DEEP LEARNING OR INTERPOLATION FOR INVERSE MODELLING OF HEAT AND FLUID FLOW PROBLEMS ?

Rainald Löhner[1], Harbir Antil[2], Hamid R. Tamaddon-Jahromi[3], Neeraj Kavan Chakshu[3] and Perumal Nithiarasu[3]

[1] *Center for Computational Fluid Dynamics and*

[2] *Center for Mathematics and Artificial Intelligence (CMAI)*

College of Science, George Mason University 4400 University Drive, Fairfax, VA 22030-4444, USA

[3] *Zienkiewicz Centre for Computational Engineering*

Swansea University, Swansea SA1 8EN, United Kingdom

Abstract

A comparison of interpolation procedures (IP) and deep learning (DL) methods is presented for inverse heat transfer problems with linear and non-linear behaviour. The proposed procedures/methods are tested for linear and non-linear heat conduction, forced convection, and natural convection problems in which the boundary conditions are determined by providing four temperature measurements. The results indicate that IP outperforms DL methods in accuracy for linear heat conduction problems while DL method is better for non-linear heat conduction problems. For heat convection problems both methods offer similar levels of accuracy.

Keywords: Interpolation, Deep Learning, Deep Neural Networks, Linear heat conduction, Non-Linear heat conduction, Forced and natural convection

1. INTRODUCTION

In a recent paper, Tamaddon-Jahromi *et al* [1] proposed the use of deep learning (DL) via specific deep neural networks (DNNs) and to quickly solve inverse problems arising in the general field of heat transfer. The approach was tested on a 2-D unit square shown in Figure 1. The inverse problem was formulated as follows: given a constant (but possibly different) temperature on each of the 4 walls (i.e. a total of 4 design variables) and the temperature at a series of internal (measuring) points, how accurately can a DNN estimate the wall temperatures ? Tamaddon-Jahromi *et al.* [1] performed forward calculations, tabulated the temperatures at the wall (input of the forward problem) and the resulting temperatures at the measuring points (output of the forward problem). This database was then used to train several DNNs, which ranged from 2-4 hidden layers and 16-64 neurons per hidden layer. The input data for the DNNs was the output of the forward problem, and the output had

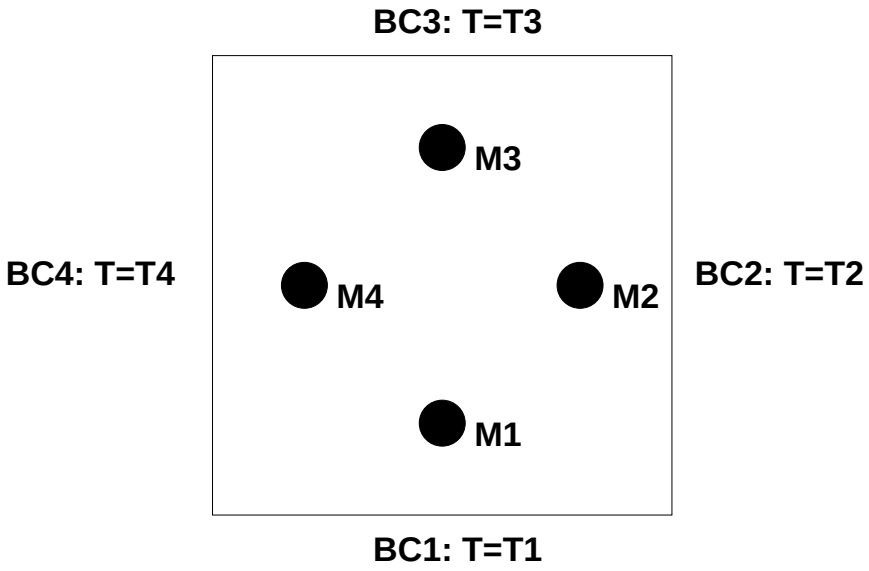


Figure 1: Problem Statement. Geometry, boundary conditions and internal measurement points.

to match as best possible the input of the forward problem. The results showed a very high predictive accuracy for the DNNs, often exceeding 95%, i.e. less than 5% error.

Intrigued by these results, as well as the high number of neurons needed to get these levels of accuracy, the question arose whether simple interpolation procedures could not yield a similar accuracy. After all: if the input-to-output map/function is ill-defined or multivalued (i.e. requiring a high-fidelity physics simulation), given an arbitrary input neither a DNN nor an interpolation algorithm would be able to accurately predict outputs.

Interpolation algorithms require fast search techniques. Assuming these are available, they offer a number of interesting possibilities for both IP and DL:

- One can 'scan' the learning data and see if there are gaps/ holes/ 'empty regions' that need further data;

- Given new entries into the learning set, one can judge whether this new data is useful or not (after all: if a point with similar data exists, why add it to the database ?);

- Given new entries into the learning set, one can judge the level of 'noise' or 'uncertainty' in the data (i.e. closeness in inputs but differences in output);

- Given new entries into the learning set, one can judge whether the input to output map is unique, i.e. whether for similar input data large differences in output data are possible (e.g. butterfly effects);

- New entries that are not duplicating already given information enhance the database and improve the accuracy of the interpolation; this implies that both IP and DL algorithms improve and 'learn' with more data;

Interpolation-based algorithms offer further advantages:

- There is no arbitrariness in interpolation algorithms; unlike DNNs, there is no need to test/evaluate the best combination of number of layers, number of neurons in each layer, activation function type, etc.

