Physics Informed Neural Networks for Two Dimensional Incompressible Thermal Convection Problems

Atakan Aygun * and Ali Karakus

Department of Mechanical Engineering, Middle East Technical University, Ankara, Turkey 06800
atakana@metu.edu.tr, ORCID: 0000-0003-4399-1935
akarakus@metu.edu.tr, ORCID: 0000-0002-9659-6712

Abstract Physics informed neural networks (PINNs) have drawn attention in recent years in engineering problems due to their effectiveness and ability to tackle the problems without generating complex meshes. PINNs use automatic differentiation to evaluate differential operators in conservation laws and hence do not need to have a discretization scheme. Using this ability, PINNs satisfy governing laws of physics in the loss function without any training data. In this work, we solve various incompressible thermal convection problems including real applications and problems with comparable numerical or analytical results. We first consider a channel problem with an analytical solution to show the accuracy of our network. Then, we solve a thermal convection problem in a closed enclosure in which the flow is only due to the temperature gradients on the boundaries. Lastly, we consider steady and unsteady partially blocked channel problems that resemble industrial applications to power electronics.

Keywords: physics informed neural networks, machine learning, automatic differentiation, incompressible, heat transfer.

*Corresponding author.
INTRODUCTION

Thermal convection problems arise in many practical engineering applications such as cooling of electronic chips. This type of real life analysis of fluid flow and heat transfer require high degrees of freedom to minimize the numerical error. This can be achieved with high quality mesh or high order discretizations. However, mesh generation is time consuming and need expertise and high order simulation tools for this type of problems are computationally demanding.

The incompressible thermal convection is studied in the literature with various numerical methods (Baïri et al., 2014). Tang and Tsang (1993) used a least squares finite element method based on a velocity, pressure, vorticity, temperature, heat flux formulation for time dependent problems. Hossain et al. (2021) developed a spectral/hp element method for the Direct Numerical Simulation (DNS) of incompressible thermal convective flows by considering Boussinesq type thermal body-forcing with periodic boundary conditions and enforcing a constant volumetric flow rate. In Karakus (2022), the author presented a GPU accelerated nodal discontinuous Galerkin method on unstructured triangular meshes for solving problems on different convective regimes.

Apart from the traditional numerical methods such as finite difference, finite volume and finite element methods, data driven machine learning methods are used to solve the partial differential equations (PDE) (Willard et al., 2020). These regression methods offer effective and mesh free approaches (Karagiannis et al., 2021). Neural networks were employed first to solve the PDEs as in Lee and Kang (1990) and Lagaris et al. (1998), and in Raissi et al. (2017a) and Raissi et al. (2017b) the authors employed Gaussian processes regression to accurately predict the solution and provide the uncertainty in the model. Raissi et al. (2019) introduced the concept of physics informed neural networks (PINNs) that use automatic differentiation (Baydin et al., 2017) to solve forward and inverse problems for several types of PDEs.

To solve PDEs using PINNs, generally fully connected networks are used. However, for different type of problems, plain fully connected networks perform poorly. In the learning process, these networks have a learning bias towards low frequency functions that is called spectral bias (Rahaman et al., 2019). This reduces the accuracy in which the target function exhibit high frequency or multi-scale behavior. To overcome this problem, Tancik et al. (2020) and Wang et al. (2021b) proposed Fourier feature mapping of the input vectors before feeding them into the network. The input coordinates $\mathbf{v}$ is mapped with a function $\gamma(\mathbf{v}) = [a_1 \cos(2\pi b_1^T \mathbf{v}), a_1 \sin(2\pi b_1^T \mathbf{v}), \ldots, a_m \cos(2\pi b_m^T \mathbf{v}), a_m \sin(2\pi b_m^T \mathbf{v})]$ and then passed into the multi layer perceptron (MLP). This mapping transforms the Neural Tangent Kernel (NTK) (Jacot et al., 2018) into a stationary kernel, and enables controlling the learning of the range of frequencies by modifying the frequency vectors $\mathbf{b}$ in the mapping function. Tancik et al. (2020) shows the performance of this method in many low-dimensional tasks in computer vision and Wang et al. (2021b) shows its efficiency for challenging problems involving partial differential equations with multi-scale behavior where conventional PINN models might have issues, such as wave propagation and reaction-diffusion equations.

