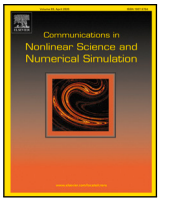
Commun Nonlinear Sci Numer Simulat 140 (2025) 108392

Contents lists available at ScienceDirect

Communications in Nonlinear Science and Numerical Simulation

journal homepage: www.elsevier.com/locate/cnsns



Research paper

A charge-preserving method for solving graph neural diffusion networks

Lidia Aceto [a,*], Pietro Antonio Grassi [a,b]

[a] Università degli Studi del Piemonte Orientale – Dipartimento di Scienze e Innovazione Tecnologica, Viale T. Michel, 11 15121 Alessandria, Italy

[b] INFN and Centro Regge, v. P. Giuria, 1 10125 Torino, Italy

ARTICLE INFO

MSC:
68T07
70H33
37J06
65P99
65L05

Keywords:
Neural networks
Dynamical systems
Symmetries and conservation laws
Numerical methods for ODEs

ABSTRACT

The aim of this paper is to give a systematic mathematical interpretation of the diffusion problem on which Graph Neural Networks (GNNs) models are based. The starting point of our approach is a dissipative functional leading to dynamical equations which allows us to study the symmetries of the model. We provide a short review of graph theory and its relation with network σ-models adapted to our analysis. We discuss the conserved charges and provide a charge-preserving numerical method for solving the dynamical equations. In any dynamical system and also in GRAph Neural Diffusion (GRAND), knowing the charge values and their conservation along the evolution flow could provide a way to understand how GNNs and other networks work with their learning capabilities.

1. Introduction

Graph Neural Networks (GNNs) are a very powerful architecture for neural networks and deep learning algorithms. They are generally based on multiple-layer structures to encode the information, non-linear activation functions and suitable cost functions to compare the forwarded input with the expected results of the training data set. This essential structure has been implemented in different variants with excellent results. Nonetheless, these models have some flaws which fails to exploit the full power of the GNNs (see e.g. [1] for a comprehensive discussion). To overcome some of these problems, a very promising technique which combines ideas from graph neural networks and diffusion models has been explored with outstanding performances. This technique is known as GRAND, acronym for GRAph Neural Diffusion [2,3]. The diffusion coefficient is the *attention matrix* [4], which depends nonlinearly on the features of the nodes, and the evolution of the system (forwarding) is obtained by numerically solving the diffusion equation. The above-mentioned layers are substituted by the steps in the numerical approximation and the number of layers is optimized for maximum efficiency. The implementation of this new technique has been put forward in several papers [1,2,5,6], tackling the problem by discretizing the diffusion equation in time and solving it with different numerical methods such as forward and backward Euler methods and Runge–Kutta methods, and optimizing the methods by some ad hoc prescriptions. Some of these methods are more efficient than others, and a limited number of applications have been developed for some of them. Nevertheless, the simplicity of the implementation makes the approach very relevant for the AI community. In this paper, we provide a more systematic approach both from theoretical point of view (the construction of a functional implementing the symmetries of the model – in the spirit of [3]) and from numerical implementation by providing a well-suited technique. Furthermore, this systematic work has been done to reveal the real knowledge of the network through the conserved charges of the dynamical system.

* Corresponding author.
E-mail addresses: lidia.aceto@uniupo.it (L. Aceto), pietro.grassi@uniupo.it (P.A. Grassi).

https://doi.org/10.1016/j.cnsns.2024.108392
Received 11 January 2024; Received in revised form 10 July 2024; Accepted 9 October 2024
Available online 10 October 2024

In order to improve the efficiency and explainability of the neural networks, we adopt a novel approach based on physical principles by studying the symmetries of the diffusion differential equation and implementing a numerical algorithm to take these symmetries into account in the learning process. The knowledge of these symmetries is definitely of great importance as it allows a deeper understanding of the mechanism underlying the GNNs – which contributes to the *explainability* of neural networks – and the discovery that the network has learned some of the constants of motion corresponding to conserved quantities under the symmetry group.

Given the diffusion equation, there are many symmetries that preserve the form of the equation, and they are widely studied in the mathematical literature. However, among these symmetries there are also some that are unessential for our purposes. To find out the really interesting symmetries (which are respected by the data set), we use the well-known Noether's theorem. It states that for each symmetry of the theory there is a conserved quantity that contains the physical information about the system. The most powerful approach to study the symmetries in this way is to formulate the problem using an action functional, to derive the Ward identities and consequently the conserved quantities. In our case, the specific nature of the differential equation, which is a parabolic differential equation, falls outside the normal techniques of the energy (or action) functional approach (see [7] and the discussion in [8]). In this case, the differential equation cannot be derived from an energy functional, since the energy itself is not a conserved quantity being a dispersive process on the graph. A more appropriate technique will be the *gradient flow* mechanism. However, the latter does not lead to a simple treatment of the symmetries. Finally, we adopt a technique proposed in [9,10] where a new functional is proposed to deal with the gradient flow. The action functional corresponds to a network σ-model (where the σ-model maps are vectors – named feature vectors – from graph vertices to a d-dimensional vector space) coupled to a parametric time-dependent background. For the reader's convenience we provide a self-contained review of graph theory, its relation with network σ-models and, in particular, we recall the definition of discrete gradient and the chain rule used in the functional approach adopted in the paper. The Euler–Lagrangian (EL) equations derived from this functional coincide with the diffusion equation only in a suitable limit of the background, far from that limit the EL equations are of the second order and they can be easily formulated in a Hamiltonian framework by introducing the phase space. The advantage of the phase space formulation is simplified expressions for the conserved charges. In that way, we have a well-suited formulation of the equations with the relevant symmetries manifest, starting point for the numerical analysis.

Before discussing the numerical approach, we point out that the difference between the present discussion and the case of the Schrödinger equation is the existence for the latter of a complex structure for the vector space, that allows a simple action functional formulation and the energy is conserved together with charges correspondent to the symmetries of the model. In our case, we do not assume any complex structure of the vector space and, for that reason we adopt the formulation of [10].

Once that the equations and the conserved charges have been constructed we should wonder whether the conservation of the charges are affected by the numerical scheme used to study the evolution of the system. Unfortunately, this is not automatically implemented by any numerical scheme and therefore we should adopt the most suitable one. It turns out that the conserved charges discussed in the present work appear to be quadratic polynomials on the phase space and they are preserved by the second order numerical scheme known as *implicit midpoint scheme* discussed for example in [11]. That scheme, in contrast to more used schemes such as forward or backward Euler schemes, preserves all quadratic invariants. Notice that our diffusion equation is non linear, nonetheless the charge expression is quadratic polynomial. However, the compatibility between the numerical scheme and the conservation of the charges is possible if the learnable parameters respect those symmetries. Those parameters are learned by the well-known backpropagation algorithm trimmed on a set of reference data. If the data, and consequently the parameter inherit those symmetries they arrange themselves to carry this information on the classification of the results, but that it is possible if the numerical scheme adopted preserves the symmetries.

We would like to mention some related works on the subject. Recently, appear several physical inspired neural network models [12–14] based on Lagrangian and Hamiltonian dynamics. The evolution equation of the system are those obtained by the Hamilton–Jacobi equations or from Euler–Lagrangian equations. Nonetheless, none of those works tackle the problem of diffusivity in their scheme and how the conserved quantities are obtained in that case. On the other side, interesting work appeared in [15–18], where the symmetries are learned from the data on the form of trajectories of a given dynamical system. The approach is different from ours, but it would be very interesting to see if given the trajectories derived from a dissipative equations one can learn all possible symmetries preserved by the system. That will matter of future investigations. Finally, the present work is a companion of a more applicative work in collaboration with computer science colleagues and the main goal would be to see whether the symmetries would be enough to reproduce the classification algorithm achieved in similar works. The network learns the most natural quantities, those which persist in the evolution of the system, like humans [19–21].

The paper is organized as follows. In Section 2 we establish some notations and recall preliminary notions of graph theory useful in the following. In Section 3 we introduce a functional which allow us to write the dynamical system depending on a parameter ϵ which, as $\epsilon \to 0$, reduces to the diffusion equation already used in GRAND. In Section 4 we discuss conserved quantities of the continuous dynamical system. Then, Section 5 is devoted to the discretization of the continuous problem using classical numerical methods, including the implicit midpoint scheme, in the light of preserving the charges along the solution with the numerical methods. Numerical experiments in support of the present analysis are reported in Section 6. Finally, we give some concluding remarks in Section 7. In Appendix, we discuss the model in the continuum. We compare the graph model studied here with the conventional σ-model used in theoretical physics pointing out the non-local and non-linear structures.

