Rigid Body Dynamics Simulation Based on Graph Neural Networks with Constraints

Anonymous Authors[1]

Abstract

In recent years, the utilization of Graph Neural Network (GNN)-based methods for simulating complex physical systems has significantly advanced the field of numerical science and engineering. Despite their success, current GNN-based methods for rigid body dynamic simulation are constrained to relatively simple scenarios, hindering their practical use in industrial settings where complex mechanical structures and interconnected components prevail. These methods face challenges in handling intricate force relationships within rigid bodies, primarily due to the difficulty in obtaining force-related data for objects in industrial environments. To address this, we propose a novel constraint-guided method that incorporates force computations into GNN-based simulations. The model incorporates computations related to both contact and non-contact forces into the prediction process. Additionally, it imposes physical constraints on the prediction process based on Kane's equations. We have rigorously demonstrated the model's rationality and effectiveness with thorough theoretical demonstration and empirical analysis.

1. Introduction

Simulation of rigid body dynamics plays a pivotal role in numerous domains within numerical science and engineering (Landau et al., 2008; Thijssen, 2007; Szabó & Babuška, 2021), including mechanical engineering (Brach, 2007), automotive design (Tong, 2000), biomechanics (Silva et al., 1997), and virtual reality (Sauer & Schömer, 1998), enabling the accurate modeling and analysis of object motion, stress, and interactions for optimization and performance enhancement. Traditional methods for simulating rigid body systems often rely on intricate physical equations tailored to specific domains (Huston & Passerello, 1979; Mirtich, 1996; Featherstone, 2014). This necessitates a precise understanding of the physical properties of the analyzed objects and the physical environment in which they exist (Thijssen, 2007). Furthermore, such methods require a thorough grasp of relevant physical principles and relationships. With the rise of deep learning, connectionist machine learning approaches have demonstrated outstanding performance across various fields, bypassing the need for acquiring intricate prior knowledge and rules about the internal complexities of systems through a data-driven approach (Dong et al., 2021). Deep learning has simplified many complex problems in different domains and made previously unsolvable problems tractable. In recent years, the emergence of the Graph Neural Network (GNN) based physics simulation methods has opened new avenues for addressing rigid body dynamics analysis problems using deep learning (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021). Such approaches effectively extend neural network methods from traditional domains such as images, videos, and text into the realm of real-world physical analysis. Leveraging GNNs, such methods analyze the motion states of objects modeled as graphs, achieving commendable results in related methodologies.

However, the current GNN-based methods concerning rigid body dynamic analysis are still limited to analyzing relatively simple and ideal scenarios, often focusing on the analysis of common geometric shapes (Allen et al., 2023; Bhattoo et al., 2022; Sanchez-Gonzalez et al., 2020; Han et al., 2022a; Bhattoo et al., 2022; Bishnoi et al., 2023). This limitation hinders the practical application of such methods in industrial settings. In industrial environments, complex mechanical structures often involve numerous interconnected components, including intricate force relationships within rigid bodies and the resulting mutual influences between objects. Presently, GNN-based methods encounter significant challenges in addressing such problems, primarily because force-related data for objects is often challenging to observe and obtain (Iglberger & Rüde, 2011). As a result, these methods typically rely solely on positional and velocity information of objects for analysis. The difficulty in acquiring training data relevant to forces, coupled with the necessity for force relationship analysis, places these

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

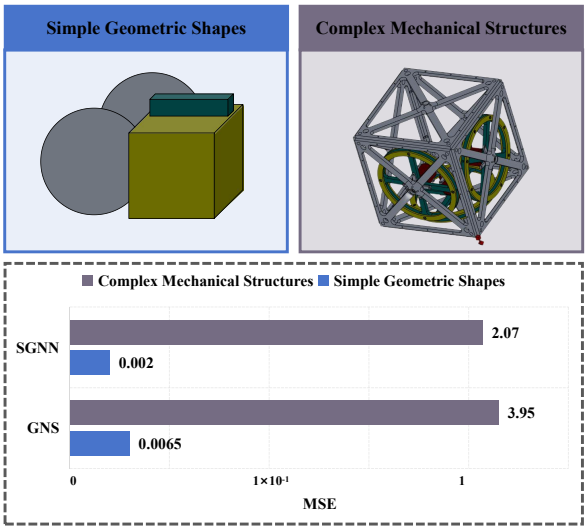


Figure 1. The comparison between the complex mechanical body scenarios we utilized and the general geometric body scenarios employed by other methods, the following illustrates the performance of two GNN-based Physical Dynamics Simulators in different scenes. From the figure, it is evident that both SGNN (Han et al., 2022a) and GNS (Sanchez-Gonzalez et al., 2020) methods exhibit significantly higher Mean Squared Error (MSE) in their predictive results for the dynamic analysis of complex mechanical structures compared to simple geometric shapes.

methods in a dilemma when applied to rigid body dynamics analysis with complex mechanical structures. The comparisons and experiments within Figure 1 validate such a conclusion, demonstrating a significant degradation in the performance of the relevant GNN-based methods for the analysis of complex mechanical structures.

To address this challenge, we endeavor to draw inspiration from traditional multibody dynamics analysis methods and incorporate relevant techniques. It is noteworthy that while GNN-based dynamic analysis methods can assist in circumventing the need for acquiring intricate prior knowledge and rules about the internal complexities of systems, the output results must adhere to the principles and rules of the physical world. Some researchers argue that a purely data-driven learning paradigm may struggle to achieve optimal results for physical problems (Bhattoo et al., 2022; Han et al., 2022a; Bishnoi et al., 2023). Therefore, we aim to integrate the principles of traditional multibody dynamics analysis with GNN-based simulators, incorporating prior constraints, to address the challenges associated with rigid body dynamics analysis in complex mechanical structures.

Building upon the aforementioned analysis, we propose a ***M**ultibody **D**ynamics Guided **G**NN **S**imulator*, dubbed MDGS, to conduct a more thorough analysis of rigid body dynamics scenarios involving intricate mechanical structures. Initially, we introduce a graph modeling pattern for efficient analysis of forces in physical systems, tailored for multibody dynamics. Subsequently, we introduce calculations related to forces, encompassing both contact and non-contact forces, into the methodology. These force calculations are performed using neural networks without the introduction of additional data beyond conventional GNN-based physical dynamics simulators. Then, the computed forces are integrated into the succeeding analyses to adapt to complex mechanical scenarios. Furthermore, we imposed constraints on the model output based on the Kane equations tailored for addressing multibody dynamics analysis problems. This ensures the accuracy and rationality of the model's computed results. The major contributions are as follows.

- We introduce a novel method, referred to as MDGS, to incorporate force computation into GNN-based simulations for the analysis of rigid body systems. This approach enhances the treatment of multibody dynamics scenarios, expanding the potential applications of such methods in industrial settings.
- We imposed physical constraints on our method based on Kane's equations, offering a more precise analysis of rigid body dynamics. These constraints are derived from rigorous theoretical analysis and proof.
- We provide the implementations of MDGS for multibody dynamics simulation tasks and tested on multiple scenarios, the results consistently showcase the outstanding performance of MDGS.

2. Related Works

2.1. GNN-based Physical Dynamics Simulators

GNN-based physical dynamics simulators (Sanchez-Gonzalez et al., 2020) leverages GNNs (Scarselli et al., 2009; Kipf & Welling, 2017) to simulate predictive representations for graph-modeled physical systems. Such approaches find widespread applications in computing atomic forces (Hu et al., 2021), simulating particle-modeled and mesh-modeled physical systems (Li et al., 2019; Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Han et al., 2022b). In recent research, some methods (Han et al., 2022a; Bhattoo et al., 2022; Bishnoi et al., 2023) have proposed incorporating prior knowledge from relevant domains into the GNN-based physical dynamics simulator to enhance its performance. This includes integrating concepts such as Subequivariant and Lagrangian equations into the model framework, making the model more closely aligned with real-world physical systems. Additionally, some approaches (Linkerhägner et al., 2023) optimize the model by collecting data in real-world environments as supplementary information. Our method introduces force analysis between physical

entities in rigid body dynamics scenarios into the system and employs Kane's equations (Kane & Levinson, 1985) to impose constraints on the system output, ensuring a more realistic simulation performance.

2.2. Dynamics Analysis

The framework of traditional mechanical analysis relies on either force-based or energy-based approaches(Wyk et al., 2022) for rigid body dynamics analysis. Energy-based approaches involve the computation of all the unknown energy based on the equations of equilibrium and hence are cumbersome for large structures(Finzi et al., 2020). The force-based multibody dynamics model solves the forward kinematics and inverse dynamics of the main mechanisms by employing two recursive algorithms centered around the Newton–Euler formulation(Gonçalves et al., 2023). And some methods use ODEs to calculate multibody corresponding conditions to approximate an optimal solution(Schubert et al., 2023; Nada & Bayoumi, 2023; Zhang et al., 2023). We amalgamate select concepts from traditional rigid body dynamics analysis with the newly introduced GNN-based approach to achieve improved and simplified predictions.

