From Coupled Oscillators to Graph Neural Networks: Reducing Over-smoothing via a Kuramoto Model-based Approach

Tuan Nguyen
FPT Software AI Center

Hirotada Honda
Toyo University

Takashi Sano
Toyo University

Vinh Nguyen*
FPT Software AI Center

Shugo Nakamura*
Toyo University

Tan M. Nguyen*
National University of Singapore

Abstract

We propose the Kuramoto Graph Neural Network (KuramotoGNN), a novel class of continuous-depth graph neural networks (GNNs) that employs the Kuramoto model to mitigate the over-smoothing phenomenon, in which node features in GNNs become indistinguishable as the number of layers increases. The Kuramoto model captures the synchronization behavior of non-linear coupled oscillators. Under the view of coupled oscillators, we first show the connection between Kuramoto model and basic GNN and then over-smoothing phenomenon in GNNs can be interpreted as phase synchronization in Kuramoto model. The KuramotoGNN replaces this phase synchronization with frequency synchronization to prevent the node features from converging into each other while allowing the system to reach a stable synchronized state. We experimentally verify the advantages of the KuramotoGNN over the baseline GNNs and existing methods in reducing over-smoothing on various graph deep learning benchmark tasks.

1 INTRODUCTION

Graph neural networks (GNNs) have been widely adopted in applications such as computational chemistry, social networks, and drug discovery due to their ability to capture complex relationships between nodes and edges in a graph (Gilmer et al., 2017; Fan et al., 2019; Xiong et al., 2019). GNNs use multiple graph propagation layers to iteratively update each node's representation by aggregating the representations of its neighbors and the node itself. However, a significant limitation of GNNs is the over-smoothing problem, which occurs when the GNN repeatedly aggregates information from neighboring nodes. This can cause the representations of nodes from different classes to become indistinguishable, leading to reduced model performance (Oono and Suzuki, 2019; Nt and Maehara, 2019).

To address the over-smoothing problem and improve our theoretical understanding of GNNs, recent research has considered GNNs as a discretization scheme of dynamical systems. In this framework, each propagation layer in GNNs is a discrete step of a differential equation (Chamberlain et al., 2021; Thorpe et al., 2021; Rusch et al., 2022; Xhonneux et al., 2020; Oono and Suzuki, 2019). This class of models is often referred to as continuous-depth models, and they are more memory-efficient and can effectively capture the dynamics of hidden layers (Chen et al., 2018).

In this paper, we propose a new physically-inspired framework for understanding GNNs and solving the over-smoothing by using the Kuramoto model (Kuramoto, 1975). The Kuramoto model describes the dynamical behavior of nonlinear coupled oscillators and provides a useful proxy for explaining the over-smoothing problem in GNNs. Specifically, we demonstrate that the over-smoothing problem in GNNs is analogous to phase synchronization in coupled oscillators, where all oscillators in the network rotate spontaneously with a common frequency and phase. Building on this insight, we propose a new training scheme, called KuramotoGNN, that encourages the model to learn representations that balance between the need to aggregate information from neighboring nodes and

* Co-last authors. Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume TBD.

the risk of over-smoothing. Our experiments on standard benchmark datasets demonstrate that KuramotoGNN outperforms several baseline models, including popular GNN variants, on node classification tasks. Overall, our work provides a promising direction for addressing the over-smoothing problem in GNNs and opens up new avenues for exploring the connection between GNNs and dynamical systems.

The rest of the paper is organized as follows. Section 2 provides an overview of related work on GNNs. Section 3 provides the background of the continuous-depth GNNs model and Kuramoto model. Section 4 presents our proposed framework KuramotoGNN, together with the similarities to other continuous-depth GNN models and the over-smoothing phenomenon. Section 5 presents our experimental results, and we conclude the paper in Section 6 with a discussion of the implications of our framework and directions for future research.

2 RELATED WORKS

Neural ODEs. Neural ODEs (NODE) are a class of continuous-depth models for neural networks based on Ordinary Differential Equations. The idea of NODE was first proposed in Chen et al. (2018), and builds on previous studies exploring the relationship between deep learning and differential equations (Haber and Ruthotto, 2017). Mathematically, NODE is represented as the following first-order ODE:

$$\frac{\mathrm{d}z(t)}{\mathrm{d}t} = f(z(t), t, \theta) \tag{1}$$

where $f(z(t), t, \theta)$ is specified by a neural network and θ is its weights. Using numerical methods, such as the Euler discretization with a step size of 1, (1) can be discretized into a vanilla residual network (He et al., 2016). The NODE architecture has several advantages in the training process, including the use of the adjoint method Pontryagin et al. (2018) for back-propagation, which is more memory-efficient than saving all states of intermediate layers.

Since the proposal of NODE, numerous works have explored the use of ODEs in deep learning for image classification and time-series prediction, such as Neural CDE (Kidger et al., 2020), Neural SDE (Liu et al., 2019), and augmented NODE (Dupont et al., 2019). Despite the advantages of NODEs, there are also some limitations and challenges associated with their use. For example, it can be difficult to determine the appropriate discretization method for a given problem. Nonetheless, NODEs have demonstrated significant promise as a class of continuous-depth models for neural networks.

GNNs. Graph Neural Networks (GNNs) (Kipf and Welling, 2016; Velickovic et al., 2017; Hamilton et al., 2017) are a class of models that can effectively learn graph representations. Generally, GNNs have multiple propagation layers, where in each layer, the representation of each node is updated according to representation features of neighboring nodes, called messages. Each type of model has a different type of message aggregation function. However, it has been shown that GNNs are prone to over-smoothing (Oono and Suzuki, 2019; Nt and Maehara, 2019), which occurs when node representations converge to the same values. In particular, it has been shown analytically that deeper GNNs exponentially lose expressive power as the number of layers goes to infinity, and nodes of equal degree in the same connected component will have the same representation Oono and Suzuki (2019).

Several approaches have been proposed to address this over-smoothing problem. An early solution is concatenation, which is effective, but does not easily scale to very deep networks, due to the size of latent features increasing with the number of layers. Another approach is residual mapping, as used in GCNII (Chen et al., 2020), which maintains good performance up to 64 layers of GNNs. However, in GCNII, a new parameter, called the scale parameter, is introduced to control the level of smoothing, increasing the complexity of the model and making it more difficult to train and optimize.

Recently, inspired by the Neural ODEs (Chen et al., 2018), works have turned conventional GNNs into continuous-depth models to solve the over-smoothing problem. For example, Thorpe et al. (2021); Xhonneux et al. (2020) added source terms to the linear ODE to change the convergence point of the equation, Chamberlain et al. (2021) used heat diffusion-type Partial Differential Equations (PDEs) to design GNNs and, to some extent, slowed the over-smoothing process, and Rusch et al. (2022) modeled GNNs as second-order oscillator PDEs with damping terms to analyze the dynamic behavior of the model to avoid over-smoothing.

The work presented in Rusch et al. (2022) (GraphCON) is closest to ours. Both KuramotoGNN and GraphCON share a common objective: addressing the oversmoothing problem inherent in GNNs, while drawing inspiration from coupled oscillators. On the one hand, our work, KuramotoGNN, contributes by establishing a formal connection between oversmoothing and phase synchronization through theoretical analysis. On the other hand, GraphCON's theoretical foundations predominantly revolve around second-order dynamics of coupled oscillators. This framework effectively addresses the oversmoothing by preventing its

occurrence under suitable parameter settings. However, GraphCON does not explicitly explore the theoretical linkage between oversmoothing and synchronization, and the model's behavior beyond the oversmoothing context needs further investigation. For instance, whether GraphCON is stable or not in the Lyapunov sense remains a question that requires deeper exploration. Note that although the Kuramoto model is first-order, it can also be extended to second-order ODE equations. In that case, KuramotoGNN and GraphCON are quite similar in the form of equations.

Synchronization. Synchronization is a phenomenon observed in complex networks across many fields, including biology, chemistry, physics, and social systems. One classic example is the synchronous flashing of fireflies, where initially the fireflies flash randomly, but after a short period of time, the entire swarm starts flashing in unison. Coupled oscillator networks are commonly used to study the dynamics of synchronization in networks, where a population of oscillators is connected by a graph that describes the interactions between the oscillators. The oscillator is a simple yet powerful concept that captures a rich dynamic behavior (Dörfler and Bullo, 2014).

The Kuramoto model has been applied across a wide range of disciplines, including biology, neuroscience, engineering, and even in data processing. For example, the Kuramoto model has been used to study brain networks (Varela et al., 2001), laser arrays (Kozyreff et al., 2000), power grids (Motter et al., 2013), wireless sensor networks (Tanaka et al., 2009), consensus problems (Olfati-Saber et al., 2007), and data clustering as a method of unsupervised machine learning (Miyano and Tsutsui, 2007, 2008). For further references, we refer to Strogatz (2000) which provides a comprehensive introduction to synchronization phenomena and their applications, while the review by Dörfler and Bullo (2014); Acebrón et al. (2005) focus on the dynamics of complex networks and synchronization.

Interestingly, there is a connection between synchronization and the over-smoothing phenomenon in GNNs. Both involve a collective behavior of the nodes in the network, where nodes become more similar to each other over time. By leveraging the insights from the study of synchronization, we can potentially gain a deeper understanding of the over-smoothing problem in GNNs and develop new solutions to address this issue.

3 BACKGROUND

3.1 Graph Neural Diffusion (GRAND)

Graph Neural Diffusion (GRAND) is an architecture for graphs that uses the diffusion process (Chamberlain et al., 2021). A graph is defined as $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} \in \mathbb{R}^{n\times f}$ represents the n vertices, each with f features, and $\mathbf{E} := E_{ij}$ is an $n \times n$ matrix that represents the edge weights between the nodes. The model is governed by the following Cauchy problem (we provide detail explanation for equation (2) in **SM**).

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \mathrm{div}(\mathcal{G}(X(t), t) \odot \nabla \mathbf{X}(t)) \tag{2}$$

$$\mathbf{X}(0) = \psi(\mathbf{V}) \tag{3}$$

where d is the size of encoded input features, $\psi : \mathbb{R}^{n\times f} \to \mathbb{R}^{n\times d}$ is an affine map that represents the encoder function to the input node features, $\mathbf{X}(\mathbf{t}) = [\big(\mathbf{x}_1(t)\big)^\top, \ldots, \big(\mathbf{x}_n(t)\big)^\top]^\top \in \mathbb{R}^{n\times d}$ is the node function matrix, $\odot$ is the point-wise multiplication, and div is the divergence operator.

In the simplest case, when $\mathcal{G}$ is only dependent on the initial value, $\mathbf{X}(0)$, then the equation becomes:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = (\hat{\mathbf{A}} - \mathbf{I})\mathbf{X}(t) \tag{4}$$

where $\hat{\mathbf{A}}$ is an $n \times n$ matrix, and it is right-stochastic (i.e., each row of $\hat{\mathbf{A}} \odot \mathbf{E}$ summing to 1) which is related to attention weight. In the GRAND model (Chamberlain et al., 2021), to formulate the matrix $\hat{\mathbf{A}}$, they used the multi-head self-attention mechanism (Vaswani et al., 2017). The scaled-dot product attention is given by

$$\mathbf{A}^l(\mathbf{X}_i(0), \mathbf{X}_j(0)) = \mathrm{softmax}\left(\frac{(\mathbf{W}_K\mathbf{X}_i(0))^\top \mathbf{W}_Q \mathbf{X}_j(0)}{d_k}\right) \tag{5}$$

where d_k is a hyper-parameter, and $\mathbf{W}_K$, $\mathbf{W}_Q \in \mathbb{R}^{d_k\times d}$ are the learnable parameters. Then, $\hat{\mathbf{A}}_{ij} = \frac{1}{h}\sum_{l=1}^{h} \mathbf{A}^l(\mathbf{X}_i(0), \mathbf{X}_j(0))$ with h is the hyper-parameter for the number of heads.

