ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Graph-Coupled Oscillator Networks

T. K. Rusch and B. P. Chamberlain and J. Rowbottom and S. Mishra and

M. M. Bronstein

Research Report No. 2022-04
February 2022

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Funding ERC: 770880 COMANFLO

Graph-Coupled Oscillator Networks

T. Konstantin Rusch [*] Benjamin P. Chamberlain [†] James Rowbottom [†]
Siddhartha Mishra [*‡] Michael M. Bronstein [§†]

Abstract

We propose Graph-Coupled Oscillator Networks (GraphCON), a novel framework for deep learning on graphs. It is based on discretizations of a second-order system of ordinary differential equations (ODEs), which model a network of nonlinear forced and damped oscillators, coupled via the adjacency structure of the underlying graph. The flexibility of our framework permits any basic GNN layer (e.g. convolutional or attentional) as the coupling function, from which a multi-layer deep neural network is built up via the dynamics of the proposed ODEs. We relate the oversmoothing problem, commonly encountered in GNNs, to the stability of steady states of the underlying ODE and show that zero-Dirichlet energy steady states are not stable for our proposed ODEs. This demonstrates that the proposed framework mitigates the oversmoothing problem. Finally, we show that our approach offers competitive performance with respect to the state-of-the-art on a variety of graph-based learning tasks.

1 Introduction

Graph Neural Networks (GNNs) Sperduti (1994); Goller & Kuchler (1996); Sperduti & Starita (1997); Frasconi et al. (1998); Gori et al. (2005); Scarselli et al. (2008); Bruna et al. (2014); Defferrard et al. (2016); Kipf & Welling (2017); Monti et al. (2017); Gilmer et al. (2017) are a widely-used class of models for learning on relations and interaction data. These models have recently been successfully applied in a variety of tasks such as computer vision and graphics Monti et al. (2017), recommender systems Ying et al. (2018), transportation Derrow-Pinion et al. (2021), computational chemistry (Gilmer et al., 2017), drug discovery Gaudelet et al. (2021), physics (Shlomi et al., 2020), and analysis of social networks (see Zhou et al. (2019); Bronstein et al. (2021) for additional applications).

Several recent works proposed Graph ML models based on differential equations coming from physics Avelar et al. (2019); Poli et al. (2019b); Zhuang et al. (2020); Xhonneux et al. (2020b), including diffusion Chamberlain et al. (2021b) and wave Eliasof et al. (2021) equations and geometric equations such as Beltrami Chamberlain et al. (2021a) and Ricci Topping et al. (2021) flows. Such approaches allow not only to recover popular GNN models as discretization schemes for the underling differential equations, but also, in some cases, can address problems encountered in traditional GNNs such as oversmoothing Nt & Maehara (2019); Oono & Suzuki (2020) and bottlenecks Alon & Yahav (2021).

In this paper, we propose a novel physically-inspired approach to learning on graphs. Our framework, termed **GraphCON** (Graph-Coupled Oscillator Network) builds upon suitable time-discretizations of a specific class of ordinary differential equations (ODEs) that model the dyanmics of a network of non-linear forced and damped oscillators, which are coupled via the adjacency structure of the underlying graph. Graph-coupled oscillators are often encountered in mechanical, electronic, and biological systems, and have been studied extensively Strogatz (2015), with a prominent example being functional circuits in the brain such as cortical columns Stiefel & Ermentrout (2016). In these circuits, each neuron oscillates with periodic firing and spiking of the action potential. The network of neurons is coupled in the form of a graph, with neurons representing nodes and edges corresponding to synapses linking neurons.

[*]Seminar for Applied Mathematics (SAM), D-MATH, ETH Zürich, Switzerland
[†]Twitter Inc., London, UK
[‡]ETH AI Center, ETH Zürich
[§]Department of Computer Science, University of Oxford, UK

Main Contributions. In the subsequent sections, we will demonstrate the following features of GraphCON:

- GraphCON is flexible enough to accommodate any standard GNN layer (such as GAT or GCN) as its coupling function. As timesteps of our discretized ODE can be interpreted as layers of a deep neural network Chen et al. (2018); Haber & Ruthotto (2018); Chamberlain et al. (2021b), one can view GraphCON as a wrapper around any underlying basic GNN layer allowing to build deep GNNs. Moreover, we will show that standard GNNs can be recovered as steady states of the underlying class of ODEs, whereas GraphCON utilizes their dynamic behavior to sample a richer set of states, which could lead to better expressive power.
- We mathematically formulate the frequently encountered oversmoothing problem for GNNs Nt & Maehara (2019); Oono & Suzuki (2020) in terms of the stability of zero-Dirichlet energy steady states of the underlying equations. By a careful analysis of the dynamics of the proposed ODEs, we demonstrate that any zero-Dirichlet energy steady states are not (exponentially) stable. Consequently, we show that the oversmoothing problem for GraphCON is mitigated by construction.
- We provide an extensive empirical evaluation of GraphCON on a wide variety of graph learning tasks such as transductive and inductive node classification and graph regression and classification, demonstrating that GraphCON achieves competitive performance.

2 GraphCON

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E} \subseteq \mathcal{V})$ be an undirected graph with $|\mathcal{V}| = v$ nodes and $|\mathcal{E}| = e$ edges consisting of unordered pairs of nodes $\{i, j\}$ and denoted $i \sim j$. We will label nodes by the index $i \in \mathcal{V} = \{1, 2, \dots, v\}$. For any $i \in \mathcal{V}$, we denote its 1-*neighborhood* as $\mathcal{N}_i = \{j \in \mathcal{V} : i \sim j\}$. Furthermore, let $\mathbf{X} \in \mathbb{R}^{v \times m}$ be given by $\mathbf{X} = \{\mathbf{X}_i\}$ for $i \in \mathcal{V}$, denoting the m-dimensional feature vector at each node i.

Central to our framework is a *graph dynamical system* represented by the following nonlinear system of ODEs:

$$\mathbf{X}'' = \sigma(\mathbf{F}_\theta(\mathbf{X}, t) + \mathbf{X}\overline{\mathbf{W}} + \overline{\mathbf{b}}) - \gamma\mathbf{X} - \alpha\mathbf{X}'. \tag{1}$$

Here, $\mathbf{X}(t)$ denotes the time-dependent $v \times m$-matrix of node features, σ is the activation function, $\mathbf{F}_\theta$ is a general learnable (possibly time-dependent) 1-neighborhood coupling function of the form

$$(\mathbf{F}_\theta(\mathbf{X}, t))_i = \mathbf{F}_\theta\left(\mathbf{X}_i(t), \mathbf{X}_j(t), t\right) \quad \forall i \sim j, \tag{2}$$

parametrized with a set of learnable parameters θ. $\overline{\mathbf{W}} \in \mathbb{R}^{m \times m}$, $\overline{\mathbf{b}} \in \mathbb{R}^m$ are learnable weights and biases that act as a residual connection.

By introducing the auxiliary *velocity* variable $\mathbf{Y}(t) = \mathbf{X}'(t) \in \mathbb{R}^{v \times m}$, we can rewrite the second-order ODEs (1) as a first-order system:

$$\begin{aligned} \mathbf{Y}' &= \sigma(\mathbf{F}_\theta(\mathbf{X}, t) + \mathbf{X}\overline{\mathbf{W}} + \overline{\mathbf{b}}) - \gamma\mathbf{X} - \alpha\mathbf{Y}, \\ \mathbf{X}' &= \mathbf{Y}. \end{aligned} \tag{3}$$

The key idea of our framework is, given the input node features $\mathbf{X}(0)$ as an initial condition, to use the solution $\mathbf{X}(T)$ at some time T as the output (more generally, one can also apply (linear) transformations (embeddings) to $\mathbf{X}(0)$ and $\mathbf{X}(T)$). As will be shown in the following section, the space of solutions of our system is a rich class of functions that can solve many learning tasks on a graph.

The system (3) must be solved by an iterative numerical solver using a suitable time-discretization. It is highly desirable for a time-discretization to preserve the structure of the underlying ODEs (3) Hairer et al. (1987). In this paper, we use the following IMEX (implicit-explicit) time-stepping scheme, which extends the *symplectic Euler method* Hairer et al. (1987) to systems with an additional damping term,

$$\begin{aligned} \mathbf{Y}^n &= \mathbf{Y}^{n-1} + \Delta t[\sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1}, t^{n-1}) + \mathbf{X}^{n-1}\overline{\mathbf{W}} + \overline{\mathbf{b}}) - \gamma\mathbf{X}^{n-1} - \alpha\mathbf{Y}^{n-1}], \\ \mathbf{X}^n &= \mathbf{X}^{n-1} + \Delta t\mathbf{Y}^n, \end{aligned} \tag{4}$$

for $n = 1, \dots, N$, where $\Delta t > 0$ is a fixed time-step and $\mathbf{Y}^n, \mathbf{X}^n$ denote the hidden node features at time $t^n = n\Delta t$. The iterative scheme (4) can be interpreted as an N-layer graph neural network (with potential

additional linear input and readout layers, omitted here for simplicity), which we refer to as **GraphCON** (see section 3 for the motivation of this nomenclature). The coupling function $\mathbf{F}_\theta$ plays the role of a message passing mechanism (Gilmer et al. (2017), also referred to, in various contexts, as 'diffusion' or 'neighborhood aggregation') in traditional GNNs.