3. Methodology

3.1. Preliminary

3.1.1. GNN-BASED SIMULATORS

GNN-based simulators predict the dynamical states of physical systems. Consider a physical system comprising M elements, collectively forming N objects. At time t, GNN-based physical dynamics simulators model the aforementioned physical system using a graph $G^{(t)} = \{\mathcal{V}^{(t)}, \mathcal{E}^{(t)}\}$, where the node set $\mathcal{V}^{(t)}$ of graph $G^{(t)}$ represents different elements constituting objects. The feature $\boldsymbol{z}_i^{(t)}$ of node i in the set $\mathcal{V}^{(t)}$ includes position information $\vec{\boldsymbol{x}}_i^{(t)}$, velocity information $\vec{\boldsymbol{v}}_i^{(t)}$, and $\boldsymbol{k}_i$ representing the properties of the object to which the element belongs. In other words, $\boldsymbol{z}_i^{(t)}$ is defined as the concatenation of these vectors: $\boldsymbol{z}_i^{(t)} = [\vec{\boldsymbol{x}}_i^{(t)} || \vec{\boldsymbol{v}}_i^{(t)} || \boldsymbol{k}_i]$, where '||' denotes vector concatenation along the first dimension. $\boldsymbol{Z}^{(t)} = \{\boldsymbol{z}_i^{(t)}\}_{i=1}^{M}$ denotes the set of all $\boldsymbol{z}_i^{(t)}$. The edge set $\mathcal{E}^{(t)}$ of graph $G^{(t)}$ characterizes the connectivity relationships between elements. For instance, some approaches (Han et al., 2022a; Bhattoo et al., 2022) judge that if the distance between element i and j is less than a certain threshold, the edge (i, j) is connected. Subsequently, GNN-based simulator utilizes the GNN model $g(\cdot)$ to predict the values of $\boldsymbol{Z}^{(t+1)}$ based on $G^{(t)}$, such process can be formalized as follows:

$$\boldsymbol{Z}^{(t+1)} = g(G^{(t)}). \tag{1}$$

3.1.2. KANE'S EQUATION

Kane's equation (Kane & Levinson, 1985) is a mathematical formulation used in the field of multibody dynamics to describe the motion of interconnected rigid bodies. For a nonholonomic system $\mathcal{S}$, the number of independently variable motions or deformations is referred to as degrees of freedom. For each degree of freedom, there exists a corresponding generalized velocity. These velocity vectors associated with generalized coordinates are typically denoted by $\dot{q}$, where q represents the generalized coordinates. Specifically, with respect to the generalized coordinate q_i and its corresponding generalized velocity $\dot{q}_i$, if $\dot{q}_i$ is independent of the other velocities in the system, meaning that it cannot be expressed or derived from other velocities in the system's description, then $\dot{q}_i$ is referred to as an independent velocity. Furthermore, the partial velocity $u'_{ij} = \frac{\partial \dot{q}_i}{\partial \dot{q}_j}$ denotes the partial derivative of the velocity with respect to coordinate q_i when coordinate q_j is varied, within the context of a multi-coordinate system.

With the aforementioned concepts, we could define the general active force according to the γ-th independent velocity of $\mathcal{S}$ as follows:

$$K_\gamma = \sum_{i=1}^{n} \vec{F}_i \cdot \vec{u}'_{i\gamma}, \tag{2}$$

where $\vec{F}_i$ denotes the force acting on the i-th mass point in the system $\mathcal{S}$, and $\vec{u}'_{i\gamma}$ represents its partial velocity with respect to the γ-th independent velocity, n is the number of mass points. Likewise, we could define the generalized inertia forces according to the γ-th independent velocity as follows:

$$K^*_\gamma = \sum_{i=1}^{n} \Big(- m_i \ddot{\vec{r}}_i \Big) \cdot \vec{u}'_{i\gamma}, \tag{3}$$

where m_i represents mass, while $\vec{r}_i$ represents the vector radius. The Kane's equation can then be formulated as follows:

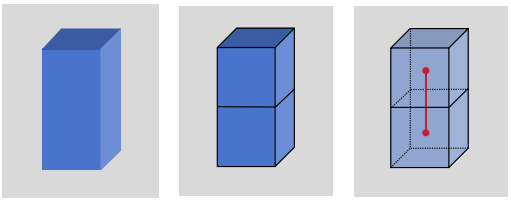
$$K^*_\gamma + K_\gamma = 0, \gamma = \{1, 2, ..., d\}, \tag{4}$$

d is the number of independent velocities.

3.2. Method

3.2.1. GRAPH CONSTRUCTION

Two commonly used object partitioning approaches in the field of physical simulation are point clouds (Gross et al., 2002) and finite elements (Pasciak, 1995), both of which have been widely applied in GNN-based simulators. Point clouds essentially involve decomposing objects into indivisible elements using point cloud construction techniques. On the other hand, the finite element method divides objects



(a) Original Body (b) Partitioned Body (c) Graph Construction

Figure 2. Illustration of the Graph Construction Process.

into individual polyhedral elements, where the edges and vertices of these elements constitute the graph for GNN-based simulators. In our approach, we integrate the above two methods by employing the finite element method to partition rigid bodies into different elements and designate the centroid of each element as a node in the graph. In the specific implementation, we assume uniform density within each element and utilize its geometric center as its centroid. We believe that this approach maximally preserves the kinematic properties of objects while aligning more closely with the commonly used analytical paradigm in multibody dynamics (Kane & Levinson, 1985). Figure 2 provides an illustrative example of our graph construction method. As illustrated in Figure 2(c), the centroids serve as nodes in the graph, and the centroids of adjacent elements are connected by edges. The specific methodology for edge construction will be elaborated in detail in Section 3.2.2.

3.2.2. Calculating Forces

In the realm of multibody dynamics, the magnitude of forces acting on an object plays a crucial role in determining its state. However, due to the inherent difficulty in measuring and observing the force conditions of objects, existing GNN-based simulators do not currently incorporate force information into the prediction computation process. To address this issue, we employ neural networks to predict the forces acting on rigid bodies and integrate these predicted values into the model. Specifically, we categorize the forces acting on objects into two components: contact force and non-contact force.

For contact forces, such as frictional and collision forces, we propose the construction of a contact-oriented graph $\bar{G}^{(t)} = \{\mathcal{V}^{(t)}, \bar{\mathcal{E}}^{(t)}\}$. In $\bar{G}^{(t)}$, $\mathcal{V}^{(t)}$ represents the centroids corresponding to distinct elements, with attributes comprising the input positional and velocity information of these elements at time t. Additionally, $\mathcal{V}^{(t)}$ includes property about the particular object each element is affiliated to, providing the information concerning the whole object. $\bar{\mathcal{E}}^{(t)}$ comprises only those edges determined by the existence of contact between elements. As depicted in Figure 4(a), if the distance between the centroids of two elements, denoted as i and j, is less than r, and they belong to different objects, we consider these elements to be in contact, and the edge (i, j) is established, r is a threshold defined by a hyperparameter. In this manner, we derive a graph $\bar{G}^{(t)}$ specifically illustrating contact relationships. Subsequently, we employ $\bar{G}^{(t)}$ to predict the force set $\bar{F}^{(t)}$, where $\bar{F}^{(t)} \in \mathbb{R}^{N\times 3}$. The i-th vector $\bar{\boldsymbol{f}}_i^{(t)}$ of $\bar{F}^{(t)}$ denotes the contact forces acting upon the centroid of the i-th element. Formally, we have:

$$\bar{F}^{(t)} = g^{\bar{F}}(\bar{G}^{(t)}), \tag{5}$$

where $g^{\bar{F}}(\cdot)$ denotes a GNN specifically designed for contact force prediction. $\bar{\boldsymbol{f}}_i^{(t)}$ will be set to the zero vector if i is not an endpoint of any edge in $\bar{\mathcal{E}}^{(t)}$.

For non-contact forces, such as gravity, we propose constructing the object-oriented graph $\widetilde{G}^{(t)} = \{\mathcal{V}^{(t)}, \widetilde{\mathcal{E}}^{(t)}\}$, where $\mathcal{V}^{(t)}$ remains identical to that of $\bar{G}^{(t)}$. As illustrated in Figure 4(b), if two elements i and j belong to the same object and share faces between them, then an edge (i, j) is established within $\widetilde{\mathcal{E}}^{(t)}$. Subsequently, we predict the non-contact force set $\widetilde{F}^{(t)}$. Formally, this prediction is expressed as:

$$\widetilde{F}^{(t)} = g^{\widetilde{F}}(\widetilde{G}^{(t)}), \tag{6}$$

where $g^{\widetilde{F}}(\cdot)$ denotes a GNN specifically designed for non-contact force prediction. The final force set is obtained by combining the contact and non-contact force sets:

$$F^{(t)} = \{\boldsymbol{f}_i^{(t)} : \boldsymbol{f}_i^{(t)} = \bar{\boldsymbol{f}}_i^{(t)} + \widetilde{\boldsymbol{f}}_i^{(t)}\}, \tag{7}$$

where $\boldsymbol{f}_i^{(t)}$ and $\widetilde{\boldsymbol{f}}_i^{(t)}$ represent the i-th force vector of $F^{(t)}$ and $\widetilde{F}^{(t)}$ respectively.

