DYMAG: Rethinking Message Passing Using Dynamical-systems-based Waveforms

Dhananjay Bhaskar[1*] **Xingzhi Sun**[1*] **Yanlei Zhang**[2,3*] **Charles Xu**[1*] **Arman Afrasiyabi**[1]
Siddharth Viswanath[1] **Oluwadamilola Fasina**[1] **Guy Wolf**[2,3] **Michael Perlmutter**[4]
Smita Krishnaswamy[1]

[*]Equal Contribution [1]Yale University [2]Univ. de Montréal [3]Mila [4]Boise State University

Abstract

We present DYMAG, a graph neural network based on a novel form of message aggregation. Standard message-passing neural networks, which often aggregate local neighbors via mean-aggregation, can be regarded as convolving with a simple rectangular waveform which is non-zero only on 1-hop neighbors of every vertex. Here, we go beyond such local averaging. We will convolve the node features with more sophisticated waveforms generated using dynamics such as the heat equation, wave equation, and the Sprott model (an example of chaotic dynamics). Furthermore, we use snapshots of these dynamics at different time points to create waveforms at many effective scales. Theoretically, we show that these dynamic waveforms can capture salient information about the graph, including connected components, connectivity, and cycle structures. Empirically, we test DYMAG on both real and synthetic benchmarks to establish that DYMAG outperforms baseline models on recovery of graph persistence, generating parameters of random graphs, as well as property prediction for proteins, molecules and materials. Our code is available at `https://github.com/KrishnaswamyLab/DYMAG`.

1 Introduction

Message passing graph neural networks (GNNs) rely on aggregating signals via local averaging, which can be interpreted as convolving the node features with a simple, rectangular waveform that is non-zero only within one-hop neighborhoods of each vertex. It is known that this type of message-passing tends to suffer from over-smoothing if too many iterations are applied and from under-reaching if too few are applied [1–3]. One possible solution is to use multiscale message passing [4]. Another approach, [5–13], more directly related to our work, is to use graph wavelets [14, 15]. These wavelets can be viewed as convolving the input features with multiscale, oscillatory waveforms, in contrast to the simple, rectangular, one-hop waveforms used in message passing.

Here, we introduce DYMAG, which uses dynamics on the graph to generate waveforms, which we will convolve with the node features. We will use these waveforms as a form of multiscale message aggregation, which we show can effectively extract graph geometric and topological information and outperform baseline methods on graph-level tasks that rely on such graph properties.

We evaluate DYMAG on a broad spectrum of graph learning benchmarks spanning synthetic, citation, molecular, and materials science datasets. To assess its ability to recover generative and topological structure, we first test on synthetic graphs, including Erdős-Rényi and stochastic block models, where the task involves inferring graph parameters and persistent features. We then evaluate on citation networks, including homophilic datasets - Cora [16], Citeseer [17], and PubMed [18] - and heterophilic datasets - Texas, Wisconsin, and Cornell [19]. We further demonstrate DYMAG's scalability on the largest dataset in the Open Graph Benchmark, `ogbn-papers100M` [20], demonstrating

1st Conference on Topology, Algebra, and Geometry in Data Science (TAG-DS 2025).

that it can recover topological properties of massive graphs. For molecular property prediction, we consider both protein graphs PROTEINS, ENZYMES, and MUTAG [21] and small-molecule graphs (DrugBank [22], Drug Therapeutics Program AIDS Antiviral Screen Data [23]). Finally, we test on the Materials Project dataset [24] to predict materials properties such as band gaps. Across these varied domains, DYMAG consistently outperforms standard GNNs and approaches the performance of pretrained, domain-specific models. Our main contributions are as follows:

1. We introduce a DYMAG, a novel GNN which uses dynamics-waveform-based message aggregation and is capable of capturing complex signal patterns on a graph.
2. We show theoretically that our waveforms capture both the low-pass and band-pass portion of the input features as well as geometric and topological information including the graph spectrum, connected components, connectivity, cycles, shortest-path distance, and curvature.
3. We show that our method better predicts geometric and topological network properties—such as curvature and extended persistence images—compared to standard message passing networks.
4. We demonstrate that DYMAG outperforms various message passing networks as well as a large pretrained domain-specific model on molecular predictions.

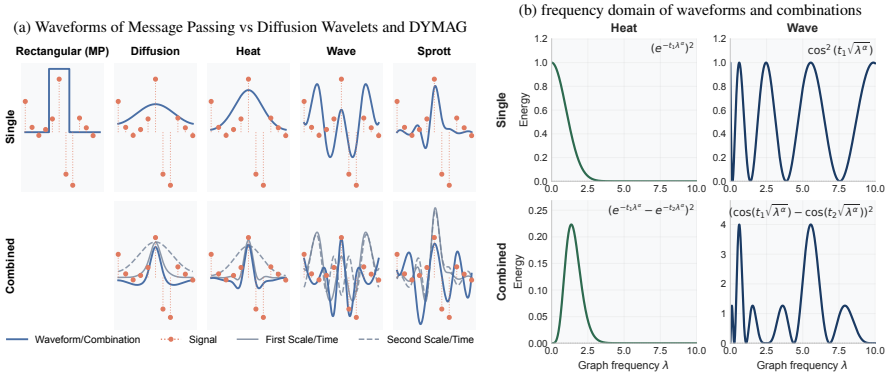

Figure 1: **Visualization of Waveforms** (a) Waveforms visualized on a path graph with a signal (feature), where DYMAG provides more diverse waveforms than standard message passing; (b) waveforms and combinations provide low-pass and bandpass filters in the frequency domain.

2 Related work and Background

Previous work has either analyzed dynamics on graphs [25–31] or aimed to use dynamics as a framework for understanding GNNs. In the latter case Chamberlain et al. [32, 33], Eliasof et al. [34] and Thorpe et al. [35] viewed message passing as a time-discretized diffusion PDE and used this insight to design novel GNNs. Unlike those methods, we view PDE solutions as waveforms and use convolution against these waveforms to define our aggregation rule. Additionally, existing work primarily focuses on parabolic equations while we also consider hyperbolic and chaotic dynamics. We provide a further discussion of related work in Appendix A.

2.1 Graph Signal Processing

In Graph Signal Processing, a node feature vector $\mathbf{x} \in \mathbb{R}^n$ is viewed as a signal on the vertices of a weighted, undirected graph $G = (V, E, w)$, $|V| = n$ [36, 37]. Let $L = U\Lambda U^\top$ be its Laplacian with eigendecomposition $L\boldsymbol{\nu}_k = \lambda_k \boldsymbol{\nu}_k$, $0 = \lambda_1 \leq \cdots \leq \lambda_n$. The *graph Fourier transform* is defined by projecting the signals onto these eigenvectors $\widehat{\mathbf{x}} = U^\top \mathbf{x}$. The projection onto the first several eigenvectors (small λ_k) captures the smooth portion of the signal, and the projection onto the later eigenvectors (large λ_k) captures the oscillatory ones. Classical message-passing GNNs often act as low-pass filters [38, 39], effectively only keeping the smooth portion of the signal; DYMAG instead aggregates with waveforms spanning low, mid, and high bands. (Details in Appendix B.1.)

2.2 Heat and Wave Dynamics on a Graph

For $\alpha > 0$, we define the α-fractional graph Laplacian by $L^\alpha := U\Lambda^\alpha U^\mathsf{T}$. For each $i \in V$, we let δ_i denote the Dirac signal at i given by $\delta_i(k) = 1$ if $i = k$, $\delta_i(k) = 0$ otherwise. Given the fractional

Laplacian, we may then define

$$-L^{\alpha}u_H^{(i)}(v,t) = \partial_t u_H^{(i)}(v,t), \quad u_H^{(i)}(v,0) = \delta_i(v), \quad \text{(Heat)} \tag{1}$$

$$-L^{\alpha}u_W^{(i)}(v,t) = \partial_t^2 u_W^{(i)}(v,t), \quad u_W^{(i)}(v,0) = \delta_i(v), \quad \partial_t u_W^{(i)}(v,0) = c\delta_i(v), \quad \text{(Wave)} \tag{2}$$

as the heat and wave equations with a initial value δ_i (and initial veclocity $c\delta_i$ for wave). On a connected graph G, they admit closed-form solutions:

$$u_H^{(i)}(v,t) = \sum_{k=1}^{n} e^{-t\lambda_k^{\alpha}} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v), \quad \text{and} \tag{3}$$

$$u_W^{(i)}(v,t) = \sum_{k=1}^{n} \cos(\sqrt{\lambda_k^{\alpha}}t)\langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v) + t\langle \boldsymbol{\nu}_1, c\delta_i \rangle \boldsymbol{\nu}_1(v) + \sum_{k=2}^{n} \frac{1}{\sqrt{\lambda_k^{\alpha}}} \sin(\sqrt{\lambda_k^{\alpha}}t)\langle \boldsymbol{\nu}_k, c\delta_i \rangle \boldsymbol{\nu}_k(v). \tag{4}$$

These expressions extend to disconnected graphs and, by Remark 1 of [40], are invariant to the choice of Laplacian eigenbasis. (See Appendix D.1 for details).

2.3 Chaotic Dynamics on a Graph

Chaotic dynamics, describing systems that have aperiodic behavior and sensitivity to initial conditions [41], can be modeled by the Sprott dynamics [42]:

$$\frac{d}{dt}u_S^{(i)}(v_k,t) = -b \cdot u_S^{(i)}(v_k,t) + \tanh\Big(\sum_{v_j \in \mathcal{N}(v_k)} c_{k,j}u^{(i)}(v_j,t)\Big), \quad u_S(\cdot,0) = \delta_i, \tag{5}$$

Solutions remain bounded for $b > 0$. For $b = 0.25$, fully connected graphs with generic couplings or sparse graphs exhibit positive Lyapunov exponents (chaos). [43, 42]. (Full details in Appendix B.3.)

3 Methods

DYMAG is a graph neural network consisting of two main parts.

1. **Waveform Bank Creation**: A diverse bank of multi-scale waveforms is constructed by solving the PDEs considered in Section 2. These waveforms define a set of basis functions that encode diverse patterns across spatial and temporal scales. (See Section 3.1.)
2. **Multi-scale Aggregations**: At each layer $1 \leq \ell \leq L$, node representations $X^{(\ell-1)}$ are convolved with the waveform bank. The result is then passed through an MLP to produce an updated representation $X^{(\ell)}$. This step replaces standard message passing mechanisms by aggregation via sophisticated, multiscale waveforms. (See Section 3.2.)

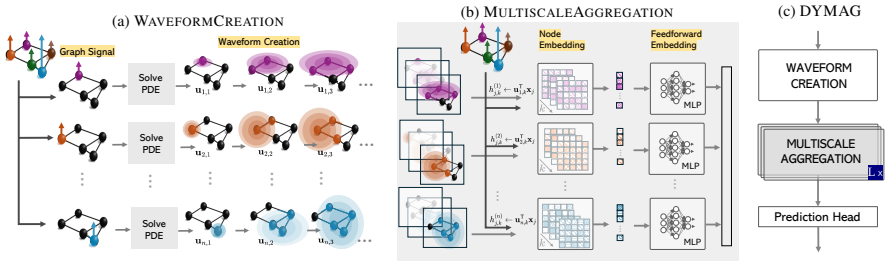

Figure 2: **Visual illustration of DYMAG** (a) Waveform bank creation solving PDEs. (b) Multiscale Aggregation by taking inner product with the waveforms. (c) DYMAG consists of stacked layers and a prediction head.

3.1 Waveform Bank Creation Using PDEs

Let $u^{(i)}(v,t)$ denote the solution to the chosen PDE dynamics (wave, heat, or Sprott equations) with initial condition $u^{(i)}(\cdot,0) = \delta_i(\cdot)$, where $\delta_i(j) = 1$ if $j = i$ and 0 otherwise (a Dirac signal centered at node i). When applicable (for second-order dynamics), we also set $\partial_t u(\cdot,0) = c\,\delta_i(\cdot)$, with a fixed hyperparameter $c \geq 0$. We choose K time points $\mathcal{T} = (t_1, \ldots, t_K)$ by fixing a maximal time T and then setting $t_k = kT/K$. We then define $\mathcal{U} = \{\mathbf{u}_{i,k}\}_{i \in V, 1 \leq k \leq K}$, where $\mathbf{u}_{i,k}$ is the vector

Algorithm 1: WAVEFORMCREATION

Input: Graph $G = (V, E, w)$; sample times: $t_1, \ldots, t_K$
Output: Waveforms $\mathcal{U}$

for $v_i \in V,\ k = 1, \ldots, K$ **do**
  $\delta_i(\cdot) \leftarrow \text{DiracSignals}(G, i)$;
  $u^{(i)}(\cdot, t_k) \leftarrow \text{SolvePDE}(\text{InitCond} = \delta_i)$;
  $\mathbf{u}_{i,k} \leftarrow u^{(i)}(\cdot, t_k)$;
return $\mathcal{U} = \{\mathbf{u}_{i,k}\}_{i \in V,\, 1 \le k \le K}$

Algorithm 2: MULTISCALEAGGR

Input: Graph $G = (V, E, w)$; node features $X^{(\ell)}$; waveforms $\mathcal{U}$
Output: Updated features $X^{(\ell+1)}$

for $v_i \in V, 1 \le j \le m, 1 \le k \le K$ **do**
  $h^{(i)}_{j,k} \leftarrow \mathbf{u}^{\top}_{i,k} \mathbf{x}_j$;
$\mathbf{y}_i \leftarrow \text{MLP}(\text{vec}(h^{(i)}_{j,k}))$;
return $X^{(\ell+1)} = (\mathbf{y}_1^{\top}, \ldots, \mathbf{y}_n^{\top})^{\top}$

Algorithm 3: DYMAG

Input: Graph $G = (V, E, w)$; node features $X = \{\mathbf{x}_j\}$; sample times $\mathcal{T}$; layers L
Output: Output Y

$\mathcal{U} \leftarrow \text{WAVEFORMCREATION}(G, \mathcal{T})$;
$X^{(0)} \leftarrow X$;
for $\ell = 1, \ldots, L$ **do**
  $X^{(\ell)} \leftarrow \text{MULTISCALEAGGR}(X^{(\ell-1)}, \mathcal{U})$;
$Y \leftarrow \text{READOUT}(X^{(L)})$;
return Y

$\mathbf{u}_{i,k} := u^{(i)}(\cdot, t_k) \in \mathbb{R}^n$. We refer to $\mathcal{U}$ as the *PDE waveform bank* (see Algorithm 1 and Figure 2a). Each waveform $\mathbf{u}_{i,k}$ is centered at node i and corresponds to a snapshot of the PDE dynamics at time scale t_k. The bank $\mathcal{U}$ collects such waveforms across all nodes and multiple time scales, similar to wavelets but more flexible thanks to the diverse dynamics. Figure 1 shows basic waveforms and more complex patterns created via combinations (from the MLP discussed below).

We note that $\mathcal{U}$ can be computed offline prior to training for increased computational efficiency. Additionally, note that the waveforms can be computed efficiently via either Chebyshev approximation or a Runge-Kutta scheme. We further discuss and report results on complexity and scalability in Appendix C.

3.2 Multi-scale Aggregation

In each layer, ℓ, we assume that we are given an $n \times m_\ell$ feature matrix $X^{(\ell)}$ (where $X^{(0)}$ consists of the initial node features). We let $\mathbf{x}_j \in \mathbb{R}^n$ denote the r-th column of $X^{(\ell)}$, which we interpret as a signal defined on V. For each waveform $\mathbf{u}_{i,k}$ in the waveform bank $\mathcal{U}$ (Section 3.1), we perform an inner product with the node features, thought of as a convolution:

$$h^{(i)}_{j,k} = \langle \mathbf{u}_{i,k}, \mathbf{x}_j \rangle = \mathbf{u}^{\mathsf{T}}_{i,k} \mathbf{x}_j. \tag{6}$$

We then combine these convolved features by applying an MLP to the states $h^{(i)}_{j,k}$ associated with each node v_i, i.e., $\mathbf{y}_i = \text{MLP}\left(\text{vec}\left(h^{(i)}_{j,k}\right)\right)$. We then reorganize the $\mathbf{y}_i$ into a transformed feature matrix $X^{(\ell+1)} = (\mathbf{y}_1^{\mathsf{T}}, \ldots, \mathbf{y}_n^{\mathsf{T}})^{\mathsf{T}}$ (so that $\mathbf{y}_i^{\mathsf{T}}$ is the i-th row of $X^{(\ell+1)}$). See Algorithm 2 and Figure 2b.

