POLITECNICO DI TORINO
Repository ISTITUZIONALE

TCAD assisted design of the Doherty Power Amplifier

Original
TCAD assisted design of the Doherty Power Amplifier / Guerrieri, S.D., Catoggio, E., Bonani, F.. - In: SOLID-STATE ELECTRONICS. - ISSN 0038-1101. - ELETTRONICO. - 207:(2023). [10.1016/j.sse.2023.108684]

Availability:
This version is available at: 11583/2979125 since: 2023-11-19T08:20:46Z

Publisher:
Elsevier

Published
DOI:10.1016/j.sse.2023.108684

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

Predicting flux in Discrete Fracture Networks via Graph Informed Neural Networks

Stefano Berrone
Politecnico di Torino
stefano.berrone@polito.it

Francesco Della Santa
Politecnico di Torino
francesco.dellasanta@polito.it

Antonio Mastropietro
Politecnico di Torino
Addfor Industriale, Turin
antonio.mastropietro@polito.it

Sandra Pieraccini
Politecnico di Torino
sandra.pieraccini@polito.it

Francesco Vaccarino
Politecnico di Torino
francesco.vaccarino@polito.it

Abstract

Discrete Fracture Network (DFN) flow simulations are commonly used to determine the outflow in fractured media for critical applications. Here, we extend the formulation of spatial graph neural networks with a new architecture, called Graph Informed Neural Network (GINN), to speed up the Uncertainty Quantification analyses for DFNs. We show that the GINN model allows better Monte Carlo estimates of the mean and standard deviation of the outflow of a test case DFN.

1 Introduction: Uncertainty Quantification for Discrete Fracture Networks

The determination of flow and transport in underground fractured media is a problem of interest for several critical modern applications concerning civil, environmental and industrial engineering. A convenient model to represent flow in fractured media is represented by Discrete Fracture Networks (DFNs) [1, 2, 3]. In these models each fracture of the network is represented by a planar polygon into a 3-dimensional domain and it is characterized by its own hydro-geological and geometrical features (namely: position, size, orientation, fracture transmissivity, etc.). The flux exchanges between fractures occur along the segments of fractures intersections, denoted as traces. The assumptions about the used DFN model are: (i) the rock matrix surrounding the polygonal assembly is impenetrable; (ii) the flux propagation on each fracture is ruled by the Darcy law; (iii) finally, at all fracture intersections head continuity and flux balance are assumed. The present assumptions are correct for fractures that are porous media with a permeability much larger than the permeability of the surrounding rock matrix.

For DFN, fractures are commonly described by sampling their hydrogeological and geometrical features from given distributions, because the knowledge of hydrogeological and geometrical fractures parameters is usually not available. Due to the amount of uncertainty in the representation of DFNs,

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

flow and transport in a real fractured medium are studied from a statistical point of view, resorting to Uncertainty Quantification (UQ) analyses. These UQ analyses are likely to involve thousands of DFN simulations; in order to speed up these analyses, it is worth considering some sort of complexity reduction techniques. Indeed, flow simulations in DFNs are likely to be quite challenging problems, due to several issues as the size of realistic networks or the geometrical complexity of the computational domain. These properties make the meshing process an hard task whenever conforming meshes are needed. In recent literature several new methods have been proposed which use different strategies for circumventing meshing problems, either partially or totally. For the DFN simulations of this work, we follow the approach described in [4, 5, 6], consisting in a reformulation of the problem as a PDE-constrained optimization problem, in which the need for conforming mesh is completely overcome. However, the computational burden of simulations on a realistic DFNs is still expensive, and it may be prohibitive for the large number of simulations required for UQ analyses.

The cost of these analyses necessarily put the focus on model reduction methods to speed up the simulations. Machine Learning (ML) has recently attracted plenty of attention in several frameworks related to the aformentioned problems. In recent contributions [7], ML techniques have been applied to DFN flow simulations in conjunction with graph-based methods. Morover, Neural Networks (NNs) have been applied in a UQ framework by [8]: given a stochastic elliptic PDE with uncertain diffusion coefficient, the authors construct a NN as surrogate model to replace the forward model solver, while performing Monte Carlo (MC) simulations for UQ.

