Glossary of Notation and Symbols

The next list describes several symbols that will be later used within the body of the document

$\{\{\cdot,\cdot\}\}$	(Irreversible) double bracket on functions with generator $\mathbf{L}^2$
$[\cdot,\cdot]$	Degenerate (irreversible) metric bracket on functions with generator $\mathbf{M}^\intercal = \mathbf{M}$
$\{\cdot,\cdot\}$	Poisson (reversible) bracket on functions with generator $\mathbf{L}^\intercal = -\mathbf{L}$
$\mathcal{N}(i), \overline{\mathcal{N}}(i)$	Neighbors of node $i \in \mathcal{V}$, neighbors of node $i \in \mathcal{V}$ including i
$[S]$	Indicator function of the statement S
$\delta f, \nabla f$	Adjoint of df with respect to $\langle\cdot,\cdot\rangle$, adjoint of df with respect to $(\cdot,\cdot)$
Δ_k	Hodge Laplacian $d_k d_k^* + d_k^* d_k$
δ_{ij}	Kronecker delta
$\dot{f}$	Derivative of f with respect to time
$(\cdot,\cdot)_k$	Learnable metric inner product on k-cliques with matrix representation $\mathbf{A}_k$
$\langle\cdot,\cdot\rangle_k$	Euclidean ℓ^2 inner product on k-cliques
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	Oriented graph, set of nodes, set of edges
$\mathcal{G}_k, \Omega_k$	Set of k-cliques, vector space of real-valued functions on k-cliques
d, d_k	Exterior derivative operator on functions, exterior derivative operator on k-cliques
$d_k^\intercal, d_k^*$	Adjoint of d_k with respect to $\langle\cdot,\cdot\rangle_k$, adjoint of d_k with respect to $(\cdot,\cdot)_k$

A Mathematical foundations

This Appendix provides the following: (1) an introduction to the ideas of graph exterior calculus, A.1, and bracket-based dynamical systems, A.2, necessary for understanding the results in the body, (2) additional explanation regarding adjoints with respect to generic inner products and associated computations, A.3, (3) a mechanism for higher-order attention expressed in terms of learnable inner products, A.4, (4) a discussion of GATs in the context of exterior calculus, A.5, and (5) proofs which are deferred from Section 4, A.6.

A.1 Graph exterior calculus

Here some basic notions from the graph exterior calculus are recalled. More details can be found in, e.g., [11, 51, 64].

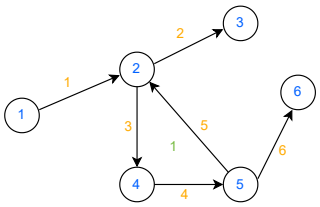


Figure 3: A toy graph with six 0-cliques (nodes), six 1-cliques (edges), and one 2-clique.

As mentioned in Section 3, an oriented graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ carries sets of *k-cliques*, denoted $\mathcal{G}_k$, which are collections of ordered subgraphs generated by $(k+1)$ nodes. For example, the graph in Figure 3 contains six 0-cliques (nodes), six 1-cliques (edges), and one 2-clique. A notion of combinatorial derivative is then given by the *signed incidence matrices* $d_k : \Omega_k \to \Omega_{k+1}$, operating on the space Ω_k of differentiable functions on k-cliques, whose entries $(d_k)_{ij}$ are 1 or -1 if the j^{th} k-clique is

incident on the i^{th} $(k+1)$-clique, and zero otherwise. For the example in Figure 3, these are:

$$d_0 = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}, \qquad d_1 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Remark A.1. *While the one-hop neighborhood of node i in $\mathcal{G}$, denoted $\mathcal{N}(i)$, does not include node i itself, many machine learning algorithms employ the extended neighborhood $\overline{\mathcal{N}}(i) = \mathcal{N}(i) \cup \{i\}$. Since this is equivalent to considering the one-hop neighborhood of node i in the self-looped graph $\overline{\mathcal{G}}$, this modification does not change the analysis of functions on graphs.*

It can be shown that the action of these matrices can be conveniently expressed in terms of totally antisymmetric functions $f \in \Omega_k$, via the expression

$$(d_k f)(i_0, i_1, ..., i_{k+1}) = \sum_{j=0}^{k+1} (-1)^j f\left(i_0, ..., \widehat{i_j}, ..., i_{k+1}\right),$$

where $(i_0, ..., i_{k+1})$ denotes a $(k+1)$-clique of vertices $v \in \mathcal{V}$. As convenient shorthand, we often write subscripts, e.g., $(d_k f)_{i_0 i_1 ... i_{k+1}}$, instead of explicit function arguments. Using $[S]$ to denote the indicator function of the statement S, it is straightforward to check that $d \circ d = 0$,

$$\begin{aligned}
(d_k d_{k-1} f)_{i_0,...,i_{k+1}} &= \sum_{j=0}^{k+1} (-1)^j (d_{k-1} f)_{i_0,...,\widehat{i_j},...,i_{k+1}} \\
&= \sum_{j=0}^{k+1} \sum_{l=0}^{k+1} [l < j] (-1)^{j+l} f_{i_0 ... \widehat{i_l} ... \widehat{i_j} ... i_{k+1}} \\
&\quad + \sum_{j=0}^{k+1} \sum_{l=0}^{k+1} [l > j] (-1)^{j+l-1} f_{i_0 ... \widehat{i_j} ... \widehat{i_l} ... i_{k+1}} \\
&= \sum_{l<j} (-1)^{j+l} f_{i_0 ... \widehat{i_l} ... \widehat{i_j} ... i_{k+1}} \\
&\quad - \sum_{l<j} (-1)^{j+l} f_{i_0 ... \widehat{i_l} ... \widehat{i_j} ... i_{k+1}} = 0,
\end{aligned}$$

since $(-1)^{j+l-1} = (-1)^{-1}(-1)^{j+l} = (-1)(-1)^{j+l}$ and the final sum follows from swapping the labels j, l. This shows that the k-cliques on $\mathcal{G}$ form a *de Rham complex* [53]: a collection of function spaces Ω_k equipped with mappings d_k satisfying $\operatorname{Im} d_{k-1} \subset \operatorname{Ker} d_k$ as shown in Figure 4. When

$$\Omega_0 \xrightarrow{d_0} \Omega_1 \xrightarrow{d_1} \Omega_2 \xrightarrow{d_2} \cdots \xrightarrow{d_{K-1}} \Omega_K$$

Figure 4: Illustration of the de Rham complex on $\mathcal{G}$ induced by the combinatorial derivatives, where $K > 0$ is the maximal clique degree.

$K = 3$, this is precisely the graph calculus analogue of the de Rham complex on $\mathbb{R}^3$ formed by the Sobolev spaces $H^1, H(\text{curl}), H(\text{div}), L^2$ which satisfies $\text{div} \circ \text{curl} = \text{curl} \circ \text{grad} = 0$.

While the construction of the graph derivatives and their associated de Rham complex is purely topological, building elliptic differential operators such as the Laplacian relies on a dual de Rham complex, which is specified by an inner product on Ω_k. In the case of ℓ^2, this leads to dual derivatives which are the matrix transposes of the d_k having the following explicit expression.

Proposition A.1. *The dual derivatives $d_k^\intercal : \Omega_{k+1} \to \Omega_k$ adjoint to d_k through the ℓ^2 inner product are given by*

$$(d_k^\intercal f)(i_0, i_1, ..., i_k) = \frac{1}{k+2} \sum_{i_{k+1}} \sum_{j=0}^{k+1} f(i_0, ..., [i_j, ..., i_{k+1}]),$$

where $[i_j, ..., i_{k+1}] = i_{k+1}, i_j, ..., i_k$ indicates a cyclic permutation forward by one index.

Proof. This is a direct calculation using the representation of d_k in terms of antisymmetric functions. More precisely, let an empty sum Σ denote summation over all unspecified indices. Then, for any $g \in \Omega_k$,

$$\begin{aligned}
\langle d_k f, g \rangle &= \sum_{i_0 \dots i_{k+1} \in \mathcal{G}_{k+1}} (d_k f)_{i_0 \dots i_{k+1}} \, g_{i_0 \dots, i_{k+1}} \\
&= \frac{1}{(k+2)!} \sum \left(\sum_{j=0}^{k+1} (-1)^j f_{i_0 \dots \widehat{i_j} \dots i_{k+1}} \right) g_{i_0 \dots i_{k+1}} \\
&= \frac{1}{(k+2)!} \sum f_{i_0 \dots i_k} \left(\sum_{i_{k+1}} \sum_{j=0}^{k+1} (-1)^j g_{i_0 \dots [i_j \dots i_{k+1}]} \right) \\
&= \frac{1}{k+2} \sum_{i_0 i_1 \dots i_k \in \mathcal{G}_k} f_{i_0 \dots i_k} \left(\sum_{i_{k+1}} \sum_{j=0}^{k+1} (-1)^j g_{i_0 \dots [i_j \dots i_{k+1}]} \right) \\
&= \sum_{i_0 i_1 \dots i_k \in \mathcal{G}_k} f_{i_0 \dots i_k} \, (d_k^\intercal g)_{i_0 \dots i_k} = \langle f, d_k^\intercal g \rangle \,,
\end{aligned}$$

which establishes the result. □

Proposition A.1 is perhaps best illustrated with a concrete example. Consider the graph gradient, defined for edge $\alpha = (i, j)$ as $(d_0 f)_\alpha = (d_0 f)_{ij} = f_j - f_i$. Notice that this object is antisymmetric with respect to edge orientation, and measures the outflow of information from source to target nodes. From this, it is easy to compute the ℓ^2-adjoint of d_0, known as the graph divergence, via

$$\begin{aligned}
\langle d_0 f, g \rangle &= \sum_{\alpha=(i,j)} (f_j - f_i) \, g_{ij} = \sum_i \sum_{(j>i) \in \mathcal{N}(i)} g_{ij} f_j - g_{ij} f_i \\
&= \frac{1}{2} \sum_i \sum_{j \in \mathcal{N}(i)} f_i \, (g_{ji} - g_{ij}) = \langle f, d_0^\intercal g \rangle \,,
\end{aligned}$$

where we have re-indexed under the double sum, used that $i \in \mathcal{N}(j)$ if and only if $j \in \mathcal{N}(i)$, and used that there are no self-edges in $\mathcal{E}$. Therefore, it follows that the graph divergence at node i is given by

$$(d_0^\intercal g)_i = \sum_{\alpha \ni i} g_{-\alpha} - g_\alpha = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} g_{ji} - g_{ij},$$

which reduces to the common form $(d_0^\intercal g)_i = -\sum_j g_{ij}$ if and only if the edge feature g_{ij} is antisymmetric.

