Proceedings of the Seventh International Conference on
Artificial Intelligence, Soft Computing, Machine Learning and Optimization,
in Civil, Structural and Environmental Engineering
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 11, Paper 1.4
Civil-Comp Press, Edinburgh, United Kingdom, 2025
ISSN: 2753-3239, doi: 10.4203/ccc.11.1.4

Enhancing Information Flow in Graph Neural Networks for Scientific Machine Learning

M. Chenaud[1,2], J. Alves[2] and F. Magoulès[1]

[1] MICS, CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, Île-de-France, France
[2] Transvalor S.A, Biot, France

Abstract

The propagation of information through discretised domains is of crucial importance in the field of scientific machine learning. Recent studies have demonstrated the efficacy of graph-based models for physical simulations, particularly due to the inductive biases inherent in such frameworks. However, ensuring efficient information flow through these graph architectures is a delicate aspect, due to the wide range of scales of the simulated phenomena. We summarise some key architectural choices that are the most prominent in the literature, and we propose a novel edge augmentation technique, based on farthest point sampling and the Möller-Trumbore algorithm, for highly non-convex geometries. The efficacy of our approach is demonstrated through the training of a graph neural network model on a challenging, non-convex geometry.

Keywords: scientific machine learning, scientific computing, graph neural networks, Möller-Trumbore algorithm, inductive bias, physics-informed neural networks

1 Introduction

Over the last decade, the application of deep learning algorithms for physical simulations has received a growing interest, in various fields such as solid and fluid mechanics [3, 21], weather prediction [14, 22], biology [29, 32] and many others. While classical numerical simulations, such as the finite element method [24], are widely known for their performance in simulating complex, multiphysics phenomena [1] and for their ability to benefit from the most recent advances in high performance computing [17], some problems are still unreachable for these techniques. While training data is critical for accurate machine-learning models, recently, physics-informed machine learning has emerged [12,23]. In this setting, the models are directly trained to respect physical properties and equations, instead of relying on costly training data.

A well-suited model architecture is key to ensure a robust and accurate prediction of physical phenomena, and to guarantee that the trained model exhibits the same invariances and equivariances as the physical phenomenon in itself [2,3]. For such applications, graph neural networks seem to be the most promising set of models, with state of the art results in a wide range of applications [4, 21, 22, 27, 31]. This success can be explained by their inductive properties, but also by their similarity with classical numerical solvers such as the finite element method. The mesh-based approaches of such techniques are strongly similar to a graph structure. Therefore, instead of opposing finite element solvers and machine learning, many works have been focusing on finding a synergy between these two approaches [5, 16, 26]. Crucially, discrete fields built on a mesh are therefore passed to graph neural networks, which then propagate such information on the whole mesh through message-passing (MP) steps [8,9].

These meshes, arising from discretised physical domains, possess unique structural properties that challenge standard message-passing frameworks. Therefore, efficient information propagation over mesh-based geometries is critical to the success of Graph Neural Networks (GNNs) in physical applications. The recent performances of graph-based approaches prove that such structures can be addressed efficiently. However, there appears to be a lack of systematic discussion on the main architectural choices for dealing with mesh-like graph structures.

The paper is organised as follows. First, in section 2, the most classical techniques for building a graph connectivity in mesh-based simulations are presented. A categorization of these techniques is introduced, and some of their key limitations on challenging, irregular and non-convex domains are demonstrated. In section 3, we propose a novel edge-augmentation procedure leveraging geometrical tools such as ray-triangle intersection algorithms, to extend the previously mentioned techniques to more challenging domains. While such geometrical tools have been investigated in the context of finite element methods [18], to the best of our knowledge, an extension to graph neural networks has never been presented. Finally, in section 4, we illustrate

the capacity of our method on a highly non-convex domain.

2 Information propagation techniques in mesh-based graph structures

Although graph networks are proven to be very efficient in a wide range of tasks, their locality bias, and the way message-passing is implemented, limits their ability to simulate long-range interactions. A message-passing block propagates information from one node to its direct neighbours, therefore the range for which the information flows through a GNN is equal to its number of blocks, i.e its number of message-passing steps on a forward pass. For a detailed explaination of information propagation through graph blocks, see, for instance, [2, 28].

