Computer Methods in Applied Mechanics and Engineering 429 (2024) 117102

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Physics-informed MeshGraphNets (PI-MGNs): Neural finite element solvers for non-stationary and nonlinear simulations on arbitrary meshes

Tobias Würth [a,*], Niklas Freymuth [b], Clemens Zimmerling [a], Gerhard Neumann [b], Luise Kärger [a]

[a] Karlsruhe Institute of Technology (KIT), Institute of Vehicle System Technology, Karlsruhe, Germany
[b] Karlsruhe Institute of Technology (KIT), Autonomous Learning Robots, Karlsruhe, Germany

ARTICLE INFO

Keywords:
Graph neural network
Machine learning
Physics-based simulation
Surrogate model
Partial differential equations

ABSTRACT

Engineering components must meet increasing technological demands in ever shorter development cycles. To face these challenges, a holistic approach is essential that allows for the concurrent development of part design, material system and manufacturing process. Current approaches employ numerical simulations, which quickly becomes computation-intensive, especially for iterative optimization. Data-driven machine learning methods can be used to replace time- and resource-intensive numerical simulations. In particular, MeshGraphNets (MGNs) have shown promising results. They enable fast and accurate predictions on unseen mesh geometries while being fully differentiable for optimization. However, these models rely on large amounts of expensive training data, such as numerical simulations. Physics-informed neural networks (PINNs) offer an opportunity to train neural networks with partial differential equations instead of labeled data, but have not yet been extended to handle time-dependent simulations of arbitrary meshes. This work introduces PI-MGNs, a hybrid approach that combines PINNs and MGNs to quickly and accurately solve non-stationary and nonlinear partial differential equations (PDEs) on arbitrary meshes. The method is exemplified for thermal process simulations of unseen parts with inhomogeneous material distribution. Further results show that the model scales well to large and complex meshes after being trained exclusively on small, generic meshes.

1. Introduction

As technology advances and requirements grow, the process of developing technical systems becomes increasingly complex. This is further complicated by shorter development times and a increasing shortage of resources. In particular, components, materials and manufacturing processes must be developed jointly to achieve the technical demands. In order to handle this challenge in a reasonable amount of time, engineers tend towards computer-aided engineering (CAE). Most notably, physics-based simulations have become indispensable in the development of complex technical systems.

Simulations are in general built up on partial differential equations (PDEs) that describe the considered technical system. Solving PDEs generally requires numerical approximation methods because analytical solutions are not tractable for arbitrary geometries. Therefore, numerical methods such as the finite element method (FEM) [1], the finite volume method (FVM) [2] or the finite

* Corresponding author.
E-mail address: tobias.wuerth@kit.edu (T. Würth).

https://doi.org/10.1016/j.cma.2024.117102
Received 16 February 2024; Received in revised form 25 April 2024; Accepted 27 May 2024
Available online 15 June 2024

difference method (FDM) [3] are frequently applied [4]. These simulations can take hours or even days for complex systems and sufficiently realistic solutions. In practice, the computation time even multiplies for parametric studies or iterative optimization [5].

To overcome this limitation, surrogate models can be applied. Surrogates are numerically efficient approximations of the "input–output"-relation of an expensive simulation. They are constructed based on a set of a-priori sampled observations. Recently, machine learning (ML) methods have gained attention as surrogate models, because they enable huge speed-ups while being fully differentiable and achieving good accuracy. This allows for replacing numerical solvers with ML-based surrogate models, but their reliability and applicability to unseen scenarios, i.e., scenarios that were not accessible during model training, is highly dependent on the model architecture. Many existing models are trained on a specific part and cannot consider new arbitrary part designs, e.g. [6,7]. However, this contradicts many use cases because the part design and its manufacturing process strongly influences the part's function and consequently its quality. This makes it crucial for a learned surrogate model to generalize to arbitrary unseen geometries of complex material systems for changing process conditions. Some approaches seek to replace geometry parameters with pixels- or voxels and analyze them with convolutional neural networks (CNNs) [5,8,9]. This substantially boosts the space of processable geometries but requires to transform input and output to conform to regular grids, which restricts their applicability. In classical numerics, so-called 'meshes' have proven a universal and machine-readable description of geometries/computational domain. Thus, new approaches have been developed, e.g. by graph neural networks (GNNs) [10,11] or transformers [12], to tackle this issue and naturally handle simulation data based on arbitrary meshes.

In addition to handling part geometries, neural PDE solver can also be distinguished in terms of time. For example, models can be categorized into next-step models and time-dependent neural operator methods [13]. Next-step models predict the solution iteratively, using the solutions of previous time steps as input. Some next-step models output the solution directly [10,13,14], while others, such as NeuralODEs, model the time derivative continuously and use numerical solvers to predict the next time step [15,16]. In contrast, time-dependent neural operators learn to predict a solution given the initial condition and its time step. This allows for fast approximation of a solution at future time steps with only one model call [17,18], while limiting the applicability of the model for out-of-distribution time steps. Similar to classical explicit time integration, next-step models can be applied at any starting point and for any simulation duration, making them a flexible option for engineering tasks. However, next-step models depend on their previous output and thus struggle with error accumulation. To overcome this limitation, several strategies have been deployed, such as training noise injection [10,14] or temporal bundling [13].

While mesh-based ML-techniques show great potential to model arbitrary geometries, their current applications are still data-driven, and as such, they require vast amounts of training data. Here, physics-informed training is quickly becoming a popular alternative [19–21]. Instead of learning the system dynamics from (sufficiently many) supplied samples, physics-informed training seeks to train a model directly on the governing physical equations – usually cast as PDEs. The obtained surrogate models generalize well to unseen PDE parameters [19–21]. However, early studies on physics-informed neural networks (PINNs) [18] use feed-forward neural networks and consider the position on the part as an additional network input, which however restricts the model to learn part design-dependent solutions. To alleviate this limitation, several works have transferred the physics-informed training idea to model architectures that perform better on unseen arbitrary part designs, e.g. based on CNNs [22–24], PointNets [25] and GNNs [26,27]. Kashefi et al. [25] developed a physics-informed training of PointNets based on automatic differentiation and for stationary problems, such as incompressible flows and thermal fields, which was able to handle unseen arbitrary geometries. Gao et al. [22] trains CNNs with FDM on 2D irregular domains by mapping the physical to a regular domain. However, solving complex geometries with the FDM is challenging, because, e.g., the required domain transformation to regular domains proves to be tedious [4]. Therefore, other numerical methods, such as the FEM, are usually used for complex engineering parts [4]. Many simulation models have been developed and successfully validated with the FEM. This makes the integration of the FEM into physics-informed training promising and necessary for simulating many real-world problems. Li et al. [27] propose a hybrid approach of a data-driven loss and an isogeometric analysis-based physics-informed loss to train GNN-submodels for local approximation, which are input to a purely data-driven GNNs assembly model. Gao et al. [26] train Chebyshev GNNs [28] with an FEM-based loss function to solve stationary forward and inverse problems. However, the physics-informed training of neural PDE solvers, such as MeshGraphNets (MGNs), requires non-stationary loss formulations, e.g., based on non-stationary FEM PDEs, to obtain a loss value for each time step. In addition, the GNNs are only trained on a single given geometry and have not been evaluated on new meshes.

The present work introduces physics-informed MeshGraphNets (PI-MGNs), a physics-informed framework for training MGNs to reliably simulate non-stationary physics of unseen arbitrary part designs, material property fields and process conditions. MeshGraphNets are incorporated into the well-known and widely used FEM to surpass existing PINN variants. Unlike prior work, this work trains MGNs with non-stationary and nonlinear FEM PDEs, which omits time-consuming data generation and even improves generalization and accuracy. Further enhancements are achieved by extending MGNs with additional methods such as time bundling [13] and global features [29].

To evaluate the PI-MGNs, the effectiveness of the proposed approach is demonstrated for three different time-dependent experiments with varying task complexity and learning difficulty. The present work initially compares PI-MGNs with data-driven MGNs [10] and physics-informed graph neural Galerkin networks (PI-GGNs) [26] and evaluates their improvements over the existing baselines. The subsequent evaluation illustrates the improvement of the trained PI-MGNs models over the baselines in predicting non-stationary physics on large unseen parts without retraining. In addition, an ablation study of the MGNs architecture reveals the contribution of the selected components to achieve reliable performance across all considered tasks.

2. Methodology

2.1. Simulating non-stationary physics

Non-stationary physical phenomena of complex physical systems are usually modeled by time-dependent and generally nonlinear PDEs. Simulating the physics consists of solving these PDEs to obtain the time evolution of the physical quantities of the system. The underlying PDEs differ from application to application, e.g. depending on the physical system and the research objectives. This work considers parabolic PDEs, described by

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T + q(T),\ \forall \mathbf{x} \in \Omega \tag{1}$$

where $T = T(\mathbf{x}, t)$ denotes a scalar and time-dependent field of a physical quantity, such as the part temperature in this work, $\alpha = \alpha(\mathbf{x}, t)$ the diffusivity of the material, e.g. the thermal diffusivity, and $q(T)$ an in general nonlinear source term.

An initial–boundary value problem additionally includes boundary conditions on the boundary $\partial\Omega$, e.g. Dirichlet boundary conditions $T = \overline{T},\ \forall \mathbf{x} \in \Gamma \subset \partial\Omega$ or Neumann boundary conditions $\nabla T \cdot \mathbf{n} = h_{\mathcal{N}},\ \forall \mathbf{x} \in \partial\Omega_{\mathcal{N}} = \partial\Omega \setminus \Gamma$, where $\mathbf{n}$ denotes the normal vector on the Neumann boundary $\partial\Omega_{\mathcal{N}}$, and an initial condition $T(\mathbf{x}, t = 0) = T^0$. The solution T of the initial–boundary value problem can be calculated with the FEM, which will be briefly revisited with the pertinent equations as necessary to outline the concept of PI-MGNs in the following.

2.2. The finite element method

In the FEM, the weak formulation of the PDEs is considered, which can be obtained by multiplying the strong PDE of Eq. (1) with a test function $\varphi \in H_0^1$ and integration by parts [1] as

$$\int_\Omega \frac{\partial T}{\partial t}\varphi + \alpha \nabla T \cdot \nabla \varphi - q(T)\varphi \,\mathrm{d}V - \int_{\partial\Omega_{\mathcal{N}}} \alpha h_{\mathcal{N}} \varphi \,\mathrm{d}A = 0. \tag{2}$$

More generally, the left side of the equation can be summarized as a functional $\boldsymbol{\Phi}$, which yields

$$\boldsymbol{\Phi}\left(\frac{\partial T}{\partial t}, T, \varphi\right) = 0. \tag{3}$$

Here, H_0^1 denotes a Sobolev space in which the test function $\varphi = 0$ on the Dirichlet boundary Γ and the test function φ as well as its first weak derivative are square integrable [1]. The Galerkin FEM uses the same basis functions $\phi_v(\mathbf{x})$, e.g. piecewise linear polynomials in this work, to approximate the solution $T(\mathbf{x}, t) = \sum_{v=1}^{N_\varphi} T_v(t)\phi_v(\mathbf{x})$ and the test function $\varphi(\mathbf{x}) = \sum_{v=1}^{N_\varphi} \phi_v(\mathbf{x})$, i.e.,

$$\sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}\left(\frac{\partial T_v}{\partial t}, T_v, \phi_v, \phi_m\right) = 0,\ \forall m = 1, \dots, N_\varphi.$$

Then, the weak form is discretized in time. This work uses an implicit Euler method with a constant time step Δt as

$$\begin{aligned}\sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}\left(\frac{\partial T_v}{\partial t}, T_v, \phi_v, \phi_m\right) &\approx \sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}\left((T_v^{n+1} - T_v^n)/\Delta t, T_v^{n+1}, \phi_v, \phi_m\right) \\ &\approx \sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}\left(T_v^{n+1}, T_v^n, \phi_v, \phi_m\right),\end{aligned}$$

resulting in N_t equations for the solution $T_v^n \in \{T_v^0, \dots, T_v^{N_t}\}$ and the time steps $t^n \in \{t^0, \dots, t^{N_t}\}$ with $n = 0, \dots, N_t$. Subsequently, the integrals of the functional $\boldsymbol{\Phi}$ over the domain Ω are separated into a sum of element integrals

$$\sum_{e=1}^{N_e} \sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}_e\left(T_v^{n+1}, T_v^n, \phi_v, \phi_m\right) = 0,\ \forall m = 1, \dots, N_\varphi \tag{4}$$

and the element integrals of the element functionals $\boldsymbol{\Phi}_e$ are approximated as a second-order accurate quadrature in this work.

