

Physics Informed Neural Networks(PINNs) for Burgers' equation

Indrajit Das

Department of Information Technology
Meghnad Saha Institute of Technology ,
Kolkata, India
indrajitdas1979@hotmail.com

Subhrapratim Nath, Member, IEEE
Department of Computer Science &
Engineering
Meghnad Saha Institute of Technology ,
Kolkata, India
suvro.n@gmail.com

Debjit Das

Department of Information Technology
Meghnad Saha Institute of Technology ,
Kolkata, India
line 5: debjitdas50@gmail.com

Manash Chanda, Senior Member, IEEE
Department of Electronics &
Communication Engineering
Meghnad Saha Institute of Technology ,
Kolkata, India
manash@msit.edu.in

Papiya Debnath, Member, IEEE

Department of Basic Science &
Humanities
Techno International New Town ,
Kolkata, India
papiya.debnath@tict.edu.in

Abstract—PINNs are neural networks that undergo training to execute supervised learning tasks while adhering to certain laws of physics, which are typically defined by basic nonlinear partial differential equations. This paper concentrates on the advancements in addressing burger equation problems using data-driven approaches for partial differential equations. Here the algorithms are separated into continuous time designs. The neural networks that emerge from this process are part of a new class of data-efficient approximators for universal functions, which automatically include all fundamental physical principles as prior information. Using these networks, this research demonstrates how to deduce solutions to partial differential equations.

Keywords— PINNs, Machine learning, Burger's Equation, PDEs Predictive modeling, Multilayer perceptron

I. INTRODUCTION

The increase in accessible information and computer capacity has resulted in substantial progress in machine learning and data analysis, leading to revolutionary achievements in diverse scientific domains such as Image identification [1], processing of natural language [2], neuroscience [3], and genetics [4]. However, frequently, when examining intricate physical, biological, or technical systems, the expense of gathering knowledge is too high, and to unavoidably confronted with the difficulty of reaching decisions and making judgments with incomplete information. When there is a scarcity of data, sophisticated machine learning mechanisms such as deep neural networks face challenges in maintaining stability and are unable to guarantee convergence.

At first, it may appear hard to train a deep learning technique to effectively map nonlinear connections from a small amount of data that could potentially have a high number of dimensions. However, there is a significant amount of existing knowledge in modelling physical and biological systems that has not been fully leveraged in contemporary machine learning methodologies. Prior knowledge, which may consist of basic physical principles, experimentally validated principles or specialised knowledge can regularise the resulting space to a reasonable

limit. Incorporating organised information into a learning system boosts data usefulness, enabling the algorithm to efficiently identify the correct solution and effectively apply it to new situations, even when there is not much training data available.

This paper proposes a novel solution i.e PINNs which are a mathematical approach to machine learning that is specifically designed to address problems related to Partial Differential Equations (PDEs). This is accomplished by constructing a neural network to minimise loss. The loss function considers the beginning and border conditions of the PDE, as well as the residual of the PDE at specific places i.e. domain collocation points. PINNs are deep learning networks that train to figure out a differential equation's response for an input point in the integration domain. Integrating a residual network that represents the governing physics equations is a notable innovation in the context of PINNs. The fundamental principle underlying PINN training is that it operates in the role of unsupervised method that doesn't employ labelled data like simulation or experiment results. The PINNs algorithm is a technique that analyses PDEs without relying on a mesh. It achieves this by converting the challenge of directly addressing the equations that regulate into an optimisation issue that involves a loss function.

Current studies [5, 6, 7] have shown that including structured prior understanding can lead to the development of learning models that are both data-efficient and guided by physics. Researchers have been using Gaussian process regression [8] to develop customized functional models for certain linear operators. Despite their elegance and flexibility, Gaussian processes encounter two main challenges when confronted with nonlinear problems when encoding prior information. Predictions in extremely nonlinear cases were less accurate at first because researchers [9,10] had to transform nonlinear components into linear form over time, limiting the approaches' utility to discrete-time domains. Additionally, the Gaussian process regression relies on Bayesian principles and necessitates specific prior assumptions. These assumptions might restrict the model's ability to describe complex patterns and may lead to robustness issues, especially when dealing with

