2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Graph Neural Model Predictive Control for High-Dimensional Systems

Patrick Benito Eberhard[1], Luis Pabon[2], Daniele Gammelli[2], Hugo Buurmeijer[2], Amon Lahr[1], Mark Leone[2], Andrea Carron[1], Marco Pavone[2,3]

***Abstract*— The control of high-dimensional systems, such as soft robots, requires models that faithfully capture complex dynamics while remaining computationally tractable. This work presents a framework that integrates Graph Neural Network (GNN)-based dynamics models with structure-exploiting Model Predictive Control to enable real-time control of high-dimensional systems. By representing the system as a graph with localized interactions, the GNN preserves sparsity, while a tailored condensing algorithm eliminates state variables from the control problem, ensuring efficient computation. The complexity of our condensing algorithm scales linearly with the number of system nodes, and leverages Graphics Processing Unit (GPU) parallelization to achieve real-time performance. The proposed approach is validated in simulation and experimentally on a physical soft robotic trunk. Results show that our method scales to systems with up to 1,000 nodes at 100 Hz in closed-loop, and demonstrates real-time reference tracking on hardware with sub-centimeter accuracy, outperforming baselines by 63.6%. Finally, we show the capability of our method to achieve effective full-body obstacle avoidance.**

Website: https://gnn-mpc.github.io/

I. Introduction

Robotic systems with high Degrees of Freedoms (DoFs) offer exceptional versatility and adaptability. Continuum soft robots exemplify this potential whereby their compliant and elastic structures naturally conform to complex surfaces and objects. This intrinsic compliance not only enhances physical robustness but also makes soft robots particularly well-suited for operation in delicate, constrained environments. Applications include safe locomotion, manipulation, and medical procedures [1].

However, the features that provide these advantages also pose significant challenges for modeling and control. Accurately capturing their nonlinear dynamics and deformations demands complex, high-dimensional models. Existing approaches attempt to address this by either employing physics-based models [2] or data-driven methods [3], [4]; yet, they face an inherent tradeoff between model fidelity and computational tractability. Moreover, most modeling approaches restrict control to a subset of the state space for computational tractability, thereby restricting the ability to impose constraints or optimize performance across an extended set of points along a robot's structure. Ultimately, the full potential of soft-robot versatility and adaptability hinges on the effectiveness of the underlying control methods.

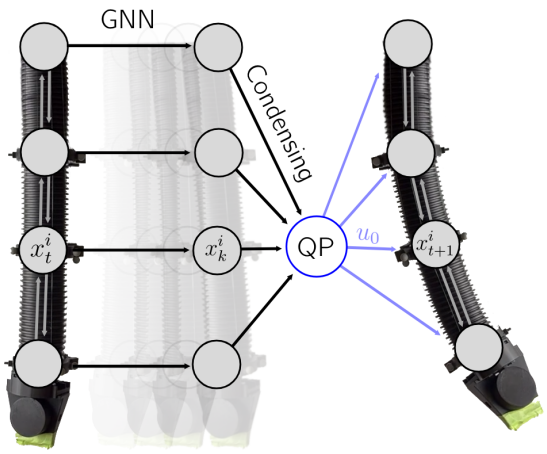


Fig. 1: A soft robotic trunk is modeled as a graph with nodes as discrete segments with state x_t^i at time t and edges as physical interactions. A GNN computes the linearized forward dynamics x_k^i, which are condensed into a Quadratic Program (QP) that only depends on the input variables u_k. The QP is then solved, and the first input u_0 is applied in a receding-horizon fashion.

Concurrently, Graph Neural Networks (GNNs) have emerged as powerful, data-driven models for representing complex dynamical systems, achieving substantial speedups over traditional numerical simulations [5]. GNNs operate on graph elements and naturally encode relational patterns inherent in high-dimensional robotic platforms, mirroring their sparse physical structure. In combination with Model Predictive Control (MPC) [6], they have the potential to enable real-time optimal control at scale. However, integrating GNNs into MPC remains largely underexplored, and current formulations do not leverage their sparse structure, leading to prohibitive solve times for large-scale problems.

Contributions: We propose a framework that, under the assumption of localized interactions, models dynamical systems as graphs and uses GNNs to learn structured, scalable representations of their dynamics. To enable real-time optimal control, we adapt existing *condensing* algorithms from [7] to eliminate the state variables from the resulting Optimal Control Problem (OCP), thus reducing its dimensionality without compromising model fidelity. This is achieved by leveraging the localized sparsity structure of the

[1]Institute for Dynamic Systems and Control, ETH Zürich, Zürich CH-8092, Switzerland (e-mail: {peberhard, amlahr, carrona}@ethz.ch).
[2]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA (e-mail: {lpabon, gammelli, hbuurmei, mleone, pavone}@stanford.edu).
[3]NVIDIA Research

linearized dynamics equations provided by the GNN, leading to linear scaling with respect to the number of graph nodes in the modeled system, provided that each node interacts with a bounded number of neighbors. We summarize our list of contributions below:

1) **GNN dynamics with structure-aware condensing for MPC:** We enable relational modeling of system dynamics and preserve structural sparsity, which we leverage to design a condensing algorithm with linear scaling properties in system size for a bounded number of neighbors. Figure 1 provides an intuitive overview of the approach.
2) **Highly efficient implementation:** We leverage parallel operations on the GPU to achieve near-constant computation times for systems with up to 1,000 nodes at 100 Hz control frequency.
3) **Experimental validation:** We demonstrate reduced trajectory tracking error on a physical soft robotic trunk by 63.6%, and showcase direct control over all observable system states.

Outline: We first provide a literature review (Sec. II), followed by a theoretical background on GNNs and condensing (Sec. III). We then provide a formal problem statement (Sec. IV) and detail the proposed GNN-MPC framework (Sec. V). Finally, we provide results from simulation and hardware experiments (Sec. VI) and present our conclusions (Sec. VII).

