2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Optimal Control of Granular Material

Yuichiro Aoyama[1], Amin Haeri[2] and Evangelos A. Theodorou[1]

***Abstract*— The control of granular materials, which are found in many industrial applications, is a challenging open research problem. Granular material systems are complex-behavior (as they could have solid-, fluid-, and gas-like behaviors) and high-dimensional (as they could have many grains/particles with at least 3 DOF in 3D) systems. Recently, a machine learning-based Graph Neural Network (GNN) simulator has been proposed to learn the underlying dynamics. In this paper, we perform optimal control of a rigid body-driven granular material system whose dynamics is learned by a GNN model trained by reduced data generated via a physics-based simulator and Principal Component Analysis (PCA). We use Differential Dynamic Programming (DDP) to obtain optimal control commands that can form granular particles into a target shape. The model and results are shown to be relatively fast and accurate. The control commands are also applied to the ground truth model, i.e., physics-based simulator, to further validate the approach.**

I. INTRODUCTION

Control of granular materials has many applications, such as mining machinery, industrial robots, e.g., excavators and robots that work with humans [1], [2], [3]. There are various aspects to consider to solve such a complex problem. One difficulty here is in obtaining the granular material model. Another difficulty is that granular materials are high-dimensional systems, as they could include many grains/particles with at least 3 DOF (in 3D). To achieve the system model, machine learning methods have recently been shown to be efficient alternatives to traditional numerical methods. They require a huge amount of training data that should be generated by either verified computer simulations or experiments (which are usually not practically feasible). Moreover, the control of such systems by actively interacting with the particles has not been addressed in the literature yet. We will discuss it in more detail below.

Simulation Method. Physics-based numerical granular flow simulation methods can be considered highly accurate candidates as system models [4]. But they predominantly suffer from their intensive computational complexity, especially in hardware-constrained robotics applications [5]. However, Machine Learning (ML) has begun to be applied to the problem of reducing the runtime of simulating complex physics [6]. Data-driven ML models are an emerging class of physics simulation methods. They are trained by either computer-generated synthetic or real experimental data. They could be accurate enough depending on the data utilized and the model architecture.

Learning Model. Here, the goal is to properly learn the underlying dynamics principles in the physical systems automatically. Several recent methods have been used for fluid simulations, including regression forest, multilayer perceptron (MLP), autoencoder (AE) convolutional neural network (CNN), generative adversarial network (GAN), and loss function-based method [7], [8], [9], [10], [11], [12]. They can handle mostly a specific state of materials (either gas-, solid-, or specifically fluid-like behaviors). However, graph neural networks (GNN) [13] are generalized models proven to be capable of simulating different states of materials. Recent state-of-the-art research has developed a GNN model [14], which has outperformed some other recent methods [15], [16] in terms of accuracy and ease of implementation. However, aside from the time complexity, GNNs might be infeasible for desktop GPUs with mid-level limits on memory (i.e. VRAM of 8-20GB).

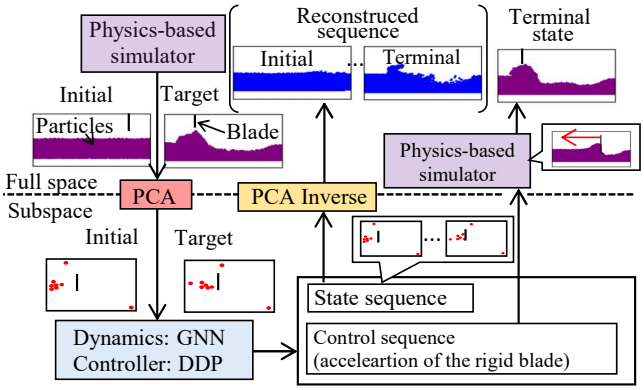


Fig. 1: Overview of our framework and task.

Reduced-Order Model. In physics simulations, model reduction methods are utilized to capture the effective degrees of freedom of a physical system [17], [18], [19], [20]. This is to reduce both processing time and the usage of memory space corresponding to the system. Recent literature [21], [22] has shown that principal component analysis (PCA) [23] could be an effective choice to capture primary modes of object deformation while maintaining the expressive power of the ML model. PCA can significantly reduce the training and rollout runtime and memory usage of ML models by learning the reduced dynamics (i.e. subspace learning [24]).

In [21], to run the ML-based simulation in real-time, a subspace GNN model has been trained by a reduced number of particles obtained through PCA. The subspace output of the GNN model can then be projected back to the full space by the PCA inverse. This approach has been able

[1]Georgia Institute of Technology, US. yaoyama3@gatech.edu
[2]University of Chicago, US. haeri@uchicago.edu

to accurately predict the interactions of rigid bodies and granular flows (e.g. the interactions of an excavator bucket or a rover wheel with soil particles). Note that the training data have been generated using an accurate and efficient continuum physics-based simulation method called Material Point Method (MPM) with a nonlocal viscoplastic granular flow constitutive model (NGF) [25], [26]. This method can also predict the interaction forces applied to the rigid body by the granular material.