Models derived from (4) have been shown to extend to a depth larger than conventional GNNs while still maintaining acceptable performance (Chamberlain et al., 2021). However, in Section 4, we argue that even with this type of model, over-smoothing cannot be completely eliminated.

3.2 The Kuramoto model

In the study of interacting limit-cycle oscillators, weakly coupled systems have been found to exhibit

dynamical behavior that depends only on the phases of the oscillators, as demonstrated by Winfree (1967). Such models are called phase-reduced models. One of the most well-known phase-reduced models is the Kuramoto model, which describes the dynamics of a network of N phase oscillators θ_i, with natural frequencies ω_i and coupling strength κ_{ij}, using the following phase equation:

$$\dot{\theta}_i = \omega_i + \sum_j \kappa_{ij} \sin(\theta_j - \theta_i) \tag{6}$$

Each oscillator, i, is characterized by intrinsic and extrinsic factors (Acebrón et al., 2005). The internal influence is the natural frequency of the oscillator, ω_i, while the external influences are the interactions with other oscillators in the network through the weight (or coupling) matrix, $\kappa \in \mathbb{R}^{n \times n}$. From (6), the synchronization behavior has been further analyzed using the mean field coupling, that is, $\kappa_{ij} = K/N$, where K is a constant and N is the number of nodes in the graph (Kuramoto, 1975). Hence, (6) becomes:

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_j \sin(\theta_j - \theta_i) \tag{7}$$

Equation (7) is the well-known *Kuramoto model*. The Kuramoto model has two types of states: a nonsynchronized state, in which each oscillator rotates independently with its own frequency, and a partially synchronized state, in which some of the oscillators rotate with the same effective frequency. It has been found that strengthening the couplings provides a synchronization transition from the nonsynchronized state to the partially synchronized state and that the continuity of the transition is determined by the natural frequency distribution (Kuramoto, 1975). Specifically, if the distribution of natural frequencies ω_i is unimodal and symmetric, then synchronization occurs in equation (7) if the coupling parameter K exceeds a certain threshold $K_{critical}$ determined by the distribution of ω_i.

To investigate synchronization behavior or measure synchronization rate, the author in Kuramoto (1975) introduced a complex-valued term called the *order parameter*:

$$re^{i\phi} = \frac{1}{N} \sum_j e^{i\theta_j} \tag{8}$$

where ϕ is defined as the average phase. $r(t)$ is a real value in the range of $0 \leq r(t) \leq 1$, where $r(t) \approx 1$ indicates that the phases are close together or that the synchronization rate is high, while a value of $r(t) \approx 0$ indicates that the phases are spread out over the circle.

4 KURAMOTO GRAPH NEURAL NETWORK

4.1 Model Formulation

In this work, we consider the generalized form of the Kuramoto model:

$$\dot{\theta}_i = \omega_i + \frac{K}{\sum_j a_{ij}} \sum_j a_{ij} \sin(\theta_j - \theta_i) \tag{9}$$

By adapting the setting of initial value and the right-stochastic matrix $\hat{\mathbf{A}} = [a_{ij}]$ which has been stated in 3.1, we now can present the KuramotoGNN, a class of continuous-depth GNNs based on the Kuramoto model (9):

$$\dot{\mathbf{x}}_i = \omega_i + K \sum_j a_{ij} \sin(\mathbf{x}_j - \mathbf{x}_i) \tag{10}$$

$$\mathbf{X}(0) = \Omega = \psi(\mathbf{V}) \tag{11}$$

with natural frequencies $\omega_i \in \mathbb{R}^d$, $\Omega = [\omega_1^\top, \ldots, \omega_n^\top]^\top$, i^{th} node vector representations $\mathbf{x}_i(t) \in \mathbb{R}^d$, and sin function operates in a component-wise manner. Unlike many previous works related to the Kuramoto model, in which they try to find the threshold $K_{critical}$ to control the dynamics of the system, we approach the problem in a data-driven way. We fix K as a hyperparameter and optimize Ω, $\mathbf{X}(0)$, and $\hat{\mathbf{A}}$ through the training process. We adopt the setting of GRAND, where a_{ij} serves as learnable parameters but depends solely on the initial value $\mathbf{X}(\mathbf{0})$, as shown in equation (5).

Proposition 4.1. *Graph Neural Diffusion* (4) *is the linearized dynamics of the Kuramoto model.*

Proof. Assuming that ω_i are identical, in which $\omega_1 = ... = \omega_N$, we obtain the following equation by using the rotating frame of reference:

$$\dot{\mathbf{x}}_i = K \sum_j a_{ij} \sin(\mathbf{x}_j - \mathbf{x}_i) \tag{12}$$

Next, we assume that $K = 1$ and roughly approximate the nonlinear sin function by using the first order of the Taylor expansion, then we can obtain the following equation which is identical to Graph Neural Diffusion (GRAND) (4):

$$\dot{\mathbf{x}}_i = \sum_j a_{ij} (\mathbf{x}_j - \mathbf{x}_i) \tag{13}$$

Hence, GRAND (4) is the linearized dynamics of the Kuramoto model. □

Remark 4.2. *In GRAND, if* $\hat{\mathbf{A}}$ *in equation* (4) *depends only on the initial value* $(\mathbf{x}_i(0), \mathbf{x}_j(0))$*, then we have a linear version of GRAND named* ***GRAND-l****. Meanwhile, if* $\hat{\mathbf{A}}$ *is time dependent and depends on* $(\mathbf{x}_i(t), \mathbf{x}_j(t))$*, we obtain the non-linear GRAND called* ***GRAND-nl****. On the contrary, the KuramotoGNN model, in general, is nonlinear even in the case that we consider* a_{ij} *to depend solely on initial value or to depend on time-dependent values.*

On the exploding and vanishing gradients. In the case of highly deep GNN architectures, it is crucial to explore potential approaches to alleviate the issues of exploding and vanishing gradients. For simplicity and without any loss of generality, we consider the case of the fully-connected graph and scalar node features by setting $d = 1$, or $x_i^t = \mathbf{x}_{i,1}(t)$, and explicitly reduce the KuramotoGNN to the Euler discretization form with a step-size $\Delta t \ll 1$,

$$x_i^t = x_i^{t-1} + \Delta t \left(x_i^0 + \frac{1}{n} \sum \sin(x_j^{t-1} - x_i^{t-1}) \right) \quad (14)$$

$$\mathbf{X}(0) = [x_1^0, \ldots, x_n^0]^\top = \psi(\mathbf{V}) = \mathbf{V}\mathbf{W} \quad (15)$$

Where W is the learnable parameters, M is the number of ODE integrations or number of model's layers, $t = 1, \ldots, M$. Furthermore, let us consider a scenario where the objective of the GNN is to approximate the ground truth vector $\hat{\mathbf{X}} \in \mathbb{R}^n$ with the following loss function.

$$J(W) = \frac{1}{2n} \sum_{i=1}^{N} \|x_i^n - \hat{x}_i\|^2 \quad (16)$$

At every step of gradient descent, we need to compute the gradient $\frac{\partial J}{\partial \mathbf{W}}$ which measures the contribution made by parameters $\mathbf{W}$. Using the chain rule, we obtain the following.

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\partial J}{\partial \mathbf{Z}^L} \frac{\partial \mathbf{Z}^L}{\partial \mathbf{Z}^1} \frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} \frac{\partial \mathbf{Z}^0}{\partial \mathbf{W}} \quad (17)$$

$$\frac{\partial \mathbf{Z}^L}{\partial \mathbf{Z}^1} = \prod_{i=1}^{L} \frac{\partial \mathbf{Z}^i}{\partial \mathbf{Z}^{i-1}} \quad (18)$$

Here, $\mathbf{Z}^L = [x_1^L, \ldots, x_n^L]$ and L is represented as the number of layers. Assuming an approximate relationship $\frac{\partial \mathbf{Z}^i}{\partial \mathbf{Z}^{i-1}} \approx \lambda$, the repeated multiplication in equation (18) implies that $\frac{\partial \mathbf{Z}^L}{\partial \mathbf{Z}^1} \approx \lambda^L$. When $\lambda > 1$, the total gradient (17) can *grow exponentially* with the number of layers, leading to the problem of exploding gradients. In contrast, when $\lambda < 1$, the total gradient (17) can *decay exponentially* with the number of layers, resulting in the problem of vanishing gradients. These scenarios can hinder successful training as the gradient either becomes excessively large or remains stagnant, impeding effective parameter updates.

Proposition 4.3. *We assume that* $\Delta t \ll 1$ *is chosen to be sufficiently small. Then, the gradient of the loss function* (16) *with respect to any learnable weight parameter* $\mathbf{W}$ *is bounded as:*

$$\|\frac{\partial J}{\partial \mathbf{W}}\|_\infty \leq \frac{1}{n} \left[\alpha(\max|x_i^0| + 1) + \max|\hat{x}_i|\right] (\beta + \alpha)\beta \|\mathbf{V}\|_\infty \quad (19)$$

$$\alpha = M\Delta t, \qquad \beta = 1 + \frac{\Delta t}{n} \quad (20)$$

Where $\|\mathbf{x}\|_\infty := \max_i |\mathbf{x}_i|$ is the infinity norm.The upper bound presented in equation (19) demonstrates that the total gradient remains globally bounded, regardless of the number of layers M, effectively addressing the issue of exploding gradients. However, it is important to note that this upper bound does not automatically eliminate the possibility of vanishing gradients. To further investigate this matter, we follow Rusch et al. (2022) to derive the following proposition for the gradients (proof provided in the **SM**).

Proposition 4.4. *We assume that* $\Delta t \ll 1$ *is chosen to be sufficiently small. Then, the gradient of the loss function* (16) *can be represented as:*

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\partial J}{\partial \mathbf{Z}^M} \left[\Delta t(\mathbf{E}' + \sum_{i}^{M} \mathbf{E}_{i-1}) + \mathbf{I} + \mathcal{O}(\Delta t^2) \right] \mathbf{W} \quad (21)$$

with $\frac{\partial J}{\partial \mathbf{Z}^M} = \frac{1}{n}[x_1^M - \hat{x}_1, \ldots, x_n^M - \hat{x}_n]$ *and the order notation,* $\mathbf{E}$*,* $\mathbf{E}'$*, and* $\frac{\partial J}{\partial \mathbf{Z}^M}$ *are defined in* ***SM****.*

Thus, although the gradient (21) can be small, *it will not vanish exponentially* by increasing the number of layers M, mitigating the vanishing gradient problem.

