Python-based software for solving clever PDEs

Basim K. Abbas[1], Nadia Mahmood Hussien[2], Yasmin Makki Mohialden[3], kawakib Mahmood Hussien[4]

[1,2,3]Computer Science Department, Collage of Science, Mustansiriyah University, Baghdad-Iraq

[4] College of Education Ibn Rushd ,University of Baghdad,Baghdad-Iraq

baasim_math@uomustansiriyah.edu.iq [1]

nadia.cs89@uomustansiriyah.edu.iq[2]

ymmiraq2009@uomustansiriyah.edu.iq[3]

[4]kawakib.mahmood@ircoedu.uobaghdad.edu.iq

Abstract

Several recent investigations have identified a technique for approximating solutions to partial differential equations (PDEs). However, there was little room for adaptive frameworks that facilitate the exploration of new concepts. We can compensate by using the PyDEns library in Python. Using the PyDEns module and the open-source Batch Flow framework, you can solve a variety of partial differential equations. There are partial differential equations, such as those that describe heat and waves, and other equations, such as this one. In this article, we explain how to solve differential equations using neural networks using a new method introduced by Python PyDEns. Partial differential equations can be solved with this software tool.

Keywords: Python**,** PyDEns-module, partial differential equations, heat equation, wave equation.

1. Introduction

Physical processes are modelled using differential equations, which are employed in a wide range of scientific and technical domains. The vast majority of differential equations that are encountered in real-world settings cannot be solved in an analytical manner. Number-crunching techniques are traditionally employed in the solution of differential equations. A partial differential equation (PDE) is a mathematical formula that describes how partial derivatives of a multivariable function are related. The algorithms for solving differential equations in time and space are quite complex. In algebraic equations such as this, x is an unsolved unknown that must be solved. However, due to the rarity of accurate formulas for partial differential equation solutions, this is typically the case in practice. Modern mathematics and science, of course, are concerned with numerically approximating partial differential equation solutions with the aid of computers. PDEs are significant in pure mathematics studies, and one of the most frequently asked topics is how to identify broad qualitative characteristics of solutions to partial differential equations (PDEs). They are in charge of numerically solving the linearized set of algebraic equations that are formed when partial differential equations are discretized.

Many scientists, however, require something more [2,3,4,5]. This paper demonstrates how programming approaches can be used to teach mathematics to students. It is tough to design an integration that is simple to understand and troubleshoot. Python, in contrast to C++ or Java, is a programming language that is easy to learn and practice [8]. Python has a number of mathematical models, including Math, Numpy, scipy, PyPDE, and Pydens, among others. PyPDEA is a Python module for solving hyperbolic or parabolic partial differential equations (PDEs). In other instances, the source terms are extremely strict. PyDEns is a framework for solving ordinary differential equations (ODEs) and partial differential equations (PDEs) with neural networks. Using Python frameworks, this paper provides a method for solving partial differential equations that has been proved to be effective. Each of the six sections of the study has an introduction, a review of relevant work, a proposed technique, and a conclusion. This is how it works:

2. Related work

There are many related works:-

[Koryagin, Khudorozkov., Tsimfer, 2019] In this study, the authors provide a PYDENS PYTHON module that solves partial differential equations using neural networks. Through GITHUB, the deep learning community can make use of and develop the module. We aim to make it more convenient for people to experiment in this new area. This is supported by a number of pillars. The first need is that a user should be able to tackle complex problems with only a few lines of code. This approach makes use of PYDENS, which allows you to generate PDEs from a variety of classes, such as heat equations and wave equations. To begin with, there should be no restrictions on the type of neural network design that can be used. For the purpose of creating a complex neural network from the ground up, convolutions and fully-connected layers in PYDENS can be utilized. Finally, the point-sampling scheme is controlled by the user. Trainable data batches can be generated by combining various probabilistic distributions, such as truncated Gaussian, uniform, and exponential distributions, or even combinations of these distributions, to produce a single set of data. [6].