In another approach, Esmaeilzadeh et al. (2020), proposed a PDE constrained deep learning algorithm that reconstructs high resolution solutions using the low resolution physical solutions in space and time and solved the well known Rayleigh-Bénard instability problem. This model is referred as super-resolution model and enables effectively scaling to large domains and having physically valid solutions by regularizing the outputs with PDE constraints. The framework is named as MeshfreeFlowNet and
consists of two subnetworks. The first network, named as Context Generation Network, is a convolutional encoder that takes low resolution physical input and creates a Latent Context Grid. This grid contains latent context vectors along with the spatio-temporal coordinates and these values are fed into another network called Continuous Decoding network modeled as a MLP. This framework allows the output to be continuous instead of a discrete output therefore removing the output resolution limitations. In addition, this continuous output allows to effectively compute the gradients of the output, therefore enables to enforce the PDE based constraints.

Due to the popular deep learning frameworks such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019), and their easy implementation, PINNs become quite popular for solving PDEs. It is used for solving incompressible and compressible Navier-Stokes equations (Rao et al., 2020; Jin et al., 2021; Cai et al., 2022), as well as in heat transfer problems (Cai et al., 2021). There are also software libraries specifically designed for physics informed machine learning such as DeepXDE (Lu et al., 2021) and NeuralPDE (Zubov et al., 2021).

PINNs do not require mesh generation. Instead, the PDEs and any other constraints can be directly enforced into the loss function of the neural network using automatic differentiation by forcing the prediction to the target value. This includes zero residuals for conservation laws and satisfying the boundary/initial conditions. In this work, we focus on the application of PINNs to coupled fluid flow and heat transfer problems. We present and validate our approach for different thermal convection regimes. The remainder of this paper is organized as follows. First, we present the mathematical formulation of incompressible Navier-Stokes equations coupled with the energy equation through Boussinesq approximation. Then we give brief information about the physics informed neural networks and their loss functions which is followed by the 2D numerical validation cases. The last section is about concluding remarks and future works.

FORMULATION

We consider a two-dimensional domain $\Omega \subset \mathbb{R}^2$ and denote the boundary of $\Omega$ by $\partial \Omega$. Following the notation presented in Karakus et al. (2019), we denote the Dirichlet and Neumann boundary conditions as $\partial \Omega_D$ and $\partial \Omega_N$ respectively on $\partial \Omega$. We are interested in the approximation of non-isothermal incompressible Navier-Stokes equations coupled by the energy equation through Boussinesq approximation which reads

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times (0, T]$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \frac{1}{Re} \Delta \mathbf{u} + \mathbf{s}_u \quad \text{in } \Omega \times (0, T],$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u} \cdot \nabla) \theta = \frac{1}{RePr} \Delta \theta + \mathbf{s}_\theta \quad \text{in } \Omega \times (0, T],$$

in non-dimensional form and subject to the initial conditions

$$\mathbf{u} = \mathbf{u}_0, \quad \theta = \theta_0 \quad \text{for } t = 0, \mathbf{x} \in \Omega,$$
and the boundary conditions

\[
\begin{align*}
\mathbf{u} &= g_D \text{ on } \mathbf{x} \in \partial\Omega_D, t \in (0, T), \\
\mathbf{u} \cdot \mathbf{n} &= 0, \quad p = 0 \text{ on } \mathbf{x} \in \partial\Omega_N, t \in (0, T), \\
\theta &= g_D \text{ on } \mathbf{x} \in \partial\Omega_D, t \in (0, T), \\
\theta \cdot \mathbf{n} &= g_N \text{ on } \mathbf{x} \in \partial\Omega_N, t \in (0, T).
\end{align*}
\]

