Isı Bilimi ve Tekniği Dergisi,42, 2, 221-232, 2022
J. of Thermal Science and Technology
©2022 TIBTD Printed in Turkey
ISSN 1300-3615
https://doi.org/10.47480/isibted.1194992

PHYSICS INFORMED NEURAL NETWORKS FOR TWO DIMENSIONAL INCOMPRESSIBLE THERMAL CONVECTION PROBLEMS

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

(Geliş Tarihi: 27.04.2021, Kabul Tarihi: 12.09.2022)

Abstract: Physics-informed neural networks (PINNs) have drawn attention in recent years in engineering problems due to their effectiveness and ability to tackle problems without generating complex meshes. PINNs use automatic differentiation to evaluate differential operators in conservation laws and hence do not need 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, and compare the results with numerical or analytical results. To evaluate the accuracy of the model we solve a channel problem with an analytical solution. The model is highly dependent on the weights of individual loss terms. Increasing the weight of boundary condition loss improves the accuracy if the flow inside the domain is not complicated. To assess the performance of different type of networks and ability to capture the Neumann boundary conditions, we solve a thermal convection problem in a closed enclosure in which the flow occurs due to the temperature gradients on the boundaries. The simple fully connected network performs well in thermal convection problems, and we do not need a Fourier mapping in the network since there is no multiscale behavior. Lastly, we consider steady and unsteady partially blocked channel problems resembling industrial applications to power electronics and show that the method can be applied to transient problems as well.
Keywords: physics-informed neural networks, machine learning, automatic differentiation, incompressible, heat transfer.

SIKIŞTIRILAMAZ ISIL TAŞINIM PROBLEMLERİNİN FİZİKLE ÖĞRENEN YAPAY SİNİR AĞLARI İLE ÇÖZÜMÜ

Öz: Fizikle öğrenen yapay sinir ağları (PINN'ler), etkinlikleri ve karmaşık ağlar oluşturmadan problemlerin üstesinden gelme yetenekleri nedeniyle son yıllarda mühendislik problemlerinde dikkat çekmiştir. PINN'ler, koruma yasalarında diferansiyel operatörleri değerlendirmek için otomatik türevlenmeyi kullanır ve bu nedenle bir ayrıklaştırma şemasına ihtiyaç duymaz. Bu yeteneği kullanarak, PINN'ler herhangi bir eğitim verisi olmadan kayıp fonksiyonunda geçerli fizik yasalarını karşılar. Bu çalışmada, gerçek uygulamalar ve karşılaştırılabilir sayısal veya analitik sonuçlara sahip problemler de dahil olmak üzere çeşitli sıkıştırılamaz ısıl taşınım problemlerini çözüyoruz. Modelin performansını değerlendirmek için analitik çözümü olan bir kanal problemini çözüyoruz. Model, bireysel kayıp terimlerinin ağırlıklarına büyük ölçüde bağımlıdır. Alan içindeki akış çok karmaşık değilse, sınır koşulu kaybının ağırlığının arttırılması doğruluğu artırır. Farklı tipteki ağların performansını ve Neumann sınır koşullarını yakalama yeteneğini değerlendirmek için, sınırlardaki sıcaklık gradyanlarından dolayı akışın meydana geldiği kapalı bir muhafazada bir termal konveksiyon problemini çözüyoruz. Basit tam bağlantılı ağ, termal konveksiyon problemlerinde iyi performans gösterir ve çok ölçekli davranış olmadığından ağda Fourier dönüşümüne ihtiyacımız yoktur. Son olarak, endüstriyel uygulamaları güç elektroniğine benzeyen sabit ve kararsız kısmen bloke kanal problemlerini ele alıyoruz ve yöntemin geçici problemlere de uygulanabileceğini gösteriyoruz.
Anahtar Kelimler: fizikle öğrenen yapay sinir ağları, makine öğrenmesi, otomatik türevlenme, sıkıştırılamaz, ısı transferi.