Particle Control. There are a few granular material dynamics models that have been proposed for control purposes in the literature. In [27], using a particle dynamics model introduced in [14], a control problem is solved, where the particles are shaped to be a target shape. The task is to find a suitable place to pour particles (instead of actively interacting with them, as done in this paper). Also, in [15], some examples of control problems are presented, including the formation of fluids and deformable material into target shapes. However, the particles are assumed to not have real mechanical properties, which limits their potential real-world applications.

Control of particle systems can be seen as a trajectory optimization problem, where state and control sequences are optimized to reach a desired state, i.e., a target shape. For trajectory optimization, Sequential Quadratic Programming (SQP) [28] and Differential Dynamic Programming (DDP) [29] have been widely used in robotic applications. The system state can be represented by a concatenation of the states of the particles and actuators (e.g., bucket and robotic arm). Thus, the control state dimension is significantly smaller than that of the system state dimension. For this type of system, DDP is computationally noticeably faster than SQP. This is because in DDP, the size of Hessian is determined by the control state dimension (whereas the size of Hessian in SQP is determined by the sum of the system and control state dimensions) [30], [31]. Also, DDP can provide feedback gain as a by-product of its optimization, whereas SQP needs to be performed in MPC fashion to enjoy feedback control.

Contribution. In this paper, we aim to actively control the shape of a granular material that interacts with a rigid body and form its shape into a target shape. We perform an optimal control using the DDP algorithm with the PCA-reduced dynamics learned by the GNN model [21]. Our method is configuration and material type agnostic as it is a generalized method by design. That is, it can handle any configuration with any type of material [14], [21], [25]. Here due to limited space, we only discuss a system of one rigid blade interacting with granular materials (i.e., soil). The control inputs are the temporal accelerations of the rigid blade. First, we generate a training dataset using a physics-based simulator [25] including several examples each with thousands of particles and different motions of rigid bodies. Second, the granular flow system is reduced via PCA (i.e., from the order of thousands of particles to a very small number of modes). We then train a GNN model, obtaining system dynamics in subspace. By applying DDP to the reduced dynamics, control input commands that drive the rigid body are obtained. The control commands can cause the particles to form a target shape. Finally, we apply the control commands to the physics-based simulator to further validate our approach. An overview of our model is shown in Fig. 1. Although a framework that can form elastoplastic material to a target shape has been proposed [32], to the best of our knowledge, such a framework has not been proposed for the granular flow control problem.

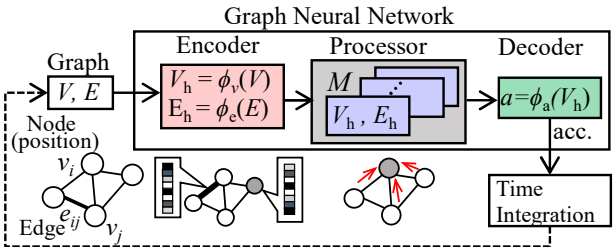


Fig. 2: Graph Neural Network (GNN) simulator.

II. Preliminaries

In this section, we will provide a background on the main components used in our framework.

A. Graph Neural Network (GNN)

The Graph Neural Network (GNN) simulator utilizes a graph representation of particles possibly with different types (e.g., rigid, granular, etc.). [14]. Let us define a graph $G = (V, E)$, where $v_i \in V$ and $e_{i,j} \in E$ are the node and edge attributes, respectively. The subscripts i, j in $e_{i,j}$ indicate the sender and receiver node indices. In the graph representation of particles, nodes v_i correspond to the particles, and the edges $e_{i,j}$ correspond to the relations between the particles. More precisely, if two particles are closer than a threshold, they can be connected with edges with specific attributes. Note that we do not aim to use such relations in subspace because the relations might not be proximity-based. In our work, we keep the graph for the reduced particles fully connected. Of course, fully connected graphs make their computational process heavy, but thanks to the reduced number of particles (nodes) by PCA, this strategy is still feasible in our setting.

The procedure performed on the GNN simulator is shown in Fig. 2. GNN takes graphs and outputs the particles' accelerations. The node and edge attributes are given by

$$v_i = [q_k^i, \dot{q}_{k-C+1}^i, \cdots, \dot{q}_k^i, s^i, g_k],$$
$$e_{i,j} = \left[q_k^i - q_k^j, \left\|q_k^i - q_k^j\right\|\right],$$

where q_k^i is the position of i'th particle at time step k, s^i is particle type (e.g., sand, rigid body), and g_k is the global feature (shared among all nodes). Note that the C sequence of velocity can be obtained by $C+1$ sequence of positions using finite difference schemes. GNN first encodes an input graph into a latent graph via encoder multilayer perceptrons (MLPs) as $V_h = \phi_v(V)$, $E_h = \phi_e(E)$. The processor then performs M times of message passing steps to propagate

information and update the node and edge attributes. The detailed equations of the processor component can be found in [21]. Finally, the particle accelerations are predicted via a decoder MLP as $\ddot{q}_k^i = \phi_{\mathrm{a}}(v_i)$. The accelerations are used to update node positions q_k using implicit Euler integration with time step Δt via

$$\dot{q}_{k+1} = \dot{q}_k + \ddot{q}_k \Delta t, \quad q_{k+1} = q_k + \dot{q}_{k+1} \Delta t \tag{1}$$

where we drop i for particles. q_{k+1} is used as the node positions of the new input graph to forward the simulation. Note that similar to the physics-based simulator, the motion of rigid particles which represent the rigid body can be predefined. Therefore, the GNN model can be learned to predict the motion of normal granular particles only. We train GNN to predict standardized acceleration and unstandardized it to predict positions as in [14].