- One can compute local gradients to estimate the accuracy of interpolation algorithms;

- Interpolation should have at least comparable speeds to DNNs; if one can order the data properly, the number of operations required may even be lower than a complex DNN;

- Unlike DNNs that 'saturate' once their coefficients are fixed after a certain number of entries in the learning data set, interpolation keeps improving with additional data;

- Unlike DNNs, IPs do not need to be retrained when new data enters the learning data set.

The biggest drawback of interpolation techniques is the storage; storage grows linearly with a dataset, while accuracy only increases $^{1/d}$, where d is the dimensionality of the problem.

There are several interpretations of DL and there are various types of DNNs; for a broad view of the field, we refer to the seminal paper of LeCun, Bengio, Hinton [2]. In particular, we use the interpretation where in each layer (level) we first apply an affine transformation followed by an application of a nonlinear activation function. In the end, the DNN approximation is a composition of such nonlinear transformations in each layer. We further emphasize there are other more sophisticated DNNs such as Convolution Neural Network etc., but we do not explore these options further in this paper. We refer to [3] for mathematical approximation properties of DNNs used in our paper and its comparison with linear and nonlinear interpolation. See also [4] for the approximation properties of finite element functions by DNNs. For completeness, we also refer to [5] for Physics informed neural networks (PINNs), and also [6] for a DNN based approach to learn constitutive relations from observations.
The goal of the present paper is not to add to this growing list of excellent publications on the role of DNNs in Physics based modeling and simulations, but to provide IP as an alternative to DL. A number of multiple inverse problems, illustrate that IP outperforms DL methods in accuracy for linear heat conduction problems while DL is better for non-linear heat conduction problems. For heat convection problems both methods provide comparable levels of accuracy. Even though unrelated, but for completeness we also mention that comparison between various approaches is somewhat standard in the literature, see for instance [7] for a comparison between interpolation, statistical, and data-driven methods for missing values in datasets.

In order to test the effectiveness of IP and DL methods, the inverse heat and fluid flow problems considered by Tamaddon-Jahromi *et al* [1] have been revisited. In the following

section IP method used is explained. Section 3 provides a brief summary of the deep learning (DL) methods used. Section 4 presents a comprehensive set of results to compare IP and DL methods for the classes of problems considered here. Finally, some conclusions are drawn in Section 5.

2. INTERPOLATION

Given that the training set for the DNNs is usually very large, a possible alternative is to use this data directly and interpolate from it. Assume as given for each of the N data entries/ points/ cases of the database: $\mathbf{u} := u(i), i = 1, \ldots, n_i$ input values and $\mathbf{v} := v(j), j = 1, \ldots, n_o$ output values. For a new input $\mathbf{u}^*$: find the closest m entries in the database that minimize some distance norm to $\mathbf{u}^*$, e.g.

$$d = \left[\sum_{i=1,n_i} |u^*(i) - u(i)|^p \right]^{\frac{1}{p}} \quad . \tag{2.1}$$

Typical values used are $p = 1$ or $p = 2$. The final output value is then obtained by some weighted interpolation, e.g. an inverse distance weight:

$$\mathbf{v}^* = \frac{\sum_{k=1,m} \frac{1}{d_k^q} \mathbf{v}_k}{\sum_{k=1,m} \frac{1}{d_k^q}} \quad . \tag{2.2}$$

Typical values used are $q = 1$ or $q = 2$.

3. DEEP LEARNING

Deep learning (DL), a new Artificial Intelligence (AI) trend that uses multi-layer perception network [8], has received increasing attention from researchers and has been widely applied to numerous real-world applications and across many fields [2, 9, 10, 11, 12, 13, 14, 15]. Deep learning is able to effectively capture the non-linear and non-trivial user-item relationships, and it enables the codification of more complex abstractions as data representations in the higher layers. The general structure or configuration of the proposed neural network consists of L number of hidden layers along with one input and one output layer. Each hidden layer, input layer and output layer have K, N and J number of neurons, respectively. The equation below shows the general structure of the neural network:

$$\mathrm{BC}_j = \phi\left(\sum_{m=1}^{K^L} \theta_{jm}^L G_m^L \right), \qquad j = 1 - 4, \tag{3.1}$$

where G_k^1 is

$$G_k^1 = g\left(\sum_{n=1}^{N} \theta_{kn}^1 M_n + [bias]_k^1 \right), \qquad k = 1 - K^1 \tag{3.2}$$