The inner product, Eqn. 6, can also be interpreted as the feature $\mathbf{x}_j$ being updated via a message from a source node v_i at scale k. Indeed, message passing neural networks can be interpreted as performing such an inner product with a limited bandwidth rectangular waveform, as shown in Figure 1 and then applying the MLP. We remark that since we use waveforms based on PDE solutions of various time snapshots, we obtain *multi-scale* embeddings. As the time t_k increases, the waveform effectively dilates and spreads to a larger neighborhood of vertices. Furthermore, via the MLP, DYMAG is able to learn novel combinations of the waveforms, either from different source nodes or at different time scales. This includes the diffusion wavelets [15] which can be obtained by subtracting solutions to the heat equation at different time scales [40].

Downstream Readout After L rounds of Multi-scale aggregation, the resulting node representations $X^{(L)} = \{\mathbf{x}_i^{(L)}\}_{i \in V}$ are used for prediction. For *node-level tasks*, a shared MLP is applied independently to each node feature vector. For *graph-level tasks*, node features are first aggregated using a permutation-invariant pooling operation (e.g., global mean or sum), followed by a task-specific MLP to produce the graph-level output. See Figure 2c and Algorithm 3.

3.3 Theoretical Properties Related to Dynamics-based Waveforms on the Graph

Below, we formulate properties of our waveforms and the information they are able to extract from the graph. These results serve as motivation for our method, which utilizes these dynamics as a novel aggregation paradigm for graph neural networks. Complete proofs are provided in Appendix D.

3.3.1 Frequency Domain Characteristics of Waveform Based Message Aggregation

Standard message passing can be viewed as convolving the node features with a single, simple, rectangular waveform. From the perspective of graph signal processing, this corresponds to a *low-pass* filtering which only preserves the low-frequency (smooth) portion of the node features (See Appendix F.) In contrast, DYMAG employees a richer, more sophisticated bank of waveforms, which we will show allows DYMAG to extract a variety of different types of information.

We first consider a function defined by $u^{(i)}_{\text{BP}}(v,t) = u^{(i)}_H(v,t_1) - u^{(i)}_H(v,t_2)$ for two fixed times, $0 < t_1 < t_2$. In the graph-Fourier domain, its response at frequency (eigenvalue) λ_k is $e^{-t_1\lambda_k^\alpha} - e^{-t_2\lambda_k^\alpha}$. This function (i) is 0 at $\lambda_k = 0$, (ii) tends to 0 as $\lambda_k \to \infty$, and (iii) reaches a single maximum at $\lambda^\star = \left(\frac{1}{t_2-t_1}\log(\frac{t_2}{t_1})\right)^{1/\alpha}$. Thus, $u^{(i)}_{BP}$ suppresses both very low and very high frequencies, but keeps information in a moderate frequency band (which depends on t_1 and t_2). Therefore, we call $u^{(i)}_{BP}$ a band-pass function. Notably, DYMAG has the ability to learn this function via the use of the MLP which is applied after Eqn. 6. We next consider the solution to the wave equation given by Eqn. 4, for simplicity focusing on the case where $c = 0$. The frequency response at each λ_k is given by $\cos(\sqrt{\lambda_k^\alpha}t)$. Since this function peaks and falls in multiple different "bands" we think of it as a *multi-band-pass* function. This leads us to the following proposition.

Proposition 3.1 (Band-pass information)**.** *DYMAG is able to extract band-pass, or even multi-band-pass information from the node features.*

Proof sketch. In heat-equation case, DYMAG can learn the band-pass function $u^{(i)}_{BP}$ via suitable weights in the MLP. In the wave equation case, DYMAG is able to capture multi-band-pass information as a consequence of the sinusoidal frequency response of the wave solution, u_W. □

In addition to the above propositions, we note that DYMAG is also able to learn low-pass information, similar to standard message passing networks. This is a direct consequence of the fact that $u^{(i)}_H$ has a decreasing frequency response $e^{-t\lambda_k^\alpha}$. It can also learn high-pass information via function $u^{(i)}_{\text{High}} = 1 - u^{(i)}_H$. We next discuss how the frequency-domain characteristics of DYMAG help alleviate the following limitations of standard GNNs:

Over-smoothing: Message passing networks utilize rectangular pulse waveforms, which act as low-pass filters, i.e., smoothing operators. With each layer, the features get smoother and smoother, eventually become nearly constant, which limits their usefulness. By contrast, DYMAG is able to learn band-pass, high-pass, and multi-band-pass information in addition to standard low-pass information. This allows it to avoid the oversmoothing problem. (We also demonstrate this empirically in Appendix G.)

Under-reaching: Message passing networks only aggregate within local, one-hop neighborhoods. Thus, their receptive field is equal to the number of layers, which must be kept small to avoid severe oversmoothing. This limits their ability to capture global structure or long-range interactions. DYMAG, on the other hand, performs aggregation via waveforms which are not confined to one-hop neighborhoods and is able to capture global structure.

Heterophily: The local averaging operation in message passing networks, are particularly problematic on heterophilic graphs where many nodes have different labels than their neighbors. DYMAG's diverse waveform banks are able to capture band-pass, multi-band-pass, and high-frequency information (in addition to low-pass). This makes them well-suited to heterophilic graphs. Additionally, we note that our experiment shows that the Sprott dynamics perform particularly well on node classification on heterophilic graphs (see Figure 3), perhaps because of their ability to detect subtle changes in different portions of the network structure.

3.3.2 General Properties of Solutions

The following result shows that DYMAG is able to identify the connected components of G.

Proposition 3.2 (Identification of Connected Components)**.** *Let $u^{(i)}(v,t)$ denote the solution to the heat equation, wave equation, or Sprott chaotic dynamics. Suppose that G is not connected. Then, for any v which is not in the same connected component as v_i, and all $t \geq 0$, we have $u^{(i)}(v,t) = 0$.*

Proof sketch. We verify that $\tilde{u}^{(i)}(v,t) := u^{(i)}(v,t) \cdot \mathbb{1}_{v \in \mathcal{C}}$, where $\mathcal{C} \subseteq V$ is the component containing v_i satisfies the same PDE as $u^{(i)}(v,t)$. The result follows by uniqueness of solutions. □

Due to Proposition 3.2, we will assume that G is connected in the following sections. However, we note that many of the results still apply to disconnected graphs with suitable modifications.

3.3.3 Heat Dynamics

The continuous and global nature of DYMAG allows it to instantaneously have a receptive field over the entire graph. Intuitively, this corresponds to information spreading instantaneously over the graph (although for small values of t the energy $u_H^{(i)}(v,\cdot)$ will be mostly concentrated near v_i). This is in contrast to message passing networks where the receptive field about each node is equal to the number of layers (which is usually kept small in order to avoid over-smoothing).

Proposition 3.3. *Let G be connected and let L be the random-walk Laplacian L_{rw} (with $\alpha = 1$). Let $u_H^{(i)}(v,t)$ be the solution to the heat equation, Eqn. 3. Then $u_H^{(i)}(v,t) > 0$ for all $v \in V$ and $t > 0$.*

Proof Sketch. This is a consequence of a relationship between $u_H^{(i)}$ and continuous-time random walks established in Lemma D.3. □

Our next two results analyze the energy decay of $u_H^{(i)}$. They suggest that graphs with a larger λ_2 will have a faster rate of energy decay. The second eigenvalue of a graph can be related to the isoperimetric ratio of a graph through Cheeger's inequality, thereby revealing information on graph structure and how "bottlenecked" a particular graph is [44]. Additionally, they show that the properties of heat energy decay can distinguish between graph structures. We note that although the assumptions for Proposition 3.5 represents a rather specific set of conditions, we expect that when two graphs have edges generated according to a similar rule or distribution, the more densely connected graph will have more rapidly decaying heat energy.

Proposition 3.4 (Heat energy)**.** *Let G be connected, and let $u_H^{(i)}(v,t)$ be as in the solution to the heat equation with initial condition δ_i as in Eqn.* (3)*. Then, $e^{-2t\lambda_n^\alpha} \leq \|u_H^{(i)}(\cdot,t)\|_2^2 \leq |\boldsymbol{\nu}_1(i)|^2 + e^{-2t\lambda_2^\alpha}$.*

Proof sketch. It follows from $\|u_H^{(i)}(\cdot,t)\|_2^2 = \sum_{k=1}^n e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 = \sum_{k=1}^n e^{-2t\lambda_k^\alpha} |\boldsymbol{\nu}_k(i)|^2$. □

Proposition 3.5 (Heat energy between graphs)**.** *Let G and G' be graphs on n vertices with fractional Laplacians L_G^α and $L_{G'}^\alpha$ and let δ_i and $\delta_{i'}$ be initial conditions for Eqn.* (1) *on G and G'. Assume: (i) $L_{G'}^\alpha \succcurlyeq L_G^\alpha$, i.e., $\mathbf{v}^\mathsf{T} L_{G'}^\alpha \mathbf{v} \geq \mathbf{v}^\mathsf{T} L_G^\alpha \mathbf{v}$ for all $\mathbf{v} \in \mathbb{R}^n$, (ii) We have $|\boldsymbol{\nu}'_k(i)|^2 \leq (1 + \eta_k(t))|\boldsymbol{\nu}_k(i)|^2$ for all $1 \leq k \leq n$, where we also assume $\eta_k(t) := \exp(2t((\lambda'_k)^\alpha - \lambda_k^\alpha)) - 1 \geq 0$.*

Then, with u_H and u'_H defined as in Eqn. (3)*, we have $\|(u_H^{(i)})'(\cdot,t)\|_2^2 \leq \|u_H^{(i)}(\cdot,t)\|_2^2$.*

Proof sketch. The result is a consequence of Parseval's identity. □

Finally, we restate some known results that provide additional foundation linking the behavior of the heat equation solutions to graph topology. Lemma 1 of Crane et al. [45] shows that the heat equation encodes shortest path distances $d(v_i, v_j)$ between nodes on the graph:

Proposition 3.6 (Relation to distances, (Lemma 1 of Crane et al. [45]))**.** *Let $u_H^{(i)}$ denote the solution to Eqn. 1 with initial condition δ_i (and $\alpha = 1$). Then, $d(v_i, v_j) = \lim_{t \to 0} \frac{\log u_H^{(i)}(v_j,t)}{\log t}$.*

We next consider the Ollivier-Ricci curvature. This is a discrete notion of curvature, meant to parallel the traditional notion of Ricci curvature in Riemannian geometry. It is defined by $\kappa(v_i, v_j) = 1 - W_1(\mu_{v_i}, \mu_{v_j})/d(v_i, v_j)$, where μ_v is a probability measure centered around v (see [46] for details), W_1 the 1-Wasserstein distance, and $d(v_i, v_j)$ is the distance (shortest path length) from v_i to v_j. The following result from Münch and Wojciechowski [46] relates κ to the heat equation.

Proposition 3.7 (Relation to Ollivier-Ricci curvature, (Theorem 5.8 of [46]))**.** *Let $L = D - A$ be the unnormalized Laplacian, then $\kappa(v_i, v_j) = \lim_{t \to 0^+} \frac{1}{t}\left(1 - \frac{W_1(u_H^{(i)}(\cdot,t), u_H^{(j)}(\cdot,t))}{d(v_i,v_j)}\right)$.*

3.3.4 Wave Dynamics

The periodicity of the solution to the wave equation endows it with the ability to capture long range interactions. Central to this argument is the wave energy, analyzed in the following proposition which focuses on the case where the initial velocity c is equal to zero:

Proposition 3.8 (Wave energy bounds)**.** *Let $u_W^{(i)}(v,t)$ be the solution to the fractional wave equation Eqn. 4 with initial conditions $u_W^{(i)}(\cdot,0) = \delta_i$ and $\partial_t u_W^{(i)}(\cdot,0) = 0$. Then, for any time $t \geq 0$, the energy of the waveform satisfies $|\boldsymbol{\nu}_1(i)|^2 \leq \|u_W^{(i)}(\cdot,t)\|_2^2 \leq 1$.*

Proof sketch. By Parseval's identity and the explicit solution in Eqn. (4), we expand: $\|u_W^{(i)}(\cdot,t)\|_2^2 = \sum_{k=1}^n \cos^2\left(\sqrt{\lambda_k^\alpha}t\right)|\langle \boldsymbol{\nu}_k, \delta_i\rangle|^2 = \sum_{k=1}^n \cos^2\left(\sqrt{\lambda_k^\alpha}t\right)|\boldsymbol{\nu}_k(i)|^2$. Since $\cos^2(\cdot) \in [0,1]$ and $\sum_k |\langle \boldsymbol{\nu}_k, \delta_i\rangle|^2 = \|\delta_i\|_2^2 = 1$, the result follows. □

This shows that, unlike heat kernels (which decay over time), the wave energy oscillates, retaining signal over time, and thus can reflect non-local interactions such as those created by cycles in the graph. These oscillations allow the waveforms to "echo" through the graph and revisit distant parts of the structure - a behavior well-suited for recovery of topological features.

The eigenspectrum of a graph, which reveals a wide range of its invariants and properties, is fully encoded in the solutions to the wave equation. This leads to the following result:

Proposition 3.9 (Cycle Length)**.** *The size of a cycle graph C_n can be determined from the solution to the fractional wave equation at a single node v.*

Proof sketch. The result follows from Lemma D.4 (Appendix) and the fact that the length of a cycle graph is contained in its eigenspectrum. □

More generally, when the graph is not a cycle but contains cyclic subgraphs as a prominent topological feature, this proposition provides some intuition for why the wave-equation is well suited to pick up that a node belongs to a cycle and recover dimension 1 homology.

4 Empirical Results

Table 1: Performance of DYMAG on four datasets: PROTEINS, DrugBank, Materials Project (MP), and the DTS AIDS Antiviral Screen. We report R^2 for the first three and balanced accuracy for the Antiviral Screen. Results are mean ± std over 10-fold CV.

Model	PROTEINS Dihedral Angles	DrugBank TPSA	DrugBank # Aromatic Rings	MP Band Gap	Antiviral Screen Active/Inactive
DYMAG (Heat)	**0.89 ± 0.01**	**0.97 ± 0.01**	**0.97 ± 0.02**	**0.61 ± 0.03**	0.54 ± 0.02
DYMAG (Wave)	**0.81 ± 0.03**	**0.90 ± 0.01**	**0.88 ± 0.01**	**0.55 ± 0.02**	**0.61 ± 0.01**
DYMAG (Sprott)	0.76 ± 0.01	0.77 ± 0.01	0.82 ± 0.03	**0.54 ± 0.03**	**0.63 ± 0.02**
MPNN	0.78 ± 0.01	0.71 ± 0.01	0.81 ± 0.01	0.37 ± 0.05	0.51 ± 0.02
GAT	0.72 ± 0.02	0.78 ± 0.02	0.83 ± 0.02	0.40 ± 0.03	**0.59 ± 0.03**
GIN	0.69 ± 0.03	0.69 ± 0.01	0.77 ± 0.01	0.38 ± 0.03	**0.60 ± 0.02**
GWT	**0.81 ± 0.02**	0.83 ± 0.02	0.85 ± 0.01	0.42 ± 0.02	0.58 ± 0.02
GraphGPS	0.64 ± 0.03	0.63 ± 0.02	0.67 ± 0.04	0.31 ± 0.02	0.54 ± 0.03
GRAND	0.76 ± 0.03	0.53 ± 0.04	0.64 ± 0.03	0.27 ± 0.03	0.49 ± 0.03
GRAND++	0.62 ± 0.03	0.56 ± 0.02	0.61 ± 0.02	0.31 ± 0.02	0.51 ± 0.02
CayleyNet	0.75 ± 0.02	0.72 ± 0.01	0.79 ± 0.02	0.41 ± 0.04	0.55 ± 0.03
AdaGNN	0.73 ± 0.01	0.75 ± 0.02	0.80 ± 0.01	0.39 ± 0.03	0.57 ± 0.02
DRew	0.68 ± 0.02	0.70 ± 0.03	0.75 ± 0.02	0.34 ± 0.03	0.53 ± 0.02
GraphCON	0.71 ± 0.02	0.74 ± 0.02	0.78 ± 0.01	0.35 ± 0.03	0.55 ± 0.01
GraFF	0.67 ± 0.02	0.66 ± 0.02	0.74 ± 0.03	0.33 ± 0.04	0.50 ± 0.03
SWAN	0.74 ± 0.01	0.76 ± 0.01	0.80 ± 0.02	0.43 ± 0.02	0.58 ± 0.01
Pretrained Model	**0.83 ± 0.03** (ProtBERT)	**0.98 ± 0.01** (MolBERT)	**0.97 ± 0.01** (MolBERT)	**0.62 ± 0.05** (GeoCGNN)	**0.59 ± 0.02** (ProtBERT)

We evaluate DYMAG across diverse tasks to assess its ability to recover geometric/topological structure and generalize to downstream biological, chemical, and materials applications. In this section, we set $\alpha = 1$ so that the fractional Laplacian coincides with the ordinary graph Laplacian. We conduct some experiments with other exponents α in Appendix I.5. Baselines include message-passing GNNs (MPNN [47], GCN [48], GraphSAGE [49], GAT [50], GIN [51]), diffusion-based methods (GRAND [33], GRAND++[35]), Bandpass methods (CayleyNet [52], AdaGNN [53]),

DE-GNN methods (GraphCON [54], GraFF [55], SWAN [56]), rewiring methods (DRew[57]), and GraphGPS [58], a state-of-the art graph transformer. We also compare against a GNN built with fixed-scale wavelets (GWT [15]) and a neural approximation algorithm of the extended persistence diagram (EPD) [59]. For molecular and materials prediction, we include pretrained models such as ProtBERT [60], MolBERT [61], and GeoCGNN [62].