2. Background

We briefly review some basic notions of graph theory and collect some notations that will be used in the following discussions.

An undirected (self-avoiding) graph is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, 2, \dots, n\}$ is a set of nodes (or vertices) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges such that if $(i, j) \in \mathcal{E}$, then $(j, i) \in \mathcal{E}$, for all $i \neq j$. The topology of the graph is encoded into the *incidence matrix*

$$w_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise}. \end{cases}$$

Let $\mathbf{x} : \mathcal{V} \times [t_0, T] \longrightarrow \mathbb{R}^d$ denote a function defined on nodes and a positive time interval such that the *feature vector* is given by

$$\mathbf{x}_i(t) = (x_i^1(t), \quad x_i^2(t), \quad \dots, \quad x_i^d(t))^T.$$

We can define differential operators acting on such function. The *graph gradient* is the operator ∇ mapping functions defined on vertices to functions defined on edges:

$$(\nabla \mathbf{x})_{ji} = \mathbf{x}_j(t) - \mathbf{x}_i(t). \tag{1}$$

It is immediate to check that $(\nabla \mathbf{x})_{ij} = -(\nabla \mathbf{x})_{ji}$.

For two column vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, such that $\mathbf{u} = (u^1, u^2, \dots, u^d)^T$, we adopt the notation

$$\mathbf{u}^T \mathbf{v} = \sum_{I=1}^{d} u^I v^I$$

for the Euclidean scalar product; hence, the Euclidean norm can be written in a coordinate-free way as

$$\|\mathbf{u}\| = \sqrt{\mathbf{u}^T \mathbf{u}}. \tag{2}$$

We represent rotations with orthogonal matrices $\Lambda \in SO(d)$ or, equivalently, $\Lambda = e^{\mathcal{R}}$ where $\mathcal{R} = \sum_{a=1}^{d(d-1)/2} c_a \mathcal{R}_a$ is a linear combination of Lie-algebra-$\mathfrak{so}(d)$ skew-symmetric matrices $\mathcal{R}_a$ [22]. Obviously, the Euclidean scalar product is invariant under any rotation.

In addition, we need the definition of partial derivatives with respect to a vector $\mathbf{u}$ which is given by

$$\frac{\partial \mathbf{u}}{\partial \mathbf{v}} = \begin{pmatrix} \frac{\partial u^1}{\partial v^1} & \cdots & \frac{\partial u^d}{\partial v^1} \\ \vdots & \ddots & \vdots \\ \frac{\partial u^1}{\partial v^d} & \cdots & \frac{\partial u^d}{\partial v^d} \end{pmatrix}. \tag{3}$$

Then, for a given scalar c the chain rule can be specified as follows:

$$\frac{\partial c(\mathbf{u})}{\partial \mathbf{v}} = \frac{\partial \mathbf{u}}{\partial \mathbf{v}} \frac{\partial c(\mathbf{u})}{\partial \mathbf{u}}. \tag{4}$$

Finally, given a generic functional in the form

$$F[q] = \int_{t_{\mathrm{i}}}^{t_{\mathrm{f}}} \mathcal{F}(t, q(t), q'(t), q''(t), \dots, q^{(m)}(t), \dots)\, dt,$$

with $q^{(m)}(t)$ the mth derivative of $q(t)$, the functional derivative of $F[q]$ is defined as

$$\frac{\delta F}{\delta q} = \sum_{m \geq 0} \frac{d^m}{dt^m} \frac{\partial F}{\partial q^{(m)}}. \tag{5}$$

Suitable boundary conditions on t_{i} and t_{f} should be assigned.

3. Variational principle

In order to describe the symmetries, the conservation of charges and the energy of the system, it is useful to introduce the following functional

$$L_\epsilon[\mathbf{x}, V] = \int_{t_0}^{T} e^{-\frac{t}{\epsilon}} \left(\frac{1}{2} \sum_{i \in \mathcal{V}} \left\| \frac{\partial \mathbf{x}_i}{\partial t} \right\|^2 + \frac{1}{2\epsilon} \sum_{i,j \in \mathcal{V}} G_{ij} \left\| (\nabla \mathbf{x})_{ij} \right\|^2 + V[\mathbf{x}] \right) dt, \tag{6}$$

letting ϵ be a small enough positive number. Here G_{ij} is the (i, j)th entry of the symmetric matrix G of order n with respect to the edge (i, j) and V denotes the potential that depends on $\mathbf{x}$ in a nonlinear way and does not depend on time and discrete-space derivatives. Also G_{ij} depends on $\mathbf{x}$ in a nonlinear way (see [4] and reference therein). Before continuing, we clarify this fact by considering the form of this matrix in more detail.

In (6) G_{ij} enters in the action multiplied by $\left\| (\nabla \mathbf{x})_{ij} \right\|^2$ which is symmetric under the exchange of i, j (see (1) and (2)). Therefore, only the symmetric part of G_{ij} enters in the action and the skew-symmetric part drops out. This can be easily implemented by restring the dependence of G_{ij} upon symmetric invariants. Introducing two matrices W_K, W_Q (K stands for keys, Q for queries [2,4,5]) which are of size $d' \times d$ with $d' \gg d$, for all i, j we can construct the following expression

$$\mathcal{M}_{ij} = (W_K \mathbf{x}_i)^T (W_Q \mathbf{x}_j). \tag{7}$$

Thus, setting

$$C_{ij} = \mathcal{M}_{ij} + \mathcal{M}_{ji}\,, \tag{8}$$

the entries of the matrix G can be given as

$$G_{ij} = w_{ij}\, g\left(C_{ij}\right); \tag{9}$$

here the factor w_{ij} is the incidence matrix of the graph that selects which node is linked to another and the *activation function* g could be the softmax function defined by

$$\text{softmax}\left(C_{ij}\right) = \frac{e^{C_{ij}}}{\sum\limits_{i,j\in\mathcal{V}} e^{C_{ij}}},$$

see [23] for further details. When C_{ij} is given as in (8) the entries of the matrix G are given in terms of "scaled dot". Other possibilities are obtained by introducing the new expressions

$$\mathcal{K}_i = \|W_K \mathbf{x}_i\|, \qquad \mathcal{Q}_j = \|W_Q \mathbf{x}_j\|$$

and by constructing C_{ij} in terms of these building blocks

- $\frac{\mathcal{M}_{ij}}{\mathcal{K}_i \mathcal{Q}_j}$,
- $\exp\left(-(\mathcal{K}_i^2 + \mathcal{Q}_j^2 - \mathcal{M}_{ij} - \mathcal{M}_{ji})/\varsigma^2\right)$, with ς a suitable scaling.

The softmax function evaluated on these two matrices is indicated in the literature as "cosine-similarity" and "exponential kernel", respectively [6]. Actually, the softmax function can be substituted with any activation function such as ReLU, sigmoids or another useful distribution.

The next step is to derive the equations of motion from the functional (6). It is worth observing that for a single graph point the second term vanishes and only the potential plays an effective role. However, although the latter is also necessary for several applications (see, for example, [8]), to simplify the analysis and without losing generality, we will omit this term from now on and then, we use

$$L_\epsilon[\mathbf{x}] := L_\epsilon[\mathbf{x}, 0] = \int_{t_0}^{T} e^{-\frac{t}{\epsilon}} \left(\frac{1}{2} \sum_{i\in\mathcal{V}} \left\| \frac{\partial \mathbf{x}_i}{\partial t} \right\|^2 + \frac{1}{2\epsilon} \sum_{i,j\in\mathcal{V}} G_{ij} \left\| (\nabla \mathbf{x})_{ij} \right\|^2 \right) dt. \tag{10}$$