(3.1) (3.2) (3.3) (3.4)

Here \( \mathbf{u}, p \) and \( \theta \) are non-dimensional velocity, static pressure and temperature fields, respectively. To get the non-dimensional representation of the equation, following approach is used.

\[
\begin{align*}
x &= \frac{x^*}{L_r}, \quad t = \frac{t^*}{L_r/U_r}, \quad u = \frac{u^*}{U_r}, \quad p = \frac{p^*}{\rho_r U_r^2}, \\
\rho &= \frac{\rho^*}{\rho_r}, \quad \nu = \frac{\nu^*}{\nu_r}, \quad \alpha = \frac{\alpha^*}{\alpha_r}, \quad \theta = \frac{T - T_r}{T_s}.
\end{align*}
\]

Here the superscript * denotes the dimensional parameters and subscript \( r \) denotes the reference values for specific terms with reference length scale \( L_r \), velocity \( U_r \), density \( \rho_r \), viscosity \( \nu_r \), thermal diffusivity \( \alpha_r \), and temperature \( T_r \). Also, \( Re \) is the Reynolds number and \( Pr \) is the Prandtl number. \( s_u = (g\beta (T - T_r) L_r / U_r) \theta \) is the forcing term for Navier-Stokes, where \( g \) is the gravitational acceleration, \( \beta \) is the expansion coefficient and subscript \( r \) refers the reference value for the corresponding field. \( s_\theta = s_\theta (\theta, \nabla \theta, \mathbf{u}) \) is the generic generation term for the energy equation written in terms of temperature. The superscripts \( \mathbf{u} \) and \( \theta \) in boundary representation separate the Dirichlet and Neumann conditions, represented with the subscripts \( D \) and \( N \), on the physical boundary set for flow and heat transfer equations.

**PHYSICS INFORMED NEURAL NETWORKS**

Consider a general partial differential equation expressed as:

\[
\begin{align*}
\mathbf{u}_t + \mathcal{N}[\mathbf{u}] &= 0, \quad \mathbf{x} \in \Omega, \quad t \in [0, T] \\
\mathbf{u}(\mathbf{x}, 0) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega \\
\mathbf{u}(\mathbf{x}, t) &= g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega, \quad t \in [0, T]
\end{align*}
\]

(5.1) (5.2) (5.3)

where \( \mathcal{N} \) is a generalized differential operator that can be linear or nonlinear, \( \mathbf{x} \in \mathbb{R}^d \) and \( t \) are the spatial and temporal coordinates. \( \Omega \) and \( \partial\Omega \) represent the computational domain and the boundary respectively. \( \mathbf{u}(\mathbf{x}, t) \) is the general solution of the PDE with \( f(\mathbf{x}) \) is the initial condition and \( g(\mathbf{x}, t) \) is the boundary condition.

The solution \( \mathbf{u}(\mathbf{x}, t) \) can be approximated by a fully connected network according to the framework of physics informed neural networks (PINN) proposed by Raissi et al. (2019). This network takes the spatiotemporal coordinates \( (\mathbf{x}, t) \) as input and outputs a solution \( \mathbf{u}_{NN}(\mathbf{x}, t) \). Between these input and output layers, there exists multiple hidden layers. Each hidden layer takes input \( X = [x_1, x_2, \ldots, x_i] \),
and outputs $Y = [y_1, y_2, \ldots, y_j]$ through a nonlinear activation function $\sigma(\cdot)$ such as:

$$y_j = \sigma(w_{i,j}x_j + b_j), \quad (6)$$

where $w_{i,j}$ and $b_j$ are trainable hyperparameters, weights and biases, respectively. These hyperparameters are tuned such that it minimizes a composite loss function in the form

$$\mathcal{L} = \omega_D \mathcal{L}_D + \omega_R \mathcal{L}_R + \omega_{BC} \mathcal{L}_{BC} + \omega_{IC} \mathcal{L}_{IC}, \quad (7)$$