In Table 1, we highlight the **best** and **second best** results, treating ties within one standard deviation as equivalent. Results are 10-fold cross-validation means unless noted otherwise. Implementation details are in Appendix E, and full results with standard deviations are in Appendix I.

4.1 Geometric and Topological Properties

To evaluate the expressivity of DYMAG, we assessed its ability to learn fundamental geometric and topological graph properties. Our evaluation centered on two main approaches: (1) direct prediction of features such as Ollivier–Ricci curvature and topological persistence images, and (2) performance on downstream proxy tasks, including random graph parameter recovery and node classification on both homophilic and heterophilic benchmarks.

The results confirm that DYMAG learns rich and adaptive graph representations. The heat and wave dynamics variants proved to be top performers on most tasks, including large-scale node classification on the ogbn-papers100M dataset. Notably, the chaotic Sprott dynamics variant demonstrated superior performance on heterophilic graphs, highlighting its sensitivity to local graph structure. A detailed presentation of these experiments, including full results and additional analyses, is provided in Appendix I.1.

4.2 Proteins, Molecules, and Materials

We evaluated DYMAG on graphs representing proteins, drug-like molecules, and materials shown in Tables 1 and 7 (appendix). We note that DYMAG's strong performance on these data sets indicates its potential to positively impact society by furthering the design of materials, drugs, or other healthcare treatments. More specifically, the tasks include predicting geometric and chemical properties such as dihedral angles, total polar surface area (TPSA), the number of aromatic rings, band gaps in materials, and anti-HIV activity. Datasets include PROTEINS [21], DrugBank [22], the Materials Project [24], and the AIDS Antiviral Screen [23]. Across all datasets, DYMAG consistently outperforms standard GNNs, GRAND, GRAND++, GraphGPS, GWT, CayleyNet, AdaGNN, GraphCON, GraFF. SWAN, and DRew by a wide margin. It matches the performance of powerful, task-specific pretrained models within 1 standard deviation overall, and significantly surpasses them on the PROTEINS and AIDS datasets.

Overall, the heat and wave versions of DYMAG perform strongly across all molecular and material prediction tasks.The Sprott (chaotic) variant shows more variable performance, which may reflect its heightened sensitivity to local graph structure. This behavior appears beneficial in settings with recurring structural motifs, such as the Antiviral Screen dataset, and may be less advantageous in tasks where such sensitivity is less critical. Additional results, including accuracy and training time, are provided in Appendix I.

5 Conclusion

We introduce DYMAG as a method for improving aggregation in GNNs. We use dynamics to generate a diverse bank of waveforms that span multiple frequency bands. Messages are aggregated by taking each node feature, interpreted as a graph signal, and projecting it onto this bank via inner products, producing a set of features that encode multi-scale information. The expressiveness of this representation, arising from the rich frequency structure of the waveforms, helps mitigate common GNN limitations such as oversmoothing, underreaching, and heterophily.

One limitation of our method is that the Sprott model is extremely sensitive to the graph structure. This is useful for some tasks which require detecting minute changes in structure. However, this may be undesirable in other settings where one may want similar representations of nearly isomorphic. Another limitation is that our method is currently only applicable to supervised tasks. Extending DYMAG to unsupervised tasks, such as clustering, denoising, or signal reconstruction could be an interesting avenue of future work. Overall, DYMAG establishes a theoretically grounded, empirically strong message aggregation paradigm; future work will broaden its application to diverse graph tasks and refine the accompanying speed-up techniques.

6 Acknowledgments

This project has been partially funded by the Yale - Boehringer Ingelheim Biomedical Data Science Fellowship and the Kavli Institute for Neuroscience Postdoctoral Fellowship [D.B.]; the National Science Foundation under grant number OIA-2242769 [M.P.]; Canada CIFAR AI Chair, NSERC Discovery grant 03267, and FRQNT grant 343567 [G.W.]; NSF CAREER award IIS-2047856, NSF CISE grant 2403317, and NIH grant R01GM130847 [S.K.]; NIH grant R01GM135929 [G.W.,S.K.]; and NSF DMS grant 2327211 [G.W.,M.P.,S.K.] The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

References

[1] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[2] Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems*, 35:2268–2281, 2022.

[3] Federico Errica, Henrik Christiansen, Viktor Zaverkin, Takashi Maruyama, Mathias Niepert, and Francesco Alesiani. Adaptive message passing: A general framework to mitigate oversmoothing, oversquashing, and underreaching. *arXiv preprint arXiv:2312.16560*, 2023.

[4] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[5] Feng Gao, Guy Wolf, and Matthew Hirn. Geometric scattering for graph data analysis. In *International Conference on Machine Learning*, pages 2122–2131. PMLR, 2019.

[6] Michael Perlmutter, Alexander Tong, Feng Gao, Guy Wolf, and Matthew Hirn. Understanding graph neural networks with generalized geometric scattering transforms. *SIAM Journal on Mathematics of Data Science*, 5(4):873–898, 2023.

[7] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Diffusion scattering transforms on graphs. January 2019. 7th International Conference on Learning Representations, ICLR 2019 ; Conference date: 06-05-2019 Through 09-05-2019.

[8] Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Stability of graph scattering transforms. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Yuanhong Jiang, Dongmian Zou, Xiaoqun Zhang, and Yu Guang Wang. Limiting over-smoothing and over-squashing of graph message passing by deep scattering transforms. *arXiv preprint arXiv:2407.06988*, 2024.

[10] Frederik Wenkel, Yimeng Min, Matthew Hirn, Michael Perlmutter, and Guy Wolf. Overcoming oversmoothness in graph convolutional networks via hybrid scattering networks. *arXiv preprint arXiv:2201.08932*, 2022.

[11] Bernhard G Bodmann and Íris Emilsdóttir. A scattering transform for graphs based on heat semigroups, with an application for the detection of anomalies in positive time series with underlying periodicities. *Sampling Theory, Signal Processing, and Data Analysis*, 22(2):17, 2024.

[12] Charles Xu, Laney Goldman, Valentina Guo, Benjamin Hollander-Bodie, Maedee Trank-Greene, Ian Adelstein, Edward De Brouwer, Rex Ying, Smita Krishnaswamy, and Michael Perlmutter. BLIS-Net: Classifying and analyzing signals on graphs. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4537–4545. PMLR, 02–04 May 2024. URL `https://proceedings.mlr.press/v238/xu24c.html`.

[13] Alexander Tong, Frederik Wenkel, Dhananjay Bhaskar, Kincaid Macdonald, Jackson Grady, Michael Perlmutter, Smita Krishnaswamy, and Guy Wolf. Learnable filters for geometric scattering modules. *arXiv preprint arXiv:2208.07458*, 2022.

[14] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[15] Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2006.04.004. URL https://www.sciencedirect.com/science/article/pii/S106352030600056X. Special Issue: Diffusion Maps and Wavelets.

[16] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2), July 2000. ISSN 1573-7659. doi: 10.1023/A:1009953814988. URL https://doi.org/10.1023/A:1009953814988.

[17] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer: an automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, May 1998. ISBN 978-0-89791-965-4.

[18] 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.

[19] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[21] Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, July 2003. ISSN 0022-2836. doi: 10.1016/s0022-2836(03)00628-4.

[22] D Wishart et al. DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, January 2018. ISSN 1362-4962. doi: 10.1093/nar/gkx1037.

[23] National Institutes of Health. Aids antiviral screen data. https://wiki.nci.nih.gov/spaces/NCIDTPdata/pages/158204006/AIDS+Antiviral+Screen+Data, 2021. Accessed: 2025-05-10.

[24] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, and Kristin A. Persson. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 07 2013. ISSN 2166-532X. doi: 10.1063/1.4812323. URL https://doi.org/10.1063/1.4812323.

[25] Jaap C Reijneveld, Sophie C Ponten, Henk W Berendse, and Cornelis J Stam. The application of graph theoretical analysis to complex networks in the brain. *Clinical neurophysiology*, 118 (11):2317–2331, 2007.

[26] S. Boccaletti, G. Bianconi, R. Criado, C.I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2014.07.001. URL https://www.sciencedirect.com/science/article/pii/S0370157314002105. The structure and dynamics of multilayer networks.

[27] Joana A Simoes. An agent-based/network approach to spatial epidemics. *Agent-based models of geographical systems*, 2011.

[28] Petter Holme and Jari Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.

[29] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020.

[30] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[31] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv:2010.03409*, 2020.

[32] Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. *Advances in Neural Information Processing Systems*, 34:1594–1609, 2021.

[33] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, 2021.

[34] Moshe Eliasof, Eldad Haber, and Eran Treister. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. In *Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:236912752`.

[35] Matthew Thorpe, Tan Minh Nguyen, Heidi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representation (ICLR)*, 2022.

[36] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[37] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106 (5):808–828, 2018.

[38] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3950–3957, 2021.

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

[40] Joyce Chew, Matthew Hirn, Smita Krishnaswamy, Deanna Needell, Michael Perlmutter, Holly Steach, Siddharth Viswanath, and Hau-Tieng Wu. Geometric scattering on measure spaces. *Applied and Computational Harmonic Analysis*, 70:101635, 2024. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2024.101635. URL `https://www.sciencedirect.com/science/article/pii/S1063520324000125`.

[41] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[42] JC Sprott. Chaotic dynamics on large networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(2), 2008.

[43] Ludwig Arnold and Volker Wihstutz. Lyapunov exponents: a survey. In *Lyapunov Exponents: Proceedings of a Workshop held in Bremen, November 12–15, 1984*, pages 1–26. Springer, 2006.

[44] Daniel A Spielman. Spectral and algebraic graph theory, 2019. *URL http://cs-www. cs. yale. edu/homes/spielman/sagt. Version dated December*, 19, 2019.

[45] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. The heat method for distance computation. *Commun. ACM*, 60(11):90–99, October 2017. ISSN 0001-0782. doi: 10.1145/3131280. URL `http://doi.acm.org/10.1145/3131280`.

[46] Florentin Münch and Radosław K. Wojciechowski. Ollivier Ricci curvature for general graph Laplacians: Heat equation, Laplacian comparison, non-explosion and diameter bounds. *Advances in Mathematics*, 356:106759, 2019. ISSN 0001-8708. doi: https://doi.org/10.1016/j.aim.2019.106759. URL https://www.sciencedirect.com/science/article/pii/S000187081930369X.

[47] 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.

[48] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

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

[50] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[51] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[52] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[53] Yushun Dong, Kaize Ding, Brian Jalaian, Shuiwang Ji, and Jundong Li. Adagnn: Graph neural networks with adaptive frequency response filter. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 392–401, 2021.

[54] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[55] Francesco Di Giovanni, James Rowbottom, Benjamin P Chamberlain, Thomas Markovich, and Michael M Bronstein. Understanding convolution on graphs via energies. *arXiv preprint arXiv:2206.10991*, 2022.

[56] Alessio Gravina, Moshe Eliasof, Claudio Gallicchio, Davide Bacciu, and Carola-Bibiane Schönlieb. On oversquashing in graph neural networks through the lens of dynamical systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16906–16914, 2025.

[57] Hugo Attali, Davide Buscaldi, and Nathalie Pernelle. Rewiring techniques to mitigate oversquashing and oversmoothing in gnns: A survey. *arXiv preprint arXiv:2411.17429*, 2024.

[58] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[59] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, Yusu Wang, and Chao Chen. Neural Approximation of Graph Topological Features, November 2022. URL http://arxiv.org/abs/2201.12032. arXiv:2201.12032 [cs].

[60] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8): 2102–2110, 2022.

[61] Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*, 2020.

[62] Jiucheng Cheng, Chunkai Zhang, and Lifeng Dong. A geometric-information-enhanced crystal graph network for predicting properties of materials. *Communications Materials*, 2(1):92, 2021.

[63] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1320–1329, 2018.

[64] Bobak Kiani, Lukas Fesser, and Melanie Weber. Unitary convolutions for learning on graphs and groups. *Advances in Neural Information Processing Systems*, 37:136922–136961, 2024.

[65] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 2016.

[66] J Clint Sprott. Some simple chaotic flows. *Physical review E*, 50(2):R647, 1994.

[67] JC Sprott. Some simple chaotic jerk functions. *American Journal of Physics*, 65(6):537–543, 1997.

[68] Earl A Coddington. *An introduction to ordinary differential equations*. Courier Corporation, 2012.

[69] Hasan A. Fallahgoul, Sergio M. Focardi, and Frank J. Fabozzi. 7 - continuous-time random walk and fractional calculus. In Hasan A. Fallahgoul, Sergio M. Focardi, and Frank J. Fabozzi, editors, *Fractional Calculus and Fractional Processes with Applications to Financial Economics*, pages 81–90. Academic Press, 2017. ISBN 978-0-12-804248-9. doi: https://doi.org/10.1016/B978-0-12-804248-9.50007-3. URL `https://www.sciencedirect.com/science/article/pii/B9780128042489500073`.

[70] H. Adams, S. Chepushtanova, T. Emerson, E. Hanson, M. Kirby, F. Motta, R. Neville, C. Peterson, P. Shipman, and L. Ziegelmeier. Persistence Images: A Stable Vector Representation of Persistent Homology, 2016. URL `http://arxiv.org/abs/1507.06217`. arXiv:1507.06217.

[71] Yann Ollivier. Ricci curvature of Markov chains on metric spaces, July 2007. URL `http://arxiv.org/abs/math/0701886`. arXiv:math/0701886.

[72] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending Persistence Using Poincaré and Lefschetz Duality. *Foundations of Computational Mathematics*, 9(1):79–103, February 2009. ISSN 1615-3383. doi: 10.1007/s10208-008-9027-z. URL `https://doi.org/10.1007/s10208-008-9027-z`.

[73] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

[74] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 347–356, 2004.

[75] C. Morris, N. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv:2007.08663*, 2020.

[76] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

[77] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[78] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009.

[79] Khang Nguyen, Hieu Nong, Vinh Nguyen, Nhat Ho, Stanley Osher, and Tan Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature 2022. *URL https://arxiv.org/abs/2211.15779*, 2022.

[80] Jayson Sia, Edmond Jonckheere, and Paul Bogdan. Ollivier-ricci curvature-based method to community detection in complex networks. *Scientific reports*, 9(1):9800, 2019.

[81] Romeil Sandhu, Tryphon Georgiou, Ed Reznik, Liangjia Zhu, Ivan Kolesov, Yasin Senbabaoglu, and Allen Tannenbaum. Graph curvature for differentiating cancer networks. *Scientific reports*, 5(1):12323, 2015.

[82] Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[83] Corinna Coupette, Sebastian Dalleiger, and Bastian Rieck. Ollivier-ricci curvature for hypergraphs: A unified framework. *arXiv preprint arXiv:2210.12048*, 2022.

[84] Wonwoo Kang and Heehyun Park. Accelerated evaluation of ollivier-ricci curvature lower bounds: Bridging theory and computation. *arXiv preprint arXiv:2405.13302*, 2024.

[85] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 02 2020. ISSN 2641-3337. doi: 10.1162/qss_a_00021. URL `https://doi.org/10.1162/qss_a_00021`.

[86] Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, ICML'12, 2012. ISBN 978-1-4503-1285-1.

[87] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning †Electronic supplementary information (ESI) available. See DOI: 10.1039/c7sc02664a. *Chemical Science*, 9(2):513–530, October 2017. ISSN 2041-6520. doi: 10.1039/c7sc02664a. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5868307/`.

A Related work

There is a long history of studying dynamics on graphs. For example, Reijneveld et al. [25], Boccaletti et al. [26], Simoes [27], Holme and Saramäki [28] analyze complex interactions such as brain processes, social networks, and spatial epidemics on graphs. The study of graph dynamics has recently crossed into the field of deep learning with Belbute-Peres et al. [29], Sanchez-Gonzalez et al. [30], Pfaff et al. [31] using neural networks to simulate complex phenomena on irregularly structured domains.