Denoting by

$$\mathcal{L}_\epsilon\left(t, \mathbf{x}_i, \frac{\partial \mathbf{x}_i}{\partial t}\right) = e^{-\frac{t}{\epsilon}} \left(\frac{1}{2} \sum_{i\in\mathcal{V}} \left(\frac{\partial \mathbf{x}_i}{\partial t} \right)^T \frac{\partial \mathbf{x}_i}{\partial t} + \frac{1}{2\epsilon} \sum_{i,j\in\mathcal{V}} G_{ij} \left((\nabla \mathbf{x})_{ij} \right)^T (\nabla \mathbf{x})_{ij} \right) \tag{11}$$

the Lagrangian, a local functional which can be viewed as t-dependent-background field theory σ-model, we can rewrite (10) as follows

$$L_\epsilon[\mathbf{x}] = \int_{t_0}^{T} \mathcal{L}_\epsilon\left(t, \mathbf{x}_i, \frac{\partial \mathbf{x}_i}{\partial t}\right) dt.$$

Consequently, taking the functional derivatives of $L_\epsilon[\mathbf{x}]$ with respect to $\mathbf{x}_i$ we obtain (see (5))[1]

$$\begin{aligned} \frac{\delta L_\epsilon}{\delta \mathbf{x}_i} &= \frac{\partial \mathcal{L}_\epsilon}{\partial \mathbf{x}_i} - \frac{d}{dt} \frac{\partial \mathcal{L}_\epsilon}{\partial (\partial \mathbf{x}_i / \partial t)} \\ &= \frac{e^{-\frac{t}{\epsilon}}}{2\epsilon} \frac{\partial}{\partial \mathbf{x}_i} \left(\sum_{i,j\in\mathcal{V}} G_{ij} \left((\nabla \mathbf{x})_{ij} \right)^T (\nabla \mathbf{x})_{ij} \right) - \frac{d}{dt} \left(e^{-\frac{t}{\epsilon}} \frac{\partial \mathbf{x}_i}{\partial t} \right) \\ &= e^{-\frac{t}{\epsilon}} \left[\frac{1}{2\epsilon} \frac{\partial}{\partial \mathbf{x}_i} \left(\sum_{i,j\in\mathcal{V}} G_{ij} \left((\nabla \mathbf{x})_{ij} \right)^T (\nabla \mathbf{x})_{ij} \right) + \frac{1}{\epsilon} \frac{\partial \mathbf{x}_i}{\partial t} - \frac{\partial^2 \mathbf{x}_i}{\partial t^2} \right]. \end{aligned} \tag{12}$$

Using (3) and (4) and the symmetry of the matrix G, by direct calculation we can write

$$\begin{aligned} &\frac{\partial}{\partial \mathbf{x}_i} \left(\sum_{\ell,k\in\mathcal{V}} G_{\ell k} \left((\nabla \mathbf{x})_{\ell k} \right)^T (\nabla \mathbf{x})_{\ell k} \right) \\ &= \sum_{\ell,k\in\mathcal{V}} \left[\frac{\partial G_{\ell k}}{\partial \mathbf{x}_i} \left((\nabla \mathbf{x})_{\ell k} \right)^T (\nabla \mathbf{x})_{\ell k} + G_{\ell k} \frac{\partial}{\partial \mathbf{x}_i} \left(\left((\nabla \mathbf{x})_{\ell k} \right)^T (\nabla \mathbf{x})_{\ell k} \right) \right] \\ &= 2 \sum_{k\in\mathcal{V}} \left\{ \left[\frac{\partial G_k}{\partial \mathbf{x}_i} \mathbf{x}_k^T - \sum_{\ell\in\mathcal{V}} \frac{\partial G_{\ell k}}{\partial \mathbf{x}_i} \mathbf{x}_\ell^T \right] + 2 \left(G_i\, \delta_{ik} - G_{ik} \right) I_d \right\} \mathbf{x}_k, \end{aligned}$$

where we have introduced the weight of the node k summing the columns of $G_{k\ell}$

$$G_k = \sum_{\ell\in\mathcal{V}} G_{k\ell},$$

[1] In deriving the differential equations we have imposed the boundary conditions $\frac{\partial \mathbf{x}_i}{\partial t}\Big|_{t=t_0} = 0$ while at $t = T$ we took in account the limit for $\epsilon \to 0$.

δ_{ik} is the Kronecker delta and I_d is the identity matrix of order d. Therefore, setting

$$A_{ij} = \sum_{\ell \in \mathcal{V}} \frac{\partial G_{\ell j}}{\partial \mathbf{x}_i} \mathbf{x}_\ell^T - \frac{\partial G_j}{\partial \mathbf{x}_i} \mathbf{x}_j^T + 2\left(G_{ij} - G_i\,\delta_{ij}\right) I_d, \tag{13}$$

relation (12) becomes

$$\frac{\delta L_\epsilon}{\delta \mathbf{x}_i} = e^{-\frac{t}{\epsilon}} \left[-\frac{1}{\epsilon} \sum_{j \in \mathcal{V}} A_{ij} \mathbf{x}_j + \frac{1}{\epsilon} \frac{\partial \mathbf{x}_i}{\partial t} - \frac{\partial^2 \mathbf{x}_i}{\partial t^2} \right].$$

Notice that for fixed i, j, A_{ij} is a square matrix of order d. Imposing

$$\frac{\delta L_\epsilon}{\delta \mathbf{x}_i} = 0$$

leads to

$$-\frac{\partial^2 \mathbf{x}_i}{\partial t^2} + \frac{1}{\epsilon} \left(\frac{\partial \mathbf{x}_i}{\partial t} - \sum_{j \in \mathcal{V}} A_{ij} \mathbf{x}_j \right) = 0. \tag{14}$$

As $\epsilon \to 0$, we recover the diffusion equation given in [2] on which GRAND is based, namely

$$\frac{\partial \mathbf{x}_i}{\partial t} - \sum_{j \in \mathcal{V}} A_{ij} \mathbf{x}_j = 0. \tag{15}$$

We emphasize that the main difference between (14) and (15) is the second derivative term reminiscent of dumped harmonic oscillators. This additional term allows us to define the conserved charges as we are going to discuss.

To convert Eq. (14) into a first-order system we use the rescaled momenta $\mathbf{P}_i$ defined as follows

$$\mathbf{p}_i(t) = \frac{\partial \mathcal{L}_\epsilon}{\partial(\partial \mathbf{x}_i / \partial t)} = e^{-\frac{t}{\epsilon}} \frac{\partial \mathbf{x}_i}{\partial t} \equiv e^{-\frac{t}{\epsilon}} \mathbf{P}_i(t)\,. \tag{16}$$

Thus, Eq. (14) becomes the system

$$\begin{cases} \dfrac{\partial \mathbf{x}_i}{\partial t} &= \mathbf{P}_i \\ \dfrac{\partial \mathbf{P}_i}{\partial t} &= \dfrac{1}{\epsilon}\left(\mathbf{P}_i - \displaystyle\sum_{j \in \mathcal{V}} A_{ij} \mathbf{x}_j \right) \end{cases}. \tag{17}$$

So far we have only focused on the dynamic of a particular node. Now we consider the equations that take into account all the nodes of the graph. At this aim we introduce the vector $\mathbf{Y}(t) = (\mathbf{x}_1^T(t), \dots, \mathbf{x}_n^T(t), \mathbf{P}_1^T(t), \dots, \mathbf{P}_n^T(t))^T \in \mathbb{R}^{2\hat{n}}$, where $\hat{n} = nd$. Under the hypothesis that G_{ij} depends upon the scalar products (see (7) and (8))

$$C_{ij} = \mathbf{x}_i^T \left(W_K^T W_Q + W_Q^T W_K \right) \mathbf{x}_j \equiv \mathbf{x}_i^T \mathbb{W} \mathbf{x}_j \tag{18}$$

we can reorganize the term $(-\sum_{j \in \mathcal{V}} A_{ij} \mathbf{x}_j)$ given in (17) and define the matrix $C(\mathbf{Y}(t))$ having the following block-entries

$$\Big(C(\mathbf{Y}(t))\Big)_{rs} = \begin{cases} -2 I_d \displaystyle\sum_{\substack{i \in \mathcal{V} \\ i \neq r}} w_{ri}\, g\left(\mathbf{x}_r^T \mathbb{W} \mathbf{x}_i\right), & \text{if } r = s \\ \\ w_{rs}\Big(2 I_d\, g\left(\mathbf{x}_r^T \mathbb{W} \mathbf{x}_s\right) - \mathbb{W} g'\left(\mathbf{x}_r^T \mathbb{W} \mathbf{x}_s\right) \|\mathbf{x}_r - \mathbf{x}_s\|^2 \Big), & \text{if } r \neq s \end{cases}, \tag{19}$$

for $r, s = 1, 2, \dots, n$; here $g'(z) = \frac{dg}{dz}$. Then, the system can be expressed in matrix form as

$$\frac{\partial \mathbf{Y}(t)}{\partial t} = \mathcal{B}(\mathbf{Y}(t))\, \mathbf{Y}(t), \qquad \mathcal{B}(\mathbf{Y}(t)) = \frac{1}{\epsilon} \begin{pmatrix} O & \epsilon I_{\hat{n}} \\ C(\mathbf{Y}(t)) & I_{\hat{n}} \end{pmatrix}, \quad t \in (t_0, T], \tag{20}$$

where $I_{\hat{n}}$ is the identity matrix of order $\hat{n}$. Notice that $C(\mathbf{Y}(t))$ is a symmetric matrix since its entries are function of $\mathbb{W}$ which is itself a symmetric matrix (see (18)). In the subsequent analysis this property will be crucial.