On the stability. In the Kuramoto model, the stability of the synchronized state if the relative frequency differences converge as $t \to \infty$, and hence if $\dot{\mathbf{x}}_i$ in (10) satisfy the following condition, we can say that the model is stable.

$$\lim_{t \to \infty} \|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\| = \mathbf{0}, \quad \forall i, j \in \mathcal{V} \quad (22)$$

In numerical simulations, it has been observed that the frequency synchronization of Kuramoto oscillators tend to zero, regardless of initial configuration of $\mathbf{x}_i(0)$. One notable advantage of utilizing the Kuramoto model in our study is the extensive availability of previous research papers, which contribute to a rich repository of results, theorems, and analyses. The following theorem help us to understand the stability of (10) concretely.

Theorem 4.5. *(Ha et al., 2016) Suppose that the initial configuration* $\mathbf{X}(0)$ *and natural frequencies* Ω *sat-*

isfy

$$r_0 > 0, \quad \mathbf{x}_i(0) \neq \mathbf{x}_j(0) \quad \forall i, j \in \mathcal{V}, \quad \max \|\Omega\| < \infty \tag{23}$$

Then there exists a large coupling strength $K_\infty > 0$ *such that if* $K > K_\infty$ *then there exists the stable state* (22) *which is the solution of* (10) *with initial data* $\mathbf{X}(0)$.

Remark 4.6. *In this stability section, we also consider the case of the scalar node feature for simplicity.*

Remark 4.7. r_0 *is the order parameter* (8) *of the initial configuration* $\mathbf{X}_0$, $r_0 = |r| = \frac{1}{N}\sum_j e^{i\mathbf{x}_j(0)}$. *Theorem 4.5 covers all initial configurations, except that with* $r_0 = 0$ *or when all the initial oscillators are distributed uniformly around the circle.*

4.2 Over-smoothing as synchronization

Definition 4.8. *The over-smoothing phenomenon occurs when the following condition converges exponentially to zero:*

$$\lim_{t \to T} \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| = 0, \forall i \neq j. \tag{24}$$

With T is the terminal time of the ODE. In conventional GNN architectures, over-smoothing has been a widely observed and analyzed phenomenon (Oono and Suzuki, 2019; Nt and Maehara, 2019). The previous reference showed that over-smoothing is a phenomenon in which the node representations converge to a fixed state exponentially and, in that fixed state, the node representations are indistinguishable. We notice a strong resemblance between over-smoothing and *phase synchronization* in coupled oscillator dynamics.

In general, oscillator synchronization occurs when they reach the *frequency synchronization* state. In other words, when the frequencies of the coupled oscillators converge to some common frequency, $\|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\| = 0, \forall i, j = 1, \ldots, N$, despite differences in the natural frequencies of the individual oscillators. If, in addition to frequency synchronization, the oscillator representations $\mathbf{x}_i(t)$ converge to a common value, $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| = 0$, it is called *phase synchronization*.

Remark 4.9. *The definition of the frequency synchronization state aligns precisely with equation* (22).

Proposition 4.10. *Over-smoothing is the phase synchronization state of the node features. Under the analysis of synchronization, conventional Graph Neural Diffusion is easily drawn into the over-smoothing phenomenon.*

Proof. We first show that the phase synchronization state is an exponentially stable solution of identical Kuramoto oscillators (12), then with the Definition 4.8 we can indicate that the over-smoothing state is exactly the same as the phase synchronization state. For simplicity, here we consider the scalar node feature, $d = 1$, and denote $x_i = \mathbf{x}_{i,1}$.

Let us say that if a solution of (12) achieves phase synchronization, then it does so with a value equal to $x_{sync}(t)$, that is, $x_1(t) = x_2(t) = \cdots = x_N(t) = x_{sync}(t)$. By transformation into a rotating frame with frequency x_{sync}, we can obtain $x_1(t) = x_2(t) = \cdots = x_N(t) = 0$ or $\mathbf{X} = 0$ if we consider $\mathbf{X} = [x_1^\top(t), \ldots, x_N^\top(t)]$.

We consider the following energy function U:

$$U(\mathbf{X}) = \sum_{i,j \in E} a_{ij}(1 - \cos(x_i - x_j)) \tag{25}$$

Because $\frac{\partial U}{\partial x_i} = \sum_j a_{ij} \sin(x_i - x_j)$, we can represent $\nabla U(\mathbf{X}) = [\frac{\partial U}{\partial x_1}, ..., \frac{\partial U}{\partial x_n}]$ as follows:

$$\nabla U(\mathbf{X}) = -\frac{\dot{\mathbf{X}}^\top}{K} \tag{26}$$

Which leads to the time derivative of U:

$$\dot{U} = \nabla U(\mathbf{X})\dot{\mathbf{X}} = -\frac{1}{K}\dot{\mathbf{X}}^\top\dot{\mathbf{X}} \leq 0 \tag{27}$$

Using the LaSalle Invariance Principle (Khalil, 2002, Theorem 4.4), which is akin to the Lyapunov method, we can analyze the behavior of all solutions of an autonomous ODE as $t \to \infty$. The principle gives conditions that describe the behavior of the system rather than focusing on the stability of a particular equilibrium solution as in the Lyapunov method. Specifically, if a positive and non-increasing scalar-valued function $U : \mathbf{R}^n \to \mathbf{R}$ (also known as a Lyapunov or energy function) satisfies $\dot{U}(x) \leq 0$ for all x, then every solution converges exponentially to a set of critical points $\{x | \dot{U}(x) = 0\}$.

Therefore, following LaSalle Invariance Principle, every solution of (12) converges exponentially to a set of critical points that are the root of the right-hand side of (12). Looking at (12), it is easily confirmed that the phase synchronization state, $x_i = x_j, \forall i \neq j$, is a solution of the equation. Furthermore, it has been shown that the phase synchronization state is the only global attractor, and the synchronization time scales with the inverse of the smallest non-zero eigenvalue of the Laplacian matrix (Arenas et al., 2008). Therefore, *the phase synchronization state is the only exponentially stable state* of equation (12). And now, together with the Definition 4.8, we can clearly see that oversmoothing is the phase synchronization state of the node features.

We have demonstrated the equivalence between oversmoothing and phase synchronization. We have also

shown that in the scenario of a single channel of node representations and identical oscillators (as in equation (12)), phase synchronization occurs exponentially. For multichannel or when $d > 1$, the same conclusion can be derived in a similar way for each channel. And thus, along with Proposition 4.1 we can say that under the analysis of synchronization, conventional Graph Neural Diffusion (13) is likely encountered with the over-smoothing phenomena. □

We next present a theorem that offers an easy way to reduce the over-smoothing phenomenon by introducing non-identical natural frequencies ω_i into the equation.

Theorem 4.11. *Let $\mathbf{x}_i(.), i = 1, \ldots, N$ be a solution of (10). If there exists $i, j \in 1, \ldots, N$ such that $\omega_i \neq \omega_j$, then (24) does not happen.*

We explain how our model can avoid over-smoothing with the help of Theorem 4.11. Recall that we set $\omega_i = \psi(\mathbf{V}_i) = \mathbf{D}\mathbf{V}_i + \mathbf{b} \in \mathbb{R}^d$, where ψ is a learnable function among linear functions, and $\mathbf{V_i}$ is the input feature vector of the i^{th} node, and there are a total of N distinct nodes. It is obvious that $\mathbf{V}_i \neq \mathbf{V}_j, \forall i \neq j$. And thus, by passing through ψ, $\omega_i = \omega_j, \forall i \neq j$ or $\psi(\mathbf{V}_i) = \psi(\mathbf{V}_j), \forall i \neq j$ is equivalent to saying $D(\mathbf{V}_i - \mathbf{V}_j) = 0, \forall i \neq j$, which is impossible because the number of row rank of $\mathbf{D} \leq N \ll \frac{N(N-1)}{2}$, with $\frac{N(N-1)}{2}$ is the number of pair $\mathbf{V}_i \neq \mathbf{V}_j$.

Remark 4.12. *Nonidentical frequencies ω_i in equation (10) prevent phase synchronization, but according to Theorem 4.5, a stable frequency synchronization state can still be achieved. Thus, nonidentical frequencies induce a transition from phase synchronization to frequency synchronization.*

Remark 4.13. *Oono and Suzuki (2019) highlighted the occurrence of over-smoothing in conventional GNN architectures, where node representations converge to the over-smoothing state exponentially. Interestingly, this phenomenon is strongly similar to the synchronization manifold in coupled oscillator dynamics. In synchronization, the oscillators evolve synchronously on the same solution, $\lim_{t\to\infty} \mathbf{x}_1(t) = \cdots = \mathbf{x}_N(t)$, which is defined by the eigenvector with the lowest eigenvalue, λ_1. Similarly, through their work, over-smoothing happens when the dynamics of node representations approach an invariant subspace that corresponds to the lowest frequency of graph spectra. Therefore, it is apparent that the concept of synchronization can be used to understand the phenomenon of over-smoothing in GNNs.*

5 EXPERIMENTAL RESULTS

We conduct experiments to compare the performance of our proposed method, KuramotoGNN, with GRAND, GRAND++, GCNII, GraphCON and other popular GNN architectures on node classification tasks, including GCN, GAT, and GraphSage. For all experiments, we run 100 splits for each dataset with 20 random seeds for each split, and we conducted on a server with one NVIDIA RTX 3090 graphics card.

For most of the settings, we adopt from GRAND (Chamberlain et al., 2021) for KuramotoGNN including adaptive numerical differential equation solvers. Except for 2 factors: the coupling hyper-parameter, which belongs solely to the KuramotoGNN, and the integration time, which measures the implicit depth of continuous-model, were slightly changed. Following Chamberlain et al. (2021), we study seven graph node classification datasets, namely CORA (McCallum et al., 2000), CiteSeer (Sen et al., 2008), PubMed (Namata et al., 2012), CoauthorCS (Shchur et al., 2018), the Amazon co-purchasing graphs Computer and Photo (McAuley et al., 2015). The descriptions of experimental settings in the **SM**.

5.1 KuramotoGNN is resilient to deep layers

To demonstrate that the model does not suffer from over-smoothing, we conducted experiments in various of depth (by changing the integration limit, T) and measure the performance in accuracy. One notable point is that the proposed equation in the GRAND paper (4) was modified in its implementation, as described below:

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \alpha(\hat{\mathbf{A}} - \mathbf{I})\mathbf{X}(t) + \beta\mathbf{X}(0) \tag{28}$$

with learnable α and β parameters. In fact, (28) bears a striking resemblance to our proposed (10), if we make a rough approximation of the function sin as previously mentioned.

In this experiment, we conduct both versions of linear GRAND, with and without adding $\mathbf{X}(0)$. For all models, we used the random split method with 10 initialization, along with the Euler step-fixed solver with step size 0.1 step size for a fair comparison in the computational process instead of using an adaptive step size scheme which gives more superior results (Chamberlain et al., 2021).