Differing from the above mentioned works, there have also been several recent papers which aim to use dynamics as a framework for understanding GNNs. Chamberlain et al. [32] and Chamberlain et al. [33] take the perspective that message-passing neural networks can be interpreted as the discretizations of diffusion-type (parabolic) partial differential equations on graph domains where each layer corresponds to a discrete time step. They then use this insight to design GRAND, a novel GNN, based on encoding the input node features, running a diffusion process for T seconds and finally applying a decoder network. Thorpe et al. [35] builds on this work by extending it to diffusion equations with "sources" placed at the labeled nodes, leading to a new network GRAND++. They then provide an analysis of both GRAND and GRAND++ and show that they are related to different graph random walks. GRAND is related to a standard graph random walk, whereas GRAND++ is related to a dual random walk started at the labeled data, which can avoid the oversmoothing problem. We also note Donnat et al. [63], which used the graph heat equation to extract structural information around each node (although not in a neural network context) and Eliasof et al. [34], which used insights from hyperbolic and parabolic PDEs on manifolds to design a GNN that does not suffer from oversmoothing as well as Kiani et al. [64] which uses convolution using unitary groups to improve GNNs ability to learn long-range dependencies.

Our network method differs from these previous works in several important ways. Most importantly, whereas Chamberlain et al. [33] and Chamberlain et al. [32] primarily focused on PDEs as a framework for understanding the behavior of message passing operations, here we propose to use the dynamics associated to the heat equation as a new form of feature aggregation, replacing traditional message passing operations. Additionally, we consider both the heat equation (the prototypical parabolic PDE) and the wave equation (the prototypical hyperbolic PDE) as well as chaotic dynamics, whereas previous work [32, 33, 35] has primarily focused on parabolic equations. Notably, similar to GRAND++, the wave-equation and the Sprott versions of DYMAG do not suffer from oversmoothing. However, the long-term behavior of these equations differs from the diffusion-with-a-source equation used in GRAND++ in that they only depend on the geometry of the network and not on the locations of the labeled data. (Additionally, since we do not require labeled data as source locations, our method can be easily adapted to unsupervised problems by removing the MLPs.)

B Detailed Background

This section is a more detailed version of the background on graph signal processing and dynamics provided in Section 2.

B.1 Graph Signal Processing

In *graph signal processing*, node features are interpreted as signals (functions) defined on the nodes of a graph [36, 37]. Each signal can then be decomposed into different frequencies defined in terms of the eigendecomposition of the graph Laplacian and an associated Fourier transform.

Formally, we let $\mathbf{x} : V \to \mathbb{R}$, denote a function (signal) defined on the vertices of a weighted, undirected graph $G = (V, E, w)$ with vertices $V = \{v_1, \dots, v_n\}$. For convenience, we will identify $\mathbf{x}$ with the vector whose k-th entry is $\mathbf{x}(v_k)$. Thus, we will write either $\mathbf{x}(v_k)$ or $\mathbf{x}(k)$, depending on context. We may also write $\mathbf{x}(v)$ if we do not wish to emphasize the ordering of the vertices. We let L denote a graph Laplacian with eigenvectors $\boldsymbol{\nu}_1, \dots, \boldsymbol{\nu}_n$ and eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, $L\boldsymbol{\nu}_k = \lambda_k \boldsymbol{\nu}_k$. Unless otherwise specified, we will assume that L is either the unnormalized Laplacian $L_U = D - A$ or the symmetric normalized Laplacian $L_{\text{sym}} = D^{-1/2} L_{\text{U}} D^{-1/2}$, where D and A are the weighted degree and adjacency matrices. In these cases, we may write $L = U \Lambda U^{\mathsf{T}}$, where U

is a matrix with columns $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n$ and Λ is a diagonal matrix, $\Lambda_{k,k} = \lambda_k$.[1] The graph Fourier transform can be defined as $\widehat{\mathbf{x}} = U^{\mathsf{T}}\mathbf{x}$ so that $\widehat{\mathbf{x}}(k) = \langle \boldsymbol{\nu}_k, \mathbf{x} \rangle$. Since the $\boldsymbol{\nu}_k$ form an orthonormal basis, we obtain the Fourier inversion formula $\mathbf{x} = \sum_{k=1}^{n} \widehat{\mathbf{x}}(k)\boldsymbol{\nu}_k$. The eigenvectors $\boldsymbol{\nu}_k$ are referred to as *Fourier modes* and the eigenvalues λ_k are interpreted as *frequencies*. Therefore, the Fourier inversion formula can be thought of as decomposing a signal $\mathbf{x}$ into the superposition of Fourier modes at different frequencies.

Standard message passing neural networks are known to essentially perform low-pass filtering [38, 39]; i.e., they preserve the portion of the signal corresponding to the first one or two eigenvectors, while suppressing the rest of the signal. As we discussed in Section 3.3.1, the waveforms utilized in DYMAG may span a broader range of frequency behavior and can highlight different aspects of the frequency spectrum by acting as either as low-pass, high-pass, or band-pass filters.

B.2 Heat and Wave Dynamics on a Graph

For $\alpha > 0$, we define the α-fractional graph Laplacian by $L^\alpha := U\Lambda^\alpha U^{\mathsf{T}}$. We note that L^α has the same eigenvectors as L and the eigenvalues of L^α are given by λ_k^α, i.e., $L^\alpha \boldsymbol{\nu}_k = \lambda_k^\alpha \boldsymbol{\nu}_k$. Additionally, we see that when $\alpha = 1/m$ for some $m \in \mathbb{N}$, we have $(L^{1/m})^m = L$. For each i, we let δ_i denote the Dirac signal at i given by $\delta_i(k) = 1$ if $i = k$, and $\delta_i(k) = 0$ otherwise. We say that a function $u_H^{(i)}(v,t)$ solves the *α-fractional heat equation* with a initial value δ_i if

$$-L^\alpha u_H^{(i)}(v,t) = \partial_t u_H^{(i)}(v,t), \quad u_H^{(i)}(v,0) = \delta_i(v). \tag{7}$$

We say that $u_W^{(i)}$ solves the *α-fractional wave equation* with initial Dirac data δ_i and an initial velocity $c\delta_i$ (where c is a constant) if

$$-L^\alpha u_W^{(i)}(v,t) = \partial_t^2 u_W^{(i)}(v,t), \quad u_W^{(i)}(v,0) = \delta_i(v), \quad \partial_t u_W^{(i)}(v,0) = c\delta_i(v). \tag{8}$$

If G is connected, solutions to the heat and wave equations are given explicitly by

$$u_H^{(i)}(v,t) = \sum_{k=1}^{n} e^{-t\lambda_k^\alpha} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v), \quad \text{and} \tag{9}$$

$$u_W^{(i)}(v,t) = \sum_{k=1}^{n} \cos(\sqrt{\lambda_k^\alpha}t)\langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v) + t\langle \boldsymbol{\nu}_1, c\delta_i \rangle \boldsymbol{\nu}_1 + \sum_{k=2}^{n} \frac{1}{\sqrt{\lambda_k^\alpha}} \sin(\sqrt{\lambda_k^\alpha}t)\langle \boldsymbol{\nu}_k, c\delta_i \rangle \boldsymbol{\nu}_k(v). \tag{10}$$

We note that Eqns. 3 and 4 can also be adapted to disconnected graphs with simple modifications. Furthermore, following Remark 1 in Chew et al. [40], we note that the solutions $u_H^{(i)}(v,t)$ and $u_W^{(i)}(v,t)$ defined in Eqn. 3 and 4 do not depend on the choice of orthonormal basis for the graph Laplacian, see Section D.1 for details.

B.3 Chaotic Dynamics on a Graph

We next consider dynamics exhibiting chaos, a behavior that may be informally summarized as "aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions" [41]. As a prototypical example of chaos on graphs, we consider the general, complex, and nonlinear graph dynamics on graphs described in Sprott [42]:

$$\frac{d}{dt}u_S^{(i)}(v_k,t) = -b \cdot u_S^{(i)}(v_k,t) + \tanh\left(\sum_{v_j \in \mathcal{N}(v_k)} c_{k,j} u^{(i)}(v_j,t)\right), \quad u_S(\cdot,0) = \delta_i, \tag{11}$$

where b is a damping coefficient, and the $c_{k,j}$ represent interactions. We refer to Eq. 5 as the Sprott equation and denote its solutions by $u_S(v,t)$. When $b > 0$, solutions $u_S(v,t)$ remain bounded. In the case of fully connected graphs, with $b = 0.25$, chaotic dynamics (corresponding to positive Lyapunov exponents, see, e.g., Arnold and Wihstutz [43]) were observed when a sufficiently large fraction

[1] If L is the random walk Laplacian $L_{\text{rw}} = D^{-1}L_U = D^{-1/2}L_{\text{sym}}D^{1/2}$, we may instead obtain an asymmetric eigendecomposition, $L_{\text{rw}} = (D^{-1/2}U)\Lambda(D^{1/2}U)^{\mathsf{T}}$, where $L_{\text{sym}} = U\Lambda U^{\mathsf{T}}$.

of interactions were neither symmetric ($c_{j,k} = c_{k,j}$) nor anti-symmetric ($c_{j,k} = -c_{k,j}$). Sparsely connected networks also exhibited positive Lyapunov exponents with a value of $b = 0.25$ [42].

The explicit purpose of using Sprott dynamics is to generate chaotic dynamics on the graph, which we believe would be beneficial for distinguishing isomorphic or structurally similar graphs.

C Computational Complexity

Here, we discuss the computational complexity of DYMAG and show that it may be scaled to large graphs.

We utilize Chebyshev polynomials to approximate $u_H^{(i)}(v,t)$ and $u_W^{(i)}(v,t)$ as defined in Eqns. 3 and 4 motivated by the success of Chebyshev polynomials in the approximation of spectral graph wavelets [14] and GNNs [65]. This removes the need for eigendecomposition (which can have $\mathcal{O}(n^3)$ computational complexity and $\mathcal{O}(n^2)$ memory). The polynomial approximation has linear complexity for sparse graphs [65]. For example, if G is a k-nearest neighbor graph and the order of the polynomial is m, then the time complexity for solving the heat/wave equation is $O(kmn)$.

DYMAG's runtime complexity is $\mathcal{O}(r|E|FT)$, where $|E|$ is the number of edges, F is the number of features, and r is the degree of the Chebyshev polynomial. No closed form solution is available for the Sprott dynamics, so we instead use a fourth-order Runge-Kutta method, which has been successfully applied to chaotic systems [66, 67]. This approach has complexity $\mathcal{O}(g|E|FT)$, where g is the number of solver steps between sampled time steps.

For each of the underlying dynamics, DYMAG exhibits linear complexity, with respect to the number of vertices for sparse graphs (setting $|E| = n\bar{d}$, where $\bar{d}$ is the average degree). This makes it efficient and scalable to large graphs, where the focus is often on local or node-level properties rather than global topological properties (although predicting global properties is the primary focus of our work). In such cases, local properties can be examined with a smaller feature set of size $F' \ll n$, by concentrating on subgraphs around nodes of interest. Parallelization is feasible for large sparse graphs since the r-th order Chebyshev polynomials act over r-hop neighborhoods, allowing DYMAG to scale with standard MPNN techniques. Furthermore, the feature space F can be selected or adjusted to be small by utilizing random features, Diracs on a subset of nodes, or natural graph signals.

Table 2 reports the training time per epoch (in seconds) for different variants of DYMAG and a wide range of baselines across four benchmark datasets. We see that the DYMAG variants have competitive training times. Table 3 shows the time complexity of precomputing the waveforms. Together, these results show that DYMAG scales well to large datasets both in training and preprocessing.

Table 2: Training time per epoch (s) for DYMAG variants and baselines. Mean ± standard deviation computed over 5 independent training runs, each with different random seeds, on an NVIDIA A100 GPU with a batch size of 512.

Method	PROTEINS (1,113 graphs)	DrugBank (6,712 graphs)	MP (69,239 graphs)	Antiviral Screen (43,850 graphs)
DYMAG (Heat)	0.48 ± 0.09	**1.80 ± 0.21**	**9.6 ± 1.2**	**6.3 ± 0.9**
DYMAG (Wave)	0.54 ± 0.09	1.95 ± 0.24	**10.5 ± 1.2**	**6.9 ± 0.9**
DYMAG (Sprott)	0.75 ± 0.15	2.70 ± 0.30	13.5 ± 1.8	9.0 ± 1.2
MPNN	**0.36 ± 0.06**	**1.35 ± 0.15**	**8.4 ± 0.9**	**5.7 ± 0.6**
GAT	**0.45 ± 0.09**	**1.65 ± 0.18**	**10.5 ± 1.2**	**6.9 ± 0.9**
GIN	**0.39 ± 0.06**	**1.59 ± 0.15**	**9.0 ± 0.9**	**7.8 ± 0.6**
GWT	0.54 ± 0.12	2.16 ± 0.24	**12.6 ± 1.5**	8.4 ± 0.9
GraphGPS	0.75 ± 0.15	3.36 ± 0.30	18.9 ± 2.1	13.8 ± 1.5
GRAND	0.60 ± 0.12	2.61 ± 0.27	15.9 ± 1.8	10.5 ± 1.2
GRAND++	0.66 ± 0.15	2.70 ± 0.30	16.5 ± 2.1	11.4 ± 1.5
CayleyNet	0.90 ± 0.18	3.87 ± 0.45	22.5 ± 2.7	15.6 ± 1.8
AdaGNN	0.84 ± 0.15	3.30 ± 0.36	22.2 ± 2.4	14.4 ± 1.8
DRew	0.51 ± 0.09	1.95 ± 0.24	**12.3 ± 1.5**	**8.1 ± 0.9**
GraphCON	0.69 ± 0.15	2.85 ± 0.33	17.4 ± 2.1	11.7 ± 1.5
GraFF	1.05 ± 0.21	4.29 ± 0.54	25.5 ± 3.0	17.4 ± 2.1
SWAN	1.20 ± 0.24	4.80 ± 0.60	30.0 ± 3.6	19.5 ± 2.4

Table 3: Waveform precomputation time (s) for DYMAG.

Method	PROTEINS	DrugBank	MP	Antiviral Screen
DYMAG (Heat)	**0.83**	**3.67**	**28.3**	**22.3**
DYMAG (Wave)	**1.00**	**4.33**	**31.3**	**25.0**
DYMAG (Sprott)	10.00	21.70	166.7	123.3

D Proofs of Theoretical Results

D.1 Proof of independence of eigenbasis

Here we provide a detailed proof for the result that Equations 3 and 4 are invariant to the choice of the Laplacian eigenbasis, as mentioned in Section 2.2:

The solutions $u_H^{(i)}$ and $u_W^{(i)}$ do not depend on the choice of eigenbasis (even when the eigenvalues have multiplicity greater than one). To see this, let $\mathcal{S}_\lambda$ be the set of *distinct* eigenvalues of L. For $\lambda \in \mathcal{S}_\lambda$, let E_λ be the corresponding eigenspace, i.e., the linear space spanned by the set of $\boldsymbol{\nu}_k$ such that $\lambda_k = \lambda$. Let π_λ denote the corresponding projection operator, i.e.,

$$\pi_\lambda \mathbf{x} = \sum_{k:\lambda_k=\lambda} \langle \boldsymbol{\nu}_k, \mathbf{x} \rangle \boldsymbol{\nu}_k,$$

for all $\mathbf{x} \in \mathbb{R}^n$, and observe that π_λ is independent of choice of eigenbasis. Then, from Eqn. 3, we may then write,

$$\begin{aligned} u_H^{(i)}(v,t) &= \sum_{k=1}^{n} e^{-t\lambda_k^\alpha} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v) \\ &= \sum_{\lambda \in \mathcal{S}_\lambda} e^{-t\lambda^\alpha} \sum_{k:\lambda_k=\lambda} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k(v) \\ &= \sum_{\lambda \in \mathcal{S}_\lambda} e^{-t\lambda^\alpha} \pi_\lambda \delta_i(v). \end{aligned}$$

This establishes that $u_H^{(i)}$ is independent of the choice of basis. The argument for $u_W^{(i)}$ is identical other than using Eqn. 4 in place of Eqn. 3. (For the second term in Eqn. 4 note that we are assuming that the graph is connected which implies that λ_1 has single multiplicity, i.e., $0 = \lambda_1 < \lambda_2$.)

D.2 Proofs of propositions

Below, we prove Proposition 3.1, (restated below) the band-pass properties of DYMAG through the waveforms.

Proposition 3.1. *(Band-pass information)* DYMAG is able to extract band-pass, or even multi-band-pass information information from the node features.

To make Proposition 3.1 more precise, we will separate it out into two propositions. However, first, we will introduce some notation and definitions.