and G_k^l is

$$G_k^l = g\Big(\sum_{m=1}^{K^{l-1}} \theta_{km}^l G_m^{l-1} + [bias]_k^l\Big), \qquad k = 1 - K^l, \quad l = 2 - L \tag{3.3}$$

where $\phi(x) = \max(0, x)$, θ are weights, and $g(x) = \begin{cases} x & \text{if } x > 0 \\ 0.3x & \text{if } x <= 0 \end{cases}$ are non-linear activation functions for the output layer and hidden layers respectively. The input layer is (M_n) has N number of input neurons. Here, the output layer produces values calculated for 4 boundary conditions, hence number of neurons in this layer, J, is four. Network weights and biases of Neural Networks (NNs) are tuned based on data using the adaptive moment estimation (Adam) algorithm [16]. The main part of the DNN methodology is the learning or training process in which the errors determined at the output layer are successively reduced by adjusting the weights and biases throughout the network. The DNN architectures employed in the present work are listed in Table 1.

Table 1: Architecture of Deep learning models

Inverse Problem	Deep Neural Network Architecture
Linear Conduction	4-64-32-16-4
Non-linear Conduction	4-64-32-16-4
Forced Convection	4-64-32-16-16-16-4
Natural Convection (10 x 10 grid)	4-64-32-Dropout(10%)-16-16-16-4
Natural Convection (20 x 20 grid)	4-64-32-Dropout(20%)-32-16-16-4

4. NUMERICAL EXAMPLES

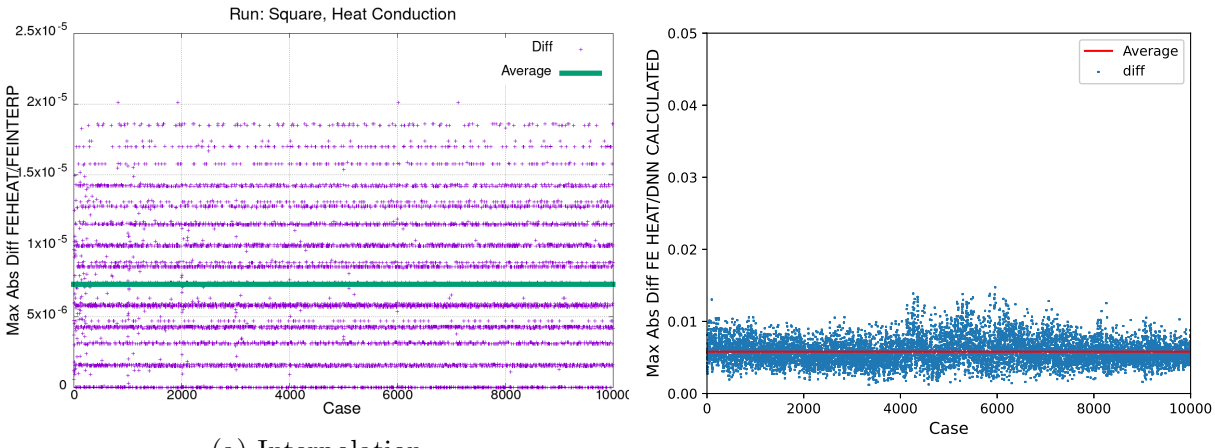
In order to test the ideas, the same test cases as in [1] are considered.

4.1. Linear Heat Conduction

The equation describing the temperature field is given by:

$$\nabla k \nabla T = 0 \quad , \quad k = 1 \quad . \tag{4.1}$$

The database was generated by an exhaustive combination of 11 possible temperatures on each wall ([0.00:1.00], with constant increments of 0.1), leading to a total of $11^4 = 14,641$ cases. FEHEAT [17] was used to solve the heat equation via finite elements. The mesh size was set to $h = 0.1$, similar to [1]. For each of these (forward) runs, the temperature at the 4 points marked M1-M4 was recorded. The data of these runs was stored in a table and used for interpolation. Thereafter, a second set of runs was performed by an exhaustive combination of 10 possible temperatures on each wall ([0.05:0.95], with constant increments of 0.1), leading to a total of $10^4 = 10,000$ cases. The logic for this choice was that these locations represent the 'furthest' possible distance from the given data, and should



(a) Interpolation

(b) Deep Learning

Figure 2: Linear heat conduction problem. Maximum error between computed and estimated boundary temperatures T1-T4; Interpolation (a) vs Deep Learning (b)