Remark A.2. *When the inner product on edges $\mathcal{E}$ is not L^2, but defined in terms of a nonnegative, orientation-invariant, and (edge-wise) diagonal weight matrix $\mathbf{W} = (w_{ij})$, a similar computation shows that the divergence becomes*

$$(d_0^* f)_i = \frac{1}{2} \sum_{j \in \mathcal{N}(i)} w_{ij} \, (f_{ji} - f_{ij}) \,.$$

The more general case of arbitrary inner products on $\mathcal{V}, \mathcal{E}$ is discussed in section A.3.

The differential operators $d_k^\intercal$ induce a dual de Rham complex since $d_{k-1}^\intercal d_k^\intercal = (d_k d_{k-1})^\intercal = 0$, which enables both the construction of Laplace operators on k-cliques, $\Delta_k = d_k^\intercal d_k + d_{k-1} d_{k-1}^\intercal$, as well as the celebrated Hodge decomposition theorem, stated below. For a proof, see, e.g., [11, Theorem 3.3].

Theorem A.3. *(Hodge Decomposition Theorem) The de Rham complexes formed by $d_k, d_k^\intercal$ induce the following direct sum decomposition of the function space Ω_k,*

$$\Omega_k = \operatorname{Im} d_{k-1} \oplus \operatorname{Ker} \Delta_k \oplus \operatorname{Im} d_k^\intercal.$$

In the case where the dual derivatives d_k^* are adjoint with respect to a learnable inner product which does not depend on graph features, the conclusion of Theorem A.3 continues to hold, leading to an interesting well-posedness result proved in [11] involving nonlinear perturbations of a Hodge-Laplace problem in mixed form.

Theorem A.4. *([11, Theorem 3.6]) Suppose $\mathbf{f}_k \in \Omega_k$, and $g(\mathbf{x};\xi)$ is a neural network with parameters ξ which is Lipschitz continuous and satisfies $g(\mathbf{0}) = \mathbf{0}$. Then, the problem*

$$\mathbf{w}_{k-1} = d_{k-1}^* \mathbf{u}_k + \epsilon g\left(d_{k-1}^* \mathbf{u}_k; \xi\right),$$
$$\mathbf{f}_k = d_{k-1}\mathbf{w}_{k-1} + d_k^* d_k \mathbf{u}_k,$$

has a unique solution on $\Omega_k / \mathrm{Ker}\, \Delta_k$.

This result shows that initial-value problems involving the Hodge-Laplacian are stable under nonlinear perturbations. Moreover, when Δ_0 is the Hodge Laplacian on nodes, there is a useful connection between Δ_0 and the degree and adjacency matrices of the graph $\mathcal{G}$. Recall that the degree matrix $\mathbf{D} = (d_{ij})$ is diagonal with entries $d_{ii} = \sum_{j \in \mathcal{N}(i)} 1$, while the adjacency matrix $\mathbf{A} = (a_{ij})$ satisfies $a_{ij} = 1$ when $j \in \mathcal{N}(i)$ and $a_{ij} = 0$ otherwise.

Proposition A.2. *The combinatorial Laplacian on $\mathcal{V}$, denoted $\Delta_0 = d_0^\intercal d_0$, satisfies $\Delta_0 = \mathbf{D} - \mathbf{A}$.*

Proof. Notice that

$$\begin{aligned}(d_0^\intercal d_0)_{ij} &= \sum_{\alpha \in \mathcal{E}} (d_0)_{\alpha i} (d_0)_{\alpha j} = [i=j] \sum_{\alpha \in \mathcal{E}} \left((d_0)_{\alpha i}\right)^2 + [i \neq j] \sum_{\alpha = (i,j)} (d_0)_{\alpha i} (d_0)_{\alpha j} \\ &= [i=j]\, d_{ii} - [i \neq j]\, a_{ij} = d_{ij} - a_{ij} = \mathbf{D} - \mathbf{A},\end{aligned}$$

where we used that $\mathbf{D}$ is diagonal, $\mathbf{A}$ is diagonal-free, and $(d_0)_{\alpha i} (d_0)_{\alpha j} = -1$ whenever $\alpha = (i,j)$ is an edge in $\mathcal{E}$, since one of $(d_0)_{\alpha i}, (d_0)_{\alpha j}$ is 1 and the other is -1. □

A.2 Bracket-based dynamical systems

Here we mention some additional facts regarding bracket-based dynamical systems. More information can be found in, e.g., [50, 65, 66, 67].

As mentioned before, the goal of bracket formalisms is to extend the Hamiltonian formalism to systems with dissipation. To understand where this originates, consider an action functional $\mathcal{A}(q) = \int_a^b L(q, \dot{q})\, dt$ on the space of curves $q(t)$, defined in terms of a Lagrangian L on the tangent bundle to some Riemannian manifold. Using $L_q, L_{\dot{q}}$ to denote partial derivatives with respect to the subscripted variable, it is straightforward to show that, for any compactly supported variation δq of q, we have

$$d\mathcal{A}(q)\delta q = \int_a^b dL(q, \dot{q})\, \delta q = \int_a^b L_q \delta q + L_{\dot{q}} \delta \dot{q} = \int_a^b \left(L_q - \partial_t L_{\dot{q}}\right) \delta q,$$

where the final equality follows from integration-by-parts and the fact that variational and temporal derivatives commute in this setting. It follows that $\mathcal{A}$ is stationary (i.e., $d\mathcal{A} = 0$) for all variations only when $\partial_t L_{\dot{q}} = L_q$. These are the classical Euler-Lagrange equations which are (under some regularity conditions) transformed to Hamiltonian form via a Legendre transformation,

$$H(q,p) = \sup_{\dot{q}} \left(\langle p, \dot{q} \rangle - L(q, \dot{q})\right),$$

which defines the Hamiltonian functional $\mathcal{H}$ on phase space, and yields the conjugate momentum vector $p = L_{\dot{q}}$. Substituting $L = \langle p, \dot{q} \rangle - H$ into the previously derived Euler-Lagrange equations leads immediately to Hamilton's equations for the state $\mathbf{x} = (q \quad p)^\intercal$,

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{q} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} H_q \\ H_p \end{pmatrix} = \mathbf{J} \nabla \mathcal{H},$$

which are an equivalent description of the system in question in terms of the anti-involution $\mathbf{J}$ and the functional gradient $\nabla \mathcal{H}$.

An advantage of the Hamiltonian description is its compact bracket-based formulation, $\dot{\mathbf{x}} = \mathbf{J}\nabla\mathcal{H} = \{\mathbf{x}, \mathcal{H}\}$, which requires only the specification of an antisymmetric Poisson bracket $\{\cdot,\cdot\}$ and a Hamiltonian functional $\mathcal{H}$. Besides admitting a direct generalization to more complex systems such as Korteweg-de Vries or incompressible Euler, where the involved bracket is state-dependent, this formulation makes the energy conservation property of the system obvious. In particular, it follows immediately from the antisymmetry of $\{\cdot,\cdot\}$ that

$$\dot{\mathcal{H}} = \langle \dot{\mathbf{x}}, \nabla \mathcal{H} \rangle = \{\mathcal{H}, \mathcal{H}\} = 0,$$

while it is more difficult to see immediately that the Euler-Lagrange system obeys this same property. The utility and ease-of-use of bracket formulations is what inspired their extension to other systems of interest which do not conserve energy. On the opposite end of this spectrum are the generalized gradient flows, which can be written in terms of a bracket which is purely dissipative. An example of this is heat flow $\dot{q} = \Delta q := -[q, \mathcal{D}]$, which is the L^2-gradient flow of Dirichlet energy $\mathcal{D}(q) = (1/2)\int_a^b |q'|^2\, dt$ (c.f. Appendix A.3). In this case, the functional gradient $\nabla\mathcal{D} = -\partial_{tt}$ is the negative of the usual Laplace operator, so that the positive-definite bracket $[\cdot,\cdot]$ is generated by the identity operator $M = \mathrm{id}$. It is interesting to note that the same system could be expressed using the usual kinetic energy $\mathcal{E}(q) = (1/2)\int_a^b |q|^2\, dt$ instead, provided that the corresponding bracket is generated by $M = -\Delta$. This is a good illustration of the flexibility afforded by bracket-based dynamical systems.

Since physical systems are not always purely reversible or irreversible, other useful bracket formalisms have been introduced to capture dynamics which are a mix of these two. The double bracket $\dot{\mathbf{x}} = \{\mathbf{x}, E\} + \{\{\mathbf{x}, E\}\} = \mathbf{L}\nabla E + \mathbf{L}^2\nabla E$ is a nice extension of the Hamiltonian bracket particularly because it is Casimir preserving, i.e., those quantities which annihilate the Poisson bracket $\{\cdot,\cdot\}$ also annihilate the double bracket. This allows for the incorporation of dissipative phenomena into idealized Hamiltonian systems without affecting desirable properties such as mass conservation, and has been used to model, e.g., the Landau-Lifschitz dissipative mechanism, as well as a mechanism for fluids where energy decays but entrophy is preserved (see [65] for additional discussion). A complementary but alternative point of view is taken by the metriplectic bracket formalism, which requires that any dissipation generated by the system is accounted for within the system itself through the generation of entropy. In the metriplectic formalism, the equations of motion are $\dot{\mathbf{x}} = \{\mathbf{x}, E\} + [\mathbf{x}, S] = \mathbf{L}\nabla E + \mathbf{M}\nabla S$, along with important and nontrivial compatibility conditions $\mathbf{L}\nabla S = \mathbf{M}\nabla E = \mathbf{0}$, also called degeneracy conditions, which ensure that the reversible and irreversible mechanisms do not cross-contaminate. As shown in the body of the paper, this guarantees that metriplectic systems obey a form of the first and second thermodynamical laws. Practically, the degeneracy conditions enforce a good deal of structure on the operators $\mathbf{L}, \mathbf{M}$ which has been exploited to generate surrogate models [29, 68, 31]. In particular, it can be shown that the reversible and irreversible brackets can be parameterized in terms of a totally antisymmetric order-3 tensor $\boldsymbol{\xi} = (\xi_{ijk})$ and a partially symmetric order-4 tensor $\boldsymbol{\zeta} = (\zeta_{ik,jl})$ through the relations (Einstein summation assumed)

$$\{A, B\} = \xi^{ijk}\, \partial_i A\, \partial_j B\, \partial_k S,$$
$$[A, B] = \zeta^{ik,jl}\, \partial_i A\, \partial_k E\, \partial_j B\, \partial_l E.$$

Moreover, using the symmetries of $\boldsymbol{\zeta}$, it follows (see [67]) that this tensor decomposes into the product $\zeta_{ik,jl} = \Lambda^m_{ik} D_{mn} \Lambda^n_{jl}$ of a symmetric matrix $\mathbf{D}$ and an order-3 tensor Λ which is skew-symmetric in its lower indices. Thus, by applying symmetry relationships, it is easy to check that $\{\cdot, S\} = [\cdot, E] = \mathbf{0}$.