3.2.3. Making Predictions

Subsequently, we predict $G^{(t+1)}$ based on both $G^{(t)}$ and $F^{(t)}$. We first attach the calculated force to the attribute $\boldsymbol{z}$ of each node. For the i-th node with attribute feature $\boldsymbol{z}_i$, the new attribute feature $\boldsymbol{z}_i'$ is calculated as follows:

$$\boldsymbol{z}_i' = [\boldsymbol{z}_i || \lambda F_i^{(t)}], \tag{8}$$

where λ is a hyperparameter for controlling the magnitude of the influence of $F^{(t)}$. We denote the modified $G^{(t)}$ as $G'^{(t)}$ and calculated the prediction as follows:

$$\boldsymbol{Z}^{(t+1)} = g(G'^{(t)}), \tag{9}$$

where $g(\cdot)$ denotes the GNN for state prediction. Motivated by (Allen et al., 2023), we introduced a rigid body structural constraint module in our method. This module ensures the invariance of rigid body structures in the predicted results. Specifically, we calculate the relative positions of various nodes on the object based on the initial values. Subsequently,

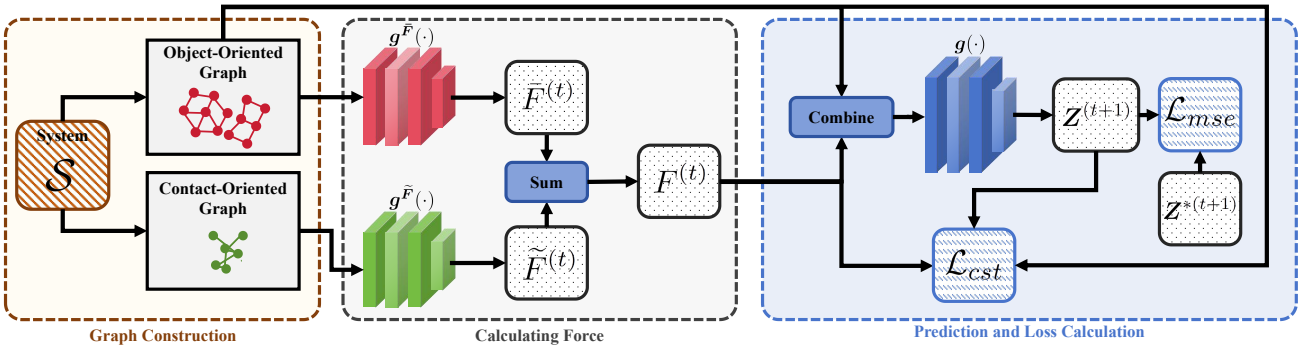
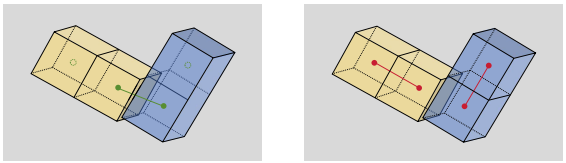


Figure 3. The framework of our proposed method. In the figure, the framework is segmented based on the stages of method execution, dividing it into three parts.



(a) Contact-oriented graph. (b) Object-oriented graph.

Figure 4. Illustration of contact-oriented graph and object-oriented graph.

we re-align our predicted results to the original relative positions, ensuring that the rigid body structure remains unchanged. A detailed description of this module can be found in **Appendix** B.1.

$\boldsymbol{Z}^{(t+1)}$ will then be utilized for calculating an MSE loss against the ground truth state $\boldsymbol{Z}^{*(t+1)}$ for the backpropagation of the whole framework:

$$\mathcal{L}_{mse} = \text{MSE}(\boldsymbol{Z}^{(t+1)}, \boldsymbol{Z}^{*(t+1)}). \tag{10}$$

3.2.4. Model Constraints Based on Kane's Equation.

So far, we have successfully incorporated force computation to assist in predictions within the GNN-based simulators. However, such computation still faces challenges in ensuring accuracy. As discussed within section 3.1.2, Kane's equations provide a physical constraint for rigid body dynamical systems, and we aim to employ such constraints to refine our model, enhancing the analytical fidelity to physical reality.

Theoretically, a neural network capable of accurately predicting rigid body systems should adhere to the following principles:

Theorem 3.1. *Consider a rigid body dynamics system denoted as $\mathcal{S}$ with d independent velocities. At time t, the system's state is represented as $G^{(t)}$, constructed according to the detailed graph construction procedure outlined in Section 3.2.1 and 3.2.2. For any Graph Neural Network (GNN) denoted as $g(\cdot)$, which demonstrates accurate prediction capabilities for $F^{(t)}$ and $\boldsymbol{Z}^{(t+1)}$ based on $G^{(t)}$, the following equation must be satisfied:*

$$\sum_i F_{\gamma,i}^{(t)} u_\gamma - \sum_i M_i \frac{1}{\delta}\big(g(G^{(t)})_{[v_\gamma,i]} - v_\gamma^{(t)}\big) u_\gamma + \Psi^{(t)} = 0, \forall \gamma \in \{1, 2, ..., d\}, \tag{11}$$

where γ denote the index of independent velocity, $F_{\gamma,i}^{(t)}$ is the predicted force value of element i along γ-th independent velocity, M_i is the mass of element i, $g(G^{(t)})_{[v_\gamma,i]}$ denote the predicted $v_\gamma^{(t+1)}$ with $g(\cdot)$, δ is the time step length, $\Psi^{(t)}$ is a disturbance term directly proportional to the sum of the radii from different mass points to the center of mass in each element.

The proof can be found in Appendix A.1. Theorem 3.1 furnishes us with a specific methodology for constraining the model. From theorem 3.1, it is evident that the constraints proposed in Equation 11 hold along any independent velocity. In practical implementation, we can determine the number of equations the model needs to satisfy based on the distinct independent velocities associated with different systems.

To further constrain the perturbation terms in Theorem 3.1, we propose the following theorem:

Theorem 3.2. *Given the conditions within Theorem 3.1, then at time t, the following inequality holds:*

$$-\tau \sum_{i=1}^{n} (|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma - \sum_{i=1}^{n} \frac{2}{5} M_i \tau^2 \dot{\omega}_i^{(t)} \le \Psi^{(t)} \le \tau \sum_{i=1}^{n} (|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma + \sum_{i=1}^{n} \frac{2}{5} M_i \tau^2 \dot{\omega}_i^{(t)}, \tag{12}$$

where $\vec{\boldsymbol{R}}_i^{(t)}$ denotes the interval force act on element i, τ is the max radius τ among all elements, ω_γ denotes unit angular velocity value along the direction of the γ-th independent

velocity, $\dot{\omega}_i^{(t)}$ represents the max angular acceleration of element i.

Based on Theorem 3.1 and Theorem 3.2, we proposed the following loss function for the upper bound of constrain:

$$\mathcal{L}_{sup} = \text{Relu}\Big(\sum_i F_{\gamma,i}^{(t)} u_\gamma - \sum_i M_i \frac{1}{\delta} \big(g(G^{(t)})_{[v_\gamma,i]} - v_\gamma^{(t)} \big) u_\gamma - \tau \sum_{i=1}^{n} (|\vec{\boldsymbol{R}}_i^{(t)}|) \omega_\gamma - \sum_{i=1}^{n} \frac{2}{5} M_i \tau^2 \dot{\omega}_i^{(t)} \Big), \quad (13)$$

where $F_{\gamma,i}^{(t)}$ and $|\vec{\boldsymbol{R}}_i^{(t)}|$ can be calculated with force prediction, while $\dot{\omega}_i^{(t)}$ is calculated based on variant within coordinates, $\text{Relu}(\cdot)$ denote the ReLU function, M_i is calculated based on the properties of the object, the details can be found in **Appendix** B.1. Similarly, we can calculate the loss function for the lower bound of constrain:

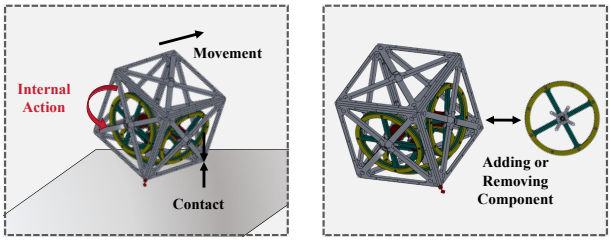
$$\mathcal{L}_{inf} = \text{Relu}\Big(- \sum_i F_{\gamma,i}^{(t)} u_\gamma + \sum_i M_i \frac{1}{\delta} \big(g(G^{(t)})_{[v_\gamma,i]} - v_\gamma^{(t)} \big) u_\gamma - \tau \sum_{i=1}^{n} (|\vec{\boldsymbol{R}}_i^{(t)}|) \omega_\gamma - \sum_{i=1}^{n} \frac{2}{5} M_i \tau^2 \dot{\omega}_i^{(t)} \Big), \quad (14)$$

The overall loss for constrain is:

$$\mathcal{L}_{cst} = \mathcal{L}_{sup} + \mathcal{L}_{inf}. \quad (15)$$

$\mathcal{L}_{cst}$ will only be used to update $g^{\bar{F}}$ and $g^{\widetilde{F}}$. $\mathcal{L}_{cst}$ and $\mathcal{L}_{mse}$ are summed up for the final loss. The overall framework is illustrated within Figure 3.

4. Experiments



(a) Our dataset incorporates interactions between complex mechanical bodies and the environment, the overall movement of mechanical bodies, and the internal actions of components within the mechanical bodies.

(b) We modify the attributes and motion patterns of mechanical bodies by adding or removing mechanical components within them. Such modifications allow us to test the performances of the algorithms across various scenarios.

Figure 5. The underlying principles behind the construction of our dataset.

