Data-driven Methods for Partial Differential Equations

Caleb Chronowski, Olivia Holdeman, Josephine Parker, Collin Sunes, Lola Torres
Faculty Mentor: Dr. M. Berezovski
Industrial Partner: Argonne National Laboratory
MA 490/390
Embry-Riddle Department of Mathematics

Abstract

Exploring recent strides in the realm of Partial Differential Equations (PDEs) solutions, this paper provides insight on the integration of data-driven methods to address challenges in traditional numerical techniques. PDEs, foundational for modeling diverse physical phenomena, often encounter computational limitations when dealing with complex systems. Machine learning and numerical analysis has ushered in innovative data-driven solutions, marking a paradigm shift in computational science. The discussion unfolds with an examination of neural networks, deep learning architectures, and other machine learning techniques in the context of PDEs. By combining the interpretability of classical techniques with the efficiency of data-driven models, these strategies adeptly navigate intricate and high-dimensional input spaces. In conclusion, this review provides a nuanced perspective on the current landscape of data-driven methods for PDEs. The scalable, adaptive, and accurate solutions offered by data-driven methodologies are expected to propel advancements in simulation, optimization, and decision-making.

Introduction

Partial differential equations (PDEs) serve as fundamental tools for modeling physical phenomena across many scientific disciplines. Traditional numerical methods are effective in solving PDEs, however, they encounter limitations when it comes to handling complex geometries and high-dimensional systems. To combat these challenges, Physics Informed Neural Networks (PINNs) are presented as an innovative approach that seamlessly combines domain knowledge with the power of neural networks. This project aims to leverage PINNs to solve a given PDE, where the system is characterized by a set of equations involving multiple variables. The PINN architecture is tailored for each variable of interest and integrates physics-based constraints. The loss function used in PINNs is created to capture multiple aspects including terms for matching synthetic data to true solutions and adherence to underlying physics. In this way, the network learns meaningful solutions.

Research Question

Using a two-dimensional elliptic heat transfer equation with Neumann boundary conditions, data-driven methods can be developed to solve the given problem. By implementing Fourier Neural Operators and Physics Informed Neural Nets, networks are created to approximate mathematical operators. Neural operators leverage neural networks to learn and approximate these operators, allowing for more flexible and data-driven approaches to solving mathematical problems. Traditional numerical methods can be applied to reach a solution. Implementing simple discretization schemes such as the Finite Difference Method will ultimately provide a solution. Additionally, data is generated for w which will in turn allow for a Neural Network to be trained and then further tested for accuracy. By using all of the methods previously mentioned, an optimal solution is reached.

Methodology

The equations provided in the project description from Argonne National Laboratory are provided below. This represents a 2D elliptic heat equation, Gaussian blob, and boundary conditions.

$$-\nabla \cdot (e^u \nabla w) = f(x,y) \text{ in } \Omega$$

$$w = 0 \text{ on } \delta\Omega$$

$$f(x,y) = 1$$

$$x \in [0,1], y \in [0,1]$$

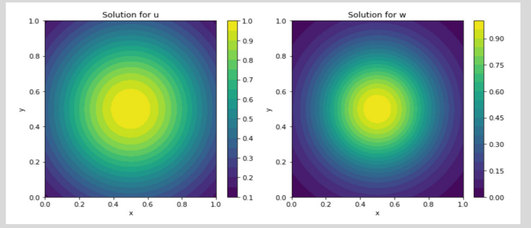
$$u = e^{(-4\times(x-0.5)^2-4\times(y-0.5)^2)}$$

- Grid Discretization (FDM): The spatial domain is discretized into a grid using Finite Difference Methods (FDM), dividing the computational domain into smaller segments to approximate derivatives and differential equations.
- Initial Guess: The process starts with initial guesses for the solutions *u* and *w*, often based on boundary conditions or known initial states.
- Laplacian Computation (FDM): FDM is employed to compute the Laplacian of *u* and *w*, crucial for understanding how these quantities change over space.
- Poisson Equation: The Poisson equation, a partial differential equation describing the distribution of *w*, is solved to understand the underlying physics of the system and its behavior.
- Physics-Informed Neural Networks (PINNs): Neural networks are utilized to predict the solutions *u* and *w* across the spatial domain. These networks learn the underlying physics of the system and provide accurate predictions.
- Training: The PINN models are trained iteratively using a combination of FDM updates and data-driven approaches. This training process refines the neural network predictions to better match the true solutions.
- Loss Minimization: During training, a loss function is defined to quantify the error between the predicted and true solutions. The training process aims to minimize this loss, ensuring accurate predictions.
- Convergence Check: The training process iterates until a convergence criterion is met or until a maximum number of iterations is reached. This ensures that the model has sufficiently learned the underlying physics and produces accurate predictions.
- Solution Estimation: Once training is complete, the final solutions for *u* and *w* are estimated. These solutions satisfy the governing equations and any imposed boundary conditions, providing valuable insights into the system's behavior.