Notation: The set of real numbers and integers are represented by $\mathbb{R}$ and $\mathbb{Z}$, respectively. We denote the set of integers in the interval $[a, b]$ as $\mathbb{Z}_{[a,b]}$. For $Q \succeq 0$, we define the weighted squared seminorm $\|x\|_Q^2 \coloneqq x^\top Q x$.

II. Related Work

Direct modeling of soft robots typically relies on continuum mechanics and multibody dynamics. While 3D solid mechanics and the Finite Element Method (FEM) provide accurate representations [8], solving the resulting differential equations is computationally expensive, and accurate characterization of material properties remains challenging. To reduce computational cost, reduced-order models such as the piecewise-constant curvature model [2], [9] approximate a soft robot through constant-curvature segments, at the expense of failing to capture more complex dynamics. In addition, Koopman operator theory enables the construction of linear models for nonlinear systems by lifting the dynamics into a higher-dimensional space [3]. However, exact finite-dimensional linear representations of most physical systems are generally not available [10], and prediction performance depends on the quality of the finite-dimensional approximation. Model reduction can be further achieved through Spectral Submanifolds (SSMs) [11], which define low-dimensional attracting invariant manifolds that approximate the dominant nonlinear behaviors of high-dimensional systems. Recent work has shown that dynamics can be learned and controlled efficiently on these manifolds using data-driven approaches [4]. However, the use of SSMs often imposes stringent assumptions, such as moderate control magnitudes and asymptotic stability of the system's origin. Instead, we propose a complementary method that relaxes these assumptions and represents high-dimensional systems as sparse networks of interacting elements.

Beyond structured physics-based models, neural networks provide an alternative modeling approach for dynamical systems. Feedforward networks can learn linearized state-space models from observed dynamics [12], facilitating real-time control while lessening the associated modeling effort. Nevertheless, such neural networks might experience difficulties in generalizing the system's dynamics due to the absence of inductive biases, i.e. structural assumptions and prior knowledge which are reflected in the model architecture and the learning process. In contrast, Graph Neural Networks incorporate relational structure through local message-passing mechanisms to capture interactions between system components [5], [13]. Further developments have shown that GNNs can outperform standard convolutional networks in mesh-based simulations, fluid dynamics, and forward dynamics prediction [14]. GNNs have also been applied to online planning, including unconstrained trajectory optimization via gradient descent [15], [16], and have also shown robustness under partial observability [17]. Applications in robotics include soft robotic hands [18], where the relational inductive bias of GNNs enables reasoning over physical interactions more effectively than unstructured neural networks. GNNs have also demonstrated their value in policy learning [19], [20], showing that GNN-based policies exhibit greater transferability, generalization, and scalability in reinforcement learning tasks than non-relational approaches. Despite this growing body of work, the use of GNNs for efficient closed-loop optimal control of high-dimensional systems remains largely unexplored. Our work addresses this gap by combining GNNs with MPC for real-time, high-dimensional control.

A major limitation in using expressive models such as GNNs within MPC lies in the computational complexity of solving the resulting OCP using Newton-type methods, such as Sequential Quadratic Programming (SQP). Most sparsity-exploiting Interior Point Method (IPM) Quadratic Program (QP) solvers suffer from cubic scaling with respect to the state dimension [21]. To address this, various structure-exploiting algorithms have been developed, among which condensing stands out as a particularly relevant approach for our problem [7]. Condensing removes state variables from the OCP at the cost of an additional preprocessing step [21], [22], [23]. While it preserves problem structure and reduces solver times, its computational overhead scales unfavorably with system dimensionality. In this work, we leverage the inherent sparse structure of the GNNs to develop a condensing algorithm which supports computationally tractable MPC for high-dimensional systems.

III. Background

This section provides a brief overview of GNNs in III-A and condensing for MPC in III-B, which form the foundation of our proposed approach.

A. Graph Neural Networks

Graph Neural Network generalize neural architectures to graph-structured data. At their core, GNNs exploit the relational structure in the data by reusing a set of local learnable functions, typically defined on nodes and edges, and applying them across the entire graph through convolution-like operators. This design enforces permutation equivariance and, using the symmetries inherent in graphs, GNNs have achieved remarkable success across a wide range of disciplines [5], [14], [24]. To formalize this notion, we represent the network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes or vertices and directed edges, respectively, where each node $v_i \in \mathcal{V}$ encodes a local state $x^i \in \mathbb{R}^{\bar{n}_x}$ and each edge $(v_i, v_j) \in \mathcal{E}$ encodes an interaction $e^{ij} \in \mathbb{R}^{n_e}$ from node v_j to v_i. Various GNN architectures exist, typically classified into convolution, attention, and message-passing architectures [13].

In the message-passing framework [5], each node attribute x_k^i at layer k is updated by aggregating information from its neighbors as

$$x_{k+1}^i = \phi\Big(x_k^i, \bigoplus_{j \in \mathcal{N}_i} \psi(x_k^i, x_k^j, e_k^{ij})\Big), \tag{1}$$

where ψ and ϕ are learnable functions, $\oplus$ is a permutation-invariant aggregation operator (e.g., sum or mean pooling), and $\mathcal{N}_i \coloneqq \{v_j \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$ denotes the neighborhood of node $i \in \mathcal{M}$, where $\mathcal{M} \coloneqq \mathbb{Z}_{[1,M]}$ is the set of node indices with $M = |\mathcal{V}|$ nodes. This formulation mirrors physical laws, learning local node functions that aggregate neighboring interactions. This motivates our approach for modeling high-dimensional dynamics with GNNs, where forces are exchanged between system components.