Choice of the coupling function $\mathbf{F}_\theta$. Our framework allows for any learnable 1-neighborhood coupling to be used as $\mathbf{F}_\theta$, including instances of message passing mechanisms commonly used in the Graph ML literature such as GraphSAGE (Hamilton et al., 2017), Graph Attention Velickovic et al. (2018), Graph Convolution Defferrard et al. (2016); Kipf & Welling (2017), SplineCNN (Fey et al., 2018), or MoNet (Monti et al., 2017)). In this paper, we focus on two particularly popular choices: *Attentional message passing* of Velickovic et al. (2018):

$$\mathbf{F}_\theta(\mathbf{X}^n, t^n) = \mathbf{A}^n(\mathbf{X}^n)\mathbf{X}^n\mathbf{W}^n,$$

with learnable weight matrices $\mathbf{W}^n \in \mathbb{R}^{m\times m}$ and attention matrices $\mathbf{A}^n \in \mathbb{R}^{n\times n}$ following the adjacency structure of the graph $\mathcal{G}$, i.e., $(\mathbf{A}^n(\mathbf{X}^n))_{ij} = 0$ if $j \notin \mathcal{N}_i$ and

$$(\mathbf{A}^n(\mathbf{X}^n))_{ij} = \frac{\exp(\mathrm{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}^n\mathbf{X}_i^n||\mathbf{W}^n\mathbf{X}_j^n]))}{\sum_{k\in\mathcal{N}_i}\exp(\mathrm{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}^n\mathbf{X}_i^n||\mathbf{W}^n\mathbf{X}_k^n]))},$$

otherwise (here $\mathbf{X}_i^n$ denotes the i-th row of $\mathbf{X}^n$ and $\mathbf{a} \in \mathbb{R}^{2m}$).

We refer to (4) based on this attentional 1-neighborhood coupling as **GraphCON-GAT**.

Graph convolution operator of Kipf & Welling (2017):

$$\mathbf{F}_\theta(\mathbf{X}^n, t^n) = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}^n\mathbf{W}^n, \tag{5}$$

with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denoting the adjacency matrix of $\mathcal{G}$ with inserted self-loops, diagonal degree matrix $\mathbf{D} = \mathrm{diag}(\sum_{l=1}^n \hat{\mathbf{A}}_{kl})$, and $\mathbf{W}_i^n \in \mathbb{R}^{m\times m}$ being learnable weight matrices. We refer to (4) based on this convolutional 1-neighborhood coupling as **GraphCON-GCN**.

Steady States of GraphCON and relation to GNNs. It is straightforward to see that the steady states $\mathbf{X}^*, \mathbf{Y}^*$ of the GraphCON dynamical system (4) with an *autonomous* coupling function $\mathbf{F}_\theta = \mathbf{F}_\theta(\mathbf{X})$ (as in GraphCON-GAT or GraphCON-GCN) are given by $\mathbf{Y}^* \equiv \mathbf{0}$ and

$$\mathbf{X}^* = \frac{\Delta t}{\gamma}\sigma(\mathbf{F}_\theta(\mathbf{X}^*)). \tag{6}$$

Using a simple fixed point iteration to find the steady states (6) yields a multi-layer GNN of the form;

$$\mathbf{X}^n = \frac{\Delta t}{\gamma}\sigma(\mathbf{F}_\theta(\mathbf{X}^{n-1})), \quad \text{for } n = 1, 2, \ldots, N. \tag{7}$$

We observe that (up to a rescaling by the factor $\Delta t/\gamma$) equation (7) corresponds to the update formula for any standard N-layer message-passing GNN Gilmer et al. (2017), including such popular variants as GAT Velickovic et al. (2018) or GCN Kipf & Welling (2017).

Thus, this interpretation of GraphCON (4) clearly brings out its relationship with standard GNNs. Unlike in standard multi-layer GNNs of the generic form (7) that can be thought of as steady states of the underlying ODEs (3), GraphCON evolves the underlying node features *dynamically in time*. Interpreting the multiple GNN layers as iterations at times $t^n = n\Delta t$ in (4), we observe that the node features in GraphCON follow the trajectories of the corresponding dynamical system and can explore a richer sampling of the underlying latent feature space, leading to possibly greater expressive power than standard GNNs (7), which might remain in the vicinity of steady states.

Moreover, this interpretation also reveals that, in principle, any GNN of the form (7) can be used within the GraphCON framework, offering a very flexible and broad class of architectures. Hence, one can think of GraphCON as an *additional wrapper* on top of any basic GNN layer allowing for a principled and stable design of deep multi-layered GNNs. In the following Section 3, we show that such an approach has several key advantages over standard GNNs.

3 Properties of GraphCON

To gain some insight into the functioning of GraphCON (4), we start by assuming no residual connections (i.e., $\overline{\mathbf{W}} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$), setting the hyperparameter $\gamma = 1$ and assuming that the 1-neighborhood coupling $\mathbf{F}_\theta$ is given by either the GAT or GCN type coupling functions. In this case, the underlying ODEs (3) takes the following node-wise form,

$$\begin{aligned} \mathbf{X}_i' &= \mathbf{Y}_i, \\ \mathbf{Y}_i' &= \sigma\left(\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\mathbf{X}_j\right) - \mathbf{X}_i - \alpha\mathbf{Y}_i, \end{aligned} \tag{8}$$

for all nodes $i \in \mathcal{V}$, with $\mathbf{A}_{ij} = \mathbf{A}\left(\mathbf{X}_i(t), \mathbf{X}_j(t)\right) \in \mathbb{R}$ stemming from the attention or convolution operators. Furthermore, the matrices are *right stochastic* i.e., the entries satisfy,

$$\begin{aligned} &0 \le \mathbf{A}_{ij} \le 1, \quad \forall j \in \mathcal{N}_i, \quad \forall i \in \mathcal{V}, \\ &\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij} = 1, \quad \forall i \in \mathcal{V}. \end{aligned} \tag{9}$$

Uncoupled case. The simplest case of (8), corresponds to setting $\sigma \equiv 0$ and $\alpha = 0$. In this case, all nodes are *uncoupled* from each other and the solutions of the resulting ODEs are of the form,

$$\mathbf{X}_i(t) = \mathbf{X}_i(0)\cos(t) + \mathbf{Y}_i(0)\sin(t). \tag{10}$$

Thus, the dynamics of the ODEs (3) in this special case correspond to a *system of uncoupled oscillators*, with each node oscillating at unit frequency.

Coupled linear case. Next, we introduce coupling between the nodes that are adjacent on the underlying graph $\mathcal{G}$ and assume identity activation function $\sigma(x) = x$. In this case, (8) is a *coupled linear system* and an exact closed form solution, such as (10) may not be possible. However, we can describe the dynamics of (8) in the form of the following proposition (proved in **SM** C.1),

Proposition 3.1. *Let the node features* $\mathbf{X}, \mathbf{Y}$ *evolve according to the ODEs* (8) *with activation function* $\sigma = \mathrm{id}$ *and time-independent matrix* $\mathbf{A}$ *(e.g.* $\mathbf{A}_{ij} = \mathbf{A}(\mathbf{X}_i(0), \mathbf{X}_j(0))$ *using the initial features). Further assume that* $\mathbf{A}$ *is* symmetric *and* $\alpha = 0$*. Then*

$$\begin{aligned} &\sum_{i\in\mathcal{V}} \|\mathbf{Y}_i(t)\|^2 + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\|\mathbf{X}_i(t) - \mathbf{X}_j(t)\|^2 \\ &= \sum_{i\in\mathcal{V}} \|\mathbf{Y}_i(0)\|^2 + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\|\mathbf{X}_i(0) - \mathbf{X}_j(0)\|^2, \end{aligned} \tag{11}$$

holds for all $t > 0$.

Thus, in this case, we have shown that the dynamics of the underlying ODEs (8) preserves the *energy*,

$$\mathscr{E}(t) := \sum_{i\in\mathcal{V}} \|\mathbf{Y}_i(t)\|^2 + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\|\mathbf{X}_i(t) - \mathbf{X}_j(t)\|^2, \tag{12}$$

and the trajectories of (8) are constrained to lie on a manifold of the node feature space, defined by the level sets of the energy. In particular, energy (12) is not produced or destroyed but simply redistributed among the nodes of the underlying graph $\mathcal{G}$. Thus, the dynamics of (3) in this setting amounts to the motion of a *linear system of coupled oscillators*.

