor (Seo and Liu, 2019) could also help address potential errors stemming from the purely data-driven architecture. Yang et al. (2022) introduced momentum conservation law to the message passing, and observed the error reduction.

The calibration of the GNS model is pivotal to ensure the accuracy and reliability of simulations by aligning them with real-world data.

Calibration often involves adjusting the model parameters, and in the case of GNS, modifying material properties m or connectivity radius are direct methods. For instance, changing the connectivity radius can affect simulation outcomes, such as granular flow deformation. However, merely tweaking parameters might not suffice for optimal accuracy. An alternative approach is transfer learning, where a GNS model pre-trained on foundation data can be further refined based on experimental observations, enhancing its predictive accuracy for specific scenarios. Continued research into GNS-specific calibration is crucial for its application in real-world engineering analyses.

7. Conclusion

Traditional numerical methods are computationally intensive when simulating large-scale granular flows. Statistical or conventional machine learning-based surrogate models are not generalizable since they do not explicitly consider the underlying physics. We implemented a graph neural network (GNN)-based simulator (GNS) as a multi-GPU
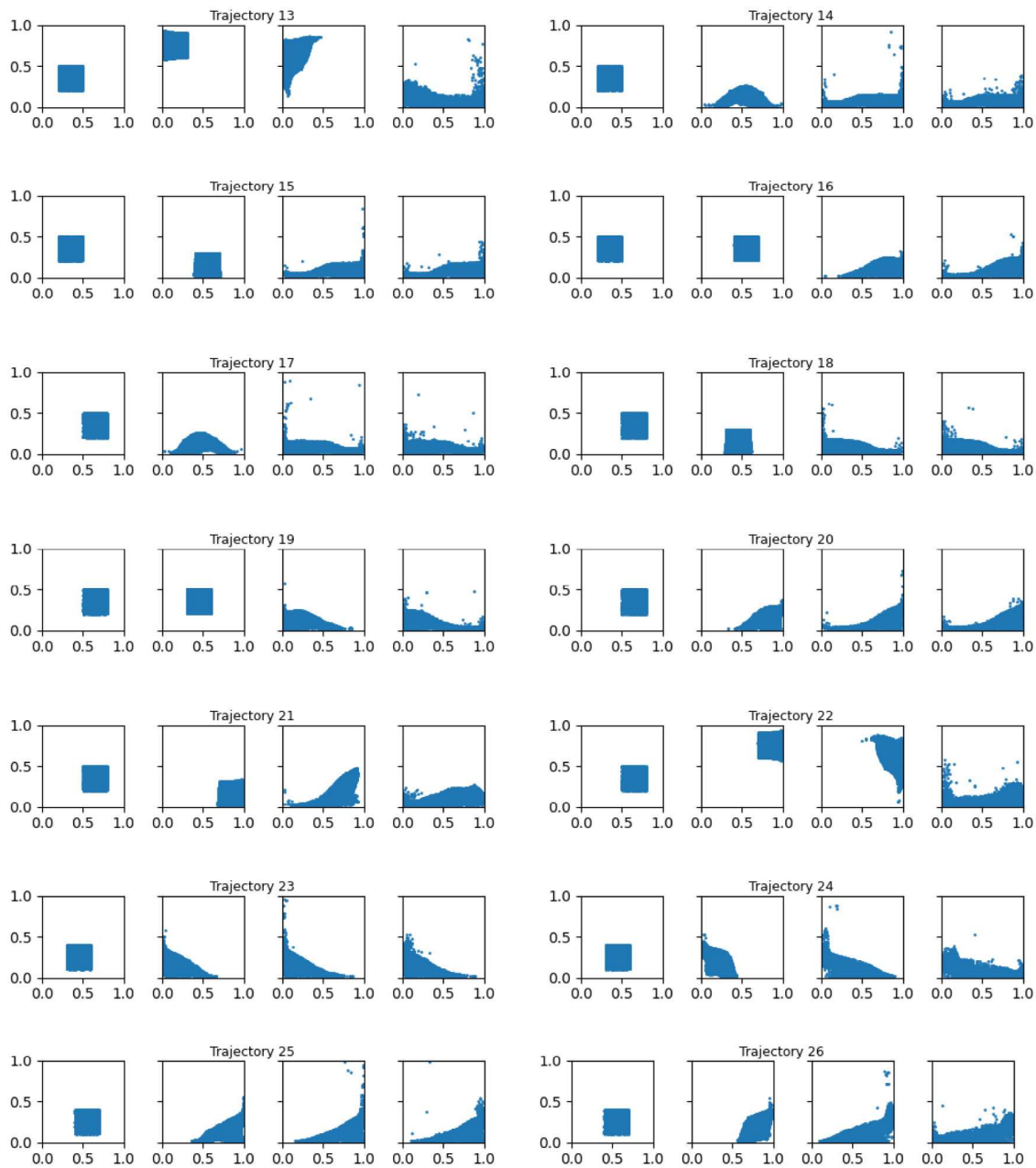
Fig. A.1. (*continued*).