An interesting approach for DFNs is the one used in [9, 10]. Similarly to [8], the approach consists of the usage of a NN-based model for the UQ analyses, instead of DFN simulations. In particular, the DFN simulations are used to build an accurate training dataset; then, a NN model is trained on the dataset to provide a surrogate model; finally, the NN model is used to build a larger set of approximated DFN simulation results useful for the UQ analyses in a negligible amount of time.

In this paper, we describe a new graph neural network model, which is based on the mathematical formulation of a so-called *Graph Informed* (GI) layer. For DFNs, the idea is to take advantage of their graph representation, i.e., of the relationships between the fractures (nodes) through the traces (edges). The layer formulation extends the previous formulations of spatial graph layers, as classified by the survey paper [11]. The GI layers are inspired by convolutional layers and they prove to be useful for regression tasks with respect to graph-structured features and targets, such as the flux regression problem in DFNs. We name the new neural networks characterized by GI layers as *Graph Informed Neural Networks* (GINNs), because they incorporate the knowledge about the graph adjacency matrix in the neuronal links. Finally, we show that the MC estimates for UQ obtained using the enlarged dataset provided by the GINN predictions returns better MC moment estimates of the DFN fluxes than the ones obtained using the simulations of the training dataset.

2 Graph Informed Layer for neural networks

The main idea of spatial graph neural networks [11] is to generalize the properties of "sparse interaction" and "parameter sharing" of convolutional layers to graph-structured features. Let us consider a directed graph $G = (V, E)$ of n nodes without self-loops, with adjacency matrix $A \in \mathbb{R}^{n\times n}$. Given a filter $\boldsymbol{w} \in \mathbb{R}^n$ such that w_j is the weight associated to node $v_j \in V$, the output feature of node $v_i \in V$ with respect to a GI layer of weights $\boldsymbol{w}$ is

$$x'_i = f\Big(\sum_{j\in \mathrm{N}_{in}(i)\cup\{i\}} x_j\, w_j + b_i \Big), \tag{1}$$

where x_j is the feature associated to node v_j, $\mathrm{N}_{in}(i) = \{j \in V \mid (j,i) \in E\}$ is the index-neighbourhood of node v_i, b_i is the bias associated to v_i, and f the activation function. Equation (1) holds also for undirected graphs: any undirected edge $\{v_j, v_i\}$ is equivalent to two directed edges (v_j, v_i) and (v_i, v_j). A GI layer characterized by (1) can also be interpreted as a "constrained" fully-connected layer with weights $w_{ji} = w_j$ if $(v_j, v_i) \in E$; $w_{ji} = w_i$ if $j = i$; null otherwise.

The above formulation can be stated in matrix form for all the node features. Let $\widehat{A}$ be $\widehat{A} := A + \mathbb{I}_n$, with $\mathbb{I}_n \in \mathbb{R}^{n\times n}$ identity matrix. Then, a GI layer L^{GI} is described by a function $\mathcal{L}^{GI} : \mathbb{R}^n \to \mathbb{R}^n$

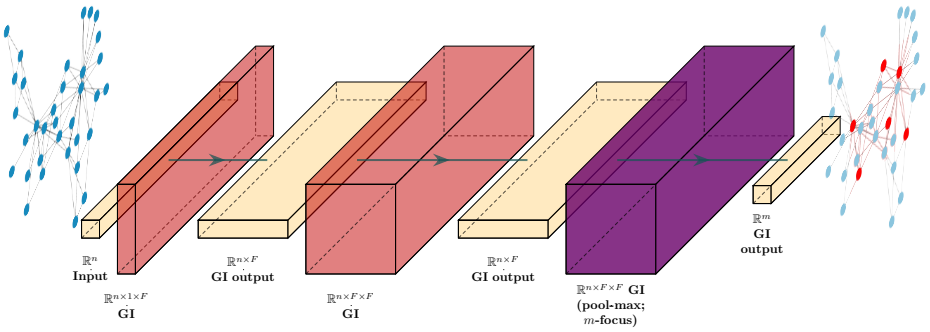

Figure 1: Archetipe of the GINN architectures. Red for weight tensors of the hidden GI layers, orange for layers output tensors, and purple for the output layer weights with max-pooling and masking.