4.1. Environmental Setup

4.1.1. Dataset and Systems

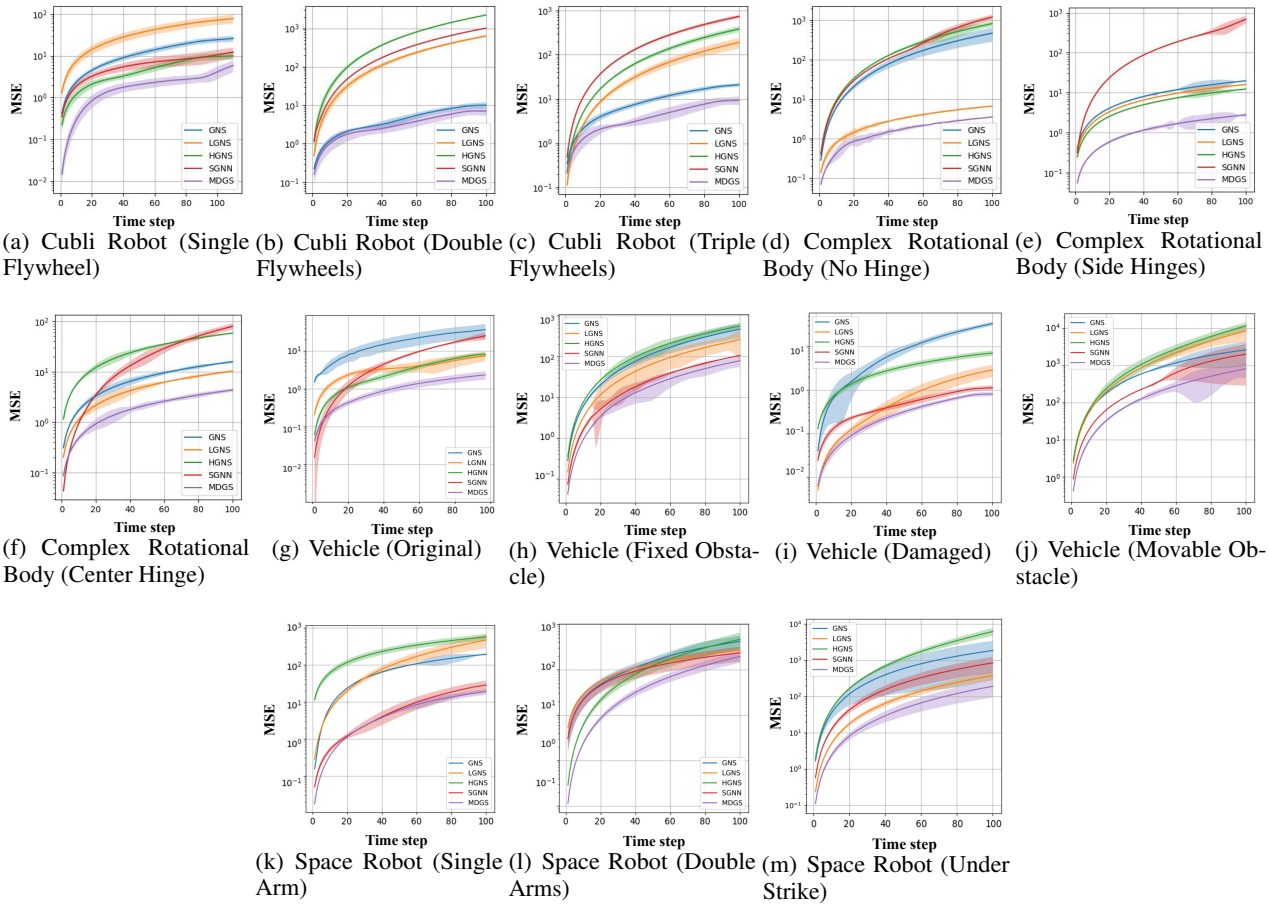
To validate our methodology, we conduct comprehensive testing on diverse datasets derived from four distinct mechanical systems. Specifically, we utilized the Cubli robot (Gajamohan et al., 2012) (a metallic mechanical cube capable of performing various maneuvers using three flywheels), complex rotating bodies (intricate mechanical structures featuring components such as bearings and hinges), vehicles (mechanical cars operating under different conditions and environments), and space robots (Wang et al., 2022) (robots equipped with mechanical arms for executing specific tasks in space environments). Utilizing Mujoco (Todorov et al., 2012) for physics-based modeling, we systematically vary motion patterns and mechanical components to generate six datasets comprising 2200 trajectories, with each trajectory consisting of 120 time steps, enabling a thorough evaluation of the robustness of our approach. We describe the construction principles within Figure 5. Specifically, in the context of the Cubli robot, we have systematically varied the number of reaction wheels, creating distinct rigid body systems comprising single, double, and triple flywheels. For intricate rotational bodies, we have constructed three structures with no hinge, hinges on both sides, and hinges on the central position, respectively. All utilized physical systems are demonstrated within Appendix B.3

4.1.2. Baselines

We conduct a comparative analysis of our proposed MDGS against various baselines, including GNS (Sanchez-Gonzalez et al., 2020), SGNN (Han et al., 2022a), LGNS (Bhattoo et al., 2022), and HGNS (Bishnoi et al., 2023). Specifically, GNS employs a pure Graph Neural Network (GNN) for simulating physical systems, while SGNN introduces the concept of subequivariant in the physical system into the GNN-based physical dynamics simulator. Furthermore, LGNS and HGNS leverage Lagrangian equations and Hamiltonian mechanics, respectively, to guide and constrain the GNN model, enhancing its capability to simulate physical reality. Apart from GNS, the other three methods incorporate varying degrees of prior knowledge from physics to reinforce the models. Through comparisons with these baselines, we can effectively analyze the algorithmic performance of our approach in the dynamic analysis of complex mechanical bodies.

4.1.3. Experimental Settings

Due to the widely applicable enhancement provided by our designed rigid body structural constraint module on the accuracy of the GNN-based simulator in predicting rigid body problems, to ensure fairness in comparisons, we incorporated this module into other baselines as well, elevating their



(a) Cubli Robot (Single Flywheel) (b) Cubli Robot (Double Flywheels) (c) Cubli Robot (Triple Flywheels) (d) Complex Rotational Body (No Hinge) (e) Complex Rotational Body (Side Hinges)

(f) Complex Rotational Body (Center Hinge) (g) Vehicle (Original) (h) Vehicle (Fixed Obstacle) (i) Vehicle (Damaged) (j) Vehicle (Movable Obstacle)

(k) Space Robot (Single Arm) (l) Space Robot (Double Arms) (m) Space Robot (Under Strike)

Figure 6. Experimental results on precision evaluation.

performance equivalently. Simultaneously, in Section 4.3.1, we presented the impact of adding or removing the rigid body structural constraint module on other baselines. Our experiments utilized MSE between predicted and ground truth values as the metric for evaluating the performance of each model. To demonstrate statistical significance, we followed the methodology outlined in (Han et al., 2022a), conducting each experiment 10 times and averaging the results. Specific hyperparameters and the backbones are detailed in **Appendix** B.2.

4.2. Performance Evaluation

4.2.1. Precision

Building upon the aforementioned experimental setup, we conducted performance testing experiments for both MDGS and other baseline algorithms across a total of six datasets. Figure 6 presents the experimental results. It is evident from the results that our proposed MDGS consistently outperforms the majority of comparisons. Figure 8 demonstrated the visualized results. We provide results represented in digital form in the **Appendix** B.4.

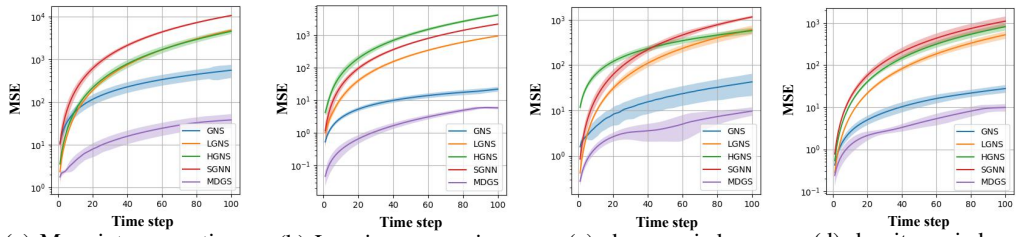
4.2.2. Generalization

We also conducted comparisons on the generalization performance of various methods. Using the complete Cubli Robot with three flywheels, we introduced variations to different attributes of the system. Subsequently, we examined whether the methods trained on the original Cubli Robot dataset could accurately predict the changed system and observed variations in algorithm prediction errors on the new test set. The test results are presented in the Figure **??**. It is evident that our method achieves the best performance. Additionally, an interesting observation is that, apart from MDGS, other methods incorporating prior knowledge of the physical system perform poorly in generalization experiments.

4.3. In-Depth Study

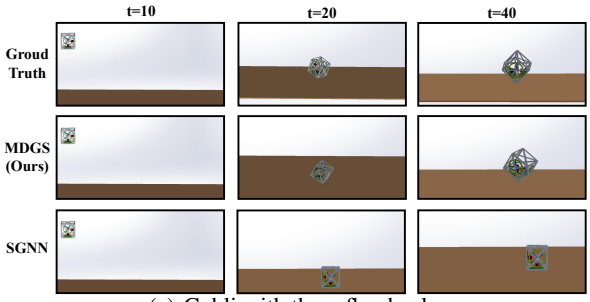
4.3.1. Structural Constraint Module Analysis

Validation experiments were conducted to analyze the capabilities of the proposed structural constraint module. A comparative study was performed on the dataset of the Cubli robot with all three flywheels, evaluating the performance of

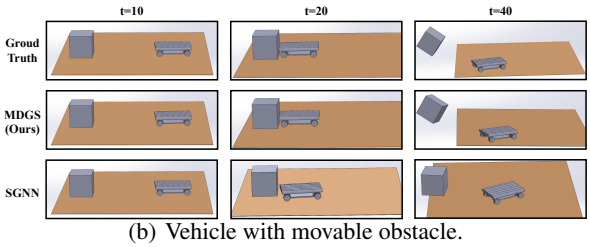


(a) More intense motion. (b) Less intense motion (c) shape varied (d) density varied

Figure 7. Experimental results on generalization.



(a) Cubli with three flywheels.



