Engineering Computations

Physics-informed and graph neural networks for enhanced inverse analysis

Daniele Di Lorenzo
PIMM Lab, ENSAM Institute of Technology, Paris, France and ESI Group, Bagneux, France
Victor Champaney and Chady Ghnatios
PIMM Lab, ENSAM Institute of Technology, Paris, France
Elias Cueto
ESI Group-UZ Chair of the National Strategy on Artificial Intelligence, Aragon Institute of Engineering Research (I3A), Universidad de Zaragoza, Zaragoza, Spain, and
Francisco Chinesta
PIMM Lab, ENSAM Institute of Technology, Paris, France; ESI Group, Bagneux, France and CNRS@CREATE Ltd., Singapore, Singapore

Received 27 December 2023
Revised 23 July 2024
Accepted 10 August 2024

Abstract

Purpose – This paper presents an original approach for learning models, partially known, of particular interest when performing source identification or structural health monitoring. The proposed procedures employ some amount of knowledge on the system under scrutiny as well as a limited amount of data efficiently assimilated.
Design/methodology/approach – Two different formulations are explored. The first, based on the use of informed neural networks, leverages data collected at specific locations and times to determine the unknown source term of a parabolic partial differential equation. The second procedure, more challenging, involves learning the unknown model from a single measured field history, enabling the localization of a region where material properties differ.
Findings – Both procedures assume some kind of sparsity, either in the source distribution or in the region where physical properties differ. This paper proposed two different neural approaches able to learn models in order to perform efficient inverse analyses.
Originality/value – Two original methodologies are explored to identify hidden property that can be recovered with the right usage of data. Both methodologies are based on neural network architecture.

Keywords Machine learning, Inverse analysis, Data completion, Graph neural networks, Model correction, Physics-informed neural networks

Paper type Research paper

1. Introduction

Inverse problems play a crucial role in various scientific and engineering disciplines, where the goal is to infer unknown causes or parameters from observed effects. These problems are

The authors knowledge the contribution and support of the ESI-ENSAM research chair, CREATE-ID. This project has received funding from the European Union Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 956401 (XS-Meta).

Engineering Computations
Emerald Publishing Limited
0264-4401
DOI 10.1108/EC-12-2023-0958

often challenging due to their ill-posed nature, meaning that solutions may not exist, may not be unique or may be highly sensitive to data perturbations. Traditional approaches to inverse problems typically rely on optimization techniques that minimize the discrepancy between observed data and model predictions.

Recently, the advent of machine learning and data-driven approaches has provided new avenues for tackling inverse problems. These methods can leverage large datasets and powerful computational tools to learn complex models that can generalize well beyond the training data (Zhang *et al.*, 2022, 2023). In this context, two promising neural network-based methodologies have emerged for performing efficient inverse analyses: physics-informed neural networks (PINNs) and graph neural networks (GNNs).

This paper explores these two methodologies to perform inverse analyses reformulation the inverse problem in a direct manner. The first approach utilizes PINNs to identify the source location and intensity of thermal and pollution problems. This is achieved by combining the physical knowledge of the problem with some measurements and a regularization technique (Lorenzo *et al.*, 2022, 2023). The second approach employs GNNs to learn from a single measurement and infer the problem solution while identifying regions where material properties, such as thermal conductivity, differ from those in the surrounding area. GNNs are well-suited for problems involving complex geometries and topologies, as they can naturally incorporate the connectivity and relationships between different parts of the system.

The present paper is structured as follows: Section 2 presents the detailed methodology of the proposed design, the implementation and the results of the inverse methodology based on PINN. Section 3 presents the implementation and the results of the proposed GNN approaches, showcasing the performance and capabilities of this methodology. Finally, Section 4 provides conclusions and perspectives, summarizing the key findings and outlining potential future research directions.

2. Learning partially known models: source identification using PINNs

In the domain of inverse problems, source detection plays a crucial role in uncovering hidden origins of various phenomena (Adel, 2012; Torre *et al.*, 2015). Whether it involves identifying the source of a signal, locating a physical anomaly or pinpointing the root cause of a complex issue, source detection harnesses the power of data and mathematical modeling to reveal elusive sources.

This section aims at proving how the proposed methodology based on PINNs provides an appealing procedure that merges machine learning with physics principles and data to enhance source detection accuracy. By integrating PINNs into the study of inverse problems, we can refine our ability to pinpoint sources with precision and extract valuable insights from complex datasets.

For that purpose, we explore source detection within the context of diffusion (thermal conduction) and advection-diffusion (pollution-related) problems. It is important to note that while our discussion is centered on these specific domains, the proposed methodology maintains a level of generality and with some necessary adjustments and adaptations, it can be readily applied to a broader range of fields.

2.1 Thermal problem with unknown location of the source

Heat source detection (Chen *et al.*, 2023; Janne, 2020; Wang *et al.*, 2022a) plays a pivotal role in structural engineering, impacting safety, maintenance and the overall integrity of buildings and infrastructure. Whether dealing with potential fire hazards, equipment malfunctions or the monitoring of complex industrial processes, the capability to identify heat sources promptly can significantly impact safety, maintenance practices and structural durability over time.

Although early detection of heat sources is crucial in various fields (e.g. fire prevention, equipment malfunctions, . . .) our primary focus here is on industrial processes. In industrial settings, monitoring heat sources is essential for maintaining the integrity of equipment, optimizing processes and preventing costly downtime. This section explores the possibility of using physics-informed learning combined with data completion (data augmentation) to devise a solution for heat source detection in structural engineering.

For that purpose the heat equation is considered:

$$\begin{cases} \rho c_p \dfrac{\partial T(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{q}(\mathbf{x},t) = S(\mathbf{x},t) & \mathbf{x} \in \Omega, t \in [0,\Theta] \\ \mathbf{q}(\mathbf{x},t) = -k(\mathbf{x})\ \nabla T(\mathbf{x},t) & \\ T(\mathbf{x},t) = \phi(\mathbf{x},t) & \mathbf{x} \in \Gamma_d, t \in [0,\Theta] \\ k(\mathbf{x}) \nabla T(\mathbf{x},t) \cdot \mathbf{n} = \ \gamma(\mathbf{x},t) & \mathbf{x} \in \Gamma_n, t \in [0,\Theta] \\ T(\mathbf{x},0) = T_0(\mathbf{x}) & \mathbf{x} \in \Omega \end{cases}, \tag{1}$$

where $\mathbf{x}$ and t are the space and time coordinates, defined in the space-time domain $\Omega \times [0, \Theta]$, with $\Gamma = \Gamma_d \cup \Gamma_n$ its boundary, decomposed in the parts in which temperature or thermal fluxes are enforced, Γ_d and Γ_n, respectively, ρ is the mass density, c_p the specific heat capacity and $k(\mathbf{x})$ the thermal conductivity.

Considering the case in which we just have Dirichlet boundary conditions, that is $\Gamma_d = \Gamma$, and constant density, heat capacity ($\rho = 1$, $c_p = 1$) and conductivity, the problem (1) reduces to:

$$\begin{cases} \dfrac{\partial T(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{q}(\mathbf{x},t) = S(\mathbf{x},t) & \mathbf{x} \in \Omega, t \in [0,\Theta] \\ \mathbf{q}(\mathbf{x},t) = -k(\mathbf{x})\ \nabla T(\mathbf{x},t) & \\ T(\mathbf{x},t) = \phi(\mathbf{x},t) & \mathbf{x} \in \Gamma, t \in [0,\Theta] \\ T(\mathbf{x},0) = T_0(\mathbf{x}) & \mathbf{x} \in \Omega \end{cases}. \tag{2}$$

We further assume that the temperature of the actual structure is measured at (N_{sens}) sensor locations, at different times, $\mathbf{T}^{Sens} = \{T_1^{Sens}, \ldots, T_{N_{sens}}^{Sens}\}$, along with the nominal source S^{nom}, which differs from the real one in a zone of the domain. We can formulate the NN loss function (Raissi *et al.*, 2017a, b) in a manner that ensures the PDE (2) upon minimization. Simultaneously, this approach enables us to address the inverse problem of identifying the actual source and distinguishing it from the nominal one. The loss function (where $\widehat{\bullet}$ is the neural network prediction) involves seven terms.

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_b + \mathcal{L}_{in} + \mathcal{L}_{c1} + \mathcal{L}_{c2} + \mathcal{L}_{dat} + \mathcal{L}_S, \tag{3}$$

where

$$\mathcal{L}_d = \frac{1}{N_d}\sum_j^{N_d}\left|\frac{\partial\widehat{T}(\mathbf{x}^j,t^j)}{\partial t}+\frac{\partial\widehat{q_x}(\mathbf{x}^j,t^j)}{\partial x}+\frac{\partial\widehat{q_y}(\mathbf{x}^j,t^j)}{\partial y}-\widehat{S}(\mathbf{x}^j,t^j)\right|^2, \tag{4}$$

$$\mathcal{L}_{in} = \frac{1}{N_{in}}\sum_j^{N_{in}}\left|\widehat{T}(\mathbf{x}^j,0)-T_0(\mathbf{x}^j)\right|^2, \tag{5}$$

$$\mathcal{L}_{bd} = \frac{1}{N_{bd}}\sum_j^{N_{dc}}\left|\widehat{T}(\mathbf{x}^j,t^j)-\phi(\mathbf{x}^j,t^j)\right|^2, \tag{6}$$

$$\mathcal{L}_{c1} = \frac{1}{N_d}\sum_j^{N_d}\left|-k\frac{\partial\widehat{T}(\mathbf{x}^j,t^j)}{\partial x}-\widehat{q_x}(\mathbf{x}^j,t^j)\right|^2, \tag{7}$$

$$\mathcal{L}_{c2} = \frac{1}{N_d}\sum_j^{N_d}\left|-k\frac{\partial\widehat{T}(\mathbf{x}^j,t^j)}{\partial y}-\widehat{q_y}(\mathbf{x}^j,t^j)\right|^2, \tag{8}$$

$$\mathcal{L}_{dat} = \frac{1}{N_{sens}}\sum_j^{N_{sens}}\left|\widehat{T}(\mathbf{x}^j,t^j)-T^{sens}(\mathbf{x}^j,t^j)\right|^2, \tag{9}$$

$$\mathcal{L}_S = \frac{1}{N_d}\sum_j^{N_d}\left|\widehat{S}(\mathbf{x}^j,t^j)-S^{nom}(\mathbf{x}^j,t^j)\right|, \tag{10}$$

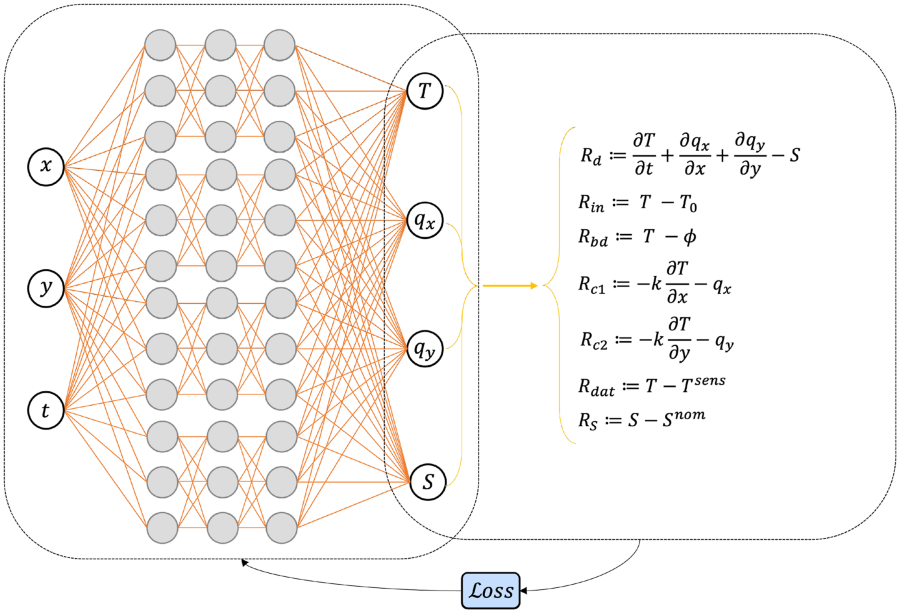
represent, respectively, the squared residuals of the heat equation, the initial condition, the boundary conditions, the Fourier's law, the data to fit and the residual of the source correction; N_d, N_{in}, N_{bd} and N_{sens} are the numbers of data points for the different terms. Usually the different terms of the loss function (3) are multiplied by a weight, whose choice affects the solution and the performance of PINN. To mitigate these pathologies, we used the proposed technique in Wang *et al.* (2022b) (based on the neural tangent kernel) theory, which allows us to define and optimize the value of these weights during the training process. For what follows in this section, we will omit the weights from all other loss functions, knowing that the same technique has been used for all of them. A schematic representation of the PINN architecture used, is shown in Figure 1. In particular, four fully connected neural networks are employed to infer the solution: one to predict the temperature, two to predict the vector heat flux and the last the source.