where

$$\mathcal{L}_D = \frac{1}{N_D} \sum_{i=1}^{N_D} |u(x^i, t^i) - u^i|^2 \quad (8.1)$$

$$\mathcal{L}_R = \frac{1}{N_R} \sum_{i=1}^{N_R} |u_i + N[u(x^i, t^i)]|^2 \quad (8.2)$$

$$\mathcal{L}_{BC} = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} |u(x^i, t^i) - g(x^i, t^i)|^2 \quad (8.3)$$

$$\mathcal{L}_{IC} = \frac{1}{N_{IC}} \sum_{i=1}^{I_{IC}} |u(x^i, 0) - f(x^i)|^2. \quad (8.4)$$

$\mathcal{L}_D$ is a loss term indicating the accuracy of the prediction if there are some data available in the domain, $\mathcal{L}_R$, $\mathcal{L}_{BC}$, and $\mathcal{L}_{IC}$ represent the residual of the governing PDE, the boundary conditions, and the initial condition respectively. $N_D$, $N_R$, $N_{BC}$, and $N_{IC}$ are the number of data points for different terms, $\omega_D$, $\omega_R$, $\omega_{BC}$ and $\omega_{IC}$ are user specified weighting coefficients of each loss term. To calculate the residuals for $\mathcal{L}_R$, one need to take the derivative of the output $u$ with respect to inputs $(x, t)$. In PINNs, this can be achieved by using automatic differentiation (Baydin et al., 2017). Automatic differentiation applies chain rule repeatedly to the elementary functions and arithmetic operations to achieve the derivative of the overall composition. It plays a key role in the development of PINNs by enabling the computation of the residual of the governing differential equation (Raissi et al., 2019). Automatic differentiation is well implemented in most deep learning frameworks such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019).

Figure 1 illustrates a schematic of the PINN framework in which the residuals of the coupled Navier-Stokes and energy equations are shown as the loss terms. This general schematic of the neural network takes the spatio-temporal coordinates as the input, and through the hidden layers it outputs the pressure, velocity, and temperature fields. After this output, the framework calculates the boundary losses from the boundary conditions and the data loss if there are any observations. In the next step, PINN uses automatic differentiation and calculates the residual inside the domain by enforcing the neural network output to the Navier-Stokes and energy equations. Also, if the boundary conditions are Neumann type, the neural network output can be differentiated to find the boundary loss. Then the overall loss can be calculated by adding the residual loss from the PDE, the boundary loss and if there exists any available observations, the data loss. This combined loss function is minimized by an optimization algorithm via changing the hyperparameters.
RESULTS

We have implemented our physics informed neural network on top of the NVIDIA Modulus framework (Hennigh et al., 2021). We use the Adam optimizer (Kingma and Ba, 2017) to minimize the loss function defined in Equation 7, and use 8 hidden layers with 40 units for the each test case where the parameters of the neural network initialized using the Glorot scheme (Glorot and Bengio, 2010). We solve different 2D thermal convection tests to show the solutions by representing the velocity, pressure and the temperature fields.

Poiseuille Flow

In the first test case, we consider two-dimensional channel flow with a fully developed Poiseuille profile. The channel dimension is $[0, 2] \times [-1, 1]$. The upper and lower walls have constant temperature of $\theta_L = 1$ and $\theta_U = 0$. No-slip boundary conditions are imposed for upper and lower walls. The fully developed solution of the velocity field with linear temperature profile shown below is implemented as the boundary conditions of the inlet and the outlet. The flow conditions are stated as $Ra = 10^3$, $Pr = 0.71$ and $Re = 100$.

$$u = 1 - y^2, \quad v = 0, \quad p = \frac{Ra}{2PrRe^2} \left( y - \frac{y^2}{2} \right) - \frac{2x}{Re}, \quad \theta = \frac{1 - y}{2}$$

