Genetic Neural Networks for Coarse Scale Corrections of Advection-Diffusion FEM Simulations

Fleur Bouwman and Ing. Freek Klabbers

***Abstract*—Integrating Machine Learning with FEM simulation has the possibility to address FEM's high computation time. This paper explores the one-dimensional advection-diffusion problem and aims to introduce machine learning to it. The advection-diffusion problem is discretised using the Galerkin method, and the goal is to learn the stabilisation parameter.**

This is achieved using Genetic Neural Networks, as no target or gradient for backpropagation is available, only the error between the coarse and fine-scale FEM simulations. After an initial attempt posed problems, 4 techniques have been tried to increase performance. In the end, the best performing version had a relative error of around 1, thus requiring further work before its implemented. It is recommended to either make significant changes to the current approach or try something else entirely.

I. Introduction

Finite Element Method (FEM) is used in many engineering applications such as structural integrity, heat, and fluid flow applications [1]. In fluid mechanics, FEM models can become computationally expensive very quickly depending on the mesh size used [2]. Non-linear or dynamic problems like turbulence [3] or non-linear material behaviour [4] require a fine mesh to be accurate because of the varying scales. This is sometimes also called Variational Multiscale Method [5], [6].

With the rise of Machine Learning, there have been attempts to integrate these new techniques in the field of fluid mechanics [7]–[9]. Recently, [10]–[12] have shown promising results by running coarse simulations and correcting them using machine learning techniques. These all share the same approach of using supervised machine learning to predict the error, and then adding it to the coarse simulation. More promising, [13] shows a method of using domain knowledge in this exact advection-diffusion problem. They however used a numerical solver to determine a good solution, and then applied supervised machine learning. This is an interesting approach, but unlikely to scale well for more complex problems.

An unexplored approach would be to use Genetic Neural Networks (GNN) to minimise the error using a correction function. These networks can explore valid correction functions and evaluate their performance on a fitness function, in this case, the error between a fine and coarse grid simulation.

Machine Learning for Multi-Physics Modeling and design: Report submitted June, 2025; Supervised by Stein Stoter. Fleur Bouwman (2204452) and Freek Klabbers (2042274) are with the Department of Electrical Engineering, Eindhoven University of Technology. Stein Stoter is with the Power & Flow Group, Eindhoven University of Technology.

The goal of this report is to explore this technique on a simplified problem to test its validity. The 1D use case is an advection-diffusion heat-flow problem. If this shows promise, this could be expanded to more difficult problems. The goal for this report is:

Applying and validating the use of Genetic Neural Networks for coarse grid FEM simulation correction problems.

First off, the theoretical background will be discussed in more detail, including the governing equations and Genetic Algorithms. Then, the methods used in experimentation will be named. The results will be shown and discussed. The report will close with the conclusion and recommendation.

II. Theoretical Background

A. Governing Equations

In this paper, an advection-diffusion problem is considered which is governed by [13], [14]:

$$\begin{cases} a(x)\cdot\nabla u - \nabla\cdot(\nu(x)\nabla u) = f(x) & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{1}$$

In Equation 1, u is the unknown scalar function, and the output of the FEM simulation. $a(x)$ determines the strength of the advection, while $\nabla\cdot(\nu(x)\nabla u)$ is the diffusion. Note that the gradient is used here to keep the formula general, but since the simulation is in 1D, this simplifies to the derivative. $f(x)$ is the forcing term, which can be thought of as a heat source. All of this works on a domain, defined by Ω which goes from $[0,1]$. On the edges of the domain $\partial\Omega$, $u = 0$, which is the Dirichlet boundary condition.

To be able to use this with FEM, it is discretised using Galerkin's method. The discretisation is performed for two solution scales, the coarse scale with size $N = 5$ and the fine scale with size $N = 10^4$. **a** is discretised to a size a $3\cdot N$. This discrete version is called ah. However, it is known that while using Galerkin method, the discrete solution can become unstable in cases where advection dominates [15]. This instability is caused by spurious oscillations which negatively affect the accuracy. To quantify the dominating factor on a domain, the local Peclet number (Pe) is often used, which is defined as follows:

$$\text{Pe} = \frac{|a|h}{2\nu} = \frac{|a|}{2\nu N} \tag{2}$$

In Equation 2, h is the element size, which with a domain of width 1 simplifies to $\frac{1}{N}$. To stabilise the discrete solution, a few methods can be used [15]. The one observed in this paper is with the use of a stabilisation parameter τ . The exact way τ influences FEM simulation is well documented [6], [13], but out of scope for this research. What should be known about τ is that in general, there does not exist a single optimal value for tau. Notably, this is not the case for the 1D problem with constant values. In this case τ can be described by:

$$\tau = \frac{h}{2|a|} \cdot \xi(\text{Pe}) = \frac{1}{2|a|N} \cdot \xi(\text{Pe}) \tag{3}$$

$$\xi(x) = coth(x - \frac{1}{x}) \tag{4}$$

In Equation 3 and Equation 4, ξ is called the "upwind" function, which describes the influence of advection of neighbouring regions.

The goal will be now to stabilise the discrete solution using τ. The previous background knowledge may prove useful in this. In many approximations of τ, Peclet is used. So from the background it can be expected that Pe and N are powerful features for a possible machine learning model.

B. Genetic Algorithms

The term genetic algorithm has been described in multiple ways over the last decades. For a description of the term genetic algorithms, most recent papers refer to an article by John Holland [16]. Here, genetic algorithms are briefly described as: a search and optimization technique based on natural selection and genetics. Therefore, each step in the training process is called a generation, and the process will loop over different generations. The total process can be described with multiple steps in and around this loop, starting with an initial population.

The initial population in this research is a set of Neural Networks (NN) that will calculate Tau. Each of these NN is assigned with random weights. With the initial population created, the loop of the algorithm can start. In this loop, the first step is evaluating the fitness of the function. Because the results of the fitness of each NN will determine the outcome of the next step in the loop: selection. During selection, the algorithm will select the "parents", which is the term for the NN that are selected. Here, the probability of selecting an NN is higher if the NN has a higher fitness during the previous step. This can be compared with the natural selection in nature.

After selection, the parents are paired, and together each pair will produce one or more "children". These children inherit parts of the properties of each parent. In this case, the NNs that are parents will combine their weights into a new NN, a child. This step in the loop is called crossover/recombination. However, if only combining the properties of parents can make a new NN, the change of finding the best solution is smaller, or it can take longer. This is because this way the algorithm can easily get stuck in a local minimum. Like in nature, small mutations are needed to get some diversity and make it possible to evolve. Therefore, small changes are applied to the properties with a certain probability. This mutation step is the last step in the loop. Cause, after this mutation, there is a new generation from which the fitness can be determined.

In the process several hyperparameters can influence the results drastically. For example, the stop criteria can affect the performance of a trained model. Research shows that choosing a reliable stop criterion is not trivial, since genetic algorithms do not naturally converge in a specific way [17]. In the base, the algorithm can explore the search space indefinitely. Adaptive stop criteria, criteria based on the performance or behaviour of the algorithm are often more suitable. For instance, monitor if the fitness improvement or the changes made over the last n generations fall below a certain threshold, and stop the algorithm if this happens. This can avoid unnecessary iterations while the process is not stopped too early.

Secondly, another important aspect in choosing hyperparameters is choosing the right balance between the crossover and mutation parameters. The difficulties around this balance problem is described in the following paper [17]. If the mutation rate is relatively high, the algorithm will explore a lot within the parameter space. However, this has as result that it will learn less from its parents, and therefore will probably take more time to find the perfect solution. In contrast, if the mutation rate is very low, the chance of getting stuck in a local minimum will increase. This is because the algorithm will explore less of the parameter space.

In this research, genetic algorithms are used with different settings. More about the exact application of this technique can be found in section III.

III. Methods

A. Neural Network design

The genetic algorithm will optimise a neural network, the design of which is also a design choice. The architecture chosen in this project has an input of 18, a hidden layer of size 8 and a output layer of size 15. The input for the network consists of ν, N, Pe and fifteen discrete values of a. The output is the τ array that corrects the FEM simulation. Since τ consists of discrete values, this is a regression task.

This network was chosen for balancing network expressive powers and trainable weights. This network already contains 2565 trainable weights, which is a lot to explore. Since our network seemed to be struggling to learn with large networks, a small one has been chosen.

B. Preliminary Research

Before the different methods can be described, the definition of a data point needs to be clear. In this research, a data point represents a specific combination of values for fields a and f, which correspond to the advection and force, respectively. The fine mesh is used to determine the fine-scale value for u. Based on this value, the optimal course solution can be determined. The optimal course solution is used to minimise the error and is incorporated into the fitness function.