It is important to note that the term affecting the source correction involves the L1-norm, to emphasize the sparse nature of the source correction.

2.2 Case study

We employ the outlined methodology to determine the temperature field and detect the thermal source in the square plate depicted in Figure 2a, of dimension $L = 10\ m$. The temperature is enforced on the domain boundary, $\phi(\mathbf{x}, t) = 0°$, being the initial condition given by $T_0(\mathbf{x}) = 0°$. In the nominal configuration, no thermal source applies in the plate, that is, $S^{nom}(\mathbf{x}, t) = 0$, whereas the real system is assumed to contain a constant source as shown in Figure 2c. The real structure is equipped with N_{sens} temperature sensors, the locations of which are shown in Figure 2b. Due to the high density of sensors, it could be considered that a thermal camera is used to measure the temperature on the real structure in this case.

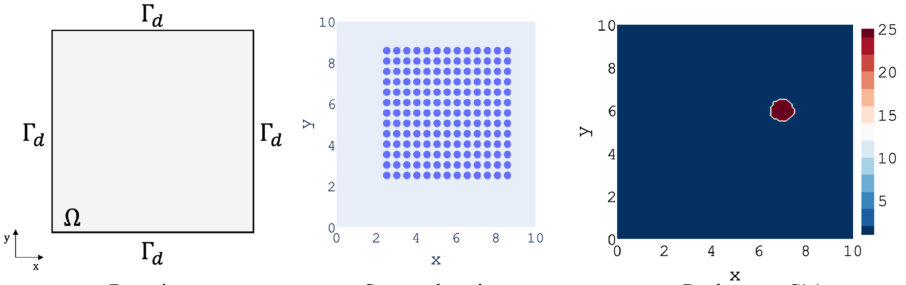
2.2.1 Data generation. The data retrieved in the sensor locations are synthetic data produced using the PINN approach. In particular, we solved system (2) for a time of



Note(s): Four fullyconnected neural network are used to approximate the solution $T(x, t)$, $q_x(x, t)$, $q_y(x, t)$, and the source $S(x, t)$, which are then used to define the loss L from the corresponding residuals R.

Source(s): Authors' own creation

Figure 1.
PINN architecture used to address the heat source detection problem



Source(s): Authors' own creation

Figure 2.
Configuration setup for the thermal source detection problem

five seconds ($\Theta = 5\ s$), with null Dirichlet and initial conditions, constant thermal conductivity ($k = 1\ W/(m \cdot K)$) and with the source location depicted in Figure 2c. A representation of the employed PINN architecture is given in Figure 3. In this architecture, one fully connected neural network (consisting of four layers, with 40 neural units per layer and an exponential linear unit – ELU – activation function) is used to predict the temperature. The network's parameters are learned through minimization of the loss function.

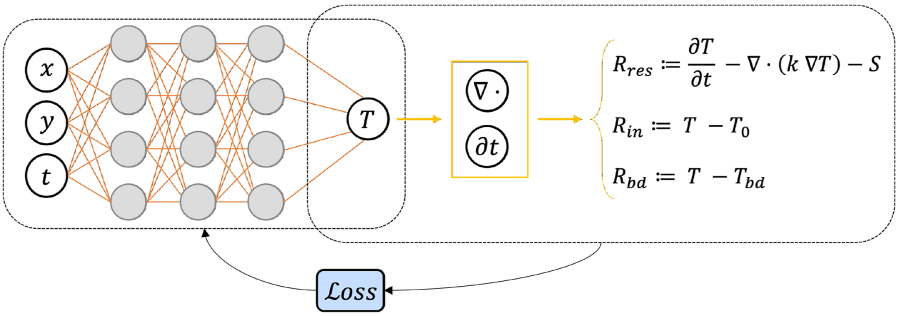


Figure 3.
PINN architecture employed to generate data for the heat source detection problem

Note(s): One fully-connected neural network is used to approximate the solution $T(x, t)$
Source(s): Authors' own creation

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_{bd} + \mathcal{L}_{in}, \tag{11}$$

where

$$\mathcal{L}_d = \frac{1}{N_d} \sum_i^{N_d} \left| \frac{\partial \widehat{T}(\mathbf{x}^i, t^i)}{\partial t} - \nabla \cdot \left(k \nabla \widehat{T}(\mathbf{x}^i, t^i) \right) - S(\mathbf{x}^i) \right|^2, \tag{12}$$

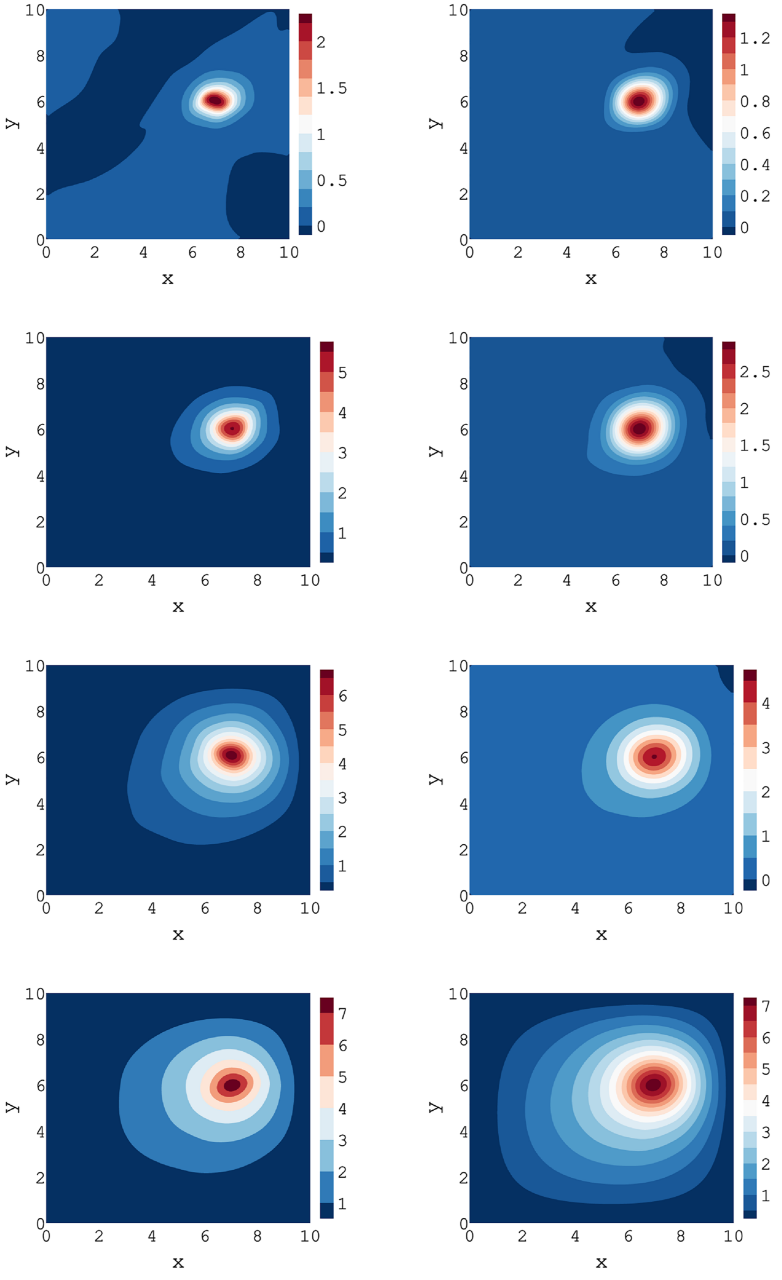
$$\mathcal{L}_{in} = \frac{1}{N_{in}} \sum_j^{N_{in}} \left| \widehat{T}(\mathbf{x}^j, 0) - T_0(\mathbf{x}^j) \right|^2, \tag{13}$$

$$\mathcal{L}_{bd} = \frac{1}{N_{bd}} \sum_j^{N_{bd}} \left| \widehat{T}(\mathbf{x}^j, t^j) - T_{bd}(\mathbf{x}^j, t^j) \right|^2, \tag{14}$$

are the squared residuals of the heat equation, boundary and the initial conditions. During the training, the boundary and initial conditions have been strongly enforced, as proposed in Lagaris *et al.* (1998), thereby eliminating the terms (13) and (14) in the loss function previously presented. After the training, the temperature is extracted at the measurement locations and used as sensor data.

2.2.2 Results of the thermal source identification problem. The four fully connected neural networks, employed to predict the temperature, the vector heat flux and the source, have the same structure: four-layers, each with 40 neural units per layer and an ELU activation function. The parameters of the network are learned by minimizing the loss function (3) across 10,000 points distributed in the space-time domain using the standard Adam optimizer with a learning rate of 0.001. Also for this case, the boundary and initial conditions have been strongly enforced, thus eliminating the terms (5) and (6) from the loss function. The temperature and the source obtained are shown and compared with their reference counterparts in Figures 4 and 5, respectively.

During the training process, approximately 800 epochs (batch size = 100) were employed to achieve convergence to an accurate enough solution. The results indicate that although the temperature field is not perfectly approximated, the neural network effectively captures the trend of the deviation between the nominal model and the actual solution. Furthermore, the methodology demonstrates a high level of accuracy in pinpointing the location of the thermal source, although its magnitude is not recovered with great precision. This discrepancy can be improved by increasing the number of epochs and data points used to evaluate the loss function.



Source(s): Authors' own creation

Figure 4.
Comparison between the reference **(left)** and computed **(right)** temperatures at different time steps, from top to bottom: **t = 0.3 s, 1 s, 2.5 s** and **5 s**

2.3 Adding advective transport mechanisms: pollutant source distribution identification

Pollutant source detection stands as a critical concern in environmental science and engineering. The early and precise identification of pollutant sources remains essential for advancing environmental sustainability, mitigating risks to ecosystems and human health and fostering responsible environmental practices.

The deployment of real-time monitoring systems, equipped with state-of-the-art sensors, facilitates a continuous stream of data, enabling the swift detection of pollutant sources. Furthermore, the Internet of Things (IoT) and sensor networks provide the means to seamlessly integrate data from a myriad of sensors, presenting a comprehensive perspective on plume dispersion and air quality predictions (Gavrilas *et al.*, 2022; Ullo and Sinha, 2020; Zhang *et al.*, 2021).