General nonlinear case. In the general case, we have (i) a nonlinear activation function σ; (ii) time-dependent non-linear coefficients $\mathbf{A}_{ij} = \mathbf{A}(\mathbf{X}_i(t), \mathbf{X}_j(t))$; and (iii) possible unsymmetrical entries $\mathbf{A}_{ij} \neq \mathbf{A}_{ji}$. All these factors destroy the energy conservation property (11) and can possibly lead to unbounded growth of the energy. Hence, we need to add some damping to the system. To this end,

the damping term in (8) is activated by setting $\alpha > 0$. Moreover, the residual terms corresponding to non-zero weights $\overline{\mathbf{W}}$ and biases $\overline{\mathbf{b}}$ in (3) can be interpreted as *forcing* or *impulse* terms. Similarly, $\gamma \neq 1$ corresponds to controlling frequencies of the nodes. Thus, the overall dynamics of the underlying ODEs (3) amounts to the motion of a nonlinear system of *coupled, controlled, damped, and forced oscillators* with the coupling structure being that of the underlying graph. This explains our choice of the name, *Graph-Coupled Oscillatory Neural Network* or 'GraphCON' for short.

We illustrate the dynamics of GraphCON in Fig. 1, where the model is applied to the graph of a molecule from the ZINC database Irwin et al. (2012), with features $\mathbf{X}$ denoting the position of the nodes and they are propagated in time through the action of GraphCON (4). The oscillatory behavior of the node features, as well as their dependence on the adjacency structure of the underlying graph can be clearly observed in this figure.

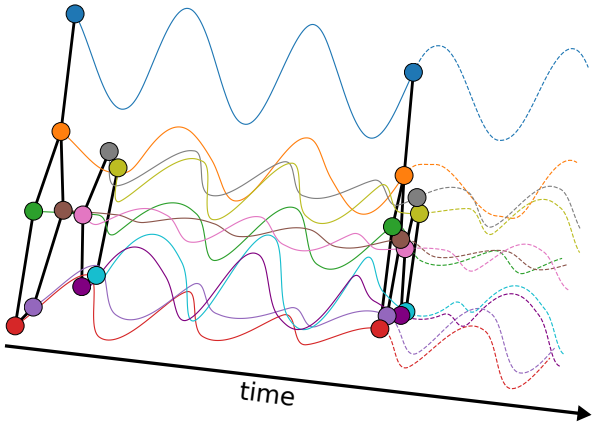

Figure 1: Illustration of GraphCON dynamics on a ZINC molecular graph. The initial positions of GraphCON ($\mathbf{X}_0$ in (4)) are represented by the 2-dimensional positions of the nodes, while the initial velocities ($\mathbf{Y}_0$ in (4)) are set to the initial positions. The positions are propagated forward in time ('layers') using GraphCON-GCN with random weights. The molecular graph is plotted at initial time $t = 0$ as well as at $t = 20$.

Oversmoothing and GraphCON. One of the common plights of GNN models such as GAT Velickovic et al. (2018), GCN Kipf & Welling (2017) and their variants is *oversmoothing* Nt & Maehara (2019); Oono & Suzuki (2020), a phenomenon where all node features in a deep GNN converge to the same constant value as the number of hidden layers is increased. Consequently, one often must resort to shallow GNNs at the expense of expressive power Nt & Maehara (2019); Oono & Suzuki (2020). Many attempts have been made in recent years to mitigate the oversmoothing problem for GNNs, including regularization procedures such as DropEdge Rong et al. (2020), using intermediate representations Xu et al. (2018b), or adding residual connections Chen et al. (2020).

We will show that GraphCON allows to mitigate this problem by construction, and set off by formulating this problem in precise mathematical terms and to this end, we recall the *Dirichlet energy*, defined on the node features $\mathbf{X}$ of an undirected graph $\mathcal{G}$ as,

$$\mathbf{E}(\mathbf{X}) = \frac{1}{v} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \|\mathbf{X}_i - \mathbf{X}_j\|^2. \tag{13}$$

Next, we define oversmoothing as follows:

Definition 3.2. *Let $\mathbf{X}^n$ denote the hidden features of the nth layer of an N-layer GNN, with $n = 0, \ldots, N$. We define* oversmoothing *as the exponential convergence to zero of the layer-wise Dirichlet energy as a function of n, i.e.,*

$$\mathbf{E}(\mathbf{X}^n) \leq C_1 e^{-C_2 n}, \tag{14}$$

with some constants $C_1, C_2 > 0$.

In other words, oversmoothing happens when the graph gradients vanish quickly (see for instance the illustration in Fig. 2) in the number of hidden layers of the GNN. As a result, the feature vectors across all nodes rapidly (exponentially) converge to the same constant value. This behavior is commonly observed in GNNs and is identified as one of the reasons for the difficulty in designing deep GNNs.

GraphCON behaves rather differently and allows to mitigate the oversmoothing problem in the sense of definition 3.2. To see this, we focus on the underlying ODEs (3). It is trivial to extend the definition of oversmoothing from the discrete case to the continuous one by requiring that oversmoothing happens for the ODEs (3) if the Dirichlet energy behaves as,

$$\mathbf{E}(\mathbf{X}(t)) \leq C_1 e^{-C_2 t}, \quad \forall t > 0, \tag{15}$$

for some $C_{1,2} > 0$.

We have the following simple proposition (proved in **SM** C.2) that characterizes the oversmoothing problem for the underlying ODEs in the standard terminology of dynamical systems Wiggins (2003),

Proposition 3.3. *The oversmoothing problem occurs for the ODEs* (3) *if and only if the hidden states* $(\mathbf{X}^*, \mathbf{Y}^*) = (\mathbf{c}, \mathbf{0})$ *are* exponentially stable steady states (fixed points) *of the ODE* (3)*, for some* $\mathbf{c} \in \mathbb{R}^m$ *and* $\mathbf{0}$ *being the* m*-dimensional vector with zeroes for all its entries.*

In other words, all the trajectories of the ODE (3), that start within the corresponding basin of attraction, have to converge exponentially fast in time (satisfy (15)) to the corresponding steady state $(\mathbf{c}, \mathbf{0})$ for the oversmoothing problem to occur for this system. Note that the basins of attraction will be different for different values of $\mathbf{c}$.

Given this characterization, the key questions are a) whether $(\mathbf{c}, \mathbf{0})$ are fixed points for the ODE (3), and b) whether these fixed points are exponentially stable. We answer these questions for the ODEs (8) in the following

Proposition 3.4. *Assume that the activation function* σ *in the ODEs* (8) *is ReLU. Then, for any* $\mathbf{c} \in \mathbb{R}^m$ *such that each entry of the vector* $\mathbf{c}_\ell \geq 0$*, for all* $1 \leq \ell \leq m$*, the hidden state* $(\mathbf{c}, \mathbf{0})$ *is a steady state for the ODEs* (8)*. However under the additional assumption of* $\alpha \geq \frac{1}{2}$*, this fixed point is* not exponentially stable.

The fact that $(\mathbf{c}, \mathbf{0})$ is a steady state of (8), for any *positive* $\mathbf{c}$ is straightforward to see from the structure of (8) and the definition of the ReLU activation function. We can already observe from the energy identity (11) for the simplified symmetric linear system that the energy (12) for the small perturbations around the steady state $(\mathbf{c}, \mathbf{0})$ is conserved in time. Hence, these small perturbations do not decay at all, let alone, exponentially fast in time. Thus, these steady states are not exponentially stable.

An extension of this analysis to the nonlinear time-dependent, possibly non-symmetric system (8) is more subtle and the proof relies on the identity (21) (expressed in Proposition C.1 in **SM** C.3) that describes how a suitably defined energy of the general system (8) evolves around small perturbations of the steady state $(\mathbf{c}, \mathbf{0})$. A careful analysis of this identity reveals that these small perturbations can grow polynomially in time (at least for short time periods) and do not decay exponentially. Consequently, the fixed point $(\mathbf{c}, \mathbf{0})$ is not stable. This shows that the oversmoothing problem, in the sense of definition 3.2, is mitigated for the ODEs (3) and structure preserving time-discretizations of it such as (4), from which, in simple words it follows that GraphCON *mitigates oversmoothing by construction.*

This analysis also illustrates the rich dynamics of (3) as we show that even if the trajectories reach a steady state of the form $(\mathbf{c}, \mathbf{0})$, very small perturbations will grow and the trajectory will veer away from this steady state, possibly towards other constant steady states which are also not stable. Thus, the trajectories can sample large parts of the latent space, contributing to the expressive power of the model.

4 Related Work

Differential equations have historically played a role in designing and interpreting various algorithms in machine learning, including non-linear dimensionality reduction methods Belkin & Niyogi (2003); Coifman & Lafon (2006) and ranking Page et al. (1999); Chakrabarti (2007) (all of which are related to closed-form solutions of diffusion PDEs). In the context of Deep Learning, differential equations have been used to derive various types of neural networks including Neural ODEs and their variants, that have been used

to design and interpret residual Chen et al. (2018) and convolutional Haber & Ruthotto (2018) neural networks. These approaches have recently gained traction in Graph ML, e.g. with ODE-based models for learning on graphs Avelar et al. (2019); Poli et al. (2019b); Zhuang et al. (2020); Xhonneux et al. (2020b).