We trained our framework with 250 samples inside the domain and 30 samples on each boundary with 10000 iterations for this case. The training points are sampled using Latin hypercube sampling, and the loss function for this problem contains only the Dirichlet boundary condition loss for the velocity and the temperature on the walls combined with the residual loss inside the domain. After training, we performed a prediction on a $(251 \times 251)$ grid and obtained the velocity, pressure, and temperature fields. The predicted fields can be seen in Figure 2. The accuracy of the PINN is highly dependent on the weights of the loss function. In this case we tried different weights of the different terms of the loss function to match our solution with the exact solution. Especially for an accurate pressure field we increased the weights of the boundary condition losses. The solution in Figure 2 is obtained with a
boundary loss weight $\omega_{BC}$ which is eight times higher than the weight of the residual loss $\omega_R$. Since the convective effects are not very dominant for this problem, boundary losses are dominant, so increasing the boundary loss weights increases the accuracy.

![Figure 2: Prediction of the Poiseuille flow with PINN. The $u$ velocity, pressure and the temperature fields are shown in order. Black contours show the exact solution while the red dashed contours show the solution with PINN.](image)

Table 1: $L_2$ norm of the error of the predicted $u$ velocity and the temperature fields.

| Number of Observations | $u$   | $T$   |
|------------------------|-------|-------|
| 20                     | 0.527 | 0.087 |
| 50                     | 0.253 | 0.057 |
| 100                    | 0.166 | 0.034 |
| 150                    | 0.141 | 0.032 |

We test the performance of the PINNs with the addition of true observations at random points on the domain. We fused different number of randomly sampled exact solutions inside the domain to the network and add a data loss term into the loss function. The training process is done with 250 points inside the domain and 30 boundary points on each boundary besides the true solution points. In Table 1, we can see the $L_2$ norm of the error of the predicted $u$ velocity and temperature fields. The number of observation represents the addition of the true solutions and increasing this number reduces the $L_2$ norm of the prediction of the velocity and the temperature from the true solution.

**Differentially Heated Square Cavity**

We focus on the natural convection problem on a two dimensional closed enclosure. The enclosure is a square cavity with its height denoted as $H = 1$, and width as $W = 1$. The boundary conditions of cavity are simple no-slip walls, $u = 0$, $v = 0$, on all four walls. The thermal boundary conditions on the left and right walls are prescribed as

$$\theta_L = 1, \quad \theta_R = 0.$$
and the upper and the lower walls are thermally insulated

\[ \frac{\partial \theta}{\partial y} = 0, \quad \text{for} \quad y = 0 \quad \text{and} \quad y = H. \]

The flow conditions are \( Pr = 0.71 \), and three different Rayleigh numbers as \( Ra = 10^3, 10^4, 10^5 \).

For the PINN solution, we sampled 150 points on each boundary, and 1000 collocation points inside the domain for the training process. Boundary points are used to minimize the loss of Dirichlet and Neumann boundary conditions, and collocation points are used to minimize the residual inside the domain. Automatic differentiation is used to calculate the derivatives on Neumann boundaries.

Table 2: Maximum and minimum velocities along the center lines of square cavity for \( Pr = 0.71 \) and \( Ra = 10^3, 10^4, 10^5 \).

| \( Ra \) | \( u_{\text{max}} \) | \( v_{\text{max}} \) | \( u_{\text{max}} \) | \( v_{\text{max}} \) | \( u_{\text{max}} \) | \( v_{\text{max}} \) |
|--------|----------------|----------------|----------------|----------------|----------------|----------------|
| \( 10^3 \) | 0.137 | 0.138 | 0.192 | 0.233 | 0.128 | 0.258 |
| \( 10^4 \) | 0.137 | 0.139 | 0.192 | 0.233 | 0.129 | 0.257 |
| \( 10^5 \) | 0.137 | 0.139 | 0.192 | 0.233 | 0.130 | 0.256 |