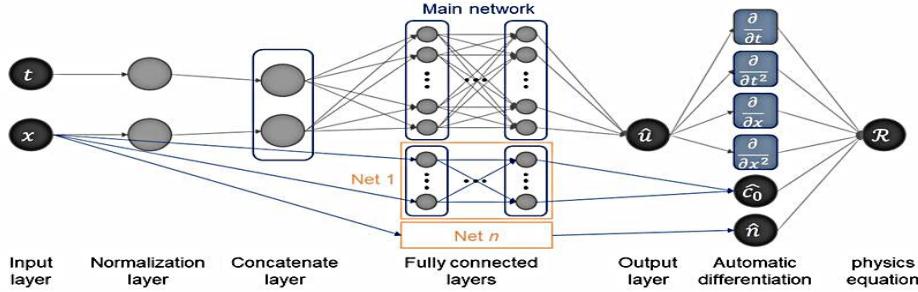


Fig.1 The physics-informed neural network is designed to accommodate non-uniform material scenarios

nonlinear situations [11]. In [12] a basic neural network to approximate a partial differential equation (PDE).

In this paper, a PINN model has been implemented which is shown in Figure 1. The initial layer is the input layer, which receives two signal characteristics: spatial and temporal. Instead of utilizing normalized input data, the architecture of the ANN incorporates a normalization layer. This lets the model get its primary information values during back-propagation. The normalisation layer calculates the initial information average value and standard deviation before training. The normalisation layer uses the estimated values to normalise the source data set to a mean of 0 and a standard deviation of 1. The concatenate layer then concatenates two single-dimensional scalar input information sets into a single vector. Normalised data enters the fully linked layer network. The main network consists of 9 hidden layers, each containing 20 neurons. A neural network approximation approach creates nonlinear PDE solution,

$$u_\theta(t, y) \approx u(t, y) \quad (1)$$

Where $u_\theta : [0, T] \times \square \rightarrow \square$ is a neural network function with parameters θ .

The continuous time technique for parabolic PDEs relies on the residual of a neural network approximation $u_\theta : [0, T] \times \square \rightarrow \square$ of the solution, i.e.,

$$s_\theta(t, y) := \delta u_\theta(t, y) + \square[u_\theta](t, y) \quad (2)$$

PINNs must differentiate the differential operators δu_θ and $\square[u_\theta]$ to include this PDE residual into a minimized loss function. The PINN term s_θ maintains the same parameters as the original network $u_\theta(t, y)$ while adhering to the "physics" of the nonlinear PDE. Automatic differentiation with modern machine learning tools like TensorFlow or PyTorch can determine both types of derivatives.

PINN solves the starting and boundary value problem by minimizing the loss function.

$$\phi(y) := \phi_b(y^s) + \phi_b(y^0) + \phi_b(y^b) \quad (3)$$

$$\phi_b := \frac{1}{N_s} \sum_{i=1}^{N_s} s_\theta(t_i^s, y_i^s)^2 \quad (4)$$

in a number of collocation points
 $y^s := \{(t_i^s, y_i^s)\}_{i=1}^{N_s} \subset [0, T] \times \square$, where s_θ is the physics-informed neural network,

- mean squared misfit to begin and boundary conditions

$$\begin{aligned} \phi_b(y^0) &:= \frac{1}{N_0} \sum_{i=1}^{N_0} |u_\theta(t_i^0, y_i^0) - u_0(y_i^0)|^2 & \text{and} \\ \phi_b(y^b) &:= \frac{1}{N_b} \sum_{i=1}^{N_b} |u_\theta(t_i^b, y_i^b) - u_b(y_i^b)|^2 \end{aligned} \quad (5)$$

in a numerous locations $y^0 := \{(t_i^0, y_i^0)\}_{i=1}^{N_0} \subset [0] \times \square$ and $y^b := \{(t_i^b, y_i^b)\}_{i=1}^{N_b} \subset [0, T] \times \partial \square$ where u_θ is the neural network approximation of the solution $u : [0, T] \times \square \rightarrow \square$.

II. PROPOSED METHODOLOGY

Figure 2 depicts the Multilayer Perceptron (MLP) consisting of 9 hidden layers, each containing 20 neurons. The neurons in these layers utilise a hyperbolic tangent activation function. There are two output variables, u and v , each corresponding to one of the variables in the issue. Following the Multilayer Perceptron (MLP), there is a boundary-encoded output layer. This layer ensures strict adherence to the Dirichlet border conditions. The layer operates by integrating the output of the MLP with a well-defined function that accurately satisfies the Dirichlet boundary condition. There are two separate loss functions, and their values are combined to calculate the overall loss. The first loss function is determined by the adherence to the governing equations, and its value is calculated as the sum of the average absolute value of the residual of each governing equation. The second form of loss is associated with the data-driven aspect of the training and is determined by calculating the average absolute distance between the projected values and the known solution values. This loss is contingent upon the prior understanding of the solution at specific locations within the domain.