B. Principle Component Analysis (PCA)

Principal Component Analysis (PCA) is a nonparametric linear dimensionality reduction method [33]. It provides a data-driven, hierarchical coordinate system to re-express high-dimensional correlated data. The resulting geometry of the coordinate system is determined by the principal components (also known as modes). These modes are uncorrelated (orthogonal) to each other, but they have a maximal correlation with the observations.

Here, we explain how PCA is used to obtain the reduced data. Define n_{n} as the number of normal particles in each example and m_{o} as the number of observations. Note that we are not using rigid particles for PCA. Given that the system is d dimensional with a time horizon N, and the number of training examples is n_{e}, the number of observations is given by $m_{\mathrm{o}} = n_{\mathrm{e}} \times N$. In 2D with $d = 2$, the data matrix X_{d} of the particle trajectories (i.e. positions) is written as

$$X_{\mathrm{d}} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^{n_{\mathrm{n}}} \\ y_1^1 & y_2^2 & \cdots & y_1^{n_{\mathrm{n}}} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m_{\mathrm{o}}}^1 & x_{m_{\mathrm{o}}}^2 & \cdots & x_{m_{\mathrm{o}}}^{n_{\mathrm{n}}} \\ y_{m_{\mathrm{o}}}^1 & y_{m_{\mathrm{o}}}^2 & \cdots & y_{m_{\mathrm{o}}}^{n_{\mathrm{n}}} \end{bmatrix} \in \mathbb{R}^{dm_{\mathrm{o}} \times n_{\mathrm{n}}}, \tag{2}$$

where superscripts and subscripts of x and y are indices for the particle and observation number, respectively. Notice that the column of X_{d} represents a flattened and concatenated trajectory of a particle, while the row represents the position of all the particles at the same time step. n_{nr} principle components of centralized (i.e. zero-mean) data are given as eigenvectors of the covariance matrix of data, which corresponds to the first n_{nr}'th largest eigenvalues, forming the loading matrix W. W projects X_{d} onto a subspace, forming Z. They are given by

$$W = [w_1, \cdots w_{n_{\mathrm{pr}}}], \quad Z = [X_{\mathrm{d}} - \mathbf{1}\mu]W. \tag{3}$$

where μ is the mean of the data and $W \in \mathbb{R}^{n_{\mathrm{n}} \times n_{\mathrm{nr}}}, W \in \mathbb{R}^{dm_{\mathrm{o}} \times n_{\mathrm{nr}}}$. Notice that the number of particles is now reduced from n_{n} to n_{nr}. Z can be mapped back to full space, resulting in the reconstruction of X_{d} denoted by $X_{\mathrm{d}}^\dagger$

$$X_{\mathrm{d}}^\dagger = ZW^{\mathsf{T}} + \mathbf{1}\mu. \tag{4}$$

When $n_{\mathrm{n}} = n_{\mathrm{nr}}$ and $W^{\mathsf{T}} = W^{-1}$, X_d is perfectly reconstructed.

C. Differential Dynamics Programming (DDP)

Consider the discrete-time optimal control problem of a dynamical system given by

$$\begin{aligned} \min_U \ J(X,U) &= \min_U \big[\sum_{k=0}^{N-1} l(x_k, u_k) + \Phi(x_N)\big], \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k), \end{aligned} \tag{5}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ denote the state and control inputs of the system at time step t_k, respectively. $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the transition dynamics function. The scalar-valued functions $l(\cdot,\cdot)$, $\Phi(\cdot)$, and $J(\cdot)$ denote the running, terminal, and total cost of the problem, respectively. We also let $X = [x_0, \ldots, x_N]$, $U = [u_0, \ldots, u_{N-1}]$ be the state and control trajectory over the time horizon N. The cost-to-go at time step $k = i$, i.e., cost starting from $k = i$ to N is given by $J_i(X_i, U_i)$ with $X_i = [x_i, \ldots, x_N]$, $U_i = [u_i, \ldots, u_{N-1}]$. The value function is defined as the minimum cost-to-go in each state and time step k via $V_k(x_k) := \min_{u_k} J_k(X_k, U_k)$. Given Bellman's principle of optimality that provides the following rule

$$V_k(x_k) = \min_{u_k} [\underbrace{l(x_k, u_k) + V_{k+1}(x_{k+1})}_{Q_k(x_k, u_k)}], \tag{6}$$