Remark A.5. *In [29], trainable 4- and 3- tensors ξ^{ijk} and $\zeta^{ik,jl}$ are constructed to achieve the degeneracy conditions, mandating a costly $O(N^3)$ computational complexity. In the current work we overcome this by instead achieving degeneracy through the exact sequence property.*

A.3 Adjoints and gradients

Beyond the basic calculus operations discussed in section A.1 which depend only on graph topology, the network architectures discussed in the body also make extensive use of learnable metric information coming from the nodal features. To understand this, it is useful to recall some information about general inner products and the derivative operators that they induce. First, recall that the usual ℓ^2 inner product on node features $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{|\mathcal{V}|}$, $\langle \mathbf{a}, \mathbf{b}\rangle = \mathbf{a}^\intercal\mathbf{b}$, is (in this context) a discretization of the standard L^2 inner product $\int_{\mathcal{V}} ab\, d\mu$ which aggregates information from across the vertex set $\mathcal{V}$. While this construction is clearly dependent only on the graph structure (i.e., topology), *any* symmetric positive definite (SPD) matrix $\mathbf{A}_0 : \Omega_0 \to \Omega_0$ also defines an inner product on functions $\mathbf{a} \in \Omega_0$ through the equality

$$(\mathbf{a}, \mathbf{b})_0 \coloneqq \langle \mathbf{a}, \mathbf{A}_0\mathbf{b}\rangle = \mathbf{a}^\intercal\mathbf{A}_0\mathbf{b},$$

which gives a different way of measuring the distance between $\mathbf{a}$ and $\mathbf{b}$. The advantage of this construction is that $\mathbf{A}_0$ can be chosen in a way that incorporates geometric information which implicitly regularizes systems obeying a variational principle. This follows from the following

intuitive fact: the Taylor series of a function does not change, regardless of the inner product on its domain. For any differentiable function(al) $E : \Omega_0 \to \mathbb{R}$, using d to denote the exterior derivative, this means that the following equality holds

$$dE(\mathbf{a})\mathbf{b} := \lim_{\varepsilon \to 0} \frac{E(\mathbf{a} + \epsilon \mathbf{b}) - E(\mathbf{a})}{\varepsilon} = \langle \delta E(\mathbf{a}), \mathbf{b} \rangle = (\nabla E(\mathbf{a}), \mathbf{b})_0 \,,$$

where δE denotes the ℓ^2-gradient of E and ∇E denotes its $\mathbf{A}_0$-gradient, i.e., its gradient with respect to the derivative operator induced by the inner product involving $\mathbf{A}_0$. From this, it is clear that $\delta E = \mathbf{A}_0 \nabla E$, so that the $\mathbf{A}_0$-gradient is just an anisotropic rescaling of the ℓ^2 version. The advantage of working with ∇ over δ in the present case of graph networks is that $\mathbf{A}_0$ can be *learned* based on the features of the graph. This means that learnable feature information (i.e., graph attention) can be directly incorporated into the differential operators governing our bracket-based dynamical systems by construction.

The prototypical example of where this technique is useful is seen in the gradient flow of Dirichlet energy. Recall that the Dirichlet energy of a differentiable function $u : \mathbb{R}^n \to \mathbb{R}$ is given by $\mathcal{D}(u) = (1/2) \int |\nabla u|^2 \, d\mu$, where ∇ now denotes the usual ℓ^2-gradient of the function u on $\mathbb{R}^n$. Using integration-by-parts, it is easy to see that $d\mathcal{D}(u)v = -\int v \Delta u$ for any test function v with compact support, implying that the L^2-gradient of $\mathcal{D}$ is $-\Delta$ and $\dot{u} = \Delta u$ is the L^2-gradient flow of Dirichlet energy: the motion which decreases the quantity $\mathcal{D}(u)$ the fastest *as measured by the L^2 norm*. It can be shown that high-frequency modes decay quickly under this flow, while low-frequency information takes much longer to dissipate. On the other hand, we could alternatively run the H^1-gradient flow of $\mathcal{D}$, which is motion of fastest decrease with respect to the H^1 inner product $(u, v) = \int \langle \nabla u, \nabla v \rangle \, d\mu$. This motion is prescribed in terms of the H^1-gradient of $\mathcal{D}$, which by the discussion above with $\mathbf{A}_0 = -\Delta$ is easily seen to be the identity. This means that the H^1-gradient flow is given by $\dot{u} = -u$, which retains the minimizers of the L^2-flow but with quite different intermediate character, since it functions by simultaneously flattening all spatial frequencies. The process of preconditioning a gradient flow by matching derivatives is known as a Sobolev gradient method (c.f. [69]), and these methods often exhibit faster convergence and better numerical behavior than their L^2 counterparts [70].

Returning to the graph setting, our learnable matrices $\mathbf{A}_k$ on k-cliques will lead to inner products $(\cdot, \cdot)_k$ on functions in Ω_k, and this will induce dual derivatives as described in Appendix A.1. However, in this case we will not have $d_0^* = d_0^\intercal$, but instead the expression given by the following result:

Proposition A.3. *The $\mathbf{A}_k$-adjoints d_k^* to the graph derivative operators d_k are given by $d_k^* = \mathbf{A}_k^{-1} d_k^\intercal \mathbf{A}_{k+1}$. Similarly, for any linear operator $\mathbf{B} : \Omega_k \to \Omega_k$, the $\mathbf{A}_k$-adjoint $\mathbf{B}^* = \mathbf{A}_k^{-1} \mathbf{B}^\intercal \mathbf{A}$.*

Proof. Let $\mathbf{q}, \mathbf{p}$ denote vectors of k-clique resp. $(k+1)$-clique features. It follows that

$$(d_k \mathbf{q}, \mathbf{p})_{k+1} = \langle d_k \mathbf{q}, \mathbf{A}_{k+1} \mathbf{p} \rangle = \langle \mathbf{q}, d_k^\intercal \mathbf{A}_{k+1} \mathbf{p} \rangle = \langle \mathbf{q}, \mathbf{A}_k d_k^* \mathbf{p} \rangle = (\mathbf{q}, d_k^* \mathbf{p})_k \,.$$

Therefore, we see that $d_k^\intercal \mathbf{A}_{k+1} = \mathbf{A}_k d_k^*$ and hence $d_k^* = \mathbf{A}_k^{-1} d_k^\intercal \mathbf{A}_{k+1}$. Similarly, if $\mathbf{q}, \mathbf{q}'$ denote vectors of k-clique features, it follows from the ℓ^2-self-adjointness of $\mathbf{A}_k$ that

$$\left(\mathbf{q}, \mathbf{B}\mathbf{q}'\right)_k = \left\langle \mathbf{q}, \mathbf{A}_k \mathbf{B} \mathbf{q}' \right\rangle = \left\langle \mathbf{B}^\intercal \mathbf{A}_k \mathbf{q}, \mathbf{q}' \right\rangle = \left\langle \mathbf{A}_k^{-1} \mathbf{B}^\intercal \mathbf{A}_k \mathbf{q}, \mathbf{A}_k \mathbf{q}' \right\rangle = \left(\mathbf{B}^* \mathbf{q}, \mathbf{q}'\right)_k \,,$$

establishing that $\mathbf{B}^* = \mathbf{A}_k^{-1} \mathbf{B}^\intercal \mathbf{A}_k$. □

Remark A.6. *It is common in graph theory to encounter the case where $a_i > 0$ are nodal weights and $w_{ij} > 0$ are edge weights. These are nothing more than the (diagonal) inner products $\mathbf{A}_0, \mathbf{A}_1$ in disguise, and so Proposition A.3 immediately yields the familiar formula for the induced divergence*

$$(d_0^* \mathbf{p})_i = \frac{1}{a_i} \sum_{j : (i,j) \in \mathcal{E}} w_{ij} \left(\mathbf{p}_{ji} - \mathbf{p}_{ij} \right) .$$

Note that all of these notions extend to the case of block inner products in the obvious way. For example, if $\mathbf{q}, \mathbf{p}$ are node resp. edge features, it follows that $\mathbf{A} = \operatorname{diag}(\mathbf{A}_0, \mathbf{A}_1)$ is an inner product on node-edge feature pairs, and the adjoints of node-edge operators with respect to $\mathbf{A}$ are computed as according to Proposition A.3.

Remark A.7. *For convenience, this work restricts to diagonal matrices $\mathbf{A}_0, \mathbf{A}_1$. However, note that a matrix which is diagonal in "edge space" $\mathcal{G}_2$ is generally full in a nodal representation. This is because an (undirected) edge is uniquely specified by the two nodes which it connects, meaning that a purely local quantity on edges is necessarily nonlocal on nodes.*

A.4 Higher order attention

As mentioned in the body, when $f = \exp$ and $\tilde{a}(\mathbf{q}_i, \mathbf{q}_j) = (1/d)\langle \mathbf{W}_K \mathbf{q}_i, \mathbf{W}_Q \mathbf{q}_j \rangle$, defining the learnable inner products $\mathbf{A}_0 = (a_{0,ii})$, $\mathbf{A}_1 = (a_{1,ij})$ as

$$a_{0,ii} = \sum_{j \in \mathcal{N}(i)} f\left(\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)\right), \qquad a_{1,ij} = f\left(\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)\right),$$

recovers scaled dot product attention as $\mathbf{A}_0^{-1}\mathbf{A}_1$.

Remark A.8. *Technically, $\mathbf{A}_1$ is an inner product only with respect to a predefined ordering of the edges $\alpha = (i, j)$, since we do not require $\mathbf{A}_1$ be orientation-invariant. On the other hand, it is both unnecessary and distracting to enforce symmetry on $\mathbf{A}_1$ in this context, since any necessary symmetrization will be handled automatically by the differential operator d_0^*.*

Similarly, other common attention mechanisms are produced by modifying the pre-attention function $\tilde{a}$. While $\mathbf{A}_0^{-1}\mathbf{A}_1$ never appears in the brackets of Section 4, letting $\alpha = (i, j)$ denote a global edge with endpoints i, j, it is straightforward to calculate the divergence of an antisymmetric edge feature $\mathbf{p}$ at node i,

$$\begin{aligned}(d_0^*\mathbf{p})_i &= \left(\mathbf{A}_0^{-1} d_0^\intercal \mathbf{A}_1 \mathbf{p}\right)_i = a_{0,ii}^{-1} \sum_\alpha \left(d_0^\intercal\right)_{i\alpha} \left(\mathbf{A}_1\mathbf{p}\right)_\alpha \\ &= a_{0,ii}^{-1} \sum_{\alpha \ni i} \left(\mathbf{A}_1\mathbf{p}\right)_{-\alpha} - \left(\mathbf{A}_1\mathbf{p}\right)_\alpha = -\sum_{j \in \mathcal{N}(i)} \frac{a_{1,ji} + a_{i,ij}}{a_{0,ii}} \mathbf{p}_{ij}.\end{aligned}$$

This shows that $b(\mathbf{q}_i, \mathbf{q}_j) = \left(a_{1,ij} + a_{1,ji}\right)/a_{0,ii}$ appears under the divergence in $d_0^* = \mathbf{A}_0^{-1} d_0^\intercal \mathbf{A}_1$, which is the usual graph attention up to a symmetrization in $\mathbf{A}_1$.