Finally, the discretized solution T_v^{n+1} can be obtained by starting with the initial condition T_v^0 and iteratively solving Eq. (4) for the next time step T_v^{n+1} for all time steps $t^n \in \{t^0, \dots, t^{N_t}\}$. The applied backward Euler time discretization provides stability for large time steps [1]. This is necessary for fast simulations of stiff PDEs, where the solution changes strongly at different time scales and explicit methods are in consequence only stable for small time steps [1]. However, solving for one time step of nonlinear PDEs with an implicit numerical solver requires multiple inner solver iterations, which can be very time-consuming [1]. Consequently, stiff and nonlinear PDEs are particularly difficult to solve numerically and would benefit in particular from a more efficient solution approach.

Applying a given Dirichlet boundary condition simplifies the problem by pre-determining the coefficients $T_{v,\Gamma}^{n+1} \subset T_v^{n+1}$ associated with the basis functions $\phi_v(\mathbf{x})$ that are 1 at the Dirichlet boundary $x \in \Gamma$. This simplification reduces the number of unknown coefficients from N_φ to $N_{\varphi,\mathcal{F}}$. To prevent an overdetermined system of PDEs, Eq. (4) is solved only for the $m = 1, \dots, N_{\varphi,\mathcal{F}}$ test functions $\phi_{m,\mathcal{F}} \in \{\phi_m(\mathbf{x}) : \phi_m(\mathbf{x}) \neq 1, \forall \mathbf{x} \in \Gamma\}$, which are not 1 on the Dirichlet boundary Γ. Then, it is sufficient to predict the solution $T_{v,\mathcal{F}}^{n+1} \subset T_v^{n+1}$ of the nodes that are not on the Dirichlet boundary Γ. Finally, the solution is obtained by enforcing the

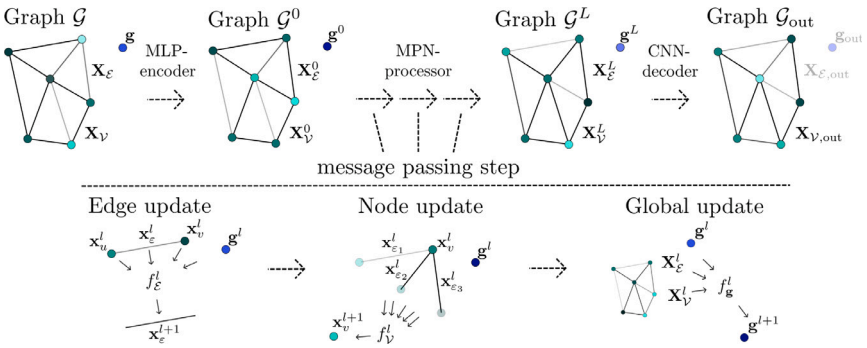

Fig. 1. Overview of the PI-MGN architecture. For a given graph $\mathcal{G}$, the MGNs encode its node features $\mathbf{X}_\mathcal{V}$, edge features $\mathbf{X}_\mathcal{E}$, and global features $\mathbf{g}$ using a separate MLP for each feature. The encoded graph is updated by multiple message passing steps in the MPN-processor. Each of the L message passing steps calculates updates for each individual node feature $\mathbf{x}_v^l$, edge feature $\mathbf{x}_\varepsilon^l$, and global feature $\mathbf{g}^l$. In general, a node, edge, and global CNN outputs a decoded graph $\mathcal{G}_{\text{out}}$. However, this work only decodes the node features, as edge and global predictions are not considered.

solution on the Dirichlet boundary as $T_v^{n+1} = T_{v,\mathcal{F}}^{n+1} + T_{v,\Gamma}^{n+1}$, where the Dirichlet solution $T_{v,\Gamma}^{n+1}$ is non-zero on the nodes of the Dirichlet boundary Γ and the non-Dirichlet solution $T_{v,\mathcal{F}}^{n+1}$ is non-zero on the remaining domain $\Omega \setminus \Gamma$. A detailed description of the non-stationary FEM can be found in the literature [1].

2.3. MeshGraphNets

The solving process of the FEM can be accelerated by learned ML simulators. Here, an MGN based on a message passing network (MPN) with global features is applied (cf. Fig. 1).

The MPN processes graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}_\mathcal{V}, \mathbf{X}_\mathcal{E}, \mathbf{g})$ by updating the node features $\mathbf{X}_\mathcal{V}$ and edge features $\mathbf{X}_\mathcal{E}$ of all nodes $v \in \mathcal{V}$ and edges $\varepsilon \in \mathcal{E}$, as well as the global features $\mathbf{g}$ in L consecutive message passing steps. The lth message passing step for updating the node feature $\mathbf{x}_v^l$, edge feature $\mathbf{x}_\varepsilon^l$ and global feature $\mathbf{g}^l$ is defined as

$$\begin{aligned}
\mathbf{x}_\varepsilon^{l+1} &= f_\mathcal{E}^l(\mathbf{x}_v^l, \mathbf{x}_u^l, \mathbf{x}_\varepsilon^l, \mathbf{g}^l), \text{ with } \varepsilon = (u,v) \in \mathcal{E} \text{ and } u, v \in \mathcal{V}, \\
\mathbf{x}_v^{l+1} &= f_\mathcal{V}^l(\mathbf{x}_v^l, \bigoplus_{\varepsilon=(v,u)\in\mathcal{E}} \mathbf{x}_\varepsilon^{l+1}, \mathbf{g}^l), \quad \text{and} \\
\mathbf{g}^{l+1} &= f_\mathbf{g}^l(\bigoplus_{v\in\mathcal{V}} \mathbf{x}_v^{l+1}, \bigoplus_{\varepsilon\in\mathcal{E}} \mathbf{x}_\varepsilon^{l+1}, \mathbf{g}^l).
\end{aligned} \tag{5}$$

The nonlinear learned functions f^l are usually implemented as MLPs, and $\bigoplus$ denotes a permutation-invariant aggregation, such as a sum, mean or max operator [10,29,30].

The MPN outputs learned quantities x_v^L per node. These values are then combined with 1D CNNs to reduce error accumulation by predicting a solution over multiple time steps, which is known as temporal bundling [13]. In particular, each time step t^n of a problem p induces a directed graph $\mathcal{G}^n$ with self-loops that contains PDE-specific node features $\mathbf{X}_\mathcal{V}^n$, edge features $\mathbf{X}_\mathcal{E}^n$ and global features $\mathbf{g}^n$. The nodes v^n of the graph $\mathcal{G}^n$ correspond to mesh nodes at time step t^n and the edges ε^n to the edges of the mesh. To predict the temporal bundle $\tilde{T}_v^{n+\tilde{n}}$ of solutions for the next $\tilde{n} = 1, \dots, N_{\text{TB}}$ time steps, the features of the graph are encoded by individual MLPs, and then subsequently processed by the MPN and the CNN decoder. The resulting bundle can be used to assemble the next graph $\mathcal{G}^{n+N_{\text{TB}}}$ to be processed by the MGN. This procedure is repeated until the desired time step t^{N_t} is reached. This methodology can be applied to batched graphs $\mathcal{G}_{\text{b}}^n$ to better utilize accelerated hardware.

In this work, a one-hot vector n_{t} [10] is added to the graph as a node feature, which encodes the node type, categorized into Dirichlet nodes, Neumann nodes and inner nodes. The relative distance vector between the nodes of the edge $\mathbf{x}_{vu} = \mathbf{x}_v - \mathbf{x}_u$ as well as the Euclidean distance $|\mathbf{x}_{vu}|$ are added as edge features. In addition, the difference of the solution $T_{vu} = T_v^n - T_u^n$, more precisely the approximation $\tilde{T}_{vu} = \tilde{T}_v^n - \tilde{T}_u^n$ at the current time step t^n between the nodes of an edge $\varepsilon = (u, v)$ are added. Additional node, edge or global features may be added individually in the specific experiments. An overview is given in the Appendix in Table A.4.

2.4. *Physics-informed* MeshGraphNets

MGNs as described in Section 2.3 are commonly trained in a data-driven fashion to learn to solve Eq. (4) for unseen tasks, e.g new part designs. This training paradigm requires simulation data that is created using numerical solvers for multiple training tasks, which often proves time- and resource-intensive. Therefore, the present work trains MGNs with a physics-informed approach that builds on the FEM formulation of Section 2.2, learning to predict the non-stationary physics for unseen part designs, materials and process settings.

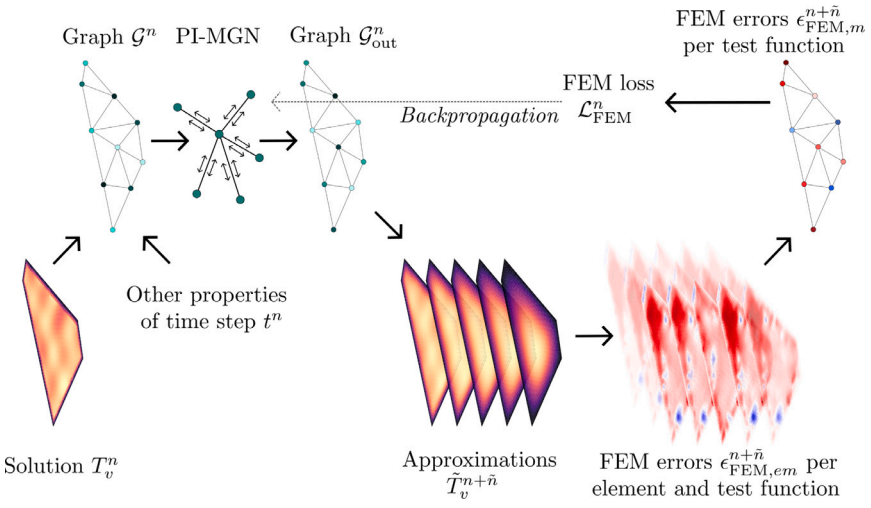

Fig. 2. Overview of the PI-MGN training. Given a solution T^n_v and additional properties of the time step t^n, a graph G^n is assembled and passed to the PI-MGNs, which output a graph G^n_{out} (cf. Fig. 1). This graph contains the approximations $\tilde{T}^{n+\tilde{n}}_v$ of the next time steps $t^{n+\tilde{n}}$, where $\tilde{n} = 1, \dots, N_{\text{TB}}$. Using the FEM, an element- and test function-wise error is calculated and summed up to a test function-wise error. Finally, the loss is calculated as the MSE of all errors to update the PI-MGN using backpropagation.