Typical mesh structures can have a diameter of a few thousand, therefore, a model would need a few thousand blocks to propagate information through the whole geometry. Because of the complexity of such models, their number of blocks is usually in the order of magnitude of 10. Using the mesh structure directly as input to the model is therefore impossible in most non-trivial scenarios. Enhancing the connectivity of the mesh is therefore required. Note that most of the current works use a radius graph, where each node is linked to all of the other nodes inside a given radius. However, this improved connectivity is still not sufficient for complex, multi-scale phenomena.

In this section, we provide an overview of existing techniques designed to facilitate information flow across such graph representations of mesh-based geometries. We will focus on three main edge augmentation approaches, that seem to receive the most attention in the recent years: multi-scale graph neural networks, multi-mesh message passing, and edge augmentation through random sampling.

2.1 Multi-scale graph neural networks

Similar to U-Nets in computer vision [25], or multigrid approaches in finite element methods [24, 30], the principle of multi-scale graph neural networks, also known as graph u-nets [7, 15] is to store hierarchical discretizations of the domain, from coarse, high-level geometric discretizations to more refined and precise meshes. Figure 1 illustrates the key components of the multi-scale message passing.

This structure is very efficient to encapsulate multi-scale phenomena, since each level of refinement can propagate information to a different scale. However, this comes with an added cost in terms of implementation and numerical complexity, since it necessitates the implementation of projection steps from one mesh to another. The order in which the information flows from one hierarchy of mesh to another can vary, similar to V and W cycles in multigrid finite element method [30].

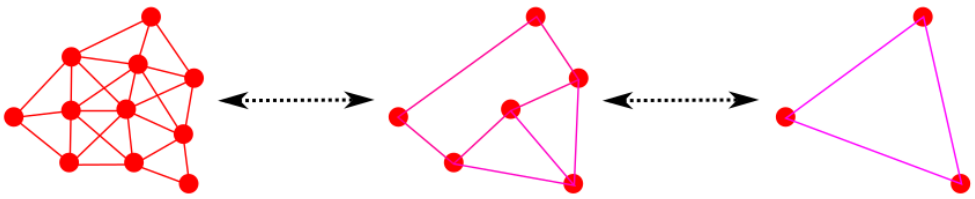

Figure 1: Multi-scale architecture. Several hierarchical refinements are stored, the message-passing is performed sequentially within each mesh, and then projected to the coarser or finer mesh.

2.2 Multi-mesh message-passing

Several prominent works proved that while multi-scale graph neural networks are efficient, a simpler solution may be used: multi-mesh message passing. The idea is similar, with hierarchical refinements of the same discretized geometries. However, instead of separating the different refinements, the mid-range and long-range edges that are built on the coarser meshes are simply added to the fine mesh. Therefore, the model only propagates through a single graph, where the edges from all the hierarchies have been added. The choice of the edge attributes is critical in this case, to ensure that the hierarchy between the edges, and the different spacial ranges that are represented, are well taken into account by the model. Usually, for physical applications, the attribute of an edge is a function of the relative position between the two corresponding nodes, and of the geometric distance between these two nodes. This choice ensures that the resulting model will contain interesting inductive biases [2], which are crucial for Physics-based applications. Figure 2 illustrates the resulting mesh.

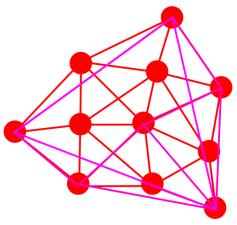

Figure 2: Multi-mesh architecture. The edges of the finer and coarser meshes are concatenated in a single graph, and the message-passing step occurs directly on the full graph.

This approach is much simpler than the multi-scale message-passing, due to the fact that the projections between the different mesh hierarchies are no longer needed.

One could think that, similar to computer vision, this simplification would limit the model's ability to simulate complex, multi-scale physical phenomena, however recent results, in particular regarding weather prediction [14, 22], tend to prove the opposite. The internal representations of a graph neural network, provided that the architecture of the graph is well suited, seem to be able to encapsulate the multi-scale phenomena in a single graph, without the added complexity of handling projections between the different refinement levels.