Figure 1 shows the change in accuracy of three kinds of models: **KuramotoGNN**, **GRAND-l** and **GRAND-l w/o X(0)** for various depth values $T = \{1, 4, 8, 16, 32, 64, 80, 100\}$. We can see that without

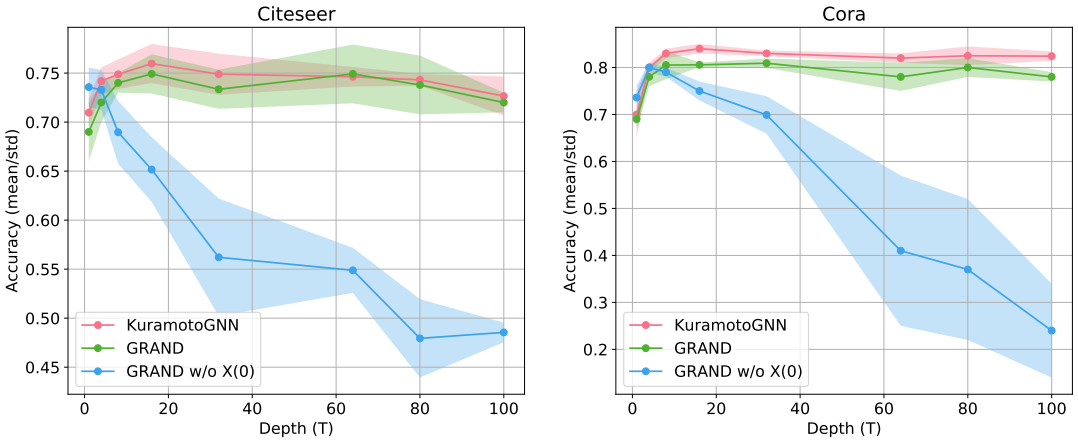

Figure 1: Change in performance at different depth (T) on Cora and Citeseer dataset.

Table 1: Mean and std of classification accuracy of KuramotoGNN and other GNNs on six benchmark graph node classification tasks. The highest accuracy is highlighted in bold for each number of labeled data per class. (Unit: %)

Model	CORA	Citeseer	Pubmed	Computers	CoauthorCS	Photo
KuramotoGNN	**85.18±1.3**	**76.01±1.4**	**80.15±0.3**	**84.6±0.59**	**92.35±0.2**	**93.99±0.17**
GraphCON	84.2±1.3	74.2±1.7	79.4±1.3	84.1±0.9	90.5±1.0	93.16±0.5
GRAND++	82.95±1.37	73.53±3.31	79.16±1.37	82.99±0.81	90.80±0.34	93.55±0.38
GRAND-l	82.46±1.64	73.4±5.05	78.8±1.63	84.27±0.6	91.24±0.4	93.6±0.4
GCNII	84.02±0.5	70.26±0.7	78.95±0.9	80.28±2.1	91.11±0.2	92.1±0.4
GCN	82.07±2.03	74.21±2.90	76.89±3.27	82.94±1.54	91.09±0.35	91.95±0.11
GAT	80.04±2.54	72.02±2.82	74.55±3.09	79.98±0.96	91.33±0.36	91.29±0.67
GraphSAGE	82.07±2.03	71.52±4.11	76.49±1.75	73.66±2.87	90.31±0.41	88.61±1.18

adding $\mathbf{X}(0)$, the performance of **GRAND-l** is reduced significantly along the depth, while the **KuramotoGNN** and **GRAND-l w/o X(0)** maintain the performance when increasing depth.

5.2 KuramotoGNN performances on various benchmarks

We evaluate the effectiveness of our model on six popular graph benchmarks. Our KuramotoGNN demonstrates better performance in terms of accuracy when compared to other continuous models: GraphCON(Rusch et al., 2022), GRAND++(Thorpe et al., 2021), GRAND-l(Chamberlain et al., 2021), GCNII(Chen et al., 2020), and traditional models: GCN(Kipf and Welling, 2016)), GAT(Velickovic et al., 2017), and GraphSAGE(Hamilton et al., 2017).. Table 1 compares the accuracy of fine-tuned Kuramoto with GRAND-l (following equation (28)) and other GNNs.

6 CONCLUSION

In summary, we introduce the Kuramoto Graph Neural Network (KuramotoGNN), a novel class of continuous-depth graph neural networks inspired by the Kuramoto model. Our theoretical analysis establishes a connection between the over-smoothing problem and *phase synchronization* in coupled oscillator networks. By incorporating non-identical natural frequency terms, we mitigate the over-smoothing issue, leveraging insights from synchronization and the Kuramoto model. Empirical experiments demonstrate the superior performance of KuramotoGNN compared to other GNN variants, especially with limited labeled data. It is worth noting that our work focuses on the classic Kuramoto model, while future research could explore variations such as time-delayed Kuramoto, adaptive coupling functions, or second-order Kuramoto with damping (Dörfler and Bullo, 2014).

References

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16): 8749–8760, 2019.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, 2021.

Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2021.

T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *ICML*, pages 18888–18909. PMLR, 2022.

Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pages 10432–10441. PMLR, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In Huzihiro Araki, editor, *International Symposium on Mathematical Problems in Theoretical Physics*, pages 420–422, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg. ISBN 978-3-540-37509-8.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1): 014004, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Liev Semiónovich Pontryagin, VG Boltyanskii, RV Gamkrelidze, EF Mishchenko, KN Trirogoff, and LW Neustadt. *LS Pontryagin Selected Works: The Mathematical Theory of Optimal Processes*. Routledge, 2018.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707, 2020.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in Neural Information Processing Systems*, 32, 2019.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

Florian Dörfler and Francesco Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539–1564, 2014.

Francisco Varela, Jean-Philippe Lachaux, Eugenio Rodriguez, and Jacques Martinerie. The brainweb: phase synchronization and large-scale integration. *Nature reviews neuroscience*, 2(4):229–239, 2001.

Gregory Kozyreff, AG Vladimirov, and Paul Mandel. Global coupling with time delay in an array of semiconductor lasers. *Physical Review Letters*, 85(18): 3809, 2000.

Adilson E Motter, Seth A Myers, Marian Anghel, and Takashi Nishikawa. Spontaneous synchrony in

power-grid networks. *Nature Physics*, 9(3):191–197, 2013.

Hisa-Aki Tanaka, Hiroya Nakao, and Kenta Shinohara. Self-organizing timing allocation mechanism in distributed wireless sensor networks. *IEICE Electronics Express*, 6(22):1562–1568, 2009.

Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

Takaya Miyano and Takako Tsutsui. Data synchronization in a network of coupled phase oscillators. *Physical review letters*, 98(2):024102, 2007.

Takaya Miyano and Takako Tsutsui. Collective synchronization as a method of learning and generalization from sparse data. *Physical Review E*, 77(2): 026112, 2008.

Steven H. Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1):1–20, 2000. ISSN 0167-2789. doi: https://doi.org/10.1016/S0167-2789(00)00094-4.

Juan A. Acebrón, L. L. Bonilla, Conrad J. Pérez Vicente, Félix Ritort, and Renato Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Rev. Mod. Phys.*, 77:137–185, Apr 2005. doi: 10.1103/RevModPhys.77.137.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Arthur T. Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoretical Biology*, 16(1):15–42, 1967. ISSN 0022-5193. doi: https://doi.org/10.1016/0022-5193(67)90051-3.

Seung-Yeal Ha, Hwa Kil Kim, and Sang Woo Ryoo. Emergence of phase-locked states for the kuramoto model in a large coupling regime. *Communications in Mathematical Sciences*, 14(4):1073–1091, 2016.

Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

Alex Arenas, Albert Díaz-Guilera, Jurgen Kurths, Yamir Moreno, and Changsong Zhou. Synchronization in complex networks. *Physics reports*, 469(3): 93–153, 2008.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, page 1, 2012.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

Supplement to "From Coupled Oscillators to Graph Neural Networks: Reducing Over-smoothing via a Kuramoto Model-based Approach"

Contents

1 FURTHER DISCUSSION

1.1 THE NEED OF ALLEVIATING OVERSMOOTHING

We believe that addressing oversmoothing can bring significant benefits to constructing very deep models and handling limited labeled training data. When faced with the challenge of limited labeled training data, the key issue is extracting meaningful features that effectively capture the underlying graph patterns and nuances. Oversmoothing exacerbates this challenge by leading to information loss, which, in turn, reduces the model's ability to accurately differentiate between nodes [Calder et al., 2020]. Our intuition suggests that mitigating oversmoothing can enable the model to capture and retain relevant features that might otherwise remain obscured. This enhancement strengthens the model's capacity to learn distinctive representations, even when working with a limited amount of labeled data under the realizability assumption, where the true function falls within the hypothesis set we consider.

The ability to construct deeper graph neural networks provides versatile opportunities across various applications. While deeper KuramotoGNNs do not inherently guarantee better performance for specific tasks, the flexibility to create models with varying depths is a significant advantage. This flexibility is particularly valuable because it opens the door to broader applicability across other neural ODE techniques. Furthermore, our observation indicates that graph neural networks have been successfully employed for learning complex dynamical systems [Pfaff et al., 2020]. Consequently, the ability to build continuous-depth graph neural networks suitable for large-time T holds substantial promise for studying the long-term behavior of complex physical systems.

1.2 FURTHER COMPARISONS TO RELATED MODELS

In this context, we provide additional comparisons involving GRAND (a representative linear ODE model) and GraphCON (a model closely related to ours).

While the Kuramoto model is inherently a first-order ODE, it can also be extended to second-order ODE equations. In this case, the equations for KuramotoGNN and GraphCON exhibit a notable similarity in their form.

For the 2nd order KuramotoGNN, the equations are as follows:

$$\begin{aligned}\dot{x}_i &= y_i \\ \dot{y}_i &= \sum_{j\in N(i)} a_{ij}\sin(x_j - x_i) - \omega_i - \alpha y_i\end{aligned}$$

On the other hand, GraphCON's equations are expressed as:

$$\begin{aligned}\dot{x}_i &= y_i \\ \dot{y}_i &= \sigma(\sum_{j\in N(i)} a_{ij}x_j) - x_i - \alpha y_i\end{aligned}$$

One can notice that the differences are the coupling function and ω, X_i terms. In GraphCON, they consider the σ function as the ReLU function.

Another notable distinction between GRAND and the Kuramoto model pertains to their mathematical nature. GRAND is a linear ODE model, whereas the Kuramoto model is inherently nonlinear. This nonlinearity imparts KuramotoGNN with a richer and more expressive dynamic behavior compared to GRAND. For instance, while GRAND may have just one equilibrium point, the Kuramoto model can exhibit multiple stable solutions that extend beyond the confines of an equilibrium point. Instead, the Kuramoto model can manifest stable limit cycles—periodic orbits characterized by complex patterns that the system trajectories converge to. Furthermore, in the case of linear GRAND, its flow map remains linear, effectively a composite of linear maps. In contrast, KuramotoGNN introduces a nonlinear flow map that significantly enhances its expressive capabilities.

It's worth noting that our utilization of the standard form of the Kuramoto model represents just one facet of its potential. Different formulations of the Kuramoto equations can give rise to various dynamics, offering the

opportunity for exploration beyond the standard model. For example, incorporating individual coupling strengths can introduce attraction-repulsion dynamics, potentially leading to cluster synchronization phenomena.

In summary, our work primarily provides a perspective on the oversmoothing phenomenon and its connection to synchronization, with the aim of enhancing our understanding of the behavior of graph neural networks. We hope this clarification sheds light on the motivation behind our approach and its valuable contributions to the field.

2 DATASET

The statistics of the datasets are summarized in Table 1.

Cora [McCallum et al., 2000]. A scientific paper citation network dataset consists of 2708 publications which are classified into one of seven classes. The citation network consists of 5429 links; each publication is represented by a vector of 0/1-valued indicating the absence/presence of the 1433 words in a corpus.