Remark A.9. *While $\mathbf{A}_1$ is diagonal on global edges $\alpha = (i, j)$, it appears sparse nondiagonal in its nodal representation. Similarly, any diagonal extension $\mathbf{A}_2$ to 2-cliques will appear as a sparse 3-tensor $\mathbf{A}_2 = (a_{2,ijk})$ when specified by its nodes.*

This inspires a straightforward extension of graph attention to higher-order cliques. In particular, denote by $K > 0$ the highest degree of clique under consideration, and define $\mathbf{A}_{K-1} = (a_{K-1,i_1 i_2 \ldots i_K})$ by

$$a_{K-1,i_1 i_2 \ldots i_K} = f\left(\mathbf{W}\left(\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \ldots, \mathbf{q}_{i_K}\right)\right),$$

where $\mathbf{W} \in \mathbb{R}^{\otimes_K n_V}$ is a learnable K-tensor. Then, for any $0 \leq k \leq K-2$ define $\mathbf{A}_k = \left(a_{k,i_1 i_2 \ldots i_{k+1}}\right)$ by

$$a_{k,i_1 i_2 \ldots i_{k+1}} = \sum_{i_K, \ldots, i_{K-k-1}} a_{K-1,i_1 i_2 \ldots i_K}.$$

This recovers the matrices $\mathbf{A}_0, \mathbf{A}_1$ from before when $K = 2$, and otherwise extends the same core idea to higher-order cliques. It's attractive that the attention mechanism captured by d_k^* remains asymmetric, meaning that the attention of any one node to the others in a k-clique need not equal the attention of the others to that particular node.

Remark A.10. *A more obvious but less expressive option for higher-order attention is to let*

$$a_{k,i_1 i_2 \ldots i_{k+1}} = \frac{a_{K-1,i_1 i_2 \ldots i_K}}{\sum_{i_K, \ldots, i_{K-k-1}} a_{K-1,i_1 i_2 \ldots i_K}},$$

for any $0 \leq k \leq K-2$. However, application of the combinatorial codifferential $d_{k-1}^\intercal$ appearing in d_{k-1}^* will necessarily symmetrize this quantity, so that the asymmetry behind the attention mechanism is lost in this formulation.

To illustrate how this works more concretely, consider the extension $K = 3$ to 2-cliques, and let $\mathcal{N}(i, j) = \mathcal{N}(i) \cap \mathcal{N}(j)$. We have the tensors $\mathbf{A}_2 = (a_{2,ijk})$, $\mathbf{A}_1 = (a_{1,ij})$, and $\mathbf{A}_0 = (a_{0,i})$ defined by

$$a_{2,ijk} = f\left(\mathbf{W}\left(\mathbf{q}_i, \mathbf{q}_j, \mathbf{q}_k\right)\right), \quad a_{1,ij} = \sum_{k \in \mathcal{N}(i,j)} a_{2,ijk}, \quad a_{0,i} = \sum_{j \in \mathcal{N}(i)} \sum_{k \in \mathcal{N}(i,j)} a_{2,ijk}.$$

This provides a way for (features on) 3-node subgraphs of $\mathcal{G}$ to attend to each other, and can be similarly built-in to the differential operator $d_1^* = \mathbf{A}_1^{-1} d_0^\intercal \mathbf{A}_2$.

A.5 Exterior calculus interpretation of GATs

Let $\mathcal{N}(i)$ denote the one-hop neighborhood of node i, and let $\overline{\mathcal{N}}(i) = \mathcal{N}(i) \cup \{i\}$. Recall the standard (single-headed) graph attention network (GAT) described in [12], described layer-wise as

$$\mathbf{q}_i^{k+1} = \boldsymbol{\sigma}\left(\sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i^k, \mathbf{q}_j^k\right) \mathbf{W}^k \mathbf{q}_j^k\right), \tag{1}$$

where $\boldsymbol{\sigma}$ is an element-wise nonlinearity, $\mathbf{W}^k$ is a layer-dependent embedding matrix, and $a\left(\mathbf{q}_i, \mathbf{q}_j\right)$ denotes the attention node i pays to node j. Traditionally, the attention mechanism is computed through

$$a\left(\mathbf{q}_i, \mathbf{q}_j\right) = \mathrm{Softmax}_j\, \tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{e^{\tilde{a}(\mathbf{q}_i, \mathbf{q}_j)}}{\sigma_i},$$

where the pre-attention coefficients $\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)$ and nodal weights σ_i are defined as

$$\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \mathrm{LeakyReLU}\left(\mathbf{a}^\intercal\left(\mathbf{W}^\intercal \mathbf{q}_i \,||\, \mathbf{W}^\intercal \mathbf{q}_j\right)\right), \qquad \sigma_i = \sum_{j \in \overline{\mathcal{N}}(i)} e^{\tilde{a}(\mathbf{q}_i, \mathbf{q}_j)}.$$

However, the exponentials in the outer $\mathrm{Softmax}$ are often replaced with other nonlinear functions, e.g. $\mathrm{Squareplus}$, and the pre-attention coefficients $\tilde{a}$ appear as variable (but learnable) functions of the nodal features. First, notice that (1) the attention coefficients $a\left(\mathbf{q}_i, \mathbf{q}_j\right)$ depend on the node features $\mathbf{q}$ and not simply the topology of the graph, and (2) the attention coefficients are not symmetric, reflecting the fact that the attention paid by node i to node j need not equal the attention paid by node j to node i. A direct consequence of this is that GATs are not purely diffusive under any circumstances, since it was shown in Appendix A.1 that the combinatorial divergence $d_0^\intercal$ will antisymmetrize the edge features it acts on. In particular, it is clear that the product $a\left(\mathbf{q}_i, \mathbf{q}_j\right)\left(\mathbf{q}_i - \mathbf{q}_j\right)$ is asymmetric in i, j under the standard attention mechanism, since even the pre-attention coefficients $\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)$ are not symmetric, meaning that there will be two distinct terms after application of the divergence. More precisely, there is the following subtle result.

Proposition A.4. *Let $\mathbf{q} \in \mathbb{R}^{|\mathcal{V}| \times n_V}$ denote an array of nodal features. The expression*

$$\sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i, \mathbf{q}_j\right)\left(\mathbf{q}_i - \mathbf{q}_j\right),$$

where $a = \mathbf{A}_0^{-1}\mathbf{A}_1$ is not the action of a Laplace operator whenever $\mathbf{A}_1$ is not symmetric.

Proof. From Appendix A.3, we know that any Laplace operator on nodes is expressible as $d_0^* d_0 = \mathbf{A}_0^{-1} d_0^\intercal \mathbf{A}_1 d_0$ for some positive definite $\mathbf{A}_0, \mathbf{A}_1$. So, we compute the action of the Laplacian at node i,

$$\begin{aligned}
\left(\Delta_0 \mathbf{q}\right)_i = \left(d_0^* d_0 \mathbf{q}\right)_i &= \left(\mathbf{A}_0^{-1} d_0^\intercal \mathbf{A}_1 d_0 \mathbf{q}\right)_i = a_{0,ii}^{-1} \sum_\alpha \left(d_0^\intercal\right)_{i\alpha} \left(\mathbf{A}_1 d_0 \mathbf{q}\right)_\alpha \\
&= a_{0,ii}^{-1} \sum_{\alpha \ni i} \left(\mathbf{A}_1 d_0 \mathbf{q}\right)_{-\alpha} - \left(\mathbf{A}_1 d_0 \mathbf{q}\right)_\alpha = -\frac{1}{2} \sum_{j \in \mathcal{N}(i)} \frac{a_{1,ji} + a_{i,ij}}{a_{0,ii}} \left(\mathbf{q}_j - \mathbf{q}_i\right), \\
&= \sum_{j \in \mathcal{N}(i)} a\left(\mathbf{q}_i, \mathbf{q}_j\right)\left(\mathbf{q}_j - \mathbf{q}_i\right),
\end{aligned}$$

which shows that $a\left(\mathbf{q}_i, \mathbf{q}_j\right) = (1/2)\left(a_{1,ji} + a_{1,ij}\right)/a_{0,ii}$ must have symmetric numerator. □

While this result shows that GATs (and their derivatives, e.g., GRAND) are not purely diffusive, it also shows that it is possible to get close to GAT (at least syntactically) with a learnable diffusion mechanism. In fact, setting $\boldsymbol{\sigma} = \mathbf{W}^k = \mathbf{I}$ in (1) yields precisely a single-step diffusion equation provided that $a\left(q_i^k, q_j^k\right)$ is right-stochastic (i.e., $\sum_j a\left(\mathbf{q}_i, \mathbf{q}_j\right) 1_j = 1_i$) and built as dictated by Proposition A.4.

Theorem A.11. *The GAT layer (1) is a single-step diffusion equation provided that $\boldsymbol{\sigma} = \mathbf{W}^k = \mathbf{I}$, and the attention mechanism $a\left(\mathbf{q}_i, \mathbf{q}_j\right) = (1/2)\left(a_{1,ji} + a_{1,ij}\right)/a_{0,ii}$ is right-stochastic.*

Proof. First, notice that the Laplacian with respect to an edge set which contains self-loops is computable via

$$(\Delta_0 \mathbf{q})_i = -\sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i, \mathbf{q}_j\right)\left(\mathbf{q}_j - \mathbf{q}_i\right) = \mathbf{q}_i - \sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i, \mathbf{q}_j\right)\mathbf{q}_j.$$

Therefore, taking a single step of heat flow $\dot{\mathbf{q}} = -\Delta_0 \mathbf{q}$ with forward Euler discretization and time step $\tau = 1$ is equivalent to

$$\mathbf{q}_i^{k+1} = \mathbf{q}_i^k - \tau\left(\Delta_0 \mathbf{q}^k\right)_i = \sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i^k, \mathbf{q}_j^k\right)\mathbf{q}_j^k,$$

which is just a modified and non-activated GAT layer with $\mathbf{W}^k = \mathbf{I}$ and attention mechanism a. □

Remark A.12. *Since* Softmax *and its variants are right-stochastic, Theorem A.11 is what establishes equivalence between the non-divergence equation*

$$\dot{\mathbf{q}}_i = \sum_{j \in \overline{\mathcal{N}}(i)} a\left(\mathbf{q}_i, \mathbf{q}_j\right)\left(\mathbf{q}_j - \mathbf{q}_i\right),$$

and the standard GAT layer seen in, e.g., [49], when $a(\mathbf{q}_i, \mathbf{q}_j)$ is the usual attention mechanism.