B. Condensing in MPC

We consider a linear time-varying system with linear constraints and a quadratic cost function defined over a finite-horizon N[1], leading to the following OCP formulation:

$$\begin{aligned}
\min_{x_k, u_k} \quad & \sum_{k=0}^{N-1} \|x_k\|_{Q_k}^2 + \|u_k\|_{R_k}^2 + q_k^T x_k + r_k^T u_k && \text{(2a)}\\
& + \|x_N\|_{Q_N}^2 + q_N^T x_N && \text{(2b)}\\
\text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k + c_k, \quad k \in \mathbb{Z}_{[0,N-1]}, && \text{(2c)}\\
& x_k \in \mathbb{X}_k = \{x \mid C_k^x x \le d_k^x\}, \quad k \in \mathbb{Z}_{[0,N]}, && \text{(2d)}\\
& u_k \in \mathbb{U}_k = \{u \mid C_k^u u \le d_k^u\}, \quad k \in \mathbb{Z}_{[0,N-1]}, && \text{(2e)}\\
& x_0 = \tilde{x}(t), && \text{(2f)}
\end{aligned}$$

where $Q_k \succeq 0 \in \mathbb{R}^{\bar{n}_x \times \bar{n}_x}$, $R_k \succeq 0 \in \mathbb{R}^{n_u \times n_u}$ are cost matrices and $q_k \in \mathbb{R}^{\bar{n}_x}$, $r_k \in \mathbb{R}^{n_u}$ represent the linear terms of the stage cost. Equation (2c) describes the linear system dynamics, where the initial condition x_0 is set to $\tilde{x}(t)$, the measured state of the system at a given time t. The state and input constraints $\mathbb{X}_k, \mathbb{U}_k$ are time-varying linear half-space constraints. We define the augmented variables

$$x = [x_0^\top, \dots, x_N^\top]^\top, \quad u = [u_0^\top, \dots, u_{N-1}^\top]^\top,$$

and the corresponding block-diagonal matrices and vectors $\bar{Q}, \bar{q}, \bar{C}_x, \bar{d}_x$ (over $k = 0, \dots, N$), and $\bar{R}, \bar{r}, \bar{C}_u, \bar{d}_u$ (over $k = 0, \dots, N-1$) as the stacked sequences of the stage matrices and vectors. The original problem can be reformulated by *condensing*, i.e. eliminating state variables through forward substitution [7], [22]. This reduces the OCP into a QP in the input sequence alone, with Nn_u optimization variables, i.e.,

$$\min_u \; u^\top H u + g^\top u, \quad \text{s.t. } \tilde{C} u \le \tilde{d}, \tag{3}$$

with

$$H = \Gamma_u^\top \bar{Q} \Gamma_u + \bar{R}, \quad g = \Gamma_u^\top \bar{Q} \Gamma_x + \Gamma_u^\top \bar{q} + \bar{r}, \tag{4a}$$

$$\tilde{C} = \begin{bmatrix} \bar{C}_u \\ \bar{C}_x \Gamma_u \end{bmatrix}, \quad \tilde{d} = \begin{bmatrix} \bar{d}_u \\ \bar{d}_x - \bar{C}_x \Gamma_x \end{bmatrix}. \tag{4b}$$

The matrices $\Gamma_u \in \mathbb{R}^{Nn_u \times Nn_u}$ and $\Gamma_x \in \mathbb{R}^{Nn_x}$ depend on the system dynamics matrices A_k, B_k, c_k, and are computed recursively, satisfying $x = \Gamma_u u + \Gamma_x$[2]. Condensing reduces the number of decision variables to be solved for, yet this benefit comes at the expense of at least $\mathcal{O}(n_x^2)$ additional floating point operations per solver iteration [7].

IV. Problem Formulation

We now introduce our assumptions on the system dynamics and formally define the control task.

A. Local System Dynamics

We consider a discrete-time nonlinear system which can be described as a composition of M subsystems, where each subsystem $i \in \mathcal{M}$ has a local state $x^i \in \mathbb{R}^{\bar{n}_x}$ and receives a global external input $u_k \in \mathbb{R}^{n_u}$. The full system state is given by $x_t = [(x_t^1)^\top, \dots, (x_t^M)^\top]^\top \in \mathbb{R}^{n_x}$ with $n_x = M\bar{n}_x$. Motivated by the typically uniform material properties throughout the continuum of a soft robot, we make the following assumption:

Assumption 1. *The forward dynamics of the overall state $x_{t+1} = \mathcal{F}(x_t, u_t)$ can be expressed as*

$$x_{t+1}^i = f(x_t^i, \{x_t^j\}_{j \in \mathcal{N}_i}, u_t), \quad i \in \mathcal{M}, \tag{5}$$

where $\mathcal{N}_i \subseteq \mathcal{M}$ is the set of subsystems affecting subsystem i, and $f : \mathbb{R}^{\bar{n}_x} \times \mathbb{R}^{|\mathcal{N}_i|\bar{n}_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{\bar{n}_x}$ is homogeneous across all subsystems and almost everywhere differentiable.

We further assume that each subsystem interacts with a small and bounded neighborhood, based on the local behavior of each material point which depends on the immediate surroundings it is in contact with, i.e.,

Assumption 2. *There exists a small bound d such that*

$$|\mathcal{N}_i| \le d, \quad i \in \mathcal{M}. \tag{6}$$

Throughout, we consider systems where the overall state dimension of the system is much larger than the number of control inputs, and we consider that each subsystem interacts with a small neighborhood, i.e.,

$$n_u \ll n_x, \quad d \ll M. \tag{7}$$

[1]Section V describes how the nonlinear OCP is solved with SQP by a linear approximation.

[2]See Section V-B for a detailed derivation.

Continuum soft robots naturally satisfy the assumptions outlined above. Moreover, in typical soft robotic systems, the number of actuators is significantly smaller than the number of states required to fully describe the robot's configuration, motivating the use of condensing to solve a QP in the input variables only.