Chamberlain et al. (2021b) used parabolic diffusion-type PDEs to design GNNs using graph gradient and divergence operators as the spatial differential operator, a transformer type-attention as a learnable diffusivity function ('1-neighborhood coupling' in our terminology), and a variety of time stepping schemes to discretize the temporal dimension in this framework. Chamberlain et al. (2021a) applied a non-euclidean diffusion equation ('Beltrami flow') to a joint positional-feature space, yielding a scheme with adaptive spatial derivatives ('graph rewiring'), and Topping et al. (2021) studied a discrete geometric PDE similar to Ricci flow to improve information propagation in GNNs. We can see the contrast between the diffusion-based methods of Chamberlain et al. (2021b,a) and GraphCON in the simple case of identity activation $\sigma(x) = x$ and no residual connection ($\overline{\mathbf{W}} = \mathbf{0}$ and $\mathbf{b} = \mathbf{0}$). Then, under the further assumption that the second-order time derivative $\mathbf{X}''$ is removed from (1) and $\alpha = \gamma = 1$, we recover the graph diffusion-PDEs of Chamberlain et al. (2021b). Hence, the presence of the temporal second-order derivative distinguishes this approach from diffusion-based PDEs.

Eliasof et al. (2021) proposed a GNN framework arising from a mixture of parabolic (diffusion) and hyperbolic (wave) PDEs on graphs with convolutional coupling operators, which describe dissipative wave propagation. We point out that a particular instance of their model (damped wave equation, also called as the *Telegrapher's equation*) can be obtained as a special case of our model (1) with the identity activation function and no residual connection. This is not surprising as the zero grid-size limit of oscillators on a regular grid yields a wave equation. However, given that we use a nonlinear activation function and the specific placement of the activation layer in (3), a local PDE interpretation of the general form of our underlying ODEs (1) does not appear to be feasible.

Finally, the explicit use of networks of coupled, controlled oscillators to design machine learning models was proposed in context of recurrent neural networks (RNNs) by Rusch & Mishra (2021a,b).

5 Experimental results

We present a detailed experimental evaluation of the proposed framework on a variety of graph learning tasks. We test two settings of GraphCON: GraphCON-GCN (using graph convolution as the 1-neighborhood coupling in (4)) and GraphCON-GAT (using the attentional coupling). Since in most experiments, these two configurations already outperform the state-of-the-art (SOTA), we only apply GraphCON with a more involved coupling functions if neither GraphCON-GCN nor GraphCON-GAT outperforms the current SOTA.

5.1 Evolution of Dirichlet Energy.

We start by illustrating the dynamics of the Dirichlet energy (13) of GraphCON for an undirected graph representing a 2-dimensional 10×10 regular grid with 4-neighbor connectivity. The node features $\mathbf{X}$ are randomly sampled from $\mathcal{U}([0, 1])$ and then propagated through 100-layer GNNs (with random weights): GAT, GCN, and their GraphCON-stacked versions (GraphCON-GAT and GraphCON-GCN) for two different values of the damping parameter $\alpha = 0, 0.5$ in (4) and with fixed $\gamma = 1$. In Fig. 2, we plot the (logarithm of) Dirichlet energy of each layer's output with respect to (logarithm) of the layer number. It can clearly be seen that GAT and GCN suffer from the oversmoothing problem as the Dirichlet energy converges exponentially fast to zero, indicating that the node features become constant, while GraphCON is devoid of this behavior. This holds true even for non-zero value of the damping parameter α, where the Dirichlet energy stabilizes after an initial decay.

5.2 Transductive node classification

We evaluate GraphCON on both homophilic and heterophilic datasets, where high homophily implies that the features in a node are similar to those of its neighbors. The homophily level reported in Table 1 and Table 2 is the measure proposed by Pei et al. (2020).

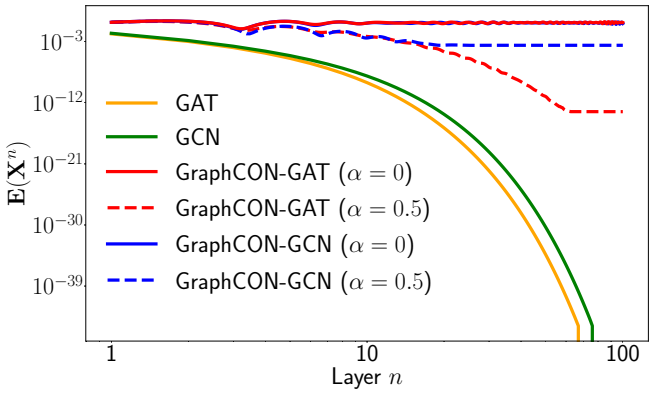

Figure 2: Dirichlet energy $\mathbf{E}(\mathbf{X}^n)$ of layer-wise node features $\mathbf{X}^n$ propagated through a GAT and GCN as well as their GraphCON-stacked versions (GraphCon-GAT and GraphCON-GCN) for two different values of $\alpha = 0, 0.5$ in (4) and fixed $\gamma = 1$.

Homophilic datasets. We consider three widely used node classification tasks, based on the citation networks Cora (McCallum et al., 2000), Citeseer (Sen et al., 2008) and Pubmed (Namata et al., 2012). We follow the evaluation protocols and training, validation, and test splits of Shchur et al. (2018); Chamberlain et al. (2021b), using only on the largest connected component in each network.

Table 1 compares GraphCON with standard GNN baselines: GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), MoNet (Monti et al., 2017), GraphSAGE (GS) (Hamilton et al., 2017), CGNN (Xhonneux et al., 2020a), GDE (Poli et al., 2019a), and GRAND Chamberlain et al. (2021b). We observe that GraphCON-GCN and GraphCON-GAT outperform pure GCN and GAT consistently. We also provide results for GraphCON based on the propagation layer used in GRAND i.e., transformer Vaswani et al. (2017) based graph attention, referred to as GraphCON-Tran, which also outperforms the basic underlying model. Overall, GraphCON models show the best performance on all these datasets.

Table 1: Transductive node classification test accuracy (MAP in %) on homophilic datasets. Mean and standard deviation are obtained using 20 random initializations on 5 random splits each. The three best performing methods are highlighted in **red** (First), **blue** (Second), and **violet** (Third).

Homophily level	**Cora** **0.81**	**Citeseer** **0.74**	**Pubmed** **0.80**
GAT-ppr	81.6 ± 0.3	68.5 ± 0.2	76.7 ± 0.3
MoNet	81.3 ± 1.3	71.2 ± 2.0	78.6 ± 2.3
GraphSage-mean	79.2 ± 7.7	71.6 ± 1.9	77.4 ± 2.2
GraphSage-maxpool	76.6 ± 1.9	67.5 ± 2.3	76.1 ± 2.3
CGNN	81.4 ± 1.6	66.9 ± 1.8	66.6 ± 4.4
GDE	78.7 ± 2.2	71.8 ± 1.1	73.9 ± 3.7
GCN	81.5 ± 1.3	71.9 ± 1.9	77.8 ± 2.9
GraphCON-GCN	81.9 ± 1.7	72.9 ± 2.1	**78.8 ± 2.6**
GAT	81.8 ± 1.3	71.4 ± 1.9	78.7 ± 2.3
GraphCON-GAT	**83.2 ± 1.4**	**73.2 ± 1.8**	**79.5 ± 1.8**
GRAND	**83.6 ± 1.0**	**73.4 ± 0.5**	**78.8 ± 1.7**
GraphCON-Tran	**84.2 ± 1.3**	**74.2 ± 1.7**	**79.4 ± 1.3**

Heterophilic datasets. We also evaluate GraphCON on the heterophilic graphs; Cornell, Texas and Wisconsin from the WebKB dataset[1]. Here, the assumption on neighbor feature similarity does not hold. Many GNN models were shown to struggle in this settings as can be seen by the poor performance of baseline GCN and GAT in Table 2. On the other hand, we see from Table 2 that not only do GraphCON-GCN and GraphCON-GAT dramatically outperform the underlying GCN and GAT models (e.g. for the most heterophilic Texas graph, GraphCON-GCN and GraphCON-GAT have mean accuracies of 85.4% and 82.2%, compared to accuracies of 55.1% and 52.2% for GCN and GAT), the GraphCON models also provide the best performance, outperforming recent baselines that are specifically designed for heterophilic graphs.

Table 2: Transductive node classification test accuracy (MAP in %) on heterophilic datasets. All results represent the average performance of the respective model over 10 fixed train/val/test splits, which are taken from Pei et al. (2020).