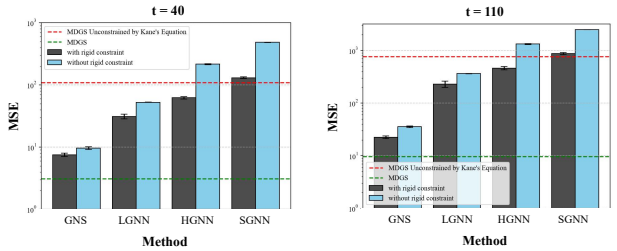
(b) Vehicle with movable obstacle.

Figure 8. Visualized experimental results.

methods with and without the structural constraint module. In Figure 9, the performance of the respective models at different time steps is illustrated. It can be observed that all methods exhibit improved performance, i.e., lower MSE, after incorporating the structural constraint module. This experimental result demonstrates the effectiveness of the proposed structural constraint module in enhancing the accuracy of model predictions in rigid body dynamics analysis, providing additional constraint information about the model structure for further predictions. Simultaneously, the experiment validates the necessity of integrating the structural constraint module when comparing our approach with other baselines. We provide results represented in digital form in the **Appendix** B.4.

4.3.2. Ablation Study

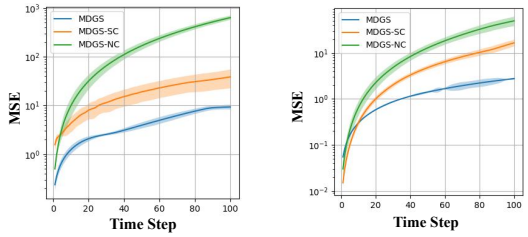
For a more in-depth analysis of our method, we conducted a series of ablation experiments. Specifically, we modified the constraints of Kane's equations to create two baseline models, namely MDGS-NC and MDGS-SC. MDGS-NC removes all constraints from Kane's equations, while MDGS-SC retains only an independent velocity constraint. The



(a) t=40 (b) t=110

Figure 9. Experimental results of baselines with and without structural constraint module.

experimental results are depicted in Figure 10. It is evident that the constraints of Kane's equations indeed contribute to the improved performance of our method. Table 3 further illustrates the comparison between MDGS-NC, MDGS-SC, and other baselines.



(a) Cubli robot with triple flywheels. (b) Complex rotational body with side hinges.

Figure 10. Ablation experiments results.

5. Conclusion

In this paper, we propose a novel approach, MDGS, to incorporate force analysis into GNN-based rigid body system simulators, aiming to enhance the accuracy of such methods in complex mechanical scenarios. We substantiate the design of MDGS through theoretical foundations and proofs and validate its performance through a series of experiments. We contend that MDGS can provide assistance to some extent in the application of GNN-based rigid body system simulators in industrial settings.

Impact Statements

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Allen, K. R., Guevara, T. L., Rubanova, Y., Stachenfeld, K., Sanchez-Gonzalez, A., Battaglia, P., and Pfaff, T. Graph network simulators can learn discontinuous, rigid contact dynamics. In *Conference on Robot Learning*, pp. 1157–1167. PMLR, 2023.

Bhattoo, R., Ranu, S., and Krishnan, N. M. A. Learning articulated rigid body dynamics with lagrangian graph neural network. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Bishnoi, S., Bhattoo, R., Jayadeva, J., Ranu, S., and Krishnan, N. A. Learning the dynamics of physical systems with hamiltonian graph neural networks. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

Brach, R. M. *Mechanical impact dynamics: rigid body collisions*. Brach Engineering, LLC, 2007.

Dong, S., Wang, P., and Abbas, K. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

Featherstone, R. *Rigid body dynamics algorithms*. Springer, 2014.

Finzi, M., Wang, K. A., and Wilson, A. G. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Gajamohan, M., Merz, M., Thommen, I., and D'Andrea, R. The cubli: A cube that can jump up and balance. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 3722–3727. IEEE, 2012. doi: 10.1109/IROS.2012.6385896. URL https://doi.org/10.1109/IROS.2012.6385896.

Gonçalves, F., Ribeiro, T., Ribeiro, A. F., Lopes, G., and Flores, P. Multibody model of the human-inspired robot charmie. *Multibody System Dynamics*, pp. 1–28, 2023.

Gross, M. H., Pfister, H., Zwicker, M., Pauly, M., Stamminger, M., and Alexa, M. Point based computer graphics. In Coquillart, S. and Müller, H. (eds.), *23rd Annual Conference of the European Association for Computer Graphics, Eurographics 2002 - Tutorials, Saarbrücken, Germany, September 2-6, 2002*. Eurographics Association, 2002. doi: 10.2312/EGT.20021060. URL https://doi.org/10.2312/egt.20021060.

Han, J., Huang, W., Ma, H., Li, J., Tenenbaum, J., and Gan, C. Learning physical dynamics with subequivariant graph neural networks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022a.

Han, X., Gao, H., Pfaff, T., Wang, J., and Liu, L. Predicting physics in mesh-reduced space with temporal attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=XctLdNfCmP.

Hu, W., Shuaibi, M., Das, A., Goyal, S., Sriram, A., Leskovec, J., Parikh, D., and Zitnick, C. L. Forcenet: A graph neural network for large-scale quantum calculations. In *ICLR 2021 SimDL Workshop*, volume 20, 2021.

Huston, R. L. and Passerello, C. On multi-rigid-body system dynamics. *Computers & Structures*, 10(3):439–446, 1979.

Iglberger, K. and Rüde, U. Large-scale rigid body simulations. *Multibody system dynamics*, 25:81–95, 2011.

Kane, T. R. and Levinson, D. A. *Dynamics, theory and applications*. McGraw Hill, 1985.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Landau, R. H., Mejía, M. J. P., and Bordeianu, C. C. *A survey of computational physics: introductory computational science*. Princeton University Press, 2008.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJgbSn09Ym.

Linkerhägner, J., Freymuth, N., Scheikl, P. M., Mathis-Ullrich, F., and Neumann, G. Grounding graph network simulators using physical sensor observations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=jsZsEd8VEY.

Mirtich, B. V. *Impulse-based dynamic simulation of rigid body systems*. University of California, Berkeley, 1996.

Nada, A. and Bayoumi, M. Development of a constraint stabilization method of multibody systems based on fuzzy logic control. *Multibody System Dynamics*, pp. 1–33, 2023.

Pasciak, J. E. The mathematical theory of finite element methods (susanne c. brenner and l. ridgway scott). *SIAM Rev.*, 37(3):472–473, 1995. doi: 10.1137/1037111. URL https://doi.org/10.1137/1037111.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468. PMLR, 2020. URL http://proceedings.mlr.press/v119/sanchez-gonzalez20a.html.

Sauer, J. and Schömer, E. A constraint-based approach to rigid body dynamics for virtual reality applications. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 153–162, 1998.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605. URL https://doi.org/10.1109/TNN.2008.2005605.

Schubert, M., Sato Martín de Almagro, R. T., Nachbagauer, K., Ober-Blöbaum, S., and Leyendecker, S. Discrete adjoint method for variational integration of constrained odes and its application to optimal control of geometrically exact beam dynamics. *Multibody System Dynamics*, pp. 1–28, 2023.

Silva, M., Ambrosio, J., and Pereira, M. Biomechanical model with joint resistance for impact simulation. *Multibody System Dynamics*, 1(1):65–84, 1997.

Szabó, B. and Babuška, I. Finite element analysis: Method, verification and validation. 2021.

Thijssen, J. *Computational physics*. Cambridge university press, 2007.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109.

Tong, Y. Y. Vehicle dynamic simulations based on flexible and rigid multibody models. Technical report, SAE Technical Paper, 2000.

Wang, S., Cao, Y., Zheng, X., and Zhang, T. Collision-free trajectory planning for a 6-dof free-floating space robot via hierarchical decoupling optimization. *IEEE Robotics Autom. Lett.*, 7(2):4953–4960, 2022. doi: 10.1109/LRA.2022.3152698. URL https://doi.org/10.1109/LRA.2022.3152698.

Wyk, K. V., Xie, M., Li, A., Rana, M. A., Babich, B., Peele, B., Wan, Q., Akinola, I., Sundaralingam, B., Fox, D., Boots, B., and Ratliff, N. D. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics Autom. Lett.*, 7(2):3202–3209, 2022. doi: 10.1109/LRA.2022.3143311. URL https://doi.org/10.1109/LRA.2022.3143311.

Zhang, X., Rui, X., Zhang, J., Gu, J., and Zhang, L. Statics analysis based on the reduced multibody system transfer matrix method. *Multibody System Dynamics*, pp. 1–25, 2023.

A. Proofs

A.1. The Proof of Theorem 3.1

We first present the following lemma to assist in establishing the rationality of the theorem.