The integration of data from various sources, including remote sensing, ground-based sensors and satellite imagery, significantly enhances the precision of pollutant source detection. Machine learning algorithms, encompassing neural networks and support vector machines, exhibit a prowess in the analysis of intricate datasets, excelling in pattern recognition and the identification of pollutant sources (Tahir Bahadur *et al.*, 2023; Bellinger *et al.*, 2017; Mahalingam *et al.*, 2019).

In addition, predictive modeling techniques, play a pivotal role in simulating the dispersion of pollutants, thereby aiding in the identification of sources. This capability is not limited to industrial settings, where the early detection of pollutant sources, such as leaks and emissions, emerges as a keystone to avoid environmental damage and optimizing process efficiency. This section explores the use of informed learning for pollutant source detection.

Following the rationale described in Section 2.1, the advection–diffusion equation to model the concentration (c) of a pollution in $\Omega \times [0, \Theta]$ is now considered. The problem reads:

$$\begin{cases} \dfrac{\partial c(\mathbf{x},t)}{\partial t} + \nabla \cdot (\mathbf{u}(\mathbf{x},t)\ c(\mathbf{x},t)) + \nabla \cdot \mathbf{q}(\mathbf{x},t) = S(\mathbf{x},t) & \mathbf{x} \in \Omega, t \in [0,\Theta] \\ \mathbf{q} = -k(\mathbf{x})\ \nabla c(\mathbf{x},t) & \\ c(\mathbf{x},t) = \phi(\mathbf{x},t) & \mathbf{x} \in \Gamma_d, t \in [0,\Theta] \\ -k(\mathbf{x}) \nabla c(\mathbf{x},t) \cdot \mathbf{n} = \ \gamma(\mathbf{x},t) & \mathbf{x} \in \Gamma_n, t \in [0,\Theta] \\ c(\mathbf{x},0) = c_0(\mathbf{x}) & \mathbf{x} \in \Omega \end{cases}, \tag{15}$$

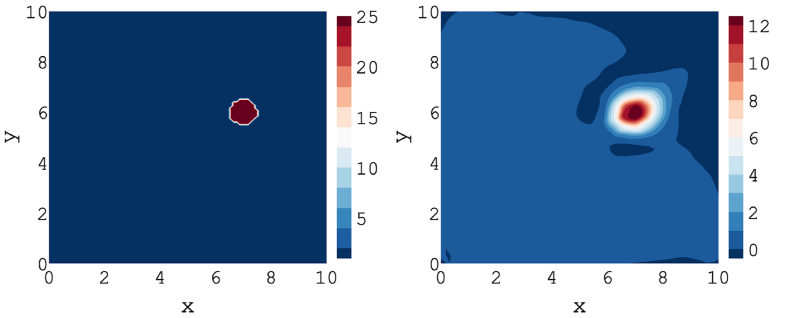


Figure 5.
The reference **(left)** and computed **(right)** source for the thermal problem

Source(s): Authors' own creation

where $\mathbf{x}$ and t are the space and time coordinates, Ω is the domain in which the problem is defined, $\Gamma = \Gamma_d \cup \Gamma_n$ its boundary, $\mathbf{u}(\mathbf{x}, t)$ is the velocity field that advects the pollutant and $k(\mathbf{x})$ is the diffusivity. Considering the case in which the diffusivity is constant, k, the flow is incompressible ($\nabla \cdot \mathbf{u} = 0$) and the source term just depends on the space coordinates, that is $S(\mathbf{x}, t) = S(\mathbf{x})$, the problem (15) reduces to:

$$\begin{cases} \dfrac{\partial c(\mathbf{x},t)}{\partial t} + \mathbf{u}(\mathbf{x},t)\cdot \nabla c(\mathbf{x},t) + \nabla \cdot \mathbf{q}(\mathbf{x},t) = S(\mathbf{x}) & \mathbf{x}\in\Omega, t\in[0,\Theta] \\ \mathbf{q} = -k\ \nabla c(\mathbf{x},t) & \\ c(\mathbf{x},t) = \phi(\mathbf{x},t) & \mathbf{x}\in\Gamma, t\in[0,\Theta] \\ -k\nabla c(\mathbf{x},t)\cdot\mathbf{n} = \ \gamma(\mathbf{x},t) & \mathbf{x}\in\Gamma_n, t\in[0,\Theta] \\ c(\mathbf{x},0) = c_0(\mathbf{x}) & \mathbf{x}\in\Omega \end{cases} . \tag{16}$$

We operate under the assumption that the pollutant concentration at the locations of N_{sens} sensors, $\mathbf{c}^{Sens} = \{c_1^{Sens}, \ldots, c_{N_{sens}}^{Sens}\}$ and the velocity field $\mathbf{u}(\mathbf{x}, t)$ are known. Additionally, it is presumed that the real source differs from its nominal representation S^{nom} only in a specific, small enough, region of the domain. In this scenario, the loss function, designed to satisfy the PDE (16) upon minimization, involves eight distinct terms.

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_{c1} + \mathcal{L}_{c2} + \mathcal{L}_{in} + \mathcal{L}_{bd} + \mathcal{L}_{bn} + \mathcal{L}_{dat} + \mathcal{L}_S, \tag{17}$$

where

$$\begin{aligned} \mathcal{L}_d \quad = \frac{1}{N_d}\sum_j^{N_d} \Bigg| & \frac{\partial \widehat{c}(\mathbf{x}^j, t^j)}{\partial t} + u_x(\mathbf{x}^j, t^j)\frac{\partial \widehat{c}(\mathbf{x}^j, t^j)}{\partial x} + u_y(\mathbf{x}^j, t^j)\frac{\partial \widehat{c}(\mathbf{x}^j, t^j)}{\partial y} \\ & -k\left(\frac{\partial \widehat{q}_x(\mathbf{x}^j, t^j)}{\partial x} + \frac{\partial \widehat{q}_y(\mathbf{x}^j, t^j)}{\partial y}\right) - \widehat{S}(\mathbf{x}^j, t^j)\Bigg|^2, \end{aligned} \tag{18}$$

$$\mathcal{L}_{c1} = \frac{1}{N_d}\sum_j^{N_d}\left| -k\frac{\partial \widehat{c}(\mathbf{x}^j, t^j)}{\partial x} - \widehat{q}_x(\mathbf{x}^j, t^j)\right|^2, \tag{19}$$

$$\mathcal{L}_{c2} = \frac{1}{N_d}\sum_j^{N_d}\left| -k\frac{\partial \widehat{c}(\mathbf{x}^j, t^j)}{\partial y} - \widehat{q}_y(\mathbf{x}^j, t^j)\right|^2, \tag{20}$$

$$\mathcal{L}_{in} = \frac{1}{N_{in}}\sum_j^{N_{in}}\left|\widehat{c}(\mathbf{x}^j, 0) - c_0(\mathbf{x}^j)\right|^2, \tag{21}$$

$$\mathcal{L}_{bd} = \frac{1}{N_{bd}}\sum_j^{N_{bd}}\left|\widehat{c}(\mathbf{x}^j, t^j) - \phi(\mathbf{x}^j, t^j)\right|^2, \tag{22}$$

$$\mathcal{L}_{bn} = \frac{1}{N_{bn}} \sum_{j}^{N_{bn}} \left| -k\nabla \widehat{c}\left(\mathbf{x}^j, t^j\right) \cdot \mathbf{n} - \ \gamma\left(\mathbf{x}^j, t^j\right) \right|^2, \tag{23}$$

$$\mathcal{L}_{dat} = \frac{1}{N_{sens}} \sum_{j}^{N_{sens}} \left| \widehat{c}\left(\mathbf{x}^j\right) - c^{sens}\left(\mathbf{x}^j\right) \right|^2, \tag{24}$$

$$\mathcal{L}_{S} = \frac{1}{N_{d}} \sum_{j}^{N_{d}} \left| \widehat{S}\left(\mathbf{x}^j, t^j\right) - S^{nom}\left(\mathbf{x}^j, t^j\right) \right|, \tag{25}$$

correspond, respectively, to the squared residuals of the PDE equation, the concentration gradient, the initial and boundary conditions, the data to fit and the detection of the source. The numbers of data points allocated for each of these terms are specified as N_d, N_{in}, N_{bd}, N_{bn} and N_{sens}, respectively.

Regarding the term (25) in the loss function, the L1-norm is preferred to the L2-norm, as for the other applications, it is assumed that the source differs from the nominal one only in a localized, small enough, area. The employed PINN architecture is shown in Figure 6.

2.3.1 Case study. In this section, we apply the proposed methodology to determine the concentration and identify the pollution source in a 2D domain. The domain dimensions are $L_x = 15\ m$ and $L_y = 6\ m$, with null concentration on the right, top and bottom sides, $\phi(\mathbf{x}, t) = 0$, null Neumann condition on the left side, $\gamma(\mathbf{x}, t) = 0$ and constant initial condition $c_0(\mathbf{x}) = 0$, as Figure 7a describes. Furthermore, a constant velocity field $\mathbf{u} = (-1.5, 0)\ m/s$ is considered. In the nominal configuration, there is no pollution source $S^{nom}(\mathbf{x}) = 0$, whereas the real (reference) system is assumed to contain a source, as the one depicted in Figure 7c. Data are collected from six mobile sensors (e.g. emulating drones) moving along the horizontal lines and measuring the concentration at the locations indicated in Figure 7b at different time instants (when the mobile sensors reach the indicated positions).

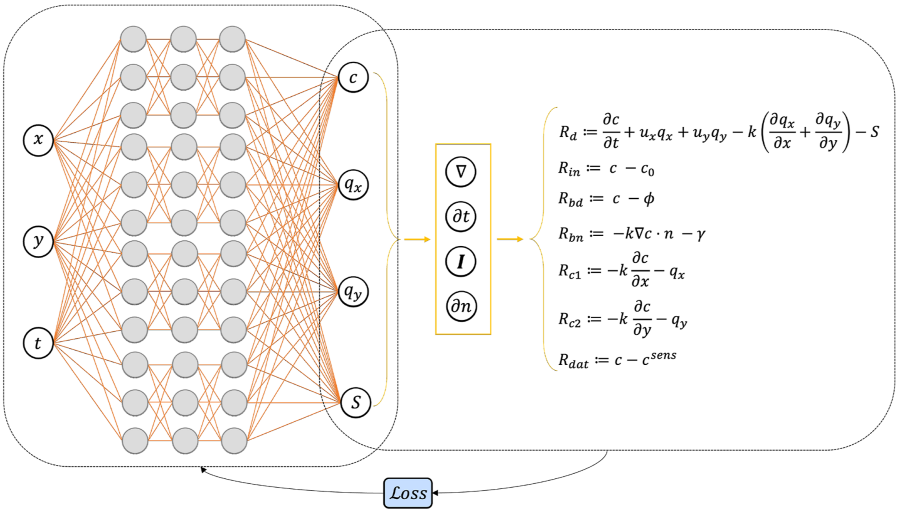


Figure 6.
PINN architecture used for addressing the source pollution problem

Note(s): Four fully-connected neural network are used to approximate the polluant concentration $c(x, t)$, the gradient of the concentration ($q_x(x, t)$, $q_y(x, t)$) and the source $S(x, t)$

Source(s): Authors' own creation

2.3.2 Data generation. Also, in this case, the data retrieved in the sensor locations are synthetic data produced using a PINN. In particular, we solved system (16) in a time interval of five seconds ($\Theta = 5\ s$), with null Dirichlet, Neumann and initial conditions, constant diffusivity ($k = 1\ m^2/s$), constant velocity field $\mathbf{u} = (-1.5, 0)\ m/s$ and with the source depicted in Figure 7c. Figure 8 illustrates the PINN architecture employed, featuring a fully connected neural network with four layers, each consisting of 40 neural units and employing an ELU activation function to predict the concentration. The network's parameters are learned by minimizing the loss function.