4. Conserved quantities

For any dynamical system, it is useful to study possible constants of motion, associated to conserved quantities, which may characterize the evolution in a simpler way. The most natural example is the energy of the system which is the conserved quantity associated to the Hamiltonian. Furthermore, there might be other conserved quantities associate to the symmetries of the model, such as rotational symmetry, scale symmetry or translation symmetry. If the Hamiltonian exhibits one of those symmetries, there are such conserved quantities.

In this case the Hamiltonian is (see (11) and (16))

$$H\left(\mathbf{x}_i, \mathbf{P}_i\right) = \sum_{i \in \mathcal{V}} e^{-\frac{t}{\epsilon}} \mathbf{P}_i^T \frac{\partial \mathbf{x}_i}{\partial t} - \mathcal{L}_\epsilon$$

$$= \frac{e^{-\frac{t}{\epsilon}}}{2}\sum_{i\in\mathcal{V}}\left(\frac{\partial \mathbf{x}_i}{\partial t}\right)^T\frac{\partial \mathbf{x}_i}{\partial t} - \frac{e^{-\frac{t}{\epsilon}}}{2\epsilon}\sum_{i,j\in\mathcal{V}} G_{ij}\left(\nabla\mathbf{x}\right)^T_{ij}\left(\nabla\mathbf{x}\right)_{ij}$$
$$= e^{-\frac{t}{\epsilon}}\left(\frac{1}{2}\sum_{i\in\mathcal{V}}\mathbf{P}_i^T\mathbf{P}_i - \frac{1}{2\epsilon}\sum_{i,j\in\mathcal{V}} G_{ij}\left(\nabla\mathbf{x}\right)^T_{ij}\left(\nabla\mathbf{x}\right)_{ij}\right) \tag{21}$$

and, as a consequence of the explicit dependence upon t in (21), we have

$$\frac{dH}{dt} = \frac{e^{-\frac{t}{\epsilon}}}{\epsilon}\sum_{i\in\mathcal{V}}\mathbf{P}_i^T\mathbf{P}_i - \frac{1}{\epsilon}H$$
$$= \frac{e^{-\frac{t}{\epsilon}}}{2\epsilon}\left(\sum_{i\in\mathcal{V}}\mathbf{P}_i^T\mathbf{P}_i + \frac{1}{\epsilon}\sum_{i,j\in\mathcal{V}} G_{ij}\left(\nabla\mathbf{x}\right)^T_{ij}\left(\nabla\mathbf{x}\right)_{ij}\right), \tag{22}$$

where in the first equality we used (17). The Hamiltonian is conserved in the limit $\epsilon \to 0$ (which corresponds also to $T\to\infty$) namely when the dispersive functional $L_\epsilon[\mathbf{x}]$ is independent of t and the energy is a well-defined concept. Therefore, the energy is not a valuable quantity in the present case.

On the other hand, we study other constants of motion that are conserved along the diffusion process and can be used to analyze the network final result. For that, we notice that while the expressions $\mathbf{P}_i^T\mathbf{P}_i$ and $\left(\nabla\mathbf{x}\right)^T_{ij}\left(\nabla\mathbf{x}\right)_{ij}$ in (21) are invariant under any rotation, the invariance of G_{ij} is restricted by the presence of $\mathbb{W}$. Denoting by

$$\Lambda = e^{\mathcal{R}} = I_d + \mathcal{R} + \mathcal{O}(\mathcal{R}^2), \qquad \Lambda\in SO(d),$$

a generic rotation – where $\mathcal{R}$ is a skew-symmetric matrix – those which are preserved commute with $\mathbb{W}$, namely

$$[\mathbb{W},\mathcal{R}] = O\,. \tag{23}$$

In the case where G_{ij} depends upon $\mathcal{M}_{ij}$, $\mathcal{K}_i$, and $\mathcal{Q}_j$, then the preserved rotations are those which commute with $\mathbb{W}, W_K^TW_K$, and $W_Q^TW_Q$. Obviously, that subgroup is smaller than the one given by (23). The matrices W_K and W_Q are the learnable parameters of the theory and are obtained using the backpropagation technique. Therefore, after a few iterations of the machine learning process, they stabilized to some specific form and the symmetry should be compatible with those.

Since the Hamiltonian given in (21) is invariant under the following rigid symmetry

$$\mathbf{x}_i \;\leftarrow\; \Lambda\mathbf{x}_i\,, \qquad \mathbf{P}_i \;\leftarrow\; \Lambda\mathbf{P}_i, \tag{24}$$

we can define the corresponding charge as

$$Q_\Lambda(t) = e^{-\frac{t}{\epsilon}}\sum_{i\in\mathcal{V}}\left(\mathbf{P}_i(t)\right)^T\mathcal{R}\mathbf{x}_i(t)\,. \tag{25}$$

Using $\mathbf{Y}(t)$, we can rewrite this equation as

$$Q_\Lambda(t) = -\frac{e^{-\frac{t}{\epsilon}}}{2}\left(\mathbf{Y}(t)\right)^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y}(t) \tag{26}$$

where $\mathcal{J}$ is the skew-symmetric matrix

$$\mathcal{J} = \begin{pmatrix} O & I_n \\ -I_n & O\end{pmatrix}.$$

By a direct calculation we have that

$$\frac{dQ_\Lambda}{dt} = -\frac{1}{\epsilon}Q_\Lambda - \frac{e^{-\frac{t}{\epsilon}}}{2}\frac{d}{dt}\left(\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y}\right)$$
$$= \frac{e^{-\frac{t}{\epsilon}}}{2\epsilon}\left(\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y}\right) - \frac{e^{-\frac{t}{\epsilon}}}{2}\frac{d}{dt}\left(\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y}\right)$$
$$= \frac{e^{-\frac{t}{\epsilon}}}{2}\left(\frac{1}{\epsilon}\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y} - \frac{d}{dt}\left(\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y}\right)\right)$$
$$= \frac{e^{-\frac{t}{\epsilon}}}{2}\left(\frac{1}{\epsilon}\mathbf{Y}^T(\mathcal{J}\otimes\mathcal{R})\mathbf{Y} - \mathbf{Y}^T\left(\mathcal{B}(\mathbf{Y})^T\mathcal{J}\otimes\mathcal{R} + \mathcal{J}\otimes\mathcal{R}\,\mathcal{B}(\mathbf{Y})\right)\mathbf{Y}\right). \tag{27}$$

Therefore

$$\frac{dQ_\Lambda}{dt} = 0 \quad\Longleftrightarrow\quad \mathbf{Y}^T\left(\frac{1}{\epsilon}(\mathcal{J}\otimes\mathcal{R}) - \mathcal{B}(\mathbf{Y})^T\mathcal{J}\otimes\mathcal{R} + \mathcal{J}\otimes\mathcal{R}\mathcal{B}(\mathbf{Y})\right)\mathbf{Y} = 0\,. \tag{28}$$

When $[\mathbb{W},\mathcal{R}] = O$, which expresses the compatibility of the learnable parameters with the simmetries, from a straightforward computation we can check that

$$\left(\mathcal{B}(\mathbf{Y})^T\mathcal{J}\otimes\mathcal{R} + \mathcal{J}\otimes\mathcal{R}\,\mathcal{B}(\mathbf{Y})\right) = \frac{1}{\epsilon}\mathcal{J}\otimes\mathcal{R}. \tag{29}$$

and this implies that $Q_\Lambda(t)$ actually is a constant. We summarize this fact in the following result.