Results

FDM Results

The output of the FDM code provides two plots, one for u and one for w. The left subplot in the figure represents the solution for the variable *u*. The color contours on the plot indicate the values of u across the spatial domain. In the given problem, *u* is set as a Gaussian-like distribution centered at (0.5, 0.5). The contour plot visualizes how this initial distribution evolves according to the specified differential equation and boundary conditions. The right subplot in the figure represents the solution for the variable *w*. The values of *w* are computed using the combined approach of FDM, Neumann boundary conditions, and the neural network-based PINN. The specific relationship between *u* and *w* is given by the relationship of the equations as defined in the defined function in the PINN file.



PINN Results

- Accuracy: Assessing how well the PINN predicts solutions compared to known or expected results.
- Convergence: Monitoring the loss function during training to ensure stable convergence.
- Generalization: Evaluating the PINN's performance on unseen data to gauge its ability to generalize.
- Interpretability: Understanding how the learned parameters relate to physical quantities in the problem domain.
- Efficiency: Considering the computational resources required for training and scalability to larger problem domains.
- Comparison: Validating the PINN's results by comparing them with traditional numerical methods or analytical solution

Loss Function

- Performance Evaluation:
  - Indicates the level of error between PINN predictions and ground truth data.
  - Lower values suggest closer alignment between predictions and actual data.
- Model Optimization:
  - Loss is minimized during training by adjusting model parameters.
  - Optimization process aims to improve model's representation of underlying physics.
- Convergence:
  - Trend of loss value indicates convergence behavior during training.
  - Decreasing loss implies model is learning and improving performance.
  - Stagnation or increase may signal issues like overfitting or insufficient training.
- Interpretation:
  - Loss value of 0.04 suggests reasonably good performance.
  - Further analysis and fine-tuning may be needed for better results.

Conclusion

The PINN architecture involves separate neural networks for different variables in the PDE, one for *u* and another for *w*. Each neural network is designed to take spatial coordinates as input and output the corresponding solution variable. The loss function in PINNs is a combination of terms that enforce constraints from initial and boundary conditions, as well as the PDE itself. The loss function includes terms for the square of the difference between predicted and true solutions, ensuring the model fits the given data. Synthetic data is generated for training PINNs, combining known initial/boundary conditions and randomly sampled points in the spatial domain. This synthetic data is crucial for training the neural network to approximate the solution across the entire domain, especially in areas not covered by explicit data. The training process involves iterative updating of the neural network weights to minimize the PINN loss. Stochastic optimization techniques, such as Adam optimizer, are commonly used for training. Debugging PINNs can be challenging due to the combination of neural networks, partial differential equations, and training complexities. Common issues include handling TensorFlow eager execution, persistent GradientTape, and ensuring correct network architecture. The results, visualized through contour plots for *u* and *w*, illustrate the PINN's ability to approximate the solutions of the given PDE. Interpretation of the results involves understanding the meaning of the variables, the impact of boundary conditions, and the significance of the PDE constraints. Further refinement of the PINN model, tuning hyperparameters, and experimenting with different neural network architectures can lead to improved accuracy. Extending this approach to more complex PDEs or real-world applications would be a logical progression for future study. Ultimately, the findings highlight the effectiveness of PINNs in solving PDEs, the importance of synthetic data for training, and the need for careful debugging and interpretation of results. This methodology provides a powerful tool for solving complex physical systems with data-driven approaches.

References

1. Hippylib. (n.d.). Poisson Deterministic. Hippylib Tutorials. Retrieved from [https://hippylib.github.io/tutorials_v2.3.0/2_PoissonDeterministic/2_PoissonDeterministic/] (https://hippylib.github.io/tutorials_v2.3.0/2_PoissonDeterministic/2_PoissonDeterministic/)
2. IEEE Xplore. (n.d.). Finite Volume Method for Poisson Equation. Retrieved from [https://ieeexplore.ieee.org/document/9945171] (https://ieeexplore.ieee.org/document/9945171)
3. University of California, Irvine. (n.d.). Finite Difference Method for Poisson Equation. Retrieved from [https://www.math.uci.edu/~chenlong/226/FDM.pdf] (https://www.math.uci.edu/~chenlong/226/FDM.pdf)
4. University College Dublin. (n.d.). Poisson Equation with Zero Boundary Conditions. Retrieved from https://john-s-butler-dit.github.io/NumericalAnalysisBook/Chapter%2009%20-%20Elliptic%20Equations/902_Poisson%20Equation-Zero%20Boundary%20Conditions.html