Homophily level	**Texas** **0.11**	**Wisconsin** **0.21**	**Cornell** **0.30**
GPRGNN	78.4 ± 4.4	82.9 ± 4.2	80.3 ± 8.1
H2GCN	$\mathbf{84.9 \pm 7.2}$	$\mathbf{87.7 \pm 5.0}$	$\mathbf{82.7 \pm 5.3}$
GCNII	77.6 ± 3.8	80.4 ± 3.4	77.9 ± 3.8
Geom-GCN	66.8 ± 2.7	64.5 ± 3.7	60.5 ± 3.7
PairNorm	60.3 ± 4.3	48.4 ± 6.1	58.9 ± 3.2
GraphSAGE	$\mathbf{82.4 \pm 6.1}$	81.2 ± 5.6	76.0 ± 5.0
MLP	80.8 ± 4.8	85.3 ± 3.3	81.9 ± 6.4
GAT	52.2 ± 6.6	49.4 ± 4.1	61.9 ± 5.1
GraphCON-GAT	82.2 ± 4.7	$\mathbf{85.7 \pm 3.6}$	$\mathbf{83.2 \pm 7.0}$
GCN	55.1 ± 5.2	51.8 ± 3.1	60.5 ± 5.3
GraphCON-GCN	$\mathbf{85.4 \pm 4.2}$	$\mathbf{87.8 \pm 3.3}$	$\mathbf{84.3 \pm 4.8}$

5.3 Inductive node classification

In this experiment, we consider the Protein-Protein-Interaction (PPI) dataset of Zitnik & Leskovec (2017), using the protocol of Hamilton et al. (2017). Table 3 shows the test performance (micro-average F1) of GraphCON and several standard GNN baselines. We can see that GraphCON significantly improves the performance of the underling models (GAT from 97.4% to 99.4% and GCN from 98.5% to 99.6%, which is the top result on this benchmark).

5.4 Molecular graph property regression

We reproduce the benchmark proposed in Dwivedi et al. (2020), regressing the constrained solubility of 12K molecular graphs from the ZINC dataset (Irwin et al., 2012). We follow verbatim the settings of Dwivedi et al. (2020); Beani et al. (2021): make no use of edge features and constrain the network sizes to ∼100K parameters. Table 4 summarizes the performance of GraphCON and standard GNN baselines. Both GraphCON-GAT and GraphCON-GCN outperform GAT and GCN respectively, by a factor of 2. Moreover, the performance of GraphCON-GCN is on par with the recent state-of-the-art method DGN (Beani et al., 2021) with significantly lower standard deviation. Given these results, it is instructive to ask why GraphCON models outperform their underlying base GNN models such as GCN. A part of the answer can be seen from **SM** Table 6, where the MAE for GCN and GraphCON-GCN for this task is shown for increasing number of layers. We observe from this table that while the MAE with GCN *increases* with the number of layers, the MAE for GraphCON-GCN *decreases monotonically* with increasing layers, allowing for the use of very deep GraphCON models with increased expressive power.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

Table 3: Test micro-averaged F_1 score on Protein-Protein Interactions (PPI) data set.

Model	Micro-averaged F_1
VR-GCN (Chen et al., 2017)	97.8
GraphSAGE (Hamilton et al., 2017)	61.2
PDE-GCN (Eliasof et al., 2021)	99.2
GCNII (Chen et al., 2020)	**99.5**
Cluster-GCN (Chiang et al., 2019)	**99.4**
GeniePath Liu et al. (2019)	98.5
JKNet (Xu et al., 2018b)	97.6
GAT (Velickovic et al., 2018)	97.3
GraphCON-GAT	**99.4**
GCN (Kipf & Welling, 2017)	98.5
GraphCON-GCN	**99.6**

Table 4: Test mean absolute error (MAE, averaged over 4 runs on different initializations) on ZINC (**without edge features, small 12k version**) restricted to small network sizes of $\sim 100k$ parameters. Baseline results are taken from Beani et al. (2021).

Model	Test MAE
GIN (Xu et al., 2018a)	0.41 ± 0.008
GatedGCN (Bresson & Laurent, 2017)	0.42 ± 0.006
GraphSAGE Hamilton et al. (2017)	0.41 ± 0.005
MoNet (Monti et al., 2017)	0.41 ± 0.007
PNA (Corso et al., 2020)	$\mathbf{0.32 \pm 0.032}$
DGN (Beani et al., 2021)	$\mathbf{0.22 \pm 0.010}$
GCN (Kipf & Welling, 2017)	0.47 ± 0.002
GraphCON-GCN	$\mathbf{0.22 \pm 0.004}$
GAT (Velickovic et al., 2018)	0.46 ± 0.002
GraphCON-GAT	$\mathbf{0.23 \pm 0.004}$

5.5 MNIST Superpixel graph classification

This experiment, first suggested by Monti et al. (2017), is based on the MNIST dataset (LeCun et al., 1998), where the grey-scale images are transformed into irregular graphs, as follows: the vertices in the graphs represent superpixels (large blobs of similar color), while the edges represent their spatial adjacency. Each graph has a fixed number of 75 superpixels (vertices). We use the standard splitting of using 55K-5K-10K for training, validation, and testing.

Table 5 shows that GraphCON-GCN dramatically improves the performance of a pure GCN (test accuracy of 88.89% vs 98.70%). We stress that both models share the parameters over all layers, i.e. GraphCON-GCN does not have more parameters despite being a deeper model. Thus, the better performance of GraphCON-GCN over GCN can be attributed to the use of more 'layers' (iterations) and not to a higher number of parameters (see **SM** Table 7 for accuracy vs. number of layers for this testcase). Finally, Table 5 also shows that GraphCON-GAT outperforms all other methods, including the recently proposed PNCNN Finzi et al. (2021), reaching a nearly-perfect test accuracy of 98.91%.

Table 5: Test accuracy in % on MNIST Superpixel 75.

Model	Test accuracy
ChebNet (Defferrard et al., 2016)	75.62
MoNet (Monti et al., 2017)	91.11
PNCNN (Finzi et al., 2021)	**98.76**
GatedGCN (Bresson & Laurent, 2017)	97.95
SplineCNN (Fey et al., 2018)	95.22
GCN (Kipf & Welling, 2017)	88.89
GraphCON-GCN	**98.68**
GAT (Velickovic et al., 2018)	96.19
GraphCON-GAT	**98.91**

6 Conclusions

In conclusion, we proposed a novel framework for designing deep Graph Neural Networks called GraphCON, based on suitable time discretizations of ODEs (1) that model the dynamics of a network of forced and damped oscillators. The coupling between the nodes is conditioned on the structure of the underlying graph.

One can readily interpret GraphCON as a framework to propagate information through multiple layers of a deep GNN, where each hidden layer has the same structure as standard GNNs such as GAT, GCN etc. Unlike in canonical constructions of deep GNNs, which stack hidden layers in a straightforward iterative fashion (7), GraphCON stacks them in a more involved manner using the dynamics of the ODE (3). Hence, in principle, any GNN hidden layer can serve as the coupling function $\mathbf{F}_\theta$ in GraphCON (4), offering it as an attractive framework for constructing very deep GNNs.

The well-known oversmoothing problem for GNNs was described mathematically in terms of the stability of zero Dirichlet energy steady states of the underlying ODE (3). We showed that such zero Dirichlet energy steady states of (3), which lead to constant node features, are not (exponentially) stable. Even if a trajectory reaches a feature vector that is constant across all nodes, very small perturbations will nudge it away and the resulting node features will deviate from each other. Thus, by construction, we demonstrated that the oversmoothing problem, in the sense of definition 3.2, is mitigated for GraphCON.

Finally, we extensively test GraphCON on a variety of node- and graph-classification and regression tasks, including heterophilic datasets known to be challenging for standard GNN models. From these experiments, we observed that (i) GraphCON models significantly outperform the underlying base GNN such as GCN or GAT and (ii) GraphCON models are either on par with or outperform state-of-the-art models on these tasks. This shows that ours is a novel, flexible, easy to use framework for constructing deep GNNs with theoretical guarantees and solid empirical performance.

Limitations and Future work. The presented approach can be extended in various directions. First, the richer dynamics of the ODEs (3) underlying our model points to a potentially higher expressive power, as also corroborated by experimental results. A future detailed theoretical investigation of the expressivity of GraphCON will entail studying subtle questions of optimal control, in particular the *controllability* Sontag (1998); Lions (1988) of the underlying ODEs (3), as one has to prove that the trainable parameters in GraphCON can be adjusted such that the resulting trajectories of (3) could be steered to reach any target state in a rich enough hypothesis class. Relating such results to traditional approaches classifying the expressive power of GNNs by analogy to the Weisfeiler-Lehman hierarchy could be another interesting direction.

Second, we focused primarily on the widely-used GNNs such as GAT and GCN to define the 1-neighborhood coupling in GraphCON. Given the inherent flexibility of GraphCON, it would be interesting to deploy more sophisticated GNNs as the 1-neighborhood coupling to see if the expressivity of GraphCON is increased further for larger sized benchmarks, as it was in the case of transductive node classification

with a Transformer-type attention.

Third, GraphCON can also serve as a foundation for designing other physics-based GNNs such as models that possess a Hamiltonian structure. Finally, high-order discretizations of the underlying ODEs (3), such as the structure preserving Stormer-Verlet algorithm and high-order Runge-Kutta methods, can be considered as variants of GraphCON.