2.3 Random edge creation

The authors of [10] have introduced an edge-augmentation technique even simpler than the previously mentioned multi-mesh. The idea is to start from a mesh, and to add, at random, edges between any pair of nodes of the geometry. This implementation does not necessitate to coarsen any given mesh, and therefore the preprocessing complexity is significantly reduced. The authors show that in their example, adding 20% of edges compared to the original connectivity was sufficient to ensure accurate results, and to beat other methods such as the multi-scale graph network. Figure 3 illustrates this approach.

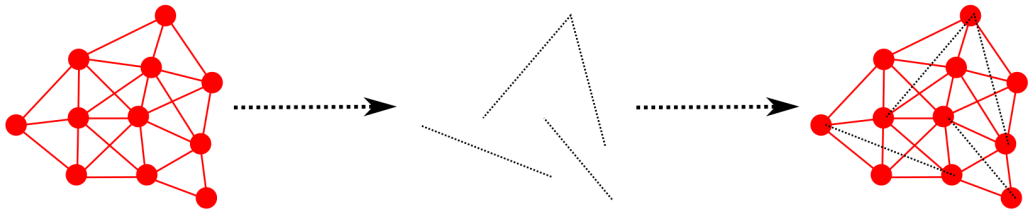

Figure 3: Random edge augmentation. The mesh structure is kept intact, and a predefined number of edges are added at random between nodes (dashed black lines).

This idea seems to be the obvious best solution, since it is by far the more simple in terms of preprocessing complexity, and it yields better results. However, we identify two drawbacks to this framework. The first one is that while, for convex domains, any added edge is guaranteed to stay inside the domain, it is not true for more complex geometries. Therefore, an added edge may break the topology of the mesh, and add a link between nodes that should be unrelated. This observation is also true for the two previous approaches, but even more so when the added edges are chosen at random, and therefore are not extracted from a coarsened version of the initial geometry. The second main drawback of this approach is a consequence of the following proposition.

Proposition 1 *Let $\mathcal{G}$ be an undirected graph with n vertices. Suppose that the edges of $\mathcal{G}$ are built at random. Let d be a positive integer. Then, for $\mathcal{G}$ to have a diameter of*

d with probability 1*, when* n *goes to infinity, the number of edges of* $\mathcal{G}$*, that we denote* $E(n)$*, must verify*

$$n^{1+\frac{1}{d}} \log^{\frac{1}{d}}(n) \ll E(n) \ll n^{1+\frac{1}{d-1}}. \tag{1}$$

Proof 1 *This result is a direct consequence of [13, Corollary].*

The applications provided by the authors of [10] typically have around 500 nodes, and they use 6 message-passing blocks. In order for the diameter of a random graph of 500 nodes to be less or equal to 6, one would need around 1700 edges according to theorem 1. If we generalize this result to a two-dimensional mesh with 500 nodes, and an average number of edges of 3 times the number of nodes, one would need to add around 200 edges, i.e around 15 % of the total. This reasoning is only an estimation, since proposition 1 only applies for random graphs, and not for mesh-like graphs in which random edges are added, but it still justifies the choice of 20% added edges. While an increase of 20% in the number of edges is still reasonable, the scaling in $\mathcal{O}\left(n^{1+\frac{1}{d}}\right)$ suggested by proposition 1 quickly becomes prohibitive when the number of nodes increases. The randomness of the added edges may not be the most efficient choice to reduce the graph diameter, and more optimal techniques, such as farthest point sampling, may be of interest. The next section details our proposed approach, based on this observation.

3 Proposed approach: Multi-mesh message-passing for non-convex geometries

The proposed edge-augmentation procedure is the following. First, a mesh is given as input. The mesh nodes are denoted $\{x_s\}_{s\in\Omega}$. The mesh connectivity in itself is not used for the edges, but for the extraction of geometric operators. From this mesh, the boundary faces $\mathcal{F}$ and their normal vectors $\mathcal{N}$ are extracted. Several key hyper-parameters need to be decided. The edges will be built with radius graph approaches, therefore the maximum number of neighbors by node $N_{\text{neighbors}}$, and the corresponding radius r_{short} need to be fixed. Mid-range and long-range edges will also be added, therefore the corresponding radius $r_{\text{medium}}, r_{\text{long}}$ should be also decided beforehand. The mid and long range edges are also built by radius graphs, but not for every node of the mesh, to limit the computation complexity. Therefore, $N_{\text{mid\_nodes}}$ and $N_{\text{long\_nodes}}$ nodes are sampled by farthest point sampling through the mesh nodes, and these nodes are then used to build the corresponding mid-range and long-range edges. The use of farthest point sampling is more efficient than pure random sampling, since it guarantees that the whole domain is covered.