FEM-based loss. A correct solution of Eq. (4) yields zero on the left side of the equation for all $N_{\varphi,\mathcal{F}}$ equations, and a non-zero value otherwise. Similar to the FEM error formulation for static PDEs in [26], this discretized equation can be used to define an error for a time step of the non-stationary PDEs. Given a solution of the current time step T^n_v, the left side of Eq. (4) can be interpreted as an error function for the approximation $\tilde{T}^{n+1}_v$ of the MGN at the time step $n+1$. For the mth test function, the error $\epsilon^{n+1}_{\text{FEM},m}$ is thus given as

$$\epsilon^{n+1}_{\text{FEM},m} = \sum_{e=1}^{N_{\text{e}}} \sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}_e \left(\tilde{T}^{n+1}_v, T^n_v, \phi_v, \phi_m \right), \ \forall m = 1, \dots, N_{\varphi,\mathcal{F}}. \tag{6}$$

Naively evaluating this formulation requires the calculation of an element functional $\boldsymbol{\Phi}_e$ in N_{e} elements for all $N_{\varphi,\mathcal{F}}$ test functions at each time step n, which is quite expensive. However, the complexity of the calculation can be simplified. The piecewise polynomial test functions ϕ_m in an element integral of the element functional $\boldsymbol{\Phi}_e$ are only non-zero for the elements they are connected to and, consequently, the element integral is only non-zero for these test functions. This simplification allows for splitting up the calculation of the error per test function $\epsilon^{n+1}_{\text{FEM},m}$ in two steps. First, an element-wise error $\epsilon^{n+1}_{\text{FEM},em}$ is calculated for each of the test functions ϕ_m, which are non-zero in the element, otherwise the error is set to zero. Formally, this amounts to,

$$\epsilon^{n+1}_{\text{FEM},em} = \begin{cases} \sum_{v=1}^{N_\varphi} \boldsymbol{\Phi}_e \left(\tilde{T}^{n+1}_v, T^n_v, \phi_v, \phi_m \right), & \phi_m \neq 0 \\ 0, & \phi_m = 0 \end{cases} \tag{7}$$

for all $m = 1, \dots, N_{\varphi,\mathcal{F}}$ and $e = 1, \dots, N_{\text{e}}$.

The total error $\epsilon^{n+1}_{\text{FEM},m}$ of the $N_{\varphi,\mathcal{F}}$ test functions at the time step n yields

$$\epsilon^{n+1}_{\text{FEM},m} = \sum_{e=1}^{N_{\text{e}}} \epsilon^{n+1}_{\text{FEM},em}, \ \forall m = 1, \dots, N_{\varphi,\mathcal{F}}. \tag{8}$$

Finally, the loss is defined as the mean squared error (MSE) of this error

$$\mathcal{L}^n_{\text{FEM}} = \frac{1}{N_{\text{TB}} N_{\varphi,\mathcal{F}}} \sum_{\tilde{n}=1}^{N_{\text{TB}}} \sum_{m=1}^{N_{\varphi,\mathcal{F}}} \left(\epsilon^{n+\tilde{n}}_{\text{FEM},m} \right)^2, \tag{9}$$

for all time steps $\tilde{n} = 1, \dots, N_{\text{TB}}$ of the temporal bundle, which enables a physics-informed training of the MGN.

PI-MGNs training. The physics-informed training is divided into multiple epochs, with each epoch containing several optimization loops with multiple optimization steps. One optimization step (cf. Fig. 2) consists of: The PI-MGN processes a graph $\mathcal{G}^n$ and outputs the solution $\tilde{T}^{n+\tilde{n}}_v$ for the next temporal bundle, containing the solutions of the next $\tilde{n} = 1, \dots, N_{\text{TB}}$ time steps. The element-wise errors for the N_{TB} time steps are evaluated in parallel with Eq. (7), using the initial solution $T^{(n)}_v$ and the approximations $\tilde{T}^{n+\tilde{n}}_v$. Then, the error $\epsilon^{n+1}_{\text{FEM},m}$ per test function is calculated with Eq. (8). For each of the N_{TB} time steps, $N_{\varphi,\mathcal{F}}$ errors are obtained, resulting in a total amount of $N_{\text{TB}} N_{\varphi,\mathcal{F}}$ errors for the $N_{\text{TB}} N_{\varphi,\mathcal{F}}$ approximations of the PI-MGNs. The FEM loss $\mathcal{L}^n_{\text{FEM}}$ for the graph input $\mathcal{G}^n$ is calculated

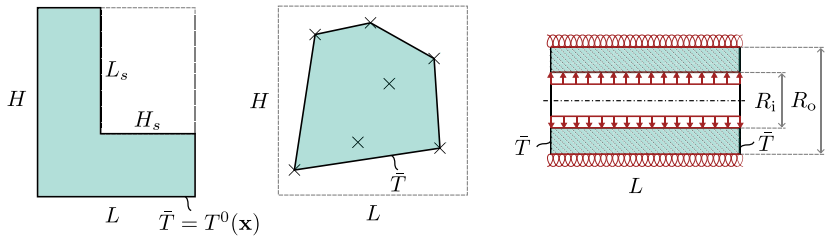

Fig. 3. Schematic overview of the experiments. Figure (a) shows an example of the L-shaped domain of a 2D linear heat diffusion experiment and Figure (b) the initialization of an convex polygon with 7 random points of the nonlinear experiment. Figure (c) depicts a hollow cylinder of the 3D experiment with a heating boundary condition inside the cylinder (red arrows) and an adiabatic boundary condition at the outer surface (red coils).

as the MSE of $N_{\text{TB}} N_{\varphi,F}$ all errors. Finally, the loss $\mathcal{L}^n_{\text{FEM}}$ is used to update the PI-MGN parameters using backpropagation through the FEM equations and a gradient-based optimizer [31].

One optimization loop starts with the initial condition at t^0 of a problem and assembles a graph $\mathcal{G}^0$ to update the PI-MGN. After this step, the time step $t^{N_{\text{TB}}}$ can be initialized to perform another optimization step. This procedure is repeated until the last time step $t^{N_{\text{T}}}$ is reached. Then, the optimization loop processes the next problem. The epoch is finished when the optimization loop has been repeated for all considered training problems p. The full training procedure consists of a given number of epochs.

It is important to note that the method is fully applicable to nonlinear equations without restrictions. The loss can also be calculated directly and in parallel for multiple time steps without the need for inner solving iterations. Additionally, since the FEM naturally integrates the Neumann boundary conditions into the PDEs and the Dirichlet boundary conditions are strictly enforced in this work, the initial–boundary-value problem has only to be solved for Eq. (9). This directly avoids competing training losses and the resulting training difficulties [32].

Noise injection in physics-informed training. As the output of an MGN at the time step n contains errors and is used to construct the next input graph $\mathcal{G}^{n+N_{\text{TB}}}$, next-step models struggle with error accumulation. Noise injection proves to be an effective strategy for MGN [10] to reduce this error accumulation. The basic idea is to add an artificial error to the input graph during training so that the MGN must detect and minimize the error. Here, noise injection is introduced for physics-informed training. Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I}\sigma)$ is added to the exact PI-MGN prediction $\tilde{T}^n_v$ to construct the input graphs during training, i.e., the graphs contain the noisy prediction $\tilde{T}^n_v + \epsilon_v$ instead of the exact prediction $\tilde{T}^n_v$. However, the noiseless approximation $\tilde{T}^{n-1}_v$ is still inserted into Eq. (9), forcing the PI-MGN to learn to output the correct solution that minimizes Eq. (9), while seeing a noisy, i.e., slightly incorrect, input graph. As in [10], the noise is only added during training and not at inference.

3. Experiments

3.1. Training on small meshes

The PI-MGNs are evaluated on three different experiments (cf. Fig. 3), where the models face different numerical and learning challenges:

- Firstly, a 2D linear heat diffusion task on randomly varying L-shaped domains is considered. The model learns to handle unseen meshed geometries and different process conditions in the form of random inhomogeneous initial conditions.
- The second experiment contains a PDE with a nonlinear, time- and material-dependent heat source and considers random 2D convex polygons with random inhomogeneous PDE parameter distributions. In particular, the model has to learn the influence of random arbitrary material fields on the solution of a nonlinear PDE.
- Finally, the method is applied to various 3D hollow cylinder meshes with mixed boundary conditions, that are heated over time due to a Neumann boundary condition.

Appendix A describes the three experiments in more detail.

Training. In the following, a problem p contains a concrete specification of the experiments. In particular, it includes the specification of all material parameters, such as the diffusivity α, the nonlinear source term $q(T)$ in Eq. (1), a geometry Ω, the initial condition T^0, the Dirichlet $\overline{T}$, $\forall \mathbf{x} \in \Gamma$ and Neumann boundary conditions $h_{\mathcal{N}}$, $\forall \mathbf{x} \in \partial\Omega_{\mathcal{N}}$. All experiments consider a total number of 100 randomly generated problems. 75 of these problems are randomly selected for training, while the other 25 are excluded from the model training and used for testing. The random parameters for training and testing are drawn from the same uniform distributions, which are specified in the tables of Appendix A. Training is conducted over 5 random seeds. Each training consists of 500 epochs with mini-batches consisting of 2 graphs. Adam [31] is utilized as the optimizer with an exponential decay of the learning rate from 1e−3 to 1e−5. The gradient norm is clipped to 1.

PI-MGN. The input vector of node features $\mathbf{X}_{\mathcal{V}}$, edge features $\mathbf{X}_{\mathcal{E}}$, and global features $\mathbf{g}$, respectively, is linearly encoded to a latent vector. All latent dimensions are 128. The MPNs of the PI-MGNs (cf. Fig. 1) consist of $L = 12$ message passing steps. Each MLP of the MPNs consists of 2 layers and uses a rectified linear unit (ReLU) activation function between layers and a layer norm [33] at the end. The latent output vector of the MPN-processor is decoded by a CNN to a vector of length 20, containing the solution approximation of the next 20 time steps. The setup of the CNN follows previous work [13], consisting of 2 layers with 1 input channel, 8 hidden channels and 1 output channel, a kernel stride of 15 and 10, and a stride of 4 and 1, respectively.

Simulation and evaluation. All problems are evaluated for a total amount of 100 time steps, starting from $t^0 = 0\,\mathrm{s}$ and progress until $t^{N_t} = 1\,\mathrm{s}$ is reached with a fixed time step $\Delta t = 0.01\,\mathrm{s}$. For evaluation, FEM-solutions from the open-source package *scikit-fem* [34] serve as a ground truth reference. The FEM-solutions solve the same equations and time steps as the PI-MGNs for comparability. These FEM-solutions are in no way used for PI-MGN training. They only serve as ground truth to assess the quality of PI-MGN solutions after training. A normalized L_2 error

$$\epsilon_{L_2,\mathrm{norm}} = \frac{\left(\sum_{n=1}^{N_t}\sum_{v=1}^{N_\varphi}\left|\tilde{T}_v^n - T_v^n\right|^2\right)^{1/2}}{\left(\sum_{n=1}^{N_t}\sum_{v=1}^{N_\varphi}\left|T_v^n\right|^2\right)^{1/2}} \tag{10}$$

is used to measure the deviation of the PI-MGN prediction from the ground truth. For visualizing error distributions, a relative metric

$$\epsilon_{\mathrm{rel}}^n = \frac{\tilde{T}_v^n - T_v^n}{\max_{v,n}(T_v^n) - \min_{v,n}(T_v^n)} \tag{11}$$

is applied.

Baselines. The PI-MGNs are compared to data-driven MeshGraphNets (DD-MGNs) [10] and to PI-GGNs from [26], which train graph Chebyshev convolutional neural network (GCCN) with the FEM for solving forward and inverse stationary problems of a single geometry.