DDP finds local solutions to (5) by expanding both sides of the rule (i.e., (6)) about given nominal trajectories, $\bar{X}$ and $\bar{U}$. Taking quadratic expansions of Q_k about $\bar{x}_k$ and $\bar{u}_k$ considering the deviation $\delta x_k = x_k - \bar{x}_k$, $\delta u_k = u_k - \bar{u}_k$, and mapping terms on both sides of Q with linearized dynamics, i.e.,

$$\delta x_{k+1} = f_x \delta x_k + f_u \delta u_k,$$

result in the Q derivatives (evaluated on $\bar{x}$ and $\bar{u}$) as follows

$$\begin{aligned} Q_{yy,k} &= l_{yy} + {f_y}^{\mathsf{T}} V_{xx,k+1} f_y \\ Q_{y,k} &= l_y + {f_y}^{\mathsf{T}} V_{x,k+1}, \end{aligned} \tag{7}$$

where we augment control and state as $y_k = [u_k^{\mathsf{T}}, x_k^{\mathsf{T}}]$ to save space. With quadratically expanded Q, we can explicitly optimize with respect to δu_k and compute the locally optimal control update via

$$\begin{aligned} \delta u_k^* &= \kappa + K \delta x_k, \\ \text{with} \quad \kappa &:= -Q_{uu}^{-1} Q_u, \ \ K = -Q_{uu}^{-1} Q_{ux}, \end{aligned} \tag{8}$$

where κ and K are known as feedforward and feedback gains, respectively. Note that we dropped the time indices for Q for readability. Now, δu_k^* is computed using $V_{x,k+1}$ and $V_{xx,k+1}$. Notice the difference in the time instances of u_k and $V_{x,k+1}$. To propagate V_k back in time, we plug the

quadratic expansion of V_{k+1} with linearized dynamics and δu_k^* into (6), and comparing coefficients of δx_k, obtaining

$$\begin{aligned} V_{x,k} &= Q_{x,k} - Q_{xu,k}Q_{uu,k}^{-1}Q_{u,k}, \\ V_{xx,k} &= Q_{xx,k} - Q_{xu,k}Q_{uu,k}^{-1}Q_{ux,k}, \end{aligned} \tag{9}$$

with boundary condition $V_N = \Phi(x_N)$. The process of computing V_x and V_{xx} is called backward pass. After the completion of the backward pass, a new state-control sequence is determined by propagating the dynamics forward in time (called forward pass). DDP repeats these two passes, decreasing the cost. Although the original DDP uses the second-order expansion of dynamics, we only use the first-order information due to its computational efficiency and numerical stability [34].

III. Learned Dynamics and Optimal Control

The dynamics of our system in reduced space, represented by a GNN model, is explained here. We have applied our method to two different configurations including a moving rigid box containing granular materials inside and a moving rigid blade interacting with granular materials (i.e., soil). However, here due to limited space, we only consider the latter. The state-space representation of this system is derived to set up a problem to achieve a task where the materials are formed into a target shape.

A. System Dynamics

Let p_k be the positions of normal reduced particles at time step k. Recall that we have normal particles and rigid particles that represent the granular material and rigid bodies, as mentioned in II-A. Although only normal particles are projected onto subspace via PCA, we can also reduce the number of rigid particles carried in the state of the system. Let b_k be a representative position of the rigid body (blade) selected as the center. With knowledge of the initial position of the rigid particles, their positional information at time step k can be recovered from b_k. Therefore, we keep only one representative point of the blade in the state of the system. Now, we can derive the dynamics of the blade system. Concatenating the sequence of the particle positions and rigid representative positions, we define the state as

$$\begin{aligned} X_{k:k+C} = [p_k^\mathsf{T}, \cdots, p_{k+C}^\mathsf{T}, b_k^\mathsf{T}, \cdots, b_{k+C}^\mathsf{T}]^\mathsf{T} \\ \in \mathbb{R}^{d(n_\mathrm{n}+1)(C+1)+d(C+1)}, \end{aligned} \tag{10}$$

where C is the length of the sequence of velocity introduced in II-A because GNN needs a sequence of velocities.

The system's control input $u_{k+C} \in \mathbb{R}^m$ is the acceleration of the blade in the vertical direction. For the horizontal direction, we set the speed constant to simplify the learning and control problem. This simplification can still describe some real-world applications, such as motor graders [35]. Here, the $C+1$ sequence of position is required to recover the C velocity sequence. The GNN model takes $X_{k:k+C}$ as input and predicts normal particle accelerations $\ddot{p}_{k+C+1}$ as detailed in II-A. Let $g_{\mathrm{GNN}}(\cdot)$ be a function of the trained GNN simulator. We obtain the accelerations of the most recent element of the state sequence by feeding the sequence of positions as

$$\ddot{p}_{k+C} = g_{\mathrm{GNN}}(X_{k:k+C}; [p]_0^\mathrm{r}), \tag{11}$$

where $[p]_0^\mathrm{r}$ is the initial position of rigid particles. Using this acceleration, we update the velocity and position with a semi-implicit Euler scheme given in (1). We put these acceleration predictions and position updates together, writing

$$p_{k+C+1} = G_{\mathrm{GNN}}(X_{k:k+C}; [p]_0^\mathrm{r}),$$

As mentioned, the blade acceleration is the control input. Hence, he representative point's velocity and position are updated through

$$\begin{aligned} \dot{b}_{k+C+1} &= \dot{b}_{k+C} + \tilde{u}_{k+C}\Delta t, \\ b_{k+C+1} &= b_{k+C} + \dot{b}_{k+C+1}\Delta t \end{aligned} \tag{12}$$