B. Control Task

Our objective is the tracking of reference trajectories $\{x_t^r, u_t^r\}_{t\geq 0}$ across all subsystems $i \in \mathcal{M}$, subject to the state and input constraints that govern the overall system. At each time t, we solve the following OCP with horizon N:

$$\min_{x_k, u_k} \quad \sum_{i=1}^{M} \left(\sum_{k=0}^{N} \left(\|x_k^i - x_k^{i,r}\|_{Q_k^i}^2 \right) \right) + \sum_{k=0}^{N-1} \|u_k - u_k^r\|_{R_k}^2 \tag{8a}$$

$$\text{s.t.} \quad x_0^i = \tilde{x}^i(t), \tag{8b}$$

$$x_{k+1}^i = f(x_k^i, \{x_k^j\}_{j\in\mathcal{N}_i}, u_k) \tag{8c}$$

$$x_k^i \in \mathbb{X}_k^i, \quad x_N^i \in \mathbb{X}_N^i, \quad u_k \in \mathbb{U}_k, \tag{8d}$$

$$k \in \mathbb{Z}_{[0,N-1]}, \quad i \in \mathcal{M}, \tag{8e}$$

where $\tilde{x}^i(t)$ is the measured state of subsystem i at time t, $Q_k^i \in \mathbb{R}^{\bar{n}_x \times \bar{n}_x}$, $Q_k^i \succeq 0$ is the cost matrix applied to subsystem i at stage k, $R_k \in \mathbb{R}^{n_u \times n_u}$, $R_k \succeq 0$ is the cost applied to the input at stage k, $\mathbb{X}_k^i = \left\{ x \in \mathbb{R}^{\bar{n}_x} \mid C_k^{x,i} x \leq d_k^{x,i} \right\}$ is a convex set of half-space constraints defined for subsystem i at time step k, and $\mathbb{U}_k = \{u \in \mathbb{R}^{n_u} \mid C_k^u u \leq d_k^u\}$ defines the admissible control inputs at time step k.

V. Graph Neural Model Predictive Control

We now introduce our proposed method for the tractable control of high-dimensional systems through GNNs and condensing. We first detail the modeling of the system dynamics, and then derive a condensing method which yields linear scaling in the number of subsystems M [3].

A. Modeling Dynamics with GNNs

Model: We represent the forward dynamics of the system as a second-order model, where each subsystem is characterized by its position and velocity, i.e. $x_k^i = [(p_k^i)^\top, (v_k^i)^\top]^\top$. We employ a GNN based on the interaction network [5], where nodes correspond to subsystems, edges encode their pairwise interactions, and we model a global input vector representing the control inputs that influence all subsystems simultaneously. We let the GNN predict a velocity increment, which is integrated via a backward Euler scheme, such that the state x_{k+1}^i is computed for each subsystem i as

$$p_{k+1}^i = p_k^i + v_{k+1}^i \Delta t, \tag{9a}$$

$$v_{k+1}^i = v_k^i + \phi \left(x_k^i, \sum_{j\in\mathcal{N}_i} \psi(e_k^{ij}), u_k \right), \tag{9b}$$

where Δt is the sampling period, v denotes velocity in this context, and ψ and ϕ are the message-passing and node-update functions, respectively. The edge feature e_k^{ij} is computed as the relative state difference $e_k^{ij} = x_k^i - x_k^j$, enabling the network to capture interactions in local coordinates. The functions ψ and ϕ represent nonlinear transformations that are implemented as Multi-Layer Perceptrons (MLPs) with Rectified Linear Unit (ReLU) activations. Further, neighborhood aggregation is performed via summation, motivated by how forces accumulate in physical systems.

Dataset Generation: The GNN learns the dynamics from open-loop trajectories. We construct a dataset of N_d elements composed of the measured states and applied control inputs at each time step, and the corresponding subsequent state:

$$\mathcal{D} = \{(x_t, u_t, x_{t+1})\}_{t=1,\dots,N_d}. \tag{10}$$

Measurements are collected at equidistant intervals Δt, equal to the intended sampling period of the MPC controller.

Training: Consequently, the dynamics are learned by minimizing the mean-squared-error loss function with ℓ_2 regularization:

$$\mathcal{L}_\theta(\mathcal{D}) \coloneqq \frac{1}{N_d} \sum_{t=1}^{N_d} \left\| x_{t+1} - \mathcal{F}_\theta(x_t, u_t) \right\|_O^2 + \lambda \|\theta\|^2, \tag{11}$$

where $\theta = (\theta_\phi, \theta_\psi)$ are the network parameters, $\mathcal{F}_\theta$ denotes the GNN operating on the full graph $\mathcal{G}$, $O \succeq 0$ is a weighting matrix, and λ is a regularization coefficient. We conduct a grid search over the hyperparameters to optimize predictive performance and train the models using the Adam optimizer [25] with a learning rate scheduler based on patience.

Linearization: To solve the resulting nonlinear MPC problem with a Newton-type method, we require a linear approximation of the GNN dynamics around a nominal trajectory $\{\hat{x}_k, \hat{u}_k\}$, which we obtain via automatic differentiation of each subsystem's local dynamics independently[4]. The linearized dynamics of each subsystem i at stage k are given by

$$x_{k+1}^i = A_k^{ii} x_k^i + \sum_{j\in\mathcal{N}_i} A_k^{ij} x_k^j + B_k^i u_k + c_k^i, \tag{12}$$

with $A_k^{ij} = \frac{\partial f_k^i}{\partial x_k^j}$, $B_k^i = \frac{\partial f_k^i}{\partial u_k}$, and $f_k^i = f(x_k^i, \{x_k^j\}_{j\in\mathcal{N}_i}, u_k)$. The term c_k^i is the resulting constant offset from linearization.

B. Local-Scalable Condensing

In this section, we present a tailored condensing algorithm for high-dimensional systems. Given the small neighborhood size d, the linearized dynamics of the overall system has a sparse structure, which is enforced by the GNN. Therefore, we derive a condensing algorithm based on [7][5] that takes advantage of the local dynamics and their resulting sparsity,

[3]Assuming a fixed horizon N, bounded neighborhood d, constant state dimension per node $\bar{n}_x$, and input dimension n_u.