NOMENCLATURE

Abbreviations

Gr	Grashof Number $\left[= g\beta(T - T_r){L_r}^3/\nu^2\right]$
Pr	Prandtl Number $[= \nu/\alpha]$
Ra	Rayleigh Number $[= GrPr]$
Re	Reynolds Number $[= U_r L_r/\nu]$
DGM	Deep Galerkin Method
DNS	Direct Numerical Simulation
FCN	Fully Connected Network
MLP	Multilayer Perceptron
NS	Navier-Stokes
NTK	Neural Tangent Kernel
PDE	Partial Differential Equation
PINN	Physics Informed Neural Network

Subscripts

u	Velocity Related Value
θ	Temperature Related Value
BC	Boundary Condition
D	Dirichlet Boundary
IC	Initial Condition
N	Neumann Boundary
R	Residual
r	Reference Value

Greek Symbols

α	Thermal Diffusivity $[= m^2/s]$
β	Expansion Coefficient $[= 1/°C]$
$\nabla \cdot$	Divergence Operator
∇	Gradient Operator
Δ	Laplace Operator
ν	Kinematic Viscosity $[= m^2/s]$
ω	Weight of loss terms
σ	Activation Function
θ	Non-dimensional Temperature

Symbols

s_u	Source Term of Momentum Equation
u	Velocity Vector
$\boldsymbol{\mathcal{L}}$	Loss Term of a Neural Network
$\mathcal{N}$	Generalized Differential Operator
g_D	Dirichlet Boundary Condition
g_N	Neumann Boundary Condition
L_2	L_2 Vector Norm of Error
p	Non-dimensional Pressure
s_θ	Source Term of Energy Equation
t	Non-dimensional Time

INTRODUCTION

Thermal convection problems arise in many practical engineering applications, such as cooling electronic chips. This type of real-life analysis of fluid flow and heat transfer requires 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 requires expertise. Also, high order simulation tools for these types 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, and 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 conventional 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 (Karniadakis et al., 2021). Neural networks were first employed 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. 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. The loss function includes zero residuals for conservation laws and satisfying the boundary/initial conditions.

To solve PDEs using PINNs, generally, fully connected networks are used. However, plain fully connected networks perform poorly for different types of problems. In the learning process, these networks have a learning bias toward low-frequency functions called spectral bias (Rahaman et al., 2019). This reduces the accuracy in which the target function exhibits 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 v is mapped with $\gamma(\boldsymbol{v}) = a_1 cos(2\pi \boldsymbol{b}_1^T \boldsymbol{v}), a_1 sin(2\pi \boldsymbol{b}_1^T \boldsymbol{v}), \ldots,$ $a_m \cos(2\pi \boldsymbol{b}_m^T \boldsymbol{v}), a_m sin(2\pi \boldsymbol{b}_m^T \boldsymbol{v})$ 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 $\boldsymbol{b}$ in the mapping function. Tancik et al. (2020) show the performance of this method in many low-dimensional tasks in computer vision. Wang et al. (2021b) show 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 to as super resolution model and enables effectively scaling to large domains and having physically reasonable solutions by regularizing the outputs with PDE constraints. The framework is named as MeshfreeFlowNet and consists of two subnetworks. The first network, called 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 an MLP. This framework allows the output to be continuous instead of a discrete output, removing the output resolution limitations. In addition, this continuous output allows us to effectively compute the output's gradients, enabling us 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 have become quite popular for solving PDEs. Moreover, some software libraries are designed explicitly for physics-informed machine learning, such as DeepXDE (Lu et al., 2021) and NeuralPDE (Zubov et al., 2021). 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 inverse heat transfer problems (Cai et al., 2021). To the best of our knowledge, solving forward thermal convection problems with PINNs, the effect of different neural networks on the accuracy, and the effect of the weights in the loss terms for different thermal convection regimes are not studied in detail.

In this work, we present the application of PINNs to coupled fluid flow and heat transfer problems in different thermal convection regimes. In particular, we examined the effectiveness of specific weights assigned to the different loss terms in the composite loss function. Changing these specific weights can increase the accuracy of the model according to the problem where boundary conditions or flow inside the computational domain dominates the residual. In addition, for thermal convection problems, different types of networks can be used instead of simple, fully connected networks. These various networks can change the convergence of the model.