For the DD-MGNs baseline, the ground truth FEM data T_v^n is integrated into the training process, which amounts to 75 simulations per experiment and per repetition for training and 25 for testing. Therefore, the loss function in Eq. (9) is replaced by a data-based MSE loss

$$\mathcal{L}_{\mathrm{data}}^n = \frac{1}{N_{\mathrm{TB}} N_{\varphi,F}} \sum_{\tilde{n}=1}^{N_{\mathrm{TB}}} \sum_{v=1}^{N_{\varphi,F}} \left|\tilde{T}_v^{n+\tilde{n}} - T_v^{n+\tilde{n}}\right|^2. \tag{12}$$

Analogous to the physics-informed loss $\mathcal{L}_{\mathrm{FEM}}^n$, the data loss measures the deviation of all N_{TB} approximations of the MGN for an input graph $\mathcal{G}^n$ at the time step n. Since the ground truth is now available during training, the noisy ground truth difference $(T_v^n + \epsilon_v) - (T_u^n + \epsilon_u)$ is added as an edge feature instead of the noisy approximation difference $(\tilde{T}_v^n + \epsilon_v) - (\tilde{T}_u^n + \epsilon_u)$, as in [10]. Apart from these two changes, everything remains the same for comparability, which includes all integrated extensions such as time bundling and global features. Hence, the DD-MGN and PI-MGN approaches differ only in their training and work identically during inference.

For the physics-informed baseline PI-GGN, the model is integrated into the PI-MGNs framework. Therefore, the GCCN of [26] replaces the MPN of this work and the absolute positions and the solution are added as node features as in [26], because the GCCN cannot consider edge features. The input size of the model is set to the experiment-specific numbers of node features and the output size to the time-bundling size $N_{\mathrm{TB}} = 20$. All other components of the GCCN remain the same as in [26], while all setups, such as the training setup, are consistent with the PI-MGN setup.

3.2. Generalization to large unseen meshes

In addition, the present work examines the ability of PI-MGNs to generalize to large, unseen meshes – that is, meshes with a significantly larger number of mesh elements than the meshes considered in training. The pre-trained PI-MGNs and baseline models are applied on four unseen large mesh problems. The first two problems investigate the performance of the trained PI-MGNs to predict the 2D linear heat diffusion for large corrugated sheets consisting of 10 and 100 components, respectively. Subsequently, the models of the 2D nonlinear heating are applied on a large grid structure. The last problem considers the PI-MGNs of the 3D heatup with mixed boundary conditions applied on a long 3D hollow cylinder. Appendix A.4 contains a detailed description of the large mesh problems.

3.3. Additional experiments

Ablation studies. Ablation studies investigate the effectiveness of the individual components of the PI-MGN architecture. In particular, the time bundling CNN decoder is compared to an MLP decoder and the sensitivity of the accuracy with respect to the time bundling size N_{TB} is investigated. Further, the relative positional encoding is compared to an absolute positional encoding and the effectiveness of the global features and the training noise is investigated. Finally, the best-performing variants from the ablation study will be applied to the large mesh experiments to evaluate their generalization ability to generalize to large parts.

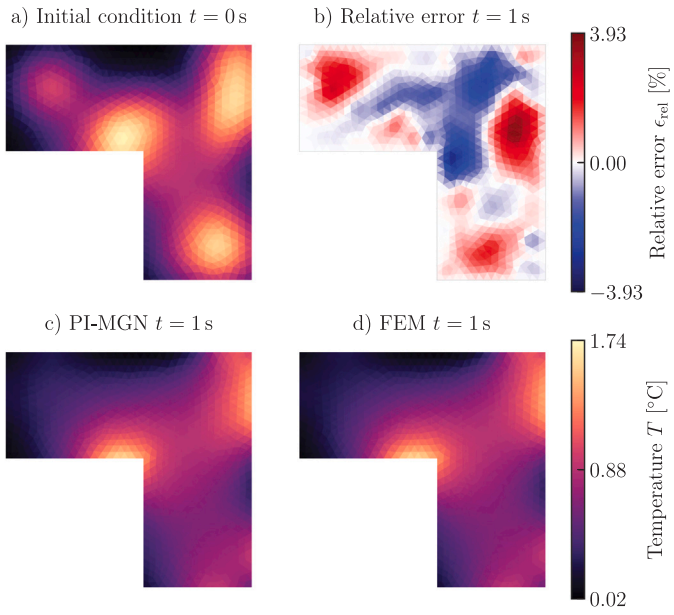

Fig. 4. Comparison of the PI-MGN approximation to the FEM solution at the last time step $t = 1\,\mathrm{s}$ for an unseen problem of the 2D linear experiment of Section 3.1. Fig. 4 (a) shows the randomly sampled initial condition on a new L-shaped mesh geometry. The relative error of the PI-MGN in Fig. 4 (b) measures the deviation of the PI-MGN approximation in Fig. 4 (c) from the ground truth in Fig. 4 (d).

Speed comparison. To evaluate the computational efficiency of the PI-MGN, the wall-clock time is compared for the 2D nonlinear and inhomogeneous heating experiments, as those are the most challenging for the FEM due to the nonlinearity. Additionally, the calculation time of the physics-informed loss Eq. (9) and the data-driven loss Eq. (12) as well as the computation time for updating the PI-MGN and the DD-MGN are compared.

Combined physics-informed and data-driven training. The proposed physics-informed training approach enables data-free training of MGNs, but can naturally be combined with data-driven training. Hybrid MGNs are analyzed that are updated during training with both the data-driven loss Eq. (12) for some of the 75 training problems and with the physics-informed Eq. (9) for the remaining problems.

4. Results

4.1. Training on small meshes

Qualitative comparison to the FEM. For a qualitative comparison of the accuracy and capability of the trained models, the solutions provided by PI-MGN are compared visually to the ground truth FEM on unseen part designs, material distributions, and process settings in Figs. 4 to 6. The results indicate that PI-MGN can produce accurate predictions that are visually similar to FEM solutions.

More precisely, Fig. 4 (a) shows a challenging example of the 2D linear heat diffusion experiment of Section 3.1, which contains a random initial condition with 4 hot spots on an unseen part design. It can be seen that the PI-MGN has learned to accurately predict the final solution at the last time step $t = 1\,\mathrm{s}$. Fig. 4 (b) depicts the relative error ϵ_{rel}.

Fig. 5 shows a nonlinear heating problem from Section 3.1 with a challenging material distribution on a randomly selected seed to demonstrate that the PI-MGNs learn the influence of nonlinear material fields. Fig. 5 (a) shows the unseen material distribution V_{f} on the unseen part. Fig. 5 (d) depicts the ground truth solution, showing that spots with low values of the material property result in a locally stronger heating. Fig. 5 (c) demonstrate that the PI-MGN precisely reproduces these solution. Fig. 5 (b) depicts the relative error between both simulations, showing that comparatively high deviations mostly appear close to the boundary.

Finally, the qualitative performance of the PI-MGN is compared to the FEM for an unseen and randomly selected instance of the 3D heating task with mixed boundary conditions of Section 3.1. Fig. 6 (a) shows the approximation of the PI-MGN, Fig. 6 (b) the ground truth and Fig. 6 (c) the relative error at the last time step $t = 1\,\mathrm{s}$ of the simulation. A quarter of the unseen 3D hollow cylinders is cut off for visualization, providing a view of the solution inside the cylinder. As before, the PI-MGN and FEM solutions are in good agreement. Only a slight underestimation of the PI-MGN is visible.

The results show that the PI-MGN can handle non-stationary and nonlinear PDEs in 2D and 3D, enabling visibly precise prediction for the unseen part designs, inhomogeneous material fields under new process conditions.

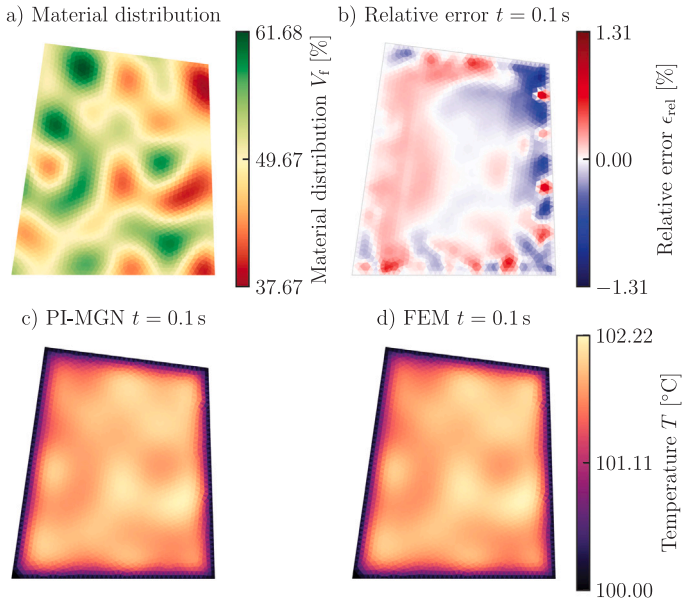

Fig. 5. The PI-MGN solution is compared to the FEM for an nonlinear problem of the experiment of Section 3.1 after $t = 0.1\,\mathrm{s}$. The geometry and material distribution (cf. Fig. 5 (a)) of the part was unseen during training. Fig. 5 (c) and Fig. 5 (d) show the approximation of the PI-MGN respectively the solution of the FEM. Fig. 5 (d) depicts the relative error between both solutions.

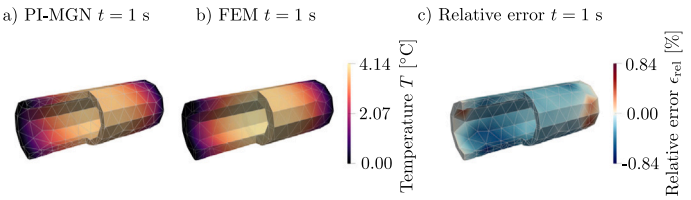

Fig. 6. The 3D experiment of Section 3.1 with mixed boundary conditions. Fig. 6 (a) shows the solution of the PI-MGN at the last time step $t = 1\,\mathrm{s}$ and Fig. 6 (b) the ground truth FEM solution. Fig. 6 (c) depicts the relative error of Fig. 6 (a) compared to Fig. 6 (b). A quarter of the 3D hollow cylinders is cut off for visualization in Fig. 6 (a), (b) and (c).

Comparison to baseline models. Fig. 7 shows the performance of the PI-MGNs relative to the baselines DD-MGNs and PI-GGNs for all three experiments: The 2D linear heat diffusion ('*2DL*'), the 2D nonlinear heating ('*2DNL*') and the 3D heating with mixed boundary conditions ('*3DMB*'). The mean $\mu_{L_2,\mathrm{norm}}$ and the standard deviation $\sigma_{L_2,\mathrm{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\mathrm{norm}}$ over five training repetitions is displayed for each method, calculated for 25 unseen test problems. The comparability of absolute values between experiments is limited as all values are normalized by their reference solution. For example, the mean temperature in the 2D nonlinear experiment is about 100 °C, while it is 0 °C in the others. The exact values are listed in Table B.7 in Appendix B.1, and are multiplied by 1e+3 in the table for readability.

The PI-MGN outperforms the DD-MGN in two out of three experiments. The mean error $\mu_{L_2,\mathrm{norm}}$ of the data-driven training is almost 50 % higher for the 2D linear heatup experiment and approximately doubles for the 2D nonlinear and inhomogeneous heating. In contrast, the mean error of PI-MGNs is only slightly higher than that of DD-MGNs for the 3D heatup task. Here, the standard deviations overlap. Furthermore, the standard deviation $\sigma_{L_2,\mathrm{norm}}$ of the data-driven approach is larger in all three tasks, with an increase of up to 200 %. For this 3D experiment, the physics does not vary as much across problems as in the other experiments, potentially explaining the similar performance across methods. However, training on similar problems may decrease generalization to out-of-distribution problems. This is demonstrated in Section 4.2, where accuracy decreases for an adapted geometry.