[4]At the non-differentiable point of the ReLU activation ($x = 0$), a subgradient of zero is used.

[5]Various condensing algorithms exist in the literature. We adopt one that admits a local, per-subsystem decomposition.

resulting in favorable scaling properties formalized in the following theorem.

Theorem 1. *Let Assumptions 1 and 2 hold and consider the OCP in* (8)*. Then, the condensing procedure in Section III-B scales linearly in the number of subsystems M, i.e. the computation of $\Gamma_u, \Gamma_x, H, g, \tilde{C}, \tilde{d}$ requires $\mathcal{O}(M)$ floating point operations and memory for fixed values of $\bar{n}_x, n_u, N, d$.*

Proof. Consider the system dynamics in Equation (12) and a horizon $N = 1$. We can equivalently describe the state evolution of subsystem i as

$$x^i = \begin{bmatrix} x_0^i \\ x_1^i \end{bmatrix} = \Gamma_u^i u_0 + \Gamma_x^i, \tag{13}$$

with

$$\Gamma_u^i = \begin{bmatrix} 0 \\ B_0^i \end{bmatrix}, \Gamma_x^i = \begin{bmatrix} x_0^i \\ \sum_{j \in \tilde{\mathcal{N}}_i} A_0^{ij} x_0^j + c_0 \end{bmatrix}, \tag{14}$$

where the matrices Γ_u^i and Γ_x^i only depend on the local neighborhood $\tilde{\mathcal{N}}_i := \mathcal{N}_i \cup \{i\}$. For larger horizons, each state x_{k+1}^i is obtained recursively from the previous state x_k^i based on (12), and therefore we can compute Γ_u^i and Γ_x^i recursively as

$$(\Gamma_x^i)_{n+1} = \begin{bmatrix} (\Gamma_x^i)_n \\ \sum_{j \in \tilde{\mathcal{N}}_i} A_n^{ij} \mathcal{E}_n (\Gamma_x^j)_n + c_n^i \end{bmatrix}, \tag{15}$$

where $(\Gamma_x^i)_n \in \mathbb{R}^{\bar{n}_x (n+1)}$. Further, $\mathcal{E}_n = [0_{\bar{n}_x \times \bar{n}_x n} \; I_{\bar{n}_x}]$ selects the last row of the matrix it operates on, and we denote the recursion index by the subscript $n = 1, \dots, N-1$. Similarly,

$$(\Gamma_u^i)_{n+1} = \begin{bmatrix} (\Gamma_u^i)_n \\ \sum_{j \in \tilde{\mathcal{N}}_i} A_n^{ij} \mathcal{E}_n (\Gamma_u^j)_n \quad B_n^i \end{bmatrix}, \tag{16}$$

where $(\Gamma_u^i)_n \in \mathbb{R}^{\bar{n}_x (n+1) \times n_u n}$, and the recursion terminates with $\Gamma_x^i = (\Gamma_x^i)_N$, $\Gamma_u^i = (\Gamma_u^i)_N$. Under Assumption 2, the computation of Γ_x^i and Γ_u^i depends only on d neighboring nodes and is therefore independent of the overall system size M, which enables a distributed evaluation within each iteration.

Moreover, the cost function in (8) features separability for each subsystem i, i.e. the individual cost terms are aggregated, and we can compute the Hessian H independently for each subsystem i as

$$H = \sum_{i=1}^{M} H^i + \bar{R}, \quad H^i = (\Gamma_u^i)^\top \bar{Q}^i \Gamma_u^i, \tag{17}$$

where the local Hessians H^i are added to construct the full Hessian. Similarly, the gradient g can be decomposed as

$$g = \sum_{i=1}^{M} g^i + \bar{r}, \quad g^i = \left(\Gamma_u^i\right)^\top \left(\bar{Q}^i \Gamma_x^i + \bar{q}^i\right), \tag{18}$$

where each subsystem contributes g^i. Therefore, the condensing procedure can be performed with M distributed operations, where each subsystem computes its local contribution to the condensed QP, which is aggregated in a final step. Finally, following the separability of the constraints in (8), the latter can be processed independently and concatenated:

$$\tilde{C} = \begin{bmatrix} \bar{C}_u \\ \bar{C}_x^1 \Gamma_u^1 \\ \vdots \\ \bar{C}_x^M \Gamma_u^M \end{bmatrix}, \quad \tilde{d} = \begin{bmatrix} \bar{d}_u \\ \bar{d}_x^1 - \bar{C}_x^1 \Gamma_x^1 \\ \vdots \\ \bar{d}_x^M - \bar{C}_x^M \Gamma_x^M \end{bmatrix}, \tag{19}$$

and we denote the corresponding entries of subsystem i as $\tilde{C}^i, \tilde{d}^i$. Therefore, the overall computational complexity grows linearly with the number of nodes in the system, i.e., it requires $\mathcal{O}(M)$ floating point operations for fixed values of $\bar{n}_x, n_u, N, d$. Since each subsystem stores only fixed-size quantities, the total memory requirement also scales $\mathcal{O}(M)$. ∎

The previous proof provides an efficient and distributed computation scheme to condense the quantities in (4) by leveraging the sparse local graph dynamics. In addition, the computations are naturally suited for GPU parallelization, since they are carried out for each node and combined efficiently through summation or stacking. This can significantly reduce solve times, as will be demonstrated in Section VI.

C. GNN-MPC Algorithm

The general procedure is summarized in Algorithm 1. At each time step, the dynamics are linearized around $\{\hat{x}_k, \hat{u}_k\}$ as part of an SQP scheme, and the condensed QP is constructed by aggregating the contributions from each subsystem. Consequently, it is then solved using an IPM QP solver, and the first control input is applied to the system. The linearization trajectory is updated with the shifted optimal solution and the process is repeated.