Remark A.13. *In the literature, there is an important (and often overlooked) distinction between the positive graph/Hodge Laplacian Δ_0 and the negative "geometer's Laplacian" Δ which is worth noting here. Particularly, we have from integration-by-parts that the gradient $\nabla = d_0$ is L^2-adjoint to minus the divergence $-\nabla\cdot = d_0^\intercal$, so that the two Laplace operators $\Delta_0 = d_0^\intercal d_0$ and $\Delta = \nabla \cdot \nabla$ differ by a sign. This is why the same ℓ^2-gradient flow of Dirichlet energy can be equivalently expressed as $\dot{\mathbf{q}} = \Delta\mathbf{q} = -\Delta_0\mathbf{q}$, but not by, e.g., $\dot{\mathbf{q}} = \Delta_0\mathbf{q}$.*

This shows that, while they are not equivalent, there is a close relationship between attention and diffusion mechanisms on graphs. The closest analogue to the standard attention expressible in this format is perhaps the choice $a_{1,ij} = f\left(\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)\right)$, $a_{0,ii} = \sum_{j \in \overline{\mathcal{N}}(i)} a_{1,ij}$, discussed in Section 4 and Appendix A.4, where f is any scalar-valued positive function. For example, when $f(x) = e^x$, it follows that

$$\begin{aligned}(\Delta_0 \mathbf{q})_i &= -\frac{1}{2}\sum_{j \in \mathcal{N}(i)} \frac{e^{\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right)} + e^{\tilde{a}\left(\mathbf{q}_j, \mathbf{q}_i\right)}}{\sigma_i}\left(\mathbf{q}_j - \mathbf{q}_i\right) \\ &= -\frac{1}{2}\sum_{j \in \mathcal{N}(i)} \left(a\left(\mathbf{q}_i, \mathbf{q}_j\right) + \frac{e^{\tilde{a}\left(\mathbf{q}_j, \mathbf{q}_i\right)}}{\sigma_i}\right)\left(\mathbf{q}_j - \mathbf{q}_i\right),\end{aligned}$$

which leads to the standard GAT propagation mechanism plus an extra term arising from the fact that the attention a is not symmetric.

Remark A.14. *Practically, GATs and their variants typically make use of multi-head attention, defined in terms of an attention mechanism which is averaged over some number $|h|$ of independent "heads",*

$$a\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{1}{|h|}\sum_h a^h\left(\mathbf{q}_i, \mathbf{q}_j\right),$$

which are distinct only in their learnable parameters. While the results of this section were presented in terms of $|h| = 1$, the reader can check that multiple attention heads can be used in this framework provided it is the pre-attention $\tilde{a}$ that is averaged instead.

A.6 Bracket derivations and properties

Here the architectures in the body are derived in greater detail. First, it will be shown that $\mathbf{L}^* = -\mathbf{L}$, $\mathbf{G}^* = \mathbf{G}$, and $\mathbf{M}^* = \mathbf{M}$, as required for structure-preservation.

Proposition A.5. *For $\mathbf{L}, \mathbf{G}, \mathbf{M}$ defined in Section 4, we have $\mathbf{L}^* = -\mathbf{L}$, $\mathbf{G}^* = \mathbf{G}$, and $\mathbf{M}^* = \mathbf{M}$.*

Proof. First, denoting $\mathbf{A} = \operatorname{diag}(\mathbf{A}_0, \mathbf{A}_1)$, it was shown in section A.3 that $\mathbf{B}^* = \mathbf{A}^{-1}\mathbf{B}^\intercal\mathbf{A}$ for any linear operator $\mathbf{B}$ of appropriate dimensions. So, applying this to $\mathbf{L}$, it follows that

$$\begin{aligned}\mathbf{L}^* &= \begin{pmatrix}\mathbf{A}_0^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{0} & -d_0^*\\ d_0 & \mathbf{0}\end{pmatrix}^\intercal\begin{pmatrix}\mathbf{A}_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{0} & \mathbf{A}_0^{-1}d_0^\intercal\mathbf{A}_1\\ -\mathbf{A}_1^{-1}(d_0^*)^\intercal\mathbf{A}_0 & \mathbf{0}\end{pmatrix} = \begin{pmatrix}\mathbf{0} & d_0^*\\ -d_0 & \mathbf{0}\end{pmatrix} = -\mathbf{L}.\end{aligned}$$

Similarly, it follows that

$$\begin{aligned}\mathbf{G}^* &= \begin{pmatrix}\mathbf{A}_0^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1^{-1}\end{pmatrix}\begin{pmatrix}d_0^*d_0 & \mathbf{0}\\ \mathbf{0} & d_1^*d_1\end{pmatrix}^\intercal\begin{pmatrix}\mathbf{A}_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{A}_0^{-1}d_0^\intercal(d_0^*)^\intercal\mathbf{A}_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1^{-1}d_1^\intercal(d_1^*)^\intercal\mathbf{A}_1\end{pmatrix} = \begin{pmatrix}d_0^*d_0 & \mathbf{0}\\ \mathbf{0} & d_1^*d_1\end{pmatrix} = \mathbf{G},\end{aligned}$$

$$\begin{aligned}\mathbf{M}^* &= \begin{pmatrix}\mathbf{A}_0^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1d_1^*d_1\mathbf{A}_1\end{pmatrix}^\intercal\begin{pmatrix}\mathbf{A}_0 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1\end{pmatrix}\\ &= \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & d_1^\intercal(d_1^*)^\intercal\mathbf{A}_1^2\end{pmatrix} = \begin{pmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1d_1^*d_1\mathbf{A}_1\end{pmatrix} = \mathbf{M},\end{aligned}$$

where the second-to-last equality used that $\mathbf{A}_1\mathbf{A}_1^{-1} = \mathbf{I}$. □

Remark A.15. *Note that the choice of zero blocks in $\mathbf{L}, \mathbf{G}$ is sufficient but not necessary for these adjointness relationships to hold. For example, one could alternatively choose the diagonal blocks of $\mathbf{L}$ to contain terms like $\mathbf{B} - \mathbf{B}^*$ for an appropriate message-passing network $\mathbf{B}$.*

Next, we compute the gradients of energy and entropy with respect to $(\cdot, \cdot)$.

Proposition A.6. *The $\mathbf{A}$-gradient of the energy*

$$E(\mathbf{q}, \mathbf{p}) = \frac{1}{2}\left(|\mathbf{q}|^2 + |\mathbf{p}|^2\right) = \frac{1}{2}\sum_{i\in\mathcal{V}}|\mathbf{q}_i|^2 + \frac{1}{2}\sum_{\alpha\in\mathcal{E}}|\mathbf{p}_\alpha|^2,$$

satisfies

$$\nabla E(\mathbf{q}, \mathbf{p}) = \begin{pmatrix}\mathbf{A}_0^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_1^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{q}\\ \mathbf{p}\end{pmatrix} = \begin{pmatrix}\mathbf{A}_0^{-1}\mathbf{q}\\ \mathbf{A}_1^{-1}\mathbf{p}\end{pmatrix}.$$

Moreover, given the energy and entropy defined as

$$\begin{aligned}E(\mathbf{q}, \mathbf{p}) &= f_E\left(s(\mathbf{q})\right) + g_E\left(s\left(d_0d_0^\intercal\mathbf{p}\right)\right),\\ S(\mathbf{q}, \mathbf{p}) &= g_S\left(s\left(d_1^\intercal d_1\mathbf{p}\right)\right),\end{aligned}$$

where $f_E : \mathbb{R}^{n_V} \to \mathbb{R}$ acts on node features, $g_E, g_S : \mathbb{R}^{n_E} \to \mathbb{R}$ act on edge features, and s denotes sum aggregation over nodes or edges, the $\mathbf{A}$-gradients are

$$\nabla E(\mathbf{q}, \mathbf{p}) = \begin{pmatrix}\mathbf{A}_0^{-1}\mathbf{1}\otimes\nabla f_E\left(s(\mathbf{q})\right)\\ \mathbf{A}_1^{-1}d_0d_0^\intercal\mathbf{1}\otimes\nabla g_E\left(s\left(d_0d_0^\intercal\mathbf{p}\right)\right)\end{pmatrix}, \quad \nabla S(\mathbf{q}, \mathbf{p}) = \begin{pmatrix}\mathbf{0}\\ \mathbf{A}_1^{-1}d_1^\intercal d_1\mathbf{1}\otimes\nabla g_S\left(s\left(d_1^\intercal d_1\mathbf{p}\right)\right)\end{pmatrix},$$

Proof. Since the theory of $\mathbf{A}$-gradients in section A.3 establishes that $\nabla E = \mathbf{A}^{-1}\delta E$, it is only necessary to compute the L^2-gradients. First, letting $\mathbf{x} = \begin{pmatrix}\mathbf{q} & \mathbf{p}\end{pmatrix}^\intercal$, it follows for the first definition of energy that

$$dE(\mathbf{x}) = \sum_{i\in\mathcal{V}}\langle\mathbf{q}_i, d\mathbf{q}_i\rangle + \sum_{\alpha\in\mathcal{E}}\langle\mathbf{p}_\alpha, d\mathbf{p}_\alpha\rangle = \langle\mathbf{q}, d\mathbf{q}\rangle + \langle\mathbf{p}, d\mathbf{p}\rangle = \langle\mathbf{x}, d\mathbf{x}\rangle,$$

showing that $\delta E(\mathbf{q}, \mathbf{p}) = \begin{pmatrix}\mathbf{q} & \mathbf{p}\end{pmatrix}^\intercal$, as desired. Moving to the metriplectic definitions, since each term of E, S has the same functional form, it suffices to compute the gradient of $f\left(s\left(\mathbf{B}\mathbf{q}\right)\right)$ for some function $f : \mathbb{R}^{n_f} \to \mathbb{R}$ and matrix $\mathbf{B} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^{|\mathcal{V}|}$. To that end, adopting the Einstein summation convention where repeated indices appearing up-and-down in an expression are implicitly summed, if $1 \le a, b \le n_f$ and $1 \le i, j \le |\mathcal{V}|$, we have

$$d\left(s(\mathbf{q})\right) = \sum_{i\in|\mathcal{V}|}d\mathbf{q}_i = \sum_{i\in|\mathcal{V}|}\delta_i^j d\mathbf{q}_j = \mathbf{1}^j dq_j^a\mathbf{e}_a = (\mathbf{1}\otimes\mathbf{I}) : d\mathbf{q} = \nabla\left(s(\mathbf{q})\right) : d\mathbf{q},$$

