Project 1, Partial differential equations with Neural Networks

FYS5429/9429, Advanced machine learning and data analysis for the physical sciences, University of Oslo, Norway

Spring semester 2025, deadline March 21

Solving partial differential equations with neural networks

This variant of project 1 is tailored to those of you who are interested in studying differential equations and may have followed popular courses on these methods. **It can also be seen as a stepping stone towards studies of PINNs, laying thereby the basis for project 2**.

For this variant of project 1, we will assume that you have some background in the solution of partial differential equations using finite difference schemes. If you are not familiar with these methods, we can give you an introduction.

We will study the solution of the diffusion equation in one dimension using a standard explicit scheme and neural networks to solve the same equations. Feel free to add more advanced finite difference or finite element methods.

For the explicit scheme, you can study for example chapter 10 of the lecture notes in Computational Physics, FYS3150/4150 or alternative sources from courses like MAT-MEK4270. For the solution of ordinary and partial differential equations using neural networks, the lectures by of week 43 at for example `https://compphysics.github.io/MachineLearning/doc/pub/week42/html/week43.html` at this course are highly recommended.

For the machine learning part you can use your own codes or the functionality of for example **Tensorflow/Keras**, **PyTorch** or other libraries such as Physics informed machine learning.

Alternative differential equations. Note that you can replace the one-dimensional diffusion equation discussed below with other sets of either ordinary differential equations or partial differential equations. A typical equation many of you may be interested in is for example the Navier-Stokes equation.

An alternative is a stochastic diffusion equation, known as the Black-Scholes equation (Nobel prize in economy, see `https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model`).

An interesting article on PINNs with the Black-Scholes equation could serve as a possible path for the second project, see https://arxiv.org/abs/2312.06711.

Part a), setting up the problem. The physical problem can be that of the temperature gradient in a rod of length $L=1$ at $x=0$ and $x=1$. We are looking at a one-dimensional problem

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, t > 0, x \in [0,L]$$

or

$$u_{xx} = u_t,$$

with initial conditions, i.e., the conditions at $t=0$,

$$u(x,0) = \sin(\pi x) \qquad 0 < x < L,$$

with $L=1$ the length of the x-region of interest. The boundary conditions are

$$u(0,t) = 0 \qquad t \geq 0,$$

and

$$u(L,t) = 0 \qquad t \geq 0.$$

The function $u(x,t)$ can be the temperature gradient of a rod. As time increases, the velocity approaches a linear variation with x.

We will limit ourselves to the so-called explicit forward Euler algorithm with discretized versions of time given by a forward formula and a centered difference in space resulting in

$$u_t \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t) - u(x_i,t_j)}{\Delta t}$$

and

$$u_{xx} \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2},$$

or

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j) - 2u(x_i,t_j) + u(x_i-\Delta x,t_j)}{\Delta x^2}.$$

Write down the algorithm and the equations you need to implement. Find also the analytical solution to the problem.

Part b). Implement the explicit scheme algorithm and perform tests of the solution for $\Delta x = 1/10$, $\Delta x = 1/100$ using Δt as dictated by the stability limit of the explicit scheme. The stability criterion for the explicit scheme requires that $\Delta t/\Delta x^2 \leq 1/2$.

Study the solutions at two time points t_1 and t_2 where $u(x,t_1)$ is smooth but still significantly curved and $u(x,t_2)$ is almost linear, close to the stationary state.

Part c) Neural networks. Study now the lecture notes on solving ODEs and PDEs with neural network and use either your own code from project 2 or the functionality of tensorflow/keras to solve the same equation as in part b). Discuss your results and compare them with the standard explicit scheme. Include also the analytical solution and compare with that.

Part d) Neural network complexity. Here we study the stability of the results of the results as functions of the number of hidden nodes, layers and activation functions for the hidden layers. Increase the number of hidden nodes and layers in order to see if this improves your results. Try also different activation functions for the hidden layers, such as the **tanh**, **ReLU**, and other activation functions. Discuss your results.

Part e). Finally, present a critical assessment of the methods you have studied and discuss the potential for the solving differential equations with machine learning methods.

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.

- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Send us an email in order to hand in your projects with a link to your GitHub/Gitlab repository.
- In your GitHub/GitLab or similar repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parameters.

Finally, we encourage you to collaborate. Optimal working groups consist of 2-3 students. You can then hand in a common report.