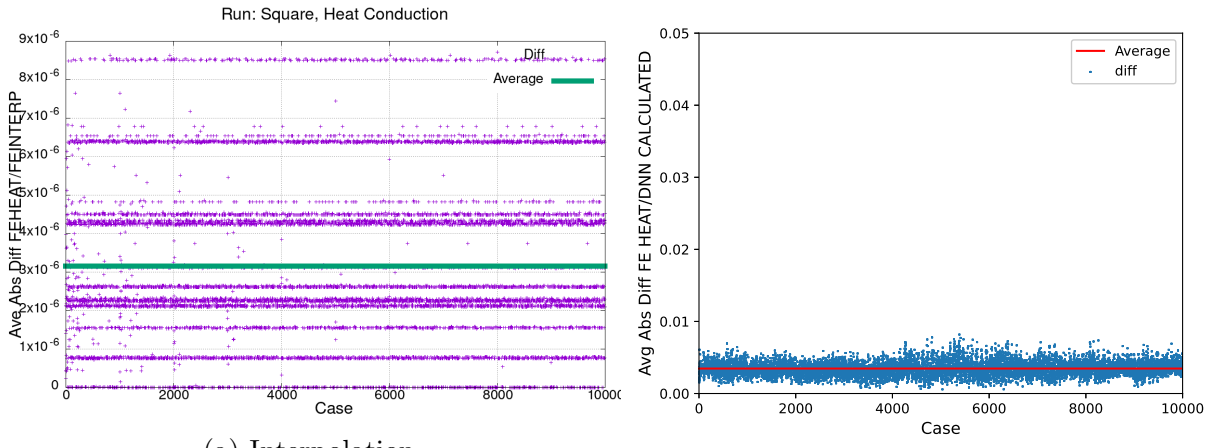
therefore be indicative of 'worst case scenarios' for both IP and DL techniques. As before, the temperature at the 4 points marked M1-M4 was recorded for each of these runs. The wall temperatures for this second set of runs was also estimated via interpolation using the original database [0.00,1.00,0.1], and the differences in estimated vs. real wall temperatures recorded. These differences are shown in Figures 2 and 3. The DNN model was trained on the first set of 14,641 cases and tested on the second set of 10,000 cases. The difference between DNN calculated vs. real wall temperatures have alse been recorded in Figures 2 and 3. Figures 2 and 3 respectively show the maximum and average errors. Note the vastly different scales in the graphs. Using interpolation results in errors that are 2-3 orders of magnitude smaller than DNNs.

4.2. Nonlinear Heat Conduction

The equation describing the temperature field is now given by:

$$\nabla k \nabla T = 0 \quad , \quad k = \max(0.01, T) \quad . \tag{4.2}$$

As before, the database was generated by an exhaustive combination of 11 possible temperatures on each wall [0.00:1.00:0.1], leading to a total of $11^4 = 14,641$ cases. FEHEAT was used to solve the heat equation. The same mesh size of $h = 0.1$ was used for this nonlinear case as well. For each of these runs, the temperature at the 4 points marked M1-M4 was recorded. The data of these runs was stored in a table and used for interpolation. Thereafter, a second set of runs was performed by an exhaustive combination of 10 possible temperatures on each wall [0.05:0.95:0.1], leading to a total of $10^4 = 10,000$ cases. The wall temperatures for this second set of runs were also estimated via interpolation using the original database [0.00,1.00,0.1], and the differences in estimated vs. real wall temperatures recorded. These differences are shown in Figures 4 (maximum) and 5 (average). Note that



(a) Interpolation

(b) Deep Learning

Figure 3: Linear heat conduction problem. Average error between computed and estimated boundary temperatures T1-T4; Interpolation (a) vs Deep Learning (b)

in this case the differences are higher, something that is to be expected for nonlinear cases. As with the linear case, the DNN model was trained on the first set of 14,641 cases and tested on the second set of 10,000 cases. The difference between the DNN calculated vs. real wall temperatures may be seen in Figures 4 and 5. It can be observed here that in this case, the DNN errors are lower, with averages hovering around 1% for IPs and 0.3% for DNNs.

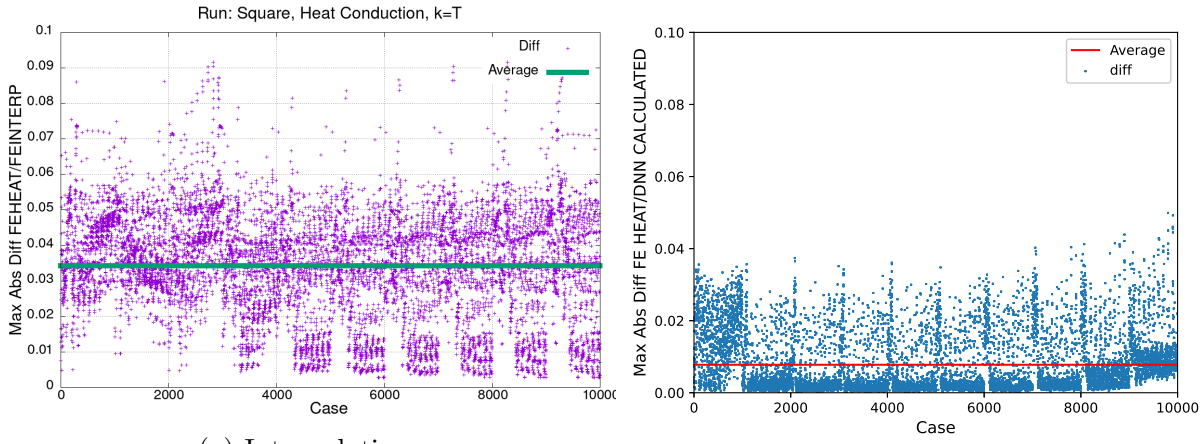
4.3. Forced convection heat transfer

The equation describing the temperature field is now given by:

$$\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0, \\ (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} &= -\nabla p + \frac{1}{Re}\nabla^2 \boldsymbol{v} \\ \boldsymbol{v} \cdot \nabla T &= \frac{1}{Pe}(\nabla^2 T) \end{aligned} \tag{4.3}$$

where Re is the Reynolds number defined as $Re = \frac{u_a L_{char}}{\nu}$ and $Pe = \frac{u_a L_{char}}{\alpha}$ is the Péclet number. u_a is a reference velocity, L_{char} is a characteristic dimension, ν is kinematic viscosity and α is thermal diffusivity of the fluid (see [18] for more details). In this study, the Reynolds number was selected to be $Re = 140$ and $Pe = 100$. FEFLO [19] was used to solve the Navier-Stokes and heat equations via finite elements.

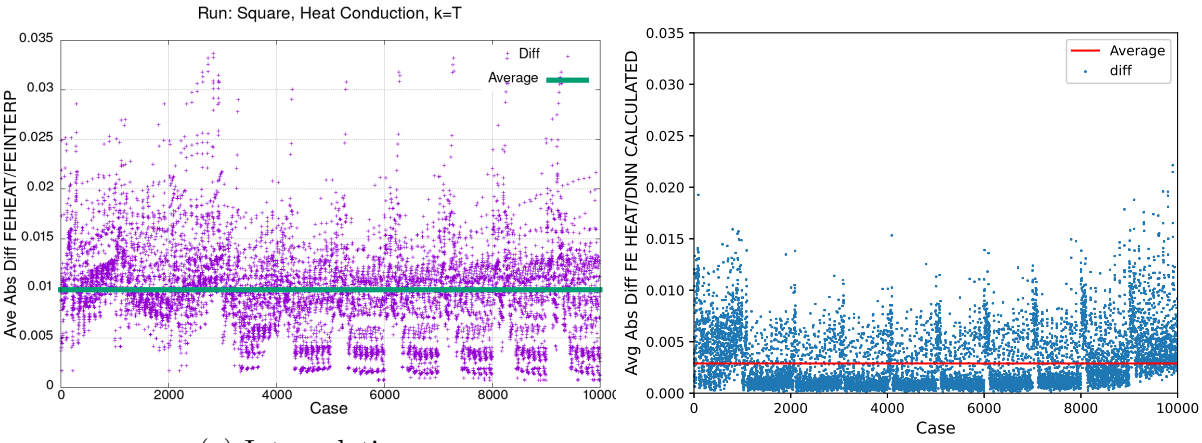
As before, the database for the IP and DNN consisted of the same 14,641 wall temperature cases, and both were tested on the second set of 10,000 cases. The results obtained have been summarized in Figures 6 and 7. It may be observed that, interestingly, IP shows constant errors at three levels of around 0.0, 0.02 and 0.03, even though the same error measures as for all other cases were employed. This can be traced to a sudden change in boundary



(a) Interpolation

(b) Deep Learning

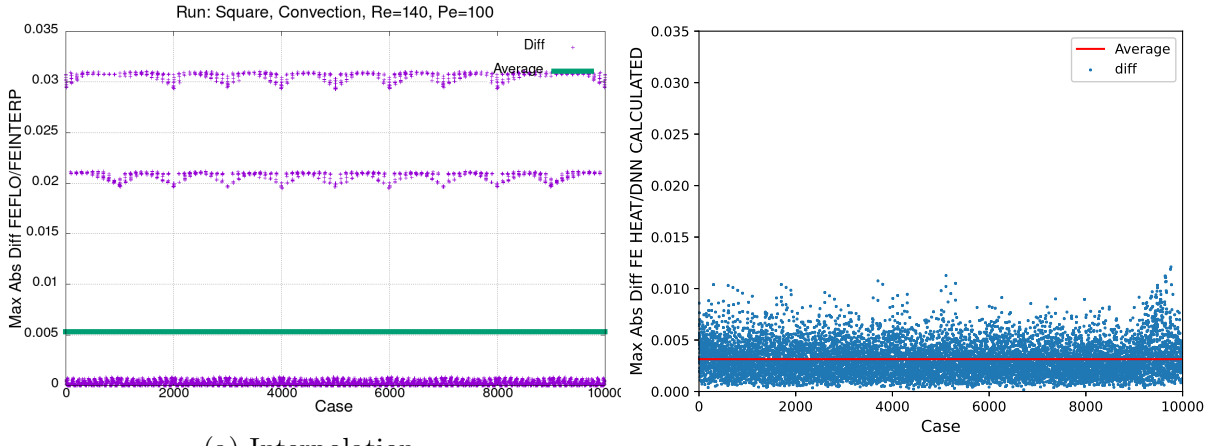
Figure 4: Nonlinear heat conduction problem. Maximum error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b)



(a) Interpolation

(b) Deep Learning

Figure 5: Nonlinear heat conduction problem. Average error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b)



(a) Interpolation

(b) Deep Learning

Figure 6: Forced convection problem. Maximum error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b)

conditions for the walls. The average errors for DNNs are around 0.008 and 0.005 (red lines in Figures 6 and 7) as compared to 0.005 and 0.002 (green lines in Figures 6 and 7) for IPs.