[Nascimento, Fricke, Viana ,2020]They describe how to use recurrent neural networks to integrate ordinary differential equations in Python. To make the implementation easier, they employed TensorFlow and Keras. These systems easily use multilayer perceptrons and recurrent neural networks. Our method allows users to mix physics-based and data-driven kernels. In other words, Data-driven kernels can close the observation-prediction gap. A model's parameters can be set by them. They used two examples to demonstrate our process. The first used Euler's forward strategy. It combines data-driven stress intensity ranges with physics-based fracture length increment models to represent fatigue crack growth. A moving two-degree-of-freedom system was modeled using Runge–Kutta integration [1]

[Samaniego, Anitescu, Goswami, Nguyen-Thanh, Guo, Hamdia, Zhuang, Rabczuk,2020] Deep Neural Networks were considered as a feasible approximation method, which they investigated further (DNNs). They've done very well in some areas, such as visual recognition, for instance. In this case, deep neural networks (DNNs) are employed to approximate functions. Because of their structural flexibility and speed in implementation techniques, DNNs are a particularly attractive alternative to approximating the solution of a partial differential equation. They are mostly concerned with the applications of computational mechanics. The majority of contributors are interested in exploring this possibility through partnership. They are primarily concerned with mechanical difficulties and the energy format of the PDE. The energy of a mechanical system appears to be the most obvious loss function for a machine learning methodology to consider. A lot of problems have to be solved in order to figure out how the method works and how it can be used in engineering [7].

3. Suggested Method

In order to come up with a solution, an equation was applied in this study: Using neural networks to solve ODEs and PDEs, PyDEns is a program that may be used to answer a variety of mathematical problems. They can also solve PDEs and ODEs with trainable coefficients, such as the heat equation, the poisson equation, and the wave equation, among other types of equations and problems. The solution of the Poisson equation

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=5\sin[\pi(x+y)],\quad (x,y)\in[0,1]^2,u(0,y)=u(x,0)=u(1,y)=u(x,1)=1$$

It made use of a feed-forward neural network, which was triggered by Tanh's input. In order to solve the equation, you must first add a grammar of tokens to the namespace where you are currently located. Tokens are phrases that are used to write down differential equations.

a. Using a PyDEns-model to solve a problem

b. Run the optimization technique and get a mesh-free approximation of the solution on [0, 1]. Figure 1 displays the approximate Poisson solution.

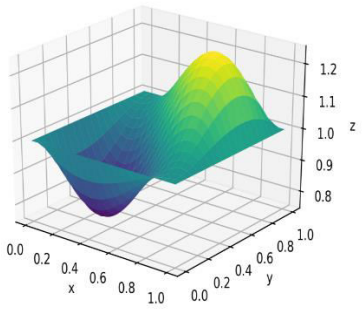


Fig. 1: approximation solution of Poisson equation

1- Solving parametric families of PDEs

PyDEns are considerably more than simple differential equation solvers. Additionally, it can deal with both I-parametric families of PDEs and (ii) trainable PDEs. Consider a collection of ordinary differential equations.

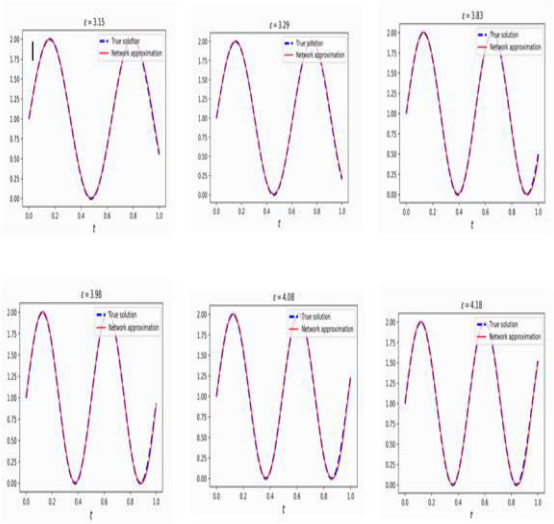
$$\frac{du}{dt} = \epsilon \cos[\epsilon \pi t]; \quad t \in [0,1], \quad u(0) = 1$$

Clearly, the solution is a sin wave with a phase parameterized by ϵ:

$$\sin[\epsilon \pi t] + 1$$

Other partial differential equations (PDEs) can be solved in the same way that this one has been solved. There is only one thing that needs to be done on your part. The following images demonstrate how neural network approximation works.