implying that $\nabla s(\mathbf{q}) = \mathbf{1} \otimes \mathbf{I}$. Continuing, it follows that

$$\begin{aligned} d\left(f \circ s \circ \mathbf{B}\mathbf{q}\right) &= f'\left(s\left(\mathbf{B}\mathbf{q}\right)\right)_a s'\left(\mathbf{B}\mathbf{q}\right)_i^a B^{ij} dq_j^a = f'\left(s\left(\mathbf{B}\mathbf{q}\right)\right)_a \mathbf{e}_a \left(B^\intercal\right)^{ij} 1_j \, dq_i^a \\ &= \left\langle \nabla f\left(s\left(\mathbf{B}\mathbf{q}\right)\right) \otimes \mathbf{B}^\intercal \mathbf{1}, d\mathbf{q} \right\rangle = \left\langle \nabla\left(f \circ s \circ \mathbf{B}\mathbf{q}\right), d\mathbf{q} \right\rangle, \end{aligned}$$

showing that $\nabla\left(f \circ s \circ \mathbf{B}\right)$ decomposes into an outer product across modalities. Applying this formula to the each term of E, S then yields the L^2-gradients,

$$\delta E(\mathbf{q}, \mathbf{p}) = \begin{pmatrix} \mathbf{1} \otimes \nabla f_E\left(s(\mathbf{q})\right) \\ d_0 d_0^\intercal \mathbf{1} \otimes \nabla g_E\left(s\left(d_0 d_0^\intercal \mathbf{p}\right)\right) \end{pmatrix}, \quad \delta S(\mathbf{q}, \mathbf{p}) = \begin{pmatrix} \mathbf{0} \\ d_1^\intercal d_1 \mathbf{1} \otimes \nabla g_S\left(s\left(d_1^\intercal d_1 \mathbf{p}\right)\right) \end{pmatrix},$$

from which the desired $\mathbf{A}$-gradients follow directly. □

Finally, we can show that the degeneracy conditions for metriplectic structure are satisfied by the network in Section 4.

Theorem A.16. *The degeneracy conditions $\mathbf{L}\nabla S = \mathbf{M}\nabla E = \mathbf{0}$ are satisfied by the metriplectic bracket in Section A.6.*

Proof. This is a direct calculation using Theorem 3.1 and Proposition A.6. In particular, it follows that

$$\begin{aligned} \mathbf{L}\nabla S &= \begin{pmatrix} \mathbf{0} & -d_0^* \\ d_0 & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_1^{-1} d_1^\intercal d_1 \mathbf{1} \otimes \nabla g_S\left(s\left(d_1^\intercal d_1 \mathbf{p}\right)\right) \end{pmatrix} \\ &= \begin{pmatrix} -\mathbf{A}_0^{-1}\left(d_1 d_0\right)^\intercal d_1 \mathbf{1} \otimes \nabla g_S\left(s\left(d_1^\intercal d_1 \mathbf{p}\right)\right) \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \\ \mathbf{M}\nabla E &= \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_1 d_1^* d_1 \mathbf{A}_1 \end{pmatrix} \begin{pmatrix} \mathbf{A}_0^{-1} \mathbf{1} \otimes \nabla f_E\left(s(\mathbf{q})\right) \\ \mathbf{A}_1^{-1} d_0 d_0^\intercal \mathbf{1} \otimes \nabla g_E\left(s\left(d_0 d_0^\intercal \mathbf{p}\right)\right) \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_1 d_1^* (d_1 d_0) d_0^\intercal \mathbf{1} \otimes \nabla g_E\left(s\left(d_0 d_0^\intercal \mathbf{p}\right)\right) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \end{aligned}$$

since $d_1 d_0 = 0$ as a consequence of the graph calculus. These calculations establish the validity of the energy conservation and entropy generation properties seen previously in the manuscript body. □

Remark A.17. *Clearly, this is not the only possible metriplectic formulation for GNNs. On the other hand, this choice is in some sense maximally general with respect to the chosen operators $\mathbf{L}, \mathbf{G}$, since only constants are in the kernel of d_0 (hence there is no reason to include a nodal term in S), and only elements in the image of $d_1^\intercal$ (which do not exist in our setting) are guaranteed to be in the kernel of $d_0^\intercal$ for any graph. Therefore, $\mathbf{M}$ is chosen to be essentially $\mathbf{G}$ without the Δ_0 term, whose kernel is graph-dependent and hence difficult to design.*

B Experimental details and more results

This Appendix provides details regarding the experiments in Section 5, as well as any additional information necessary for reproducing them. We implement the proposed algorithms with PYTHON and PYTORCH [71] that supports CUDA. The experiments are conducted on systems that are equipped with NVIDIA RTX A100 and V100 GPUs. For NODEs capabilities, we use the TORCHDIFFEQ library [16].

B.1 Damped double pendulum

The governing equations for the damped double pendulum can be written in terms of four coupled first-order ODEs for the angles that the two pendula make with the vertical axis θ_1, θ_2 and their associated angular momenta ω_1, ω_2 (see [72]),

$$\dot{\theta}_i = \omega_i, \qquad 1 \leq i \leq 2, \tag{2}$$

$$\dot{\omega}_1 = \frac{m_2 l_1 \omega_1^2 \sin\left(2\Delta\theta\right) + 2 m_2 l_2 \omega_2^2 \sin\left(\Delta\theta\right) + 2 g m_2 \cos\theta_2 \sin\Delta\theta + 2 g m_1 \sin\theta_1 + \gamma_1}{-2 l_1\left(m_1 + m_2 \sin^2\Delta\theta\right)}, \tag{3}$$

$$\dot{\omega}_2 = \frac{m_2 l_2 \omega_2^2 \sin\left(2\Delta\theta\right) + 2\left(m_1 + m_2\right) l_1 \omega_1^2 \sin\Delta\theta + 2 g\left(m_1 + m_2\right) \cos\theta_1 \sin\Delta\theta + \gamma_2}{2 l_2\left(m_1 + m_2 \sin^2\Delta\theta\right)}, \tag{4}$$

where m_1, m_2, l_1, l_2 are the masses resp. lengths of the pendula, $\Delta\theta = \theta_1 - \theta_2$ is the (signed) difference in vertical angle, g is the acceleration due to gravity, and

$$\gamma_1 = 2k_1\dot{\theta}_1 - 2k_2\dot{\theta}_2 \cos\Delta\theta.$$
$$\gamma_2 = 2k_1\dot{\theta}_1 \cos\Delta\theta - \frac{2\,(m_1+m_2)}{m_2}k_2\dot{\theta}_2,$$

for damping constants k_1, k_2.

Dataset. A trajectory of the damped double pendulum by solving an initial value problem associated with the ODE 2. The initial condition used is (1.0, π/2, 0.0, 0.0), and the parameters are $m_1 = m_2 = 1, g = 1, l_1 = 1, l_2 = 0.9, k_1 = k_2 = 0.1$. For time integrator, we use the TorchDiffeq library [16] with Dormand–Prince 5 (DOPRI5) as the numerical solver. The total simulation time is 50 (long enough for significant dissipation to occur), and solution snapshots are collected at 500 evenly-spaced temporal indices.

To simulate the practical case where only positional data for the system is available, the double pendulum solution is integrated to time $T = 50$ (long enough for significant dissipation to occur) and post-processed once the angles and angular momenta are determined from the equations above, yielding the (x, y)-coordinates of each pendulum mass at intervals of 0.1s. This is accomplished using the relationships

$$x_1 = l_1 \sin\theta_1$$
$$y_1 = -l_1 \cos\theta_1$$
$$x_2 = x_1 + l_2 \sin\theta_2 = l_1 \sin\theta_1 + l_2 \sin\theta_2$$
$$y_2 = y_1 - l_2 \cos\theta_2 = -l_1 \cos\theta_1 - l_2 \cos\theta_2.$$

The double pendulum is then treated as a fully connected three-node graph with positional coordinates $\mathbf{q}_i = (x_i, y_i)$ as nodal features, and relative velocities $\mathbf{p}_\alpha = (d_0\mathbf{q})_\alpha$ as edge features. Note that the positional coordinates $(x_0, y_0) = (0, 0)$ of the anchor node are held constant during training. To allow for the necessary flexibility of coordinate changes, each architecture from Section 4 makes use of a message-passing feature encoder before time integration, acting on node features and edge features separately, with corresponding decoder returning the original features after time integration.

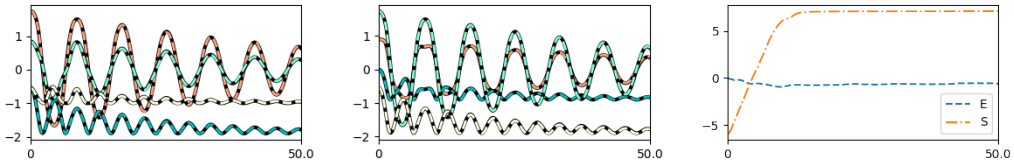
To elicit a fair comparison, both the NODE and NODE+AE architectures are chosen to contain comparable numbers of parameters to the bracket architectures ($\sim$ 30k), and all networks are trained for 100,000 epochs. For each network, the configuration of weights producing the lowest overall error during training is used for prediction.

Hyperparameters. The networks are trained to reconstruct the node/edge features in mean absolute error (MAE) using the Adam optimizer [73]. The NODEs and metriplectic bracket use an initial learning rate of 10^{-4}, while the other models use an initial learning rate of 10^{-3}. The width of the hidden layers in the message passing encoder/decoder is 64, and the number of hidden features for nodes/edges is 32. The time integrator used is simple forward Euler.

Network architectures. The message passing encoders/decoders are 3-layer MLPs mapping, in the node case, nodal features and their graph derivatives, and in the edge case, edge features and their graph coderivatives, to a hidden representation. For the bracket architectures, the attention mechanism used in the learnable coderivatives is scaled dot product. The metriplectic network uses 2-layer MLPs f_E, g_E, g_S with scalar output and hidden width 64. For the basic NODE, node and edge features are concatenated, flattened, and passed through a 4-layer fully connected network of width 128 in each hidden layer, before being reshaped at the end. The NODE+AE architecture uses a 3-layer fully connected network which operates on the concatenated and flattened latent embedding of size $32 * 6 = 192$, with constant width throughout all layers.