We define heat-kernel as $H^t = e^{-tL^\alpha}$, i.e.,

$$H^t = U \operatorname{diag}\left(\exp(-t\lambda_1)^\alpha, \ldots, \exp(-t\lambda_n^\alpha)\right) U^\mathsf{T}.$$

We observe that we may rewrite the solution to the heat equation, with any initial condition $u_H(\cdot, 0)$ as

$$u_H(\cdot,t) = \sum_{m=1}^{n} e^{-t\lambda_m^\alpha} \langle \boldsymbol{\nu}_m, u(\cdot,0) \rangle \boldsymbol{\nu}_m = H^t u_H(\cdot,0). \tag{12}$$

In particular, we have $\mathbf{u}_{i,k} = H^{t_k}\delta_i$. Thus, we see that the features

$$h_{j,k}^{(i)} = \langle \mathbf{u}_{i,k}, \mathbf{x}_j \rangle$$

defined in Eqn. 6 can be rewritten as

$$\begin{aligned} h_{j,k}^{(i)} &= \sum_{\ell=1}^{n} H^{t_k}\delta_i(\ell)\mathbf{x}_j(\ell) \\ &= \sum_{\ell=1}^{n}\sum_{m=1}^{n} e^{-t\lambda_m^\alpha}\langle \boldsymbol{\nu}_m, \delta_i\rangle \boldsymbol{\nu}_m(\ell)\mathbf{x}_j(\ell) \\ &= \sum_{\ell=1}^{n}\sum_{m=1}^{n} e^{-t\lambda_m^\alpha}\boldsymbol{\nu}_m(i)\boldsymbol{\nu}_m(\ell)\mathbf{x}_j(\ell) \\ &= (H^{t_k}\mathbf{x}_j)(i). \end{aligned}$$

Next, for fixed times $t_1 < t_2$, we define

$$\Psi_{t_1,t_2} = H^{t_1} - H^{t_2} \tag{13}$$

as the difference of two heat kernels. In the spectral domain, we note that we may write

$$\Psi_{t_1,t_2}\mathbf{x} = \sum_{m=1}^{n}(e^{-t_1\lambda_m^\alpha} - e^{-t_2\lambda_m^\alpha})\langle \boldsymbol{\nu}_m, \mathbf{x}\rangle \boldsymbol{\nu}_m.$$

The function,

$$\psi_{t_1,t_2}(\lambda) = e^{-t_1\lambda^\alpha} - e^{-t_2\lambda^\alpha}, \tag{14}$$

which is referred to as the frequency response is zero both at $\lambda = 0$, and as $\lambda \to \infty$. Its support is concentrated in a *frequency band* centered around $\lambda^\star = \left(\frac{\log(t_2/t_1)}{t_2 - t_1}\right)^{1/\alpha}$. Therefore, Ψ_{t_1,t_2} is referred to as a band-pass filter. We summarize this in the following proposition.

Proposition D.1 (Difference of two heat waveforms is band-pass)**.** *Let $t_1 < t_2$ and define Ψ_{t_1,t_2} as in Eqn. 13. Then the function $\mathbf{x} \mapsto \Psi_{t_1,t_2}\mathbf{x}$ performs a band-pass filtering.*

Proof. This follows immediately from the frequency response illustrated in Eqn. 14 and the subsequent discussion. □

Importantly, we observe that DYMAG has the capacity to learn Ψ_{t_1,t_2} through the MLP in Algorithm 2. Therefore, Proposition D.1 shows that DYMAG, with the heat-equation has the capacity to perform band-pass filtering. This is in contrast to standard message passing networks which are known to effectively perform low-pass filtering (i.e., averaging type operations).

We next turn our attention to the wave equation, focusing on the case with zero initial velocity (i.e., $c = 0$) for the sake of simplicity. Similar to the heat-case (Eqn. 12), we may define a wave kernel by $W^t = \cos\left(t\sqrt{L^\alpha}\right) = U\,\mathrm{diag}\left(\cos(t\sqrt{\lambda_1^\alpha}), \ldots, \cos(t\sqrt{\lambda_n^\alpha})\right)U^\mathsf{T}$ and observe that the solution to the wave equation, with initial condition $u_W(\cdot, 0)$ is given by $W^t u_W(\cdot, 0)$. In the spectral domain, we may write

$$\Phi_t\mathbf{x} = \sum_{m=1}^{n}\cos(t\sqrt{\lambda_m^\alpha})\langle \boldsymbol{\nu}_m, \mathbf{x}\rangle \boldsymbol{\nu}_m. \tag{15}$$

The associated frequency response is given by

$$\psi(\lambda) = \cos(t\sqrt{\lambda^\alpha}). \tag{16}$$

This function attains its maximum absolute value at $\lambda = \lambda_m := (m\pi/t)^{2/\alpha},\ m = 0, 1, 2, \ldots$ (band-passes), and vanishes at $\lambda = \lambda_{m+\frac{1}{2}} := ((2m+1)\pi/2t)^{2/\alpha}$ (stop-bands). Hence $u_{\mathrm{W}}^{(i)}$ alternates between preserving and suppressing successive frequency intervals and thus acts as a *multi-band filter*. We summarize this in the following proposition.

Proposition D.2 (Wave equation is multi-band)**.** *Fix a time $t > 0$ and, for simplicity, set the initial velocity to zero ($c = 0$). Consider the wave kernel $W^t = cos\left(t\sqrt{L^\alpha}\right)$ and define Φ_t as in Eqn.* (15)*. Then the function $\mathbf{x} \mapsto \Phi_t\mathbf{x}$ performs a multi-band-pass filtering.*

Proof of Proposition D.2. This follows from Eqn. 16 and the subsequent analysis of the maxima and zeros of $\psi(\lambda)$.

□

We now turn our attention to Proposition 3.2.

Proposition 3.2. *(Identification of Connected Components)* Let $u^{(i)}(v,t)$ denote the solution to the heat equation, wave equation, or Sprott chaotic dynamics. Suppose that G is not connected. Then, for any v which is not in the same connected component as v_i, and all $t \geq 0$, we have $u^{(i)}(v,t) = 0$.

Proof of Proposition 3.2. First observe that it suffices to prove the result for $0 \leq t \leq T$ where T is arbitrary (since we may then let $T \to \infty$).

Let $u^{(i)}$ be a solution to the differential equation and consider the function $\tilde{u}^{(i)}$ defined by

$$\tilde{u}^{(i)}(v,t) = \begin{cases} u^{(i)}(v,t) & \text{if } v \in \mathcal{C} \\ 0 & \text{otherwise} \end{cases},$$

where $\mathcal{C}$ is the connected component containing v_i.

We observe that $\tilde{u}^{(i)}$ is also a solution since the right hand side of each differential equation is localized in the sense that no energy passes between components and the initial condition has support contained in $\mathcal{C}$. However, since the right hand side of all three differential equations is Lipschitz on $[0,T]$, Theorem 5, Section 6 of Coddington [68] implies that there is at most one solution to the differential equation and thus $\tilde{u}^{(i)} = u^{(i)}$ on $V \times [0,T]$. Therefore, since T was arbitrary, we have $u^{(i)}(v,t) = 0$ for all $v \notin \mathcal{C}$.

Importantly, we note that the fractional Laplacian L^α is not a graph shift operator, i.e., we may have $L^\alpha_{i,j} \neq 0$ even if $i \neq j$ and $\{v_i, v_j\} \notin E$. However, it is still component-localized in the sense that $L^\alpha_{i,j} \neq 0$ implies that v_i and v_j are in the same connected component. To see this, note that if G is disconnected, then we can choose an ordering of the vertices so that L is block-diagonal (with the diagonal blocks corresponding to connected components). This implies that we may choose an eigenbasis such that each eigenvector $\boldsymbol{\nu}_i$ has its support (non-zero entries) contained in a single connected component. Therefore, writing L^α in its outer-product expansion,

$$L^\alpha = \sum_{i=1}^{n} \lambda^\alpha \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\mathsf{T}$$

implies that L^α is component-localized. □

Proposition 3.3. *Let G be connected and let L be the random-walk Laplacian L_{rw} (with $\alpha = 1$). Let $u_H^{(i)}(v,t)$ be the solution to the heat equation, Eqn. 3. Then $u_H^{(i)}(v,t) > 0$ for all $v \in V$ and $t > 0$.*

In order to prove Proposition 3.3, we need the following lemma which relates the heat equation to a continuous time random walker (see e.g., Fallahgoul et al. [69]) $\{X_t^{\text{continuous}}\}_{t \geq 0}$ defined by $X_t^{\text{continuous}} = X_{N(t)}^{\text{discrete}}$, where $\{X_k^{\text{discrete}}\}_{k=0}^{\infty}$ is a standard discrete-time random walker (i.e., a walker who moves to a neighboring vertex at each discrete time step) and $\{N(t)\}_{t \geq 0}$ is an ordinary Poisson process.

Lemma D.3. *Let $u_H^{(i)}(v,t)$ be the solution to the heat equation with L chosen to be the random-walk Laplacian, $L = L_{RW} = I - P$ where $P = D^{-1}A$ and initial condition δ_i. Then $u_H^{(i)}(\cdot,t)$ is the probability distribution of a continuous-time random walker started at v_i at time t.*

Proof of Lemma D.3. We first note that $L = L_{rw}$ can be written in terms of the symmetric normalized Laplacian $L_s = I_n - D^{-1/2}AD^{-1/2}$ as $L_{rw} = D^{-1/2}L_sD^{1/2}$. Therefore, L_{rw} is diagonalizable and may be written as $L_{rw} = S\Lambda S^{-1}$ where $L_s = U\Lambda U^{-1}$ is a diagonalization of L_s and $S = D^{-1/2}U$. This allows us to write

$$P = I - L_{rw} = S(I - \Lambda)S^{-1},$$

which implies that for $k \geq 0$, we have

$$P^k = S(I - \Lambda)^k S^{-1}.$$

We next note that Eqn. 3 may be written as $u_H^{(i)}(\cdot, t) = H^t \delta_i$, where H^t is the heat kernel defined by

$$H^t := e^{-tL} = S e^{-t\Lambda} S^{-1}. \tag{17}$$

Therefore, it suffices to show that H^t is the t-second transition matrix of the continuous-time random walker $X_t^{\text{continuous}} = X_{N(t)}^{\text{discrete}}$.

By the definition of a Poisson process, $N(t)$ is a Poisson random variable with parameter t. Thus, for $k \geq 0$, $t \geq 0$, we have

$$\mathbb{P}(N_t = k) = A_t(k) = t^k e^{-t}/k!.$$

We next observe that for all $\mu \in \mathbb{R}$ we have

$$e^{-t(1-\mu)} = e^{-t} \sum_{k \geq 0} \frac{(t\mu)^k}{k!} = \sum_{k \geq 0} A_t(k) \mu^k. \tag{18}$$

Substituting $\lambda = 1 - \mu$ in Eqn. 18 links the eigenvalues of U_H^t and P by

$$\begin{aligned} U_H^t = S e^{-t\Lambda} S^{-1} &= S \sum_{k \geq 0} A_t(k) (I_n - \Lambda)^k S^{-1} \\ &= \sum_{k \geq 0} A_t(k) P^k. \end{aligned}$$

This implies that H^t is the t-second transition matrix of the continuous-time random walker and thus completes the proof. □

Now we prove Proposition 3.3.

Proof of Proposition 3.3. Let $v \in V$ be arbitrary. As shown in Lemma D.3, $u_H^{(i)}(\cdot, t)$ is the probability distribution of a continuous-time random walk with initial distribution δ_i at time t. Therefore, if d is the length of the shortest path from v to v_i, then

$$\begin{aligned} u_H^{(i)}(v, t) &= \mathbb{P}(X_t^{\text{continuous}} = v | X_0^{\text{discrete}} = v_i) \\ &\geq \mathbb{P}(N(t) = d) \mathbb{P}(X_d^{\text{discrete}} = v | X_0^{\text{discrete}} = v_i) > 0. \end{aligned}$$ □

Proposition 3.4. *(Heat energy) Let G be connected, and let $u_H^{(i)}(v, t)$ be as in Eqn. 3 and let $u_H^{(i)}(t) = u_H^{(i)}(\cdot, t)$. Then,*

$$e^{-2t\lambda_n^\alpha} \leq \|u_H^{(i)}(\cdot, t)\|_2^2 \leq |\boldsymbol{\nu}_1(i)|^2 + e^{-2t\lambda_2^\alpha} \tag{19}$$

for all $t > 0$. Furthermore, as t converges to infinity, we have

$$\lim_{t \to \infty} u_H^{(i)}(t) = \langle \boldsymbol{\nu}_1, \delta_i \rangle \boldsymbol{\nu}_1 = \boldsymbol{\nu}_1(i) \boldsymbol{\nu}_1. \tag{20}$$

Proof of Proposition 3.4. From Eqn. 3, and the fact that $\{\boldsymbol{\nu}_k\}_{k=1}^n$ is an ONB, we see

$$\begin{aligned} \|u_H^{(i)}(t)\|_2^2 &= \left\langle \sum_{k=1}^n e^{-t\lambda_k^\alpha} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k, \sum_{k=1}^n e^{-t\lambda_k^\alpha} \langle \boldsymbol{\nu}_k, \delta_i \rangle \boldsymbol{\nu}_k \right\rangle \\ &= \sum_{k=1}^n e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2. \end{aligned} \tag{21}$$

Thus, since $\lambda_1 = 0$, upper bound in Eqn. 19 is obtained by:

$$\begin{aligned}
\|u_H^{(i)}(t)\|_2^2 &= \sum_{k=1}^{n} e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \\
&= |\langle \boldsymbol{\nu}_1, \delta_i \rangle|^2 + \sum_{k=2}^{n} e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \\
&\leq |\langle \boldsymbol{\nu}_1, \mathbf{x} \rangle|^2 + e^{-2t\lambda_2^\alpha} \sum_{k=2}^{n} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \\
&\leq |\langle \boldsymbol{\nu}_1, \delta_i \rangle|^2 + e^{-2t\lambda_2^\alpha}.
\end{aligned}$$

The lower bound in Eqn. 19 may be obtained by noting

$$\sum_{k=1}^{n} e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \geq e^{-2t\lambda_n^\alpha} \sum_{k=1}^{n} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 = e^{-2t\lambda_n^\alpha}.$$

Eqn. 20 immediately follows Eqn. 21 and the fact that $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$. □

Proposition 3.5. *(Heat energy between graphs)* Let G and G' be graphs on n vertices with fractional Laplacians L_G^α and $L_{G'}^\alpha$ and let δ_i and $\delta_{i'}$ be initial conditions for Eqn. 1 on G and G'. Assume: (i) $L_{G'}^\alpha \succcurlyeq L_G^\alpha$, i.e., $\mathbf{v}^\mathsf{T} L_{G'}^\alpha \mathbf{v} \geq \mathbf{v}^\mathsf{T} L_G^\alpha \mathbf{v}$ for all $\mathbf{v} \in \mathbb{R}^n$, (ii) We have $|\boldsymbol{\nu}'_k(i)|^2 \leq (1 + \eta_k(t))|\boldsymbol{\nu}_k(i)|^2$ for all $1 \leq k \leq n$, where we also assume $\eta_k(t) := \exp(2t((\lambda'_k)^\alpha - \lambda_k^\alpha)) - 1 \geq 0$. Then, with u_H and u'_H defined as in Eqn. 3, we have $\|(u_H^{(i)})'(\cdot, t)\|_2^2 \leq \|u_H^{(i)}(\cdot, t)\|_2^2$.

Proof of Proposition 3.5. By Eqn. 21, we have

$$\begin{aligned}
&\|u_H^{(i)}(\cdot, t)\|_2^2 - \|(u_H^{(i)})'(\cdot, t)\|_2^2 \\
&\quad = \sum_{k=1}^{n} e^{-2t\lambda_k^\alpha} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 - e^{-2t\lambda_k'^\alpha} |\langle \boldsymbol{\nu}'_k, \delta_{i'} \rangle|^2 \\
&\quad \geq \sum_{k=1}^{n} \left[e^{-2t\lambda_k^\alpha} - e^{-2t\lambda_k'^\alpha} (1 + \eta_i(t)) \right] |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \\
&\quad = 0.
\end{aligned}$$

□

Proposition 3.8. *(Wave energy bounds)* Let $u_W^{(i)}(v, t)$ be the solution to the fractional wave equation Eqn. 4 with initial conditions $u_W^{(i)}(\cdot, 0) = \delta_i$ and $\partial_t u_W^{(i)}(\cdot, 0) = 0$. Then, for any time $t \geq 0$, the energy of the waveform satisfies $|\boldsymbol{\nu}_1(i)|^2 \leq \|u_W^{(i)}(\cdot, t)\|_2^2 \leq 1$.