Theorem 1. *Let $\mathbb{W}$ be the matrix defined in (18) and $\mathcal{R}$ any skew-symmetric matrix of order d. With reference to (20), when*

$$[\mathbb{W}, \mathcal{R}] = O,$$

the charge Q_Λ given in (26) is preserved, that is

$$Q_\Lambda(t) = Q_\Lambda(t_0), \qquad \forall\, t \in (t_0, T].$$

Since we need to solve numerically the dynamical system (20), in the next section we are interested in selecting an appropriate numerical scheme that is able to inherit the conservation of the underlying continuous invariants. In particular, in this analysis we are interested to select a method that preserves the charge.

5. Computing discrete invariants

Instead of discretizing the dynamical system (20), from a numerical point of view it is more convenient to consider the system in the canonical coordinates $(\mathbf{x}_i, \mathbf{p}_i)$, (see (16) and (17)), which in matrix form can be represented as

$$\frac{\partial \mathbf{y}(t)}{\partial t} = \mathcal{E}(\mathbf{y}(t))\,\mathbf{y}(t), \qquad \mathcal{E}(\mathbf{y}(t)) = \begin{pmatrix} O & e^{\frac{t}{\epsilon}} I_{\hat{n}} \\ -\frac{e^{-\frac{t}{\epsilon}}}{\epsilon} C(\mathbf{y}(t)) & O \end{pmatrix}, \quad t \in (t_0, T], \tag{30}$$

with $\mathbf{y}(t) = (\mathbf{x}_1^T(t), \dots, \mathbf{x}_n^T(t), \mathbf{p}_1^T(t), \dots, \mathbf{p}_n^T(t))^T$. In this framework the charge is given by (see (25))

$$Q_\Lambda(t) = \sum_{i \in \mathcal{V}} \left(\mathbf{p}_i(t)\right)^T \mathcal{R}\mathbf{x}_i(t) = -\frac{1}{2}\,(\mathbf{y}(t))^T\,(\mathcal{J} \otimes \mathcal{R})\mathbf{y}(t). \tag{31}$$

Consequently,

$$\frac{dQ_\Lambda}{dt} = 0 \quad \Longleftrightarrow \quad \mathbf{y}^T \left(\mathcal{E}(\mathbf{y})^T \mathcal{J} \otimes \mathcal{R} + \mathcal{J} \otimes \mathcal{R}\, \mathcal{E}(\mathbf{y}) \right) \mathbf{y} = 0.$$

Under the hypothesis that $[\mathbb{W}, \mathcal{R}] = O$, from a direct computation we can check that

$$\left(\mathcal{E}(\mathbf{y})^T \mathcal{J} \otimes \mathcal{R} + \mathcal{J} \otimes \mathcal{R}\, \mathcal{E}(\mathbf{y}) \right) = O \tag{32}$$

and then the charge is trivially preserved.

Denoting by $\{\mathbf{y}^{(k)}\}_{k=0}^N$ the numerical solution provided by the method used to discretize (30), the conservation of charge (31) in the discrete frame means that at two consecutive grid points the following relation should be examined

$$(\mathbf{y}^{(k+1)})^T (\mathcal{J} \otimes \mathcal{R})\mathbf{y}^{(k+1)} = (\mathbf{y}^{(k)})^T (\mathcal{J} \otimes \mathcal{R})\mathbf{y}^{(k)}. \tag{33}$$

We first consider the most commonly used schemes, namely the forward Euler and backward Euler methods. Since both are unable to preserve the charges – as will be proved below – we introduce another numerical method capable of doing so. Following Hairer et al. [11, Sec. IV.2], we consider the simplest case of Gauss collocation methods. In all cases, starting from an initial guess $\mathbf{y}^{(0)} = \mathbf{y}(t_0)$, we solve the equation in (30) on the uniform grid $\{t_k = t_0 + kh; k = 0, 1, \dots, N; t_N = T\}$.

5.0.1. The forward Euler method

The numerical solution obtained by applying the forward Euler method can be computed by

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + h\mathcal{E}(\mathbf{y}^{(k)})\mathbf{y}^{(k)} \tag{34}$$

where $\mathbf{y}^{(k)} \approx \mathbf{y}(t_k)$. Using this recurrence relation we can prove the following result.

Proposition 1. *The forward Euler method does not preserve the charge (31).*

Proof. By substituting in (33) the expression of $\mathbf{y}^{(k+1)}$ provided by (34) we can write

$$\begin{aligned} (\mathbf{y}^{(k+1)})^T (\mathcal{J} \otimes \mathcal{R})\mathbf{y}^{(k+1)} &= (\mathbf{y}^{(k)})^T (\mathcal{J} \otimes \mathcal{R})\mathbf{y}^{(k)} \\ &+ h(\mathbf{y}^{(k)})^T \left[\left(\mathcal{E}(\mathbf{y}^{(k)})\right)^T (\mathcal{J} \otimes \mathcal{R}) + (\mathcal{J} \otimes \mathcal{R})\mathcal{E}(\mathbf{y}^{(k)}) \right] \mathbf{y}^{(k)} \\ &+ h^2 (\mathbf{y}^{(k)})^T \left(\mathcal{E}(\mathbf{y}^{(k)})\right)^T (\mathcal{J} \otimes \mathcal{R})\mathcal{E}(\mathbf{y}^{(k)})\mathbf{y}^{(k)}. \end{aligned} \tag{35}$$

Then, taking into account (32) and observing that

$$\left(\mathcal{E}(\mathbf{y}^{(k)})\right)^T (\mathcal{J} \otimes \mathcal{R})\mathcal{E}(\mathbf{y}^{(k)}) = \frac{1}{\epsilon} \begin{pmatrix} O & C(\mathbf{y}^{(k)}) \\ C(\mathbf{y}^{(k)}) & O \end{pmatrix} (\mathcal{J} \otimes \mathcal{R}) \tag{36}$$

we have

$$(\mathbf{y}^{(k+1)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k+1)} = (\mathbf{y}^{(k)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k)} + \mathcal{O}\left(\frac{h^2}{\epsilon}\right).$$

This last relation leads us to the conclusion that the forward Euler method does not preserve the charge (see (33)). □

As just proved, the conservation of the charge is spoiled at order h^2. Furthermore, we observe that for $\epsilon \to \infty$ the charge is conserved. This is expected since in that limit we recover a simple model which conserve any charge. On the other side, in the limit $\epsilon \to 0$, that is when the diffusion equation is recovered, the second term could become arbitrarily large. Thus, we have to choose a different discretization scheme.

5.0.2. The backward Euler method

When we discretize the system (30) by the backward Euler method we get

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + h\mathcal{E}(\mathbf{y}^{(k+1)})\mathbf{y}^{(k+1)}$$

and then

$$\left(I_{2\hat{n}} - h\mathcal{E}(\mathbf{y}^{(k+1)})\right)\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)}. \tag{37}$$

Using this recurrence relation we can prove the following result.

Proposition 2. *The backward Euler method does not preserve the charge (31).*

Proof. By substituting in (33) the expression of $\mathbf{y}^{(k)}$ provided by (37), we can write

$$\begin{aligned}(\mathbf{y}^{(k)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k)} &= (\mathbf{y}^{(k+1)})^T\left(I_{2\hat{n}} - h\mathcal{E}(\mathbf{y}^{(k+1)})\right)^T(\mathcal{J}\otimes\mathcal{R})\left(I_{2\hat{n}} - h\mathcal{E}(\mathbf{y}^{(k+1)})\right)\mathbf{y}^{(k+1)}\\ &= (\mathbf{y}^{(k+1)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k+1)}\\ &- h(\mathbf{y}^{(k+1)})^T\left[\left(\mathcal{E}(\mathbf{y}^{(k+1)})\right)^T(\mathcal{J}\otimes\mathcal{R}) + (\mathcal{J}\otimes\mathcal{R})\mathcal{E}(\mathbf{y}^{(k+1)})\right]\mathbf{y}^{(k+1)}\\ &+ h^2(\mathbf{y}^{(k+1)})^T\left[\left(\mathcal{E}(\mathbf{y}^{(k+1)})\right)^T(\mathcal{J}\otimes\mathcal{R})\mathcal{E}(\mathbf{y}^{(k+1)})\right]\mathbf{y}^{(k+1)}.\end{aligned}$$