Algorithm 1 GNN-MPC

Input: $\{x_k^r, u_k^r\}$, Q_k^i, R_k, N, $\mathbb{X}_k^i, \mathbb{U}_k$, GNN
Initialize: $\{\hat{x}_k, \hat{u}_k\} \leftarrow \{x(t), 0\}$ {*Set lin. trajectory*}
for $t = 0, \Delta t, \dots$ **do**
  Set $x_0 \leftarrow \tilde{x}(t)$
  for $k = 0, \dots, N-1$ **do**
    for $i = 1, \dots, M$ **do**
      Compute A_k^{ij}, B_k^i, c_k^i around $\{\hat{x}_k, \hat{u}_k\}$
      for $j \in \tilde{\mathcal{N}}_i$ {*Linearize GNN dynamics*}
    end
  end
  for $i = 1, \dots, M$ **do**
    Compute $\Gamma_u^i, \Gamma_x^i, H^i, g^i, \tilde{C}^i, \tilde{d}^i$ {*Condense*}
  end
  Compute $H, g, \tilde{C}, \tilde{d}$ {*Aggregate*}
  $\min_u u^\top H u + g^\top u$, s.t. $\tilde{C} u \leq \tilde{d}$ {*Solve QP*}
  $\{\hat{x}_k, \hat{u}_k\} \leftarrow \{x_{k+1}^\star, u_{k+1}^\star\}$ {*Update lin. trajectory*}
  Apply $u_0^\star$ to system
end

VI. Experimental Results

This section details the experimental validation of the proposed GNN-MPC algorithm. Following a description of

the experimental setup, we present simulation and hardware results designed to quantify the model's open-loop prediction accuracy, assess its performance in closed-loop control, and demonstrate scalability, enabling new applications for high-dimensional systems.

A. Setup

We trained the message-passing GNN in *PyTorch Geometric*, and translated it to *Flax* to enable Just-In-Time (JIT) compilation and efficient batching. The condensing algorithm was implemented in *JAX*, leveraging vectorized mapping and *einsum*-based primitives for efficient GPU execution via Accelerated Linear Algebra (XLA). The resulting QP is solved using the IPM solver *HPIPM* [21] within a SQP Real-Time-Iteration (RTI) scheme [26].

The experiments were conducted on an AMD Ryzen 9 7950X3D Central Processing Unit (CPU) and an NVIDIA RTX 4090 GPU. The simulation models a tendon-driven trunk robot in MuJoCo. Hardware experiments use the corresponding physical platform, tracked via an *OptiTrack* motion capture system at 100 Hz and actuated by 6 XM540 Dynamixel motors. The software is implemented in ROS2.

B. Simulation experiments

This section first compares open-loop predictions of our approach with four baseline models, and then demonstrates the scalability of our method.

Open-loop Predictions: We first evaluate the open-loop prediction accuracy of the learned GNN dynamics model against a Koopman operator model [3], SSM reduction with orthogonal projections [4], SSM with optimal linear projections [27], and a standard MLP baseline. All models are trained on the same dataset distribution from the trunk simulator, consisting of quasi-periodic input trajectories with varying amplitudes and periods. Table I summarizes the average end effector Root Mean Square Error (RMSE) on a one second horizon across 100 different trajectories. The GNN achieves open-loop RMSE comparable to the Koopman method while surpassing all other baselines. As observed in [4], the Koopman operator performs better on certain simulation trajectories, supporting our results.

TABLE I: Average end-effector open-loop RMSE across models, reported as mean $\pm 1\sigma$ standard deviation.

Model	**RMSE [m]**
Koopman	$(1.16 \pm 0.61) \times 10^{-3}$
GNN	$(1.38 \pm 0.76) \times 10^{-3}$
SSM (Opt.)	$(2.88 \pm 1.25) \times 10^{-3}$
MLP	$(3.59 \pm 1.67) \times 10^{-3}$
SSM (Orth.)	$(3.67 \pm 2.22) \times 10^{-3}$

Scalability: Further, we evaluate the scalability of the proposed GNN-MPC framework by varying the number of segments in the simulated trunk robot from 1 to 1000. The MPC problem is solved over a horizon of $N = 20$ with input constraints and state constraints on the end effector, constituting a fixed subset of nodes as in Theorem 1. Figure 2 shows the average solve time over 100 runs for each robot configuration. The GNN-MPC exhibits sub-linear scaling with respect to the number of segments considered, enabling real-time control at 100 Hz.