Acknowledgements.

The research of TKR and SM was performed under a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 770880). MB and JR are supported in part by ERC Grant No. 724228 (LEMAN).

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *ICML*, 2021.

Avelar, P. H. C., Tavares, A. R., , Gori, M., and Lamb, L. C. Discrete and continuous deep residual learning over graphs. *arXiv preprint*, 2019.

Beani, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. Directional graph networks. In *ICML*. PMLR, 2021.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv:1711.07553*, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Chakrabarti, S. Dynamic personalized pagerank in entity-relation graphs. In *WWW*, 2007.

Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., and Bronstein, M. Beltrami flow and neural diffusion on graphs. In *NeurIPS*, 2021a.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M. M., Webb, S., and Rossi, E. GRAND: graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1407–1418. PMLR, 2021b.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. *arXiv:1710.10568*, 2017.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *ICML*. PMLR, 2020.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *KDD*, 2019.

Coifman, R. R. and Lafon, S. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *arXiv:2004.05718*, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

Derrow-Pinion, A., She, J., Wong, D., Lange, O., Hester, T., Perez, L., Nunkesser, M., Lee, S., Guo, X., Battaglia, P. W., Gupta, V., Li, A., Xu, Z., Sanchez-Gonzalez, A., Li, Y., and Veličković, P. Traffic Prediction with Graph Neural Networks in Google Maps. 2021.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv:2003.00982*, 2020.

Eliasof, M., Haber, E., and Treister, E. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. In *NeurIPS*, 2021.

Fey, M., Lenssen, J. E., Weichert, F., and Müller, H. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *CVPR*, 2018.

Finzi, M. A., Bondesan, R., and Welling, M. Probabilistic numeric convolutional neural networks. In *9th International Conference on Learning Representations, ICLR*, 2021.

Frasconi, P., Gori, M., and Sperduti, A. A general framework for adaptive processing of data structures. *IEEE Trans. Neural Networks*, 9(5):768–786, 1998.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6), 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Goller, C. and Kuchler, A. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 1996.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *IJCNN*, 2005.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34, 2018.

Hairer, E., Norsett, S. P., and Wanner, G. *Solving ordinary differential equations I.* Springer, 1987.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

Lions, J.-L. Exact controllability, stabilization and perturbations for distributed systems. *SIAM Review*, 30, 1988.

Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., and Qi, Y. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, 2019.

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

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017.

Namata, G., London, B., Getoor, L., Huang, B., and EDU, U. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, pp. 1, 2012.

Nt, H. and Maehara, T. Revisiting graph neural networks: all we have is low pass filters. *arXiv:1812.08434v4*, 2019.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Technical report, 1999.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv:2002.05287*, 2020.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. *arXiv:1911.07532*, 2019a.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. pp. 6571–6583, 2019b.

Rong, Y., Huang, W., Xu, T., and Huang, J. Towards deep graph convolutional networks on node classification. In *ICLR*, 2020.

Rusch, T. K. and Mishra, S. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. In *ICLR*, 2021a.

Rusch, T. K. and Mishra, S. Unicornn: A recurrent model for learning very long time dependencies. In *ICML*, 2021b.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2008.

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

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv:1811.05868*, 2018.

Shlomi, J., Battaglia, P., and Vlimant, J.-R. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.

Sontag, E. D. *Mathematical Control Theory*. Springer, 1998.

Sperduti, A. Encoding labeled graphs by labeling RAAM. In *NIPS*, 1994.

Sperduti, A. and Starita, A. Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks*, 8(3):714–735, 1997.

Stiefel, K. M. and Ermentrout, G. B. Neurons as oscillators. *Journal of Neurophysiology*, 116:2950–2960, 2016.

Strogatz, S. *Nonlinear Dynamics and Chaos*. Westview, Boulder CO, 2015.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv:2111.14522*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *6th International Conference on Learning Representations, ICLR*, 2018.

Wiggins, S. *Introduction to nonlinear dynamical systems and chaos*. Springer, 2003.

Xhonneux, L.-P., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*. PMLR, 2020a.

Xhonneux, L.-p. A. C., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*, 2020b.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv:1810.00826*, 2018a.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *ICML*. PMLR, 2018b.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., , Li, C., and Sun, M. Graph neural networks: a review of methods and applications. *arXiv:1812.08434v4*, 2019.

Zhuang, J., Dvornek, N., Li, X., and Duncan, J. S. Ordinary differential equations on graph networks. *Technical Report*, 2020.

Zitnik, M. and Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

Supplementary Material for:
Graph-Coupled Oscillator Networks

A Further experimental results

A.1 Performance of GraphCON with respect to number of layers

As we have argued in the main text, GraphCON is designed to be a deep GNN architecture with many layers. Depth could enhance the expressive power of GraphCON and we investigate this issue in two of the datasets, presented in Section 5 of the main text. In both experiments, we will focus on the GraphCON-GCN model and compare and contrast its performance, with respect to increasing depth, with the baseline GCN model.

We start with the molecular graph property regression example for the ZINC dataset of Irwin et al. (2012). In Table 6, we present the mean absolute error (MAE) of the model on the test set with respect to increasing number of layers (up to 20 layers) of the respective GNNs. As observed from this table, the MAE with standard GCN increases with depth. On the other hand, the MAE with GraphCON decreases as more layers are added.

Table 6: Test mean absolute errors of GraphCON-GCN as well as its baseline model GCN on the ZINC task for different number of layers $N = 5, 10, 15, 20$.

Model	Layers			
	5	10	15	20
GraphCON-GCN	0.241	0.233	0.228	0.214
GCN	0.442	0.463	0.478	0.489

Next, we consider the MNIST Superpixel graph classification task and present the test accuracy with increasing depth (number of layers) for both GCN and GraphCON-GCN. As in the previous example, we observe that increasing depth leads to worsening of the test accuracy for GCN. On the other hand, the test accuracy for GraphCON-GCN increases as more layers (up to 32 layers) are added to the model. Thus, both experiments demonstrate that GraphCON leverages more depth to improve performance.

Table 7: Test accuracies in % of GraphCON-GCN as well as its baseline model GCN on the MNIST Superpixel 75 task for different number of layers $N = 4, 8, 16, 32$.

Model	Layers			
	4	8	16	32
GraphCON-GCN	97.78	98.51	98.55	98.68
GCN	88.09	87.26	86.78	85.67

A.2 Sensitivity of performance of GraphCON to hyperparameters α and γ

We recall that GraphCON, (4) of the main text, has two additional hyperparameters, namely the damping parameter $\alpha \geq 0$ and the frequency control parameter $\gamma > 0$. In Table 8, we present the values of α, γ that led to the best performance of the resulting GraphCON models. It is natural to ask how sensitive the performance of GraphCON is to the variation of these hyperparameters. To this end, we choose the MNIST Superpixel graph classification task and perform a sensitivity study of the GraphCON-GCN model with respect to these hyperparameters. First, we fix a value of $\gamma = 0.76$ (corresponding to the best

results in Table 8) and vary α in the range of $\alpha \in [0, 2]$. The results are plotted in Fig. 3 and show that the accuracy is extremely robust to a very large parameter range in α. Only for large values $\alpha > 1.6$, we see that the accuracy deteriorates when the damping is too high.

Next for this model and task, we fix $\alpha = 1$ (which provides the best performance as reported in Table 8) and vary $\gamma \in [0, 2]$. Again, for a large range of values corresponding to $\gamma \in [0.2, 2]$, the accuracy is very robust. However, for very small values of γ, the accuracy falls significantly. This is to be expected as the model loses its interpretation as system of oscillators for $\gamma \approx 0$.

Thus, these sensitivity results demonstrate that GraphCON performs very robustly with respect to variations of the parameters α, γ, within a reasonable range.

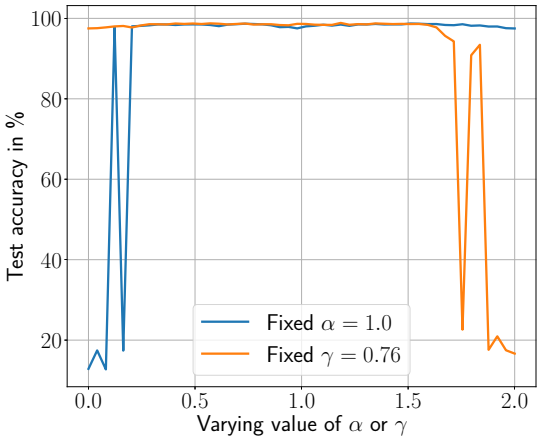

Figure 3: Sensitivity (measured as test accuracy) plot for α and γ hyperparameters of GraphCON-GCN (with 32 layers) trained on MNIST superpixel 75 experiment. First, $\alpha = 1.0$ is fixed and γ is varied in $[0, 2]$. Second, $\gamma = 0.76$ is fixed and α is varied in $[0, 2]$. The fixed α, γ are taken from the best performing GraphCON-GCN on the MNIST superpixel 75 task (Table 8)

B Training details

All experiments were run on NVIDIA GeForce GTX 1080 Ti, RTX 2080 Ti as well as RTX 2080 Ti GPUs. The tuning of the hyperparameters was done using a standard random search algorithm. We fix the time-step Δt in (4) to 1 in all experiments. The damping parameter α as well as the frequency control parameter γ are set to 1 for all Cora, Citeseer and Pubmed experiments, while we set them to 0 for all experiments based on the Texas, Cornell and Wisconsin network graphs. For all other experiments we include α and γ to the hyperparameter search-space. The tuned values can be found in Table 8.