Citeseer [Sen et al., 2008]. Similar to Cora, Citeseer is another scientific publications network consists of 3312 publications and each publication is classified into one of 6 classes. The publication is represented by a vector of 0/1 valued that also indicates the absence or presence of the corresponding word from a dictionary of 3703 unique words.

Pubmed [Namata et al., 2012]. The Pubmed dataset consists of 19717 scientific publications related to diabetes, and all publications in the dataset are taken from the Pubmed database. Each publication is classified into one of 3 classes. The network has 44338 links and each publication is represented by TF/IDF weighted word vector from a dictionary consists of 500 unique words.

CoauthorCS [Shchur et al., 2018]. The CoauthorCS is a co-authorship graph of authors with publications related to the field of computer science. The dataset is based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. In this dataset, nodes represent the authors and an edge is established if they are co-authored in a paper. Each node is classified into one of 15 classes, and each node is represented by a vector of size 6805 indicating the paper keywords for each author's papers. The network consists of 18333 nodes and 163788 edges.

Computers [McAuley et al., 2015]. Computers dataset is a segment of the Amazon co-purchase graph. In this graph, each node is classified into one of 10 classes and each node is represented as a product. If two products are often bought together, an edge will be established. Each product is represented by a bag-of-words features vector of size 767. The data set consists of a total of 13752 products and 491722 relations between two products.

Photo [McAuley et al., 2015]. Similar to Computers, Photo is another segment of Amazon co-purchase graph, the properties of nodes and edges are exactly the same with Computers. In this dataset, the network consists of 238163 edges and 7650 nodes, each node is classified into one of eight classes and each node is represented by a vector size of 745.

Table 1: Statitics of 6 datasets

Dataset	Classes	Features	#Nodes	#Edges
CORA	7	1433	2485	5069
Citeseer	6	3703	2120	3679
Pubmed	3	500	19717	44324
CoauthorCS	15	6805	18333	81894
Computer	10	767	13381	245778
Photo	8	745	7487	119043

3 DETAIL ON GRAPH NEURAL DIFFUSION

We recall that the Graph Neural Diffusion (GRAND) is governed by the following Cauchy problem.

$$\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t} = \mathrm{div}(\mathcal{G}(X(t), t) \odot \nabla\mathbf{X}(t)) \tag{1}$$

$$\mathbf{X}(0) = \psi(\mathbf{V}) \tag{2}$$

Where d is the size of encoded input features, $\psi : \mathbb{R}^{n\times f} \to \mathbb{R}^{n\times d}$ is an affine map that represents the encoder function to the input node features, $\mathbf{X(t)} = [\left(\mathbf{x}_1(t)\right)^\top, \ldots, \left(\mathbf{x}_n(t)\right)^\top]^\top \in \mathbb{R}^{n\times d}$ is the node function matrix, $\odot$ is the point-wise multiplication, and div is the divergence operator.

The gradient of a node-function matrix $\mathbf{X}$ is an edge-function $\nabla\mathbf{X} \in \mathbb{R}^{n\times n\times d}$ with $[\nabla\mathbf{X}]_{ij} = \mathbf{x}_j - \mathbf{x}_i \in \mathbb{R}^d$. And $\mathcal{G} = (\mathbf{X}(t), t) \in \mathbb{R}^{n\times n}$ is a matrix function which takes $\mathbf{X}(t)$ as input of the function. Furthermore, $\mathcal{G}$ always satisfies the condition that each row of $\mathcal{G} \odot \mathbf{E}$ summing to 1. Finally, div or the divergence of an edge function $\nabla\mathbf{X}$, $\mathrm{div}(\nabla\mathbf{X}) = ([\mathrm{div}(\nabla\mathbf{X})]_1^\top, \ldots, [\mathrm{div}(\nabla\mathbf{X})]_n^\top) \in \mathbb{R}^{n\times d}$ is defined as:

$$[\mathrm{div}(\nabla\mathbf{X})]_i = \sum_{i=1}^{n} E_{ij}[\nabla\mathbf{X}]_{ij} \tag{3}$$

4 TRAINING OBJECTIVE

The full training optimizes the cross-entropy loss:

$$\mathcal{L}(\mathbf{Y}, \mathbf{T}) = H(\mathbf{Y}, \mathbf{T}) = \sum_{i=1}^{n} \mathbf{t_i}^\top \log \mathbf{y}_i \tag{4}$$

where $\mathbf{t}_i$ is the one-hot truth vector of the i^{th} node and $\mathbf{y}_i = \phi(\mathbf{x}_i(T))$ is the prediction of the KuramotoGNN with $\phi : \mathbb{R}^d \to \mathbb{R}^{num\_class}$ is a linear decoder function:

$$\mathbf{y}_i = \mathbf{D}\mathbf{x}_i(T) + b \tag{5}$$

$$= \mathbf{D}\left(\mathbf{x}_i(0) + \int_0^T \frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t} dt\right) + b \tag{6}$$

$$= \mathbf{D}\left(\psi(\mathbf{V_i}) + \int_0^T \frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t} dt\right) + b \tag{7}$$

$$= \mathbf{D}\left(\mathbf{M}\mathbf{V_i} + b_\psi + \int_0^T \frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t} dt\right) + b \tag{8}$$

Moreover, T is the terminated time of the ODE and $\frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t}$ is the KuramotoGNN equation which is the below equation. Note that $\mathbf{V}_i$ is the input feature vector of the i^{th} node and $\mathbf{M}, \mathbf{D}$, $\frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t}$ contains learnable parameters.

$$\frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t} = \omega_i + K \sum_{j\in\mathcal{N}(\mathbf{x}_i)} a_{ij} \sin(\mathbf{x}_j - \mathbf{x}_i) \tag{9}$$

In here, $\omega_i = \mathbf{x}_i(0) = \mathbf{M}\mathbf{V_i}$, and $a_{ij} = softmax\left(\frac{\mathbf{W}_K\mathbf{X}(0)(\mathbf{W}_Q\mathbf{X}(0))^\top}{\sqrt{d_k}}\right)$ with d_k is a constant, $\mathbf{W}_K, \mathbf{W}_Q$ are learnable parameters, and K is the coupling strength constant.

5 EXPERIMENTAL DETAILS AND MORE RESULTS

For solving the ODEs, we use torchdiffeq library ODE Solver [Chen et al., 2018]. For the encoder ψ, we employ a simple fully connected layer with dropout. Also for the decoder, after obtaining results from solving the ODEs, $X(T)$, we pass it through a simple fully connected layer to get final labels.

For all six graph node classification datasets, including CORA, CiteSeer, PubMed, coauthor graph CoauthorCS, and Amazon co-purchasing graphs Computer and Photo, we consider the largest connected component. Table 3 lists the fine-tuned T, and Table 6 lists the fine-tuned coupling strength K for the results in the main paper.

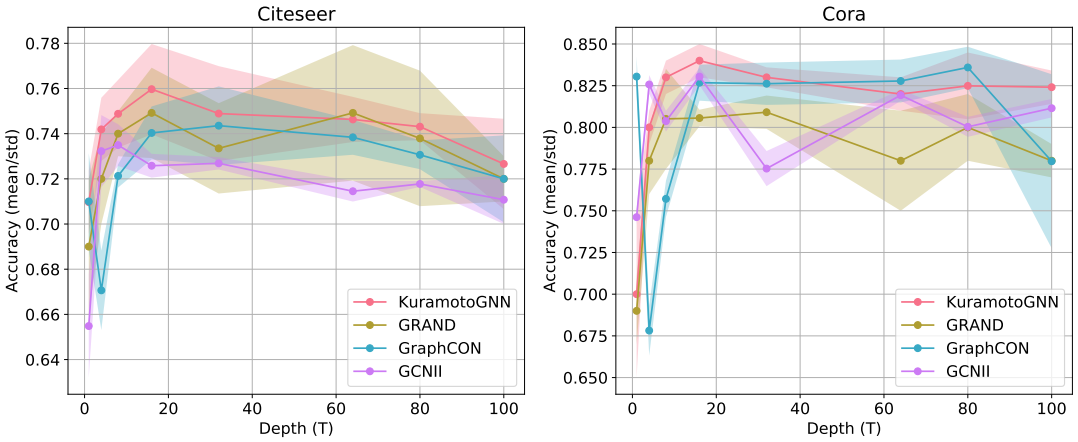

Figure 1: Comparisions between KuramotoGNN and GNNs architectures that specifically tackle over-smoothing in different depth.

Although our T values are smaller, please note that we use a non-linear interaction function sin instead of a linear interaction function $f(\mathbf{x}) = \mathbf{x}$. That indicates our model requires more iterations for the ode solver to solve it, and each iteration is equivalent to a layer of neural network. Therefore, in terms of "real depth", our model is still deeper than GRAND-l. Table 2 shows the iterations in one epoch (we used the adaptive solver dopri5) of our model compared to GRAND-l.

Table 2: Comparing solver iterations for KuramotoGNN and GRAND's ODE equation.

Model	CORA	Citeseer	Pubmed	CoauthorCS	Computer	Photo
KuramotoGNN	1900	1200	120	100	115	85
GRAND-l	200	300	50	50	100	70

Table 3: Fine-tuned T for KuramotoGNN and GRAND-l.

Model	CORA	Citeseer	Pubmed	CoauthorCS	Computer	Photo
KuramotoGNN	12	5	8	0.8	1	1.5
GRAND-l	18.2948	7.8741	12.9423	3.2490	3.5824	3.6760

To further test the resilience to depth, we compare KuramotoGNN with other GNNs architectures that specifically tackle over-smoothing in Table 1 with different $T = 1, 4, 8, 16, 32, 64, 80, 100$. Again, we used fixed-step solver Euler with step size 0.1 for fair comparison in computational process for all continuous model, except for GCNII [Chen et al., 2020] which is already a discretized model.

We also further explore the effects of the depth and coupling strength for KuramotoGNN by conducting further experiements based on various depths and coupling strengths. Table 5 shows the performances in accuracy of KuramotoGNN on three datasets: CORA, Citeseer, and Pubmed. Overall, the coupling strength is more sensitive in case of small depths, but in larger depths, the chances in performances are not significant between choices of coupling strengths.

5.1 EVALUATIONS ON LIMITED TRAINING DATA

Besides helping to avoid over-smoothing and being able to train in deep layers, KuramotoGNN also can boost the performance of different tasks with low-labeling rates. Table 4 compares the accuracy of KuramotoGNN with other GNNs. We notice that with few labeled data, in most tasks, KuramotoGNN is significantly more accurate than the other GNNs including GRAND-l. Only for CoauthorCS and Photo datasets, the GCN outperforms both KuramotoGNN and GRAND-l on extreme limited label cases.