where $\tilde{u}_{k+C}$ distributes u_{k+C} to the corresponding dimensions. In our example $b_k \in \mathbb{R}^2$, while $u_k \in \mathbb{R}$ and thus cannot be added as they are. Hence, we have $\tilde{u}_k = [0, u_k]^\mathsf{T}$. Also, because we set the horizontal velocity constant, $\dot{b}_k^1 = v_\mathrm{c}$, where the superscript indicates the first element and v_c is a constant. Since elements of $X_{k+1:k+C+1}$ except for p_{k+C+1} and b_{k+C+1} are obtained by shifting $X_{k:k+C}$, we have dynamics for the rigid body example as

$$X_{k+1:k+C+1} = f(X_{k:k+C}, u_{k+C}; [p]_0^\mathrm{r}), \tag{13}$$

whose detailed expression is given by

$$\begin{bmatrix} p_{k+1} \\ p_{k+2} \\ \vdots \\ p_{k+C-1} \\ p_{k+C} \\ b_{k+1} \\ \vdots \\ b_{k+C} \\ b_{k+C+1} \end{bmatrix} = \begin{bmatrix} M_1 \begin{bmatrix} p_k \\ p_{k+1} \\ \vdots \\ p_{k+C} \end{bmatrix} \\ G_{\mathrm{GNN}}(X_{k+C}; [p_0]^\mathrm{r}) \\ M_2 \begin{bmatrix} b_k \\ b_{k+1} \\ \vdots \\ b_{k+C} \end{bmatrix} \\ 2b_{k+C} - b_{k+C-1} + \tilde{u}_{k+C}(\Delta t)^2 \end{bmatrix},$$

$$\text{where} \quad \begin{aligned} M_1 &= [O_{dn_\mathrm{p}C, dn_\mathrm{p}}, I_{dn_\mathrm{p}C}] \in \mathbb{R}^{dn_\mathrm{p}C \times dn_\mathrm{p}(C+1)}, \\ M_2 &= [O_{dC,d}, I_{dC}] \in \mathbb{R}^{dC \times d(C+1)}. \end{aligned}$$

For this system, the dimension $d = 2$, the number of the reduced particles $n_\mathrm{r} = 8$, and $C = 5$, thus the state dimension is $dn_\mathrm{r}(C+1) + (C+1)d = 108$, where we can see the benefit of not having all the rigid particles. If we need them, they add (the number of rigid particles,10) $\times(C+1) = 60$ particles to the state. Notice that the dynamics takes a linear form except for the GNN part and that its Jacobian matrix for DDP is computed efficiently by leveraging this structure without solely relying on numerical computation.

B. Constrained Optimal Control.

Now, we can set up an optimal control problem (i.e. (5)) for our system. The dynamics is given by (13). We set quadratic running and terminal costs as follows.

$$l(X_k, u_k) = 0.5(u_k^\mathsf{T} R u_k + Y_k^\mathsf{T} P Y_k), \qquad (14)$$
$$\Phi(Y_N) = 0.5 Y_N^\mathsf{T} Q Y_N,$$

with desired state X_g, difference with the state $Y_k = X_k - X_\mathrm{g}$ and positive semi-definite weight matrices P, Q, and R. Note that X_g is given by reduced particles. Since we now have everything required for DDP, we can compute the optimal control sequence for our task, i.e., the sequence of accelerations of rigid bodies to form the particles into a target shape. Since our final validation is performed by moving rigid bodies in the physics-based simulation, which has spatial bounds, it is necessary to constrain the motion of the rigid bodies in DDP. Also, constraining the motion of the rigid body close to the space shown in the training phase helps the model inference. Because the bounds are described as inequality constraints, we add a log barrier function to the objective [36], [37]. Let us define the constraints $g(x_k, u_k) = g(y_k) \in \mathbb{R}^w$. The constraints are satisfied when $g(\cdot) < 0$. With a positive penalty parameter ϵ, the log barrier function is given by $-\epsilon \log(-g)$, modifying the Q functions of DDP as

$$\hat{Q}_y = Q_y - \epsilon \left[\sum_{i=1}^{w} \frac{g_{i,y}(y)}{g_i(y)} \right],$$
$$\hat{Q}_{yy} = Q_{yy} - \epsilon \sum_{i=1}^{w} \left[\frac{g_{i,yy}(y)}{g_i(y)} \right] + \epsilon \sum_{i=1}^{w} \left[\frac{g_{i,y}(y) g_{i,y}(y)^\mathsf{T}}{g_i(y)^2} \right].$$

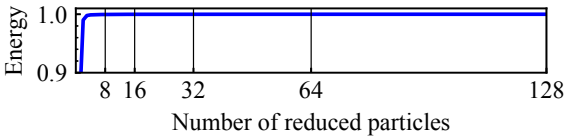
By using these modified Q, the constrained problem is solved by DDP. We omit the second term of Q_{yy} for better conditioning of Q_{uu}, which corresponds to the Gauss-Newton approximation [38]. In the original formulation of the log barrier method, ϵ is driven to zero as optimization proceeds. In practice, however, a small fixed ϵ, say 10^{-3}, works well [39], and thus we keep ϵ constant.