Karakus (2022) | 0.136 | 0.138 | 0.192 | 0.234 | 0.153 | 0.261 |

Stokos et al. (2015) | 0.132 | 0.261 |

De Vahl Davis (1983) | 0.153 | 0.261 |

After the training process, prediction is performed on a \((251 \times 251)\) grid. In Table 2, we presented the maximum and minimum velocities on the horizontal and vertical centerlines after the prediction with PINN. For cases with different \( Ra \) numbers, our framework have values that are comparable with the ones in the literature. In Figure 3, the solution of PINN and its comparison with a high fidelity solver through the temperature contours can be seen. Also in Figure 4, the center line profiles of velocity and temperature for different \( Ra \) numbers are shown. These contours and profiles qualitatively match with the high order solutions (Karakus, 2022). To increase the accuracy, we changed the weights of different loss terms as \( Ra \) changes. In Table 3, we presented the center line velocities for different \( Ra \) numbers and different weight ratio of the residual loss over the boundary loss where \( \omega_R \) represents the weight of the residual loss, and \( \omega_{BC} \) represents the weight of loss on the boundary conditions. We stopped changing weights when we match the center line velocities with the reference solutions. As the \( Ra \) increases, the convective effects inside the domain becomes more dominant. Hence we need to decrease the weight of the boundary losses, and focus more on the residual inside the domain. We select the loss ratio according to Table 3 that minimizes the error both inside the domain and on the boundaries.

Table 3: Maximum and minimum velocities along the center lines with different weight ratio of residual loss and the boundary loss.

| \( \omega_R/\omega_{BC} \) | \( Ra = 10^3 \) | \( Ra = 10^4 \) | \( Ra = 10^5 \) |
|-----------------|----------------|----------------|----------------|
|                 | \( u_{\text{max}} \) | \( v_{\text{max}} \) | \( u_{\text{max}} \) | \( v_{\text{max}} \) | \( u_{\text{max}} \) | \( v_{\text{max}} \) |
| 0.5             | 0.137 | 0.138 | 0.190 | 0.231 | 0.137 | 0.273 |
| 1               | 0.192 | 0.233 | 0.128 | 0.258 |
| 2               | 0.132 | 0.261 |
| 4               | 0.130 | 0.261 |

We test different types of neural network architectures and monitored the behavior of the total loss of the Adam optimizer for the cavity problem with \( Ra = 10^3 \) and presented in Figure 5. The plain
Figure 3: Temperature contours for the square cavity test. The high fidelity solution obtained with high fidelity discontinuous Galerkin solver between 1 and 0 with the increment of 0.05 for $Ra = 10^3, 10^4, 10^5$ from left to right shown with the black contours while the red dashed contours are the solution with PINNs.

Figure 4: Velocity and temperature profiles along the $y = 0.5$ and $x = 0.5$ lines for different $Ra$ numbers. The first row shows the values obtained with the PINN, while the second row shows the values of high order discontinuous Galerkin solver.

fully connected network (FCN), a variation of the fully connected network named as Deep Galerkin Method (DGM) (Sirignano and Spiliopoulos, 2018), a Fourier network and a modified Fourier network (Wang et al., 2021a), a modified highway network using Fourier features (Srivastava et al., 2015), and a multiplicative filter network (Fathony et al., 2021) are used. All of these architectures are readily available in NVIDIA Modulus framework. In all the tests 8 layer networks are constructed with 40 units. Hyperbolic tangent is set as the activation function and the learning rate is $10^{-3}$. The architectures that use Fourier mapping converges later than the plain fully connected network since the problem does not
have multi-scale behavior. For this simple problem we do not need a Fourier mapping, hence networks that are basically build on plain fully connected networks converges in less iterations.

![Graph](image)

Figure 5: Behavior of the total loss in the square cavity problem with \( Ra = 10^3 \) with different types of neural network architectures.

**Heated Block**

In this section we focus on an application of coupled heat transfer with a heat transfer in a partially blocked channel. The domain and the boundary conditions can be seen in Figure 6. The heated block represents an electronic part on a vertical electronic board (Habchi and Acharya, 1986). The top wall is adiabatic and the bottom wall is at a prescribed temperature. A low temperature flow comes from the inlet and the outflow is a fully developed outlet meaning the changes in the \( x \) direction is zero. The Prandtl number is set to 0.7 for this problem and the Reynolds number is 37.8. The ratio of \( Gr/Re^2 \) is 1 and the forcing is on the \( x \) direction.