defined as

$$\mathcal{L}^{GI}(\boldsymbol{x}) = \boldsymbol{f}\left(\widehat{W}^\top \boldsymbol{x} + \boldsymbol{b}\right), \quad \text{such that } \widehat{W} := \widehat{A} \odot \underbrace{\left[\boldsymbol{w}, \cdots, \boldsymbol{w}\right]}_{n \text{ times}}, \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of input features, $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector, $\odot$ is the Hadamard product, $\boldsymbol{f}$ is the element-wise application of the activation function, and $\boldsymbol{b} \in \mathbb{R}^n$ is the bias vector.

A careful reader can observe that equation (2) partially resemble the NN4G layer (see [11, sec. V.B.] and [12]), without residual nor skip connections. Nonetheless, our formulation of equation (2) can be generalized in tensor form to multiple input features $K \in \mathbb{N}$ per node and multiple filters $F \in \mathbb{N}$ (i.e., output features per node). Let $\widehat{\boldsymbol{x}} \in \mathbb{R}^{nK}$ be the vectorization of the input $X \in \mathbb{R}^{n\times K}$, let $B \in \mathbb{R}^{n\times F}$ be the bias matrix, and let us denote by $\widehat{\mathbf{W}} \in \mathbb{R}^{nK\times F\times n}$ the tensor (generalization of the weight matrix $\widehat{W}$). Then, the generalized formulation of the GI layers is the function $\mathcal{L}^{GI} : \mathbb{R}^{n\times K} \to \mathbb{R}^{n\times F}$ such that

$$\mathcal{L}^{GI}(X) = \boldsymbol{f}\left(\widehat{\mathbf{W}}^\top \cdot \widehat{\boldsymbol{x}} + B\right). \tag{3}$$

Similarly to convolutional layers, we define also a *pooling* operation for the GI layers, to gather the information of the F filters of the output matrix through arbitrary "reducing operations" (e.g., the mean, the max, the sum). Moreover, since the regression targets could be associated only to a subset of the graph node, we add a *masking* option for the GI layers. For DFNs the mask is applied only to the output layer.

We observe that the L^{GI} general formulation belongs to the framework of Message Passing NN (MPNN) [13], although the different formulations of MPNN discussed in [13] do not provide a tensorization similar to equation (3). To the best of the authors' knowledge, other formulations of spatial graph neural networks did not propose either a tensor formulation like equation (3) (cfr. [11]). Other similarities to GINNs can be observed in the works [14, 15], which exploits the adjacency matrix to characterize the information flow. However, in [14] the NN is composed of many interconnected simple NNs according to the adjacency matrix, and in [15] the same NN architecture is used following the principle of the physics informed NNs (see [16, 17]).

3 Graph Informed Neural Networks for Discrete Fracture Networks

We consider the flux regression problem over a DFN. We name the test case DFN158, which consists of a fixed geometry with $n = 158$ fractures, immersed in a cubic domain $\mathbb{D}$ with a 1000 meters long edge. Fractures have been randomly generated using geological distributions of the geometrical features [18, 19]. The flow problem for DFN158 is defined by fixed boundary Dirichlet conditions