As a preliminary investigation for this project, Genetic Neural Networks (GNN) were used as an initial training run. This genetic algorithm used a single data point to optimize the fitness. When the fitness reaches a certain level, it would generate a new data point and optimize again. Repeating this process was supposed to improve the NN over time. However, after processing many data points, the algorithm appeared to lose the information or patterns it had learned from the first ones. This was also confirmed when evaluating the best NN after training with this method. The algorithm was evaluated by calculating the RMSE for fifty random new data points. The evaluation showed that the mean error was low, but there were some big outliers, see Fig. 1.

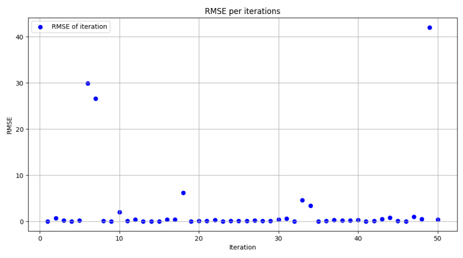

Fig. 1: RMSE plot showing general accuracy and large error outliers.

C. Alternate Training methods

Accordingly, the method aims to investigate alternative training strategies for genetic algorithms to enhance the model's generalization ability. While the preliminary research result is not used directly, it serves as a warm start for the training process in all new methods. The new methods that will be compared with the preliminary research are:

- Use batch to determine the fitness
- Increase the genetic pressure
- Change the fitness function
- Simplify ah

In the results section, the findings from the preliminary investigation are presented first, followed by a comparison with the outcomes of the different strategies. The same evaluation metrics are used for all trained models to ensure consistency in comparison. The following metrics are evaluated and compared:

- Average RMSE;
- Highest RMSE;
- Number of outliers.

These values are calculated by first loading the best-trained network from the training process. Next, the performance of the models is evaluated based on the selected metrics mentioned above. This evaluation is conducted using fifty new data points. Outliers in the RMSE values are identified using the interquartile range (IQR) method, where any value falling 1.5 times below the first quartile or above the third quartile is considered an outlier. By choosing a seed value beyond the range used during training, overlap between the training and test datasets is avoided, ensuring a proper separation.

For the first strategy, a batch was introduced to the code. In this project, the batch is a set of 100 data points generated to use for evaluating each candidate solution in every generation of the genetic algorithm. It can be seen as a dataset on which the fitness of the NN models is assessed. Using a batch-based training approach within genetic algorithms has been shown to improve generalization [18]. In addition, genetic algorithms are known to easily overfit when limited samples are given [19]. Hence, using a batch can also help to prevent overfitting.

A second possible solution is to increase the genetic pressure. The mutation rate can be adjusted to control genetic pressure. Increasing the mutation rate boosts genetic diversity, allowing the algorithm to explore the solution space more broadly. This can reduce the risk of getting trapped in a local minimum, which often results in poor generalization and may explain the outliers observed in the preliminary investigation. However, this research also explores lower mutation rates, as excessive mutation can introduce too much randomness, disrupting convergence and preventing the algorithm from refining good solutions. Testing a range of mutation rates can therefore help improve the model's generalization.

As stated before, the third option explored in this project is to change the fitness function. During the preliminary investigation, the fitness function was based on the average RMSE. Another option is to determine the fitness based on the maximum RMSE. This can improve the amount of outliers because it will place a higher priority on improving the worst error in the dataset. While the average RMSE focuses on overall performance, it can mask poor predictions on more difficult or rare samples. Therefore, this approach will hopefully result in a more generalized model.

Finally, the GNN might still struggle to adequately explore the high-dimensional weight space. To help the model, it might be helpful to learn on a simplified problem in which the right weights are easier to learn. These weights can then be used as a starting place for the complex problem. For this, the discrete advection field ah is kept constant in first instance. As noted earlier, this case has a known solution as given by Equation 3. The complex solution will likely also be parametrised by the same values.

D. Technical Implementation

For the implementation, Python 3 has been used.

The FEM simulations were implemented using Dolphinx which is part of the FEniCS project [20]–[23]. The code for which was kindly provided to us by Dr. Ir. S. Stoter.

Within Python, the PyGAD [24] library has been used for most of the genetic algorithm implementations.