A. Burger's equation

The Burgers' equation [13] is a fundamental partial differential equation. When the viscosity parameters are modest, the Burgers' equation can result in the creation of shocks, which is known to be challenging to accurately represent using traditional numerical approaches. The

Burgers' equation with Dirichlet boundary conditions in a single spatial dimension is expressed as:

$$u_s + u u_s - \left(\frac{0.01}{\pi}\right) u u_s = 0, y \in [-1, 1], s \in [0, 1] \quad (5)$$

$$u(0, y) = -\sin(y),$$

$$f := u + u u_y - \left(\frac{0.01}{\pi}\right) u u_s \quad (6)$$

The proposed algorithm is given below:

Step 1: Establish the problem description PDE (Burgers' equation): Import the necessary libraries, such as TensorFlow and NumPy, and set the type of information to float32. Next, create constants such as pi and viscosity, which is equal to 0.01 divided by pi.

Step 2: Specify the starting requirement as $f_0(y) = -\sin(\pi * y)$ and the boundary conditions as $f_b(t, y)$.

Step 3: Specify the residuals of the PDE $r(t, y, u, u_t, u_y, u_{yy})$, while t represents time, u_t denotes the partial derivative of u with regards to time, and u_y represents the partial derivative of u with regard to y. The symbol u_{yy} represents the second derivative of the function u concerning the variable y.

Step 4: Data Preparation

- Specify the quantity of data points and the spatial or temporal limits.
- Generate the starting condition f_0 , boundary condition f_b , and residual collocation points r
- Step 5:** Set up the network model. In this context, setting up a feed-forward neural network that consists of 9 hidden layers, with each layer containing 20 neurons.

Step 6: Calculate the residuals function called `get_r()` that calculates the residuals by utilizing automatic differentiation.

Step 7: Calculate the loss and gradients of the loss for implementation of backpropagation.

Step 8: Begin by initializing the model. Then, establish a fixed learning rate and select the Adam optimizer with the stated learning rate schedule.

Step 9: After completing the network establishment

Mean squared error loss given in equation (7)

$$MSE = MSE_u + MSE_f \quad (7)$$

Where

$$MSE_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |u(t_i^u, y_i^u) - u_i|^2$$

$$MSE_f = \frac{1}{N_u} \sum_{i=1}^{N_u} |f(t_i^f, y_i^f)|^2$$

Here, $\{t_i^u, y_i^u, u_i\}_{i=1}^{N_u}$ represent the initial and boundary training data for (t, y) , and $\{t_i^f, y_i^f\}_{i=1}^{N_f}$ specify the collocation points for $f(t, y)$. Loss factor MSE_u represents preliminary and boundary information. While MSE_f imposes a form at a finite set of collocation sites. When dealing with larger datasets, it is possible to use a more efficient method called a mini-batch setup with stochastic gradient descent and its current variations. While there is no theoretical assurance of reaching a global minimum, empirical data indicates that our method can achieve accurate predictions if the PDE has an established and distinct solution. This can be accomplished by using a neural network architecture that is expressive enough and has an appropriate number of collocation points N_f .

This discovery is directly connected to the optimization environment created by the MSE loss in equation (7). In this paper will evaluate the resilience of the suggested approach by conducting a series of methodical sensitivity analyses that are associated with the numerical findings provided below.

III. EXPERIMENTAL RESULT

The experiment is done Model training took approximately 9 minutes and 3 seconds on a single Nvidia T4 Tensor core GPU provided by Google Colab.

Figure 3 presents a concise overview of the outcomes obtained from the data-driven approach to solve the Burgers' A solution based on analysis has been found for this issue, and the consequent error in prediction has been calculated at 6.8×10^{-4} in the comparative L2-rule. Figure 5 demonstrates that the faults in this scenario are considerably reduced in comparison to the error documented in previous studies on data-driven solutions of PDEs utilising Gaussian techniques. Figure 6 presents a thorough assessment of the projected solution by comparing the PINNs and without PINNs solutions at various time intervals, namely $t = 0.0, 0.25$ and 0.5 . The PINN accurately models the intricate nonlinear dynamics of the Burgers' equation using a small set of initial and boundary data. As a consequence, a clearly defined interior layer is formed at $t = 0.4$. Solving this problem effectively using traditional numerical methods is known to be difficult and requires careful division of equation (3) into small parts in both space and time. To comprehensively assess the efficacy of this method, here conducted systematic enquiries to gauge its predicted accuracy across several circumstances, encompassing varying amounts of training and collocation data.