C Mathematical details for Section 3 of main text

In this section, we provide details for the mathematical results in section 3 of the main text. We start with,

Table 8: Hyperparameters α and γ of GraphCON (4) for each best performing GraphCON model (based on a validation set).

Model	**Experiment**	α	γ
GraphCON-GCN	**PPI**	0.242	1.0
GraphCON-GAT		0.785	1.0
GraphCON-GCN	**ZINC**	0.215	1.115
GraphCON-GAT		1.475	1.324
GraphCON-GCN	**MNIST (superpixel)**	1.0	0.76
GraphCON-GAT		0.76	0.105

C.1 Proof of Proposition 3.1

Proof. We multiply $\mathbf{Y}_i^\top$ to the second equation of (8) and obtain,

$$\mathbf{Y}_i^\top \frac{d\mathbf{Y}_i}{dt} = \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\mathbf{Y}_i^\top (\mathbf{X}_j - \mathbf{X}_i), \quad \left(\text{as } \sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij} = 1\right)$$

Summing over $i \in \mathcal{G}$ and using the symmetry condition $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ in the above expression yields,

$$\begin{aligned}
\frac{d}{dt}\sum_{i\in\mathcal{G}} \frac{\|\mathbf{Y}_i\|^2}{2} &= -\sum_{i\in\mathcal{G}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij} (\mathbf{Y}_j - \mathbf{Y}_i)^\top (\mathbf{X}_j - \mathbf{X}_i), \\
&= -\sum_{i\in\mathcal{G}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij} \left(\frac{d(\mathbf{X}_j - \mathbf{X}_i)}{dt}\right)^\top (\mathbf{X}_j - \mathbf{X}_i) \\
\Rightarrow \frac{1}{2}\frac{d}{dt} &\left(\sum_{i\in\mathcal{G}} \frac{\|\mathbf{Y}_i\|^2}{2} + \sum_{i\in\mathcal{G}}\sum_{j\in\mathcal{N}_i} \mathbf{A}_{ij}\|\mathbf{X}_j - \mathbf{X}_i\|^2\right) = 0.
\end{aligned}$$

Integrating the last line in the above expression over time $[0, t]$ yields the desired identity (11)

□

C.2 Proof of Proposition 3.3

Proof. By the definition of the Dirichlet energy (13), (15) implies that,

$$\lim_{t\to\infty} \mathbf{X}_i(t) \equiv \mathbf{c}, \quad \forall i \in \mathcal{V}, \tag{16}$$

for some $\mathbf{c} \in \mathbb{R}^m$. In other words, all the hidden node features converge to the same feature vector $\mathbf{c}$ as time increases. Moreover, by (15), this convergence is exponentially fast.

Plugging in (16) in to the first equation of the ODE (3), we obtain that,

$$\lim_{t\to\infty} \mathbf{Y}_i(t) \equiv \mathbf{0}, \quad \forall i \in \mathcal{G}, \tag{17}$$

with $\mathbf{0}$ being the m vector with zeroes for all its entries. Thus, oversmoothing in the sense of definition 3.2, amounts to $(\mathbf{c}, \mathbf{0})$ being an exponentially stable fixed point (steady state) for the dynamics of (8)

On the other hand, if $(\mathbf{c}, \mathbf{0})$ is an exponentially stable steady state of (8), then the trajectories converge to this state exponentially fast satisfying (15). Consequently, by the definition of the Dirichlet energy (13), we readily observe that the oversmoothing problem, in the sense of definition 3.2, occurs in this case.

□

C.3 Proof of Proposition 3.4

The main aim of the section is to show that steady states of (8), of the form $(\mathbf{c}, \mathbf{0})$ are not exponentially stable.

To this end, we fix $\mathbf{c}$ and start by considering small perturbations around the fixed point $(\mathbf{c}, \mathbf{0})$. We define,

$$\hat{\mathbf{X}}_i = \mathbf{X}_i - \mathbf{c}, \hat{\mathbf{Y}}_i = \mathbf{Y}_i,$$

and evolve these perturbations by the linearized ODE,

$$\begin{aligned} \hat{\mathbf{X}}_i' &= \hat{\mathbf{Y}}_i, \\ \hat{\mathbf{Y}}_i{}' &= \sigma'(\mathbf{c}) \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{i,j} \hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i - \alpha \hat{\mathbf{Y}}_i, \end{aligned} \tag{18}$$

As $\sigma(x) = \max(x, 0)$ and $\mathbf{c} \geq 0$, we have that $\sigma'(\mathbf{c}) = ID$ and linearized system (19) reduces to,

$$\begin{aligned} \hat{\mathbf{X}}_i' &= \hat{\mathbf{Y}}_i, \\ \hat{\mathbf{Y}}_i{}' &= \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} \hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i - \alpha \hat{\mathbf{Y}}_i, \end{aligned} \tag{19}$$

with

$$\begin{aligned} \hat{\mathbf{A}}_{ij} = \mathbf{A}_{ij}(\mathbf{c}, \mathbf{c}), \quad \forall j \in \mathcal{N}_i, \quad \forall i \in \mathcal{G}, \\ 0 \leq \hat{A}_{ij} \leq 1, \quad \sum_{j \in \mathcal{N}_i} \hat{A}_{ij} = 1. \end{aligned} \tag{20}$$

We have the following proposition on the dynamics of linearized system (19) with respect to perturbations of the fixed point $(\mathbf{c}, \mathbf{0})$,

Proposition C.1. *Perturbations $\hat{\mathbf{X}}(t), \hat{\mathbf{Y}}(t)$ of the fixed point $(\mathbf{c}, \mathbf{0})$, which evolve according to (19) satisfy the following identity,*

$$\begin{aligned} &\frac{1}{v}\left(\sum_{i \in \mathcal{V}} \|\hat{\mathbf{Y}_i}(t)\|^2 + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \frac{\hat{\mathbf{A}}_{ij} + \hat{\mathbf{A}}_{ji}}{2} \left(\|\hat{\mathbf{X}}_j(t) - \hat{\mathbf{X}}_i(t)\|^2\right)\right) = T_1(t) + T_2(t) + T_3(t), \\ T_1(t) &= \frac{1}{v} \sum_{i \in \mathcal{V}} \left(\|\hat{\mathbf{Y}_i}(0)\|^2\right) e^{-2\alpha t} + \frac{1}{v} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \frac{\hat{\mathbf{A}}_{ij} + \hat{\mathbf{A}}_{ji}}{2} \left(\|\hat{\mathbf{X}}_j(0) - \hat{\mathbf{X}}_i(0)\|^2\right) e^{-2\alpha t} \\ T_2(t) &= \frac{2\alpha}{v} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \frac{\hat{\mathbf{A}}_{ij} + \hat{\mathbf{A}}_{ji}}{2} \int_0^t \|\hat{\mathbf{X}}_j(s) - \hat{\mathbf{X}}_i(s)\|^2 e^{2\alpha(s-t)} ds \\ T_3(t) &= \frac{1}{v} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \left(\hat{\mathbf{A}}_{ij} - \hat{\mathbf{A}}_{ji}\right) \int_0^t \left(\frac{\hat{\mathbf{Y}}_i(s) + \hat{\mathbf{Y}}_j(s)}{2}\right)^\top \left(\hat{\mathbf{X}}_j(s) - \hat{\mathbf{X}}_i(s)\right) e^{2\alpha(s-t)} ds \end{aligned} \tag{21}$$

Proof. Multiplying the second equation in (19) with $\hat{\mathbf{Y}}_i^\top$ and using the fact that $\sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} = 1$, we obtain,

$$\begin{aligned} &\frac{d}{dt} \frac{\|\hat{\mathbf{Y}}_i\|^2}{2} + \alpha \|\hat{\mathbf{Y}}_i\|^2 = \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} \hat{\mathbf{Y}}_i^\top \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right), \\ &= \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} \frac{\left(\hat{\mathbf{Y}}_i + \hat{\mathbf{Y}}_j\right)^\top}{2} \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right) - \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} \frac{\left(\hat{\mathbf{Y}}_j - \hat{\mathbf{Y}}_i\right)^\top}{2} \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right), \\ &= \sum_{j \in \mathcal{N}_i} \hat{\mathbf{A}}_{ij} \frac{\left(\hat{\mathbf{Y}}_i + \hat{\mathbf{Y}}_j\right)^\top}{2} \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right) - \sum_{j \in \mathcal{N}_i} \frac{\hat{\mathbf{A}}_{ij}}{2} \frac{d}{dt} \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right)^\top \left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right), \end{aligned} \tag{22}$$

where we have used the first equation of (19) in the last line of (22). Consequently, we have for all $i \in \mathcal{V}$,

$$\begin{aligned} &\frac{d}{dt}\frac{\|\hat{\mathbf{Y}}_i\|^2}{2} + \alpha\|\hat{\mathbf{Y}}_i\|^2 + \frac{d}{dt}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}}{2}\frac{\|\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\|^2}{2} \\ &\qquad = \sum_{j\in\mathcal{N}_i}\hat{\mathbf{A}}_{ij}\frac{\left(\hat{\mathbf{Y}}_i + \hat{\mathbf{Y}}_j\right)^\top}{2}\left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right) \end{aligned} \tag{23}$$