Using (32) and considering the expression of the matrix in (36) we obtain

$$(\mathbf{y}^{(k)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k)} = (\mathbf{y}^{(k+1)})^T(\mathcal{J}\otimes\mathcal{R})\mathbf{y}^{(k+1)} + \mathcal{O}\left(\frac{h^2}{\epsilon}\right). \tag{38}$$

and therefore the charge is not conserved. □

5.0.3. The implicit midpoint method

The scheme we propose to use in this setting works as follows

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + h\mathcal{E}(\mathbf{y}^{(\xi)})\frac{\mathbf{y}^{(k+1)} + \mathbf{y}^{(k)}}{2}, \tag{39}$$

where $\mathbf{y}^{(\xi)}$ approximates the continuous solution at a point $t_\xi \in [t_n, t_{n+1}]$. Recombining the above expression gives

$$\left(I_{2\hat{n}} - \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)\mathbf{y}^{(k+1)} = \left(I_{2\hat{n}} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)\mathbf{y}^{(k)}.$$

Since (see (30))

$$I_{2\hat{n}} - \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)}) = \begin{pmatrix} I_{\hat{n}} & -\frac{h\,e^{\frac{t}{\epsilon}}}{2}I_{\hat{n}} \\ \frac{h\,e^{-\frac{t}{\epsilon}}}{2\epsilon}C(\mathbf{y}^{(\xi)}) & I_{\hat{n}}\end{pmatrix},$$

the blocks on the main diagonal are both invertible and then this matrix can be inverted blockwise as follows

$$\left(I_{2\hat{n}} - \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)^{-1} = \left(I_2 \otimes \left[I_{\hat{n}} + \frac{h^2}{4\epsilon}C(\mathbf{y}^{(\xi)})\right]^{-1}\right)\left(I_{2\hat{n}} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right).$$

From the above considerations and by a direct calculation we have

$$\begin{aligned}\mathbf{y}^{(k+1)} &= \left(I_2 \otimes \left[I_{\hat{n}} + \frac{h^2}{4\epsilon}C(\mathbf{y}^{(\xi)})\right]^{-1}\right)\left(I_{2\hat{n}} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)^2\mathbf{y}^{(k)}\\ &= \left(I_{2\hat{n}} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)^2\left(I_2 \otimes \left[I_{\hat{n}} + \frac{h^2}{4\epsilon}C(\mathbf{y}^{(\xi)})\right]^{-1}\right)\mathbf{y}^{(k)}\\ &\equiv \mathcal{P}(\mathbf{y}^{(\xi)})\mathbf{y}^{(k)}.\end{aligned} \tag{40}$$

Now, denoting for the reader's convenience

$$\mathcal{G}(\mathbf{y}^{(\xi)}) := \left(I_{\hat{n}} + \frac{h^2}{4\epsilon} C(\mathbf{y}^{(\xi)})\right)^2 (I_n \otimes \mathcal{R})$$

and recalling that

$$-C(\mathbf{y}^{(\xi)})(I_n \otimes \mathcal{R}) + (I_n \otimes \mathcal{R}) C(\mathbf{y}^{(\xi)}) = O$$

since $(I_n \otimes \mathcal{R})$ is a block diagonal matrix whose entries commute with $\mathbb{W}$, we get

$$\left(I_{2\hat{n}} + \frac{h}{2}\left(\mathcal{E}(\mathbf{y}^{(\xi)})\right)^T\right)^2 (\mathcal{J} \otimes \mathcal{R}) \left(I_{2\hat{n}} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(\xi)})\right)^2 = \begin{pmatrix} O & \mathcal{G}(\mathbf{y}^{(\xi)}) \\ \left(\mathcal{G}(\mathbf{y}^{(\xi)})\right)^T & O \end{pmatrix}$$

$$= \begin{pmatrix} O & \left(I_{\hat{n}} + \frac{h^2}{4\epsilon} C(\mathbf{y}^{(\xi)})\right)^2 (I_n \otimes \mathcal{R}) \\ -\left(I_{\hat{n}} + \frac{h^2}{4\epsilon} C(\mathbf{y}^{(\xi)})\right)^2 (I_n \otimes \mathcal{R}) & O \end{pmatrix};$$

then

$$\left(\mathcal{P}(\mathbf{y}^{(\xi)})\right)^T (\mathcal{J} \otimes \mathcal{R}) \mathcal{P}(\mathbf{y}^{(\xi)}) = \mathcal{J} \otimes \mathcal{R}.$$

This equality together with (40) allows us to write

$$(\mathbf{y}^{(k+1)})^T (\mathcal{J} \otimes \mathcal{R}) \mathbf{y}^{(k+1)} = (\mathbf{y}^{(k)})^T (\mathcal{J} \otimes \mathcal{R}) \mathbf{y}^{(k)}.$$

Remark 1. It is important to emphasize that any point t_ξ is suitable for charge conservation. In particular, setting $t_\xi = t_k + \frac{h}{2} \equiv t_{k+\frac{1}{2}}$, the method (39) is an A-stable implicit method of the second order known in the literature as implicit midpoint method (the simple instance of Gauss collocation methods, with one collocation point per step). Actually this method can be seen as a sequence of backward Euler and then forward Euler methods. In fact, applying both methods step by step on a mesh with stepsize $h/2$ we have

$$\mathbf{y}^{(k+\frac{1}{2})} = \mathbf{y}^{(k)} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(k+\frac{1}{2})})\mathbf{y}^{(k+\frac{1}{2})}$$

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k+\frac{1}{2})} + \frac{h}{2}\mathcal{E}(\mathbf{y}^{(k+\frac{1}{2})})\mathbf{y}^{(k+\frac{1}{2})}$$

from which we deduce that

$$\mathbf{y}^{(k+\frac{1}{2})} - \mathbf{y}^{(k)} = \mathbf{y}^{(k+1)} - \mathbf{y}^{(k+\frac{1}{2})},$$

or, equivalently,

$$\mathbf{y}^{(k+\frac{1}{2})} = \frac{\mathbf{y}^{(k+1)} + \mathbf{y}^{(k)}}{2}.$$

We are now in the position to enunciate the following proposition.

Proposition 3. *The implicit midpoint method applied for solving the system (30) reads as*

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + h\mathcal{E}\left(\frac{\mathbf{y}^{(k+1)} + \mathbf{y}^{(k)}}{2}\right)\frac{\mathbf{y}^{(k+1)} + \mathbf{y}^{(k)}}{2}. \tag{41}$$

It preserves the charge (31) which is a quadratic integral invariant for the system.

6. Numerical experiments

In this section we report the results of some numerical experiments we have performed to show in a simple case the choices we have made in this work.

We consider a graph with three nodes (i.e. $n = 3$) with vector features of dimension $d = 4$, whose dynamic is modeled by the continuous system of the form (30) defined on the interval $(0, 1]$. The incidence matrix is defined as

$$w = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

so that each node is connected to all others. As the activation function g, we use the softmax function. Here we focus on two tests corresponding to two different ways of setting the matrix $\mathbb{W}$. In the first test we set $\mathbb{W} = 10^{-3} I_4$, while in the second one we consider $\mathbb{W} = \mathrm{diag}(10^{-3}, 10^{-3}, 1, 1)$. We will later refer to these two matrices as $\mathbb{W}_1$ and $\mathbb{W}_2$, respectively. To solve the corresponding two problems, we use the forward Euler method (FE) (34) and the (modified) implicit midpoint method (IM), that is the method given in (39) with the matrix $\mathcal{E}(\mathbf{y}^{(\xi)})$ evaluated at $t_\xi = t_k$. Although the latter method converges with order one, the numerical approximation of the solution to the next grid point can be achieved by solving only a linear system per step rather than a nonlinear

Table 1
Correspondence rotation-charge.