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, followed by the 2D numerical validation cases. The last section is about concluding remarks and future works.

FORMULATION

We consider a closed two-dimensional domain $\Omega \subset R^2$ and denote the boundary of Ω by $\partial\Omega$. Following the notation presented in Karakus et al. (2019b), we assume that $\partial\Omega$ can be partitioned into two non-overlapping regions denoted by $\partial\Omega_D$ and $\partial\Omega_N$ referring prescribed Dirichlet or Neumann boundary conditions, respectively. We are interested in the approximation of non-isothermal incompressible Navier-Stokes equations coupled by the energy equations through Boussinesq approximation which reads:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re}\Delta \boldsymbol{u} + \boldsymbol{s_u}, \tag{1.2}$$

$$\frac{\partial \theta}{\partial t} + (\boldsymbol{u} \cdot \nabla)\theta = \frac{1}{Re\,Pr}\Delta\theta + s_\theta, \tag{1.3}$$

in non-dimensional form and space-time slap $\Omega \times (0, \mathcal{T}]$ subject to the initial conditions

$$\boldsymbol{u} = \boldsymbol{u_0}, \theta = \theta_0 \ \ for \ \ t = 0, \boldsymbol{x} \in \Omega, \tag{2}$$

and the boundary conditions

$$\boldsymbol{u} = \boldsymbol{g_D} \ \ on \ \ x \in \partial\Omega_D^u, t \in (0, \mathcal{T}], \tag{3.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = 0, p = 0 \ \ on \ \ \boldsymbol{x} \in \partial\Omega_N^{\boldsymbol{u}}, t \in (0, \mathcal{T}], \tag{3.2}$$

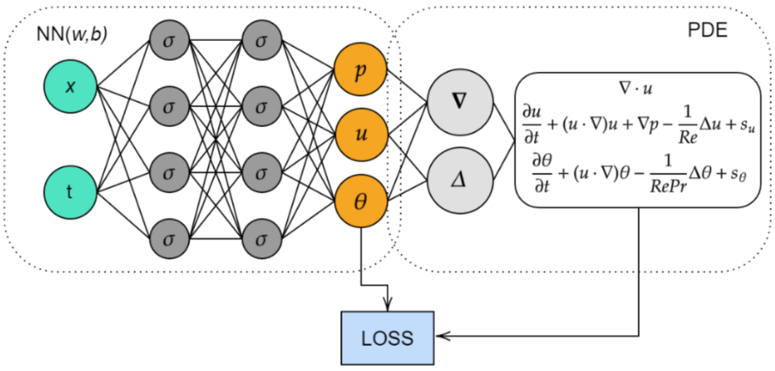
$$\theta = g_D \ \ on \ \ x \in \partial\Omega_D^\theta, t \in (0, \mathcal{T}], \tag{3.3}$$

$$\frac{\partial \theta}{\partial n} = g_N \ \ on \ \ x \in \partial\Omega_N^\theta, t \in (0, \mathcal{T}]. \tag{3.4}$$

Here u, p, and θ are non-dimensional velocity, static pressure, and temperature fields, respectively. In the equation, following parameters are used to get dimensionless quantities,

$$\begin{aligned} x &= \frac{x^*}{L_r}, t = \frac{t^*}{L_r/U_r}, \boldsymbol{u} = \frac{\boldsymbol{u}^*}{U_r}, p = \frac{p^*}{\rho_r U_r^2} \\ \rho &= \frac{\rho^*}{\rho_r}, \nu = \frac{\nu^*}{\nu_r}, \alpha = \frac{\alpha^*}{\alpha_r}, \theta = \frac{T - T_r}{T_s}, \end{aligned} \tag{4}$$