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_{bd} + \mathcal{L}_{bn} + \mathcal{L}_{in}, \tag{26}$$

where

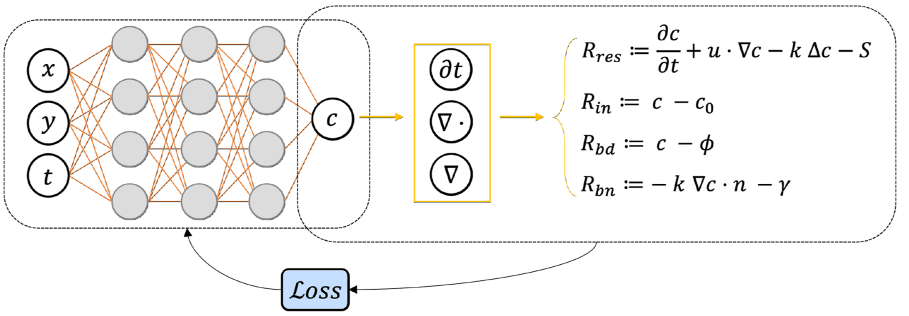
$$\mathcal{L}_d = \frac{1}{N_d}\sum_j^{N_d}\left|\frac{\partial\widehat{c}(\mathbf{x}^j,t^j)}{\partial t} + \mathbf{u}(\mathbf{x}^j,t^j)\cdot\nabla c(\mathbf{x}^j,t^j) - k\Delta\widehat{c}(\mathbf{x}^j,t^j) - \widehat{S}(\mathbf{x}^j,t^j)\right|^2, \tag{27}$$

$$\mathcal{L}_{bd} = \frac{1}{N_{bd}}\sum_j^{N_{dc}}\left|\widehat{c}(\mathbf{x}^j,t^j) - \phi(\mathbf{x}^j,t^j)\right|^2, \tag{28}$$

$$\mathcal{L}_{bn} = \frac{1}{N_{bn}}\sum_j^{N_{dc}}\left|-k\nabla\widehat{c}(\mathbf{x}^j,t^j)\cdot n - \gamma(\mathbf{x}^j,t^j)\right|^2, \tag{29}$$

Source(s): Authors' own creation

Figure 7.
Problem configuration



Note(s): One fully-connected neural network is used to approximate the concentration $c(x, t)$

Source(s): Authors' own creation

Figure 8.
PINN architecture used to compute the data used in the identification of the pollution source problem

$$\mathcal{L}_{in} = \frac{1}{N_{in}} \sum_{j}^{N_{in}} \left| \widehat{c}\left(\mathbf{x}^j, 0\right) - c_0\left(\mathbf{x}^j\right) \right|^2, \tag{30}$$

in 8000 points distributed within the space-time domain.

The terms (28) and (29) have been strongly enforced and thus removed, along with (30), from the loss function. After training, the output is used to retrieve the data corresponding to the sensor locations.

2.3.3 Results of the pollutant source identification problem. By considering the methodology previously introduced and utilizing the data computed as described in the previous section, we employ four fully connected neural networks to infer the solution: one for predicting the concentration, one for determining the source and two others to define the gradient of concentration. Each neural network consists of four layers with 40 neurons per layer, and the ELU activation function has been adopted as the activation function for all of them. The parameters of the network are learned by minimizing the loss function (17) across 12,000 points distributed in the space-time domain using the standard Adam optimizer with a learning rate of 0.001. Also for this case, the boundary and initial conditions have been strongly enforced, thus eliminating the terms (21)–(23) from the loss function. The concentration and the source obtained are displayed and compared with their reference counterparts in Figures 9 and 10, respectively.

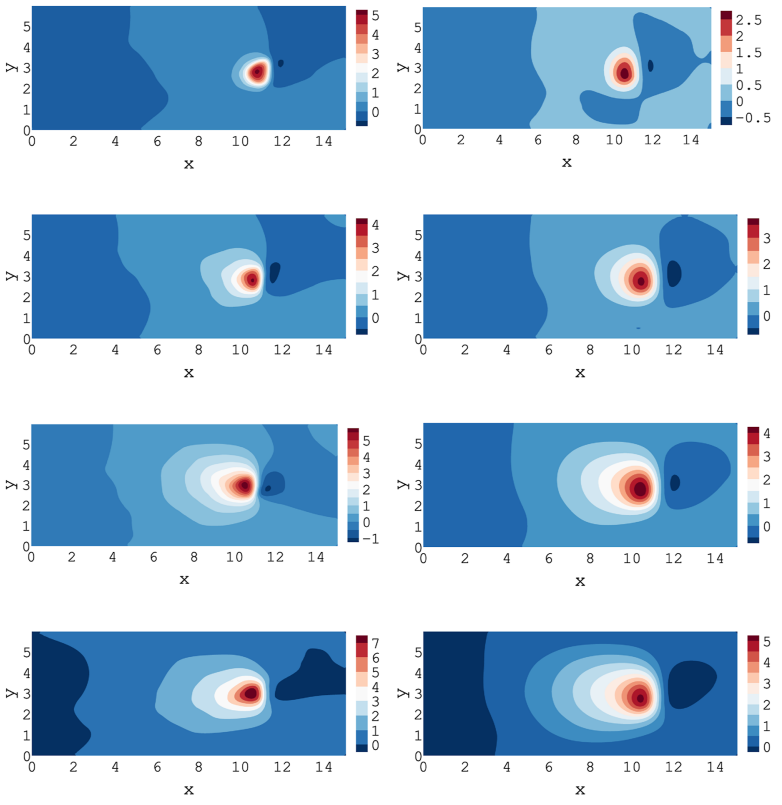
During the training process, approximately 750 epochs (batch size = 80) were required to obtain a solution accurate enough. The outcomes suggest that while the concentration field is not replicated with high accuracy, the neural network effectively captures the discrepancy between the nominal model and the reference solution. Furthermore, the methodology demonstrates good accuracy in detecting the location of the pollution source. However, estimating its magnitude lacks precision. This discrepancy can be reduced by increasing the number of epochs and data points used in training the loss function.

3. Learning the model and physical parameters from the use of graph neural networks – GNN

Finite element methods (FEM), have been long employed as a vital tool for solving partial differential equations (PDEs). However, in the framework of ML, GNNs (Hamilton, 2020; Wu *et al.*, 2020; Zhou *et al.*, 2018) have emerged as a powerful tool for learning representations and conducting inference on graph-structured data. And unlike traditional neural networks designed for grid-like data, GNNs can capture complex relationships and dependencies present in graphs, making them well-suited for a wide range of applications such as social network analysis, molecular chemistry (Oliver *et al.*, 2020; Wu *et al.*, 2023), complex systems (Ha and Jeong, 2021; Hernandez *et al.*, 2022) and more (Pfaff *et al.*, 2020).

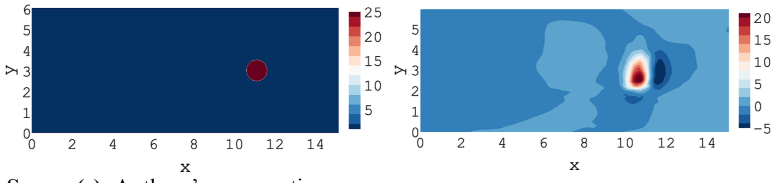
GNNs keep nodal connectivity at a central point in their formulation and learning process, as the FEM performs when making discretization from piecewise approximations. The fact of being formulated at the vertex and edge levels allows capturing the physics' locality, while *message passing* (discussed later) enables accounting for the global behaviors that PDEs represent. GNNs can extract models at a much lower scale than usual multilayer perceptrons – MLP – (artificial deep neural networks), by extracting the model of vertices (that in general corresponds to conservation balances) and edges (more concerned by the behavior laws). Thus, GNNs remain closer to the local physics scale than their NN counterparts that operate at the domain scale.

A strong and intriguing parallel can be drawn between GNNs and FEM in the use of spatial entities. FEM operates by discretizing a spatial domain into smaller patches, the so-



Source(s): Authors' own creation

Figure 9. Comparison between the reference **(left)** and computed **(right)** pollutant concentration at different time steps, from top to bottom: **t = 0.3 s, 1 s, 2.5 s** and **5 s**



Source(s): Authors' own creationz

Figure 10. The reference **(left)** and computed **(right)** source of pollution

called elements, in which PDE solutions can be effectively approximated and the differential operator discretization operated. In a similar analogy, GNNs employ nodes within the spatial domain, with each node designed to model the localized state of the system. By establishing connections among these nodes, we create a connectivity graph, which can take the form of a regular mesh or any other arbitrary network structure, mirroring the underlying system's spatial characteristics.

Additionally, the integration of input values is a pivotal step in both FEM and GNN. In FEM, these values correspond to the specification of boundary conditions and initial conditions, thereby kickstarting the numerical analysis. In GNNs, input values are attributed to multiple spatial locations, serving as initializations for the nodes within the graph.

The following step entails the evolution of the system's states. In FEM, this involves the numerical approximation of the PDE solution within each element, ultimately providing insights into the global behavior of the system. In GNNs, the nodes update their states through specialized functions, facilitating the modeling of intricate relationships and behaviors throughout the system.

By drawing parallels between these two methodologies, it becomes noticeable that while FEM is designed for solving PDEs by dividing a space into elements, GNNs employ a graph-based approach to capture and analyze the localized states of a system within a spatial domain while keeping at a central position nodal connectivity.

Since the two approaches have a lot in common, numerous studies have begun incorporating elements of FEMs into GNN (Gao *et al.*, n.d.). For instance, develop encoders and decoders that perform interpolation between random field points and graph nodes, emulating the FEM's shape function-based interpolation within elements (Alet *et al.*, 2019). Or, some recent research has innovatively applied finite-element principles to the computation of training loss. They introduced a method where the loss is calculated by integrating prediction errors across the simulation domain, using Gaussian quadrature integration with high-order shape functions, diverging from the conventional mean-squared loss calculation on nodes (Gao *et al.*, 2022).

Within this intricate framework, the present paper addresses the following question: Can GNN benefit more from all the tools employed in the classical FEM approach to address inverse problems, such as the identification of local properties within a model?

The present section is organized into three subsections. The first revisits the fundamental concepts associated with GNNs. The second proposes a novel architecture that combines GNN with elements of FEM. The third and last presents the results of applying this methodology to a heat problem case study. This section aims at proving the potential and promising capabilities of GNNs when they are enriched by integrating information from the FEM. By combining the strengths of GNN, which excels in modeling complex relationships and dependencies within graph-structured data, with the robust foundation of FEM, which is renowned for solving partial differential equations and accurately approximating physical phenomena, we aim to unlock a new frontier of problem-solving and analysis.

3.1 Revisiting graph neural networks

3.1.1 Definitions. Graph is a key concept in mathematics and computer science, essentially consisting of two primary elements: nodes (or vertices) and edges. Nodes represent entities or points, while edges connect these nodes, symbolizing the relationships or interactions between them. They are commonly denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ represents the set of edges connecting these nodes.