$\mathcal{R}$	$\mathcal{R}_{2,1}$	$\mathcal{R}_{3,1}$	$\mathcal{R}_{4,1}$	$\mathcal{R}_{3,2}$	$\mathcal{R}_{4,2}$	$\mathcal{R}_{4,3}$
Q_A	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6

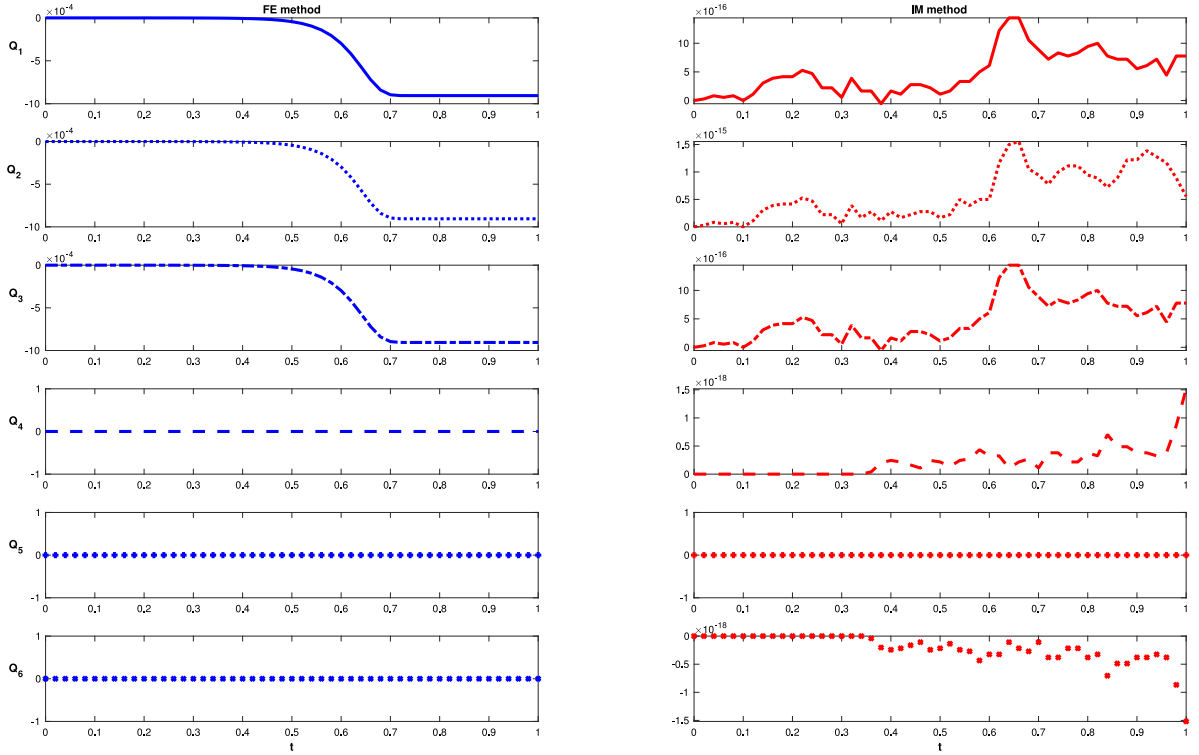


Fig. 1. Charge of the system (30) with $\epsilon = 0.1$ and $\mathbb{W}_1$ from Q_1 up to Q_6 on the left computed by the forward Euler (FE) method and on the right by the (modified) implicit midpoint (IM) method.

problem as would occur using the implicit midpoint method. This obviously leads to a significant saving on computational cost since the numerical solution of a nonlinear problem needs solve at least one linear system. Furthermore, as already highlighted in the previous section, this modification does not alter the system's charge conservation.

For the initial value $\mathbf{y}^{(0)} \in \mathbb{R}^{24}$ with

$$\begin{aligned}\mathbf{x}_1^{(0)} &= (0,1,1,1)^T,\\ \mathbf{x}_j^{(0)} &= (1,1,1,1)^T, \quad j = 2,3,\\ \mathbf{p}_\ell^{(0)} &= (0,0,0,0)^T, \quad \ell = 1,2,3,\end{aligned}$$

we calculate the numerical solution on a uniform grid with step size $h = 1/50$. In this case, the charge associated to the rotation $\mathcal{R}$ which commutes with $\mathbb{W}$ is $Q_A = 0$.

One of the rotation matrices that we considered in our tests is the matrix

$$\mathcal{R}_{2,1} = \begin{pmatrix} 0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The charge here called Q_1 is associated with this matrix. In a similar way we can define the other rotations we used. In Table 1 we name the charge associated with each of these rotations.

Since $\mathbb{W}_1$ commutes with all considered rotations, we expect according to the theoretical results of the previous section that the corresponding charges are preserved by the implicit midpoint method at each grid point; however, we generally do not expect this in the forward Euler method. This fact is confirmed by the images on the right and left in Fig. 1, where we plot the charges from Q_1 to Q_6 against time $t_k = kh, k = 0,1,\dots,50$ on both sides.

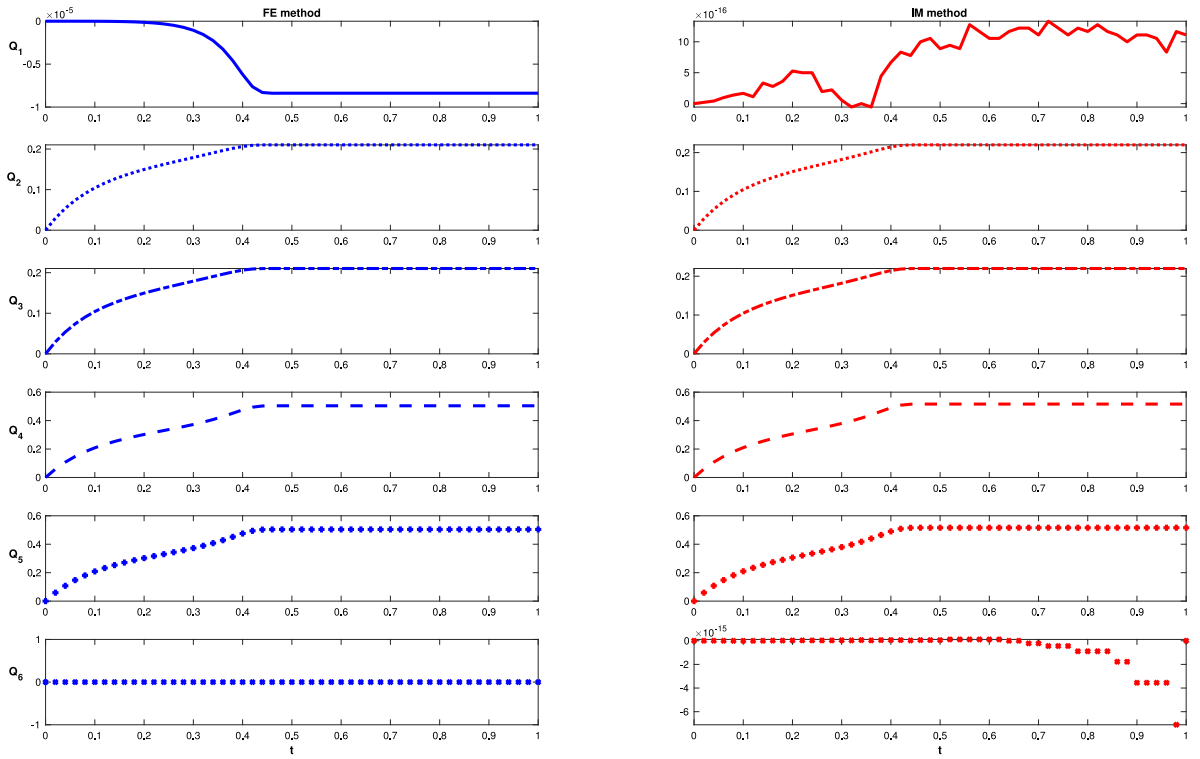


Fig. 2. Charge of the system (30) with $\epsilon = 0.1$ and $\mathbb{W}_2$ from Q_1 up to Q_6 on the left computed by the forward Euler (FE) method and on the right by the (modified) implicit midpoint (IM) method.

On the other hand, since $\mathbb{W}_2$ commutes only with $\mathcal{R}_{2,1}$ and $\mathcal{R}_{4,3}$, we expect the charges Q_1 and Q_6 to be preserved by the implicit midpoint method. This result is confirmed by the first and last images on the right-hand side in Fig. 2. As can be seen on the left side of the same figure, this does not happen when using the forward Euler method.

In both figures, we have given the results for $\epsilon = 0.1$. However, it should be emphasized that similar results can be obtained for other values of this parameter.