Proof of Proposition 3.8. The proof is similar to the proof of Eqn. 19. By the same reasoning as in Eqn. 21, we have

$$\|u_W^{(i)}(t)\|_2^2 = \sum_{k=1}^{n} \cos^2\left(\sqrt{\lambda_k^\alpha} t\right) |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2. \tag{22}$$

Therefore,

$$\|u_W^{(i)}(t)\|_2^2 = \sum_{k=1}^{n} \cos^2(\sqrt{\lambda_k^\alpha} t) |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 \leq \sum_{k=1}^{n} |\langle \boldsymbol{\nu}_k, \delta_i \rangle|^2 = \|\delta_i\|_2^2 = 1.$$

The lower bound follows by noting that since $\lambda_1 = 0$ we have:

$$\begin{aligned}\|u_W^{(i)}(t)\|_2^2 &= \sum_{k=1}^{n} \cos^2(\sqrt{\lambda_k^\alpha}t)|\langle \boldsymbol{\nu}_k, \delta_i\rangle|^2 \\ &\geq \cos^2(\sqrt{\lambda_1^\alpha}t)|\langle \boldsymbol{\nu}_1, \delta_i\rangle|^2 \\ &= |\langle \boldsymbol{\nu}_1, \delta_i\rangle|^2. \\ &= |\boldsymbol{\nu}_1(i)|^2.\end{aligned}$$

□

To prove Proposition 3.9, we first prove the following Proposition:

Lemma D.4 (Recovery of eigenspectrum from waveforms)**.** *Let G be connected, and let $u_W^{(i)}(v,t)$ be the solution to the fractional wave equation (Eqn.* (4)*), with initial conditions $u_W^{(i)}(\cdot,0) = \delta_i$ and $\partial_t u_W^{(i)}(\cdot,0) = 0$. Then, for any fixed node v, the sequence of values $u_W^{(i)}(v,t_1),\ldots,u_W^{(i)}(v,t_m)$ obtained from time samples can be used to approximate the full Laplacian eigenspectrum $\{\lambda_k^\alpha\}_{k=1}^n$ up to arbitrary precision, provided sufficient time resolution.*

Proof. Fix v. Since $\mathbf{y} = 0$, we may rewrite Eqn. 4 as

$$f(t) \coloneqq u_W^{(i)}(v,t) = \sum_{k=1}^{n} \cos(\sqrt{\lambda_k^\alpha}t)c_k(v)$$

where $c_k(v) = \langle \boldsymbol{\nu}_k, \delta_i\rangle \boldsymbol{\nu}_k(v)$ is a constant with respect to time and depends only on the node position.

Now, let $\epsilon > 0$ be a degree of precision and choose K such that $\frac{1}{K} < \epsilon$. Approximate

$$f(t) \approx \tilde{f}(t) \coloneqq \sum_{k=1}^{n} \cos(a_k t)c_k(v)$$

where a_k is the multiple of $1/K$ such that $|a_k - \sqrt{\lambda_k^\alpha}| < \epsilon$. The function $\tilde{f}$ has a finite Fourier expansion and therefore is uniquely characterized by finitely many samples which allows us to recover the a_k and thus approximately recover the λ_k.

□

The graph eigenspectrum encodes a wide range of graph invariants and properties. Proposition D.4 demonstrates that the solutions to the wave equation relate to graph spectral properties, and that this entire information is contained in the solutions of the wave equation *at each node*. Now we prove:

Proposition 3.9. *(Cycle Length)* The size of a cycle graph C_n can be determined from the solution to the fractional wave equation at a single node v.

Proof. Denote C_n the cycle graph with vertices numbered $0,\ldots,n-1$ and edges $(v, v+1)$ modulo n. Since C_n is 2-regular, the unnormalized and normalized Laplacian differ only by a constant multiple of 2. Therefore, without loss of generality, we will focus on the unnormalized Laplacian.

It is known that an orthogonal eigenbasis is given by $\{\phi_k\}_{k=0}^{k=\lfloor n/2\rfloor} \cup \{\psi_k\}_{k=1}^{\lceil (n-1)/2\rceil}$ defined:

$$\phi_k(v) = \cos\left(\frac{2\pi k v}{n}\right), \qquad \psi_k(v) = \sin\left(\frac{2\pi k v}{n}\right),$$

where the corresponding eigenvalues are given by

$$\lambda_k = 2 - 2\cos\left(\frac{2\pi k}{n}\right) = 4\sin^2\left(\frac{\pi k}{n}\right) \tag{23}$$

Consider the case where the wave equation has an initial condition of $\mathbf{y} = 0$ and the initial signal is given by $\mathbf{x} = \delta_a$. The solution to this equation at a fixed node v is given by:

$$u_W(t) = \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \cos\left(2t \sin\left(\frac{\pi k}{n}\right)\right) \cos(2\pi k a/n)\phi_k/\|\phi_k\|^2$$
$$+ \sum_{k=1}^{\lceil (n-1)/2 \rceil} \cos\left(2t \sin\left(\frac{\pi k}{n}\right)\right) \sin(2\pi k a/n)\psi_k/\|\psi_k\|^2, \tag{24}$$
$$+ \phi_0/\|\phi_0\|^2 \tag{25}$$

where the third term in Eqn. 24 corresponding to the smallest non-zero eigenvalue is nonzero. By Proposition D.4 each of the λ_k are uniquely determined by our measurements and

$$n = \frac{2\pi}{\cos^{-1}\left(\frac{2-\lambda_1}{2}\right)}.$$

Thus n is uniquely determined by our measurements. □

E Implementation Details

E.1 Experimental Computation Resources

All experiments were conducted on a high-performance computing server equipped with an Intel® Xeon® Gold 6240 CPU (18 cores, 36 threads, 2.60 GHz base frequency) and 730 GB of system RAM. The server is configured with 4 NVIDIA A100 GPUs, each with 40 GB of VRAM, enabling efficient parallel training of deep learning models. The system runs on Red Hat Enterprise Linux 8.8 with CUDA version 11.8 and cuDNN 8.5.0. Experiments were executed using PyTorch 2.0.0 and Python 3.10. Unless otherwise stated, all models were trained using mixed precision to optimize memory usage and throughput.

E.2 DYMAG Parameters

Here we describe the architecture and training setup of DYMAG used to generate the experimental results presented in this paper. DYMAG supports both node-level and graph-level tasks, and is evaluated on datasets comprising either a single large graph (e.g., citation networks such as PubMed) or collections of small graphs (e.g., synthetic graphs, molecular graphs, and protein structures).

We use a stacked architecture consisting of $L = 3$ DYMAG layers, each simulating K discrete time steps of heat or wave dynamics, where K is selected via grid search specific to each dataset. Between layers, we apply a 3-layer MLP with LeakyReLU activations for node-wise transformation. For all downstream tasks, we use a 5-layer MLP with LeakyReLU activations as the task-specific head.

For node-level tasks such as node classification in citation graphs, we directly use the hidden node embeddings produced by DYMAG and feed them into the 5-layer MLP to perform classification or regression. For datasets composed of multiple small graphs (e.g., synthetic Erdős-Rényi graphs, molecular graphs, etc.), we apply mean pooling across the node dimension to obtain a graph-level embedding. This pooled embedding is passed to the same 5-layer MLP for classification or regression.

E.3 Parameters for Sprott Dynamics

For Sprott dynamics (Eqn. (5)), in our experiments, we set $b = 0.25$ and the coupling coefficients as $c_{k,j} \sim \frac{1}{\sqrt{n-1}}\left(2 \cdot \text{Bernoulli}(0.5) - 1\right)$, which assigns each $c_{k,j}$ a value of $\pm\frac{1}{\sqrt{n-1}}$ with equal probability.

E.4 Hyperparameter sensitivity

The main hyperparameters in DYMAG are the number of time points T, the number of waveform samples K, and the fractional derivative exponent α. We conducted a hyperparameter search over T using a 1, 2, 5 log-scale scheme (i.e., $T \in \{2, 5, 10, 20, 50, 100\}$) on Erdős-Rényi graphs with 100 -

5000 nodes, selecting the best value based on recovery of generating parameters. For downstream tasks, we set T based on the typical graph size; a good heuristic is to scale T with graph diameter. For a fixed T, we sample the dynamics at K equidistant time points to create the waveform representation. We used $K = 100$ in all experiments. Additinally, we evaluated sensitivity to the fractional exponent α in Table 9, ranging from 0.25 to 1. DYMAG shows strong robustness across tasks, with most performance differences within one standard deviation. The only exception is TPSA prediction, where $\alpha = 1$ (i.e., standard diffusion) performs best, likely due to the smooth, global nature of the TPSA feature.

F Classic spectral GNNs: low-pass filtering, over-smoothing, and overfitting

Classic spectral domain graph neural networks like GCNs and ChebNets are, in theory, capable of learning arbitrary spectral responses. However, in practice, their spectral response functions are usually limited to low degree polynomial filters [48, 65] with the degree even limited to $K = 1$ or 2. This gap motivates the development of modern, non-low-pass GNNs, which do not require difficult spectral computations. The choice of a low-degree polynomial inherently limits a GNN's ability to model long-range relationships, leading to under-reaching. Attempting to fix this by stacking many layers often introduces over-smoothing and overfitting.

G DYMAG Is Effective Against Over-Smoothing and Under-Reaching

To test DYMAG's ability to handle heterogeneous node features without oversmoothing, we evaluated DYMAG on heterophilic graphs. These heterophilic graphs have the characteristic that adjacent nodes belong to different classes (as opposed to homophilic graphs), and thus smoothing would inhibit high performance on such datasets. In Table 12, we show that DYMAG outperforms the baselines on heterophilic datasets such as Texas, Wisconsin, and Cornell as well as homophilic datasets such as Cora and PubMed. Our claims regarding over-smoothing and under-reaching are backed by our theoretical results. Proposition 3.1 that establishes that DYMAG has the capacity to perform band-pass filtering via wave-propagation rather than low-pass filters which perform "smoothing." Further, DYMAG combats under-reaching by aggregating information across the graph (or large swaths of it depending on the T parameter) using multiscale waveforms for aggregation. We further establish this by computing the Dirichlet energy of DYMAG hidden layer features. We compute the Dirichlet energy of node embeddings $x \in \mathbb{R}^n$ using the standard normalized form:

$$\mathcal{E}(x) = \frac{x^\top L x}{x^\top x},$$

where L is the graph Laplacian and x is the node embedding vector from a hidden layer. This quantity reflects how much the embedding varies over adjacent nodes. Values near 0 indicate oversmoothing, while values closer to 1 reflect sharper transitions across edges. Table 4 shows that DYMAG generates hidden node representations with Dirichlet energy in the 0.4 - 0.5 range, indicating that it preserves meaningful variation across the graph without excessive smoothing. In contrast, models like MPNN, GIN, and GAT exhibit significantly lower Dirichlet energy, consistent with oversmoothing, where node features become too similar and local structure is lost. Only GWT (graph wavelet transform) and GraphGPS (graph transformer) achieve comparable or higher energy levels, but DYMAG matches or exceeds them in classification accuracy (see Table 8), suggesting a better trade-off between smoothness and expressivity.

Table 4: Dirichlet energy of hidden node representations for various models on the ogbn-papers100M dataset. Values reported are mean ± standard deviation over 3-fold cross validation.

Model	Dirichlet Energy
DYMAG (Heat)	0.443 ± 0.018
DYMAG (Wave)	$\mathbf{0.515 \pm 0.014}$
MPNN	0.303 ± 0.016
GAT	0.353 ± 0.029
GIN	0.289 ± 0.013
GWT	0.419 ± 0.017
GraphGPS	$\mathbf{0.500 \pm 0.015}$

H DYMAG can be made inductive

DYMAG can be made inductive with minor modifications. We have specifically designed the waveform bank such that there are KT (T is number of time points, and K is the number of waveform samples) waveforms centered at every node. With the addition of new nodes, we would add these waveforms and also modify other waveforms depending on their proximity to the new nodes. In large graphs, T would likely be much smaller than the span of the graph. The "wavelet transforms" resulting from the newly placed waveforms are placed back on the new node as a node feature which can be used to classify the node. Thus, the computation resulting from the addition of these waveforms is incremental. Therefore, our model could be made inductive.

I Additional Experiments

I.1 Geometric and Topological Properties

We first evaluated the expressivity of multiscale dynamics as a replacement for message passing by training DYMAG to recover geometric and topological features, including Ollivier–Ricci curvature and the persistence image representation [70] of the extended persistence diagram. Ollivier–Ricci curvature [71], a discrete analog of Ricci curvature from Riemannian geometry, captures local graph geometry. We discuss the motivation of using Ollivier-Ricci curvature as a node regression task in Appendix I.2. For persistent homology, diagrams are computed from ascending and descending filtrations of each node's 5-hop neighborhood, using node degree as the filter [72, 59]. The resulting persistence images encode connected components (0 homology) and loops (1st homology).

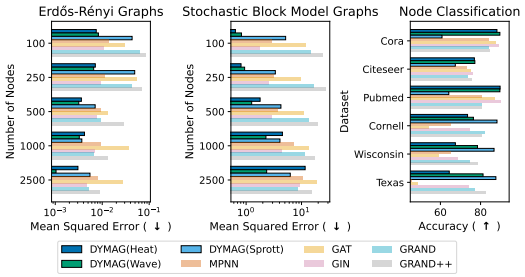

Figure 3: Mean squared error (MSE, lower is better) for predicting the generating parameters of random graphs and node classification accuracy (higher is better) for homophilic and heterophilic datasets. (See also Tables 11 and 12)

We evaluated the model on Erdős-Rényi graphs [73], $G(n, p)$, with $n \in \{100, 200\}$ and $p \in \{0.04, 0.06, 0.08\}$, stochastic block model (SBM) graphs, and several citation graphs. The results are shown in Table 5 and Table 6. DYMAG significantly improves prediction accuracy on both Ollivier–Ricci curvature and persistent homology compared to standard GNNs and GRAND. For persistent homology, DYMAG performs on par with the purpose-built model of Yan et al. [59], a neural approximation of the Union-Find EPD algorithm, despite DYMAG not being designed specifically for this task. DYMAG generalizes well across both synthetic and real-world graphs, including Cora [16], Citeseer [17], and PubMed [18]. We note that while persistent homology can be computed directly, it is computationally expensive with cost $\mathcal{O}(g^3)$, g being the number of generators [74]. More importantly, these experiments highlight that DYMAG learns rich graph representations that capture topological features when useful for prediction, and it can adapt to the task to extract relevant information for regression or classification.

As shown in Figure 3 and Tables 11 and 12, we evaluate expressivity via two tasks as proxies: recovering parameters of random graphs and node classification on homophilic (Pubmed, Citeseer, Cora) and heterophilic (Texas, Wisconsin, Cornell) [19] networks. We see that the heat and wave versions of DYMAG are the top performing models on most of the data sets whereas the Sprott version of DYMAG is the top performing model on the heterophilic data sets, possibly because these data sets need a model which is sensitive to small changes in local graph structure.

In Table 8, we evaluate DYMAG on ogbn-papers100M for node classification, complementing the curvature prediction (node regression) results above. The results show that DYMAG-wave and DYMAG-heat outperform other baselines on large graphs, further demonstrating that DYMAG scales effectively to large datasets.

We present further experimental validation of DYMAG, focusing on its ability to predict both biomolecular properties and topological descriptors such as curvature.

Table 5: Mean squared error (MSE) for predicting Ollivier–Ricci curvature (κ) and extended persistence images (EP) across citation and OGBN graph benchmarks. Results are averaged over 10 runs; lower is better. "–" indicates not applicable, timeout, or out of memory.

Model	**Cora**		**Citeseer**		**PubMed**		**ogbn-mag**	**ogbn-papers100M**
	κ	EP	κ	EP	κ	EP	EP	EP
$\text{DYMAG}_{(\text{Heat})}$	**1.73e-2**	**1.45e-4**	**2.09e-2**	**3.94e-4**	**7.30e-3**	**7.93e-3**	**5.39e-2**	**9.03e-2**
$\text{DYMAG}_{(\text{Wave})}$	**1.85e-2**	**7.41e-5**	**3.41e-2**	**1.58e-4**	**6.51e-3**	**3.29e-3**	**8.01e-3**	**3.25e-2**
$\text{DYMAG}_{(\text{Sprott})}$	1.34e-1	3.51e-3	1.46e-1	7.64e-3	2.97e-2	8.96e-2	–	–
MPNN	2.40e-1	2.72e-3	2.20e-1	6.83e-3	1.69	4.29e-1	2.93e-1	6.19e-1
GAT	7.36e-1	5.04e-2	9.04e-1	2.93e-2	1.55e-1	4.79e-1	4.47e-1	8.03e-1
GIN	1.56e-1	5.53e-3	1.72e-1	3.94e-3	8.34e-3	1.27e-1	2.49e-1	6.43e-1
GWT	4.07e-2	6.79e-4	3.58e-2	9.12e-4	9.15e-3	2.16e-2	5.83e-2	2.71e-1
GraphGPS	4.41e-1	2.71e-2	2.82e-1	3.16e-2	7.68e-1	1.94e-1	3.78e-1	4.05e-1
GRAND	6.81e-2	8.13e-4	1.24e-1	6.87e-4	2.73e-2	3.37e-2	–	–
GRAND++	1.74e-1	8.16e-3	3.09e-1	4.21e-3	8.72e-2	3.07e-1	–	–
Neural EPD Approx.	–	5.80e-5	–	1.29e-4	–	2.74e-3	6.73e-3	3.18e-2

Table 6: Mean squared error (MSE) for predicting Ollivier–Ricci curvature (κ) and extended persistence images (EP) on Erdős–Rényi and citation graphs. Results are shown as mean (standard deviation). DYMAG with heat or wave dynamics outperforms all baselines. A uniform signal was used on graphs without node features. Due to computational cost, Ollivier–Ricci curvature was computed only on a 2,000-node subgraph for PubMed and omitted for OGBN-MAG and OGBN-Papers100M.