Graphs can be either directed or undirected, depending on whether their edges have a specific direction. In directed graphs, edges indicate a one-way relationship, whereas in undirected graphs, the relationships are bidirectional or non-directional. This distinction is critical in applications like network flow analysis or route mapping, where directionality can significantly impact the analysis or outcome.

Graphs can also be categorized based on whether their edges are weighted or unweighted. Weighted graphs assign a value or weight to each edge, which can represent quantities like distance, cost or capacity, adding a layer of complexity to the graph structure. Unweighted graphs, on the other hand, treat all connections uniformly.

The representation of graphs in computational contexts typically involves structures like adjacency matrices or adjacency lists. An adjacency matrix is a 2D array, $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, that maps the connections between nodes, while an adjacency list is a more space-efficient method, especially for sparse graphs, where each node directly points to its connected nodes.

For instance, consider the graph shown in Figure 11, composed of four nodes and four undirected edges. The corresponding adjacency matrix is as follows:

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{31}$$

Despite their apparent simplicity, graphs are incredibly versatile tools for modeling complex systems, offering a framework to abstract and solve a variety of computational problems. Their study involves addressing challenges like cycle detection, shortest path determination and exploring the most efficient ways to traverse these networks.

In the discussion that follows, we will focus only on "simple graphs," where there is at most, one edge between any pair of nodes, no edges connecting a node to itself, and all edges are undirected.

3.1.2 Graph-based learning. GNNs are deep learning algorithms that focus on learning representations for nodes and edges in a graph by utilizing the graph's inherent structure (Alessio, 2009; Sanchez-Lengeling *et al.*, 2021; Franco *et al.*, 2009). Although during the years different types of GNNs have been proposed, i.e. graph convolutional networks (GCNs) (Kipf and Welling, 2017), graph isomorphism networks (GINs) (Xu *et al.*, 2019) and graph attention networks (GATs) (Veličković *et al.*, 2018), their fundamental design feature is the use of pairwise *message passing*. In this process, graph nodes iteratively enhance their representations by exchanging information with their neighbor nodes. Message passing is formed by two key components:

(1) The "Aggregate" step: a function is applied to a node's neighbors to create an aggregated node feature.

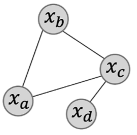
(2) The "Combine" step: the aggregated node feature is passed through a learnable layer to generate the node embedding specific to that GNN layer.

In each message-passing iteration within a GNN, the embedding h_u (associated with each node u in the set $\mathcal{V}$), is updated based on the information gathered from u's graph neighborhood $N(v)$, as depicted in Figure 12. This update can be represented as follows:

$$h_u^{(k+1)} = \texttt{COMBINE}^{(k)}\left(h_u^{(k)}, \texttt{AGGREGATE}^{(k)}\left\{h_v^{(k)}, \forall v \in N(u)\right\}\right), \tag{32}$$

where COMBINE and AGGREGATE are two differentiable functions (e.g. artificial neural networks).

In each iteration k, the AGGREGATE function takes as input the embeddings of the nodes neighbors of u and produces a message. The COMBINE function then combines this message with the current embedding $h_u^{(k)}$ to produce an update to the embedding $h_u^{(k+1)}$ of u. After completing M-iterations of the GNN message passing, the output of the final layer defines the



Source(s): Authors' own creation

Figure 11.
Graph with four nodes and four undirected edges

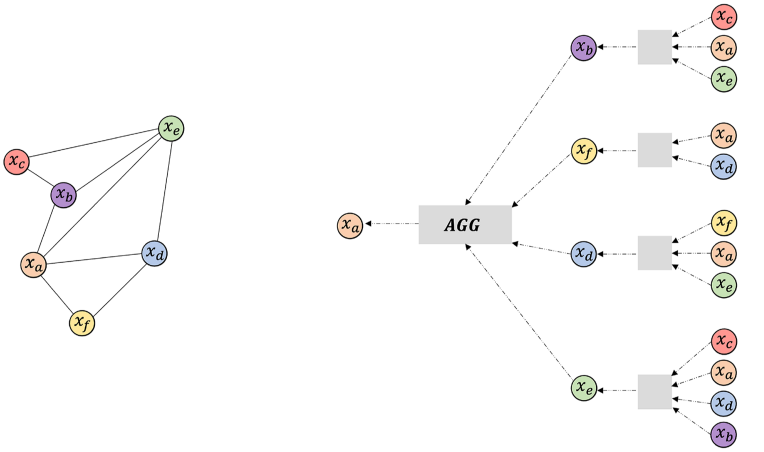


Figure 12.
Visualization of two-layer version of a message passing model

Note(s): (Left) Schematic representation of a graph where the target node to show the message passing feature is the node x_a. **(Right)** The model aggregated messages from x_a neighbors (i.e. x_b, x_d, x_e and x_f) where this messages are also aggregation of the messages from their respective neighbors
Source(s): Authors' own creation

embeddings for each node. The choice of these two functions may differ considerably based on the particular field of application. Nevertheless, the most used model, as proposed in Merkwirth and Lengauer (2005), is:

$$h_u^{(k+1)} = \sigma\left(W_1^{(k)} h_u^{(k)} + W_2^{(k)} \sum_{v \in N(u)} h_v^{(k)} \right), \tag{33}$$

where W_1 and W_2 are trainable parameter matrices, while σ represents a pointwise nonlinearity (e.g. a tanh or *ReLU*). This message passing scheme can be compared to a standard MLP, as it involves linear operations followed by a single pointwise nonlinearity. First, messages received from neighboring nodes are summed. Then, the neighborhood information is integrated with the node's previous embedding through a linear combination. Finally, an element-wise nonlinearity is applied.

However, in this approach, the choice of the sum as an aggregation function has a drawback, as it can be unstable and highly influenced by the degrees of nodes. For example, if node u has N-times as many neighbor as node u_0, we would reasonably expect that the sum of neighbor embeddings for u, i.e. $\sum_{v \in N(u)} h_v$, would be much larger than the sum for u_0, i.e. $\sum_{w \in N(u_0)} h_w$. This significant difference in magnitude can lead to numerical instabilities. A straightforward solution to this issue is to normalize the aggregation operation by considering the degrees of the involved nodes. The simplest approach is to compute an average instead of a sum.

$$\frac{\sum_{v \in N(u)} h_v}{|N(u)|}. \tag{34}$$

A more sophisticated normalization factor can be used, as the one proposed in Kipf and Welling (2017). In addition to the summation, alternative approaches incorporate element-

wise maximum or minimum operations (Qiu *et al.*, n.d.; Veličković *et al.*, 2020). Nonetheless, in Zaheer *et al.* (2017), a more complex yet versatile aggregation function approximating any permutation invariant function was proposed.

In what follows, we illustrate the importance of the aggregation function in mitigating overfitting and propose a novel aggregation function. This new function leveraging the spatial information within the underlying graph structure allows improved performance.

3.2 Learning procedure

3.2.1 Model. Our objective is to develop a predictive model capable of anticipating the state of the field at time $t+1$ based on the available current data and, optionally, a historical record of the field. To this end, we introduce a GNN model with an architecture consisting of encode-process-decode components. A visual representation of this architectural framework is provided in Figure 13.

The architecture consists of:

(1) *Encoder:* We employ a single MLP, denoted as ϵ_v, to convert the initial feature vectors of vertices, represented as $\mathbf{x}_i \in \mathbb{R}^n$, into higher-dimensional embeddings, denoted as $\mathbf{v}_i \in \mathbb{R}^v$:

$$\begin{aligned} \epsilon_v &: \mathbb{R}^n \to \mathbb{R}^v \\ \mathbf{x}_i &\to \mathbf{v}_i \end{aligned} \tag{35}$$

It should be noted that while $\mathbf{z}_t$ represents the field of the entire structure under consideration, the vector feature $\mathbf{x}_i$ pertains to nodal information. This in fact can include additional details related to the geometry or specific characteristics of certain nodes in contrast to others, such as distinguishing between boundary nodes and internal ones.

(2) *Processor:* The processor serves as the algorithm's central component, facilitating the exchange of information between vertices through message passing. It also plays a crucial role in modifying hidden vectors to extract the desired system output. At this stage, the following steps are undertaken:

- An MLP $(\widehat{\phi})$ calculates the updated edge features, denoted as $\mathbf{v}'_{ij}$ for each edge in the graph (36). This calculation takes into account the information from the

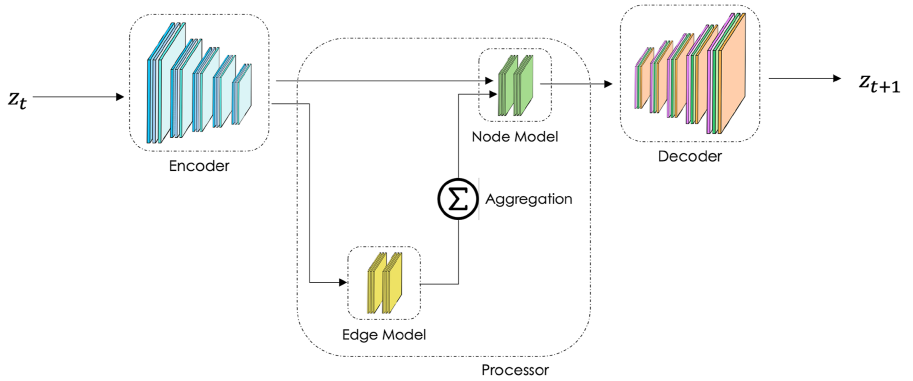


Figure 13. Schematic representation of GNN architecture

Source(s): Authors' own creation

sending ($\mathbf{v}_i$) and receiving nodes ($\mathbf{v}_j$). When available, also the current edge features and global features could be used as input to update the edge feature.

$$\mathbf{v}'_{ij} = \widehat{\phi}(\mathbf{v}_i, \mathbf{v}_j) \tag{36}$$

- For each node, the messages are aggregated with a permutation invariant function $S = S(\mathbf{v}'_{ij})$, based on the neighborhood of the node i ($N(i)$). More details on the choice of the aggregation function can be found in Section 3.2.3.
- Another MLP ($\widehat{\gamma}$) calculates the updated node features $\mathbf{v}'_i$ (37), for each node in the graph. This calculation takes into account the current state of the node ($\mathbf{v}_i$) and the result at the previous step (aggregation).

$$\mathbf{v}'_i = \widehat{\gamma}(\mathbf{v}_i, S_i) \tag{37}$$

The processing step corresponds to the exchange of messages between nodes that are 1-step adjacent. To capture the influence of more distant graph nodes, this process can be iteratively applied through N-processing blocks (Defferrard *et al.*, 2017; Hamilton *et al.*, 2017). These blocks may employ shared or unshared parameters and can optionally incorporate residual connections (He *et al.*, 2016).

(3) *Decoder:* For predicting the time $t + 1$ state from the time t input, the decoder uses an MLP(δ_v) to transform the latent node features $\mathbf{v}'_i$ after the final processing step into one or more physical output features $\mathbf{y}_i \in \mathbb{R}^n$

$$\begin{aligned} \epsilon_v &: \mathbb{R}^v \rightarrow \mathbb{R}^n \\ \mathbf{v}'_i &\rightarrow \mathbf{y}_i \end{aligned} \tag{38}$$

where $\mathbf{y}_i$ could be the updated nodal state, composing the updated global state $\mathbf{y}$ that should represent $\mathbf{z}_{t+1}$.