After this procedure, three levels of edges are present. The initial radius graph ensures information flow on the short scale, and the mid and long scale propagation is ensured by the other added edges. Note that three levels are presented here, but the

same procedure could be extended to more levels. Once this connectivity has been obtained, the full graph may not respect the initial mesh topology, some edges may be present outside of the domain. To remove these undesirable edges, and to ensure that information flows through the mesh topology and not according to euclidean geometry, a filtering layer has been added. This filter removes the undesirable edges, with a simple procedure: for a given edge, if it goes through a boundary face, then it goes through the domain, therefore it must be removed. This test has been implemented with the Möller-Trumbore algorithm [19] because of its efficiency. This method is a ray-triangle intersection test, therefore directly useful in our case. For a given edge, a function INTERSECTSFACE extracts the boundary faces that are close to this edge, and verifies if this edge goes through one of these faces following the Möller-Trumbore procedure. This function takes as input the position of the edge, the set of boundary faces and the related normal vectors. For an in-depth presentation of the Möller-Trumbore algorithm, see [19].

Algorithm 1 summarizes the key steps of the proposed procedure. In addition to the INTERSECTSFACE function, a FARTHESTPOINTSAMPLING function is also used. This function takes as input the desired number of points, and samples this number of points from the initial mesh $\{x_s\}_{s\in\Omega}$ by farthest point sampling. Finally, we use the function RADIUSGRAPH, which is a modification of the `radius_graph` function of the PyTorch Geometric Python package [6]. Our version of RADIUSGRAPH takes as input sampled nodes, the number of neighbors for each of this sampled nodes, and two radiuses: a shorter and a longer one. Instead of sampling the edges on a ball centered on a point, here, the sampled edges must have a length included between the shorter and the longer radius. This ensures that the added mid and long range edges have the desired lengths.

Figure 4 illustrates our added edges on a complex, 3 dimensional geometry, with and without the Möller-Trumbore check. The geometry chosen, representing the Olympic rings, is highly non-convex, and it is made of 76659 nodes. With no check, edges are added outside of the domain, therefore modifying its topology and introducing non-physical shortcuts between parts of the geometry. On the contrary, with filtering layer, the added edges seem to stay inside the domain.

This procedure has the drawback that the number of added edges cannot be directly controlled, it is dependent on the number of invalid edges removed by the algorithm. While in our experiments, we found that this limitation was not too restrictive, it is possible to modify the algorithm 1 to add a corresponding number of valid edges.

4 Application on a three-dimensional, non-convex domain

In order to test the proposed edge augmentation algorithm, a challenging two-dimensional geometry representing the Olympic rings has been selected. Because of its non-

Algorithm 1 Mesh-Based Radius Graph Construction with Möller-Trumbore Filtering

Initialization:
  Input: $\{x_s\}_{s\in\Omega}$ ▷ Mesh nodes
  Input: $r_{\text{short}}, r_{\text{medium}}, r_{\text{long}}$ ▷ Short, medium, and long radius
  Input: $N_{\text{mid\_nodes}}, N_{\text{long\_nodes}}, N_{\text{neighbors}}$ ▷ Node and neighbor parameters

Preprocessing:
  Extract mesh faces $\mathcal{F}$ and face normals $\mathcal{N}$
  $\{x_s\}_{s\in S_{\text{mid}}} \leftarrow \text{FARTHESTPOINTSAMPLING}(N_{\text{mid\_nodes}})$
  $\{x_s\}_{s\in S_{\text{long}}} \leftarrow \text{FARTHESTPOINTSAMPLING}(N_{\text{long\_nodes}})$