where superscript $*$ denotes the dimensional parameter, and the subscript r refers to the corresponding reference value i.e., reference length scale L_r, velocity U_r, density ρ_r, viscosity ν_r, thermal diffusivity α_r and temperature T_r. The non-dimensional Reynolds and Prandtl numbers are defined as $Re = U_r L_r/\nu_r$ and $Pr = \nu_r/\alpha_r$. $\boldsymbol{s_u} = (\boldsymbol{g}\beta(T - T_r)L_r/U_r)\theta$ is the forcing term for Navier-Stokes, where $\boldsymbol{g}$ is the gravitational acceleration, β is the expansion coefficient. In free convection problems, the reference velocity is selected as $U_r = g\beta(T - T_r)L_r$. $s_\theta = s_\theta(\theta, \nabla\theta, \boldsymbol{u})$ is the generic generation term for the energy equation written in terms of temperature. We would like to emphasize that superscripts u and θ in boundary representation separate the Dirichlet and Neumann conditions on the physical boundary set for flow and heat transfer equations.



NN(*w*,*b*)
x
t
σ
p
u
θ
PDE
∇
Δ
$\nabla \cdot u$
$\frac{\partial u}{\partial t} + (u \cdot \nabla)u + \nabla p - \frac{1}{Re}\Delta u + s_u$
$\frac{\partial \theta}{\partial t} + (u \cdot \nabla)\theta - \frac{1}{RePr}\Delta \theta + s_\theta$
LOSS

Figure 1. Schematic of a PINN framework

PHYSICS-INFORMED NEURAL NETWORKS

Consider a general partial differential equation expressed as:

$$\boldsymbol{u}_t + \mathcal{N}[\boldsymbol{u}] = 0, \qquad \boldsymbol{x} \in \Omega, \mathrm{t} \in [0, T] \qquad (5.1)$$

$$\boldsymbol{u}(x, 0) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega \qquad (5.2)$$

$$\boldsymbol{u}(x, t) = g(\boldsymbol{x}, t), \qquad \boldsymbol{x} \in \partial\Omega, \mathrm{t} \in [0, T] \qquad (5.3)$$

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

The solution $u(x, t)$ 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 spatio-temporal coordinates (x, t) as input and outputs a solution $u_{NN}(x, t)$. Between these input and output layers, there exist multiple hidden layers. Each hidden layer takes input $X = [x_1, x_2, \dots, x_i]$ and outputs $Y = [y_1, y_2, \dots, y_j]$ through a nonlinear activation function $\sigma(\cdot)$ such as

$$y_j = \sigma(\omega_{i,j} + b_j), \qquad (6)$$

where $\omega_{i,j}$ 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

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

where

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{N_D} \sum_{i=1}^{N_D} \left| u(x^i, t^i) - u^i \right|^2 \qquad (8.1)$$

$$\mathcal{L}_{\mathcal{R}} = \frac{1}{N_R} \sum_{i=1}^{N_R} \left| u_t + \mathcal{N}[u(x^i, t^i)] \right|^2 \qquad (8.2)$$

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

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

$\mathcal{L}$ is a loss term indicating the accuracy of the prediction if there are some data available in the domain, $\mathcal{L}_{\mathcal{R}}, \mathcal{L}_{BC}$, and $\mathcal{L}_{IC}$ represent the residual of the governing PDE, the boundary conditions, and the initial condition, respectively. $\mathcal{N}_{\mathcal{D}}, \mathcal{N}_R, \mathcal{N}_{BC}$, and $\mathcal{N}_{IC}$ are the number of data points for different terms, $\omega_D, \omega_R, \omega_{BC}$ and ω_{IC} are user specified weighting coefficients of each loss term. To calculate the residuals for $\mathcal{L}_{\mathcal{R}}$, one needs to take the derivative of the output $\boldsymbol{u}$ with respect to inputs $(\boldsymbol{x}, t)$. In PINNs, this can be achieved by using automatic differentiation (Baydin et al., 2017). Automatic differentiation applies the 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 exist any available observations, the data loss. An optimization algorithm minimizes this combined loss function 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 each test case where the neural network parameters are 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 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 a 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 the linear temperature profile shown below is implemented as the boundary conditions of the inlet and the outlet. the flow conditions are stated as $Ra = 10^3$, $Pr = 0.71, and\ Re = 100$.

$$u = 1 - y^2, \quad v = 0,$$

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

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 (251×251) 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 ω_{BC} which is eight times higher than the weight of the residual loss ω_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.