parallel version for generalizable granular flow surrogate simulation. The use of graphs efficiently represents the physical state of interacting material points. At the same time, the message passing operation of GNN encourages the neural network to learn the interaction between material points. The expressive power of graphs and message passing that models the interaction between material points allows GNS to accurately predict granular flow dynamics for various conditions, including those not seen during training. We demonstrate the performance of GNS on granular column collapse and granular flow interaction with barriers. GNS precisely simulates different flow dynamics involved in columns for different initial aspect ratios and can also be applied to the upscaled domain with more than 2× material points with a longer simulation duration than the data provided for training. GNS can predict the granular flow interacting with barriers that include different granular mass and barrier configurations not seen during training. GNS also shows a remarkable speed-up of up to 2900× computation speed compared to the parallelized CPU version of MPM.

The computational efficiency and generalizability of the GNS surrogate can expedite evaluating runout hazards requiring numerous scenarios.

CRediT authorship contribution statement

Yongjin Choi: Data curation, Formal analysis, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Krishna Kumar:** Conceptualization, Funding acquisition, Resources, Software, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

The GNS code (Kumar and Vantassel, 2022) is available under the MIT license on GitHub (https://github.com/geoelements/gns). The training dataset and trained models are published under CC-By license on DesignSafe Data Depot (Choi and Kumar, 2023).

Acknowledgments

Appendix. Training data

Fig. A.1 illustrates the training data used for the Granular Neural Simulation (GNS) model. These images depict 26 trajectories of granular collapses generated through the CB-Geo MPM (Material Point Method) code. Each trajectory shows the dynamic behavior of granular materials used in training the GNS.

References

Arduino, P., Bonus, J., Motley, M.R., Eberhard, M.O., 2021. Tsunami-driven debris effects on structures using a multi-GPU MPM tool. Mecánica Comput. 38 (1), 3.

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

Battaglia, P.W., Pascanu, R., Lai, M., Rezende, D.J., Kavukcuoglu, K., 2016. Interaction networks for learning about objects, relations and physics. CoRR abs/1612.00222.

Choi, Y., Kumar, K., 2023. Training, validation, testing data, and trained model. http://dx.doi.org/10.17603/DS2-4NQZ-S548.

Choi, C., Ng, C.W.W., Law, R.P., Song, D., Kwan, J., Ho, K., 2015. Computational investigation of baffle configuration on impedance of channelized debris flow. Can. Geotech. J. 52 (2), 182–197.

Choi, C.E., Ng, C.W.W., Song, D., Kwan, J., Shiu, H., Ho, K.K.S., Koo, R.C., 2014. Flume investigation of landslide debris–resisting baffles. Can. Geotech. J. 51 (5), 540–553.

Durante, M.G., Rathje, E.M., 2021. An exploration of the use of machine learning to predict lateral spreading. Earthq. Spectra 37 (4), 2288–2314. http://dx.doi.org/10.1177/87552930211004613.

Gao, L., Zhang, L.M., Chen, H.X., Fei, K., Hong, Y., 2021. Topography and geology effects on travel distances of natural terrain landslides: Evidence from a large multi-temporal landslide inventory in Hong Kong. Eng. Geol. 292, http://dx.doi.org/10.1016/j.enggeo.2021.106266.

Haeri, A., Skonieczny, K., 2022. Three-dimensional granular flow continuum modeling via material point method with hyperelastic nonlocal granular fluidity. Comput. Methods Appl. Mech. Engrg. 394, 114904. http://dx.doi.org/10.1016/j.cma.2022.114904.

Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., Durand, F., 2019. Taichi: a language for high-performance computation on spatially sparse data structures. ACM Trans. Graph. 38 (6), 201.

Ju, L.Y., Xiao, T., He, J., Wang, H.J., Zhang, L.M., 2022. Predicting landslide runout paths using terrain matching-targeted machine learning. Eng. Geol. 311, http://dx.doi.org/10.1016/j.enggeo.2022.106902.

Kermani, E., Qiu, T., Li, T., 2015. Simulation of collapse of granular columns using the discrete element method. Int. J. Geomech. 15 (6), 04015004. http://dx.doi.org/10.1061/(asce)gm.1943-5622.0000467.

Kingma, D.P., Ba, J., 2014. Adam: a method for stochastic optimization. CoRR abs/1412.6980.