Fig. 2: Neural network approximation



2- Solving PDEs with trainable coefficients

If the system's initial state is unknown:

$$\frac{du}{dt} = 2\pi \cos[2\pi t]; \quad t \in [0,1] \quad ,u(0) = \omega$$

if at some point, the state of the system is different from how it is now

$$u(t = 0.5) = 2$$

A more sophisticated approach is required in this circumstance. It is used to indicate that there is a trainable variable in the issue's first state by using the V-token. In the configuration, there are two stages of training: one for refining the solution to the problem and another for satisfying an additional rule, both of which are required. Figure 3 depicts the outcome of resolving the problem and altering the original circumstances in order to comply with the new regulation. There are now two distinct positions available.

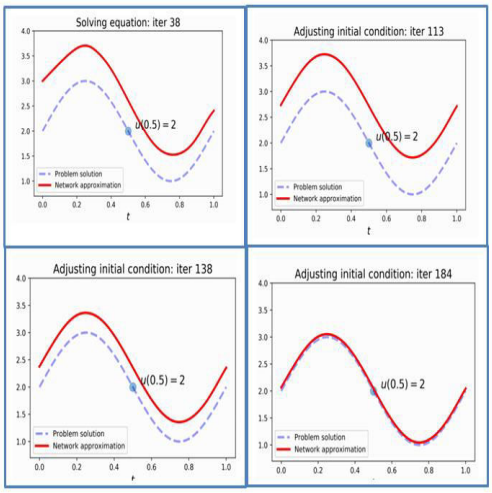


Fig. 3: Solving PDEs results.

4. Conclusion

This paper suggests a novel method of developing websites that makes use of the Python framework. The method is described in detail. An alternative solution to the problem was proposed in this paper. It is possible to solve a wide range of partial differential equations problems (a neural network) using the PyDEns module, which combines machine learning and partial differential equations problems.The Poisson equation, partial differential equations with trainable coefficients, and parametric families of partial differential equations are all solved in this study, as are all of the other equations.

References

[1] R.G. Nascimento, K. Fricke, F.A.C. Viana, A tutorial on solving ordinary differential equations using Python and hybrid physics-informed neural networks, Eng. Appl. Artif. Intell. 96 (2020) 103996, https://doi.org/10.1016/j.engappai.2020.103996.

[2] HADAMARD, Jacques. Lectures on Cauchy's problem in linear partial differential equations. Courier Corporation, 2003.

[3] GUYER, Jonathan E.; WHEELER, Daniel; WARREN, James A. FiPy: Partial differential equations with Python. Computing in Science & Engineering, 2009, 11.3: 6-15.

[4] Chen, Feiyu, et al. "NeuroDiffEq: A Python package for solving differential equations with neural networks." Journal of Open Source Software 5.46 (2020): 1931.

[5] FLUNKERT, Valentin; SCHOELL, Eckehard. Pydelay-a python tool for solving delay differential equations. arXiv preprint arXiv:0911.1633, 2009.

[6] Koryagin, A., Khudorozkov, R., & Tsimfer, S,." PyDEns: A python framework for solving differential equations with neural networks,". arXiv preprint arXiv:1909.11544,2019.

[7] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, Implementation and Applications, Computer Methods in Applied Mechanics and Engineering 362 (2020) 112790.

[8] ALANI, Lamya A. Omer. Efficiency of Engaging Python in Teaching Discrete Mathematics: Case Study. Al-Nahrain Journal of Science, 2016, 19.3: 131-137.