The baseline PI-GGN model based on GCCN performs significantly worse for the 2D experiments, compared to the proposed PI-MGN and the DD-MGN baseline. For the 2D linear problem, the error is even more than an order of magnitude higher. In addition, the accuracy of the model is highly uncertain, which makes the training unreliable. The performance in the 3D mixed boundary task is slightly decreased, but still comparable. As for the DD-MGNs, it will be demonstrated in Section 4.2 that the trained models generalize worse to large meshes. This shows that the proposed MGN architecture plays an important role in the

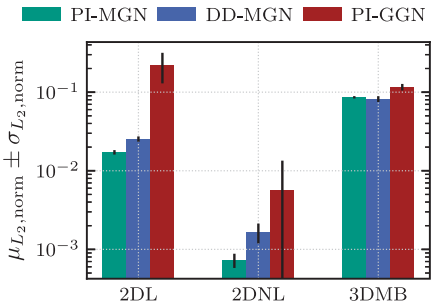

Fig. 7. Comparison of the PI-MGNs to DD-MGNs and to the physics-informed baseline PI-GGN. Fig. 7 shows the mean $\mu_{L_2,\text{norm}}$ (bar) and the standard deviation $\sigma_{L_2,\text{norm}}$ (black line on bar) of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$, calculated for five training repetitions per model and experiment. Each bar group contains the results of the PI-MGNs, DD-MGNs and PI-GGNs for one of the three experiments of Section 3.1.

generalization capabilities and reliability of PI-MGNs, allowing the approach to perform well on non-stationary problems of unseen arbitrary meshes, material distributions and process conditions.

Overall, the proposed PI-MGN alleviates the expensive data creation process required for DD-MGNs while showing improved accuracy and consistency on different tasks. Further, the performance of the PI-MGN increases up to an order of magnitude, compared to the physics-informed baseline PI-GGN.

4.2. Generalization to large unseen meshes

Qualitative comparison to the FEM. The pre-trained models from Section 4.1 are applied to predict the time evolution on two large mesh experiments from Section 3.2 without any retraining or modification. The results are compared qualitatively to the FEM. Fig. 8 (a) shows the initial condition for a randomly selected model of the 5 trained models. Even though the corrugated sheet mesh differs greatly from the L-shapes seen during training, the PI-MGN accurately predicts (cf. Fig. 8 (b)) the solution at the last time step when compared to the solution of the FEM in Fig. 8 (c). Fig. 8 (d) depicts the relative error between the PI-MGN and the FEM, showing that the PI-MGN tends to overestimate the solution. The highest error occurs at a point with high temperatures and temperature gradients, which are significantly larger than those observed during training.

The material distribution of the grid structure is depicted in Fig. 9 (a). The solution of the randomly chosen PI-MGN in Fig. 9 (c) is in good agreement with the FEM solution of Fig. 9 (d). As before, the local temperature increases depend on the surrounding material distribution and are especially prominent at the crossings of the grid structure. The PI-MGN provide accurate predictions and resolves fine local temperature changes. Interestingly, this is in contrast to classical PINNs, which have been reported to have a spectral bias towards learning low frequencies and may not be able to resolve such fine temperature changes [35,36]. Fig. 9 (b) shows that the PI-MGN tends to overestimate the temperature at the bars and to underestimate it at the crossings.

Comparison to baseline models. Fig. 10 compares the PI-MGN, DD-MGN and PI-GGN accuracy on the four large mesh experiments: Two 2D linear heat diffusion experiments with corrugated sheet parts, one considering a part with 10 repeating components (‘*2DL-10*’) and the other considering a part with 100 repeating components (‘*2DL-100*’), one 2D nonlinear and inhomogeneous heating experiment with the grid structure (‘*2DNL-L*’) and one 3D heating experiment with mixed boundary conditions and the long hollow cylinder (‘*3DMB-L*’). As before, the models from Section 4.1 are re-used without any further training or modification. Fig. 10 shows the mean $\mu_{L_2,\text{norm}}$ and standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error across the 5 repetitions. Table B.7 lists the exact values (multiplied by 1e+3 for readability) in Appendix B.2.

The DD-MGNs show comparable performance for the 2D linear heat diffusion task of the corrugated sheet mesh with 10 components. The mean error $\mu_{L_2,\text{norm}}$ is slightly higher and the standard deviation $\sigma_{L_2,\text{norm}}$ lower. For the other three problems, the PI-MGNs outperform the DD-MGNs. Hence, the physics-informed training of MGNs not only saves on expensive data creation but also improves generalization to larger meshes. A reason for that could be that the physics-informed training has to explore the solution during training, because no ground truth is known. The solution approaches the ground truth during training, but always fluctuates due to the model inaccuracy. Therefore, the model must predict a slightly different solution with respect to the imprecise output of the previous time step, which can help to stabilize the model rollout.

4.3. Additional experiments

Ablation studies. The ablation studies conducted in Appendix B.3 demonstrate the effectiveness of each component of the PI-MGN architecture in the small mesh experiments and its ability to generalize to larger, unseen meshes. For the training on small meshes, the PI-MGN shows the highest reliability in predicting accurate results across all experiments. Time bundling and global features are important components to achieve this reliability. Replacing the relative positional encoding with absolute positions as well as noise free training prove to be competitive alternatives. However, the results on the large meshes suggest that PI-MGN require relative encoding and robustness to noise to efficiently generalize beyond their training distribution.

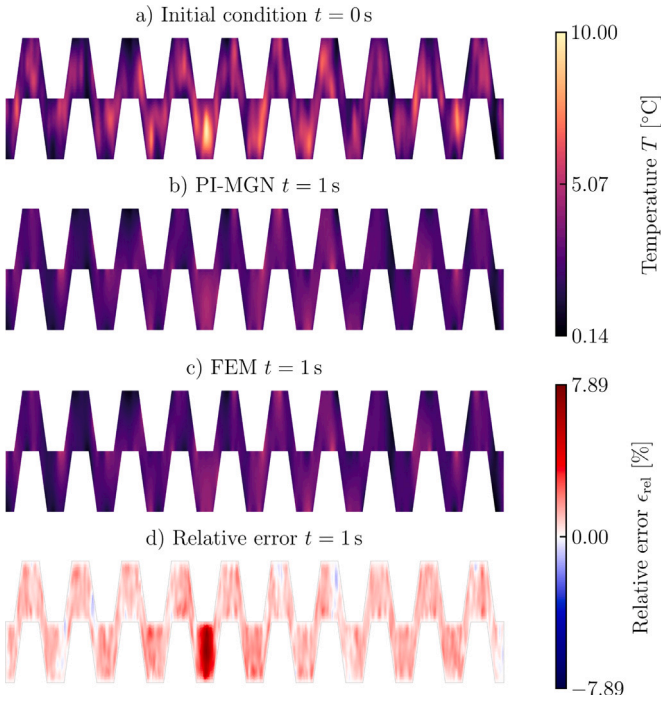

Fig. 8. 2D linear heat diffusion experiment of Section 3.2 on a large mesh consisting of 10 components. The part geometry and the random initial condition of Fig. 8 (a) was not part of the training. Fig. 8 (c) shows the solution of a PI-MGN, which was trained on the small meshes of Section 3.1. The deviation from the ground truth FEM solution (cf. Fig. 8 (c)) is depicted in Fig. 8 (d). For visualization purposes, the image is not to scale.

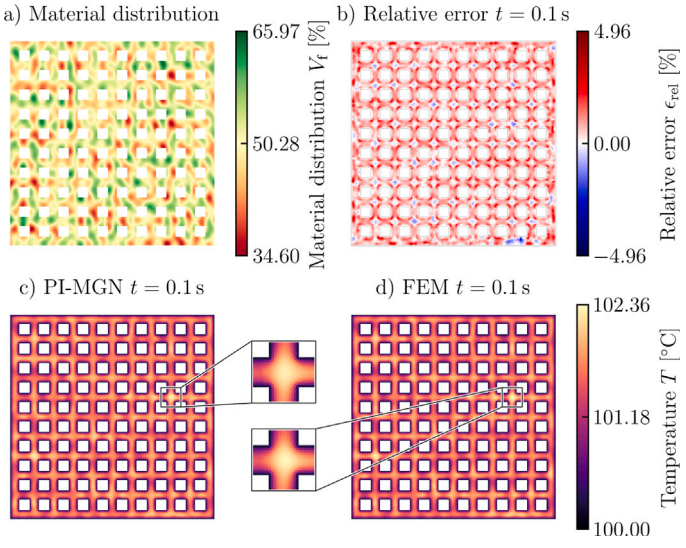

Fig. 9. A trained PI-MGN predicts the solution of the nonlinear experiment of Section 3.2 on an large grid mesh. These PI-MGN is only trained on small polygon meshes. The large part contains an inhomogeneous material distribution, which is depicted in Fig. 9 (a). Fig. 9 (c) shows the approximation of the PI-MGN, Fig. 9 (d) the ground truth and (b) the relative error the PI-MGN solution.

Speed comparison. Fig. 11 compares the computation time of the PI-MGNs to the FEM for the 2D nonlinear and inhomogeneous heating. The results of the PI-MGNs are calculated on a GPU (NVIDIA® GeForce® RTX 2080) and on a CPU (Intel® Xeon® CPU E5-2680 v4 @ 2.40 GHz), while the FEM results are only calculated on the CPU.

The PI-MGNs require on average only about 0.3 s on the GPU and 0.9 s on the CPU for the small meshes compared to approximately 50 s with the FEM on the same CPU. For the large mesh grid, the PI-MGNs output the solution in about 4 s on the GPU and in about 33.8 s on the CPU while the FEM solver requires 45 min on the CPU.

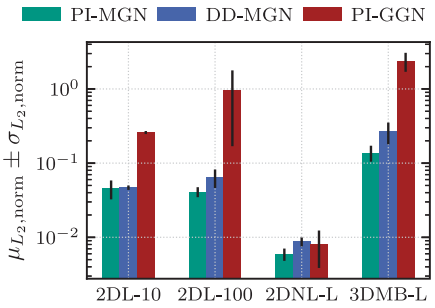

Fig. 10. Performance of the PI-MGNs on the large mesh experiments of Section 3.2 compared to DD-MGNs and to PI-GGNs. The mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$ is calculated for five training repetitions per model and experiment. For each large mesh experiment, Fig. 10 displays the mean $\mu_{L_2,\text{norm}}$ (bar) and the standard deviation $\sigma_{L_2,\text{norm}}$ (black line on bar) of the three models.

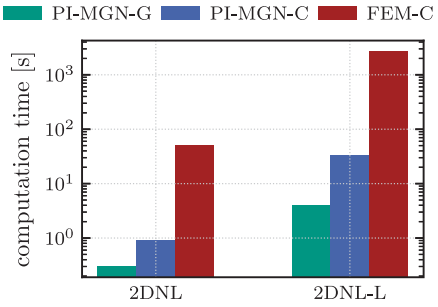

Fig. 11. Speed comparison of the PI-MGNs to the FEM on the 2D nonlinear and inhomogeneous heating for the small meshes ('*2DNL*') and large meshes ('*2DNL-L*'). Fig. 11 compares the computation time of the PI-MGN on a GPU ('*PI-MGN-G*') to its computation time on a CPU ('*PI-MGN-C*') and to the FEM on a CPU ('*FEM-C*'). The GPU results are calculated on an NVIDIA® GeForce® RTX 2080 and the CPU results on an Intel® Xeon® CPU E5-2680 v4 @ 2.40 GHz.

Appendix B.3 compares and analyzes the average computation time for the physics-informed loss and the data-driven loss for 75 of the small mesh experiments. Additionally, the average time of a training step with the PI-MGNs and DD-MGNs is considered. Overall, the calculation of the FEM loss is, as expected, more expensive than the data error. However, it is still very fast with 1–2 ms, so that the additional effort of a training step with PI-MGNs is only 13 to 30 % more than with the DD-MGNs. This additional training cost is negligible when compared to the effort of creating the DD-MGNs training data with the FEM.