Kumar, K., 2015. Multi-Scale Multiphase Modelling of Granular Flows (Ph.D. thesis). University of Cambridge, http://dx.doi.org/10.5281/zenodo.160339.

Kumar, K., Choi, Y., 2023. Granular column collapse with graph neural network-based simulator. http://dx.doi.org/10.17603/DS2-GVVW-GT60.

Kumar, K., Delenne, J.Y., Soga, K., 2017a. Mechanics of granular column collapse in fluid at varying slope angles. J. Hydrodyn. 29 (4), 529–541. http://dx.doi.org/10.1016/s1001-6058(16)60766-7.

Kumar, K., Salmond, J., Kularathna, S., Wilkes, C., Tjung, E., Biscontin, G., Soga, K., 2019. Scalable and modular material point method for large-scale simulations. arXiv:1909.13380.

Kumar, K., Soga, K., Delenne, J.Y., Radjai, F., 2017b. Modelling transient dynamics of granular slopes: MPM and DEM. Procedia Eng. 175, 94–101. http://dx.doi.org/10.1016/j.proeng.2017.01.032.

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

Lajeunesse, E., Mangeney-Castelnau, A., Vilotte, J.P., 2004. Spreading of a granular mass on a horizontal plane. Phys. Fluids 16 (7), 2371–2381. http://dx.doi.org/10.1063/1.1736611.

Lube, G., Huppert, H.E., Sparks, R.S.J., Freundt, A., 2005. Collapses of two-dimensional granular columns. Phys. Rev. E 72 (4), http://dx.doi.org/10.1103/physreve.72.041301.

Mast, C.M., Arduino, P., Mackenzie-Helnwein, P., Miller, G.R., 2014. Simulating granular column collapse using the Material Point Method. Acta Geotech. 10 (1), 101–116. http://dx.doi.org/10.1007/s11440-014-0309-0.

Nguyen, N.H.T., Bui, H.H., Nguyen, G.D., 2020. Effects of material properties on the mobility of granular flow. Granul. Matter 22 (3), http://dx.doi.org/10.1007/s10035-020-01024-y.

Rubanova, Y., Sanchez-Gonzalez, A., Pfaff, T., Battaglia, P., 2021. Constraint-based graph network simulator. arXiv preprint arXiv:2112.09161.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.W., 2020. Learning to simulate complex physics with graph networks. CoRR abs/2002.09405, URL: https://arxiv.org/abs/2002.09405, arXiv:2002.09405.

Seo, S., Liu, Y., 2019. Differentiable physics-informed graph networks. arXiv preprint arXiv:1902.02950.

Soga, K., Alonso, E., Yerro, A., Kumar, K., Bandara, S., 2016. Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. Géotechnique 66 (3), 248–273. http://dx.doi.org/10.1680/jgeot.15.LM.005.

Staron, L., Hinch, E.J., 2005. Study of the collapse of granular columns using two-dimensional discrete-grain simulation. J. Fluid Mech. 545, 1–27. http://dx.doi.org/10.1017/S0022112005006415.

Sun, X., Zeng, P., Li, T., Wang, S., Jimenez, R., Feng, X., Xu, Q., 2021. From probabilistic back analyses to probabilistic run-out predictions of landslides: A case study of Heifangtai terrace, Gansu Province, China. Eng. Geol. 280, 105950. http://dx.doi.org/10.1016/j.enggeo.2020.105950.

Utili, S., Zhao, T., Houlsby, G.T., 2015. 3D DEM investigation of granular column collapse: Evaluation of debris motion and its destructive power. Eng. Geol. 186, 3–16. http://dx.doi.org/10.1016/j.enggeo.2014.08.018.

Yang, C., Gao, W., Wu, D., Wang, C., 2022. Learning to simulate unseen physical systems with graph neural networks. arXiv preprint arXiv:2201.11976.

Yang, Q., Hambleton, J.P., 2021. Data-driven modeling of granular column collapse. In: Geo-Extreme 2021. pp. 79–88. http://dx.doi.org/10.1061/9780784483701.008.

Yang, S., Wang, X., Liu, Q., Pan, M., 2020. Numerical simulation of fast granular flow facing obstacles on steep terrains. J. Fluids Struct. 99, 103162.

Zeng, P., Sun, X., Xu, Q., Li, T., Zhang, T., 2021. 3D probabilistic landslide run-out hazard evaluation for quantitative risk assessment purposes. Eng. Geol. 293, 106303. http://dx.doi.org/10.1016/j.enggeo.2021.106303.

Zhao, T., Lei, J., Xu, L., 2022. An efficient Bayesian method for estimating runout distance of region-specific landslides using sparse data. Georisk: Assess. Manag. Risk Eng. Syst. Geohazards 16 (1), 140–153. http://dx.doi.org/10.1080/17499518.2021.1952613.