For training the network, we sampled 30 points on the inlet and the outlet, 210 points on the adiabatic wall, total of 40 points on the heated block, total of 180 points on the bottom wall and 1400 points inside the domain. We used 25000 iterations for the Adam optimizer with the learning rate of \( 5 \times 10^{-4} \) and obtained the solution presented in Figure 7. PINN solution well predicts the Neumann boundary conditions on the top wall and the fully developed outlet, and the no-slip Dirichlet velocity conditions.

**Multiple Heated Blocks**

In this section we focus on a similar problem in which multiple blocks are present. The problem is time dependent and it is solved for a final time of 8 seconds. The geometric representation of the case is presented in Figure 9. The geometric parameters are given such that \( H = 1, H/w = 2.5, L/w = \)
Figure 6: Schematic of the partially blocked channel

Figure 7: Velocity and temperature profiles for the heated block case. The figure on the top shows the velocity profile and the figure on the bottom shows the temperature field predicted by the PINN.

25, \( h/w = 0.5 \) as presented in Wu and Perng (1999). At time \( t = 0 \), the initial condition is \( u = v = \theta = 0 \) in the domain. There is a uniform inflow with \( u = 1, v = 0 \) with the temperature of \( \theta = 0 \). The upper and bottom walls have no-slip conditions as the velocity boundary condition, and Neumann temperature boundary condition of \( \partial \theta / \partial n = 0 \). The blocks also have no-slip conditions and temperature boundary condition of \( \partial \theta / \partial n = -1 \). Gravity is on \( y \) direction, and flow parameters are given as \( Re = 400, Pr = 0.7, \) and \( Gr/Re^2 = 0.5 \).

In the training phase, 50 points on the inlet and the outlet is generated uniformly. 40 points on each block, 500 points on top wall and 450 points on the bottom wall is sampled. The problem is solved with continuous time approach such that we treat time \( t \) as another variable as the spatial coordinates instead of approaching the time sequentially. Adam optimizer is used again to find the optimized hyperparameters with a learning rate of \( 5 \times 10^{-4} \). The solution at the final time \( t = 8s \) is shown in Figure 9. The horizontal
velocity and the temperature fields are plotted and it can be seen that PINN solution represents the flow field well physically inside the domain and also satisfy the Dirichlet and Neumann boundary conditions.

Figure 9: Velocity and temperature profiles for the multiple heated blocks case at time $t = 8s$. The figure on the top shows the velocity profile and the figure on the bottom shows the temperature field predicted by the PINN.

CONCLUSION

In this work, we solve two dimensional incompressible thermal convection problems with physics informed neural networks. Using Boussinesq approximation, the incompressible Navier-Stokes equation is coupled with the energy equation written in terms of temperature. The loss function is constructed using these PDEs and necessary boundary/initial conditions. The PINN solution predicts the flow and temperature fields well compared to the analytical solutions and high order solvers in various problems. We show that adding observations into the flow field increases the accuracy of the prediction and the framework is very sensitive to the weights of the individual loss terms. Furthermore, we consider two different channel problems with partial blockage that resemble power electronics applications. The first
one is a simple steady problem and the other is an unsteady problem in which we tackle it with a con-
tinuous approach such that the time is treated as another variable and we obtained results that resembles
the flow physics well. For a future work, the time dependent problems can be implemented with time
marching approaches such as Recurrent Neural Networks or Gated Response Units.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G.,
Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker,
P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: A system
for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and
Implementation (OSDI 16), pages 265–283.