IV. Results & Discussion

To keep the evaluation process fair, all the models are evaluated with the same fifty data points. The results of the different models can be found in Table I, where the mutation rate is abbreviated to MR. Based on these results, several insights can be drawn regarding the performance of the different training strategies.

Starting with introducing a batch of 100 samples to determine the fitness. This adjustment had the most significant positive impact. The method achieved the lowest average RMSE, namely 1.76, and reduced the number of outliers compared to the preliminary research model. This suggests that evaluating fitness over a batch helps improve generalization and prevents overfitting to individual samples. This is in line with the expectation, and therefore was still the best starting position for varying the mutation rates and the fitness function.

On the other hand, very low mutation rates, like 1-3, resulted in substantially higher average and maximum RMSE values. These results suggest that insufficient genetic diversity may cause premature convergence, leading the model to possibly overfit or become stuck in local minima of the solution space. High mutation rates, such as 7, also performed worse than the baseline in terms of average error, although the number of outliers was slightly reduced.

While the average RMSE is a useful indicator of overall model accuracy, it may not fully reflect the reliability of a correction method for FEM simulations. In the context of advection-diffusion problems, particularly when using coarse grids, a few extreme prediction errors can significantly degrade the overall solution quality. This is especially important in engineering applications where localized inaccuracies may cause structural or thermal miscalculations. For this reason, the number of outliers becomes a critical metric. When comparing the mutation rates of 5 and 7, both combined with the 100-sample batch, the average RMSE is similar. However, the mutation rate of 7 results in fewer outliers. Although its average RMSE is slightly higher, this reduction in extreme errors suggests that a higher mutation rate could lead to more stable and trustworthy correction functions. Given that the ultimate goal is to correct coarse FEM results in a way that scales to more complex, high-stakes simulations, robustness to outliers may outweigh a marginal increase in average RMSE, making the mutation rate of 7 a potentially more suitable choice.

Besides, changing the fitness function to a function that focused on the maximum RMSE instead of the average offered mixed results. Here, the same discussion can be taken place. Although this strategy slightly improved the performance of the outliers compared to the baseline and the 100 samples models with a mutation rate of 5, it came with a higher average RMSE. This trade-off indicates that targeting outliers more aggressively may reduce extreme prediction errors but can also weaken overall model accuracy. However, if minimizing the number of outliers is prioritized over achieving the lowest average RMSE, the model using the max fitness function outperforms the others, yielding only six outliers.

The simplified function of ah was also evaluated to assess whether this method would improve the model's performance. However, the results indicate that this simplification is not beneficial to the model's accuracy or robustness. With an average RMSE of 4.74 and a maximum RMSE of 125.24, the model performs worse than most of the other variants, including the model from the preliminary study. Moreover, the number of outliers remains high at nine, matching the preliminary model.

TABLE I: Comparison of different models with RMSE and outlier statistics, with in green the best value for each evaluation metric

Model	Average RMSE	Highest RMSE	# Outliers
Base	2.84	62.24	9
100 samples	1.76	52.62	8
MR = 1	6.12	72.96	7
MR = 2	14.59	310.87	6
MR = 3	15.36	326.40	6
MR = 5	1.76	52.62	8
MR = 7	5.05	133.71	6
MR = 8	4.19	122.29	8
Max fitness	4.05	100.46	6
Simple ah	4.74	125.24	9

These values for RMSE don't mean anything on their own; they depend on the size of their original value. A relative RMSE has been investigated, which is just the RMSE divided by the average value of the solution u. As it turns out, this value often lies at 1, with some values far exceeding 1. If the mean error is roughly the size of the true solution, then the predictions are not at all useful. This means that the nice moderate RMSE values were just small solution fields to begin with. The outliers are almost always cases where the true field is just large.

V. Conclusion and Recommendations

The goal for this report was to assess whether GNNs can be used for coarse grid FEM simulation correction. To start off with a proof of concept, this report focused on the simple 1D advection-diffusion problem. To make the main problems of this application even clearer, a small preliminary investigation was done. This showed that there were big outliers compared to the average Root Mean Square Error (RMSE), and that the outliers were also quite high.