IV. Numerical Experiments

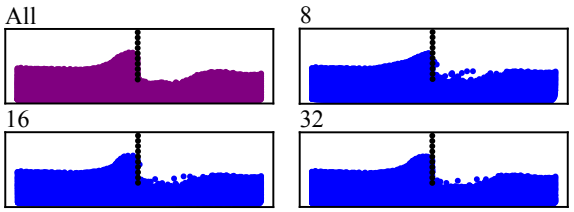
In this section, we provide the results of our numerical experiments, as well as the procedures used to obtain the results, including data generation, PCA, and model training. We perform DDP on the trained GNN model in the subspace. We can then recover the full space particles by applying PCA inverse (see (4)) to see the results approximately. To validate our system's dynamics and control commands, we compare the target and terminal states in full space obtained by applying the control commands from DDP to the physics-based simulator. That is, we move the blade in the physics-based simulation, following the trajectory of the blade from DDP.

A. Data generation

In order to generate training data, we parameterize a blade trajectory, simulating 320 trajectories using the physics-based simulator with 3150 particles [25]. The data acquisition time, which is the time discretization interval Δt in DDP, is 1/60 s. Note that this is different from that of the physics-based simulator which is 5×10^{-5}.



(a) Total energy (cumulative) versus the number of reduced particles.



(b) Example of reconstruction. The numbers correspond to those of the reduced particles that are used to reconstruct the full particles. The blade moves from right to left.

Fig. 3: Total energy (a) and reconstruction from different numbers of reduced particles. Particles in full space are drawn in purple and reconstructed particles are drawn in blue.

B. PCA for reduced particles

After generating the training data, PCA is applied to reduce the dimensionality of the data as explained in II-B. Fig. 3a shows the relationship between the number of reduced particles (PCA modes) and the cumulative total energy (variance). The total energy of the first i reduced particles is given by $\sum_{k=1}^{i} \lambda_k / \sum_{k=1}^{n_\mathrm{n}} \lambda_k$, where λ_i is the i'th largest eigenvalue of the data covariance matrix. The total energy plot indicates at least how many reduced particles are required to capture the full space information. As seen in Fig. 3a, 8 reduced particles can cover more than 99% of the energy. Note that an optimal number of reduced particles (i.e., modes) heavily depends on the complexity of the configuration as it can also be applied to 3D configurations [21]. Fig. 3b shows examples of reconstructed particles from reduced particles. Although some information is lost, eight particles can still describe the entire motion of the full system. Thus, we use eight reduced particles (PCA modes) in our application. We compute and store the matrix W (and mean μ) for reduction and reconstruction.

C. Model training

The GNN model is trained using the reduced data. The inputs and outputs are the reduced particle positions and their accelerations, respectively. When computing the particle accelerations, we use finite difference as

$$\dot{p}_k \Delta t = p_k - p_{k-1}, \quad \ddot{p}_k \Delta t = \dot{p}_k - \dot{p}_{k-1},$$

leading to

$$\ddot{p}_k (\Delta t)^2 = p_k - 2p_{k-1} + p_{k-2}.$$

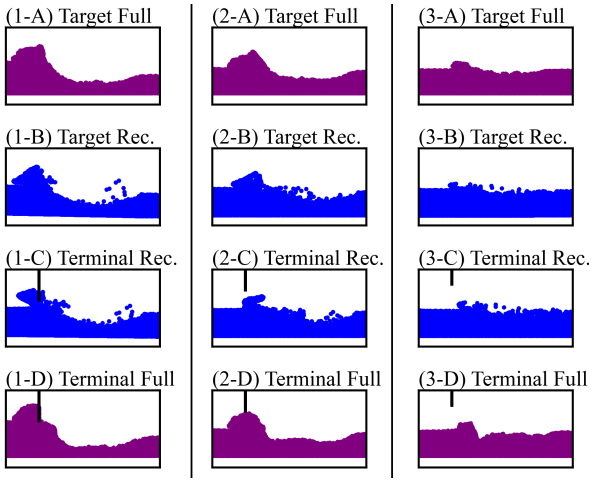


Fig. 4: Results and validation plots. Full and reconstructed particles are drawn in purple and blue, respectively. (A)'s are target shapes in full space, (B)'s are PCA reconstructed target shapes in full space, (C)'s are reconstructed terminal states from the GNN model with DDP control commands, and (D)'s are terminal states from the physics-based simulator with the same control commands. Blades are drawn with black.

We use $\ddot{p_k}(\Delta t)^2$ as labels and thus the output of GNN.

D. Results

The results are shown in Fig. 4. Here, we present three experiments with different targets numbered 1 to 3. In the figure, (A) is the target shape in full space, (B) is the reconstructed target shape in full space, (C) is the PCA reconstructed terminal state from the GNN model with the DDP control commands, and (D) is the terminal states from the physics-based simulator with the control commands. Since DDP is performed in the reduced space, comparing the target and terminal states in the reduced space is more meaningful than comparing (B) and (C), but we have these here to see how the reduced-order model can recover the full state. Overall, the controller seems to know in which direction to move the blade to achieve the target shape. The Root Means Square Errors (RMSEs) between the target and terminal state in the full space are given in Table I. Although all particles are distinguished and a quadratic cost in (14) is used in our work, permutation-invariant cost, such as the Earth Mover's Distance (EMD) [40] is more appropriate for the task because the goal is to form the distribution of particles to a target one, rather than assigning each particle to its goal position. We therefore add EMD in the table as a reference, which is given by

$$d_{\mathrm{EMD}}(S,T) = \min_{\phi:S\to T} \sum_{x\in S} \|x - \phi(x)\|_2,$$

where S, T are source and target sets with the same number of elements, i.e., particles, and $\phi : S \to T$ is a bijection. We used [41] to solve the minimization problem to obtain ϕ.