Table 1 and 2 display the relative L2 error achieved for various quantities of initial and boundary training data (N_u). The overall pattern indicates that the accuracy of predictions improves as the total amount of training data N_u grows, assuming there are enough collocation points N_f . The aforementioned observation emphasizes a fundamental advantage of physics-informed neural networks: their ability to incorporate the inherent physical structure, enabling them to attain accurate predictions even when working with limited datasets.

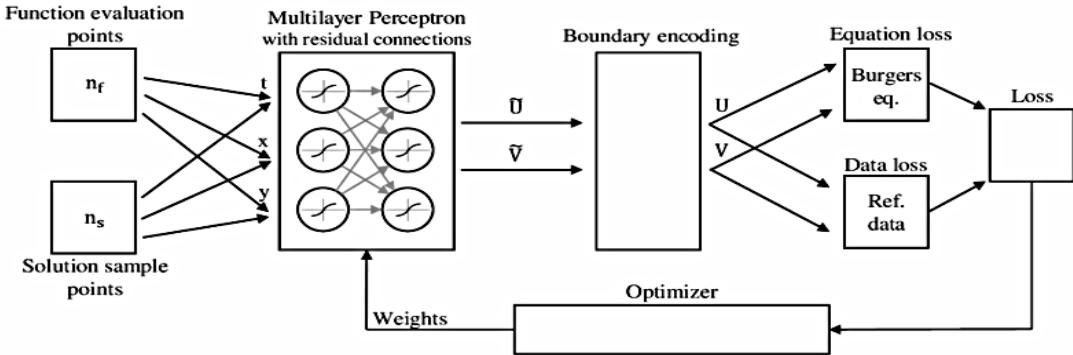


Fig.2 Proposed Methodology

Table I. Burgers' equation investigates the comparative L2 fault

N_f N_u	2000	4000	6000	7000	8000	10000
60	0.36	0.015	0.12	0.0054	0.0021	0.0081
80	0.0051	0.09	0.031	0.0075	0.049	0.0045
100	0.0062	0.025	0.0069	0.0007	0.0022	0.00065
200	0.13	0.0025	0.00085	0.0009	0.00065	0.00049

TableII. Burgers' equation is used to measure the relative L2 error

Layers Neurons	10	20	40
2	0.72	0.055	0.008
4	0.0027	0.00085	0.00057
6	0.0092	0.0007	0.00057
8	0.0017	0.00092	0.00051

IV. COMPARATIVE STUDY AND DISCUSSIONS

The brute force approach effectively resolves the problem, but it raises the question of the necessity of employing numerous advanced ideas such as deep learning and neural networks. Let's get comprehension by utilizing the graph. In this study, mathematical principles were utilized to standardize the partial differential equations (PDEs). Based on Figure 7, it is clear that the boundary conditions have been altered, indicating that there may have been an incorrect computation.