Table 4: Mean and std of classification accuracy of KuramotoGNN and other GNNs with different number of labeled data per class (#per class) on six benchmark graph node classification tasks. The highest accuracy is highlighted in bold for each number of labeled data per class. (Unit: %)

Model	#per class	CORA	Citeseer	Pubmed	Computers	CoauthorCS	Photo
KuramotoGNN	1	**63.48±7.2**	**62.06±4.55**	**65.93±3.65**	62.26±7.73	60.48±2.7	80.18±1.8
	2	**71.17±5.0**	**66.85±6.72**	**72.62±3.15**	76.24±2.72	75.89±0.73	82.67±0.8
	5	**79.11±0.91**	**72.42±2.0**	76.43±1.73	81.43±0.78	87.22±0.99	**89.35±0.29**
	10	**83.53±1.36**	**74.27±1.5**	76.86±2.17	**83.84±0.54**	**90.49±0.28**	**91.35±0.1**
	20	**85.18±1.3**	**76.01±1.4**	**80.15±0.3**	**84.6±0.59**	**92.35±0.2**	**93.99±0.17**
GRAND++	1	54.94±16.0	58.95±9.59	65.94±4.87	**67.65±0.37**	60.30±1.5	**83.12±0.78**
	2	66.92±10.04	64.98±8.31	69.31±4.87	76.47±1.48	76.53±1.85	**87.31±0.9**
	5	77.80±4.46	70.03±3.63	71.99±1.91	**82.64±0.56**	84.83±0.84	88.33±1.21
	10	80.86±2.99	72.34±2.42	75.13±3.88	82.99±0.81	86.94±0.46	90.65±1.19
	20	82.95±1.37	73.53±3.31	79.16±1.37	82.99±0.81	90.80±0.34	93.55±0.38
GRAND-l with $X(0)$	1	54.14±11.0	50.58±17.3	55.47±12.5	47.96±1.3	58.1±4.6	76.89±2.25
	2	68.56±9.1	57.65±13.2	69.71±7.01	75.47±1.7	75.2±4.2	80.54±2.3
	5	77.52±3.1	67.48±4.2	70.17±4.52	81.23±0.6	85.27±2.1	88.58±1.7
	10	81.9±2.4	71.7±7.3	77.37±2.31	82.71±1.5	87.6±1.8	90.95±0.6
	20	82.46±1.64	73.4±5.05	78.8±1.63	84.27±0.6	91.24±0.4	93.6±0.4
GCNII	1	58.64±9.2	56.44±8.4	58.18±7.5	48.46±10.3	**70.49±6.35**	42.02±1.9
	2	64.5±6.4	53.61±8.7	65.05±4.09	71.29±3.4	**83.13±1.6**	61.66±6.4
	5	76.22±0.88	69.2±0.9	70.24±0.63	73.60±2.1	**89.02±0.8**	83.31±2.1
	10	75.35±1.1	66.29±1.2	**76.63±1.2**	77.83±3.9	89.31±0.25	90.2±0.8
	20	84.02±0.5	70.26±0.7	78.95±0.9	80.28±2.1	91.11±0.2	92.1±0.4
GCN	1	47.72±15.33	48.94±10.24	58.61±12.83	49.46±1.65	65.22±2.25	82.94±2.17
	2	60.85±14.01	58.06±9.76	60.45±16.20	**76.90±1.49**	83.61±1.49	83.61±0.71
	5	73.86±7.97	67.24±4.19	68.69±7.93	82.47±0.97	86.66±0.43	88.86±1.56
	10	78.82±5.38	72.18±3.48	72.59±3.19	82.53±0.74	88.60±0.50	90.41±0.35
	20	82.07±2.03	74.21±2.90	76.89±3.27	82.94±1.54	91.09±0.35	91.95±0.11
GAT	1	47.86±15.38	50.31±14.27	58.84±12.81	37.14±7.87	51.13±5.24	73.58±8.15
	2	58.30±13.55	55.55±9.19	60.24±14.44	65.07±8.86	63.12±6.09	76.89±4.89
	5	71.04±5.74	67.37±5.08	68.54±5.75	71.43±7.34	71.65±4.56	83.01±3.64
	10	76.31±4.87	71.35±4.92	72.44±3.50	76.04±0.35	74.71±3.35	87.42±2.38
	20	80.04±2.54	72.02±2.82	74.55±3.09	79.98±0.96	91.33±0.36	91.29±0.67
GraphSAGE	1	43.04±14.01	48.81±11.45	55.53±12.71	27.65±2.39	61.35±1.35	45.36±7.13
	2	53.96±12.18	54.39±11.37	58.97±12.65	42.63±4.29	76.51±1.31	51.93±4.21
	5	68.14±6.95	64.79±5.16	66.07±6.16	64.83±1.62	89.06±0.69	78.26±1.93
	10	75.04±5.03	68.90±5.08	70.74±3.11	74.66±1.29	89.68±0.39	84.38±1.75
	20	82.07±2.03	71.52±4.11	76.49±1.75	73.66±2.87	90.31±0.41	88.61±1.18

5.2 EVALUATIONS ON HETEROPHILIC DATASET

To further demonstrate the effectiveness of KuramotoGNN, we include more experiments on the node classification task using heterophilic graph datasets: Cornell, Texas and Wisconsin from the CMU WebKB[1] project. The edges in these graphs represent the hyperlinks between webpages nodes. The labels are manually selected into five classes, student, project, course, staff, and faculty. The features on node are the bag-of-words of the web pages.The 10 generated splits of data are provided by [Pei et al., 2020].

Table 8 shows the performances of KuramotoGNN when compare with other differential equations based models: GRAND-l, GraphCON and discretized model: GCNII [Chen et al., 2020] . All the baselines are proceduced/re-proceduced from public code. Note that for some reasons, GraphCON [Rusch et al., 2022] re-proceduced results, which is created from their public code[2], are different from reported ones in their paper. Especially for Cornell dataset, the reported result was $\mathbf{84.3 \pm 4.8}$ while our reproduced one is only $\mathbf{74.3 \pm 4.6}$. Note that we have been trying to further tuning the model. In Table 8, we put the reported result in the original paper of GraphCON.

We also conducted experiments on two recent challenging heterophilic datasets: **roman-empire** and **amazon-**

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/.
[2] https://github.com/tk-rusch/GraphCON/

Table 5: Mean and std of classification accuracy of KuramotoGNN in different depths and coupling strengths on three CORA, Citeseer, and Pubmed graph node classification tasks. (Unit: %)

Depth T	Coupling Strength K	CORA	Citeseer	Pubmed
2	0.7	75.13±1.35	70.24±3.41	76.81±1.69
	0.8	76.27±2.89	71.85±2.16	78.07±1.77
	0.9	79.8±0.77	73.87±1.63	79.85±1.19
	1	78.15±0.98	70.89±1.81	77.61±2.22
3	0.7	75.89±1.77	72.42±2.26	76.88±2.58
	0.8	78.43±1.08	76.33±2.48	79.34±1.48
	0.9	79.67±0.77	72.58±2.91	78.77±0.99
	1	79.54±2.14	74.8±1.19	79.13±0.99
5	0.7	82.03±1.79	72.54±1.31	77.98±2.95
	0.8	79.37±0.49	72.58±2.77	79.46±0.48
	0.9	81.98±1.96	74.88±1.22	78.91±2.47
	1	82.92±0.88	73.99±0.84	80.46±1.82
8	0.7	81.37±1.13	74.56±1.65	80.17±0.80
	0.8	82.49±0.74	75.24±2.39	79.75±1.28
	0.9	82.77±1.29	75.4±2.43	80.07±0.57
	1	83.22±1.57	75.04±0.7	78.49±3.03
10	0.7	82.26±1.05	74.4±3.4	79.49±1.16
	0.8	82.49±0.98	75.93±1.18	79.27±0.52
	0.9	81.6±0.98	74.56±0.77	78.17±1.86
	1	83.43±1.3	75.12±1.02	79.08±1.93
12	0.7	83.53±0.72	74.88±1.87	78.84±1.69
	0.8	83.83±0.59	75.93±1.44	79.9±0.95
	0.9	81.75±1.61	74.35±1.99	79.93±0.52
	1	85.18±1.35	73.63±1.72	79.35±0.8
16	0.7	84.06±2.46	73.55±0.96	77.49±1.41
	0.8	82.06±2.05	74.68±1.17	78.67±1.36
	0.9	83.35±0.48	76.01±1.45	75.77±0.12
	1	82.82±1.13	75.16±1.19	75.51±2.18
18	0.7	84.37±0.68	75.16±0.94	78.46±2.46
	0.8	82.97±0.75	75.28±1.21	76.60±2.15
	0.9	82.99±0.44	73.83±1.95	75.67±1.63
	1	82.79±0.38	75.4±1.49	74.2±1.71

ratings. The experimental results have been included in Table 7 for GRAND-l, GraphCON, and KuramotoGNN.

Notably, the new heterophilic datasets appear to present challenges for all models, and we believe that further investigation into their dynamics and characteristics is needed. We'd like to highlight that, similar to GRAND-l, GraphCON employs a task-specific residual trick in its code. This trick's impact can vary, proving beneficial for some datasets while potentially negatively affecting others. We have thoroughly explored both versions of GraphCON, with and without this trick, to provide a comprehensive comparison. In the table, we define **GraphCON-res** as the version we used residual trick as standard public code, while GraphCON is the version we remove the trick. The same notion goes for KuramotoGNN and **KuramotoGNN-res**.

Additionally, we've observed that incorporating a similar trick into our model also yields performance improvements in certain datasets. This finding emphasizes the importance of considering dataset-specific characteristics when applying such techniques. In conclusion, the inclusion of these new heterophilic datasets has enabled us to broaden our insights into the performance of KuramotoGNN, GraphCON, and GRAND-l.

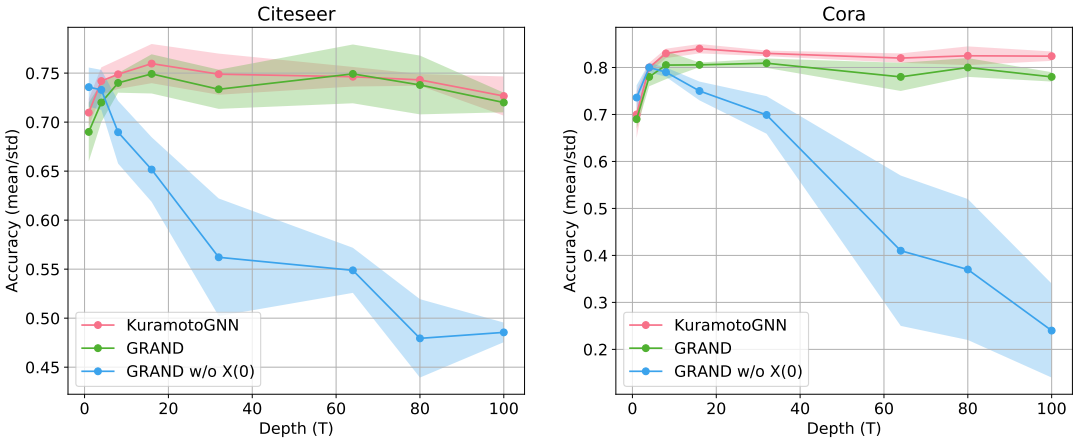

Figure 2: Change in performance at different depth (T) on Cora and Citeseer dataset.

Table 6: fine-tuned coupling strength K for KuramotoGNN.

Dataset	Coupling strength K
CORA	1
Citeseer	2
Pubmed	0.9
CoauthorCS	1.8
Computer	4
Photo	2

Table 7: Performance comparison between KuramotoGNN and GRAND, GraphCON for two new heterophlic datasets.