3.2.2 Model training. We train the final network with the mean square error loss:

$$\mathcal{L} = \|\mathbf{y} - \mathbf{z}_{t+1}\|_2^2, \tag{39}$$

where $\mathbf{y}$ is the output of our GNN and $\mathbf{z}_{t+1}$ represents the ground truth field at the next time step with respect to the input. The networks' inputs and outputs are normalized using the statistical characteristics of the training dataset. Additionally, during the training process, Gaussian noise is introduced into the inputs. The variance of this added noise is adjustable as a hyperparameter, being zero as its mean value. We adopted the Adam optimizer (Kingma and Ba, 2014) algorithm for training.

3.2.3 On the choice of the aggregation function. In GNNs, both the choice of aggregation function and the integration of attention mechanisms play pivotal roles in defining how information is processed and represented. Aggregation functions like sum, mean, max and pooling operations are essential for combining features from neighboring nodes, thereby encapsulating the local neighborhood's information and the structural essence of the graph.

Such aggregation is crucial for tasks such as node classification and graph classification. On the other hand, attention mechanisms (Gilmer *et al.*, 2017), notably GATs (Veličković *et al.*, 2018), bring a dynamic and context-dependent layer to feature aggregation. GATs use

self-attention mechanisms to assign varying degrees of importance to each neighbor's features, enhancing the expressive power of the model beyond traditional convolutional methods. This approach, further enriched with variations like multi-head attention (Ai *et al.*, 2021; Cordonnier *et al.*, n.d.) and the inclusion of edge features, has demonstrated improved performance in various graph-related tasks.

Moreover, attention mechanisms based on geometric properties focusing on spatial or structural relationships within graphs have emerged as a crucial improvement (Veličković *et al.*, 2018; Zhang *et al.*, 2018). Spatial attention, for instance, is highly relevant in applications involving physical space, such as 3D point cloud (Yue *et al.*, 2019) processing or molecular structure analysis, where the relative positions and orientations of nodes (like atoms in a molecule) are critical. Geometrically based attention methods thus offer enhanced capabilities in processing and interpreting graphs where the spatial arrangement and structural roles of nodes are key factors. However, these improvements also bring challenges like increased computational complexity and the need for rich datasets with detailed spatial or structural information.

In essence, the evolution of GNNs through sophisticated aggregation functions and various forms of attention mechanisms, including geometrically based approaches, marks a significant stride in the field. These developments have broadened the effectiveness of GNNs in interpreting complex graph-structured data, opening new possibilities in various domains where understanding the intricate relationships and structures within data is essential.

Building on the advancements in GNNs with sophisticated aggregation functions and attention mechanisms, we introduce a novel aggregation function that uniquely incorporates information derived from the assembly matrices, specifically stiffness and mass matrices, of the FEM. The essence of this integration is represented by

$$S\left(\mathbf{v}'_{ij}\right) = \frac{1}{M_{ii}^{lump}} \sum_{j \in N(i)} K_{ij}\ \mathbf{v}'_{ij}, \tag{40}$$

where the coefficients K_{ij} and M_{ii}^{lump}, respectively, are the elements of the stiffness matrix and the lumped mass matrix of the FEM counterpart.

The integration of these matrices into the GNN aggregation process enables the network to capture not only just the topological and feature-based information of the graph but also the inherent physical and mechanical properties encoded in the FEM matrices. This could enhance the model's ability to understand and predict behaviors in engineering and physical systems, where such properties are crucial, in particular in the physics of continuous media.

Integrating the assembled FEM matrices into the aggregation function of GNNs also offers a strategic advantage in simplifying the model architecture. By using stiffness and mass matrices, the architecture can effectively capture the geometric and physical information of the graph without the need to construct a separate, geometry-dependent edge model. This reduction in complexity not only simplifies the model but also potentially mitigates the risk of overfitting.

By embedding essential geometric and physical insights directly into the node aggregation process, the network remains robust and generalizable, avoiding the pitfalls of overly complex models that are too tightly tailored to specific datasets or geometries. This balance between richness of information and model simplicity could be a key factor in enhancing the performance and applicability of GNNs in fields concerned with computational engineering and mechanics.

The fact of including lumped mass and stiffness matrices can be viewed as physics-aware learning, where physical biases are employed to better learn and operate GNNs.

3.2.4 Parallelism with the FEM. To illustrate the connection between the proposed aggregation function and the FEM rationale, let us start with an illustrative example by considering the heat equation

$$\rho c_p \frac{\partial T(\mathbf{x},t)}{\partial t} - \nabla \cdot (k(\mathbf{x})\ \nabla T(\mathbf{x},t)) = S(\mathbf{x},t), \tag{41}$$

assuming for simplicity a constant conductivity ($k(\mathbf{x}) = 1$), constant density and heat capacity ($\rho = 1$, $c_p = 1$) and null source term ($S(\mathbf{x}, t) = 0$).

Thus, the algebraic FEM counterpart with a forward Euler scheme for the time discretization, reads

$$\frac{\mathbf{M}}{\Delta t}\left(\mathbf{T}^{n+1} - \mathbf{T}^{n}\right) - \mathbf{K}\mathbf{T}^{n} = \mathbf{0}, \tag{42}$$

or

$$\mathbf{T}^{n+1} = \mathbf{T}^{n} + \Delta t \mathbf{M}^{-1}\mathbf{K}\ \mathbf{T}^{n}, \tag{43}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{K}$ is the stiffness matrix and Δt is the integration time-step.

In the case of considering the lumped mass matrix, its inversion is trivial because of the fact that it is diagonal. From (43) we have that for each node of the mesh the temperature can be updated as:

$$T_i^{n+1} = T_i^{n} + \frac{\Delta t}{M_{ii}^{lum}} \sum_{j=1}^{N_d} K_{ij} T_j^{n} = T_i^{n} + \frac{\Delta t}{M_{ii}^{lum}} \sum_{j \in V(i)} K_{ij} T_j^{n}, \tag{44}$$

where N_d is the number of nodes in the mesh and $V(i) = \{N(i) \cup i\}$ is the set ($N(i)$) of all the nodes connected with i, and the node i itself. In particular, in the last equality of (44) we utilized the property that all nodes not connected to node i contain a zero in the corresponding column of the stiffness matrix.

In what follows, a GNN with the architecture described in Section 3.2 is considered. If we define an edge model such that

$$\phi_{ij} = \widehat{\phi}(T_i, T_j) = T_i - T_j, \tag{45}$$

and the node model such that

$$\widehat{\gamma}\left(T_i, S_i(\phi_{ij})\right) = T_i + \Delta t S_i(\phi_{ij}), \tag{46}$$

using (40) as aggregation function we have:

$$S_i(\phi_{ij}) = \frac{1}{M_{ii}^{lum}} \sum_{j \in N(i)} K_{ij}\phi_{ij}, \tag{47}$$

and so (46) at each iteration of the GNN becomes:

$$T_i^{n+1} = T_i^n + \frac{\Delta t}{M_{ii}^{lum}} \sum_{j \in N(i)} K_{ij} \left(T_j^n - T_i^n \right), \tag{48}$$

that is,

$$T_i^{n+1} = T_i^n + \frac{\Delta t}{M_{ii}^{lum}} \left(\sum_{j \in N(i)} K_{ij} T_j^n - \sum_{j \in N(i)} K_{ij} T_i^n \right), \tag{49}$$

that by considering now then the properties of the stiffness matrix

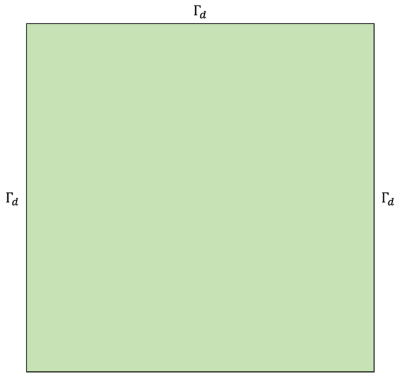
$$\forall i \sum_{j \in V(i)} K_{ij} = 0 \Rightarrow \forall i \sum_{j \in N(i)} K_{ij} = -K_{ii}, \tag{50}$$

Equation (49) becomes

$$T_i^{n+1} = T_i^n + \frac{\Delta t}{M_{ii}^{lum}} \left(\sum_{j \in N(i)} K_{ij} T_j^n + K_{ii} T_i^n \right) = T_i^n + \frac{\Delta t}{M_{ii}^{lum}} \sum_{j \in V(i)} K_{ij} T_j^n. \tag{51}$$

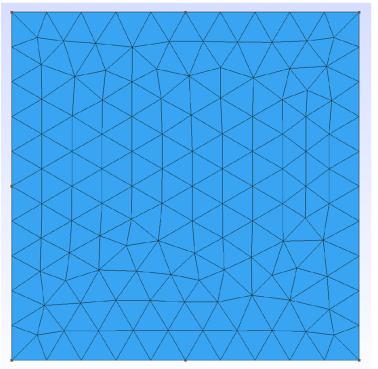
Thus, Equation (51) is found to be equivalent to Eq. (44), having exactly the same form. To demonstrate the validity of this architecture, which essentially emulates an explicit Euler scheme, we will evaluate its ability to replicate the dynamics of a system. Specifically, let's consider again the heat Equation (41) with constant conductivity ($k(\mathbf{x}) = 1$), constant density and heat capacity ($\rho = 1, c_p = 1$) and null source term ($S(\mathbf{x}, t) = 0$). For this analysis, we will focus on a square domain as illustrated in Figure 14 with a fixed temperature on each side.

The ground truth – GT – simulations are computed using the FEM using an explicit time integration. The simulations are conducted over a period of five seconds, with a discretization consisting of $N_T = 600$ time increments, each of $\Delta t = 8.3 \cdot 10^{-3}$. One



Schematic representation of the problem

(a)



Mesh used to solve the FEM problem and to define the graph for the GNN

(b)

Source(s): Authors' own creation

Figure 14.
Problem set-up used to prove the validity of the proposed methodology

hundred different cases are simulated, with varying values of temperature enforced on the domain boundary. The dataset has been divided into two sets, with 80% in the training set and 20% in the test set.

Two two-layer neural networks, each with ten neural units per layer and using rectified linear unit (ReLU) activation, are employed to define the node and edge models. The learning rate is set at $l_r = 10^{-4}$, with a decrease in order of magnitude at epochs 100 and 200, a total number of epochs set at $N_{epoch} = 200$ and a batch size of 15. The training noise variance is set to $\sigma_{noise} = 5 \cdot 10^{-4}$.

The chosen state variable is the Temperature, which defines a node feature. Additionally, a vector **n** is incorporated into the node features to identify the internal and the boundary nodes of the domain. The results for one simulation in the test set are shown in Figure 15.