We test the performance of the PINNs with the addition of true observations at random points on the domain. We fused a different number of randomly sampled exact solutions inside the domain to the network and added 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 beside 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 observations 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.

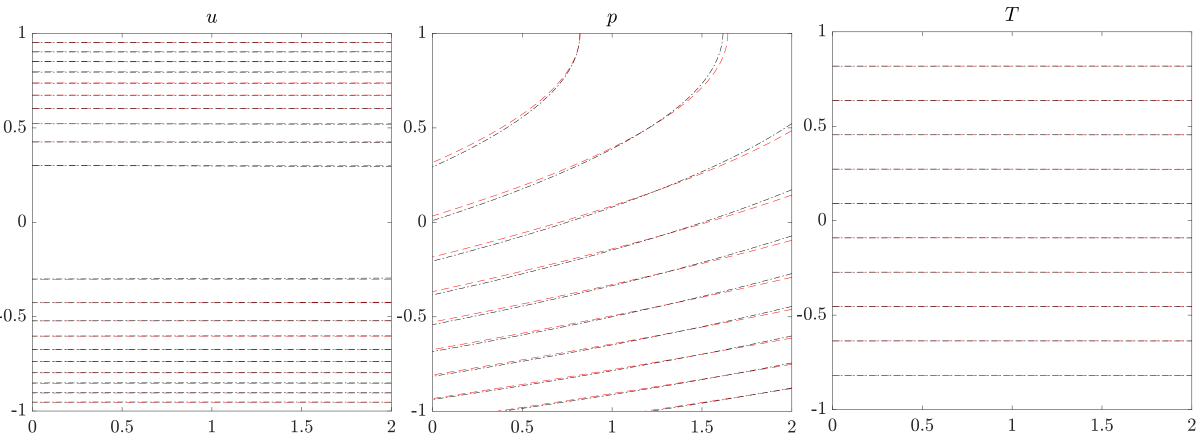


Figure 2. Prediction of Poiseuille flow with PINN. The u velocity, pressure and temperature fields are shown in order. Black contours show the exact solution and red dashed contours show the solution with PINN

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

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 the cavity are simple no-slip walls, $= 0, v = 0$, 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

$$\frac{\partial \theta}{\partial y} = 0, \quad for \quad y = 0, y = H.$$

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 the square cavity for $Pr = 0.71$ and $Ra = 10^3, 10^4, 10^5$.

	$Ra = 10^3$		$Ra = 10^4$		$Ra = 10^5$	
	u_{max}	v_{max}	u_{max}	v_{max}	u_{max}	v_{max}
PINN	0.137	0.138	0.192	0.233	0.128	0.258
Karakus (2022)	0.137	0.139	0.192	0.233	0.129	0.257
Stokos et al. (2015)	0.137	0.139	0.192	0.233	0.130	0.256
De Vahl Davis (1983)	0.136	0.138	0.192	0.234	0.153	0.261

After the training process, prediction is performed on a (251×251) 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 has 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 ratios of the residual loss over the boundary loss where ω_R represents the weight of the residual loss, and ω_{BC} represents the weight of loss on the boundary conditions. We stopped changing weights when we matched the center line velocities with the reference solutions. As the Ra increases, the convective effect 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 which minimizes the error both inside the domain and on the boundaries.

Table3. Maximum and minimum velocities along the centerlines with different weight ratio of residual loss and the boundary loss

ω_R/ω_{BC}	$Ra = 10^3$		$Ra = 10^4$		$Ra = 10^5$	
	u_{max}	v_{max}	u_{max}	v_{max}	u_{max}	v_{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 tested 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 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 built on plain fully connected networks converge in fewer iterations.

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.