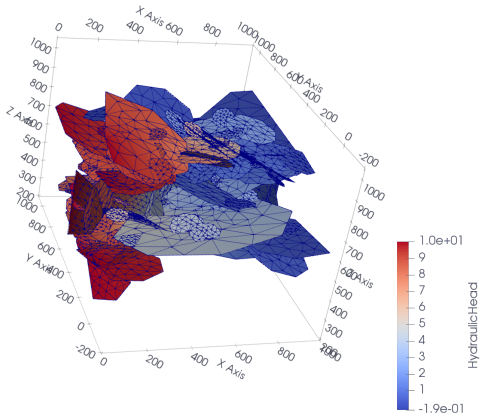

Figure 2: 3D view of the DFN158 used in the experiments

Table 1: Monte Carlo estimates of mean and standard deviation of total outflow fluxes. In brackets the relative error w.r.t. $\mathrm{MC}_{\mathcal{P}}$. The flux unit measure is $\mathrm{mm}^2\mathrm{s}^{-1}$.

	$\mathrm{MC}_{\mathcal{P}}$	$\mathrm{MC}_{\mathrm{GI}}$	$\mathrm{MC}_{\mathcal{P}\vert_\vartheta}$
mean	141.54	141.07 (0.3%)	143.93 (1.7%)
std	27.84	27.31 (1.9%)	30.10 (8.1%)

on the fracture edges of the DFN intersecting the leftmost and rightmost faces of the domain (w.r.t. x axis, see Figure 2). Dirichlet conditions impose a fixed pressure difference of 10 meters between the same two faces of $\mathbb{D}$, characterizing the flux directionality; therefore, the resulting $m = 7$ outflow fractures are fixed independently of the fracture transmissivities. The fracture transmissivities κ_i for each fracture $\mathcal{F}_i$ are modelled as random variables with log-normal distribution [20, 21] : $\log_{10} \kappa_i \sim \mathcal{N}(-5, 1/3)$.

We use GINN models to approximate by regression the m fluxes exiting from DFN158, for any vector of fracture transmissivities $\boldsymbol{\kappa} = [\kappa_1, \dots, \kappa_n]^\top$. We sample randomly 3000 transmissivities $\boldsymbol{\kappa}$ and we compute the corresponding fluxes $\boldsymbol{\varphi}$ through DFN simulations, obtaining a test set of $\mathcal{P}$ of 3000 pairs. (see appendix A for details about the dataset) $(\boldsymbol{\kappa}, \boldsymbol{\varphi}) \in \mathbb{R}^n \times \mathbb{R}^m$. Equivalently, we build a dataset $\mathcal{D}$ of $\vartheta = 500$ pairs $(\boldsymbol{\kappa}, \boldsymbol{\varphi})$ to train the GINNs; in particular, $\mathcal{D}$ is randomly split into a training set (80%) and a validation set for early stopping (20%). We conduct a grid search study of 360 GINN models over $\mathcal{D}$ and we select the best GINN model according to the best total outflux relative error over $\mathcal{P}$ (see appendix A for details about the grid search).

In Table 1 and Figure 3 we report the results of a numerical experiment to show the GINN potentials for the outflow moment estimates of DFN158 in a UQ scenario, varying the transmissivities $\boldsymbol{\kappa}$. We establish as "best estimate" and ground truth the MC estimates obtained by the 3000 fluxes of the test set $\mathcal{P}$, denoted by $\mathrm{MC}_{\mathcal{P}}$. Then, we compare $\mathrm{MC}_{\mathcal{P}}$ with: (i) the MC estimates given by a subset of $\vartheta = 500$ fluxes sampled from $\mathcal{P}$, denoted by $\mathrm{MC}_{\mathcal{P}\vert_\vartheta}$; (ii) the MC estimates given by the GINN predictions over the 3000 transmissivities in $\mathcal{P}$, denoted by $\mathrm{MC}_{\mathrm{GI}}$. The target of the experiment is to investigate if the GINN trained on ϑ samples returns estimates closer to $\mathrm{MC}_{\mathcal{P}}$ than the estimates $\mathrm{MC}_{\mathcal{P}\vert_\vartheta}$. Looking at Table 1 and Figure 3, we clearly see that estimates $\mathrm{MC}_{\mathrm{GI}}$ are closer to the estimates $\mathrm{MC}_{\mathcal{P}}$; on the contrary, estimates $\mathrm{MC}_{\mathcal{P}\vert_\vartheta}$ are farther from $\mathrm{MC}_{\mathcal{P}}$. Therefore, we can state that GINN model trained on $\theta = 500$ samples allows to improve the MC outflow moment estimates obtained directly with the same amount of DFN flow simulations. We can also analyse the results by the point of view of the computational cost. We recall that a GINN can make thousands of flux predictions in the order of the seconds, while DFN simulations take hours for hundreds of simulations, so that the computational cost of each MC estimate can be roughly evaluated by the number of DFN simulations it requires. The MC estimate obtained by the GINN predictions costs just θ simulations. By contrast, the MC estimate $\mathrm{MC}_{\mathcal{P}}$ requires hours of computation for the additional 2500 DFN simulations. Hence, at the cost of $\vartheta = 500$ DFN simulations, the GINN-based method returns an MC estimate very close to $\mathrm{MC}_{\mathcal{P}}$ and in practice it allows to save 2500 simulations.