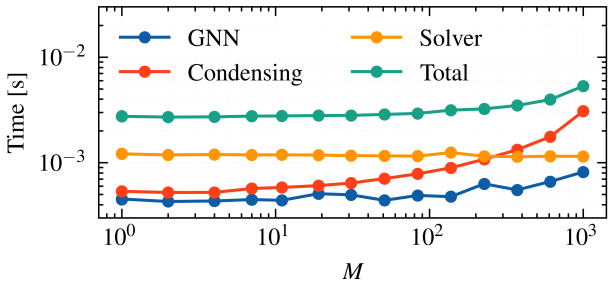
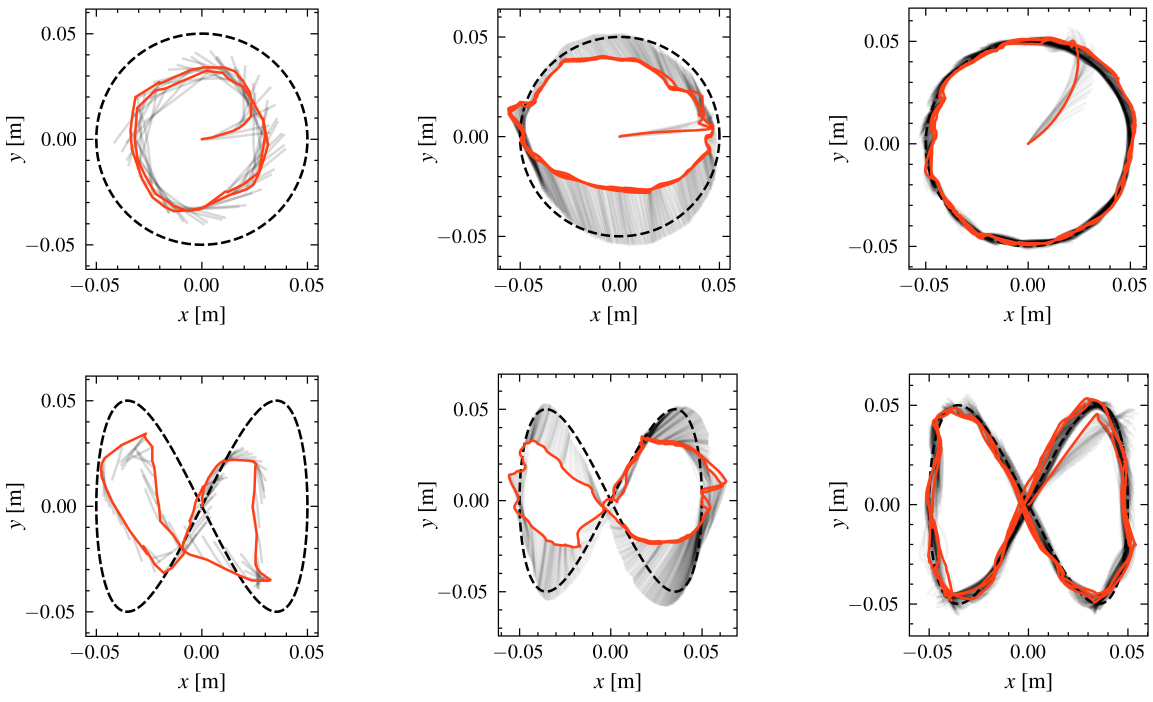


Fig. 2: Scalability of the proposed GNN-MPC framework with respect to the number of subsystems M. The computation times of the relevant stages in Algorithm 1 are shown in log-log scale.

We note that the linearization step represents the smallest contribution to the overall solve time, indicating that efficient solution algorithms of the OCP are crucial for high-dimensional systems. The QP solve time remains almost constant, since the number of inputs and constraints remains unchanged, while condensing captures most of the computational burden, which is effectively parallelized on the GPU.

C. Hardware experiments

We validate the proposed GNN-MPC framework on a physical soft trunk robot. Two tasks are considered: (i) trajectory tracking of a figure-eight and a circle, and (ii) obstacle avoidance. The MPC runs at 100 Hz with a prediction horizon of $N = 20$. The robot is represented by $M = 4$ nodes, whose positions are measured via motion capture, each interacting with $d = 2$ neighbors. Each node is described by $\bar{n}_x = 6$ states, corresponding to its position and velocity in $\mathbb{R}^3$. The GNN functions ψ and ϕ are parameterized by 4 and 8 hidden layers with 50 and 250 hidden units, respectively. The cost weights are tuned separately for each model to ensure the best closed-loop performance. Figure 3 illustrates the closed-loop end effector trajectories for MPC involving the Koopman method, SSM orthogonal reduction and our approach. We note that SSM orthogonal offered better closed-loop results than the optimal linear projections variant on hardware. The GNN-MPC achieves average tracking errors of 6.25 mm (figure-eight) and 3.67 mm (circle), outperforming the baselines by at least 63.6%. For the Koopman method, we obtained average errors of 23.76 mm (figure-eight) and 19.60 mm (circle), and 17.15 mm (figure-eight) and 15.77 mm (circle) for the SSM baseline. These results highlight the ability of the GNN to capture hardware-specific effects such as input nonlinearities and tendon slackness. Furthermore, the average solve times are 332 ms (Koopman), 2.9 ms (SSM), and 9.11 ms (ours).



(a) Koopman (b) SSM-Orth (c) GNN

Fig. 3: Reference tracking of a circle and figure-eight using (a) Koopman Operator modeling, (b) SSM orthogonal reduction and (c) our GNN-MPC approach. The reference is shown in dashed black, while the closed-loop trajectories are shown in red. We additionally provide the MPC plans for each time step in grey.

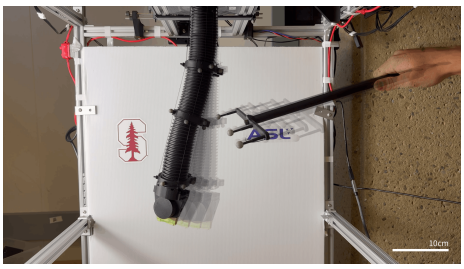


Fig. 4: Deflection of the trunk as the obstacle approaches the middle node. The end effector attempts to remain as close as possible to its resting position.

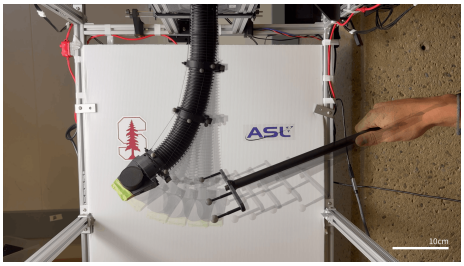


Fig. 5: Deflection of the trunk as the obstacle approaches the end effector node. The latter moves further away from the obstacle compared to the previous scenario.

To address system resonances induced by control delays, we apply a low-pass filter with time-constant $T_f = 0.24s$ to the control inputs generated by the GNN. The induced delay is compensated by shifting the reference trajectory by 19 steps. Further, all considered models require careful tuning and may exhibit unstable behavior depending on the training data and regularization parameters. The baseline models are particularly sensitive to the choice of training trajectories, showing stable predictions for quasi-periodic trajectories in case of SSM and random-walk trajectories for the Koopman method. In contrast, the GNN demonstrates the ability to learn generalized dynamics from both types, yielding the best results on random-walk trajectories on hardware. We use 50 training trajectories of 20s each for all models and apply no filtering other than the removal of outliers caused by the motion capture system. Finally, we note that the GNN still requires hyperparameter tuning through grid search to ensure smooth and reliable closed-loop control.