TABLE I: RMSE and EMD between target (A) and terminal states(D) in Fig. 4. The numbers correspond to those in the figure. The initial state is a flat shape shown in Fig. 1.

Cost	1	2	3
RMSE Init.	8.65×10^{-2}	5.92×10^{-2}	2.67×10^{-2}
RMSE Terminal	1.26×10^{-2}	1.39×10^{-2}	1.55×10^{-2}
EMD Init.	190	106	48.6
EMD Terminal	22.4	12.2	9.77

We deduce that a large portion of the error between the target and the terminal state in the reduced space is due to the DDP method and the reduced-order model. DDP is a local method that approximates the value function around the nominal trajectory. Thus, it converges to a local minimum and may get stuck at it. Adjusting the weight matrices and penalty parameter ϵ in the penalty term might partially resolve this issue [39]. The property, including controllability and observability, of the reduced-order model by PCA for linear systems is well studied [42]. For the nonlinear system, however, the same analysis is not applicable, and it might have been difficult for our system in the reduced space to be steered in a desired direction even though there exists a way to steer the original system.

We also speculate that the number of reduced particles could be another cause of the overall error. Consequently, we tried to use more modes in the hope of potentially increasing the accuracy of the model. However, it turned out that a higher number of reduced particles makes the optimization problem more challenging with underactuation. With 16 reduced particles, DDP gets stuck at poor local minima in the early stage of optimization. It will be further examined in future work. Furthermore, we tried a deep autoencoder as a model reduction method, because of its superior reconstruction accuracy compared to PCA. However, the obtained reduced-space representation does not preserve the structure of particles for GNN, and thus we did not proceed with it.

V. Conclusion

We performed an optimal control using DDP with the PCA-reduced dynamics learned by the GNN model to form granular materials into target shapes. First, we generated the data via a physics-based simulator. Then, we reduced the data dimensionality via PCA and trained a GNN model accordingly. We derived the state-space representation of learned reduced dynamics and implemented DDP coupled with to compute control command that forms particles into a target shape. The control command was applied to the physics-based simulator, showing the validity of our approach.

One future direction is to utilize cost metrics that are appropriate for forming particles, such as Chamfer and Earth Mover's distances. While the utilized physics-based simulation method, as a data generator, uses an accurate nonlocal viscoplastic constitutive model, we will additionally consider testing the algorithm on real experiments, by developing a framework that can recognize particles from images.

REFERENCES

[1] A. Hemami, "Motion trajectory study in the scooping operation of an lhd-loader," *IEEE Transactions on Industry Applications*, vol. 30, no. 5, pp. 1333–1338, 1994. [Online]. Available: https://ieeexplore.ieee.org/document/315248

[2] C. Schenck, J. Tompson, S. Levine, and D. Fox, "Learning robotic manipulation of granular media," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 239–248. [Online]. Available: https://proceedings.mlr.press/v78/schenck17a.html

[3] C. M. Mateo, J. A. Corrales, and Y. Mezouar, "A manipulation control strategy for granular materials based on a gaussian mixture model," in *Robot 2019: Fourth Iberian Robotics Conference*, M. F. Silva, J. Luís Lima, L. P. Reis, A. Sanfeliu, and D. Tardioli, Eds. Cham: Springer International Publishing, 2020, pp. 171–183. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-36150-1_15

[4] A. Haeri and K. Skonieczny, "Gravity sensitivity of continuum numerical solvers for granular flow modeling," *Granular Matter*, vol. 24, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10035-022-01228-4

[5] A. Haeri, D. Tremblay, K. Skonieczny, D. Holz, and M. Teichmann, "Efficient numerical methods for accurate modeling of soil cutting operations," *International Symposium on Automation and Robotics in Construction (ISARC), Japan*, 2020. [Online]. Available: https://www.iaarc.org/publications/2020_proceedings_of_the_37th_isarc/efficient_numerical_methods_for_accurate_modeling_of_soil_cutting_operations.html

[6] A. Haeri and K. Skonieczny, "Accurate and real-time simulation of rover wheel traction," in *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/9438132

[7] L. Ladicky, S. Jeong, N. Bartolovic, M. Pollefeys, and M. Gross, "Physicsforests: Real-time fluid simulation using machine learning," in *ACM SIGGRAPH 2017 Real Time Live!*, ser. SIGGRAPH '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 22. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3098333.3098337

[8] S. Wiewel, M. Becher, and N. Thuerey, "Latent space physics: Towards learning the temporal evolution of fluid flow," *Computer Graphics Forum*, vol. 38, no. 2, pp. 71–82, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13620

[9] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows," *AIAA Journal*, vol. 58, no. 1, pp. 25–36, 2020. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/1.J058291?journalCode=aiaaj