Combined physics-informed and data-driven training. The combined physics-informed and data-driven training is analyzed in Appendix B.3 for the small mesh experiments of Section 3.1. The results for these hybrid models is consistent with those of Section 4.1. The PI-MGNs still perform best on the 2D experiments, and hybrid variants with a high proportion of physics-informed problems outperform the other models. The opposite is true for the 3D mixed BC experiment, where the models with a predominance of data are slightly more accurate. However, aside from one case, the PI-MGNs and hybrid models have a lower standard deviation than the DD-MGNs in all three experiments, illustrating the increased reliability of the physics-informed training.

To sum up, the PI-MGNs trained on small meshes enable the most accurate and reliable predictions on large meshes over all considered learned models, while achieving speedups of several orders of magnitude compared to the FEM.

5. Conclusion and outlook

This work introduces physics-informed MeshGraphNets (PI-MGNs) for fast, accurate and reliable finite element method (FEM) simulations of non-stationary problems on arbitrary mesh-geometries, material distributions and process conditions. PI-MGNs efficiently combine physics-informed neural networks (PINNs) and MeshGraphNets (MGNs) to overcome the time-intensive and cumbersome data creation process required for data-driven MeshGraphNets (DD-MGNs) while improving accuracy and generalization compared to existing PINNs. During training, the FEM equations are discretized and evaluated per mesh element, allowing computations of the physics-informed loss for multiple time steps and nonlinear problems directly and in parallel. The state of the art MGNs [10] are extended with additional methods such as time-bundling and global graph features, all of which empirically improve performance.

The PI-MGNs predict accurate solutions in good agreement with numerical FEM solvers for unseen arbitrary parts with inhomogeneous materials under changed process conditions. For a 2D heat diffusion problem, the PI-MGNs precisely predict the thermal cool down of unseen parts under random initialized temperature distributions. Further, the PI-MGNs learn to account for

random material distributions in a part when predicting a nonlinear 2D heat release process. Finally, the approach accurately predicts the heating for 3D parts with mixed boundary conditions as a result of heat flow at the boundary.

The introduced physics-informed training of the MGNs is compared to a data-driven training approach and to a physics-informed neural FEM solver. Even though PI-MGNs require no simulation data during training, they perform on par with or better than the data-driven variant in terms of accuracy on all experiments. This advantage of the physics-based training is even more significant when generalizing to larger parts during evaluation. In contrast to the data-driven training, the PI-MGNs reliably predict accurate simulations on large unseen meshes when trained only on small and generic meshes. The existing physics-informed neural FEM solvers in literature use architectures and parameterizations that are shown to perform worse in all experiments for both small and large meshes, highlighting the importance of the proposed PI-MGNs. An ablation study reveals the individual importance of several components of the MGNs, such as time bundling and global features. Additional experiments demonstrate the speed-up of the PI-MGNs compared to classical FEM and the low computational cost of the physics-informed approach. Experiments on small meshes suggest that a combined physics-informed and data-driven training of MGNs yields overall more accurate and reliable models than data-driven training. However, purely physics-informed training still performs best for the 2D experiments.

Overall, the PI-MGNs prove to be a reliable and accurate solver for the non-stationary and nonlinear FEM with strong generalization to unseen processes with new meshes consisting of inhomogeneous materials. In addition, the PI-MGNs are significantly faster and fully differentiable, making them a promising option for inverse design optimization. Here, the MGNs could also be extended to a Bayesian formulation to respect the model uncertainty during optimization. This work considers the Galerkin FEM with piecewise linear polynomials, but could be augmented for handling higher order polynomials or be transferred to other FEM formulations, e.g., the discontinuous FEM, by developing appropriate graph representations and message passing schemes. Finally, PI-MGNs could be applied to speed-up simulations with other challenging nonlinearities, such as contact or large deformation simulations.

CRediT authorship contribution statement

Tobias Würth: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Niklas Freymuth:** Writing – review & editing, Visualization, Validation, Software, Methodology. **Clemens Zimmerling:** Writing – review & editing, Visualization, Methodology, Conceptualization. **Gerhard Neumann:** Validation, Funding acquisition. **Luise Kärger:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

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.

Data availability

The data that has been used is confidential.

Acknowledgments

This work has been performed in subprojects T2 and M2 of the DFG AI Research Unit 5339, funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459291153. This work is also part of the Heisenberg project "Digitalization of fiber-reinforced polymer processes for resource-efficient manufacturing of lightweight components", funded by the DFG (project no. 455807141). The authors thank the German Research Foundation for its financial support.

Appendix A. Experiments

A.1. Experiment 1: 2D linear heat diffusion

The first experiment (cf. Fig. 3 (a)) considers a heat diffusion problem on a 2D L-Shaped domain, setting the source term $q(T) = 0$ in Eq. (1) and the thermal diffusivity α to a constant value. The initial condition is described by a sum of N_{ic} Gaussians

$$T^0(\mathbf{x}) = \sum_{i=0}^{N_{\text{ic}}} a_i \exp\left(-\left(\frac{(x - x_{0,i})^2}{2\sigma_{x,i}} + \frac{(y - y_{0,i})^2}{2\sigma_{y,i}}\right)\right).$$

The factors a_i and the standard deviations $\sigma_{x,i}, \sigma_{y,i}$ are randomly selected from uniform distributions. The mean positions $x_{0,i}, y_{0,i} \in \Omega \cup \partial\Omega$ are randomly selected node positions. The initial temperature on the boundary is fixed during the whole simulation, i.e., $\overline{T} = T^0(\mathbf{x})$, $\forall \mathbf{x} \in \Gamma$. Therefore, the problem contains no Neumann boundary condition, i.e. $\partial\Omega_{\mathcal{N}} = \{\}$.

The 2D L-shaped domain is defined as the difference of two rectangles. The large one has length L and height H and the small rectangle the length $L_{\text{s}} = a_{\text{L}} L$ and the height $H_{\text{s}} = a_{\text{H}} H$, where L, H, a_{L} and a_{H} are selected randomly, cf. Fig. 3 (a). The small rectangle is randomly placed at one of the corners of the large domain and subtracted from the large rectangle to yield the final L-shaped domain. A total amount of 100 L-shaped domains are created and remain the same for all five seeds. However, mesh selection and initial conditions are still randomized over seeds for training and testing. The exact definitions of the constants and the distributions of the randomly selected values are listed in Table A.1.

Table A.1
Constants and randomly defined values of Experiment 1, the 2D linear heat diffusion experiment.

Property	Symbol	Value	Unit
Initial–boundary value problem			
Source term	q	0	K/s
Thermal diffusivity	α	5e−2	m^2/s
Number of Gaussians	N_{ic}	10	–
Summand weight	a_i	$\sim \mathcal{U}(0.5, 1)$	°C
Standard deviation x-direction	$\sigma_{x,i}$	$\sim \mathcal{U}(1/12, 1/6)$	m^2
Standard deviation y-direction	$\sigma_{y,i}$	$\sim \mathcal{U}(1/12, 1/6)$	m^2
Geometry – L-shape			
Length	L	$\sim \mathcal{U}(0.5, 1)$	m
Height	H	$\sim \mathcal{U}(0.5, 1)$	m
Length factor	a_L	$\sim \mathcal{U}(1/3, 2/3)$	–
Height factor	a_H	$\sim \mathcal{U}(1/3, 2/3)$	–

A.2. Experiment 2: 2D nonlinear and inhomogeneous heating

This experiment (cf. Fig. 3 (b)) contains a nonlinear source term

$$q(T) = q(T, t, \mathbf{x}) = \tilde{q}_0(\mathbf{x}) \exp\left(-CTt/t_0\right) CT \tag{A.1}$$

in Eq. (1), where $C = 8/T_0$, t_0, T_0 and q_0 are constants and the heat factor $\tilde{q}_0(\mathbf{x}) = q_0(1 - V_f(\mathbf{x}))$ depends on an inhomogeneous material property V_f

$$V_f(\mathbf{x}) = V_{f,0} + \sum_{i=1}^{N_f} a_i \sin\left(k_{x,i}x + d_{x,i}\right) \sin\left(k_{y,i}y + d_{y,i}\right).$$

These material distributions are remotely inspired by inhomogeneities which may occur in press forming processes of short fiber reinforced composites. The material field contains N_f functions, each depending on a product of sine waves in x and y-direction. The factor a_i, the wavenumbers $k_{x,i}$ and $k_{y,i}$ as well as function shifts $d_{x,i}, d_{y,i}$ are randomly chosen from uniform distributions, while $V_{f,0}$ is a constant. Additionally, the diffusion coefficient

$$\alpha = \alpha(\mathbf{x}) = 1/\left(V_f(\mathbf{x})/\alpha_f + (1 - V_f(\mathbf{x}))/\alpha_m\right) \tag{A.2}$$

also depends on the inhomogeneous material distribution $V_f(\mathbf{x})$ as well as on the constants α_f and α_m. The temperature at the initial condition T^0 and at the whole boundary $\overline{T}$ is set to $T^0 = \overline{T} = T_0$. The considered domains are convex polygons. These are constructed by uniformly sampling 7 points x_P and y_P in the 2D domain $\Omega = (0, L) \times (0, H)$, cf. Fig. 3 (b). Subsequently, the convex hull of the set of the seven points is chosen as the domain, yielding a convex polygon with 3 to 7 boundary points. Table A.2 lists the values of the constants and distributions of the random quantities.

The polygons and material distributions $V_f(\mathbf{x})$ of the 100 problems are randomly selected in a pre-processing step and fixed for all training seeds. The problem selection of the 75 training and 25 testing problems is randomized across the five seeds. As the heat source $q(T, t, \mathbf{x})$ depends on time and temperature, the temperature T or rather the noisy temperature approximation $\tilde{T} + \epsilon$ is added as a node feature. The time is added as both, a global and a node feature. In addition, the material property, evaluated at the nodes, is added as a node feature.

To obtain a ground truth solution for this nonlinear equation from a numerical solver, the equation is linearized and solved in an inner loop for each time step for a total number of 100 inner iteration or until the solution is converged, depending on what happens first.

A.3. Experiment 3: 3D heating with mixed boundary conditions

In the following, the PDE of Appendix A.1 is considered without a heat source $q(T) = 0$ for a 3D hollow cylinder mesh with mixed boundary conditions (cf. Fig. 3 (c)).

The hollow cylinder has the length L, the outer radius R_o and the inner radius $R_i = a_i R_o$, where L, R_o and the fracture a_i are randomly selected values. At the beginning, the temperature is $T^0 = T_0$ across the domain. The temperature at the left and right cylinder boundary $\Gamma_{l,r} = \{\mathbf{x} = (x, y) \in \partial\Omega : x = 0, L\}$ is also fixed to $\overline{T} = T_0$. A constant Neumann boundary condition is defined at the boundary surface at the inner side of the cylinder, i.e. at $\partial\Omega_i = \{\mathbf{x} = (x, y) \in \partial\Omega : y = R_i\}$ as $\alpha h_{\mathcal{N}} = h_0$. At the outer side of the cylinder $\partial\Omega_o = \{\mathbf{x} = (x, y) \in \partial\Omega : y = R_o\}$ an adiabatic Neumann boundary condition $\alpha h_{\mathcal{N}} = 0$ is considered. The thermal diffusivity α is constant.

The initial learning rate is changed to 1e−4, but still decays to 1e−5. The 100 meshes of the 3D hollow cylinders are the same for all five training seeds, but the part selection for the training and testing remains random for the individual seeds. The inner heat h_0 of the Neumann boundary condition is added as an additional node feature. The exact constants and the randomly selected value definitions are given in Table A.3.

Table A.4 provides an overview of the node, edge and global features used in the experiments as well as the output values of the PI-MGNs.

Table A.2
Constants and randomly defined values of Experiment 2, the 2D nonlinear and inhomogeneous experiment.