To improve the generalization, different strategies were applied. By incorporating batch-based fitness evaluation, generalization improved significantly, with a noticeable reduction in both average RMSE and the number of outliers. This confirms the hypothesis that using a representative subset of the data during training enhances the network's ability to generalize and prevents overfitting. Therefore, training with a batch size was further explored. However, different mutation rates, a simplified ah, and prioritizing maximum RMSE instead of average RMSE for the fitness function, resulted in higher values for the average RMSE during testing.

While some mutation rates and changing the fitness function raised the mean error, they succeeded in reducing the number of outliers. This trade-off is specifically important when using GNNs for FEM simulation corrections. Small errors in the first steps of the process can create large negative effect for the end simulation and therefore the reliability of the FEM simulation. The expectation is that this problem becomes even bigger when scaling the method to higher dimensions.

Overall, the relative size of the error is around 1. With these kinds of error sizes, it is not recommended that any of these models be used. Significant work would need to be put into choosing the right networks and training to improve the results from here. A path that may be explored is either graph neural networks or convolutional neural networks. Both of these neural networks can make use of the properties of the data, namely the adjacency of the ah. Convolutional networks are attractive due to that they require fewer weights to train, and graphs can be very powerful too, since they can take adjacent nodes into account better. Changing how the genetic algorithm is trained may also work, for example, going over each data point one-by-one, but only for 1 generation, might yield improvements but was explored during this project.

More research into how this approach could be generalised would also benefit its case. Two main problems are generalisation for each mesh size and generalisation for higher dimensions could both be explored. A few more simple solutions were skipped due to fear that they would not generalise, but challenging this assumption might allow for easier solutions.

References

[1] S. Direct. "Finite element method." (2021), URL: https://www.sciencedirect.com/topics/engineering/finite-element-method.

[2] T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza, "Model order reduction in fluid dynamics: Challenges and perspectives," in *Reduced Order Methods for Modeling and Computational Reduction*, ser. MS&A - Modeling, Simulation and Applications, Q. A and R. G, Eds., vol. 9, Cham, Germany: Springer, 2014, pp. 235–273, ISBN: 978-3-319-02089-1. DOI: 10.1007/978-3-319-02090-7_9.

[3] K. E. Jansen and J. Brown, "Chapter 5 - finite element methods for turbulence," in *Numerical Methods in Turbulence Simulation*, ser. Numerical Methods in Turbulence, R. D. Moser, Ed., Academic Press, 2023, pp. 189–234, ISBN: 978-0-323-91144-3. DOI: https://doi.org/10.1016/B978-0-32-391144-3.00011-5. URL: https://www.sciencedirect.com/science/article/pii/B9780323911443000115.

[4] H.-C. Huang, Z.-H. Li, and A. S. Usmani, *Finite Element Analysis of Non-Newtonian Flow, Theory and Software*, 1st ed. London: Springer London, 1999, 218 pp., Published as part of the Springer Book Archive, ISBN: 978-1-4471-1204-4. DOI: 10.1007/978-1-4471-0799-6.

[5] T. J. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy, "The variational multiscale method—a paradigm for computational mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 166, no. 1-2, pp. 3–24, 1998. DOI: 10.1016/S0045-7825(98)00079-1.

[6] K. F. S. Stoter, "The variational multiscale method for discontinuous galerkin type finite element formulations," Ph.D. Thesis, University of Minnesota, Dec. 2019.

[7] R. Vinuesa and S. L. Brunton, "Enhancing computational fluid dynamics with machine learning," *arXiv preprint arXiv:2110.02085*, 2022. arXiv: 2110.02085 [physics.flu-dyn]. URL: https://arxiv.org/abs/2110.02085.

[8] M. P. Brenner, J. D. Eldredge, and J. B. Freund, "Perspective on machine learning for advancing fluid mechanics," *Phys. Rev. Fluids*, vol. 4, p. 100 501, 10 Oct. 2019. DOI: 10.1103/PhysRevFluids.4.100501. URL: https://link.aps.org/doi/10.1103/PhysRevFluids.4.100501.

[9] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. Volume 52, 2020, pp. 477–508, 2020, ISSN: 1545-4479. DOI: https://doi.org/10.1146/annurev-fluid-010719-060214. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010719-060214.

[10] A. Kiener, S. Langer, and P. Bekemeyer, "Data-driven correction of coarse grid cfd simulations," *Computers Fluids*, vol. 264, p. 105 971, 2023, ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2023.105971. URL: https://www.sciencedirect.com/science/article/pii/S0045793023001962.