[10] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," *Computer Graphics Forum*, vol. 38, no. 2, pp. 59–70, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13619

[11] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "Tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Trans. Graph.*, vol. 37, no. 4, July 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3197517.3201304

[12] L. Prantl, N. Chentanez, S. Jeschke, and N. Thuerey, "Tranquil clouds: Neural networks for learning temporally coherent features in point clouds," *CoRR*, vol. abs/1907.05279, 2019. [Online]. Available: http://arxiv.org/abs/1907.05279

[13] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4700287

[14] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International conference on machine learning*. PMLR, 2020, pp. 8459–8468. [Online]. Available: http://proceedings.mlr.press/v119/

[15] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJgbSn09Ym

[16] B. Ummenhofer, L. Prantl, N. Thurey, and V. Koltun, "Lagrangian fluid simulation with continuous convolutions," *In International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1lDoJSYDH

[17] D. Harmon and D. Zorin, "Subspace integration with local deformations," *ACM Trans. Graph.*, vol. 32, no. 4, July 2013. [Online]. Available: https://dl.acm.org/doi/10.1145/2461912.2461922

[18] J. Huang, Y. Tong, K. Zhou, H. Bao, and M. Desbrun, "Interactive shape interpolation through controllable dynamic deformation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 7, pp. 983–992, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5557868

[19] E. Sifakis and J. Barbic, "Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*, ser. SIGGRAPH '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2343483.2343501

[20] P. von Radziewsky, E. Eisemann, H.-P. Seidel, and K. Hildebrandt, "Optimized subspaces for deformation-based modeling and shape interpolation," *Comput. Graph.*, vol. 58, no. C, p. 128–138, Aug. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0097849316300590

[21] A. Haeri and K. Skonieczny, "Subspace graph physics: Real-time rigid body-driven granular flow simulation," 2021. [Online]. Available: https://arxiv.org/abs/2111.10206

[22] D. Holden, B. C. Duong, S. Datta, and D. Nowrouzezahrai, "Subspace neural physics: Fast data-driven interactive simulation," in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3309486.3340245

[23] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/14786440109462720

[24] K. P. Champion, B. Lusch, J. Kutz, and S. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, pp. 22 445 – 22 451, 2019. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1906995116

[25] A. Haeri and K. Skonieczny, "Three-dimensionsal granular flow continuum modeling via material point method with hyperelastic nonlocal granular fluidity," *Computer Methods in Applied Mechanics and Engineering*, vol. 394, p. 114904, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0045782522001876

[26] K. Kamrin, "Non-locality in granular flow: Phenomenology and modeling approaches," *Frontiers in Physics*, vol. 7, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2019.00116

[27] N. Tuomainen, D. Blanco-Mulero, and V. Kyrki, "Manipulation of granular materials by learning particle interactions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5663–5670, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9732690

[28] P. Boggs and J. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 01 1995. [Online]. Available: https://www.researchgate.net/publication/230872679_Sequential_Quadratic_Programming

[29] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*. Elsevier, 1970.

[30] L.-Z. Liao and C. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991. [Online]. Available: https://ieeexplore.ieee.org/document/86943

[31] P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold, and V. Sharma, "An sqp method for the optimal control of large-scale dynamical systems," *Journal of Computational and Applied Mathematics*, vol. 120, no. 1, pp. 197–213, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377042700003101

[32] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu, "Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks," 2022. [Online]. Available: https://arxiv.org/pdf/2205.02909.pdf

[33] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of Educational Psychology*, vol. 24,

pp. 498–520, 1933. [Online]. Available: https://babel.hathitrust.org/cgi/pt?id=wu.89097139406&view=1up&seq=3

[34] W. Li. and E. Todorov., "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, INSTICC. SciTePress, 2004, pp. 222–229. [Online]. Available: https://www.researchgate.net/publication/220271553_Iterative_Linear_Quadratic_Regulator_Design_for_Nonlinear_Biological_Movement_Systems

[35] V. Shevchenko, O. Chaplyhina, I. Pimonov, O. Reznikov, and S. Ponikarovska, "Mathematical model of a motor-grader movement in the process of performing working operations," *IOP Conference Series: Materials Science and Engineering*, vol. 985, no. 1, p. 012009, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/985/1/012009

[36] A. V. Fiacco and G. P. McCormick., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. New York-London-Sydney: John Wiley and Sons, Inc., 1968.

[37] K. Frisch, "The logarithmic potential method of convex programming," *Memorandum, University Institute of Economics, Oslo*, vol. 5, no. 6, 1955.

[38] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006. [Online]. Available: https://link.springer.com/book/10.1007/978-0-387-40065-5

[39] H. Almubarak, K. Stachowicz, N. Sadegh, and E. A. Theodorou, "Safety embedded differential dynamic programming using discrete barrier states," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2755–2762, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9682554

[40] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000. [Online]. Available: https://link.springer.com/article/10.1023/A:1026543900054

[41] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, "Pot: Python optimal transport," *Journal of Machine Learning Research*, vol. 22, no. 78, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-451.html

[42] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Transactions on Automatic Control*, vol. 26, no. 1, pp. 17–32, 1981. [Online]. Available: https://ieeexplore.ieee.org/document/1102568