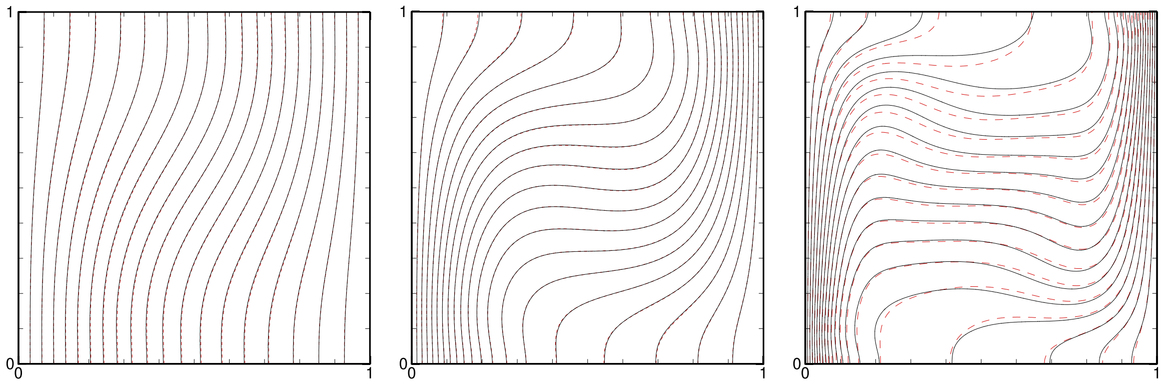


Figure 3: Temperature contours for the square cavity test. The high fidelity solution obtained with high fidelity discontinuous Galerkin solver between 1 and 0 with the increment of 0.05 for $Ra = 10^3, 10^4, 10^5$ from left to right shown with the black contours while the red contours are the solution with PINNs

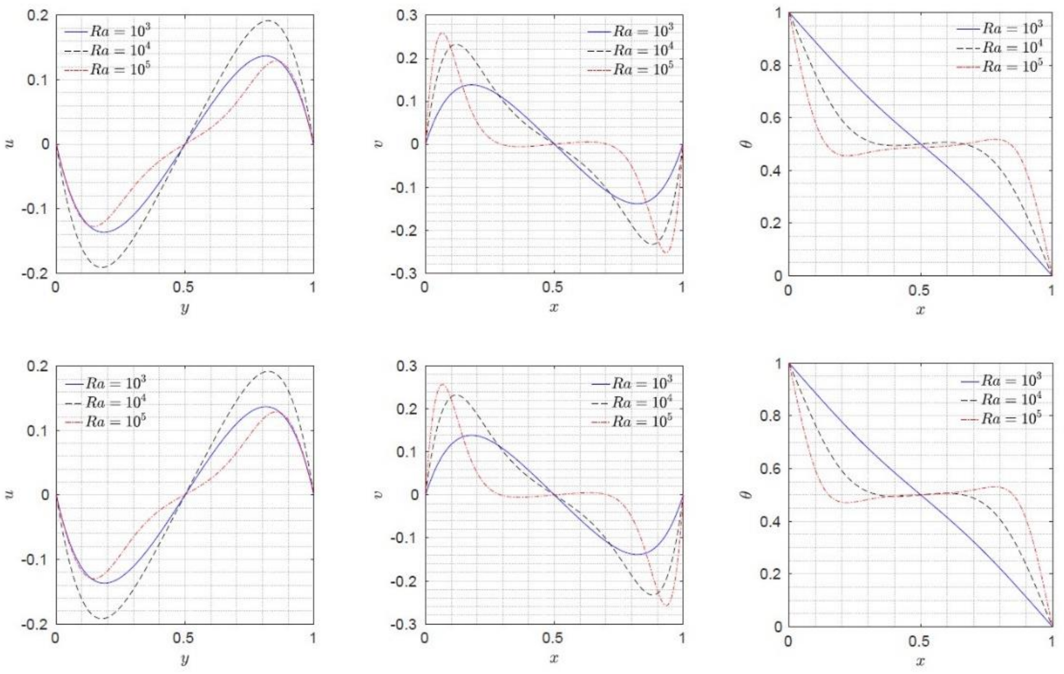


Figure 4. Velocity and temperature profiles along $y = 0.5$ and $x = 0.5$ lines for different Ra numbers. The first row shows the values obtained with the PINN, while the second row shows the values of high order discontinuous Galerkin solver.