Lemma A.1. *By partitioning each rigid body within system $\mathcal{S}$ into an element set $C = \{c_i\}_{i=0}^n$ following the procedure in Section 3.2.1, the motion states of all constructed elements within C, as well as the external forces $\{\vec{\boldsymbol{F}}_i\}_{i=0}^n$ acting on these elements, conform to Kane's equation.*

Proof. According to the definition of generalized active force (Kane & Levinson, 1985), $\mathcal{S}$'s generalized active force K_γ according to the γ-th independent velocity can be expressed as follows:

$$K_\gamma = \sum_{k=1}^{m} \vec{\boldsymbol{F}}_k \cdot \vec{\boldsymbol{u}}_{k,\gamma}, \tag{16}$$

where m represents the number of point masses existing within $\mathcal{S}$, $\vec{\boldsymbol{F}}_k$ denotes the external force vector acting on point mass k, $\vec{\boldsymbol{u}}_{k,\gamma}$ denotes the partial speed corresponding to the γ-th independent speed. The i-th element can be regarded as a rigid that consists of multiple point masses. Therefore, we have:

$$\begin{aligned} K_\gamma &= \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} \vec{\boldsymbol{F}}'_{i,l} \cdot (\vec{\boldsymbol{u}}_{i,\gamma} + \vec{\boldsymbol{\omega}}_{i,\gamma} \times \vec{\boldsymbol{r}}_{i,l}) = \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} \vec{\boldsymbol{F}}'_{i,l} \cdot \vec{\boldsymbol{u}}_{i,\gamma} + \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} \vec{\boldsymbol{F}}'_{i,l} \cdot (\vec{\boldsymbol{\omega}}_{i,\gamma} \times \vec{\boldsymbol{r}}_{i,l}) \\ &= \sum_{i=1}^{n} \Big(\sum_{l=1}^{\widetilde{m}_i} \vec{\boldsymbol{F}}'_{i,l} \Big) \cdot \vec{\boldsymbol{u}}_{i,\gamma} + \sum_{i=1}^{n} \Big(\sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times \vec{\boldsymbol{F}}'_{i,l}) \Big) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma} = \sum_{i=1}^{n} \vec{\boldsymbol{R}}_i \cdot \vec{\boldsymbol{u}}_{i,\gamma} + \sum_{i=1}^{n} \vec{\boldsymbol{M}}_i \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}, \end{aligned} \tag{17}$$

where $\vec{\boldsymbol{F}}'_{i,l}$ denote the external force acting on l-th point mass among the $\widetilde{m}_i$ point masses that consist element c_i, $\vec{\boldsymbol{u}}_{i,\gamma}$ denote the partial velocity according to the γ-th independent velocity acting on the mass center of the element c_i, $\vec{\boldsymbol{\omega}}_{i,\gamma}$ denote the partial angular velocity of element c_i according to the γ-th independent velocity, $\vec{\boldsymbol{r}}_{i,l}$ denotes the radius vector of the l-th point mass towards the mass center of the element c_i. $\vec{\boldsymbol{R}}_i$ and $\vec{\boldsymbol{M}}_i$ represent the forces and moments acting on individual elements. Such forces and moments can be calculated based on the properties of external forces $\{\vec{\boldsymbol{F}}_i\}_{i=0}^n$. Consequently, we can derive that K_γ can be expressed as aggregating the external forces and moments acting on these elements.

Furthermore, we can represent $\mathcal{S}'s$ generalized inertia force as follows:

$$K^*_\gamma = \sum_{k=1}^{m} (-m_k \vec{\boldsymbol{a}}_k) \cdot \vec{\boldsymbol{u}}_{k,\gamma}, \tag{18}$$

where m_k denote the mass of the k-th point mass, $\boldsymbol{a}'_k$ represents its acceleration. Equation 18 can be reformulated as:

$$\begin{aligned} K^*_\gamma &= \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} (-m_{i,l}\vec{\boldsymbol{a}}_{i,l}) \cdot (\vec{\boldsymbol{u}}_{i,\gamma} + \vec{\boldsymbol{\omega}}_{i,\gamma} \times \vec{\boldsymbol{r}}_{i,l}) = \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} (-m_{i,l}\vec{\boldsymbol{a}}_{i,l}) \cdot \vec{\boldsymbol{u}}_{i,\gamma} + \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} (-\vec{\boldsymbol{r}}_{i,l} \times m_{i,l}\vec{\boldsymbol{a}}_{i,l}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma} \\ &= -\sum_{i=1}^{n} M_i \vec{\boldsymbol{a}}_i \cdot \vec{\boldsymbol{u}}_{i,\gamma} - \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times m_{i,l}\vec{\boldsymbol{a}}_{i,l}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma} = -\sum_{i=1}^{n} M_i \vec{\boldsymbol{a}}_i \cdot \vec{\boldsymbol{u}}_{i,\gamma} - \sum_{i=1}^{n} \vec{\boldsymbol{L}}^*_i \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}, \end{aligned} \tag{19}$$

where $m_{i,l}$ denotes the mass of l-th point mass that consists element c_i, $\vec{\boldsymbol{a}}_{i,l}$ denotes the corresponding acceleration. M_i and $\vec{\boldsymbol{a}}_i$ denotes the mass and the acceleration of c_i, $\vec{\boldsymbol{L}}^*_i$ can be determined based on the rotational state of c_i. Therefore, we can derive that K^*_γ can be expressed as the aggregation of the motion states of elements within C. As demonstrated afore, K_γ can be expressed as aggregating the external forces and moments acting on C. As both K^*_γ and K_γ within Kane's equation can be calculated with the given factors, we can conclude that the lemma holds. □

Lemma A.1 demonstrates that the motion of element set C and external forces $\{\vec{\boldsymbol{F}}_i\}_{i=0}^n$ conform to the Kane's equation. Next, we need to prove that equation 11 holds. According to the theorem, model $g(\cdot)$ is strictly accurate. Furthermore, from

Equation 17, we have:

$$\vec{\boldsymbol{R}}_i = \sum_{l=1}^{\widetilde{m}_i} \vec{\boldsymbol{F}}_{i,l}, \tag{20}$$

which can be regarded as the resultant force acting on an element. Therefore, at time t, we have:

$$\vec{\boldsymbol{R}}_i^{(t)} = [F_{1,i}^{(t)}, F_{2,i}^{(t)}, ..., F_{d,i}^{(t)}]^\intercal, \tag{21}$$

where $F_{\gamma,i}^t$ denote the force value along coordinate q_γ at time t. Based on the statements in the theorem, due to the independence of generalized velocities from each other, we could take the generalized velocity corresponding to each generalized coordinate as independent velocity, the partial velocity $\vec{\boldsymbol{u}}'_{i,\gamma}$ can be represented as follows:

$$\vec{\boldsymbol{u}}'_{i,\gamma} = \frac{\partial \vec{\boldsymbol{r}}_i}{\partial q_\gamma} = \frac{\partial [q_{1,i}, ..., q_{d,i}]^\intercal}{\partial q_\gamma}, \tag{22}$$

where q_γ is the generalized coordinate corresponding to the γ-th independent velocity. $\vec{\boldsymbol{u}}'_{i,\gamma}$ is a one-hot vector in such circumstances. Therefore, we have:

$$\vec{\boldsymbol{R}}_i^{(t)} \cdot \vec{\boldsymbol{u}}_{i,\gamma}^{\prime(t)} = [F_{1,i}^{(t)}, F_{2,i}^{(t)}, ..., F_{d,i}^{(t)}]^\intercal \cdot \frac{\partial [q_{1,i}, ..., q_{d,i}]^\intercal}{\partial q_\gamma} = F_{\gamma,i}^{(t)} u_\gamma, \tag{23}$$

where u_γ represents the unit velocity value along the direction of the γ-th independent velocity. Therefore, at time t, we have:

$$\begin{aligned} K_\gamma^{(t)} &= \sum_{i=1}^{n} \vec{\boldsymbol{r}}_i \cdot \vec{\boldsymbol{u}}_{i,\gamma}^{\prime(t)} + \sum_{i=1}^{n} \vec{\boldsymbol{M}}_i^{(t)} \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} = \sum_{i=1}^{n} F_{\gamma,i}^{(t)} + \sum_{i=1}^{n} \vec{\boldsymbol{M}}_i^{(t)} \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} \\ &= \sum_{i=1}^{n} F_{\gamma,i}^{(t)} + \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)}. \end{aligned} \tag{24}$$

Similarly, we can also derive the following result:

$$K_\gamma^{*(t)} = -\sum_{i=1}^{n} M_i \vec{\boldsymbol{a}}_i^{(t)} \cdot \vec{\boldsymbol{u}}_{i,\gamma}^{\prime(t)} - \sum_{i=1}^{n} \vec{\boldsymbol{L}}_i^{*(t)} \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} = -\sum_{i=1}^{n} M_i a_{\gamma,i}^{(t)} u_\gamma - \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times m_{i,l} \vec{\boldsymbol{a}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}, \tag{25}$$

where $a_{\gamma,i}^{(t)}$ denotes the accelerate speed value along coordinate q_γ at time t. According to Kane's equation, we have:

$$K_\gamma^{(t)} + K_\gamma^{*(t)} = 0. \tag{26}$$

With Equation 24, 25, and 26, we have:

$$\sum_{i=1}^{n} F_{\gamma,i}^{(t)} u_\gamma + \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} - \sum_{i=1}^{n} M_i a_{\gamma,i}^{(t)} u_\gamma - \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times m_{i,l} \vec{\boldsymbol{a}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} = 0. \tag{27}$$

As the prediction model $g(\cdot)$ has assumed to be able to accurately predict the state of $\mathcal{S}$ in the Theorem 3.1, we have:

$$\sum_{i=1}^{n} F_{\gamma,i}^{(t)} u_\gamma - \sum_i M_i \frac{1}{\delta} \big(g(G^{(t)})_{[v_\gamma,i]} - v_\gamma^{(t)} \big) u_\gamma + \Psi^{(t)} = 0. \tag{28}$$

$\Psi^{(t)}$ is regarded as a disturbance term, formally:

$$\Psi^{(t)} = \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times \boldsymbol{F}_{i,l}^{(t)}) \cdot \boldsymbol{\omega}_{i,\gamma}^{(t)} - \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} (\vec{\boldsymbol{r}}_{i,l} \times m_{i,l} \boldsymbol{a}_{i,l}^{(t)}) \cdot \boldsymbol{\omega}_{i,\gamma}^{(t)} \tag{29}$$

From Equation 29, it can be easily conclude that:

$$\Psi^{(t)} \propto \sum_{i=1}^{n} \sum_{l=1}^{\widetilde{m}_i} |\vec{\boldsymbol{r}}_{i,l}|. \tag{30}$$

So far, we have demonstrated Equation 11, along with the property of $\Psi^{(t)}$.