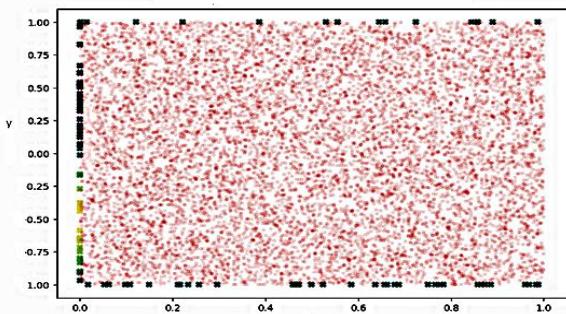


Fig. 3. Position of the collocation points and boundary data

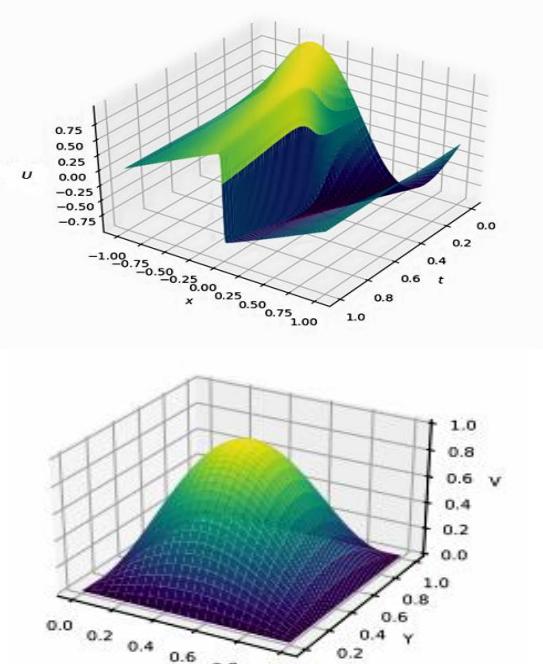


Fig.4 Initial condition for U and V

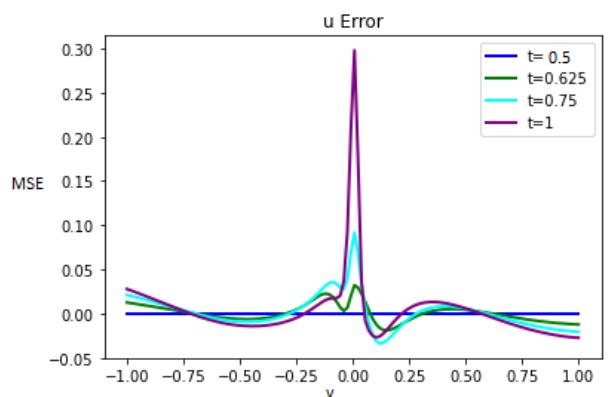


Fig.5 Plot of evaluation of loss

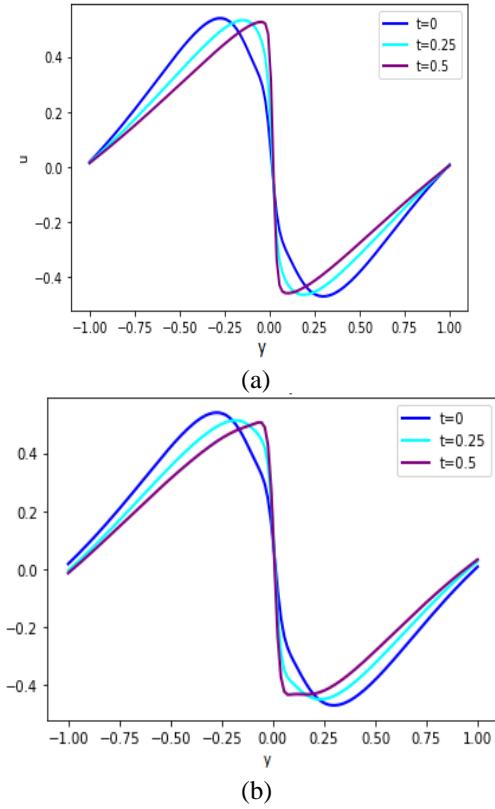


Fig.6 Comparison between the using without PINNs (a) and PINNs(b) outcomes for the three spatiotemporal samples for Burger's equation

Despite being initialized appropriately, the output could not be produced inside the range of $[-1, 0]$. There is a possibility of encountering an exception known as a NaN error. This occurs because some calculations are performed by the CPU, even if the values are truncated during the process of normalizing the PDEs. So, to manage these exceptions, in this paper PINNs algorithm is used and still obtains a small number of errors shown in Figure 8. This approach can be used in device modelling [14]-[17] also which will be highly efficient and time-efficient.

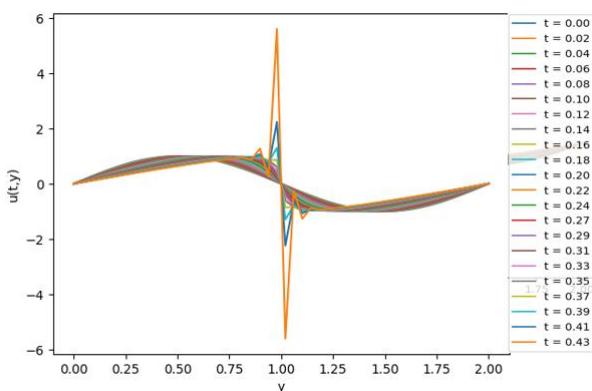


Fig.7 Solution of Burgers' equation (Normal)