Processing:
  $E_{\text{short}} \leftarrow \text{RADIUSGRAPH}(\{x_s\}_{s\in\Omega}, 0, r_{\text{short}}, N_{\text{neighbors}})$
  $E_{\text{medium}} \leftarrow \text{RADIUSGRAPH}(\{x_s\}_{s\in S_{\text{mid}}}, r_{\text{short}}, r_{\text{medium}}, N_{\text{neighbors}})$
  $E_{\text{long}} \leftarrow \text{RADIUSGRAPH}(\{x_s\}_{s\in S_{\text{long}}}, r_{\text{medium}}, r_{\text{long}}, N_{\text{neighbors}})$
  $E_{\text{total}} \leftarrow \text{MERGE}(E_{\text{short}}, E_{\text{medium}}, E_{\text{long}})$

Filtering:
for all $e \in E_{\text{total}}$ **do**
  if $\text{INTERSECTSFACE}(e, \mathcal{F}, \mathcal{N})$ **then**
    Remove e from E_{total}
  end if
end for

convexity and its high number of nodes, random edge augmentation would not be straightforward, since it would introduce non-physical edges that would not respect the topology of the considered domain. The target field that has been chosen is a solution to the static heat problem, where the heat source is located in an extremity of the domain. The target solution has been computed by the commercial finite element solver Forge ® [1]. Note that the chosen domain is a two-dimensional cut plane of the previous case, presented in figure 4, therefore this case exhibits the same challenges in terms of non-convexity. The resulting mesh is made of 5104 nodes, and this simplifications allows to compute the graph diameter with `networkx` [11], to assess the performance of the different edge-augmentation techniques to decrease the graph diameter. The restriction of the Möller-Trumblore algorithm to two-dimensional cases is straightforward. Figure 5 illustrates the selected mesh, along with the target and input fields.

To assess the capability of the models to propagate the information, the tested models are given as input the target field on half the geometry, a quarter on both sides. The models are assessed on the remaining half, in the center. The models are trained to minimize the mean squared error between the predicted and target fields on the whole domain. The GNN chosen is an encoder-processor-decoder, similar to [21]. The encoders, decoders and edge and node processors are Multi-Layer Perceptrons (MLP) with two hidden layers of width 64 and the Tanh activation function. The edge and

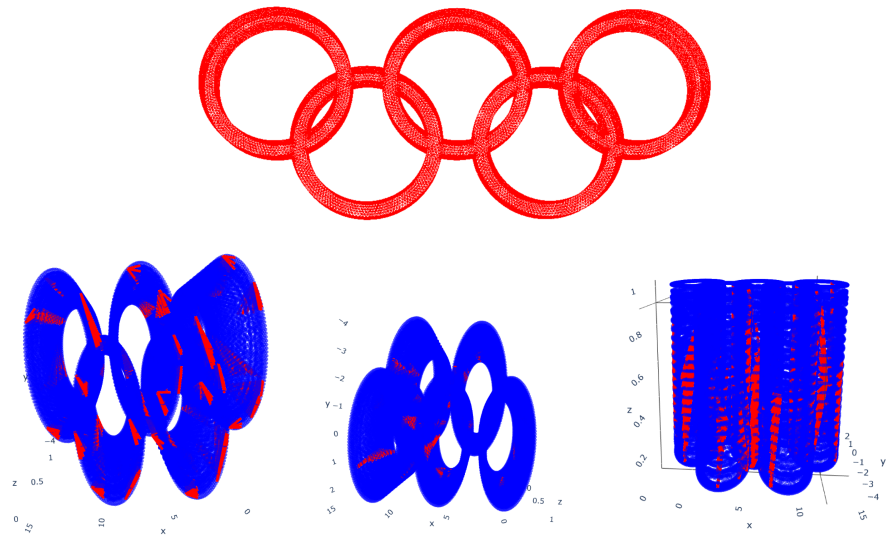

Figure 4: (top) Initial meshed geometry. (left) Added edges by farthest point sampling (in red), with no previous check for valid edges with the Möller-Trumbore algorithm. (middle, right) Added edges by farthest point sampling (in red). The edges that do not stay inside the domain are identified with the Möller-Trumbore algorithm, and removed.