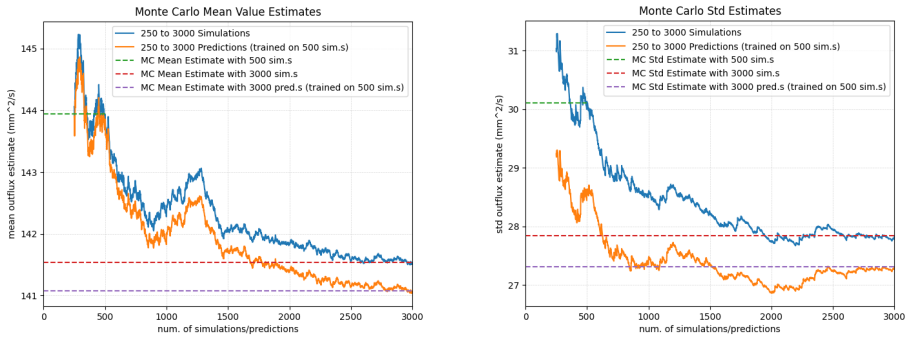

Figure 3: Monte Carlo estimates of the DFN158 total outflux (right: mean, left: std). Outflux estimates by DFN simulations (blue) or GINN prediction (orange) trained on $\vartheta = 500$ simulations.

4 Summary and Overview

We proposed a new spatial graph NN architecture for flux prediction in DFNs to exploit the graph structure of the DFN models. The new NN architecture is characterized by Graph Informed layers; these layers are defined with respect to the adjacency matrix of a given graph and re-define the convolution operation for graph-structured data in a new tensor formulation that allows multivariate input features ($K \geq 1$) and multivariate output features ($F \geq 1$). We tested the applicability of GINNs to a test case DFN, comparing the MC estimates obtained through the best-GINN predictions with a ground truth and with the MC estimates obtained with a number of simulations equal to the training set size. The experiment results showed that the GINN allowed to obtain a more accurate estimate of the moments, with a significant saving in computational cost. Moreover, we observe that in our experiments GINNs performances usually improve if depth increases (see A). This characteristic is extremely interesting because previous works suggest that going deeper in a graph neural network is not usually beneficial; indeed, the role of depth in graph NNs is still an open question [11]. Due to space limitation, our exposition is limited to a single test case and further experiments could strengthen the impact of our work.

In conclusion, we believe that the proposed GINN architecture could open new possibilities for all the physical problems that can be faced as regression and function approximation over a graph-structured domain, especially if it involves multivariate inputs and/or outputs.

Acknowledgments and Disclosure of Funding

Research performed in the framework of the Italian MIUR Award "Dipartimento di Eccellenza 2018-2022" granted to the Department of Mathematical Sciences, Politecnico di Torino, CUP: E11G18000350001. The research leading to these results has also been partially funded by INdAM-GNCS and by the SmartData@PoliTO center for Big Data and Machine Learning technologies. A. Mastropietro gratefully acknowledges the support from Addfor Industriale. S. Pieraccini also acknowledges support from Italian MIUR PRIN project 201752HKH8_003. F. Vaccarino acknowledges partial support from Intesa Sanpaolo Innovation Center. The funder had no role in study design, data collection, and analysis, decision to publish, or preparation of the manuscript.