4.4. Natural convection heat transfer

The equation describing the temperature field is now given by:

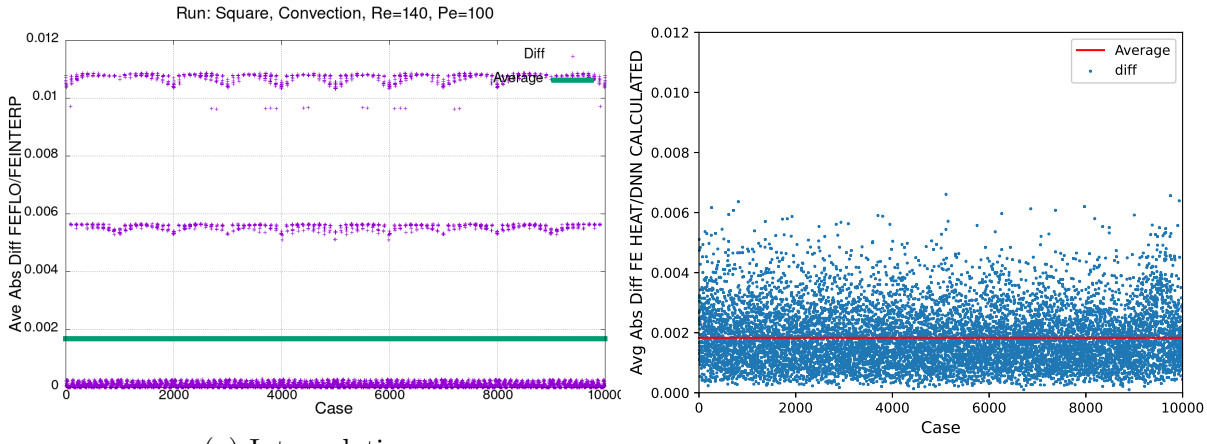
$$\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0, \\ (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} &= -\nabla p + Pr\nabla^2 \boldsymbol{v} + Gr\, Pr^2\, T\, \hat{\boldsymbol{y}} \\ \boldsymbol{v} \cdot \nabla T &= \nabla^2 T \end{aligned} \tag{4.4}$$

where p is the pressure, $\hat{\boldsymbol{y}}$ is the unit vector in the y-direction, and Pr and Gr are the Prandtl and Grashof numbers, respectively (see [18] for more details). Gr is taken here to be $Gr = 10^5$ and $Pr = 0.71$ which corresponds to air.

As before, FEFLO was used to generate these data sets, using unstructured grids of size comparable to 10x10 and 20x20 structured meshes. The first set of 14,641 data points were chosen as the database for interpolation/training set and the second set of 10,000 were selected as test candidates. The differences between estimates and actual values have been summarized in Figures 8, 9 for the 10x10 mesh and Figures 10 and 11 for the 20x20 mesh. One can see that for this case the error levels are almost the same: by some measures/cases IPs are better, by others DNNs are better.

5. Conclusions and Outlook

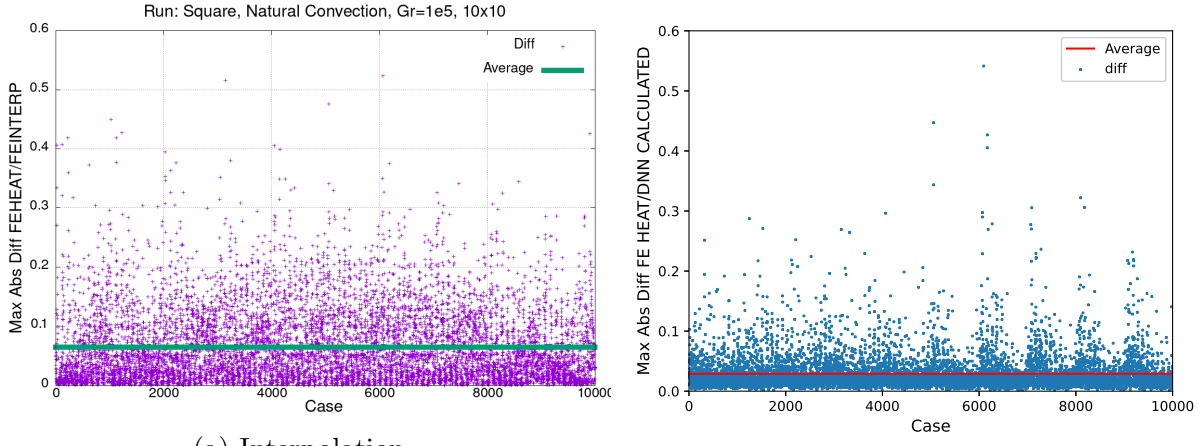
In summary, the comparison of this limited set of results between the two approaches provide us with the following observations:



(a) Interpolation

(b) Deep Learning

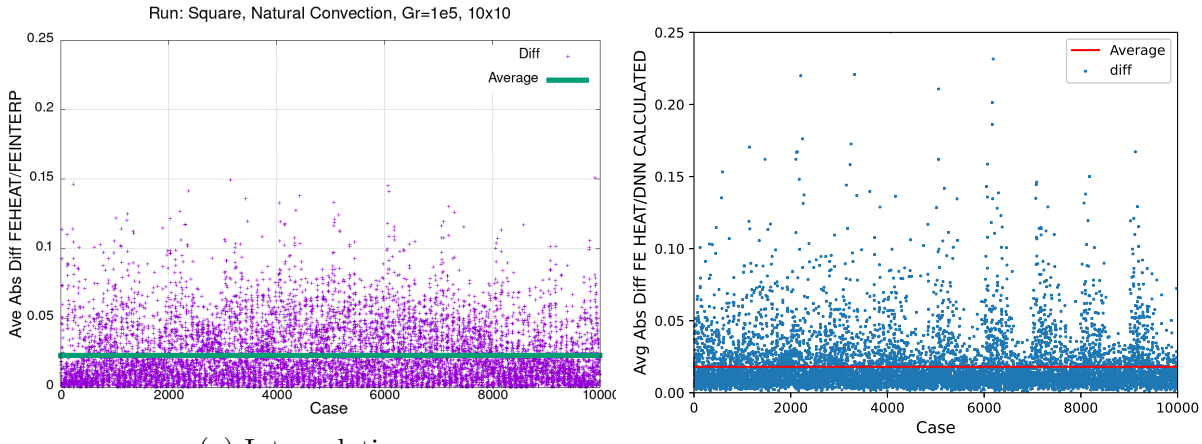
Figure 7: Forced convection problem. Average error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b)



(a) Interpolation

(b) Deep Learning

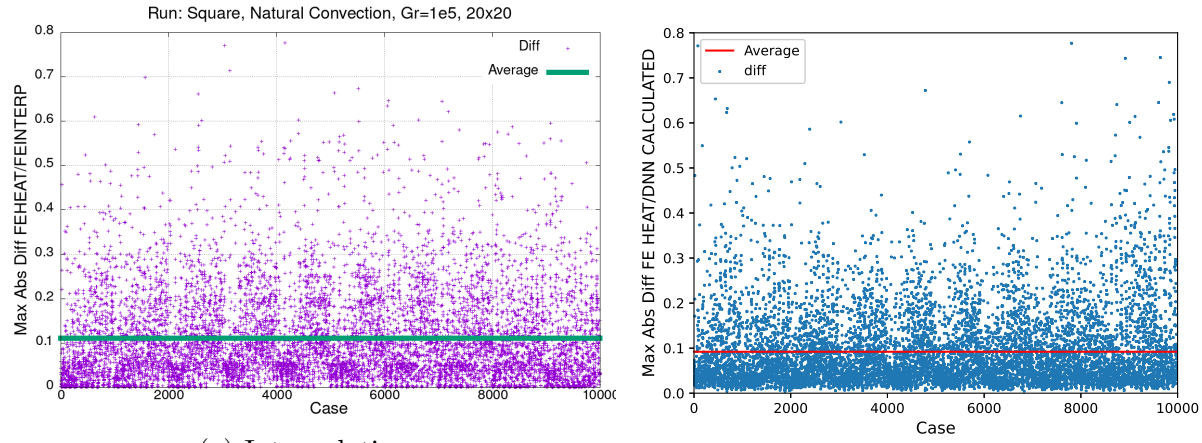
Figure 8: Natural convection problem. Maximum error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b), 10x10 structured mesh



(a) Interpolation

(b) Deep Learning

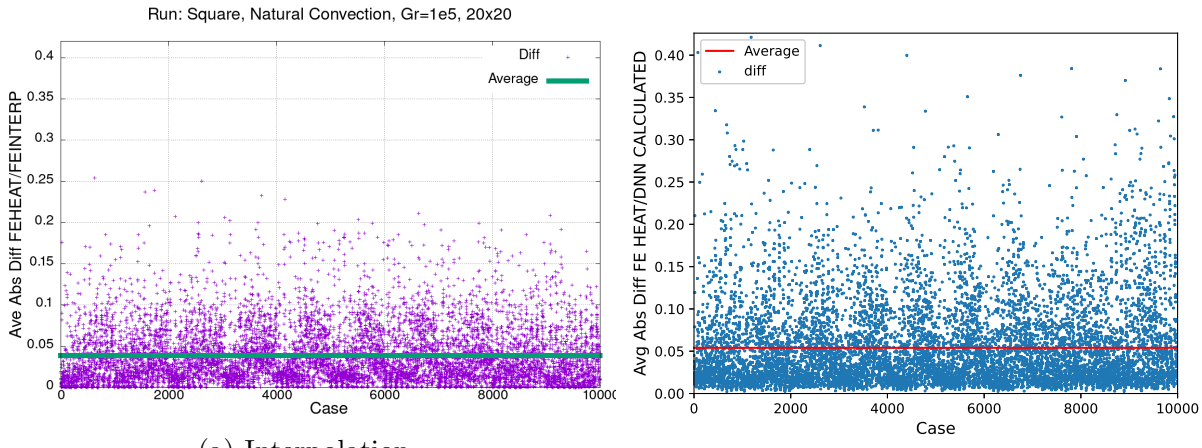
Figure 9: Natural convection problem. Average error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b), 10x10 structured mesh