Summing (23) over all nodes $i \in \mathcal{V}$ yields,

$$\begin{aligned} &\frac{d}{dt}\sum_{i\in\mathcal{V}}\frac{\|\hat{\mathbf{Y}}_i\|^2}{2} + \alpha\sum_{i\in\mathcal{V}}\|\hat{\mathbf{Y}}_i\|^2 + \frac{d}{dt}\sum_{i\in\mathcal{G}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\frac{\|\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\|^2}{2} \\ &\qquad = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}-\hat{\mathbf{A}}_{ji}}{2}\left(\hat{\mathbf{Y}}_i + \hat{\mathbf{Y}}_j\right)^\top\left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right) \end{aligned} \tag{24}$$

Multiplying $e^{2\alpha t}$ to both sides of (24) and using the chain rule, we readily obtain,

$$\begin{aligned} &\frac{d}{dt}\sum_{i\in\mathcal{V}} e^{2\alpha t}\left(\frac{\|\hat{\mathbf{Y}}_i\|^2}{2} + \sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\frac{\|\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\|^2}{2}\right) \\ &= \alpha e^{2\alpha t}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\|\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\|^2 \\ &+ e^{2\alpha t}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}-\hat{\mathbf{A}}_{ji}}{2}\left(\hat{\mathbf{Y}}_i + \hat{\mathbf{Y}}_j\right)^\top\left(\hat{\mathbf{X}}_j - \hat{\mathbf{X}}_i\right) \end{aligned} \tag{25}$$

Integrating (25) over the time interval $[0, t]$ yields,

$$\begin{aligned} &\sum_{i\in\mathcal{V}}\left(\frac{\|\hat{\mathbf{Y}_i}(t)\|^2}{2}\right)e^{2\alpha t} + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\left(\frac{\|\hat{\mathbf{X}}_j(t) - \hat{\mathbf{X}}_i(t)\|^2}{2}\right)e^{2\alpha t} \\ &= \sum_{i\in\mathcal{V}}\left(\frac{\|\hat{\mathbf{Y}_i}(0)\|^2}{2}\right) + \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\left(\frac{\|\hat{\mathbf{X}}_j(0) - \hat{\mathbf{X}}_i(0)\|^2}{2}\right) \\ &+ \alpha\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}+\hat{\mathbf{A}}_{ji}}{2}\int_0^t\|\hat{\mathbf{X}}_j(s) - \hat{\mathbf{X}}_i(s)\|^2 e^{2\alpha s}ds \\ &+ \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\frac{\hat{\mathbf{A}}_{ij}-\hat{\mathbf{A}}_{ji}}{2}\int_0^t\left(\hat{\mathbf{Y}}_i(s) + \hat{\mathbf{Y}}_j(s)\right)^\top\left(\hat{\mathbf{X}}_j(s) - \hat{\mathbf{X}}_i(s)\right)e^{2\alpha s}ds \end{aligned} \tag{26}$$

We readily obtain the desired identity (21) from (26). □

Next, we observe that the right-hand side of the nonlinear ODEs (8) is *globally Lipschitz*. Therefore, solutions exist for all time $t > 0$, are unique and depend continuously on the data.

We assume that the initial perturbations around the steady state $(\mathbf{c}, \mathbf{0})$ are small i.e., they satisfy

$$\begin{aligned} \|\hat{\mathbf{X}}_i(0) - \hat{\mathbf{X}}_j(0)\| \le \epsilon, \quad \forall j \in \mathcal{N}_i, \quad \forall i \in \mathcal{G}, \\ \|\hat{\mathbf{Y}}_i(0)\| \le \epsilon, \quad \forall i \in \mathcal{G}, \end{aligned}$$

for some $0 < \epsilon << 1$.

Hence, there exists a small time $\tau > 0$ such that the time-evolution of these perturbations can be approximated to arbitrary accuracy by solutions of the linearized system (19).

Next, we see from the identity (21) that the evolution of the perturbations $\hat{\mathbf{X}}, \hat{\mathbf{Y}}$ from the fixed point $(\mathbf{c}, \mathbf{0})$ for the linearized system (19) is balanced by three terms $T_{1,2,3}$. The term T_1 is clearly a *dissipative* term and says that the initial perturbations are damped exponentially fast in time.

On the other hand, the term T_2, which has a positive sign, is a *production* term and says that the initial perturbations will *grow* with time t. Given the continuous dependence of the dynamics evolved by the ODE (19), there exists a time, still called τ by choosing it even smaller than the τ encountered before, such that

$$\begin{aligned} \|\hat{\mathbf{X}}_i(t) - \hat{\mathbf{X}}_j(t)\| \sim \mathcal{O}(\epsilon), \quad \forall j \in \mathcal{N}_i, \quad \forall i \in \mathcal{G}, \quad \forall t \in [0, \tau], \\ \|\hat{\mathbf{Y}}_i(t)\| \sim \mathcal{O}(\epsilon), \quad \forall i \in \mathcal{G}, \forall t \in [0, \tau]. \end{aligned} \tag{27}$$

Plugging the above expression into the term T_2 in (21) and using the right-stochatisticity of the matrix $\hat{\mathbf{A}}$, we obtain that,

$$T_2(t) \sim \mathcal{O}(\epsilon^2) \left(1 - e^{-2\alpha t}\right), \quad \forall t \leq \tau \tag{28}$$

Thus, the leading term is T_2 *grows* algebraically with respect to the initial perturbations.

Next we turn our attention to the term T_3 in (21). This term is proportional to the *asymmetry* in the graph-coupling matrix $\hat{\mathbf{A}} = \mathbf{A}(\mathbf{c}, \mathbf{c})$. If this matrix were symmetric, then T_3 vanishes. On the other hand, for many 1-neighborhood couplings considered in this article, the matrix $\hat{\mathbf{A}}$ is not symmetric. In fact, one can explicitly compute that for the GAT and Transformers attention and GCN-couplings, we have,

$$\hat{\mathbf{A}}_{ij} = \frac{1}{\deg(i)}, \quad \forall j \in \mathcal{N}_i, \quad \forall i \in \mathcal{V}. \tag{29}$$

Here, deg is refers to the degree of the node, with possibly inserted self-loops.

As the ordering of nodes of the graph $\mathcal{G}$ is arbitrary, we can order them in such a manner that $\hat{\mathbf{A}}_{ij} > \hat{\mathbf{A}}_{ji}$. Even with this ordering, as long as the matrix $\hat{\mathbf{A}}$ is not symmetric, the term T_3 is of *indefinite sign*. If it is positive, then we have additional growth with respect to time in (21). On the other hand, if T_3 is negative, it will have a *dissipative effect*. The rate of this dissipation can be readily calculated for a short time $t \leq \tau$ under the assumption (27) to be,

$$|T_3(t)| \sim \frac{\overline{D} - \underline{D}}{\overline{D}\underline{D}} \left(\frac{1 - e^{-2\alpha t}}{2\alpha}\right) \mathcal{O}(\epsilon^2). \tag{30}$$

Here, we define,

$$\overline{D} = \max_{i \in \mathcal{V}} \deg(i), \quad \underline{D} = \min_{i \in \mathcal{V}} \deg(i) \tag{31}$$

Thus by combining (28) with (30), we obtain,

$$T_2 + T_3 \sim \left(1 - \frac{\overline{D} - \underline{D}}{2\alpha \overline{D}\underline{D}}\right) \left(1 - e^{-2\alpha t}\right) \mathcal{O}(\epsilon^2) \tag{32}$$

In particular for $\alpha \geq 1/2$, we see from (32), that the overall balance (21) leads to an algebraic growth, rather than exponential decay, of the initial perturbations of the fixed point $(\mathbf{c}, \mathbf{0})$. Thus, we have shown that this steady state is not exponentially stable and small perturbations will take the trajectories of the ODE (19) away from this fixed point, completing the proof of Proposition 3.4.

Remark C.2. *We see from the above proof, the condition $\alpha \geq \frac{1}{2}$ is only a sufficient condition for the proof of Proposition 3.4, we can readily replace it by,*

$$\alpha \geq \frac{\overline{D} - \underline{D}}{2\overline{D}\underline{D}}$$