References

[1] P. Adler, *Fractures and Fracture Networks*. Kluwer Academic, Dordrecht, 1999.

[2] G. Cammarata, C. Fidelibus, M. Cravero, and G. Barla, "The hydro-mechanically coupled response of rock fractures," *Rock Mechanics and Rock Engineering*, vol. 40, no. 1, pp. 41–61, 2007.

[3] C. Fidelibus, G. Cammarata, and M. Cravero, *Hydraulic characterization of fractured rocks. In: Abbie M, Bedford JS (eds) Rock mechanics: new research.* Nova Science Publishers Inc., New York, 2009.

[4] S. Berrone, A. Borio, and S. Scialò, "A posteriori error estimate for a PDE-constrained optimization formulation for the flow in DFNs," *SIAM J. Numer. Anal.*, vol. 54, no. 1, pp. 242–261, 2016.

[5] S. Berrone, S. Pieraccini, and S. Scialò, "On simulations of discrete fracture network flows with an optimization-based extended finite element method," *SIAM J. Sci. Comput.*, vol. 35, no. 2, pp. A908–A935, 2013. [Online]. Available: http://dx.doi.org/10.1137/120882883

[6] ——, "An optimization approach for large scale simulations of discrete fracture network flows," *J. Comput. Phys.*, vol. 256, pp. 838–853, 2014.

[7] S. Srinivasan, S. Karra, J. Hyman, H. Viswanathan, and G. Srinivasan, "Model reduction for fractured porous media: a machine learning approach for identifying main flow pathways," *Computational Geosciences*, Mar 2019. [Online]. Available: https://doi.org/10.1007/s10596-019-9811-7

[8] R. K. Tripathy and I. Bilionis, "Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification," *Journal of Computational Physics*, vol. 375, pp. 565 – 588, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999118305655

[9] S. Berrone, F. Della Santa, S. Pieraccini, and F. Vaccarino, "Machine learning for flux regression in discrete fracture networks," *GEM - International Journal on Geomathematics*, vol. 12, no. 1, p. 9, April 2021. [Online]. Available: https://doi.org/10.1007/s13137-021-00176-0

[10] S. Berrone and F. Della Santa, "Performance analysis of multi-task deep learning models for flux regression in discrete fracture networks," *Geosciences*, vol. 11, no. 3, p. 131, mar 2021. [Online]. Available: https://doi.org/10.3390%2Fgeosciences11030131

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2020.

[12] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1263–1272. [Online]. Available: https://proceedings.mlr.press/v70/gilmer17a.html

[14] B. Donon, B. Donnot, I. Guyon, and A. Marot, "Graph Neural Solver for Power Systems," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2019-July, no. July, pp. 1–8, 2019.

[15] B. Donon, R. Clément, B. Donnot, A. Marot, I. Guyon, and M. Schoenauer, "Neural networks for power flow: Graph neural solver," *Electric Power Systems Research*, vol. 189, no. October 2019, p. 106547, 2020. [Online]. Available: https://doi.org/10.1016/j.epsr.2020.106547

[16] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125 – 141, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999117309014

[17] M. Raissi, P. Perdikaris, and E. Karniadakis, George, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686 – 707, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999118307125

[18] Svensk Kärnbränslehantering AB, "Data report for the safety assessment, SR-site," SKB, Stockholm, Sweden, Tech. Rep. TR-10-52, 2010.

[19] J. D. Hyman, G. Aldrich, H. Viswanathan, N. Makedonska, and S. Karra, "Fracture size and transmissivity correlations: Implications for transport simulations in sparse three-dimensional discrete fracture networks following a truncated power law distribution of fracture size," *Water Resources Research*, 2016.