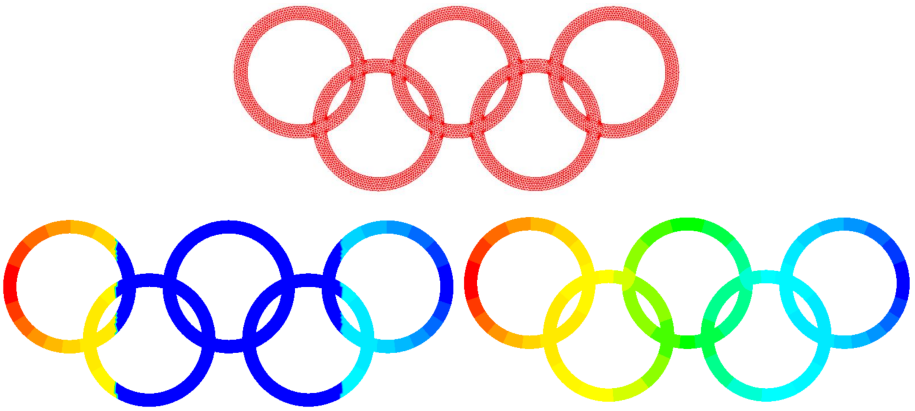

Figure 5: Selected case. (top) Initial meshed domain. (left) Field given as input to the tested models. (right) Target field.

node processors also have residual connections. The edge attributes are relative node positions and the euclidean distance between the two nodes. All of the models have been trained for 250 epochs with the Adam optimizer, and 50 epochs of the L-BFGS optimizer in PyTorch [20]. Table 1 gathers the relative errors made by the tested models after training. 'MLP' refers to a simple MLP of width 64 and depth 3. 'mesh GNN' refers to a GNN trained on the initial edge connectivity of the mesh. 'radius GNN' is a GNN trained on a radius graph, where the radius has been chosen to be

equal to one-tenth of the span of the whole geometry. The maximal number of edges by nodes is set to 25. 'GNN - ours' refers to our model, the same radius graph but with additional nodes sampled by farthest point sampling, from which radius graphs with higher radius values are built. 100 mid-range and long-range nodes are sampled.

Model	Graph Diameter	# MP blocks	Relative error (%)
MLP	∞	-	3.1
mesh GNN	232	5	2.6
mesh GNN	232	10	2.5
radius GNN	13	5	3.1
radius GNN	13	10	2.1
GNN - ours	7	5	**0.12**
GNN - ours	7	10	0.31

Table 1: Relative error made by each variant of the GNN and MLP after training. The GNN architecture does not change, except for the number of message-passing blocks, but the graph connectivity is changed. The corresponding graph diameter is reported in each case.

It is of interest to note that the diameter is almost divided by two with our approach, with less than 4% more edges compared to the radius graph. The proposed edge augmentation procedure allowed the tested GNNs to reach the smallest relative error. The results for 5 and 10 message-passing blocks are similar, which is coherent with the graph diameter equal to 7. Figure 6 shows the predicted field of the 5 blocks GNN with our edge augmentation technique. Even if the predicted field is not perfectly smooth, the models trained on this graph connectivity were able to propagate the information throughout the whole domain, due to the reduced diameter. The computational complexity of our approach is on par with the radius graph-based models, since the added edges are optimized through farthest point sampling to cover the whole geometry.

5 Concluding remarks

Graph neural networks are ubiquitous in scientific machine learning. Their ability to simulate complex physical phenomena with suitable inductive biases paves the way to numerous applications. In this framework, graph connectivity is a sensitive topic that can greatly impact the model's performance. After presenting an overview of different edge augmentation techniques for mesh-based graph structures, some limitations of the current approaches have been highlighted, in particular when dealing with non convex geometries. A novel method, based on geometric tools, is presented to address

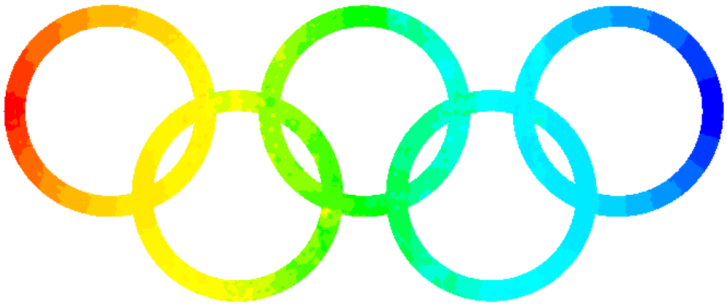

Figure 6: Predicted field, GNN with 5 message-passing blocks and our enhanced graph connectivity. The initial field has been well diffused throughout the whole domain. The relative error is 0.12%.

this issue. The numerical experiment that is conducted proves the ability of our edge-augmented graph neural network to propagate a physical field through challenging geometries.