A.2. The Proof of Theorem 3.2

Based on Equation 29, we have:

$$\Psi^{(t)} = \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \boldsymbol{\omega}_{i,\gamma}^{(t)} - \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times m_{i,l}\vec{\boldsymbol{a}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)}. \tag{31}$$

Within Equation 31, the first term of Ψ can be derived as follows:

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} \leq \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(|\boldsymbol{r}_{i,l}||\vec{\boldsymbol{F}}_{i,l}^{(t)}|sin(\theta_{i,l}^{rF}))\omega_\gamma \\
\leq \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(|\boldsymbol{r}_{i,\arg\max_l(|\boldsymbol{r}_{i,l}|)}||\vec{\boldsymbol{F}}_{i,l}^{(t)}|sin(\theta_{i,l}^{rF}))\omega_\gamma &= \sum_{i=1}^{n}|\boldsymbol{r}_{i,\arg\max_l(|\boldsymbol{r}_{i,l}|)}|\sum_{l=1}^{\widetilde{m}_i}(|\vec{\boldsymbol{F}}_{i,l}^{(t)}|sin(\theta_{i,l}^{rF}))\omega_\gamma \\
&\leq \boldsymbol{r}_{\arg\max_i|\boldsymbol{r}_{i,\arg\max_l(|\boldsymbol{r}_{i,l}|)}|,\arg\max_l(|\boldsymbol{r}_{i,l}|)}\sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(|\vec{\boldsymbol{F}}_{i,l}^{(t)}|)sin(\theta_{i,l}^{rF}))\omega_\gamma,
\end{aligned} \tag{32}$$

where $\boldsymbol{r}_{\arg\max_i|\boldsymbol{r}_{i,\arg\max_l(|\boldsymbol{r}_{i,l}|)}|,\arg\max_l(|\boldsymbol{r}_{i,l}|)}$ is the max radius τ among all elements, $\theta_{i,l}^{rF}$ denotes the angle between $\boldsymbol{r}_{i,l}$ and $\boldsymbol{F}_{i,l}^{(t)}$. The above expression can be further derived as follows:

$$\tau\sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(|\vec{\boldsymbol{F}}_{i,l}^{(t)}|sin(\theta_{i,l}^{rF}))\omega_\gamma = \tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|sin(\theta_{i,l}^{RF})) \leq \tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma, \tag{33}$$

where θ_i^{RF} denote the angle between $\vec{\boldsymbol{R}}_i^{(t)}$ and $\boldsymbol{r}_{i,l}$. Using the same method, we can also determine the lower bound of $\sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)}$, and thus we can obtain:

$$-\tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma \leq \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times \vec{\boldsymbol{F}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} \leq \tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma. \tag{34}$$

Meanwhile, the second term of Equation 31 can be derived as follows:

$$\sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times m_{i,l}\vec{\boldsymbol{a}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} \leq \sum_{i=1}^{n} I_i\dot{\omega}_i^{(t)}, \tag{35}$$

where I_i denotes the rotational inertia of element c_i. However, calculating the rotational inertia of irregular objects can be quite challenging, and this situation may arise with the elements we have divided. Therefore, we further derive Expressing 35 as follows:

$$\sum_{i=1}^{n} I_i\dot{\omega}_i^{(t)} \leq \sum_{i=1}^{n}\frac{2}{5}m_i\tau^2\dot{\omega}_i^{(t)}, \tag{36}$$

where $\frac{2}{5}m_i\tau^2$ be the rotational inertia of a sphere that contains element c_i. Based on Equation 35 and 36, we have:

$$-\sum_{i=1}^{n}\frac{2}{5}m_i\tau^2\dot{\omega}_i^{(t)} \leq \sum_{i=1}^{n}\sum_{l=1}^{\widetilde{m}_i}(\boldsymbol{r}_{i,l} \times m_{i,l}\vec{\boldsymbol{a}}_{i,l}^{(t)}) \cdot \vec{\boldsymbol{\omega}}_{i,\gamma}^{(t)} \leq \sum_{i=1}^{n}\frac{2}{5}m_i\tau^2\dot{\omega}_i^{(t)}, \tag{37}$$

Therefore, we have:

$$-\tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma - \sum_{i=1}^{n}\frac{2}{5}m_i\tau^2\dot{\omega}_i^{(t)} \leq \Psi \leq \tau\sum_{i=1}^{n}(|\vec{\boldsymbol{R}}_i^{(t)}|)\omega_\gamma + \sum_{i=1}^{n}\frac{2}{5}m_i\tau^2\dot{\omega}_i^{(t)}, \tag{38}$$

The theorem is proved.

B. Experimental Details

B.1. Implementations

B.1.1. Edge Construction for Contact-Oriented Graph.

In this section, we introduce the construction of edges in our CONTACT-ORIENTED GRAPH. Many physical scenarios are complex, involving multiple object interactions (including collision and friction) when the distance between them is minimal. The interactions between particles belonging to different objects and those within the same object are often distinct. Hence, we propose the concept of a contact graph. We continuously observe the distances between particles. When the distance between particles falls below a certain threshold, and they do not belong to the same object, we determine that contact has occurred. A new edge is established between these contacting particles to facilitate message passing, thus forming a new, contact-related graph. This newly constructed graph is utilized to predict and generate the contact forces experienced by the objects. Additionally, we leverage the pooling of particle-level information from the objects themselves to facilitate interactions between different objects.

B.1.2. The Specific Calculation Method for Each Component Within $\mathcal{L}_{cst}$.

For the prediction of $F_{\gamma,i}^{(t)}$, it is essential to first clarify the number of independent velocities in our system. The rigid body system under study is a six-degree-of-freedom system, encompassing x, y, and z coordinates, as well as angular velocities along the x, y, and z axes. Nevertheless, given the complexity of acquiring rotational inertia related to angular velocity and the challenging nature of obtaining torque data, we limit our calculations to determining the magnitudes of force components associated with independent velocities along the x, y, and z directions. These force components can be directly obtained from the corresponding elements of the force vector $\boldsymbol{f}_i$ calculated in Section 3.2.2. In this way, $|\vec{\boldsymbol{R}}_i^{(t)}|$ can also be directly obtained by calculating the magnitude of the force vector $\boldsymbol{f}_i$. M_i can be obtained by calculating the volume of each element and multiplying it by the density.

We utilize the most commonly known laws of physics for solving angles issues. The angular velocity is derived by dividing the velocity difference of the current particle relative to the central particle by the distance between the current and the central particle. The formula is thus expressed as $\omega = \frac{v}{r}$, where $\vec{\boldsymbol{v}} = v_i(vec, present) - v_0(vec, center) = \sqrt{(v_{i,x} - v_{0,x})^2 + (v_{i,y} - v_{0,y})^2 + (v_{i,z} - v_{0,z})^2}$, $r = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2}$. Similarly, for the velocity component of each axis, $\omega_{x,y,z} = \frac{v_{x,y,z}}{r_{x,y,z}}$, where $\vec{\boldsymbol{v_x}} = v_{i,x} - v_{0,x}, \vec{\boldsymbol{v_y}} = v_{i,y} - v_{0,y}, \vec{\boldsymbol{v_z}} = v_{i,z} - v_{0,z}$; $r_x = x_i - x_0, r_y = y_i - y_0, r_z = z_i - z_0$. We will select the angular velocity component along the largest axis to use as our angular velocity, $\omega_{max} = Max(\omega_x, \omega_y, \omega_z)$.

B.1.3. Structural Constraint Module

In this study, we introduce a novel algorithm aimed at restoring deformed objects to their original, regular rigid body shapes. This algorithm is based on the geometric transformation of three-dimensional point sets, achieving the restoration of the object's shape by precisely calculating the optimal rotation and translation matrices. Specifically, the algorithm selects elements from the current and reference states of the object for processing. It calculates the centroid of each set and centralizes the data, shifting the coordinates of both sets to be centered around the origin. This step eliminates the influence of positional deviations. Subsequently, it computes the covariance matrix of the two centralized data sets and employs Singular Value Decomposition (SVD) to extract the rotation matrix. To ensure the correct rotational direction of the object, we introduce a correction matrix to adjust the rotation matrix. This rotation matrix not only reflects the directional differences between the two point sets but also ensures that the transformation adheres to the right-hand rule. Finally, by calculating the translation vector, the adjusted reference point set is moved to a position as close as possible to the current point set. Our experimental results indicate that this method can effectively restore deformed objects back to their original shapes.