Dataset	$\text{DYMAG}_{(\text{Heat})}$	$\text{DYMAG}_{(\text{Wave})}$	$\text{DYMAG}_{(\text{Sprott})}$	**MPNN**	**GAT**	**GIN**	**GWT**	**GraphGPS**	**GRAND**	**GRAND++**	**Neural EPD Approx.**
Ollivier–Ricci Curvature (κ)											
ER ($p = 0.04, n = 100$)	**1.86e-01** (4.01e-03)	**1.93e-01** (8.78e-03)	7.44e+00 (2.73e-01)	3.20e+01 (1.28e+00)	2.37e+01 (5.45e-01)	5.93e+00 (2.37e-01)	**3.60e-01** (2.96e-02)	1.14e+01 (1.51e-01)	1.10e+01 (3.28e-01)	1.48e+01 (1.80e-01)	
ER ($p = 0.06, n = 100$)	**1.80e-01** (8.03e-03)	**1.76e-01** (5.28e-03)	7.39e+00 (4.47e-01)	3.19e+01 (5.47e-01)	2.13e+01 (2.08e+00)	**2.06e+00** (2.62e-02)	3.63e-01 (1.34e-02)	8.58e+00 (9.51e-02)	9.26e+00 (5.83e-01)	1.61e+01 (6.00e-01)	
ER ($p = 0.08, n = 100$)	**1.78e-01** (8.39e-03)	**1.79e-01** (2.83e-03)	6.81e+00 (2.23e-01)	3.19e+01 (1.33e+00)	2.99e+01 (2.74e-01)	**8.62e-01** (9.19e-02)	3.47e-01 (2.15e-02)	9.13e+00 (3.42e-01)	2.27e+00 (2.18e-02)	1.35e+01 (3.76e-01)	
ER ($p = 0.04, n = 200$)	**3.63e-01** (9.09e-03)	**3.58e-01** (1.47e-02)	1.52e+00 (8.83e-02)	1.81e+01 (1.01e+00)	6.74e+00 (5.36e-01)	7.86e-01 (1.93e-02)	7.39e-01 (6.13e-02)	3.07e+00 (8.83e-02)	**5.90e-01** (5.74e-02)	2.06e+00 (1.36e-02)	
ER ($p = 0.06, n = 200$)	**3.19e-01** (6.32e-02)	**2.63e-01** (1.03e-02)	5.28e-01 (1.47e-02)	1.74e+01 (1.88e-01)	4.39e+00 (1.76e-01)	**3.39e-01** (1.68e-02)	6.86e-01 (8.37e-03)	1.61e+00 (7.56e-02)	7.38e-01 (9.22e-03)	1.82e+00 (2.54e-02)	N/A
ER ($p = 0.08, n = 200$)	**2.14e-01** (9.55e-03)	**2.57e-01** (5.22e-02)	5.73e-01 (1.35e-02)	1.55e+01 (7.13e-01)	6.18e+00 (1.41e-01)	**4.27e-01** (2.64e-02)	5.93e-01 (6.76e-03)	9.92e-01 (3.25e-02)	4.33e-01 (1.28e-02)	1.76e+00 (5.41e-02)	
Cora	**1.73e-02** (4.44e-04)	**1.85e-02** (2.99e-03)	1.34e-01 (5.98e-03)	2.40e-01 (8.12e-03)	7.36e-01 (3.65e-02)	1.56e-01 (2.05e-03)	**4.07e-02** (5.47e-04)	4.41e-01 (2.16e-02)	6.81e-02 (1.02e-03)	1.74e-01 (5.03e-03)	
Citeseer	**2.09e-02** (1.04e-03)	**3.41e-02** (2.29e-02)	1.46e-01 (5.76e-03)	2.20e-01 (2.82e-03)	9.04e-01 (8.06e-03)	1.72e-01 (1.51e-03)	**3.58e-02** (6.59e-04)	2.82e-01 (2.63e-02)	1.24e-01 (5.66e-03)	3.09e-01 (5.61e-03)	
PubMed	**7.30e-03** (3.00e-05)	**6.51e-03** (1.68e-04)	2.97e-02 (2.67e-03)	1.69e+00 (9.92e-02)	1.55e-01 (7.06e-03)	8.34e-03 (5.55e-04)	9.15e-03 (5.12e-04)	7.68e-01 (1.38e-02)	2.73e-02 (6.70e-04)	8.72e-02 (1.11e-03)	
Extended Persistence Image (EP)											
ER ($p = 0.04, n = 100$)	1.48e-02 (1.26e-03)	**6.37e-03** (3.30e-03)	6.63e-01 (1.01e-02)	7.83e-01 (1.25e-02)	3.82e+00 (1.41e-01)	4.83e-01 (1.90e-02)	3.76e-02 (3.46e-03)	9.17e-01 (4.13e-02)	7.72e-01 (4.33e-02)	2.39e+00 (1.08e-01)	3.07e-03 (5.43e-05)
ER ($p = 0.06, n = 100$)	8.65e-03 (7.91e-04)	**2.79e-03** (1.42e-03)	6.24e-01 (7.97e-03)	7.35e-01 (1.46e-02)	1.57e+00 (7.12e-02)	4.09e-01 (2.64e-02)	3.54e-02 (2.64e-03)	6.31e-01 (5.82e-02)	5.72e-01 (3.03e-02)	1.15e+00 (5.05e-02)	1.37e-03 (2.26e-05)
ER ($p = 0.08, n = 100$)	**8.82e-03** (2.59e-04)	**2.54e-03** (6.41e-04)	5.71e-01 (1.57e-02)	9.46e-01 (5.08e-02)	6.65e-01 (6.56e-02)	3.92e-02 (1.86e-03)	3.29e-02 (2.03e-03)	4.85e-01 (7.19e-03)	1.07e-02 (1.14e-04)	8.59e-01 (5.42e-02)	1.90e-03 (4.81e-05)
ER ($p = 0.04, n = 200$)	8.91e-03 (9.43e-05)	**5.18e-03** (1.94e-03)	8.04e-02 (2.70e-03)	6.73e-01 (2.66e-02)	1.93e-01 (2.18e-03)	3.72e-02 (2.92e-03)	**2.88e-02** (1.54e-03)	3.92e-01 (2.58e-02)	7.31e-02 (2.31e-03)	2.85e-01 (1.84e-02)	3.24e-03 (1.28e-05)
ER ($p = 0.06, n = 200$)	**7.41e-03** (1.23e-04)	**4.76e-03** (4.62e-04)	1.39e-01 (4.08e-03)	7.29e-01 (2.10e-02)	5.48e-01 (2.81e-02)	3.69e-02 (5.56e-04)	2.57e-02 (8.74e-04)	3.68e-01 (5.41e-03)	8.39e-02 (4.93e-03)	3.28e-01 (2.78e-02)	4.30e-03 (5.53e-05)
ER ($p = 0.08, n = 200$)	**4.57e-03** (6.46e-05)	**1.62e-03** (5.01e-04)	1.35e-01 (3.22e-03)	1.28e+00 (9.03e-02)	1.87e-01 (1.52e-02)	3.96e-02 (9.58e-04)	2.43e-02 (5.02e-04)	3.12e-01 (1.15e-02)	4.12e-02 (6.35e-04)	3.48e-01 (4.65e-03)	1.12e-03 (1.73e-05)
Cora	1.45e-04 (9.14e-06)	**7.41e-05** (1.61e-05)	3.51e-03 (3.30e-04)	2.72e-03 (1.17e-04)	5.04e-02 (3.90e-03)	5.53e-03 (1.73e-04)	6.79e-04 (1.53e-05)	2.71e-02 (1.40e-03)	8.13e-04 (2.59e-05)	8.16e-03 (1.37e-04)	5.80e-05 (3.22e-07)
Citeseer	3.94e-04 (2.45e-05)	**1.58e-04** (2.91e-05)	7.64e-03 (6.28e-04)	6.83e-03 (6.02e-04)	2.93e-02 (1.00e-03)	3.94e-03 (7.05e-05)	9.12e-04 (4.80e-05)	3.16e-02 (7.36e-04)	6.87e-04 (2.17e-05)	4.21e-03 (7.09e-05)	1.29e-04 (9.21e-07)
PubMed	7.93e-03 (3.55e-04)	**3.29e-03** (5.51e-04)	8.96e-02 (5.26e-03)	4.29e-01 (2.60e-02)	4.79e-01 (3.41e-02)	1.27e-01 (1.13e-02)	2.16e-02 (2.77e-04)	1.94e-01 (1.58e-02)	3.37e-02 (7.31e-04)	3.07e-01 (2.06e-02)	2.74e-03 (6.85e-05)
ogbn-mag	5.39e-02 (5.95e-04)	**8.01e-03** (1.28e-03)	–	2.93e-01 (5.77e-03)	4.47e-01 (1.89e-02)	2.49e-01 (3.04e-03)	5.83e-02 (8.10e-04)	3.78e-01 (1.24e-02)	–	–	6.73e-03 (1.04e-04)
ogbn-papers100M	9.03e-02 (6.49e-03)	**3.25e-02** (7.21e-04)	–	6.19e-01 (2.36e-02)	8.03e-01 (4.13e-02)	6.43e-01 (1.15e-02)	2.71e-01 (2.15e-02)	4.05e-01 (1.12e-02)	–	–	3.18e-02 (1.05e-04)

Table 7: Accuracy (mean ± standard deviation) of each model on the ENZYMES, PROTEINS, and MUTAG datasets, averaged over 10-fold cross-validation. DYMAG variants based on heat and wave dynamics achieve the best or second-best performance across all datasets.

Model	ENZYMES	PROTEINS	MUTAG
DYMAG$_{\text{(Heat)}}$	0.82 ± 0.02	0.54 ± 0.04	0.79 ± 0.02
DYMAG$_{\text{(Wave)}}$	0.79 ± 0.02	0.71 ± 0.02	0.83 ± 0.02
DYMAG$_{\text{(Sprott)}}$	0.60 ± 0.04	0.64 ± 0.03	0.74 ± 0.03
MPNN	0.63 ± 0.01	0.67 ± 0.01	0.76 ± 0.02
GAT	0.65 ± 0.01	0.64 ± 0.01	0.75 ± 0.02
GIN	0.68 ± 0.01	0.69 ± 0.02	0.77 ± 0.02
GWT	0.66 ± 0.01	0.66 ± 0.02	0.72 ± 0.02
GraphGPS	0.70 ± 0.02	0.68 ± 0.02	0.78 ± 0.02
GRAND	0.71 ± 0.02	0.65 ± 0.02	0.76 ± 0.02
GRAND++	0.74 ± 0.01	0.69 ± 0.01	0.80 ± 0.02

In Table 6, we present the complete results for predicting Ollivier–Ricci curvature and extended persistence images using synthetic and real-world datasets comprising of Erdős–Rényi graphs and citation networks. For all experiments, DYMAG is trained using a uniform signal as input. Ground truth Ollivier–Ricci curvature values are computed directly from the adjacency matrix, and persistence images are generated using node degree as the filtration function. Results are reported as mean with standard deviation in parentheses. Across all settings, DYMAG variants with heat or wave dynamics consistently outperform baseline methods. Given the high computational cost of computing Ollivier–Ricci curvature on large graphs, we restricted the PubMed evaluation to a subgraph comprising the 2,000 most highly cited papers and omitted this evaluation for OGBN-MAG and OGBN-Papers100M.

In Table 7, we evaluated the performance of DYMAG on three publicly available datasets for biomolecular graph classification from the TUDatasets benchmark [75]. The ENZYMES dataset [76] consists of protein secondary structures with ground truth annotations of catalytic activity. In the PROTEINS dataset [21], the task is to classify whether a protein functions as an enzyme. The MUTAG dataset [77] contains small nitroaromatic compounds, and the task is to classify their mutagenicity on the *S.typhimurium* bacterium. DYMAG achieves strong validation accuracy across all datasets, with the wave equation variant performing best on PROTEINS and MUTAG.

In Table 11, we present results corresponding to Figure 3, where the task is to recover generating parameters of random graphs. On both Erdős–Rényi and stochastic block model (SBM) datasets, DYMAG achieves the best overall performance, further validating its capacity to capture latent structural properties.

In Table 8, we evaluated DYMAG on ogbn-papers100M for node classification, in addition to the curvature prediction (node regression) results. The results show that DYMAG-wave and -heat outperform other baselines.

Table 8: Classification accuracy on the ogbn-papers100M dataset. Results are reported as mean ± standard deviation. DYMAG-wave and DYMAG-heat outperform other baselines. Models marked with a dash (–) did not scale to the dataset's size.

Model	Accuracy (%)
DYMAG $_{\text{(Heat)}}$	68.1 ± 0.3
DYMAG $_{\text{(Wave)}}$	$\mathbf{69.4 \pm 0.3}$
DYMAG $_{\text{(Sprott)}}$	–
MPNN	55.2 ± 0.3
GAT	62.1 ± 0.5
GIN	58.7 ± 0.4
GWT	65.3 ± 0.5
GraphGPS	$\mathbf{68.9 \pm 0.4}$
GRAND	–
GRAND++	–

I.2 Ollivier-Ricci Curvature

Olivier-Ricci curvature [78] is used extensively in the graph learning community for graph rewiring [79], community detection [80, 81], and graph sparsification [82]. In particular, negative curvature values identify bridges between communities in a graph, making them informative for rewiring graphs. In our setting, curvature is an important geometric quantity for graph tasks. Moreover, it is a fairly challenging node level task, and often requires information about optimal transport between nodes [83, 84]. Therefore, it is an ideal target for node regression task.

In Riemannian geometry, Ricci curvature quantifies how a space deviates from being locally Euclidean by measuring volume distortion along geodesics. The discrete Ollivier-Ricci formulation extends this notion to graphs by capturing how neighborhood structures contract or expand under local optimal transport, thereby providing a principled measure of local geometric distortion.

To evaluate the capacity of DYMAG to capture such local geometric properties, we consider the task of predicting node-level Ollivier-Ricci curvature [71]. Ground truth curvature values are computed using the `GraphRicciCurvature` package (v0.5.3.1). The method first calculates edge-level curvature scores via optimal transport between neighborhood distributions, and then aggregates these values to the node level by averaging over all incident edges.

Because Ollivier-Ricci curvature is determined entirely by the graph topology - specifically, the adjacency structure and any edge weights - it does not require node features. We therefore compute curvature values directly from the graph's adjacency matrix. The resulting node-level values are used as regression targets in a node-level prediction task.

We report results on both real-world and synthetic graphs in Table 6. Due to the computational cost of Ricci curvature estimation on large graphs, we restrict evaluation on PubMed (19,717 nodes) to a subgraph comprising the 2,000 most highly cited nodes.

I.3 Extended Persistence Image

To compute extended persistence diagrams for graphs, we define a scalar filtration function $f : V \to \mathbb{R}$ that assigns a real value to each node. By default, we use node degree, but the framework supports any scalar-valued function (e.g., centrality, clustering coefficient, or domain-specific metadata).

From this node-level function, edge values are induced by setting $f(u, v) = \max\{f(u), f(v)\}$. We then construct a filtration over the graph using both sublevel and superlevel sets: in the sublevel filtration, nodes and edges are added in order of increasing f, capturing the evolution of connected components and cycles; in the superlevel filtration, nodes and edges are included in decreasing order, allowing for the identification of global topological features that persist across the entire graph. The extended persistence diagram combines information from ordinary, relative, and extended homology classes to characterize these multiscale topological changes.

Each extended persistence diagram contains a collection of birth–death pairs for two types of features: dimension 0 features correspond to connected components, while dimension 1 features correspond to cycles. To convert these diagrams into a vector representation, we map each birth–death pair (b, d) to a birth–persistence pair (b, p) where $p = d - b$. We then place a Gaussian kernel (with bandwidth $\sigma = 0.005$) centered at each (b, p) coordinate and discretize the resulting function onto a grid to obtain persistence images [70]. To reflect the distinct statistical profiles of the two types of features, we use different grid resolutions: for dimension 0 features, which are typically short-lived, we use a compact 25×1 grid; for dimension 1 features, which exhibit greater variability in both birth and persistence, we use a full 25×25 grid. These two images are flattened and concatenated into a 650-dimensional vector.