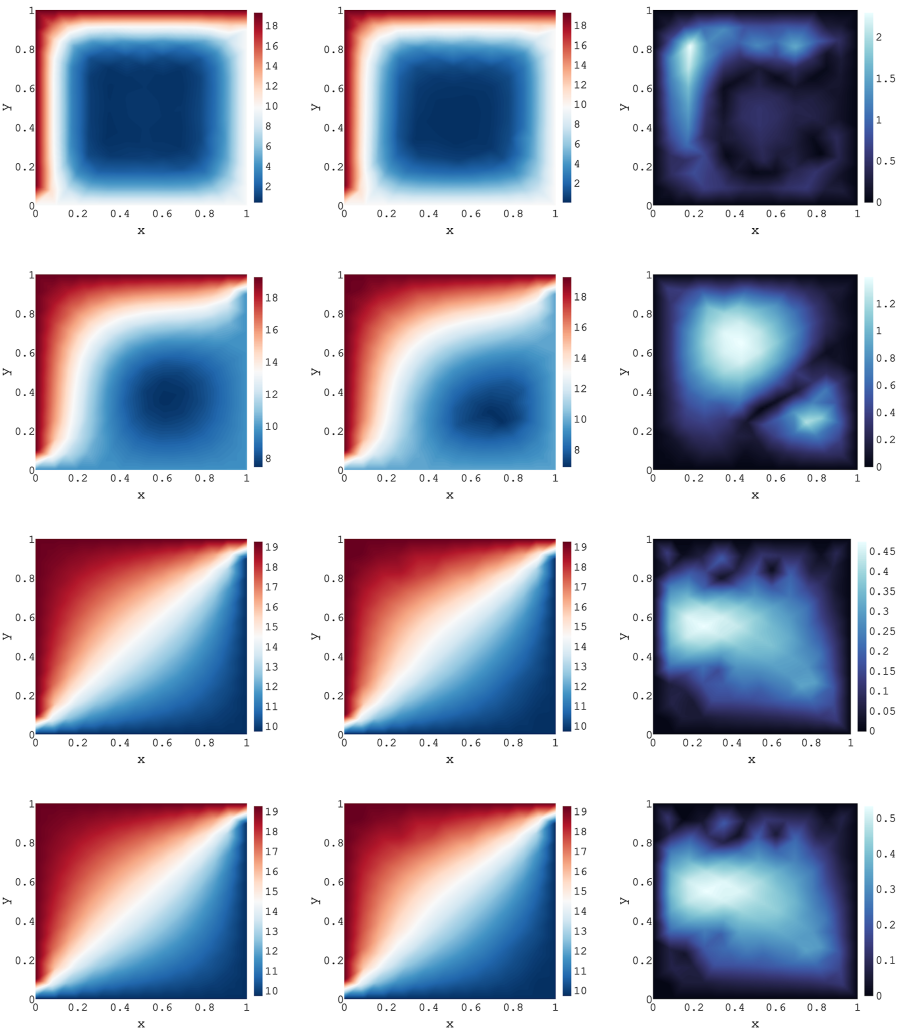


Figure 15.
Ground truth solution **(left)**, GNN-solution **(center)** and the difference in absolute value between both **(right)**

Note(s): From top to bottom, the solutions at **0.08 s**, **0.8 s**, **3.21 s**, and **4.01 s**

Source(s): Authors' own creation

As it can be noticed, there is an overall consistency between the ground truth solution and the prediction made by the GNN. In particular, the maximum relative error computed on the solution at five seconds across all the datasets remains lower than 4%.

3.3 Heat equation with unknown conductivity distribution

This section addresses the application of the methodology just outlined to identify the thermal conductivity deviations in a small region of a larger domain that represents a topic of special interest in structural health diagnosis, as considered before by applying the PINN rationale.

Here, we assume that the system under scrutiny contains regions with different thermal conductivities, and our goal is to accurately locate these regions and construct a model that can faithfully reproduce the system's thermal behavior. The model intent is thus to pinpoint variations in the structural properties, with a focus on thermal conductivity.

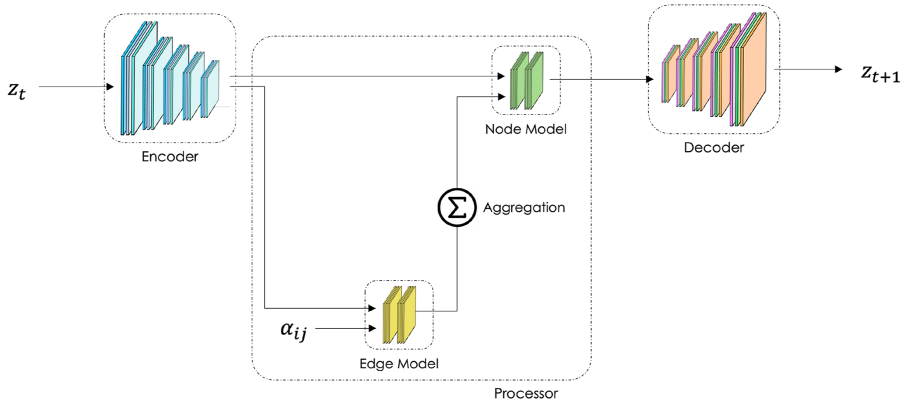
Adapting the architecture detailed in Section 3.2 is necessary for this task. We need to enhance the edge model to reflect the presence of zones where conductivity differs from the one existing in the larger part of the domain. This involves parametrizing the model, for example, with a new trainable parameter, $\widehat{\boldsymbol{\alpha}}$, whose task is to discriminate between areas that exhibit different thermal properties. This new architecture is presented in Figure 16.

Specifically, the MLP $(\widehat{\phi})$ defined in (36), which calculates the updated edge features, becomes:

$$\mathbf{v}'_{ij} = \widehat{\phi}(\mathbf{v}_i, \mathbf{v}_j, \widehat{\alpha}_{ij}), \tag{52}$$

where $\mathbf{v}_i$ is the information from the sending node, $\mathbf{v}_j$ the information of the receiving nodes and $\widehat{\alpha}_{ij}$ is a trainable parameter defined for each edge in the graph that represents the different conductivity zone.

Given our assumption that the area in which the property differs from the one existing in the remaining larger domain, that is, that the deviation remains localized, we expect that the zone where the alpha parameter differs from zero will also be localized. Consequently, we have incorporated a penalty term (Goodfellow *et al.*, 2016) into the loss function (39) that now reads



Source(s): Authors' own creation

Figure 16. Schematic representation of GNN architecture used to differentiate a zone with different conductivity in the whole structure

$$\mathcal{L} = \lambda_1 \left\| \mathbf{y} - \mathbf{z}_{t+1} \right\|_2^2 + \lambda_2 \left\| \boldsymbol{\alpha} \right\|_1, \tag{53}$$

where $\{\lambda_1, \lambda_2\}$ are the weights for the two terms of the cost function.

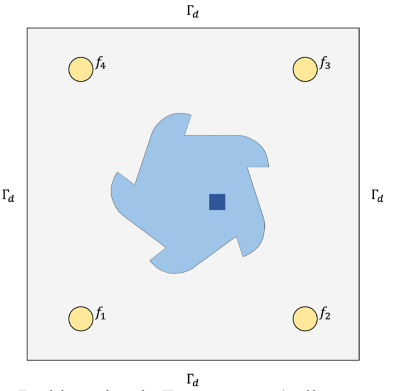
Moreover, it is important to mention that the penalty term also serves to mitigate overfitting, especially in scenarios with limited training data where non-physical correlations could appear.

3.3.1 Case study. The case study here addressed involves the analysis of the heat flow in a two-dimensional domain, as depicted in Figure 17a. The focus is to identify areas within the structure where the thermal properties deviate from the ones existing in the bulk.

The target is achieved by heating the structure with four distinct heat sources ($f_1, \ldots, f_4$) positioned as indicated in Figure 17a. The temperature changes in the domain of interest are then monitored and recorded using a thermal imaging camera. This data serves then as input into a GNN architecture, as shown in Figure 16. The GNN is trained to estimate the coefficients $\widehat{\boldsymbol{\alpha}}$, which identify the zones where there is a possible variation of the conductivity law. In order to employ the expression (40) as an aggregation function, the stiffness and the mass matrix must be computed, assuming a constant conductivity ($k = 1$), and passed to the model.

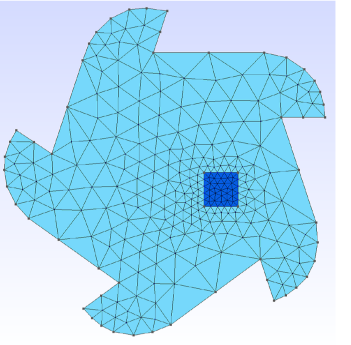
The case study here considered can be easily extended to other processes, like casting in materials processing or to cases where conductivity exhibits nonlinear behavior (e.g. temperature dependent properties).

3.3.2 Data production. The data used to train the GNN are synthetic data produced provided by a FEM solver. In particular, we solved the Equation (41) in the domain shown in Figure 17a with fixed temperature on the boundary ($T = 20°$), fixed temperature of the four sources ($f_1 = 42°, f_2 = 50°, f_3 = 45°\&f_4 = 35°$) and uniform initial condition $T(x, 0) = 20°$. Moreover, we consider that the domain of interest presents a zone with different conductivity as shown in Figure 17a (blue square). The heat conduction problem is simulated during 2.5 s from its initial condition.



Problem sketch: Four sources (yellow circles) heat the blade (domain of interest), with a small region (blue square) where the conductivity differs

(a)



Mesh considered in the FEM solution that defines the graph for the GNN

(b)

Source(s): Authors' own creation

Figure 17.
Problem set-up

Although the simulation domain considers both the domain of interest and the surrounding mold, only the temperature in the domain of interest has been extracted and used to train the GNN.

3.3.3 Results. Two two-layer neural networks with ten neural units per layer equipped with a ReLU function are used for the node and the edge models. The learning rate is set to $l_r = 10^{-5}$ with decreasing order of magnitude on epochs 60 and 100, and a total number of $N_{epoch} = 100$. The training noise variance is set to $\sigma_{noise} = 5 \cdot 10^{-4}$.

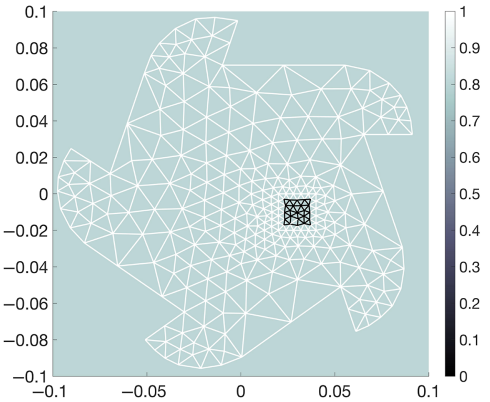
The state variable chosen is the Temperature which defines a node feature. An additional one-hot vector $\mathbf{n}$ is added to the node features in order to represent the boundary. The coefficients $\boldsymbol{\alpha}$ resulting from the solution are showed in Figure 18. It can be concluded that by using a single simulation to train for the GNN, it accurately identifies the zone where the conductivity exhibits a difference with respect to the remaining much larger domain, proving its potential applicability for inverse analyses.

We would like to emphasize that, unlike the traditional use of GNNs or most neural networks where multiple experiences (and thus a big dataset) are required to train the model, the proposed architecture does not have training and testing phases. Instead, it features only a training phase with data coming from one process (one simulation in our case), making it particularly suitable for online exploration.

4. Conclusions

This paper proposed two different neural network approaches able to learn models in order to perform efficient inverse analyses. The first approach, based on the PINN, focuses on completing the measured data and identifying the sources on which the measured data depends.

The second proposed strategy involves a GNN capable of learning from a single measure the problem solution and to identify the region where the thermal conductivity differs from that in the remaining larger region. This methodology has been tested on the case of linear behavior of the conductivity, but we are persuaded that a real advantage could be realized when it is applied to discover nonlinear conductivity in materials. Further research will be conducted in this direction.



Note(s): Specifically, the model has the ability to identify zones within the graph where the conductivity differers, thereby constructing an edge model capable of replicating the correct heat conduction behavior in both distinct zones

Source(s): Authors' own creation

Figure 18. Coefficient $\boldsymbol{\alpha}$ computed with the proposed architecture

The present work focuses its attention mainly on the development of ideas; its application is kept here for academic examples. Thus, for both presented methodologies, extensions should be carried out to be applied to cases closer to the industrial world.

References

Adel, H. (2012), "Inverse source problem in a 2D linear evolution transport equation: detection of pollution source", *Inverse Problems in Science and Engineering*, Vol. 20.3, pp. 401-421, doi: 10.1080/17415977.2011.637207.