Baïri, A., Zarco-Pernia, E., and De María, J.-M. G. (2014). A review on natural convection in enclosures
for engineering applications. the particular case of the parallelogrammic diode cavity. Applied Thermal
Engineering, 63(1):304–322.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2017). Automatic differentiation in
machine learning: a survey. The Journal of Machine Learning Research, 18(1):5595–5637.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. (2022). Physics-informed neural networks
(PINNs) for fluid mechanics: a review. Acta Mechanica Sinica.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G. (2021). Physics-informed neural net-
works (PINNs) for heat transfer problems. Journal of Heat Transfer, 143.

De Vahl Davis, G. (1983). Natural convection of air in a square cavity: A bench mark numerical solution.
International Journal for Numerical Methods in Fluids, 3(3).

Esmaeilzadeh, S., Azizzadenesheli, K., Kashinath, K., Mustafa, M., Tchelepi, H. A., Marcus, P., Prab-
hat, M., Anandkumar, A., et al. (2020). Meshfreeflownet: A physics-constrained deep continuous
space-time super-resolution framework. In SC20: International Conference for High Performance
Computing, Networking, Storage and Analysis, pages 1–15. IEEE.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z. (2021). Multiplicative filter networks. In
International Conference on Learning Representations.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural
networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and
Statistics, pages 249–256. JMLR Workshop and Conference Proceedings. ISSN: 1938-7228.

Habchi, S. and Acharya, S. (1986). Laminar mixed convection in a partially blocked, vertical channel.
International Journal of Heat and Mass Transfer, 29(11):1711–1722.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Fang, Z., Rietmann, M.,
Byeon, W., and Choudhry, S. (2021). Nvidia SimNet™: An ai-accelerated multi-physics simulation
framework. In International Conference on Computational Science, pages 447–461. Springer.

Hossain, M. Z., Cantwell, C. D., and Sherwin, S. J. (2021). A spectral/hp element method for thermal
convection. International Journal for Numerical Methods in Fluids, 93(7):2380–2395.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in
neural networks. Advances in neural information processing systems, 31.
Jin, X., Cai, S., Li, H., and Karniadakis, G. E. (2021). NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951.

Karakus, A. (2022). An accelerated nodal discontinuous Galerkin method for thermal convection on unstructured meshes: Formulation and Validation. *Journal of Thermal Science and Technology*. Accepted.

Karakus, A., Chalmers, N., Świrydowicz, K., and Warburton, T. (2019). A GPU accelerated discontinuous Galerkin incompressible flow solver. *Journal of Computational Physics*, 390:380–404.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*.

Lagaris, I., Likas, A., and Fotiadis, D. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000.

Lee, H. and Kang, I. S. (1990). Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). 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.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017a). Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017b). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348:683–693.

Rao, C., Sun, H., and Liu, Y. (2020). Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212.

Sirignano, J. and Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Stokos, K., Vrahliotis, S., Pappou, T., and Tsangaris, S. (2015). Development and validation of an incompressible Navier-Stokes solver including convective heat transfer. *International Journal of Numerical Methods for Heat & Fluid Flow*, 25(4):861–886.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi,
R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547.

Tang, L. Q. and Tsang, T. T. (1993). A least-squares finite element method for time-dependent incompressible flows with thermal convection. *International Journal for Numerical Methods in Fluids*, 17(4):271–289.

Wang, S., Teng, Y., and Perdikaris, P. (2021a). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081.

Wang, S., Wang, H., and Perdikaris, P. (2021b). On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V. (2020). Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34.

Wu, H.-W. and Perng, S.-W. (1999). Effect of an oblique plate on the heat transfer enhancement of mixed convection over heated blocks in a horizontal channel. *International Journal of Heat and Mass Transfer*, 42(7):1217–1235.

Zubov, K., McCarthy, Z., Ma, Y., Calisto, F., Pagliarino, V., Azeglio, S., Bottero, L., Luján, E., Sulzer, V., Bharambe, A., Vinchhi, N., Balakrishnan, K., Upadhyay, D., and Rackauckas, C. (2021). NeuralPDE: Automating physics-informed neural networks (PINNs) with error approximations. *arXiv:2107.09443*. 