Finally, we present the benefit of full-node control by enabling collision avoidance across the entire soft robot through individual barrier terms for each of the four measured nodes in the cost function. Figure 4 shows the deflection of the trunk as the obstacle approaches the middle node, while Figure 5 shows the deformation when the obstacle approaches the end effector node. The GNN-MPC successfully avoids collisions while keeping other nodes close to their resting position, demonstrating the capability to handle complex interactions with the environment.

VII. Conclusion

We presented a framework for real-time optimal control of high-dimensional systems that integrates Graph Neural Networks within Model Predictive Control. By modeling dynamics as spatial graphs under the assumption of localized interactions, our approach preserves structural sparsity and enables efficient control through a custom condensing algorithm that scales linearly with the number of system nodes, further accelerated by GPU parallelization. In simulation, the GNN sustains closed-loop control at 100 Hz for systems with up to 1,000 nodes. While the GNN shows slightly lower prediction accuracy than the Koopman operator in simulation, it achieves 63.6% better tracking performance on the considered hardware platform. The high dimensionality of the Koopman lifting [3] further leads to larger solve times, hindering high-frequency real-time control.

Moreover, our GNN model is capable of learning the system dynamics from 50 arbitrary random-walk open-loop trajectories, streamlining the training process compared to the multi-step identification procedures required by the Koopman and SSM baselines. Our method further enables applications where control of the entire system state is explicitly considered.

Avenues for future work include extending the framework to time-varying graph structures, improving computational efficiency by addressing constraint handling bottlenecks with advanced solvers, exploring applications beyond soft robotics, and establishing formal stability and robustness guarantees for the closed-loop system.

VIII. Acknowledgments

The authors warmly thank Paul Wolff, Carmen Amo Alonso and Lukas Schroth. This work was partially supported by Toyota Motor Engineering & Manufacturing North America (TEMA). The views expressed in this paper are solely those of the authors and do not necessarily reflect those of the supporting entity.

References

[1] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[2] R. J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *Int. Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[3] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, "Modeling and control of soft robots using the koopman operator and model predictive control," 2019.

[4] J. I. Alora, M. Cenedese, G. Haller, and M. Pavone, "Discovering dominant dynamics for nonlinear continuum robot control," *npj Robotics*, vol. 3, no. 1, p. 5, 2025.

[5] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, and k. kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *Conf. on Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[6] J. B. Rawlings, D. Q. Mayne, M. Diehl, *et al.*, *Model Predictive Control: Theory, Computation, and Design*, vol. 2. Nob Hill Publishing Madison, WI, 2020.

[7] G. Frison, *Algorithms and Methods for High-Performance Model Predictive Control*. PhD thesis, 2016.

[8] G. Mengaldo, F. Renda, S. L. Brunton, M. Bächer, M. Calisti, C. Duriez, G. S. Chirikjian, and C. Laschi, "A concise guide to modelling the physics of embodied intelligence in soft robotics," *Nature Reviews Physics*, vol. 4, pp. 595–610, Sept. 2022.

[9] C. Della Santina, R. K. Katzschmann, A. Biechi, and D. Rus, "Dynamic control of soft robots interacting with the environment," in *IEEE International Conference on Soft Robotics*, pp. 46–53, 2018.

[10] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLoS ONE*, vol. 11, pp. 1–19, Feb. 2016.

[11] G. Haller and S. Ponsioen, "Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction," *Nonlinear Dynamics*, vol. 86, pp. 1493–1534, Nov. 2016.

[12] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," in *IEEE International Conference on Soft Robotics*, pp. 39–45, 2018.

[13] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges," May 2021. arXiv:2104.13478 [cs].

[14] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in *Int. Conf. on Learning Representations*, 2021.

[15] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *Int. Conf. on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 4470–4479, PMLR, July 2018.

[16] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *Int. Conf. on Learning Representations*, 2019.

[17] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, "Propagation networks for model-based control under partial observation," in *Proc. IEEE Conf. on Robotics and Automation*, pp. 1205–1211, 2019.

[18] J. D. Almeida, P. Schydlo, A. Dehban, and J. Santos-Victor, "Sensorimotor graph: Action-conditioned graph neural network for learning robotic soft hand dynamics," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pp. 9569–9576, 2021.

[19] D. Gammelli, K. Yang, J. Harrison, F. Rodrigues, F. C. Pereira, and M. Pavone, "Graph neural network reinforcement learning for autonomous mobility-on-demand systems," in *Proc. IEEE Conf. on Decision and Control*, pp. 2996–3003, 2021.

[20] D. Gammelli, J. Harrison, K. Yang, M. Pavone, F. Rodrigues, and F. C. Pereira, "Graph reinforcement learning for network control via bi-level optimization," in *Int. Conf. on Machine Learning* (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of *Proceedings of Machine Learning Research*, pp. 10587–10610, PMLR, July 2023.

[21] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control∗∗this research was supported by the german federal ministry for economic affairs and energy (bmwi) via eco4wind (0324125b) and dyconpv (0324166b), and by dfg via research unit for 2401.," *IFAC-Papers Online*, vol. 53, no. 2, pp. 6563–6569, 2020. 21st IFAC World Congress.

[22] G. Frison, D. Kouzoupis, J. B. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for nonlinear model predictive control," in *Proc. IEEE Conf. on Decision and Control*, pp. 4457–4462, 2016.

[23] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A sparse and condensed qp formulation for predictive control of lti systems," *Automatica*, vol. 48, no. 5, pp. 999–1002, 2012.

[24] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *Int. Conf. on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 8459–8468, PMLR, July 2020.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[26] M. Diehl, H. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[27] H. Buurmeijer, L. A. Pabon, J. I. Alora, R. S. Kaundinya, G. Haller, and M. Pavone, "Taming high-dimensional dynamics: Learning optimal projections onto spectral submanifolds," 2025.