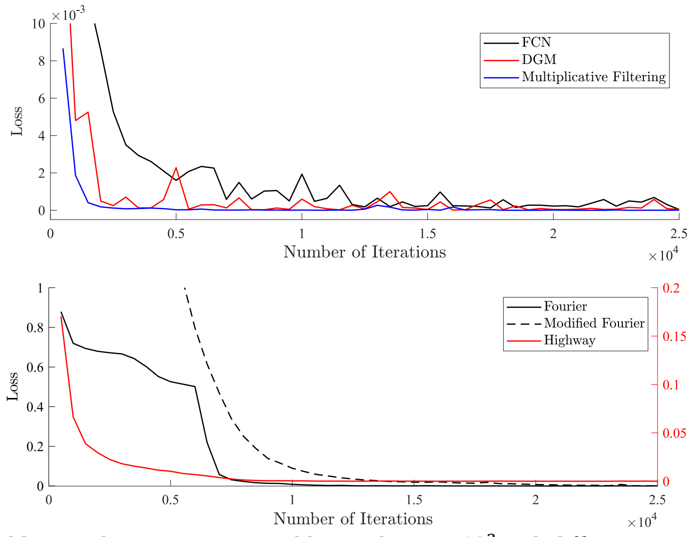


Figure 5. Behavior of the total loss on the square cavity problem with $Ra = 10^3$ with different types of neural network architectures

For training the network, we sampled 30 points on the inlet and the outlet, 210 points on the adiabatic wall, a total of 40 points on the heated block, a 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 5×10^{-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 8. The geometric parameters are given such that $H = 1, H/w = 2.5, L/w = 25, h/w = 0.5,$ as presented in Wu and Perng (1999).

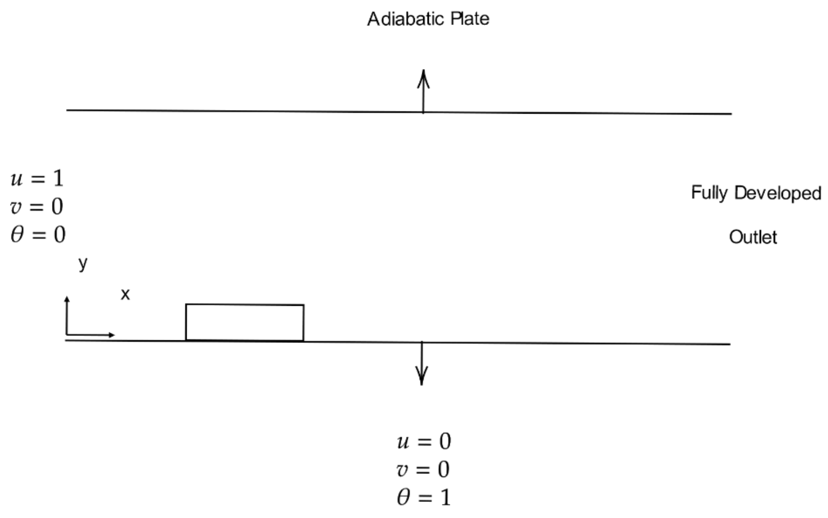


Figure 6. Schematic of the partially blocked channel

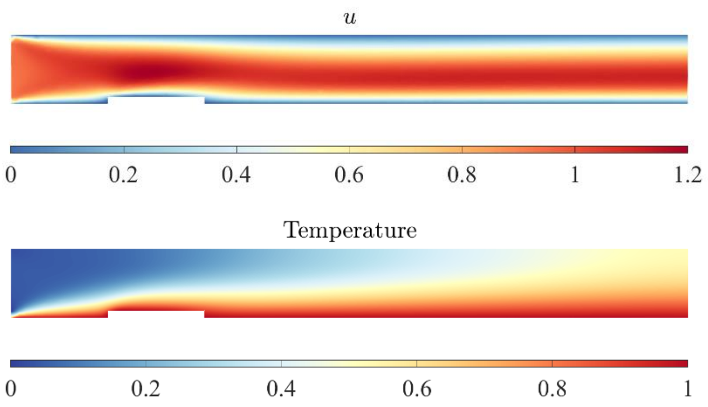


Figure 7. Velocity and temperature profiles for the heated block case. The figure on the top shows the velocity profile and the figure on the bottom shows the temperature field predicted by the PINN.