[20] X. Sanchez-Vila, A. Guadagnini, and J. Carrera, "Representative hydraulic conductivities in saturated grqundwater flow," *Reviews of Geophysics*, vol. 44, no. 3, pp. 1–46, 2006.

[21] J. D. Hyman, A. Hagberg, D. Osthus, S. Srinivasan, H. Viswanathan, and G. Srinivasan, "Identifying Backbones in Three-Dimensional Discrete Fracture Networks: A Bipartite Graph-Based Approach," *Multiscale Modeling & Simulation*, vol. 16, no. 4, pp. 1948–1968, 2018.

[22] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

Checklist

1. For all authors...
   1.1. Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   1.2. Did you describe the limitations of your work? [Yes] See section 4. Due to space limitation, we exposed only the application of the model GINN on a single test case.
   1.3. Did you discuss any potential negative societal impacts of your work? [No] We think this application has only positive impact.
   1.4. Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   2.1. Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results.
   2.2. Did you include complete proofs of all theoretical results? [N/A] No theoretical results.
3. If you ran experiments...
   3.1. Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In the paper we completely described the details about the dataset and the GINN model. In case of acceptance, we plan to provide an URL to the dataset in the camera-ready paper.
   3.2. Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In the appendix A we included all the details about the grid search study for the hyperparameters selection and the best GINN model found.
   3.3. Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The available computational power did not allow to do such an intensive experimental study on the GINN training.
   3.4. Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   4.1. If your work uses existing assets, did you cite the creators? [Yes] See Appendix A.
   4.2. Did you mention the license of the assets? [Yes] See Appendix A.
   4.3. Did you include any new assets either in the supplemental material or as a URL? [No] We do not provide the code of our models because we plan to submit the complete work as a journal paper.
   4.4. Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We generated the dataset.
   4.5. Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   5.1. Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] No crowdsourcing.
   5.2. Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   5.3. Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

We give more details about the dataset. The dataset is generated assuming a standard deviation of the sampling of the transmissivity parameter equal to 0.33. The complete dataset can be downloaded from the URL https://smartdata.polito.it/discrete-fracture-network-flow-simulations/. We use a subsampling of 3000 thousands of DFN outflow simulation to build the test set $\mathcal{P}$ and a second separate subsampling of other $\theta = 500$ samples to build the training dataset for the GINN models.

We give more details about the grid search conducted over the GINN hyperparameters for the DFN158 flux regression problem exposed in Section 3. The GINN model is characterized by the following architecture (see Figure 1): one input layer of shape n, followed by $H \in \mathbb{N}$ consecutive GI Layers, each one characterized by $F \in \mathbb{N}$ filters and activation function f; the last GI (output) layer is characterized by a *max-pooling* operation and a *mask* with respect to the m indices of the outflux fractures. All the GI Layers are designed to receive $K = F$ input features per node; the only exception is the first one where $K = 1$ (the input transmissivities). We varied the number of hidden layers $H \in \{3, 5, 7\}$, the output features of each hidden layer $F \in \{3, 5, 10, 20, 40\}$, the activation function $f \in \{$relu,elu, softplus,swish$\}$ and the mini-batch size $\beta \in \{10, 50, 100\}$, with an early stopping criterion over the validation loss, with patience of 150 epochs. The optimizer is Adam with starting learning rate 0.001. The best architecture resulted with $H = 5$, $F = 40$, $f =$ elu, $\beta = 10$. We observed a robustness of the result wrt the activation function and mini-batch, while the GINN architecture seemed to be more sensitive to F and H. The GINN implementation make use of TensorFlow 2.3.1 [22] with Apache 2.0. licence. The training of a GINN model lasts 10 minutes on average and it is influenced by the early stopping intervention. The grid search has been conducted on a GPU Nvidia 1080 8GB on a workstation of 4 Core, 8 Threads, 32 GB RAM. All the simulations have been computed on a workstation with two AMD Opteron Processors, Interlagos type, 12 cores, Ram 32 GB; the computational time for each simulation is in the order of tens of seconds.