We treat persistence image prediction as a graph-level regression task. The pooled graph embedding from DYMAG is passed through a 5-layer MLP to predict the flattened persistence image vector.

Results using node degree as the filtration function are reported in Table 6 (bottom). In Table 13, we present results obtained using clustering coefficient as the filtration function. The clustering coefficient of a node v is defined as

$$c(v) = \frac{2T(v)}{\deg(v)(\deg(v) - 1)},$$

where $T(v)$ is the number of triangles through node v and $\deg(v)$ is the degree of node v. Compared to degree-based filtrations, the higher MSE values in this setting suggest that persistence images generated using clustering coefficients are more challenging to predict.

I.4 Node Classification Accuracy on Homophilic and Heterophilic Datasets

In Table 12, we also consider node classification on homophilic (Pubmed, Citeseer, and Cora) and heterophilic (Texas, Wisconsin, and Cornell) networks from Pei et al. [19]. On the homophilic real-world graph datasets, we see that the wave version of DYMAG outperforms the heat/Sprott versions, achieving performance that is roughly comparable with the more standard GNNs. However, on the heterophilic datasets, we see that DYMAG with chaotic Sprott dynamics outperforms other models.

I.5 Fractional Heat Equation Dynamics

In Tables 9 and 10, we conduct experiments investigating the role of α in the fractional Laplacian L^α. We highlight the case $\alpha = 1$ in gray to emphasize that when $\alpha = 1$, the fractional Laplacian reduces to the standard (non-fractional) graph Laplacian. The values of α that achieve the best performance are highlighted in blue. In cases where there is a tie and $\alpha = 1$ is one of the co-best methods (e.g., in the MP dataset), we highlight the $\alpha = 1$ case in gray and the other top-performing method in blue.

Table 9, reports the mean squared error (MSE) for predicting extended persistence images. We observe that varying α significantly impacts the model's performance. For most Erdős–Rényi (ER) graphs with $n = 100$ nodes, lower values of α yield better performance than the standard Laplacian ($\alpha = 1$), suggesting that fractional heat diffusion processes capture relevant graph features more effectively in this context. For larger graphs with $n = 200$ nodes, the optimal α is still lower than 1, though not as low as 0.25.

In Table 10, which reports the R^2 scores for predicting various geometric and graph topological properties of molecules, we observe that the performance across different α values is relatively similar, indicating robustness to the choice of α. For example, on the PROTEINS dataset for predicting dihedral angles, the highest R^2 score is 0.89 at both $\alpha = 0.25$ and $\alpha = 0.50$, while at $\alpha = 1$, the score is 0.87. In the case of the Materials Project (MP) dataset for predicting band gap, there is a tie in performance between $\alpha = 0.50$ and $\alpha = 1.00$, both achieving an R^2 score of 0.59.

Overall, these results demonstrate that non-local smoothing achieved with the fractional Laplacian featuring various α parameters allows the model to perform better on certain tasks. Specifically, fractional Laplacians with $\alpha < 1$ can enhance performance in recovering the topology of randomly generated graphs, while different values of α do not significantly impact DYMAG's performance on molecular and material science datasets.

Table 9: Mean squared error (MSE) for predicting extended persistence images using vertex degree as the filtration function (lower is better). We compare DYMAG models with wave dynamics across different fractional orders α. The first group of ER graphs are generated with $n = 100$ nodes, and the second with $n = 200$ nodes.

Graph	**Fraction α**			
	0.25	0.50	0.75	1.00
ER($p = 0.04, n = 100$)	4.47e-2 ± 1.0e-3	**3.51e-2 ± 7.2e-4**	**1.39e-2 ± 2.6e-4**	**1.48e-02 ± 1.26e-03**
ER($p = 0.06, n = 100$)	**3.60e-3 ± 1.6e-4**	**6.03e-3 ± 8.3e-5**	**6.24e-3 ± 1.8e-4**	8.65e-03 ± 7.91e-04
ER($p = 0.08, n = 100$)	**4.72e-3 ± 1.8e-4**	**7.39e-3 ± 2.1e-4**	8.60e-3 ± 3.1e-4	8.82e-03 ± 2.59e-04
ER($p = 0.04, n = 200$)	**8.12e-3 ± 5.0e-4**	**7.73e-3 ± 3.4e-4**	9.28e-3 ± 8.5e-4	**8.91e-03 ± 9.43e-05**
ER($p = 0.06, n = 200$)	7.85e-3 ± 3.6e-4	**4.98e-3 ± 9.3e-5**	**4.52e-3 ± 1.8e-4**	7.41e-03 ± 1.23e-04
ER($p = 0.08, n = 200$)	5.02e-3 ± 1.6e-4	**3.47e-3 ± 9.7e-5**	**1.12e-3 ± 2.5e-4**	4.57e-03 ± 6.46e-05
Cora	2.84e-3 ± 3.2e-4	1.79e-3 ± 4.0e-4	**5.16e-4 ± 8.0e-6**	**1.45e-04 ± 9.14e-06**
Citeseer	1.35e-3 ± 9.7e-5	1.04e-3 ± 1.0e-4	**6.34e-4 ± 1.2e-5**	**3.94e-04 ± 2.45e-05**
PubMed	5.62e-3 ± 8.8e-5	**1.17e-4 ± 1.0e-5**	**2.35e-4 ± 7.4e-6**	7.93e-03 ± 3.55e-04

Table 10: Performance of DYMAG (heat dynamics) across different fractional orders α on four datasets: PROTEINS, DrugBank, Materials Project (MP), and the DTS AIDS Antiviral Screen. We report R^2 score (higher is better) for the first three datasets and balanced accuracy for the Antiviral Screen. Results are reported as mean $\pm$ standard deviation over 10-fold cross-validation.

α	**PROTEINS**	**DrugBank**		**MP**	**Antiviral Screen**
	Dihedral Angles	TPSA	# Aromatic Rings	Band Gap	Active/Inactive
0.25	**0.89 ± 0.04**	**0.94 ± 0.02**	**0.96 ± 0.03**	**0.57 ± 0.04**	**0.52 ± 0.02**
0.50	**0.89 ± 0.03**	0.92 ± 0.02	**0.96 ± 0.02**	**0.59 ± 0.01**	**0.56 ± 0.01**
0.75	**0.86 ± 0.02**	0.92 ± 0.03	**0.97 ± 0.03**	**0.58 ± 0.02**	**0.56 ± 0.02**
1.00	**0.89 ± 0.01**	**0.97 ± 0.01**	**0.97 ± 0.02**	**0.61 ± 0.03**	**0.54 ± 0.02**

Table 11: Mean squared error (MSE) for the prediction of generating parameters of random graphs (lower is better). The number of nodes for each type of random graph is specified in each data column (i.e. $n \in \{100, 250, 500, 1000, 2500\}$).

Method	**Erdős–Rényi**					**Stochastic Block Model**				
	100	250	500	1000	2500	100	250	500	1000	2500
DYMAG$_{(Heat)}$	**7.46e-3**	**7.13e-3**	**3.60e-3**	4.19e-3	**3.04e-3**	**6.41e-1**	**8.10e-1**	**1.79**	4.52	**6.27**
DYMAG$_{(Wave)}$	**8.29e-3**	**6.58e-3**	**3.17e-3**	**3.25e-3**	**1.04e-3**	**8.25e-1**	**9.40e-1**	**1.26**	**2.28**	**2.35**
DYMAG$_{(Sprott)}$	4.33e-2	4.92e-2	7.08e-3	**3.68e-3**	5.49e-3	5.17	3.37	4.25	**4.08**	**6.27**
MPNN	1.37e-2	1.14e-2	9.26e-3	9.49e-3	8.02e-3	2.93	3.07	3.68	7.14	10.26
GAT	3.05e-2	5.60e-2	1.35e-2	3.74e-2	2.69e-2	11.79	9.42	10.83	13.62	18.60
GIN	1.08e-2	9.37e-3	7.74e-3	6.98e-3	4.81e-3	1.74	2.59	2.92	4.37	9.15
GWT	9.72e-3	1.04e-2	6.29e-3	6.56e-3	5.41e-3	2.47	3.18	2.14	4.87	6.52
GraphGPS	5.28e-2	8.48e-2	1.26e-2	1.31e-2	8.24e-3	12.06	8.21	9.44	11.63	12.67
GRAND	6.36e-2	4.22e-2	9.27e-3	6.58e-3	5.30e-3	14.52	16.78	13.50	11.28	8.58
GRAND++	8.52e-2	6.91e-2	2.84e-2	1.29e-2	8.72e-3	23.71	26.84	19.64	16.97	15.42

Table 12: Node classification accuracy (%) on homophilic and heterophilic datasets.

Method	**Homophilic Datasets**			**Heterophilic Datasets**		
	Cora	Citeseer	PubMed	Cornell	Wisconsin	Texas
Homophily	0.81	0.80	0.74	0.30	0.21	0.11
Nodes	2,708	3,312	19,717	183	251	183
Classes	7	6	3	5	5	5
DYMAG$_{(Heat)}$	88.16	76.92	89.73	73.52	67.46	64.41
DYMAG$_{(Wave)}$	**89.62**	77.16	89.63	76.44	78.47	81.24
DYMAG$_{(Sprott)}$	60.81	67.42	64.18	**88.19**	**86.72**	**87.63**
MPNN	83.93	72.81	80.43	65.17	65.29	45.87
GAT	87.28	75.03	86.94	54.27	59.14	48.62
GIN	88.95	76.04	89.74	74.68	68.47	73.87
GWT	86.23	75.92	88.37	70.34	66.25	62.11
GraphGPS	87.31	75.87	88.91	73.95	69.13	74.01
GRAND	84.18	73.62	80.39	81.94	74.65	77.06
GRAND++	84.33	75.61	80.53	80.27	78.38	82.58
TrigoNet	**90.07 ± 1.33**	**78.91 ± 1.28**	**89.93 ± 0.58**	80.12 ± 3.77	-	81.53 ± 4.02
PDE-GCN$_M$	88.60	**78.48**	**89.93**	**89.73**	**91.76**	**93.24**
GCN+Multiscale QDC	87.85 ± 5.44	73.78 ± 4.53	88.32 ± 0.47	66.22 ± 5.44	64.71 ± 4.47	73.70 ± 4.53
GAT+Multiscale QDC	87.68 ± 3.87	77.57 ± 5.56	88.04 ± 3.33	77.57 ± 3.87	**88.04 ± 4.06**	67.57 ± 5.56
H2GNN+Multiscale QDC	88.38 ± 3.72	77.34 ± 4.84	89.13 ± 3.51	76.01 ± 3.72	86.66 ± 3.65	86.84 ± 4.17
GCNII*	88.01	77.13	**90.30**	76.49	81.57	77.84
Diag-NSD	87.14 ± 1.06	77.14 ± 1.85	89.42 ± 0.43	86.49 ± 7.35	88.63 ± 2.75	85.67 ± 6.95
O(d)-NSD	86.90 ± 1.13	76.70 ± 1.57	89.49 ± 0.40	84.86 ± 4.71	89.41 ± 4.74	85.95 ± 5.51
Gen-NSD	87.30 ± 1.34	76.32 ± 1.65	89.33 ± 0.35	85.68 ± 6.51	89.21 ± 3.84	82.97 ± 5.13

Table 13: MSE (lower is better) for extended persistence image prediction (clustering-coefficient filtration, degree features).

Model	Cora	Citeseer	PubMed
$\text{DYMAG}_{\text{(Heat)}}$	**2.48**	7.35	**1.28**
$\text{DYMAG}_{\text{(Wave)}}$	**1.76**	**2.45**	6.04
$\text{DYMAG}_{\text{(Sprott)}}$	8.37	13.58	6.26
MPNN	4.20	12.6	7.94
GAT	9.25	4.45	7.50
GIN	9.89	7.34	2.17
GWT	4.12	6.94	3.22
GraphGPS	14.3	11.7	9.45
GRAND	13.1	10.8	6.07
GRAND++	15.1	6.49	4.75
Neural EPD Approx.	9.38	1.94	4.52

J Dataset Description

Cora [16] is a citation network comprising 2,708 scientific publications classified into one of seven categories. Each node represents a publication and is associated with a 1,433-dimensional binary feature vector indicating the presence or absence of specific words from a predefined dictionary. Edges represent the 5,429 citation links between documents.

Citeseer [17] is a citation network containing 3,312 scientific publications categorized into six classes. Each node corresponds to a publication and is described by a 3,703-dimensional binary feature vector based on the presence or absence of specific dictionary words. The graph includes 4,732 citation links, forming edges between related documents.

PubMed [18] is a citation network of 19,717 biomedical research articles from the PubMed database, all related to diabetes, and categorized into three classes. Each node represents a publication and is associated with a 500-dimensional feature vector based on TF-IDF weighted word frequencies. The graph contains 44,338 citation edges.

Cornell, **Texas**, and **Wisconsin** [19] are subgraphs extracted from the WebKB dataset, comprising webpages from the computer science departments of the respective universities. Each node represents a webpage, described by a bag-of-words feature vector derived from its textual content. Edges correspond to hyperlinks between pages. The classification task involves predicting the type of webpage (e.g., student, faculty, course, project, staff). These graphs are relatively small, with 183-251 nodes and 295-499 edges. Notably, all three datasets exhibit strong heterophily, where connected nodes often belong to different classes, posing a challenge for traditional homophily-based graph learning methods.

ogbn-papers100M and **ogbn-mag** are large-scale academic graphs from the Open Graph Benchmark (OGB) collection. ogbn-papers100M is a directed citation network comprising over 111 million papers indexed in the Microsoft Academic Graph (MAG) [85], where each node represents a paper with a 128-dimensional word2vec feature vector, and edges denote citation links. ogbn-mag is a heterogeneous graph also derived from MAG, containing four node types - papers (736K), authors (1.1M), institutions (8.7K), and fields of study (60K) - and four directed edge types: authorship, citation, affiliation, and topic assignment. Only paper nodes have input features (128-dimensional word2vec embeddings), while the other node types are featureless.

PROTEINS [21], part of the TUDataset benchmark suite [75], is a graph classification dataset consisting of 1,113 protein structures, each labeled as either an enzyme or a non-enzyme. In each graph, nodes represent amino acids, and edges are formed between pairs of amino acids that are within 6 Ångströms of each other in 3D space.

ENZYMES [76], part of the TUDataset benchmark suite [75], contains 600 protein tertiary structures categorized into six enzyme classes, as defined by the BRENDA enzyme database. Each protein is represented as a graph, where nodes correspond to amino acids and edges capture spatial or sequential proximity.

MUTAG [86], part of the TUDataset benchmark suite [75], is a graph classification dataset consisting of 188 chemical compounds labeled according to their mutagenic effect on Salmonella typhimurium. Each compound is represented as a graph, where nodes correspond to atoms (with one-hot encoded atom types as features) and edges represent chemical bonds. There are 7 discrete node labels. The task is to predict the binary mutagenicity label based on molecular structure.

DrugBank [22] is a publicly available resource that integrates detailed information about drugs and their molecular targets. We use version 5.0 of the database, released in 2018, which contains 6,712 drug entries, including 1,448 FDA-approved small-molecule drugs. While the database includes a wide range of chemical, pharmacological, and structural properties, we focus on predicting two geometry- and topology-related molecular attributes: total polar surface area (TPSA) and the number of aromatic rings. Each molecule is represented as a graph, with atoms as nodes and bonds as edges.

The **Materials Project (MP)** dataset [24] consists of a large collection of inorganic compounds labeled with physical and chemical properties computed using density functional theory (DFT). We use version 2018.6.1, which includes 69,239 materials and a range of properties such as formation energy, bulk and shear moduli, and electronic band gap. In our experiments, we focus on predicting the band gap (e.g., in eV), a key electronic property available for 45,901 compounds. Each material is represented as a graph, with atoms as nodes and edges defined by interatomic bonds or distances derived from crystal structures.

The **Antiviral Screen Dataset** [23] originates from the Drug Therapeutics Program (DTP) AIDS Antiviral Screen, which evaluated the anti-HIV activity of 43,850 chemical compounds based on their ability to inhibit HIV replication. Each compound is represented as a molecular graph, with atoms as nodes and bonds as edges. Screening outcomes were originally categorized into three groups: confirmed active (CA), confirmed moderately active (CM), and confirmed inactive (CI). As part of the MoleculeNet benchmark [87], the CA and CM categories are merged, resulting in a binary classification task: predicting whether a compound is active (CA/CM) or inactive (CI).