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 conditions of $\partial\theta/\partial n = -1$. Gravity is in the 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 are generated uniformly. 40 points on each block, 500 points on the top wall, and 450 points on the bottom wall are sampled. The problem is solved with a 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 5×10^{-4}. The solution at the final time $t = 8$s is shown in Figure 9. The horizontal velocity and the temperature fields are plotted, and it can be seen that the PINN solution represents the flowfield well physically inside the domain and also satisfy the Dirichlet and Neumann boundary conditions.

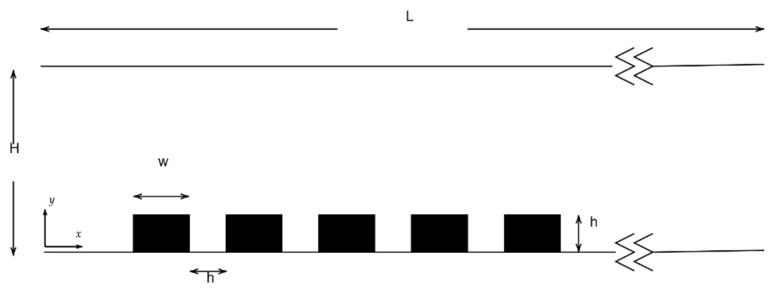


Figure 8. Geometric representation of the channel with multiple blocks

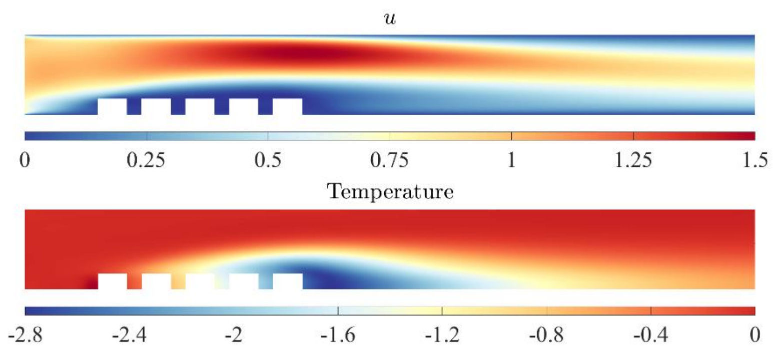


Figure 9. Velocity and temperature profiles for the multiple heated blocks case at time t = 8s. The figure on the top shows the velocity profile and the figure on the bottom shows the temperature field predicted by the PINN.

CONCLUSION

In this work, we solve two dimensional incompressible thermal convection problems with PINNs. The PINN solutions predict the flow and temperature fields well compared to the solution of high order solvers and analytical solutions 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. In addition, different types of networks can be used to solve thermal convection problems instead of fully connected networks. However, since these types of problems do not have multiscale behavior, it is not necessary to use Fourier mapping. Furthermore, we consider two different channel problems with a partial blockage that resemble power electronics applications. The model can be implemented into time dependent problems by adding time as a continuous input variable. For 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. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265‑283.

Bairi, A., Zarco-Pernia, E., and De Maria, 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*, 1-12.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., and Karniadakis, G., 2021, Physics-informed neural networks (PINNs) for heat transfer problems. *Journal of Heat Transfer*, 143(6).

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., Prabhat, M., Anandkumar, A., et al., 2020, Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 1‑15.

Fathony, R., Sahu, A. K., Willmott, D., and Kolter, J. Z., 2020, 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*, 249‑256.

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*, 447‑461.

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*, 42(1), 91-100.

Karakus, A., Chalmers, N., Swirydowicz, 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*, 3(6), 422-440.

Kingma, D. P. and Ba, J., 2017, Adam: A method for stochastic optimization. *arXiv:1412.6980.*

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*, 32.

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*, 5301-5310.

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*, 28.

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. *In 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:2003.04919*.

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., Lujan, 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*.