(a) Interpolation

(b) Deep Learning

Figure 10: Natural convection problem. Maximum error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b), 20x20 structured mesh



(a) Interpolation

(b) Deep Learning

Figure 11: Natural convection problem. Average error between computed and estimated temperatures T1-T4; Interpolation (a) vs Deep Learning (b), 20x20 structured mesh

- IPs perform better than DNNs for linear heat conduction problems;
- DNNs perform better than IPs for nonlinear heat conduction problems;
- IPs and DNNs perform similarly for forced and natural convection problems.

Acknowledgements

H. Antil is partially supported by NSF Grants DMS-1818772, DMS-1913004, the Air Force Office of Scientific Research (AFOSR) under Award No.: FA9550-19-1-0036, and Department of Navy, Naval PostGraduate School under Award No.: N00244-20-1-0005.

P.Nithiarasu acknowledges partial support from Ser Cymru III - Tackling Covid 19 fund, Welsh Government Project number 095.

The authors would also acknowledge the very thorough review and comments of an anonymous reviewer, which was very constructive and helped to improve the archival quality of the paper.

References

[1] Hamid Reza Tamaddon-Jahromi, Neeraj Kavan Chakshu, Igor Sazonov, Llion M. Evans, Hywel Thomas, and Perumal Nithiarasu. Data-driven inverse modelling through neural network (deep learning) and computational heat transfer. *Computer Methods in Applied Mechanics and Engineering*, 369:113217, 2020.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear approximation and (deep) ReLU networks. *arXiv preprint arXiv:1905.02199*, 2019.

[4] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020.

[5] 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.

[6] Daniel Z. Huang, Kailai Xu, Charbel Farhat, and Eric Darve. Learning constitutive relations from indirect observations using deep neural networks. *J. Comput. Phys.*, 416:109491, 2020.

[7] Kurt Kornelsen and Paulin Coulibaly. Comparison of interpolation, statistical, and data-driven methods for imputation of missing values in a distributed soil moisture dataset. *Journal of Hydrologic Engineering*, 19(1):26–43, 2014.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Atsuya Oishi and Genki Yagawa. Computational mechanics enhanced by deep learning. *Computer Methods in Applied Mechanics and Engineering*, 327:327–351, 2017.

[10] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.

[11] Stefanie Günther, Lars Ruthotto, Jacob Schroder, Eric Cyr, and Nicolas Gauger. Layer-parallel training of deep residual neural networks. *SIAM Journal on Mathematics of Data Science*, 2:1–23, 01 2020.

[12] Harbir Antil, Ratna Khatri, Rainald Löhner, and Deepanshu Verma. Fractional deep neural network via constrained optimization. *arXiv preprint arXiv:2004.00719*, 2020.

[13] Harbir Antil, Zichao Wendy Di, and Ratna Khatri. Bilevel optimization, deep learning and fractional laplacian regularizatin with applications in tomography. *Inverse Problems*, 36(6):064001, may 2020.

[14] Jason M. Carson, Neeraj Kavan Chakshu, Igor Sazonov, and Perumal Nithiarasu. Artificial intelligence approaches to predict coronary stenosis severity using non-invasive fractional flow reserve. *Proceedings of the Institution of Mechanical Engineers Part H-Journal of Engineering in Medicine*, DOI:10.1177/0954411920946526, 2020.

[15] Neeraj Kavan Chakshu, Igor Sazonov, and Perumal Nithiarasu. Towards enabling a cardiovascular digital twin for human systemic circulation using inverse analysis. *Biomechanics and Modeling in Mechanobiology*, DOI:10.1007/s10237-020-01393-6, 2020.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] R Löhner and J McAnally. Transient and steady heat conduction using an adaptive finite element cad-based approach. *International Journal of Numerical Methods for Heat & Fluid Flow*, 4:311–327, 1994.

[18] Perumal Nithiarasu, Roland W. Lewis, and Kankanhalli N. Seetharamu. *Fundamentals of the finite element method for heat and fluid flow*. John Wiley & Sons, 2016.

[19] R Löhner, C Yang, JR Cebral, F Camelli, O Soto, and J Waltz. Improving the speed and accuracy of projection-type incompressible flow solvers. *Computer Methods in Applied Mechanics and Engineering*, 195:2087–3109, 2006.