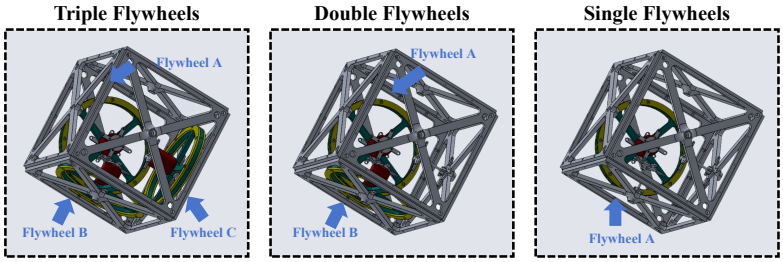
B.2. Hyperparameters and Settings

We adopt GCN as the backbone for all methods. Specifically, for LGNS and HGNS, we employ a hidden dimension of 200 for the node update function and 300 for message computation, with each hidden dimension comprising 3 layers and utilizing Squareplus as the activation function. Due to its multi-stage hierarchical modeling, we use an Adam optimizer with an initial learning rate of 0.001 and beta values of (0.9, 0.999), along with a patience-based Plateau scheduler having 3 cycles

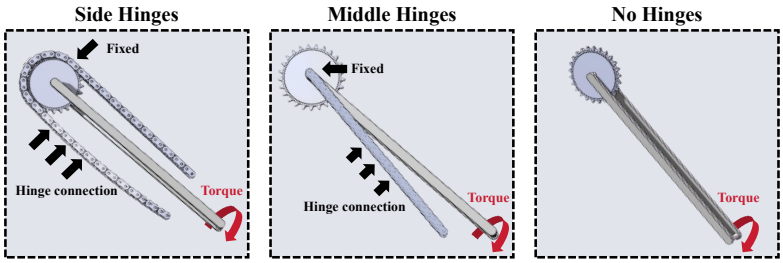
and a decay factor of 0.8. However, we still make very minor modifications based on the aforementioned hyperparameters. For SGNN, GNS, and MDGS, we employ SiLu or ReLU as the activation function and uniformly use a hidden dimension of 200, with early stopping after 10 epochs. Given the larger size of each system, we use a batch size of 1 for each dataset. Furthermore, we inject noise during training to achieve better long-term prediction at test time, with the noise ratio set to 4e-4 in Cubli, and 0.06 times the standard deviation in the other dataset. The cut-off radius r for both datasets is set to 0.001. On these two datasets, we only use the state information of the last frame t as input to predict the information of frame t + 1. Experiments are conducted on a single NVIDIA RTX A6000 GPU.

B.3. Physical Systems Used in The Experiments

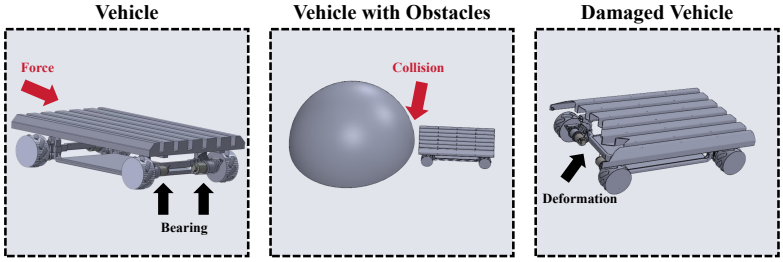
Figure 11 shows the detailed structures of all physical systems utilized in the experiments.



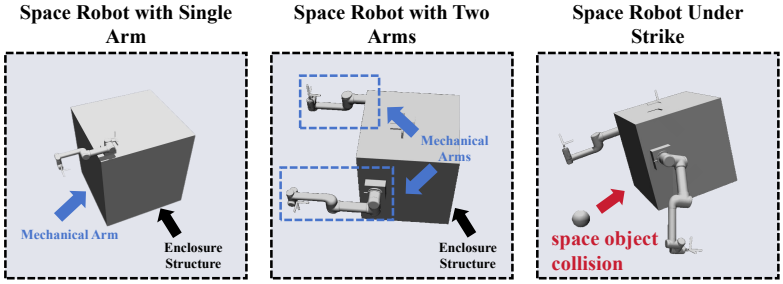
(a) Cubli Robot



(b) Complex Rotational Body



(c) Vehicle



(d) Space Robot

Figure 11. The physical systems used in all twelve datasets.

B.4. Further Results

Tables 1, 2, and 3 present the digital representation of the conducted experiments, to more clearly display the results.

Table 1. Comparative experiment results. **Bold** indicates the method with optimal performance. Underline denotes the method with second-best performance. Some standard deviations are marked as 0.00 due to being too small to effectively represent.

Senario		Cubli Robot			Complex Rotational Body		
Time	Methods	Single Flywheel	Double Flywheels	Triple Flywheels	No Hinge	Side Hinges	Center Hinge
t=20 (MSE $\times 10^0$)	GNS	4.47±0.39	2.08±0.17	3.95±0.74	20.81±3.14	4.09±0.82	3.31±0.96
	LGNS	14.05±0.43	31.30±6.80	8.52±1.05	1.42±0.63	3.33±0.86	2.16±1.15
	SGNN	3.22±0.39	47.76±3.55	36.27±2.76	29.23±5.29	23.86±0.38	3.49±0.47
	HGNS	2.08±0.69	97.02±7.14	16.67±0.81	33.61±6.33	2.55±0.37	12.18±2.05
	MDGS (Ours)	**0.84±0.28**	**1.76±0.63**	**2.08±0.24**	**0.84±0.04**	**0.58±0.03**	**0.92±0.15**
t=40 (MSE $\times 10^1$)	GNS	0.89±0.17	0.32 ±0.26	0.75±0.05	7.77±1.79	0.80±0.09	0.65±0.19
	LGNS	2.80±1.08	11.00±1.73	3.12±0.26	0.27±0.01	0.65±0.22	0.42±0.22
	SGNN	0.54±0.32	17.24±1.12	13.09±0.38	10.65±0.21	8.73±1.06	1.31±0.83
	HGNS	0.33±0.13	36.51±2.29	6.24±0.27	12.84±0.52	0.50±0.09	2.37±0.98
	MDGS (Ours)	**0.17±0.08**	**0.24 ±0.06**	**0.31±0.03**	**0.15±0.04**	**0.11±0.08**	**0.18±0.04**
t=110 (MSE $\times 10^2$)	GNS	0.25±0.11	0.14±0.02	0.22±0.01	5.71±2.37	0.22±0.04	0.17±0.03
	LGNS	0.77±0.23	7.61±0.92	2.30±1.13	0.07±0.01	0.18±0.03	0.11±0.02
	SGNN	0.12±0.07	12.20±0.37	8.73±0.79	15.27±1.21	9.54±1.06	0.96±0.21
	HGNS	0.09±0.01	26.79±0.44	4.61±0.35	10.13±0.78	0.14±0.01	0.64±0.39
	MDGS (Ours)	**0.05±0.01**	**0.06±0.01**	**0.09±0.01**	**0.04±0.01**	**0.03±0.00**	**0.05±0.00**

Table 2. Generalization experiment results. Δ indicates the disparity in model performance between the test set with changes and the original test set. **Bold** indicates the method with optimal performance. Underline denotes the method with second-best performance.

Time	Methods	More Intense Motion	Δ	Less Intense Motion	Δ	Shape Varied	Δ	Density Varied	Δ
t=20 (MSE $\times 10^0$)	GNS	115.61±18.08	+113.53	5.41±0.27	+3.33	5.01±0.62	+1.06	8.17±3.57	+4.22
	LGNS	188.71±28.27	+157.41	40.79±13.93	+9.49	21.77±0.83	+13.25	30.72±9.43	+22.20
	SGNN	614.13±31.04	566.36	97.02±49.25	+49.26	57.53±1.02	+21.26	61.92±48.41	+25.65
	HGNS	216.76±47.45	+119.74	197.58±61.12	+100.56	40.53±1.02	+23.86	120.01±34.54	+103.34
	MDGS (Ours)	**7.91± 2.85**	**+6.14**	**4.47±0.38**	**+2.71**	**2.13±0.26**	**+0.05**	**2.63±0.58**	**+0.55**
t=40 (MSE $\times 10^1$)	GNS	22.51±3.15	+22.19	1.00±0.15	+0.68	1.03±0.13	+0.28	1.55±0.72	+0.79
	LGNS	73.64±2.51	+62.63	15.62±3.44	+4.61	8.22±1.21	+5.10	11.05±1.98	+7.93
	SGNN	211.42±19.05	+194.18	36.51±6.29	+19.26	20.61±3.53	+7.52	22.02±3.08	+8.93
	HGNS	78.16±16.71	+41.66	70.53±21.02	+33.54	14.68±2.55	+8.44	23.37±2.88	+17.13
	MDGS (Ours)	**1.57±0.56**	**+1.32**	**0.90±0.19**	**+0.64**	**0.33±0.03**	**+0.02**	**0.36±0.15**	**+0.05**
t=110 (MSE $\times 10^2$)	GNS	6.08±0.05	+5.97	0.36±0.14	+0.22	0.31±0.03	+0.09	0.48±0.26	+0.26
	LGNS	58.33±4.91	+50.72	11.60±0.15	+3.97	6.41±1.32	+4.11	7.25±0.16	+4.95
	SGNN	124.04±11.04	+111.83	26.80±2.47	+1.45	13.11±2.37	+4.38	13.67±0.02	+4.94
	HGNS	54.40±7.15	+28.61	50.86±8.32	+24.07	9.76±0.98	+5.15	6.37±0.24	+1.76
	MDGS (Ours)	**0.40±0.15**	**+0.33**	**0.25±0.02**	**+0.19**	**0.10±0.01**	**+0.01**	**0.11±0.02**	**+0.02**

Table 3. Ablation experiments compared with other baselines.

Methods	Cubli Robot		Complex Rotational Body	
	t=40	t=110	t=40	t=110
GNS	7.52±0.49	22.36±1.21	8.01±0.69	21.70±3.71
LGNS	31.2±2.66	230.09±31.94	6.52±0.87	17.59±2.52
SGNN	130.95±3.77	873.55±39.42	87.38±7.26	954.46±60.82
HGNS	62.44±2.68	461.5±35.41	4.98±0.19	13.53±1.07
MDGS-NC	110.84±12.47	763.36±1.46	8.40±0.33	60.15±7.05
MDGS-SC	15.24±5.88	43.47±18.94	3.32± 0.36	20.85±3.27
MDGS	3.11±0.32	9.59±1.27	1.15±0.31	3.04±0.64