References

[1] J Alves, S Acevedo, Stephane Marie, Bernhard Adams, Katia Mocellin, and François Bay. Numerical modeling of electrical upsetting manufacturing processes based on forge® environment. In *AIP conference proceedings*, volume 1896. AIP Publishing, 2017.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] M Chenaud, F Magoulès, and J Alves. Physics-informed graph-mesh networks for pdes: A hybrid approach for complex problems. *Advances in Engineering Software*, 197:103758, 2024.

[4] Marien Chenaud, José Alves, and Frédéric Magoulès. Physics-informed graph convolutional networks: Towards a generalized framework for complex geometries. *arXiv preprint arXiv:2310.14948*, 2023.

[5] Mohammad Sadegh Eshaghi, Cosmin Anitescu, Manish Thombre, Yizheng Wang, Xiaoying Zhuang, and Timon Rabczuk. Variational physics-informed neural operator (vino) for solving partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 437:117785, 2025.

[6] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[7] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Message passing neural networks. In *Machine learning meets quantum physics*, pages 199–214. Springer, 2020.

[10] Rini Jasmine Gladstone, Helia Rahmani, Vishvas Suryakumar, Hadi Meidani, Marta D'Elia, and Ahmad Zareei. Mesh-based gnn surrogates for time-independent pdes. *Scientific reports*, 14(1):3394, 2024.

[11] Aric Hagberg and Drew Conway. Networkx: Network analysis with python. *URL: https://networkx. github. io*, pages 1–48, 2020.

[12] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[13] Victor Klee and David Larman. Diameters of random graphs. *Canadian Journal of Mathematics*, 33(3):618–640, 1981.

[14] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[15] Maosen Li, Siheng Chen, Yangheng Zhao, Ya Zhang, Yanfeng Wang, and Qi Tian. Dynamic multiscale graph neural networks for 3d skeleton based human motion prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 214–223, 2020.

[16] Lawson Oliveira Lima, Julien Rosenberger, Esteban Antier, and Frédéric Magoulès. Multilayer perceptron-based surrogate models for finite element analysis. In *2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 115–118. IEEE, 2022.

[17] Frédéric Magoules and Abal-Kassim Cheik Ahamed. Alinea: An advanced linear algebra library for massively parallel computations on graphics processing units. *The International Journal of High Performance Computing Applications*, 29(3):284–310, 2015.

[18] Frederic Magoules, Guillaume Gbikpi-Benissan, and Patrick Callet. Ray-tracing domain decomposition methods for real-time simulation on multi-core and multi-processor systems. *Concurrency and Computation: Practice and Experience*, 28(16):4352–4364, 2016.

[19] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es. 2005.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[22] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

[23] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[24] Junuthula Narasimha Reddy. An introduction to the finite element method. *New York*, 27(14), 1993.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[26] Esteban Samaniego, Cosmin Anitescu, Somdatta Goswami, Vien Minh Nguyen-Thanh, Hongwei Guo, Khader Hamdia, X Zhuang, and T Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020.

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

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[29] Ahmet Sen, Elnaz Ghajar-Rahimi, Miquel Aguirre, Laurent Navarro, Craig J Goergen, and Stephane Avril. Physics-informed graph neural networks to solve 1-d equations of blood flow. *Computer methods and programs in biomedicine*, 257:108427, 2024.

[30] Vladimir Viktorovich Shaidurov. *Multigrid methods for finite elements*, volume 318. Springer Science & Business Media, 2013.

[31] Meduri Venkata Shivaditya, José Alves, Francesca Bugiotti, and Frederic Magoules. Graph neural network-based surrogate models for finite element analysis. In *2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 54–57. IEEE, 2022.

[32] Alireza Yazdani, Lu Lu, Maziar Raissi, and George Em Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS computational biology*, 16(11):e1007575, 2020.