B.2 Mujoco

We represent an object as a fully-connected graph, where a node corresponds to a body part of the object and, thus, the nodal feature corresponds to a position of a body part or joint. To learn the dynamics of an object, we again follow the encoder-decoder-type architecture considered in the



(a) $\mathbf{q}$ (b) $\mathbf{p}$ (c) E and S

Figure 5: [Double pendulum] Trajectories of $\mathbf{q}$ and $\mathbf{p}$: ground-truth (solid lines) and predictions of the metriplectic bracket model (dashed lines). The evolution of the energy E and the entropy S over the simulation time. Note that slight fluctuations appear in E due to the fact that forward Euler is not a symplectic integrator.

double-pendulum experiments. First we employ a node-wise linear layer to embed the nodal feature into node-wise hidden representations (i.e., the nodal feature $\mathbf{q}_i$ corresponds to a position of a body part or an angle of a joint.). As an alternative encoding scheme for the nodal feature, in addition to the position or the angle, nodal velocities are considered as additional nodal features, i.e., $\mathbf{q}_i = (q_i, v_i)$. The experimental results of the alternative scheme is represented in the following section B.2.2.

The proposed dynamics models also require edge features (e.g., edge velocity), which are not presented in the dataset. Thus, to extract a hidden representation for an edge, we employ a linear layer, which takes velocities of the source and destination nodes of the edge as an input and outputs edge-wise hidden representations, i.e., the edge feature correspond to a pair of nodal velocities $\mathbf{p}_\alpha = (v_{\text{src}(\alpha)}, v_{\text{dst}(\alpha)})$, where $v_{\text{src}(\alpha)}$ and $v_{\text{dst}(\alpha)}$ denote velocities of the source and destination nodes connected to the edge.

The MuJoCo trajectories are generated in the presence of an actor applying controls. To handle the changes in dynamics due to the control input, we introduce an additive forcing term, parameterized by an MLP, to the dynamics models, which is a similar approach considered in dissipative SymODEN [32]. In dissipative SymODEN, the forcing term is designed to affect only the change of the generalized momenta (also known as the port-Hamiltonian dynamics [74]). As opposed to this approach, our proposed forcing term affects the evolution of both the generalized coordinates that are defined in the latent space. Once the latent states are computed at specified time indices, a node-wise linear decoder is applied to reconstruct the position of body parts of the object. Then the models are trained based on the data matching loss measured in mean-square errors between the reconstructed and the ground-truth positions.

B.2.1 Experiment details

We largely follow the experimental settings considered in [23].

Dataset. As elaborated in [23], the standard Open AI Gym [75] environments preprocess observations in ad-hoc ways, e.g., Hopper clips the velocity observations to $[-10, 10]^d$. Thus, the authors in [23] modified the environments to simply return the position and the velocity (q, v) as the observation and we use the same dataset, which is made publicly available by the authors. The dataset consists of training and test data, which are constructed by randomly splitting the episodes in the replay buffer into training and test data. Training and test data consist of $\sim$40K and $\sim$300 or $\sim$ 85 trajectories, respectively. For both training and test data, we include the first 20 measurements (i.e., 19 transitions) in each trajectory.

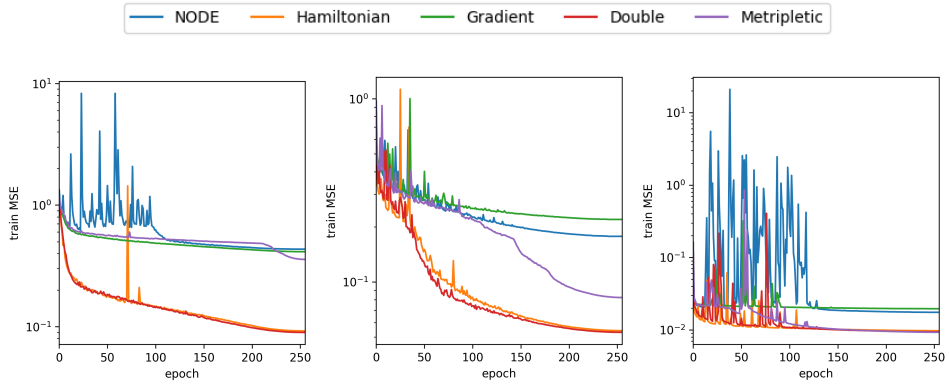
Hyperparameters. For training, we use the Adam optimizer [73] with the initial learning rate 5e-3 and weight decay 1e-4. With the batch size of 200 trajectories, we train the models for 256 epochs. We also employ a cosine annealing learning rate scheduler with the minimum learning rate 1e-6. For time integrator, we use the Torchdiffeq library with the Euler method.

Network architectures. The encoder and decoder networks are parameterized as a linear layer and the dimension of the hidden representations is set to 80. For attention, the scaled dot-product attention is used with 8 heads and the embedding dimension is set to 16. The MLP for handling the

forcing term consists of three fully-connected layers (i.e., input, output layers and one hidden layer with 128 neurons). The MLP used for parameterizing the "black-box" NODEs also consists of three fully-connected layers with 128 neurons in each layer.

B.2.2 Additional results

Figure 6 reports the loss trajectories for all considered dynamics models. For the given number of maximum epochs (i.e., 256), the Hamiltonian and double bracket models tend to reach much lower training losses (an order of magnitude smaller) errors than the NODE and Gradient models do. The metriplectic model produces smaller training losses compared to the NODE and gradient models after a certain number of epochs (e.g., 100 epochs for Hopper).



(a) HalfCheetah (b) Hopper (c) Swimmer

Figure 6: [Mujoco] Train MSE over epoch for all considered dynamics models. For the nodal feature, only the position or the angle of the body part/joint is considered.

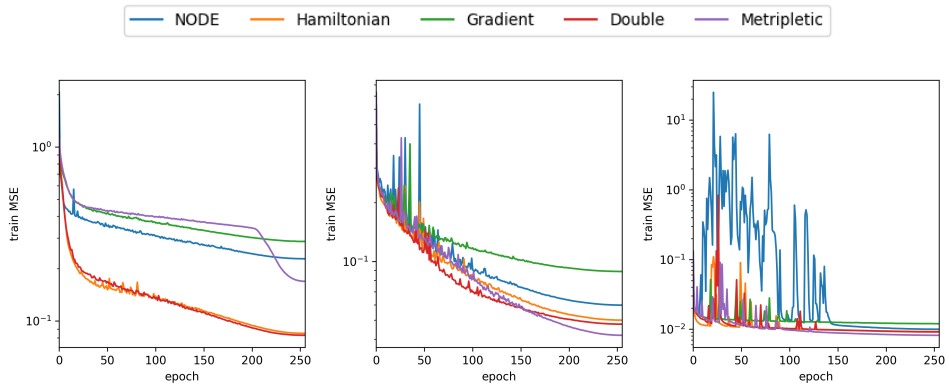
In the next set of experiments, we provide not only positions/angles of body parts as nodal features, but also velocities of the body parts as nodal features (i.e., $\mathbf{q}_i = (q_i, v_i)$). Table 6 reports the relative errors measured in L2-norm; again, the Hamiltonian, double bracket, and metriplectic outperform other dynamics models. In particular, the metriplectic bracket produces the most accurate predictions in the Hopper and Swimmer environments. Figure 7 reports the loss trajectories for all considered models. Similar to the previous experiments with the position as the only nodal feature, the Hamiltonian and Double bracket produces the lower training losses than the NODE and Gradient models do. For the Hopper and Swimmer environments, however, among all considered models, the metriplectic model produces the lowest training MSEs after 256 training epochs.

Dataset	**HalfCheetah**	**Hopper**	**Swimmer**
NODE+AE	0.0848 ± 0.0011	0.0421 ± 0.0041	0.0135 ± 0.00082
Hamiltonian	0.0403 ± 0.0052	0.0294 ± 0.0028	0.0120 ± 0.00022
Gradient	0.0846 ± 0.00358	0.0490 ± 0.0013	0.0158 ± 0.00030
Double Bracket	0.0653 ± 0.010	0.0274 ± 0.00090	0.0120 ± 0.00060
Metriplectic	0.0757 ± 0.0021	0.0269 ± 0.00035	0.0114 ± 0.00067

Table 6: Relative errors of the network predictions of the MuJoCo environments on the test set, reported as avg±stdev over 4 runs.

B.3 Node classification

To facilitate comparison with previous work, we follow the experimental methodology of [61].



(a) HalfCheetah (b) Hopper (c) Swimmer

Figure 7: [Mujoco] Train MSE over epoch for all considered dynamics models. For the nodal feature, along with the position or the angle of the body part/joint, the node velocities are also considered.

B.3.1 Experiment details

Datasets. We consider the three well-known citation networks, Cora [58], Citeseer [59], and Pubmed [60]; the proposed models are tested on the datasets with the original fixed Planetoid traing/test splits, as well as random train/test splits. In addition, we also consider the coauthor graph, CoauthorCS [61] and the Amazon co-purchasing graphs, Computer and Photo [62]. Table 7 provides some basic statistics about each dataset.

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

Table 7: Dataset statistics.

Hyperparameters The bracket architectures employed for this task are identical to those in Section 4 except for that the right-hand side of the Hamiltonian, gradient, and double bracket networks is scaled by a learnable parameter $\mathrm{Sigmoid}\,(\alpha) > 0$, and the matrix $\mathbf{A}_2 = \mathbf{I}$ is used as the inner product on 3-cliques. It is easy to verify that this does not affect the structural properties or conservation character of the networks. Nodal features $\mathbf{q}_i$ are specified by the datasets, and edge features $\mathbf{p}_\alpha = (d_0\mathbf{q})_\alpha$ are taken as the combinatorial gradient of the nodal features. In order to determine good hyperparameter configurations for each bracket, a Bayesian search is conducted using Weights and Biases [76] for each bracket and each dataset using a random 80/10/10 train/valid/test split with random seed 123. The number of runs per bracket was 500 for CORA, CiteSeer, and PubMed, and 250 for CoauthorCS, Computer, and Photo. The hyperparameter configurations leading to the best validation accuracy are used when carrying out the experiments in Table 4 and Table 5.