Ai, B., Wang, Y., Ji, L., Yi, J., Wang, T., Liu, W. and Zhou, H. (2021), "A graph neural network fused with multi-head attention for text classification", *Journal of Physics: Conference Series*, Vol. 2132, p. 1, doi: 10.1088/1742-6596/2132/1/012032.

Alessio, M. (2009), "Neural network for graphs: a contextual constructive approach", *Neural Networks*, IEEE Transactions on 20, doi: 10.1109/TNN.2008.2010350.

Alet, F., Jeewajee, A.K., Bauza, M., Rodriguez, A., Lozano-Perez, T. and Kaelbling, L.P. (2019), "Graph element networks: adaptive, structured computation and memory", in Chaudhuri, K. and Salakhutdinov, R. (Eds), *Proceedings of Machine Learning Research*, Vol. 97.

Bellinger, C., Mohomed Jabbar, M.S., Zaïane, O. and Osornio-Vargas, A. (2017), "A systematic review of data mining and machine learning for air pollution epidemiology", *BMC Public Health*, Vol. 17 No. 1, p. 907, doi: 10.1186/s12889-017-4914-3.

Chen, X., Gong, Z., Zhao, X., Zhou, W. and Yao, W. (2023), "A machine learning surrogate modeling benchmark for temperature field reconstruction of heat source systems", *Science China Information Sciences*, Vol. 66, p. 5, doi: 10.1007/s11432-021-3645-4.

Cordonnier, J.-B., Loukas, A. and Jaggi, M. (n.d.), "Multi-head attention: collaborate instead of concatenate", *CoRR*, abs/2006.16362 (2020). arXiv: 2006.16362, available at: https://arxiv.org/abs/2006.16362

Defferrard, M., Bresson, X. and Vandergheynst, P. (2017), "Convolutional neural networks on graphs with fast localized spectral filtering", arXiv 1606.09375.

Franco, S., Gori, M., Ah Chung Tsoi, Hagenbuchner, M. and Monfardini, G. (2009), "The graph neural network model", *IEEE Transactions on Neural Networks*, Vol. 20 No. 1, pp. 61-80, doi: 10.1109/TNN.2008.2005605.

Gao, H., Zahr, M.J. and Wang, J.-X. (2022), "Physics-informed graph neural Galerkin networks: a unified framework for solving PDE-governed forward and inverse problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 390, 114502, doi: 10.1016/j.cma.2021.114502.

Gao, R., Deo, I. and Jaiman, R. (n.d.), *A Finite Element-Inspired Hypergraph Neural Network: Application to Fluid Dynamics Simulations*, April, 2023, doi: 10.48550/arXiv.2212.14545.

Gavrilas, S., Ursachi, C.S., Perta-Crisan, S. and Munteanu, F.-D. (2022), "Recent trends in biosensors for environmental quality monitoring", *Sensors*, Vol. 22, p. 1513, doi: 10.3390/s22041513.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E. (2017), "Neural message passing for quantum chemistry", *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press, available at: http://www.deeplearningbook.org

Ha, S. and Jeong, H. (2021), "Unraveling hidden interactions in complex systems with deep learning", *Scientific Reports*, Vol. 11 No. 1, 12804, doi: 10.1038/s41598-021-91878-w.

Hamilton, W.L. (2020), "The graph neural network model", in *Graph Representation Learning*, Springer International Publishing, doi: 10.1007/978-3-031-01588-5_5.

Hamilton, W., Ying, R. and Leskovec, J. (2017), "Inductive representation learning on large graphs" in Guyon, I. Von Luxburg, U. Bengio, S. Wallach, H. Fergus, R. Vishwanathan, S. and Garnett, R. (Eds), *Advances in Neural Information Processing Systems*, Curran Associates, Vol. 30, available at:

{https://proceedings.neurips.cc/paper_files/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf

He, K., Zhang, X., Ren, S. and Sun, J. (2016), "Deep residual learning for image recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, doi: 10.1109/CVPR.2016.90.

Hernandez, Q., Badías, A., Chinesta, F. and Cueto, E. (2022), "Thermodynamics-informed graph neural networks", *IEEE Transactions on Artificial Intelligence*, doi: 10.1109/tai.2022.3179681.

Janne, P.T. (2020), "Detection of time-varying heat sources using an analytic forward model", *Journal of Computational and Applied Mathematics*, Vol. 379, 112801, doi: 10.1016/j.cam.2020.112801.

Kingma, D. and Ba, J. (2014), "Adam: a method for stochastic optimization", *International Conference on Learning Representations*.

Kipf, T.N. and Welling, M. (2017), "Semi-supervised classification with graph convolutional networks", *International Conference on Learning Representations (ICLR)*.

Lagaris, I.E., Likas, A. and Fotiadis, D.I. (1998), "Artificial neural networks for solving ordinary and partial differential equations", *IEEE Transactions on Neural Networks*, Vol. 9 No. 5, pp. 987-1000, doi: 10.1109/72.712178.

Lorenzo, D.Di, Champaney, V., Germoso, C., Cueto, E. and Chinesta, F. (2022), "Data completion, model correction and enrichment based on sparse identification and data assimilation", *Applied Sciences*, Vol. 12, p. 7458, doi: 10.3390/app12157458.

Lorenzo, D.Di, Champaney, V., Marzin, J., Farhat, C. and Chinesta, F. (2023), "Physics informed and data-based augmented learning in structural health diagnosis", *Computer Methods in Applied Mechanicatios s and Engineering*, Vol. 414, 116186, doi: 10.1016/j.cma.2023.116186.

Mahalingam, U., Elangovan, K., Dobhal, H., Valliappa, C., Shrestha, S. and Kedam, G. (2019), "A machine learning model for air quality prediction for smart cities", *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, pp. 452-457, doi: 10.1109/WiSPNET45539.2019.9032734.

Merkwirth, C. and Lengauer, T. (2005), "Automatic generation of complementary descriptors with molecular graph networks", *Journal of Chemical Information and Modeling*, Vol. 45 No. 5, pp. 1159-1168, doi: 10.1021/ci049613b.

Oliver, W., Kohlbacher, S., Kuenemann, M., Garon, A., Ducrot, P., Seidel, T. and Langer, T. (2020), "A compact review of molecular property prediction with graph neural networks", *Drug Discovery Today: Technologies*, Vol. 37, pp. 1-12, doi: 10.1016/j.ddtec.2020.11.009.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. and Battaglia, P.W. (2020), "Learning mesh-based simulation with graph networks", *CoRR*, available at: https://arxiv.org/abs/2010.03409

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K. and Tang, J. (n.d.), "Network embedding as matrix factorization", *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, doi: 10.1145/3159652.3159706.

Raissi, M., Perdikaris, P. and George, K. (2017a), "Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations", *arXiv*.

Raissi, M., Perdikaris, P. and George, K. (2017b), "Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations", *arXiv*.

Sanchez-Lengeling, B., Reif, E. and Wiltschko, A. (2021), "A gentle introduction to graph neural networks", in *Distill*, Vol. 2021.

Tahir Bahadur, F., Shah, S.R. and Nidamanuri, R.R. (2023), "Applications of remote sensing vis-à-vis machine learning in air quality monitoring and modelling: a review", *Environmental Monitoring and Assessment*, Vol. 195 No. 12, p. 1502, doi: 10.1007/s10661-023-12001-2.

Torre, D.L., Kunze, H., Mendivil, F., Galan, M.R. and Zaki, R. (2015), "Inverse problems: theory and application to science and engineering 2015", *Mathematical Problems in Engineering*, 796094, doi: 10.1155/2015/796094.

Ullo, S.L. and Sinha, G.R. (2020), "Advances in smart environment monitoring systems using IoT and sensors", *Sensors*, Vol. 20 No. 11, p. 3113, doi: 10.3390/s20113113.

Veličković, P., Ying, R., Padovano, M., Hadsell, R. and Blundell, C. (2018), "Graph attention networks", *arXiv:* 1710.10903.

Veličković, P., Ying, R., Padovano, M., Hadsell, R., Blundell, C. (2020), "Neural execution of graph algorithms", arXiv: 1910.10593.

Wang, J., Liu, Y., Lu, H., Qi, J. and Ai, P. (2022a), "Research of industrial heat sources detection method based on thermal infrared satellite data", in Wang, L., Wu, Y. and Gong, J. (Eds), *Proceedings of the 7th China High Resolution Earth Observation Conference (CHREOC 2020)*, CHREOC 2020, Lecture Notes in Electrical Engineering, Springer, Singapore, Vol. 757, doi: 10.1007/978-981-16-5735-1_12.

Wang, S., Yu, X. and Perdikaris, P. (2022b), "When and why PINNs fail to train: a neural tangent kernel perspective", *Journal of Computational Physics*, Vol. 449, 110768, ISSN: 0021-9991, doi: 10.1016/j.jcp.2021.110768.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S. (2020), "A comprehensive survey on graph neural networks", *IEEE Transactions on Neural Networks and Learning Systems*.

Wu, Z., Wang, J., Du, H., Jiang, D., Kang, Y., Li, D., Pan, P., Deng, Y., Cao, D., Hsieh, C.Y. and Hou, T. (2023), "Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking", *Nature Communications*, Vol. 14 No. 1, p. 2585, doi: 10.1038/s41467-023-38192-3.

Xu, K., Hu, W., Leskovec, J. and Jegelka, S. (2019), "How powerful are graph neural networks?", *International Conference on Learning Representations (ICLR)*.

Yue, W., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M. and Solomon, J.M. (2019), "Dynamic graph CNN for learning on point clouds", *ACM Transactions on Graphics*, Vol. 38 No. 5, pp. 1-12, doi: 10.1145/3326362.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. and Smola, A. (2017), "Deep sets", available at: https://api.semanticscholar.org/CorpusID:4870287

Zhang, Yi, Zhu, Y., Zeng, Z., Zeng, G., Xiao, R., Wang, Y., Hu, Y., Tang, L. and Feng, C. (2021), "Sensors for the environmental pollutant detection: are we already there?", *Coordination Chemistry Reviews*, Vol. 431, 213681, ISSN: 0010-8545, doi: 10.1016/j.ccr.2020.213681.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I. and Yeung, D.-Y. (2018), "GaAN: gated attention networks for learning on large and spatiotemporal graphs".

Zhang, X.-L., Xiao, H., Luo, X. and He, G. (2022), "Ensemble Kalman method for learning turbulence models from indirect observation data", *Journal of Fluid Mechanics*, Vol. 949, p. A26, doi: 10.1017/jfm.2022.744.

Zhang, X.-L., Xiao, H., Luo, X. and He, G. (2023), "Combining direct and indirect sparse data for learning generalizable turbulence models", *Journal of Computational Physics*, Vol. 489, 112272, ISSN: 0021-9991, doi: 10.1016/j.jcp.2023.112272.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C. and Sun, M. (2018), "Graph neural networks: a review of methods and applications", *arXiv Preprint, arXiv:1812.08434*.

Further reading

Abadi, M., Barham, P., Chen, J., Chen, Z. Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G. Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016), TensorFlow: a system for large-scale machine learning, *arXiv*, 1605.08695, available at: https://arxiv.org/abs/1605.08695

Kralovec, C. and Schagerl, M. (2020), "Review of structural health monitoring methods regarding a multi-sensor approach for damage assessment of metal and composite structures", *Sensors*, Vol. 20, p. 3, doi: 10.3390/s20030826.

Corresponding author
Daniele Di Lorenzo can be contacted at: daniele.di_lorenzo@ensam.eu