[11] J. Pathak, M. Mustafa, K. Kashinath, E. Motheau, T. Kurth, and M. Day, "Using machine learning to augment coarse-grid computational fluid dynamics simulations," *arXiv preprint arXiv:2010.00072*, 2020. arXiv: 2010.00072 [physics.comp-ph]. URL: https://arxiv.org/abs/2010.00072.

[12] F. J. Cantarero-Rivera, R. Yang, H. Li, H. Qi, and J. Chen, "An artificial neural network-based machine learning approach to correct coarse-mesh-induced error in computational fluid dynamics modeling of cell culture bioreactor," *Food and Bioproducts Processing*, vol. 143, pp. 128–142, 2024, ISSN: 0960-3085. DOI: https://doi.org/10.1016/j.fbp.2023.11.004. URL: https://www.sciencedirect.com/science/article/pii/S0960308523001402.

[13] T. Tassi, A. Zingaro, and L. Dede', "A machine learning approach to enhance the supg stabilization method for advection-dominated differential problems," *arXiv preprint arXiv:2111.00260v2*, Jul. 2022, Version 2, updated July 8, 2022. URL: https://arxiv.org/abs/2111.00260v2.

[14] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced Basis Methods for Partial Differential Equations: An Introduction* (Unitext). Springer International

Publishing, 2016, vol. 92, ISBN: 978-3-319-49315-2. DOI: 10.1007/978-3-319-49316-9. URL: https://link.springer.com/book/10.1007/978-3-319-49316-9.

[15] V. John and P. Knobloch, "On spurious oscillations at layers diminishing (sold) methods for convection–diffusion equations: Part i – a review," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 2197–2215, 2007, Received 31 October 2005; revised 13 November 2006; accepted 21 November 2006. DOI: 10.1016/j.cma.2006.11.006.

[16] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[17] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, "On stopping criteria for genetic algorithms," in *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA 2004)*, ser. Lecture Notes in Artificial Intelligence, vol. 3171, Berlin, Heidelberg: Springer-Verlag, 2004, pp. 405–413. DOI: 10.1007/978-3-540-28650-9_41.

[18] I. Gonçalves, L. M. Oliveira, B. Veloso, and S. Silva, "Batch training in genetic programming for symbolic regression," in *Proceedings of the 2016 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2016, pp. 4698–4705. DOI: 10.1109/CEC.2016.7744357.

[19] T. Žegklitz and P. Pošík, "A symbolic regression benchmark for genetic programming," in *Proceedings of the 18th European Conference on Genetic Programming (EuroGP)*, ser. Lecture Notes in Computer Science, vol. 9025, Springer, 2015, pp. 218–229. DOI: 10.1007/978-3-319-16501-1_18. URL: https://doi.org/10.1007/978-3-319-16501-1_18.

[20] I. A. Baratta, J. P. Dean, J. S. Dokken, *et al.*, *DOLFINx: The next generation FEniCS problem solving environment*, preprint, 2023. DOI: 10.5281/zenodo.10447666.

[21] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells, "Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes," *ACM Transactions on Mathematical Software*, vol. 48, no. 2, 18:1–18:23, 2022. DOI: 10.1145/3524456.

[22] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells, "Basix: A runtime finite element basis evaluation library," *Journal of Open Source Software*, vol. 7, no. 73, p. 3982, 2022. DOI: 10.21105/joss.03982.

[23] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, "Unified form language: A domain-specific language for weak formulations of partial differential equations," *ACM Transactions on Mathematical Software*, vol. 40, 2014. DOI: 10.1145/2566630.

[24] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," *Multimedia Tools and Applications*, pp. 1–14, 2023.

Declaration of AI Tools

AI tools have been utilized in various aspects of this project and the report's writing. This section aims to declare all AI tools used in both the report and other work done. The authors hereby declare that all AI output has been checked and take full responsibility for all content delivered, both in the report and in the code.

Writing tools used

- ChatGPT-4o and ChatGPT-4o mini have both been used to format certain Biblatex references. Its "Deep Research" feature has also been used.
- Grammarly has been used; this is a spelling and grammar check on texts. It checks written text in real time and rewrites small portions, but never more than a sentence.

Coding tools used

- GitHub Co-pilot is a coding AI that can be installed in Visual Studio Code which has access to the entire code base (if the user so chooses). It has been used to help debug code and rewrite sections to be more concise.