Model	roman-empire	amazon-ratings
GRAND-l	60.1±0.4	40.3±0.4
GraphCON	73.2±0.4	42.3±0.4
GraphCON-res	85.5±0.7	41.2±0.6
KuramotoGNN	83.0±0.5	41.9±0.4
KuramotoGNN-res	**86.07±0.6**	**42.9±0.7**

5.3 FURTHER EVALUATING ON LARGER DATASET

Open graph benchmark with paper citation network (ogbn-arxiv) [Hu et al., 2020]. Ogbn-arxiv consists of 169,343 nodes and 1,166,243 directed edges. Each node is an arxiv paper represented by a 128-dimensional features and each directed edge indicates the citation direction. This dataset is used for node property prediction and has been a popular benchmark to test the advantage of deep graph neural networks over shallow graph neural networks.

We have conducted additional experiments on the OGBN-arXiv node classification task. The results in Table 9 show that our KuramotoGNN improves over GRAND-l. We used the Euler fixed step solver with step size 0.1 for a fair comparison in the computational process.

5.4 EFFECT OF COUPLING STRENGHT K ON KURAMOTOGNN

To further investigate the effect of hyper-parameter K using empirical results, in the following experiments, we tried different settings of $K = \{0.4, 0.6, 0.8, 1, 1.5, 2, 3, \}$ on the Citeseer dataset using standard Planetoid split and on different depth $T = \{2, 4, 8\}$.

Table 8: Classification accuracy on heterophilic graph node classification task.

Model	Texas	Wisconsin	Cornell
KuramotoGNN	**85.4±6.2**	**87.6±3.3**	77.49±3.3
GCNII	81.08±4.5	82.31±3.1	79.7±6.7
GraphCON	81.1±3.6	85.2±3.1	**84.3±4.8**
GRAND-l	78.11±7.4	80.39±5.4	62.97±6.8

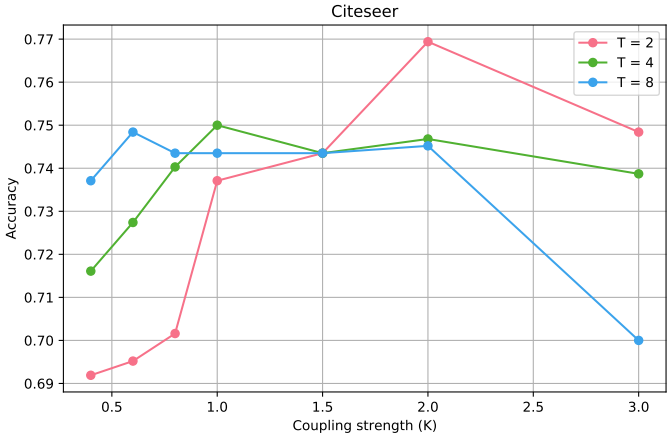

Figure 3: Change in performance of KuramotoGNN at different coupling strength (K) on Citeseer dataset.

Figure 3 shows the change in performances of Kuramoto on the Citeseer dataset on different settings of K. It is observed that the KuramotoGNN performs well on small values of K, while for too small K, it indicates not so much change for the coupled function, and for higher K, the performances start decreasing. However, this phenomenon is quite well matched with the analysis of the Kuramoto model [Kuramoto, 1975, Strogatz, 2000], in which the higher the coupling strength K, the system tends to synchronize better. Furthermore, we also do not suggest putting K too high, since it will increase the NFE (Number of Function Evaluations) of the solver to obtain an accurate solution, and thus, increasing the time of training.

6 PROOF FOR THEOREM 4.11

Let us recall the definition of over-smoothing and the KuramotoGNN equation from the main manuscript:

$$\lim_{t\to T} \|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| = 0, \forall i \neq j. \tag{10}$$

$$\dot{\mathbf{x}}_i = \omega_i + K \sum_j a_{ij} \sin(\mathbf{x}_j - \mathbf{x}_i) \tag{11}$$

We prove the contrapositive. It means that if condition (10) occurs, then $\omega_i = \omega_j, \forall i, j = 1, \ldots, N$.

We prove this in contradiction. We assume that equation (10) happens and $\omega_1 \neq \omega_2$, then we will try to reach a contradiction.

Since condition (10) happens, we substitute it into equation (11) to have the following limits:

$$\lim_{t\to\infty} (\dot{\mathbf{x}}_i(t) - \omega_i) = \mathbf{0}, \quad i = 1, 2. \tag{12}$$

Now, for all $T \in \mathbb{Z}^+$, we apply the mean value theorem to have

$$\mathbf{x}_1(T+1) - \mathbf{x}_1(T) = \dot{\mathbf{x}}_1(a_T), a_T \in (T, T+1),$$
$$\mathbf{x}_2(T+1) - \mathbf{x}_2(T) = \dot{\mathbf{x}}_2(b_T), b_T \in (T, T+1).$$

Table 9: Classification accuracy of the GRAND-l and KuramotoGNN trained with different depth on the OGBN-arXiv graph node classification task.

Model	$T = 1$	$T = 8$	$T = 32$	$T = 64$
KuramotoGNN	**66.00±0.8**	**69.87±0.2**	**69.31±0.3**	**68.32±0.2**
GRAND-l	64.43±0.5	69 .02±0.4	67.81±0.4	66.58±1.2

Using (12), we get

$$\mathbf{x}_1(T+1) - \mathbf{x}_1(T) \to \omega_1 \quad as \quad T \to \infty,$$
$$\mathbf{x}_2(T+1) - \mathbf{x}_2(T) \to \omega_2 \quad as \quad T \to \infty.$$

Hence, with $\mathbf{d}_{12}(t) = \mathbf{x}_1(t) - \mathbf{x}_2(t)$

$$\mathbf{d}_{12}(T+1) - \mathbf{d}_{12}(T) \to \omega_1 - \omega_2 \neq 0 \qquad as \quad T \to \infty,$$

which is a clear contradiction to the fact that condition (10) happens.

7 PROOF FOR PROPOSITION 4.3 AND 4.4

Our proofs are motivated by [Rusch et al., 2022].

7.1 Proposition 4.3

Let us recall the equations from the main manuscript. We consider the features of the scalar node $d = 1$ for simplicity.

$$x_i^m = x_i^{m-1} + \Delta t \left(x_i^0 + \frac{1}{m} \sum \sin(x_j^{m-1} - x_i^{m-1}) \right) \tag{13}$$
$$\mathbf{X}^0 = [x_1^0, \ldots, x_n^0]^\top = \mathbf{V} * \mathbf{W} \tag{14}$$

where $\mathbf{V} \in \mathbb{R}^{n \times f}$, $\mathbf{W} \in \mathbb{R}^{f \times 1}$, $\Delta t \ll 1$, $m = 1, 2, \ldots, M$.

Moreover, we are in a setting where the learning task is for the GNN to approximate the ground truth vector $\hat{\mathbf{X}} \in \mathbb{R}^n$. Consequently, we set up the following loss function.

$$J(W) = \frac{1}{2n} \sum_{i \in \mathcal{V}} \|x_i^M - \hat{x}_i\|^2 \tag{15}$$

We need to compute the gradient $\partial_W J$. Using the chain rule, we obtain the following.

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{\partial J}{\partial \mathbf{Z}^M} \frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1} \frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} \frac{\partial \mathbf{Z}^0}{\partial \mathbf{W}} \tag{16}$$
$$\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1} = \prod_{i=1}^{M} \frac{\partial \mathbf{Z}^i}{\partial \mathbf{Z}^{i-1}} \tag{17}$$

First, we find the bound of $\|\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1}\|_\infty$, $\|\frac{\partial J}{\partial \mathbf{Z}^M}\|_\infty$, $\|\frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0}\|_\infty$, $\|\frac{\partial \mathbf{Z}^0}{\partial \mathbf{W}}\|_\infty$, then we can multiply these terms together to get the final upper bound.

$$\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^{M-1}} = diag(\mathbf{A}) + \Delta t \mathbf{B} \tag{18}$$

$$\mathbf{A} = \begin{bmatrix} 1 - \Delta t \frac{1}{n} \sum_j \cos(x_j^{N-1} - x_n^{N-1}) \\ \vdots \\ 1 - \Delta t \frac{1}{n} \sum_j \cos(x_j^{N-1} - x_n^{N-1}) \end{bmatrix} \tag{19}$$

$$\mathbf{B} = \begin{bmatrix} 0 & \frac{1}{n}\cos(x_2^{N-1} - x_1^{N-1}) & \cdots & \frac{1}{n}\cos(x_n^{N-1} - x_1^{N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\cos(x_1^{N-1} - x_n^{N-1}) & \frac{1}{n}\cos(x_2^{N-1} - x_n^{N-1}) & \cdots & 0 \end{bmatrix} \tag{20}$$

Using the triangle inequality, we can obtain the upper bound for $\left\| \frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^{M-1}} \right\|_\infty$:

$$\left\| \frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^{M-1}} \right\|_\infty \leq \|diag(\mathbf{A})\|_\infty + \Delta t \|\mathbf{B}\|_\infty \leq (1 + \Delta t) + \frac{\Delta t}{n} \tag{21}$$

$$\text{Thus, } \left\| \frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1} \right\|_\infty \leq \left[1 + (1 + \frac{1}{n})\Delta t \right]^M \tag{22}$$

$$\tag{23}$$

With sufficiently small Δt, we have this inequality:

$$\left[1 + (1 + \frac{1}{n})\Delta t \right]^M \leq 1 + 2M(1 + \frac{1}{n})\Delta t \tag{24}$$

leads to the following bound,

$$\left\| \frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1} \right\|_\infty \leq 1 + 2M(1 + \frac{1}{n})\Delta t \tag{25}$$

A straight-forward differentiation of $\frac{\partial J}{\partial \mathbf{Z}^M}$ yields,

$$\frac{\partial J}{\partial \mathbf{Z}^M} = \frac{1}{n}[x_1^M - \hat{x}_1, \ldots, x_n^M - \hat{x}_n] \tag{26}$$

From (13) we can easily obtain the following inequality:

$$|x_i^M| \leq |x_i^{M-1}| + \Delta t(|x_i^0| + 1) \tag{27}$$

$$\text{Thus, } |x_i^M| \leq N\Delta t(|x_i^0| + 1) \tag{28}$$

Hence,

$$\left\| \frac{\partial J}{\partial \mathbf{Z}^M} \right\|_\infty \leq \frac{1}{n}(\max|x_i^M| + \max|\overline{x}_i|) \leq \frac{1}{n}(M\Delta t \max|x_i^0| + M\Delta t + \max|\overline{x}_i|) \tag{29}$$

Finding the bound for $\left\| \frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} \right\|_\infty$ is similar to $\left\| \frac{\partial \mathbf{Z}^N}{\partial \mathbf{Z}^1} \right\|_\infty$,

$$\frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} = diag(\mathbf{C}) + \Delta t \mathbf{D} \tag{30}$$

$$\mathbf{C} = \begin{bmatrix} 1 - \Delta t \left(1 + \frac{1}{n} \sum_j \cos(x_j^0 - x_n^0) \right) \\ \vdots \\ 1 - \Delta t \left(1 + \frac{1}{n} \sum_j \cos(x_j^0 - x_n^0) \right) \end{bmatrix} \tag{31}$$

$$\mathbf{D} = \begin{bmatrix} 0 & \frac{1}{n}\cos(x_2^0 - x_1^0) & \cdots & \frac{1}{n}\cos(x_n^0 - x_1^0) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\cos(x_1^0 - x_n^0) & \frac{1}{n}\cos(x_2^0 - x_n^0) & \cdots & 0 \end{bmatrix} \tag{32}$$

$$\left\| \frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} \right\|_\infty \leq \|diag(\mathbf{C})\|_\infty + \Delta t \|\mathbf{D}\|_\infty \leq 1 + \frac{\Delta t}{n} \tag{33}$$