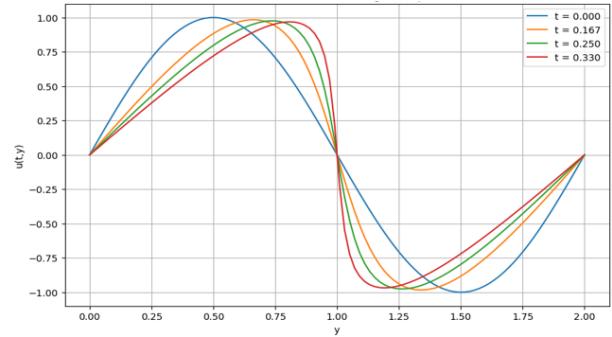


Fig.8 Solution of Burgers' equation (PINNs)

V. CONCLUSION

PINNs have been created as a novel type of universal function approximators. These networks can encode the fundamental physical rules that control a given dataset, which may be expressed by partial differential equations. This paper focuses on developing data-driven algorithms for example burgers' equations to deduce solutions for complex nonlinear PDEs and create computationally efficient surrogate models that incorporate physical principles. The resulting methodologies demonstrate a range of promising outcomes for many challenges in computational science and pave the way for enhancing a vast array of scientific fields. These benefits can be conveniently enjoyed in specific applications such as data-driven forecasting of physical processes, model predictive control, and multi-physics/multi-scale modelling and simulation. It is important to acknowledge that the suggested approaches should not be seen as substitutes for traditional numerical methods used to solve partial differential equations, such as finite elements or spectral methods. These methods have developed and improved over the past 50 years and, in many instances, they meet the practical criterion for being both strong and computationally efficient.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105..
- [2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
- [3] B. M. Lake, R. Salakhutdinov, J. B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (2015) 1332–1338.
- [4] B. Alipanahi, A. Delong, M. T. Weirauch, B. J. Frey, Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning, Nature biotechnology 33 (2015) 831–838.
- [5] M. Raissi, P. Perdikaris, G. E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, Journal of Computational Physics 335 (2017) 736–746.
- [6] M. Raissi, P. Perdikaris, G. E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, Journal of Computational Physics 348 (2017) 683 – 693.
- [7] H. Owhadi, Bayesian numerical homogenization, Multiscale Modeling & Simulation 13 (2015) 812–828.
- [8] C. E. Rasmussen, C. K. Williams, Gaussian processes for machine learning, volume 1, MIT press Cambridge, 2006.
- [9] M. Raissi, P. Perdikaris, G. E. Karniadakis, Numerical Gaussian processes for time-dependent and non-linear partial differential equations, arXiv preprint arXiv:1703.10230 (2017).

- [10] M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, arXiv preprint arXiv:1708.00588 (2017).
- [11] H. Owhadi, C. Scovel, T. Sullivan, et al., Brittleness of Bayesian inference under finite information in a continuous world, *Electronic Journal of Statistics* 9 (2015) 1–79.
- [12] Dissanayake, M.W.M.G., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Methods Eng.* 10(3), 195–201 (1994).
- [13] C. Basdevant, M. Deville, P. Haldenwang, J. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, A. Patera, Spectral and finite difference solutions of the Burgers equation, *Computers & fluids* 14 (1986) 23–41.
- [14] Roy, S., Jana, G. & Chanda, M. Analysis of Sub-Threshold Adiabatic Logic Model Using Junctionless MOSFET for Low Power Application. *Silicon* 14, 903–911 (2022). <https://doi.org/10.1007/s12633-020-00870-y>.
- [15] Basak, A., Chanda, M. & Sarkar, A. Drain current modelling of unipolar junction dual material double-gate MOSFET (UJDMDG) for SoC applications. *Microsyst Technol* 27, 3995–4005 (2021). <https://doi.org/10.1007/s00542-019-04691-x>
- [16] T. Ganguli, M. Chanda, A. Sarkar, "Impact of Interface Trap Charges on the Performances of Junctionless MOSFET in Sub-Threshold Regime," *Computers and Electrical Engineering*, Vol. 100, 2022, 107914, <https://doi.org/10.1016/j.compeleceng.2022.107914>.
- [17] A. Mukherjee, P. Debnath, D. Nirmal, M. Chanda, A new analytical modelling of 10 nm negative capacitance-double gate TFET with improved cross talk and miller effects in digital circuit applications, *Microelectronics Journal*, vol. 133, 2023, 105689, <https://doi.org/10.1016/j.mejo.2023.105689>.
- [18]