7. Conclusions

We systematically analyzed the mathematical framework of some GNNs based on differential equations. In particular, we studied the example of the diffusion equation on a graph. We translated the study of the equation into a functional language that exploits all the relevant symmetries of the model, and we provided the best-suited numerical scheme for the system of equations. The present work aimed to provide a unifying framework for GNNs that could have several potential applications. Primarily, this could be implemented in the context of the graph neural diffusion and any physical system governed by diffusion equations. We expected that this framework, being qualitatively closer to the continuous model, could lead to a significant improvement of the entire procedure. Complementary to more efficiency and implementability, we believe that this framework could bring a small step forward in understanding how GNNs capture the essential features from the data by learning the *constants of motions*, namely the conserved quantities of dynamic evolution, as the human brain does. Based on the idea that led us to develop the present approach, we plan to study numerical methods that preserve the conservation laws of other dynamical systems, such as the Schrödinger equation.

CRediT authorship contribution statement

Lidia Aceto: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Pietro Antonio Grassi:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank M. Dossena, C. Irwin, S. Montani and L. Portinale for valuable discussions. P.A.G. would like to thank M. Caselle for encouraging him to explore these new paths.

Appendix. A model on a smooth surface

In this appendix, we provide a continuum version of the model considered in the text. Instead of the graph, we consider a $\mathbb{R}^n \times [t_0, T]$ worldvolume and the fields $\vec{\phi}(x,t)$ where x, t are the coordinates (we replaced the nodes $i \in \mathcal{V}$ with local coordinates x). The features vectors $\mathbf{x}_i(t)$ on the nodes are replaced by continuum fields $\vec{\phi}(x,t)$ which are vectors in the feature space $\mathbb{R}^d$ as well.

The action is

$$I_\epsilon[\vec{\phi}, V] = \int_{t_0}^{T} dt e^{-\frac{t}{\epsilon}} \left(\int d^n x \frac{1}{2} \frac{\partial \vec{\phi}^T}{\partial t} \frac{\partial \vec{\phi}}{\partial t} + \frac{1}{2\epsilon} \int d^n x d^n y \nabla^x_\mu \vec{\phi}^T K[x,y] \nabla^{y,\mu} \vec{\phi} + V[\vec{\phi}] \right)$$

where $K[x,y]$ is a non-trivial kernel and $\nabla^x_\mu \vec{\phi}, \nabla^y_\mu \vec{\phi}$ are the derivatives with respect to the coordinates x, y. The second term is written in terms of a second integration over y's. This is due to the non-linear non-local nature of the model as G_{ij} in (10). We added to the Lagrangian a potential $V[\vec{\phi}]$ in order to describe further interactions.

Compared with the graph theory, the kernel is

$$K[x,y] = K[\vec{\phi}(x,t), \vec{\phi}(y,t)]$$

namely a function of the field $\vec{\phi}(x,t)$. Introducing the function

$$C(x,y) = \vec{\phi}^T(x,t) \mathbb{W} \vec{\phi}(y,t)\,,$$

we can construct the bilocal expression $K[\vec{\phi}(x,t), \vec{\phi}(y,t)]$ as a function of $C(x,y)$. For example, a possible non-linear non-local expression used in GNNs is

$$K[\vec{\phi}(x,t), \vec{\phi}(y,t)] = \frac{e^{C(x,y)}}{\int d^n y e^{C(x,y)}}\,,$$

with some additional conditions to make the expression well-defined.

Computing the functional derivative with respect to $\vec{\phi}(z,t)$ we get the equation of motion

$$-\frac{\partial^2 \phi^I(z,t)}{\partial t^2} + \frac{1}{\epsilon}\left[\frac{\partial \phi^I(z,t)}{\partial t} - \nabla^I_\phi V + \int d^n y \left(\nabla^x_\mu K[x,z]\Big|_{x=y} + \nabla^x_\mu K[z,x]\Big|_{x=y} \right) \nabla^{y,\mu} \phi^I + \int d^n x d^n y \nabla^x_\mu \vec{\phi}^T \frac{\partial K[x,y]}{\partial \phi^I(z,t)} \nabla^{y,\mu} \vec{\phi} \right] = 0\,. \tag{42}$$

In the limit $\epsilon \to 0$, we retrieve the Gradient Flow equation

$$\frac{\partial \phi^I}{\partial t} - \nabla^I_\phi V + \int d^n y \left(\nabla^x_\mu K[x,z]\Big|_{x=y} + \nabla^x_\mu K[z,x]\Big|_{x=y} \right) \nabla^{y,\mu} \phi^I + \int d^n x d^n y \nabla^x_\mu \vec{\phi}^T \frac{\partial K[x,y]}{\partial \phi^I(z,t)} \nabla^{y,\mu} \vec{\phi} = 0. \tag{43}$$

If $K[x,y] = \delta(x-y)$, (43) reduces to heat equation.

Data availability

No data was used for the research described in the article.

References

[1] Bronstein M, Bruna J, LeCun Y, Szlam A, Vandergheynst P. Geometric deep learning: Going beyond Euclidean data. IEEE Signal Process Mag 2016;34:18–42.
[2] Chamberlain B, Rowbottom J, Gorinova M, Webb S, Rossi E, Bronstein M. GRAND: Graph neural diffusion. In: International conference on machine learning. 2021.
[3] Di Giovanni F, Rowbottom J, Chamberlain B, Markovich T, Bronstein M. Graph neural networks as gradient flows. 2022, ArXiv Preprint ArXiv:2206.10991.
[4] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser Ł, Polosukhin I. Attention is all you need. Adv Neural Inf Process Syst 2017;30.
[5] Choi J, Hong S, Park N, Cho S. GREAD: Graph neural reaction-diffusion equations. 2022, http://dx.doi.org/10.48550/arXiv.2211.14208, CoRR. abs/2211.14208.
[6] Chamberlain B, Rowbottom J, Eynard D, Giovanni F, Dong X, Bronstein M. Beltrami flow and neural diffusion on graphs. 2021.
[7] Evans L. Partial differential equations. American Mathematical Society; 2022.
[8] Giovanni F, Rowbottom J, Chamberlain B, Markovich T, Bronstein M. Understanding convolution on graphs via energies. 2023.
[9] Mielke A, Ortiz M. A class of minimum principles for characterizing the trajectories and the relaxation of dissipative systems. ESAIM Control Optim Calc Var 2008;14:494–516. http://dx.doi.org/10.1051/cocv:2007064.
[10] Mielke A, Stefanelli U. Weighted energy-dissipation functionals for gradient flows. ESAIM Control Optim Calc Var 2011;17:52–85. http://dx.doi.org/10.1051/cocv/2009043.
[11] Hairer E, Lubich C, Wanner G. Geometric numerical integration. In: Structure-preserving algorithms for ordinary differential equations. Berlin: Springer-Verlag; 2006.

[12] Lutter M, Ritter C, Peters J. Deep Lagrangian networks: Using physics as model prior for deep learning. 2019.
[13] Cranmer M, Greydanus S, Hoyer S, Battaglia P, Spergel D, Ho S. Lagrangian neural networks. 2020.
[14] Greydanus S, Dzamba M, Yosinski J. Hamiltonian neural networks. Adv Neural Inf Process Syst 2019;32.
[15] Liu Z, Tegmark M. Machine learning conservation laws from trajectories. Phys Rev Lett 2021;126:180604.
[16] Mattheakis M, Protopapas P, Sondak D, Giovanni M, Kaxiras E. Physical symmetries embedded in neural networks. 2020.
[17] Bondesan R, Lamacraft A. Learning symmetries of classical integrable systems. 2019.
[18] Liu Z, Madhavan V, Tegmark M. Machine learning conservation laws from differential equations. Phys Rev E 2022;106:045307.
[19] Wang H, Fu T, Du Y, Gao W, Huang K, Liu Z, Chandak P, Liu S, Van Katwyk P, Deac A, et al. Scientific discovery in the age of artificial intelligence. Nature 2023;620:47–60.
[20] Forestano R, Matchev K, Matcheva K, Roman A, Unlu E, Verner S. Deep learning symmetries and their Lie groups, algebras, and subalgebras from first principles. Mach Learn: Sci Technol 2023.
[21] Hagemeyer C. Learning linear symmetries in data using moment matching. 2022.
[22] Gilmore R. Lie groups, Lie algebras, and some of their applications. Malabar, FL: Robert E. Krieger Publishing Co.,Inc.; 1994, Reprint of the 1974 original.
[23] Blanchard P, Higham D, Higham N. Accurately computing the log-sum-exp and softmax functions. IMA J Numer Anal 2021;41:2311–30. http://dx.doi.org/10.1093/imanum/draa038.