Specifically, the hyperparameters that are optimized are as follows: initial learning rate (from 0.0005 to 0.05), number of training epochs (from 25 to 150), method of integration (rk4 or dopri5), integration time (from 1 to 5), latent dimension (from 10 to 150 in increments of 10), pre-attention mechanism $\tilde{a}$ (see below), positive function f (either $\exp$ or $\mathrm{Squareplus}$), number of pre-attention heads (from 1 to 15, c.f. Remark A.14), attention embedding dimension (from $1\times$ to $15\times$ the number of heads), weight decay rate (from 0 to 0.05), dropout/input dropout rates (from 0 to 0.8), and the MLP activation function for the metriplectic bracket (either relu, $\tanh$, or $\mathrm{squareplus}$). The pre-attention is chosen

from one of four choices, defined as follows:

$$\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{\left(\mathbf{W}_K \mathbf{q}_i\right)^\intercal \mathbf{W}_Q \mathbf{q}_j}{d} \qquad \text{scaled dot product,}$$

$$\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{\left(\mathbf{W}_K \mathbf{q}_i\right)^\intercal \mathbf{W}_Q \mathbf{q}_j}{\left|\mathbf{W}_K \mathbf{q}_i\right| \left|\mathbf{W}_Q \mathbf{q}_j\right|} \qquad \text{cosine similarity,}$$

$$\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \frac{\left(\mathbf{W}_K \mathbf{q}_i - \overline{\mathbf{W}_K \mathbf{q}_i}\right)^\intercal \left(\mathbf{W}_Q \mathbf{q}_j - \overline{\mathbf{W}_Q \mathbf{q}_j}\right)}{\left|\mathbf{W}_K \mathbf{q}_i - \overline{\mathbf{W}_K \mathbf{q}_i}\right| \left|\mathbf{W}_Q \mathbf{q}_j - \overline{\mathbf{W}_Q \mathbf{q}_j}\right|}, \qquad \text{Pearson correlation}$$

$$\tilde{a}\left(\mathbf{q}_i, \mathbf{q}_j\right) = \left(\sigma_u \sigma_x\right)^2 \exp\left(-\frac{\left|\mathbf{W}_K \mathbf{u}_i - \mathbf{W}_Q \mathbf{u}_j\right|^2}{2\ell_u^2}\right) \exp\left(-\frac{\left|\mathbf{W}_K \mathbf{x}_i - \mathbf{W}_Q \mathbf{x}_j\right|^2}{2\ell_x^2}\right), \qquad \text{exponential kernel}$$

Network architectures. The architectures used for this experiment follow that of GRAND [49], consisting of the learnable affine encoder/decoder networks ϕ, ψ and learnable bracket-based dynamics in the latent space. However, recall that the bracket-based dynamics require edge features, which are manufactured as $\mathbf{p}_\alpha = \left(d_0 \mathbf{q}\right)_\alpha$. In summary, the inference procedure is as follows:

$$\begin{aligned}
\mathbf{q}(0) &= \phi(\mathbf{q}) && \text{(nodal feature encoding)}, \\
\mathbf{p}(0) &= d_0 \mathbf{q}(0) && \text{(edge feature manufacturing)}, \\
\left(\mathbf{q}(T), \mathbf{p}(T)\right) &= \left(\mathbf{q}(0), \mathbf{p}(0)\right) + \int_0^T \left(\dot{\mathbf{q}}, \dot{\mathbf{p}}\right) \mathrm{d}t, && \text{(latent dynamics)} \\
\tilde{\mathbf{q}} &= \psi\left(\mathbf{q}(T)\right), && \text{(nodal feature decoding)} \\
\mathbf{y} &= c(\tilde{\mathbf{q}}). && \text{(class prediction)}
\end{aligned}$$

Training is accomplished using the standard cross entropy

$$H\left(\mathbf{t}, \mathbf{y}\right) = \sum_{i=1}^{|\mathcal{V}|} \mathbf{t}_i^\intercal \log \mathbf{y}_i,$$

where $\mathbf{t}_i$ is the one-hot truth vector corresponding to the i^{th} node. In the case of the metriplectic network, the networks f_E, g_E, g_S are 2-layer MLPs with hidden dimension equal to the latent feature dimension and output dimension 1.

B.3.2 Additional depth study

Here we report the results of a depth study on Cora with the Planetoid train/val/test split. Table 8 shows the train/test accuracy of the different bracket architectures as the number of layers (rk4 timesteps) is increased. It is interesting to note that both the purely reversible Hamiltonian bracket and the purely dissipative Gradient bracket achieve almost perfect training accuracy despite the number of layers, but that the Hamiltonian bracket observes a steep decrease in test accuracy with increasing number of layers while the gradient bracket does not. This could be partially due to the fact that the Hamiltonian bracket structure depends on edge feature information, which is manufactured for these examples and may not correlate with node classification performance. Notice that the double bracket and metriplectic formulations exhibit noticeable drops in both training and testing accuracy as depth increases, and that the double bracket system, which is the most performant at low numbers of layers, becomes too stiff to solve once the number of layers is sufficiently large.

Remark B.1. *It is interesting that the architecture most known for oversmoothing (i.e., gradient) exhibits the most stable classification performance with increasing depth. This is perhaps due to the fact that the gradient system decouples over nodes and edges, while the others do not, meaning that the gradient network does not have the added challenge of learning a useful association between the manufactured edge feature information and the nodal labels.*

Depth study	**1 layer**	**2 layers**	**4 layers**	**8 layers**	**16 layers**	**32 layers**	**64 layers**
Hamiltonian	98.4 ± 1.3	99.6 ± 0.9	99.1 ± 0.6	100 ± 0.0	99.6 ± 0.4	99.9 ± 0.3	100 ± 0.0
Gradient	99.1 ± 1.5	100 ± 0.0	100 ± 0.0	99.4 ± 0.9	99.1 ± 0.9	100 ± 0.0	100 ± 0.0
Double Bracket	97.4 ± 1.9	98.0 ± 2.6	98.1 ± 1.3	97.6 ± 0.8	71.6 ± 1.5	-	-
Metriplectic	98.7 ± 1.8	99.1 ± 0.6	98.1 ± 3.0	62.4 ± 8.7	63.3 ± 7.6	64.1 ± 16.4	57.4 ± 7.5
Hamiltonian	80.1 ± 0.8	74.0 ± 1.8	33.9 ± 2.1	32.8 ± 0.8	28.1 ± 1.1	27.5 ± 0.9	31.3 ± 3.1
Gradient	71.3 ± 2.2	76.2 ± 1.4	77.9 ± 0.9	79.3 ± 1.2	81.4 ± 1.6	80.6 ± 1.0	80.9 ± 0.3
Double Bracket	78.4 ± 1.2	81.2 ± 0.7	82.9 ± 0.3	81.7 ± 0.6	65.0 ± 1.0	-	-
Metriplectic	60.3 ± 2.2	60.4 ± 1.8	58.6 ± 6.7	31.0 ± 4.3	33.2 ± 1.9	34.3 ± 7.8	29.8 ± 3.8

Table 8: Results of a depth study on Cora. Accuracies reported as mean±stdev over 5 randomly initialized runs. Top/bottom groups are training/testing.

CORA networks	**Trainable Parameters**	**Integration Time**
Hamiltonian	60723	1.49625
Gradient	160772	14.82404
Double Bracket	30718	5.36151
Metriplectic	104088	7.53107

Table 9: The integration time and number of trainable parameters corresponding to the best networks trained on CORA. Note that integration time can be considered as a surrogate for depth, since the temporal step-size of each network is fixed to 1.

Depth study (acc)	**1 layer**	**2 layers**	**4 layers**	**8 layers**	**16 layers**	**32 layers**	**64 layers**
Hamiltonian	75.0 ± 1.4	77.9 ± 1.0	62.0 ± 0.6	38.8 ± 1.0	32.0 ± 0.8	25.4 ± 0.5	17.1 ± 1.3
Gradient	69.6 ± 1.1	72.7 ± 1.0	74.5 ± 1.1	77.7 ± 1.0	80.2 ± 1.2	81.4 ± 1.0	82.0 ± 1.2
Double Bracket	77.8 ± 1.1	81.2 ± 0.8	83.9 ± 1.1	83.8 ± 1.0	80.1 ± 1.3	58.9 ± 1.0	19.3 ± 1.4
Metriplectic	61.0 ± 1.6	62.4 ± 1.9	62.0 ± 0.8	61.9 ± 1.3	61.6 ± 1.0	60.6 ± 1.3	61.2 ± 1.0
Hamiltonian	77.2 ± 0.8	77.5 ± 1.2	77.4 ± 1.2	77.0 ± 0.6	78.0 ± 0.7	77.8 ± 1.2	77.4 ± 0.7
Gradient	79.9 ± 1.4	79.9 ± 0.6	79.6 ± 0.6	79.6 ± 1.1	80.4 ± 1.1	80.5 ± 0.9	79.8 ± 1.1
Double Bracket	84.2 ± 0.9	84.5 ± 1.3	84.1 ± 0.5	84.2 ± 1.3	84.1 ± 0.8	83.8 ± 0.9	84.2 ± 1.0
Metriplectic	61.7 ± 1.4	61.0 ± 0.9	61.0 ± 1.0	61.4 ± 1.3	61.6 ± 1.4	61.7 ± 1.2	61.9 ± 1.1

Table 10: Accuracy results of a depth study on CORA. Test accuracies reported as mean±stdev over 10 runs with random train/valid/test splits. Top/bottom groups correspond to tasks 1 and 2 of the study, respectively, where task 1 uses a fixed step-size while task 2 uses a fixed integration domain.

Depth study (time)	**1 layer**	**2 layers**	**4 layers**	**8 layers**	**16 layers**	**32 layers**	**64 layers**
Hamiltonian	0.038 ± 0.003	0.060 ± 0.006	0.106 ± 0.011	0.191 ± 0.035	0.308 ± 0.035	0.497 ± 0.045	0.874 ± 0.035
Gradient	0.053 ± 0.005	0.091 ± 0.011	0.159 ± 0.015	0.273 ± 0.023	0.470 ± 0.031	0.809 ± 0.037	1.44 ± 0.038
Double Bracket	0.068 ± 0.007	0.125 ± 0.010	0.232 ± 0.023	0.434 ± 0.039	0.770 ± 0.068	1.36 ± 0.098	2.52 ± 0.103
Metriplectic	0.161 ± 0.010	0.305 ± 0.011	0.574 ± 0.014	1.10 ± 0.012	2.13 ± 0.040	4.11 ± 0.075	7.86 ± 0.129
Hamiltonian	0.052 ± 0.009	0.067 ± 0.013	0.114 ± 0.021	0.200 ± 0.040	0.350 ± 0.054	0.613 ± 0.056	1.17 ± 0.084
Gradient	0.365 ± 0.020	0.680 ± 0.012	1.26 ± 0.022	2.26 ± 0.058	4.22 ± 0.069	8.38 ± 0.451	19.4 ± 0.152
Double Bracket	0.241 ± 0.036	0.394 ± 0.046	0.731 ± 0.057	1.44 ± 0.132	2.98 ± 0.286	5.42 ± 0.609	9.44 ± 0.487
Metriplectic	1.05 ± 0.051	2.05 ± 0.097	3.87 ± 0.100	7.35 ± 0.149	14.1 ± 0.332	27.2 ± 0.378	53.8 ± 0.370

Table 11: Runtime results of a depth study on CORA. Wall clock times reported as mean±stdev over 10 runs with random train/valid/test splits. Top/bottom groups correspond to tasks 1 and 2 of the study, respectively, where task 1 uses a fixed step-size while task 2 uses a fixed integration domain.