Then we can find a bound for (17) by multiplying (29), (25), (33) together with $\frac{\partial \mathbf{Z}^0}{\partial \mathbf{W}} = \mathbf{V}$,

$$\left\| \frac{\partial J}{\partial \mathbf{W}} \right\|_\infty \leq \frac{1}{n} \left[\alpha(\max |x_i^0| + 1) + \max |\overline{x}_i| \right] (\beta + \alpha)\beta \|\mathbf{V}\|_\infty \tag{34}$$

$$\alpha = M\Delta t, \qquad \beta = 1 + \frac{\Delta t}{n} \tag{35}$$

7.2 Proposition 4.4

Motivated by [Rusch et al., 2022], we will need the following order notation:

$$\beta = \mathcal{O}(\alpha), \text{for } \alpha, \beta \in \mathbb{R}_+ \text{ if there exists constants } \overline{C}, \underline{C} \text{ that } \underline{C}\alpha \leq \beta \leq \overline{C}\alpha \tag{36}$$

$$\mathbf{M} = \mathcal{O}(\alpha), \text{for } \mathbf{M} \in \mathbb{R}^{d_1 \times d_2}, \alpha \in \mathbb{R}_+ \text{ if there exists constants } \overline{C} \text{ that } \|\mathbf{M}\| \leq \overline{C}\alpha \tag{37}$$

We can rewrite $\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^{M-1}}$ as the following

$$\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^{M-1}} = \mathbf{I} + \Delta t \mathbf{E}_{M-1} \tag{38}$$

$$\mathbf{E}_{M-1} = \begin{bmatrix} -\frac{1}{n}\sum_j \cos(x_j^{M-1} - x_n^{M-1}) & \frac{1}{n}\cos(x_2^{M-1} - x_1^{M-1}) & \cdots & \frac{1}{n}\cos(x_n^{M-1} - x_1^{M-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\cos(x_1^{M-1} - x_n^{M-1}) & \frac{1}{n}\cos(x_2^{M-1} - x_n^{M-1}) & \cdots & -\frac{1}{n}\sum_j \cos(x_j^{M-1} - x_n^{M-1}) \end{bmatrix} \tag{39}$$

And then, we can calculate $\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1}$

$$\frac{\partial \mathbf{Z}^M}{\partial \mathbf{Z}^1} = \mathbf{I} + \Delta t \sum_{i=1}^{M} \mathbf{E}_{i-1} + \mathcal{O}(\Delta t^2) \tag{40}$$

With the same manner, we can rewrite $\frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0}$ as

$$\frac{\partial \mathbf{Z}^1}{\partial \mathbf{Z}^0} = \mathbf{I} + \Delta t \mathbf{E}' \tag{41}$$

$$\mathbf{E}' = \begin{bmatrix} 1 - \frac{1}{n}\sum_j \cos(x_j^0 - x_n^0) & \frac{1}{n}\cos(x_2^0 - x_1^0) & \cdots & \frac{1}{n}\cos(x_n^0 - x_1^0) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\cos(x_1^0 - x_n^0) & \frac{1}{n}\cos(x_2^0 - x_n^0) & \cdots & 1 - \frac{1}{n}\sum_j \cos(x_j^0 - x_n^0) \end{bmatrix} \tag{42}$$

Then we can obtain proposition 4.4 by multiplying (26), (40), (41) together with $\frac{\partial \mathbf{Z}^0}{\partial \mathbf{W}} = \mathbf{V}$.

8 ON THE GENERALIZATION PERFORMANCE OF KURAMOTOGNN

Given a space Z and a fixed distribution D on Z. Let G be a class of hypothesis functions: $h : Z \to \mathbb{R}^{num\_classes}$. Given a loss function, say, $l(h; z)$, whose first and second arguments are a hypothesis and input, respectively, we define $L(h)$ for the *predictive loss*:

$$L_D[h] = \mathbf{E}_{z \sim D}[l(h, z)]$$

which is the expectation of a loss function with a hypothesis h over a distribution D of datasets. Similarly, given a set of examples $S = (z_1, \ldots, z_m)$ drawn i.i.d from D, writes $L_S(g)$ for the *empirical loss*:

$$L_S[h] = \frac{1}{m}\sum_{i=1}^{m}[l(h, z_i)]$$

In statistical learning theory, our focus lies in determining the bound between estimated error (empirical loss) and true error (predictive loss) across all functions in H. Smaller bound is better, since it means that the true error

of a classifier is not much higher than its estimated error, and so selecting a classifier that has a low estimated error will ensure that the true error is also low.

In order to finding such bound, we will need a complexity measure for the class of hypothesis functions H. To this end, let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$ be a list of independent random variables, where, $P(\sigma_i = +1) = P(\sigma_i = -1) = 1/2$. Then *the empirical Rademacher complexity* of l and H with respect to S is defined to be

$$R(l \circ H \circ S) = \mathbf{E}_{\boldsymbol{\sigma}} \left[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i l(h, z_i) \right] \tag{43}$$

Then for any integer $m \geq 1$, *the Radamacher complexity* of H with respect to samples size m drawn according to D is

$$R_{D,m} = \mathbf{E}_{S \sim D^m} \left[R(l \circ H \circ S) \right] \tag{44}$$

Remark 8.1. *Intuitively, the empirical Rademacher complexity measures how well the class of functions H correlates with randomly generated labels on the set S. The richer the class of functions H the better the chance of finding $h \in H$ that correlates with a given σ, and hence the larger empirical Rademacher complexity.*

Suppose that we are given a dataset $\{(z_i, \hat{y}_i)\}_{i=1}^m$ where m is the number of observable nodes in the graph, z_i and $\hat{y}_i$ are the node feature vector, and label of i^{th} node, respectively. H is the hypothesis set of KuramotoGNN. The following Proposition indicates the generalization performance of KuramotoGNN with sufficient training sample size.

Proposition 8.1. *Given H is the hypothesis set of KuramotoGNN. If a sufficiently large sample is drawn from distribution D, then with high probability $L_D[h]$ and $L_S[h]$ are not too far apart for all functions $h \in H$:*

$$\lim_{m \to +\infty} (L_D[h] - L_S[h]) = 0 \tag{45}$$

Proof. The following bound holds with at least probability $1 - \delta$, which is well known as the **Rademacher-based uniform convergence**.

$$L_D[h] - L_S[h] \leq 2\mathbf{E}_{S \sim D^m} \left[R(l \circ H \circ S) \right] + c\sqrt{\frac{2\log(2/\delta)}{m}} \tag{46}$$

where $c > 0$ is a constant, m is the sample size, $h \in H$. Now, we estimate the first term of the RHS. Following the training objective in Section 3 of the **SM**, we used a linear discriminator for classification. Hence, in the case of binary classification, $\mathbf{D} \in \mathbb{R}^{1 \times d}$. Hereafter, we use a notation

$$H \circ S \equiv \{\mathbf{D}x(T; z_1) + b, \ldots, \mathbf{D}x(T; z_m) + b\} \subset \mathbb{R}$$

with T being the terminating time of the ODE. It is known that

$$R(l \circ H \circ S) \leq \rho R(H \circ S)$$

where ρ is the Lipschitz coefficient of l. Thus, it is enough now to estimate $R(H \circ S)$. Under assumption

$\|\mathbf{D}\| < +\infty$, this can be done as follows.

$$
\begin{aligned}
mR(H \circ S) &= \mathbf{E}_\sigma \left[\sup_{\hat{y}} \sum_{i=1}^{m} \sigma_i \hat{y} \right] \\
&= \mathbf{E}_\sigma \left[\sup \sum_{i=1}^{m} \sigma_i (\mathbf{D} x(T; z_i) + b) \right] \\
&\leq \|\mathbf{D}\|_\infty \mathbf{E}_\sigma \left[\sup_{\mathbf{D}} \sum_{i=1}^{m} |\sigma_i x(T; z_i)| \right] + \mathbf{E}_\sigma \left[\sup_{b} \sum_{i=1}^{m} \sigma_i b \right] \\
&\leq \|\mathbf{D}\|_\infty \left(\mathbf{E}_\sigma \left[\left\| \sum_{i=1}^{m} \sigma_i x(T; z_i) \right\|^2 \right] \right)^{\frac{1}{2}} + \sup_b b \left(\mathbf{E}_\sigma \left[\sum_{i=1}^{m} \sigma_i \right] \right) \qquad \text{(Jensen's inequality)} \\
&\leq \|\mathbf{D}\|_\infty \left(\mathbf{E}_\sigma \left[\sum_{i=1}^{m} \|x(T; z_i)\|^2 \right] \right)^{\frac{1}{2}}, \qquad (\mathbf{E}_\sigma[\sigma_i] = 0) \\
&\leq \|\mathbf{D}\|_\infty \left(m \|x_M(T; z_i)\|^2 \right)^{\frac{1}{2}} \qquad x_M = \text{argmax } x_i
\end{aligned}
$$

Thus, the problem reduces to the estimate of $\|x_i(T; z_i)\|^2$. Recall that the solution to equation (10) in the main text is:

$$x_i(T; z_i) = (1 + T)x_i(0) + K \int_0^T \sum_{j=1}^{n} a_{ij} \sin(x_j(t) - x_i(t))\, dt$$

Together with $|a_{ij}| \leq 1$, we have

$$\|x_i(T; z_i)\| \leq (1 + T)x_i(0) + nTK$$

Combining these, we find that the RHS of (46) tends to 0 as $m \to +\infty$, and therefore, with sufficient sample size, the KuramotoGNN training process is consistent with predictive loss. □

References

[Calder et al., 2020] Calder, J., Cook, B., Thorpe, M., and Slepcev, D. (2020). Poisson learning: Graph based semi-supervised learning at very low label rates. In *International Conference on Machine Learning*, pages 1306–1316. PMLR.

[Chen et al., 2020] Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020). Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR.

[Chen et al., 2018] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

[Hu et al., 2020] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

[Kuramoto, 1975] Kuramoto, Y. (1975). Self-entrainment of a population of coupled non-linear oscillators. In Araki, H., editor, *International Symposium on Mathematical Problems in Theoretical Physics*, pages 420–422, Berlin, Heidelberg. Springer Berlin Heidelberg.

[McAuley et al., 2015] McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.

[McCallum et al., 2000] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

[Namata et al., 2012] Namata, G., London, B., Getoor, L., Huang, B., and Edu, U. (2012). Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, page 1.

[Pei et al., 2020] Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*.

[Pfaff et al., 2020] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*.

[Rusch et al., 2022] Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., and Bronstein, M. (2022). Graph-coupled oscillator networks. In *ICML*, pages 18888–18909. PMLR.

[Sen et al., 2008] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

[Shchur et al., 2018] Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

[Strogatz, 2000] Strogatz, S. H. (2000). From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1):1–20.