Property	Symbol	Value	Unit
Initial–boundary value problem			
Reference time	t_0	1	s
Reference temperature	T_0	100	°C
Scaling factor	q_0	20	K/s
Number of functions	N_f	10	–
Summand weight	a_i	$\sim \mathcal{U}(0, 1/20)$	–
Wavenumber x-direction	$k_{x,i}$	$\sim \mathcal{U}(0, 8\pi)$	1/m
Wavenumber y-direction	$k_{y,i}$	$\sim \mathcal{U}(0, 8\pi)$	1/m
Wave shift x-direction	$d_{x,i}$	$\sim \mathcal{U}(0, 2\pi)$	–
Wave shift y-direction	$d_{y,i}$	$\sim \mathcal{U}(0, 2\pi)$	–
Median material property	$V_{\mathrm{f},0}$	0.5	–
Thermal diffusivity fiber	α_{f}	0.1	m^2/s
Thermal diffusivity matrix	α_{m}	0.01	m^2/s
Geometry – Convex polygon			
Geometry length	L	1	m
Geometry height	H	1	m

Table A.3
Constants and randomly defined values of Experiment 3, the 3D heating experiment with mixed boundary conditions.

Property	Symbol	Value	Unit
Initial–boundary value problem			
Source term	q	0	K/s
Reference temperature	T_0	0	°C
Thermal diffusivity	α	1	m^2/s
Inner heat constant	h_0	1	m K/s
Geometry – Hollow cylinder			
Length	L	$\sim \mathcal{U}(4, 5)$	m
Outer radius	R_{o}	$\sim \mathcal{U}(0.8, 1)$	m
Fracture radius	a_{i}	$\sim \mathcal{U}(0.6, 0.8)$	m

Table A.4
Input and output features used in each experiment, including node, edge, and global input features, as well as the output features on the nodes.

Experiment	Node	Edge	Global	Outputs
2D Linear	n_{t}	$\mathbf{x}_{vu}$, $\lvert\mathbf{x}_{vu}\rvert$, T_{vu}	–	T
2D Nonlinear	n_{t}, t, T, V_{f}	$\mathbf{x}_{vu}$, $\lvert\mathbf{x}_{vu}\rvert$, T_{vu}	t	T
3D mixed BCs	n_{t}, h_0	$\mathbf{x}_{vu}$, $\lvert\mathbf{x}_{vu}\rvert$, T_{vu}	–	T

A.4. Large mesh problems of the experiments

For the first two problems, the models have to predict the 2D linear heat diffusion of Appendix A.1 for large corrugated sheets, which consist of repeating components of length L and thickness D. The first problem considers a mesh consisting of $N_{\mathrm{c1}} = 10$ repeating components with 10 840 total mesh elements and the second of $N_{\mathrm{c2}} = 100$ components with 107 592 elements. The number of Gaussians for the initial condition is increased to $N_{\mathrm{ic}} = 1000$. As a result, the density of the Gaussians in the initial state in the two parts varies by a factor of 10. These evaluation problems are multiple magnitudes larger larger than anything seen during training, as the training process for all methods uses meshes with 155 to 524 elements (cf. Table A.6) and $N_{\mathrm{ic}} = 10$ Gaussians.

The third problem considers the 2D nonlinear heating of Appendix A.2 for a grid structure. The grid part is a large rectangle with the length and height L and contains $N_{\mathrm{h}} = 100$ rectangle holes equally distributed along the length and height direction and with a distance of d_{h}. The mesh of the grid contains 23 296 elements, while the training meshes contain between 148 and 1122 elements (cf. Table A.6).

The last problem considers the 3D heatup with mixed boundary conditions of Appendix A.3 for a long 3D hollow cylinder, whose length L is 40–50 times larger than that of the training meshes, while the outer radius R_{o} and the inner radius multiplier a_{i} correspond to the mean value of the training meshes. The other settings remain the same as in Appendix A.3.

The added constants and distributions of the randomly selected values for these experiments are listed in Table A.5. Table A.6 displays the concrete number of elements, nodes, and edges of the considered meshes in this work.

Table A.5
Constants and randomly defined values of the large mesh experiments.

Property	Symbol	Value	Unit
Experiment 1: 2D Linear			
Initial–boundary value problem			
Number of Gaussians	N_{ic}	1000	–
Geometry – Corrugated sheet			
Component length	L	3	m
Component thickness	D	0.5	m
Number of components part 1	N_{c1}	10	–
Number of components part 2	N_{c2}	100	–
Experiment 2: 2D Nonlinear			
Geometry – Large rectangle			
Length/Height	L	4	m
Number of holes	N_{h}	100	–
Distance holes	d_{h}	0.175	m
Experiment 3: 3D mixed BCs			
Geometry – Long hollow cylinder			
Length	L	200	m
Outer radius	R_{o}	0.9	m
Fracture radius	a_{i}	0.7	m

Table A.6
Number of elements, nodes, and edges of the meshes considered in this work.

Experiment	Elements	Nodes	Edges
Experiment 1: 2D Linear			
L-shapes	155 - 524	98 - 295	252 - 818
Corrugated sheet 1	10 840	6020	16 859
Corrugated sheet 2	107 592	59 706	167 297
Experiment 2: 2D Nonlinear			
Convex polygons	148 - 1122	98 - 606	245 - 1727
Large rectangle	23 296	12 949	36 344
Experiment 2: 3D mixed BCs			
Hollow cylinders	583 - 784	209 - 281	1003 - 1346
Long Hollow cylinder	10 277	3420	17 117

Table B.7
Comparison of the PI-MGNs to DD-MGNs and to the physics-informed baseline PI-GGN. The table displays the mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$ over five training repetitions for each method and experiment, calculated for 25 test problems. The format of the results is $\mu_{L_2,\text{norm}} \pm \sigma_{L_2,\text{norm}}$ and the results are multiplied by 1e+3 for readability. The best performing results are highlighted.

Methods	**2D Linear**	**2D Nonlinear**	**3D mixed BCs**
PI-MGN	**17.19 ± 1.13**	**0.73 ± 0.15**	86.07 ± 2.97
DD-MGN	25.37 ± 1.97	1.66 ± 0.47	**81.90 ± 6.97**
PI-GGN	223.00 ± 93.66	5.67 ± 7.78	116.11 ± 11.13

Appendix B. Results

B.1. Training on small meshes

Table B.7 displays the results of the small mesh experiments. Section 4.1 analyzes these results and Fig. 7 visualizes them.

B.2. Generalization to large unseen meshes

The exact values of the large mesh experiments results are listed in Table B.8. The results are visually compared in Fig. 10 and discussed in Section 4.2, respectively.

B.3. Additional experiments

Ablation studies. The contribution and effectiveness of individual components of the proposed PI-MGN architecture is investigated in an ablation study. In each ablation, a component of the architecture is removed and, if necessary, replaced by a suitable alternative.

Table B.8
Comparison of the PI-MGNs to the DD-MGNs and PI-GGNs for the experiments on large meshes. The mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$ are calculated for five training repetitions per model and experiment. The tables displays the values in the format $\mu_{L_2,\text{norm}} \pm \sigma_{L_2,\text{norm}}$ multiplied by 1e+3 and the best methods are highlighted.

Methods	2DL-10	2DL-100	2DNL-L	3DMB-L
PI-MGN	**45.42 ± 12.97**	**41.06 ± 6.32**	**5.93 ± 1.12**	**138.13 ± 33.55**
DD-MGN	46.74 ± 3.23	64.13 ± 17.94	8.47 ± 1.20	265.81 ± 86.49
PI-GGN	258.85 ± 12.22	974.42 ± 805.28	8.10 ± 4.24	2386.90 ± 684.33

Table B.9
Ablation study of the PI-MGNs for five training repetitions per method and experiment, considering 25 test problems. The table displays $\mu_{L_2,\text{norm}} \pm \sigma_{L_2,\text{norm}}$, where $\mu_{L_2,\text{norm}}$ is the mean and $\sigma_{L_2,\text{norm}}$ the standard deviation of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$, each multiplied by 1e+3. The top-performing methods are highlighted: blue for the best, green for the second-best, and yellow for the third-best.

Methods	2D Linear	2D Nonlinear	3D mixed BCs
PI-MGN	**17.19 ± 1.13**	**0.73 ± 0.15**	**86.07 ± 2.97**
abs. pos.	17.73 ± 0.96	0.97 ± 0.13	**47.57 ± 18.37**
w/o noise	**16.08 ± 15.53**	0.86 ± 0.16	88.53 ± 6.57
w/o global	**16.37 ± 0.72**	**0.68 ± 0.05**	162.71 ± 8.65
MLP decoder	173.52 ± 143.60	0.86 ± 0.15	**85.13 ± 4.38**
TBS-10	75.89 ± 125.50	**0.77 ± 0.08**	90.94 ± 9.73
TBS-50	21.80 ± 3.93	1.74 ± 0.06	88.37 ± 7.10

Ablation studies - training on small meshes. The first ablation uses absolute positions as node features instead of a relative positional encoding at the edges of the input graph of the PI-MGN ('*abs. pos.*'). Next, the training noise is left out, instead using unmodified predictions as input for the next MGN step ('*w/o noise*'). Then, global features are removed in Eq. (5) of the PI-MGN ('*w/o global*') and the CNN decoder of the model is replaced by an MLP decoder ('*MLP decoder*'). The last two ablations use different time bundling sizes N_{TB}, in particular $N_{\text{TB}} = 10$ and $N_{\text{TB}} = 50$ ('*TBS-10*' and '*TBS-50*', respectively) instead of $N_{\text{TB}} = 20$. These ablations investigate how accuracy changes when the number of model calls is doubled or approximately halved in the experiments.

Table B.9 shows the results for PI-MGNs and all ablations. The table contains the mean $\mu_{L_2,\text{norm}}$ and standard deviation $\sigma_{L_2,\text{norm}}$ of the error $\epsilon_{L_2,\text{norm}}$ over five training repetitions for 25 unseen problems per experiment. The errors are multiplied by 1e+3 for readability.

Only the proposed PI-MGNs architecture perform well in all experiments and consistently score within the top three models in terms of mean error $\mu_{L_2,\text{norm}}$. The absolute position encoding shows comparable performance in two of the three experiments and even outperforms the PI-MGNs in the 3D experiment of Appendix A.3. However, the relative positional encoding is a more solid representation for generalization to geometries, which will be shown in Section 4.2. Removing the training noise leads to increased standard deviations in two of the three experiments, while the mean error remains comparable. Without global features, the prediction error slightly decreases for the 2D experiments, but almost doubles in the 3D case. The same applies for the MLP decoder ablation *MLP decoder* and the two time bundling size ablations *TBS-10* and *TBS-50*. All of them perform similarly to the PI-MGN in two of the experiments, but significantly worse in the third. These components of the architecture might not be necessary in all cases, but prove to be crucial for the reliability of the model. It is important to note that accuracy tends to decrease with fewer model calls (*TBS-50*), even though less error accumulation should increase accuracy. This phenomenon may be attributed to poorer generalization.

Ablation studies - generalization to large unseen meshes. Since training without noise and absolute position encoding showed the most competitive performance, these two ablations are compared to the proposed PI-MGNs architecture for generalization to large unseen meshes. Table B.10 contains the results of the proposed PI-MGN architecture, the PI-MGN with absolute positional encoding ('*abs. pos.*') and the noise-free variant ('*w/o noise*'). Therefore, the pre-trained models from the small mesh ablations are applied on the four large mesh experiments. The values of Table B.10 are the mean $\mu_{L_2,\text{norm}}$ and standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error across the 5 repetitions . Both metrics are scaled by 1e+3.

The PI-MGNs have the lowest error in 3 out of 4 problems. Only the noise free training slightly outperforms the PI-MGNs for the 2D linear heat diffusion problem of the corrugated sheet with 100 components, but clearly struggles to solve the long 3D hollow cylinder problem. The absolute positional encoding only performs well on the nonlinear task with the large grid structure and fails to provide accurate predictions in the other three problems. Remarkably, the noise-free training and absolute positional encoding ablations show decreased performance for the seemingly small change of the 3D problem setup, which considers only a longer hollow cylinder.

Table B.10
Comparison of the PI-MGNs to the best performing variants of the small mesh ablations for the experiments on large meshes. The mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$ is calculated for five training repetitions per model and experiment. The tables displays the values in the format $\mu_{L_2,\text{norm}} \pm \sigma_{L_2,\text{norm}}$ multiplied by 1e+3. The best methods are highlighted.

Methods	2DL-10	2DL-100	2DNL-L	3DMB-L
PI-MGN	**45.42 ± 12.97**	41.06 ± 6.32	**5.93 ± 1.12**	**138.13 ± 33.55**
abs. pos.	183.83 ± 82.99	285.54 ± 78.13	6.33 ± 1.43	766.06 ± 116.25
w/o noise	48.53 ± 26.34	**34.00 ± 1.82**	6.85 ± 0.81	240.78 ± 141.71

Table B.11
Mean calculation time (in ms) per graph of the physics-informed FEM and the data loss, as well as of a training step with the PI-MGN and the DD-MGN. The results are calculated on the same GPU (NVIDIA® GeForce® RTX 2080). .

Methods	2D Linear	2D Nonlinear	3D mixed BCs
Loss calculation			
Physics-informed	1.51 ± 0.09	2.08 ± 0.19	1.674 ± 0.24
Data-driven	0.28 ± 0.03	0.31 ± 0.02	0.30 ± 0.03
Training step			
PI-MGN	14.29 ± 0.70	17.53 ± 1.34	17.29 ± 1.70
DD-MGN	10.97 ± 1.11	13.42 ± 0.52	15.14 ± 0.78

Table B.12
Comparison of PI-MGNs and DD-MGNs with a combined model PI/DD-MGN. The split of the combined approach is denoted as *physics-informed/data-driven*. For example, a split of 75 problems into 50 physics-informed and 25 data-driven training problems is denoted as 50/25. The table presents the mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error $\epsilon_{L_2,\text{norm}}$ for each model and experiment, based on five repetitions ($\mu_{L_2,\text{norm}} \pm \sigma_{L_2,\text{norm}}$). The values are scaled by a factor of 1e+3 and the best results are highlighted.

Methods	Splits	2D Linear	2D Nonlinear	3D mixed BCs
PI-MGN	75/0	**17.19 ± 1.13**	**0.73 ± 0.15**	86.07 ± 2.97
PI/DD-MGN	50/25	18.79 ± 2.03	1.08 ± 0.46	86.35 ± 5.77
PI/DD-MGN	38/37	20.35 ± 1.49	1.53 ± 0.27	85.32 ± 4.81
PI/DD-MGN	25/50	19.28 ± 1.66	1.71 ± 0.15	**81.75 ± 4.04**
DD-MGN	0/75	25.37 ± 1.97	1.66 ± 0.47	81.90 ± 6.97

Notes on the PI-MGNs architecture. Larger time bundling sizes N_{TB} were useful during model development because of the shorter training time, despite the fact that the accuracy decreases. For smaller step batch sizes, noise proved to be important. More message passing steps, wider MLPs, and more training problems have been shown to yield minor improvements. However, this significantly increases the training effort. Reducing the size of these settings resulted in a decrease in accuracy and an increase in standard deviation.

Speed comparison. Table B.11 compares the mean calculation time (in ms) of one forward pass of the physics-informed FEM-loss to the data-driven loss. Additionally, the average time for a training step of the PI-MGN and DD-MGN is evaluated, which includes loading a graph onto the device, the forward pass, the backward pass, and the update of the model. All results are averaged over 75 problems per experiment using the same GPU (NVIDIA® GeForce® RTX 2080).

As expected, the data error is calculated faster than the FEM error, because the latter additionally requires calculating the element-wise error of Eq. (7) and aggregating of the total error of Eq. (8) of the test functions before calculating the loss. However, the computation of the FEM error with about 1–2 ms is still inexpensive compared to a training step, which takes about an order of magnitude longer. The difference between the data-driven training step and the physics-informed training step is therefore much smaller with only about 13 to 30 % additional effort. The fast calculation time will pay off particularly for large components by saving the expensive data creation. The proposed physics-informed training, which locally calculates a total error Eq. (8) for each test function in parallel, is expected to scale linearly with the number of test functions. It is also worth noting that the nonlinear case does not differ significantly from the linear experiments due to the proposed calculation method.

Combined physics-informed and data-driven training. Table B.12 compares the PI-MGNs and DD-MGNs to a hybrid training scheme that interpolates between the two. The 75 training problems of each of the three small mesh experiments are partitioned into data-driven and physics-informed problems. The same model is then trained using both types of problems and their respective losses. For each experiment, three different splits (physics-informed/data-driven) are considered: (50/25), (38/37), and (25/50). Table B.12 shows the mean $\mu_{L_2,\text{norm}}$ and standard deviation $\sigma_{L_2,\text{norm}}$ of the normalized L_2 error across 5 repetitions. All values are scaled by 1e+3 for readability.

In the 2D experiments, the PI-MGNs still perform best in terms of both the mean $\mu_{L_2,\text{norm}}$ and the standard deviation $\sigma_{L_2,\text{norm}}$. The results show a slight trend that the performance increases as the amount of physics-informed samples increases. The 3D experiment also demonstrates a similar trend, but in reverse, as the DD-MGNs exhibit a lower mean error. However, replacing 25 data points with physics-informed training samples slightly reduced the mean error $\mu_{L_2,\text{norm}}$ and significantly reduced the standard deviation $\sigma_{L_2,\text{norm}}$. Overall, the inclusion of physics-informed problems reduces the standard deviation in most cases, demonstrating increased reliability.

References

[1] M.G. Larson, F. Bengzon, The finite element method: theory, implementation, and applications, in: Texts in Computational Science and Engineering, vol. 10, Springer, Berlin, Heidelberg, 2013, http://dx.doi.org/10.1007/978-3-642-33287-6.

[2] F. Moukalled, L. Mangani, M. Darwish, The finite volume method in computational fluid dynamics: an advanced introduction with OpenFOAM® and matlab, in: Fluid Mechanics and Its Applications, vol. 113, Springer International Publishing, Cham, 2016, http://dx.doi.org/10.1007/978-3-319-16874-6.

[3] M.N. Özişik, H.R.B. Orlande, M.J. Colaço, R.M. Cotta, Finite Difference Methods in Heat Transfer, CRC Press, 2017.

[4] S.P. Venkateshan, P. Swaminathan, Computational Methods in Engineering, Springer International Publishing, Cham, 2023, http://dx.doi.org/10.1007/978-3-031-08226-9.

[5] C. Zimmerling, C. Poppe, O. Stein, L. Kärger, Optimisation of manufacturing process parameters for variable component geometries using reinforcement learning, Mater. Des. 214 (2022) 110423, http://dx.doi.org/10.1016/j.matdes.2022.110423.

[6] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-Net: Learning PDEs from data, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.

[7] J. Pfrommer, C. Zimmerling, J. Liu, L. Kärger, F. Henning, J. Beyerer, Optimisation of manufacturing process parameters using deep neural networks as surrogate models, Procedia CIRP 72 (2018) 426–431, http://dx.doi.org/10.1016/j.procir.2018.03.046.

[8] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco California USA, 2016, pp. 481–490, http://dx.doi.org/10.1145/2939672.2939738.

[9] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2) (2019) 525–545, http://dx.doi.org/10.1007/s00466-019-01740-0.

[10] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2020.

[11] M. Horie, N. Mitsume, Physics-embedded neural networks: graph neural PDE solvers with mixed boundary conditions, Adv. Neural Inf. Process. Syst. 35 (2022) 23218–23229.

[12] Z. Li, K. Meidani, A.B. Farimani, Transformer for partial differential equations' operator learning, Trans. Mach. Learn. Res. (2022).

[13] J. Brandstetter, D.E. Worrall, M. Welling, Message Passing Neural PDE Solvers, in: International Conference on Learning Representations, 2021.

[14] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: Proceedings of the 37th International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.

[15] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural Ordinary Differential Equations, in: Advances in Neural Information Processing Systems, vol. 31, Curran Associates, Inc., 2018.

[16] V. Iakovlev, M. Heinonen, H. Lähdesmäki, Learning continuous-time PDEs from sparse data with graph neural networks, in: International Conference on Learning Representations, 2020.

[17] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364, http://dx.doi.org/10.1016/j.jcp.2018.08.029.

[18] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[19] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732, http://dx.doi.org/10.1016/j.cma.2019.112732.

[20] S. Amini Niaki, E. Haghighat, T. Campbell, A. Poursartip, R. Vaziri, Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture, Comput. Methods Appl. Mech. Engrg. 384 (2021) 113959, http://dx.doi.org/10.1016/j.cma.2021.113959.

[21] T. Würth, C. Krauß, C. Zimmerling, L. Kärger, Physics-informed neural networks for data-free surrogate modelling and engineering optimization – An example from composite manufacturing, Mater. Des. 231 (2023) 112034, http://dx.doi.org/10.1016/j.matdes.2023.112034.

[22] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079, http://dx.doi.org/10.1016/j.jcp.2020.110079.

[23] P. Ren, C. Rao, Y. Liu, J.-X. Wang, H. Sun, PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114399, http://dx.doi.org/10.1016/j.cma.2021.114399.

[24] X. Zhao, Z. Gong, Y. Zhang, W. Yao, X. Chen, Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data, Eng. Appl. Artif. Intell. 117 (2023) 105516, http://dx.doi.org/10.1016/j.engappai.2022.105516.

[25] A. Kashefi, T. Mukerji, Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries, J. Comput. Phys. 468 (2022) 111510, http://dx.doi.org/10.1016/j.jcp.2022.111510.

[26] H. Gao, M.J. Zahr, J.-X. Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502, http://dx.doi.org/10.1016/j.cma.2021.114502.

[27] A. Li, Y.J. Zhang, Isogeometric analysis-based physics-informed graph neural network for studying traffic jam in neurons, Comput. Methods Appl. Mech. Engrg. 403 (2023) 115757, http://dx.doi.org/10.1016/j.cma.2022.115757.

[28] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, vol. 29, Curran Associates, Inc, 2016.

[29] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, R. Pascanu, Relational inductive biases, deep learning, and graph networks, 2018, http://dx.doi.org/10.48550/arXiv.1806.01261, arXiv:1806.01261.

[30] N. Freymuth, P. Dahlinger, T. Würth, S. Reisch, L. Kärger, G. Neumann, Swarm Reinforcement Learning For Adaptive Mesh Refinement, 2023, http://dx.doi.org/10.48550/arXiv.2304.00818, arXiv:2304.00818.

[31] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[32] R. Bischof, M.A. Kraus, Multi-objective loss balancing for physics-informed deep learning, 2023, http://dx.doi.org/10.2139/ssrn.4596537, Preprint, SSRN.

[33] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, Stat 1050 (2016) 21.
[34] T. Gustafsson, G. McBain, Scikit-fem: A Python package for finite element assembly, J. Open Source Softw. 5 (52) (2020) 2369, http://dx.doi.org/10.21105/joss.02369.
[35] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, Q. Gu, Towards understanding the spectral bias of deep learning, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, 2021, pp. 2205–2211, http://dx.doi.org/10.24963/ijcai.2021/304.
[36] U.B. Waheed, Kronecker neural networks overcome spectral bias for PINN-based wavefield computation, IEEE Geosci. Remote Sens. Lett. 19 (2022) 1–5, http://dx.doi.org/10.1109/LGRS.2022.3209901.