Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Comparison of FDM, FVM with NN for solving the Forward Problem

Master's Final Degree Project

Pijus Makauskas

Project author

A. Prof. Mayur Pal

Supervisor

Kaunas, 2022

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Comparison of FDM, FVM with NN for solving the Forward Problem

Master's Final Degree Project

Applied Mathematics (6211AX006)

Pijus Makauskas

Project author

A. Prof. Mayur Pal

Supervisor

Prof. habil. dr. Rimantas Barauskas

Reviewer

Kaunas, 2022

Kaunas University of Technology

Faculty of Mathematics and Natural Sciences

Pijus Makauskas

Comparison of FDM, FVM with NN for solving the Forward Problem

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Pijus Makauskas

Confirmed electronically

Makauskas, Pijus. Comparison of FDM, FVM with NN for solving the Forward Problem / supervisor A. Prof. Mayur Pal; Faculty of Mathematics and Natural Sciences, Kaunas University of Technology.

Study field and area (study field group): Applied Mathematics (Mathematical Sciences).

Keywords: Finite Difference Method, Finite Volume Method, Artificial Neural Networks, Poisson‘s equation, heterogeneous domain.

Kaunas, 2021. 55 p.

Summary

This thesis focuses on a problem of determining field distribution of a particular measure *u* throughout a highly heterogeneous and layered permeability domain assuming some driving function. An elliptic Partial Differential Equation, used to describe such distribution at an equilibrium state, is called Poisson's Equation which originates from Darcy's Law. The aforementioned equation describes diffusion and has no temporal dimension, thus requiring sufficient boundary conditions to be uniquely solved using numerical methods. After performing a comparison study of two such methods known as Finite Difference Method (FDM) and Finite Volume Method (FVM), a proposition of an alternative approach of using Artificial Neural Network (ANN) to solve a given problem was made. The ANN is referred to as a Coordinate Discretized Neural Network (CDNN) throughout the thesis.

Both numerical methods were implemented on a two-dimensional grid and compared with each other on highly heterogeneous and layered permeability domains. Since FVM is is both locally and globally conservative method it outperformed FDM in several ways. The latter created non-monotonous solutions with sharp transitions in between the sub-domains and overall drew unrealistic maps of *u* that were excessively impacted by the underlying permeability domain.

Conclusions were drawn from these observations to train CDNN solely on FVM's generated data, keeping in mind how a proper method should behave. CDNN produced accurate results, managing to replicate the FVM, even on some complex permeability distributions. However, high noise contamination present in the results leaves room for further network architecture improvement and there might be even possibilities to reformulate the modelling approach to improve the results. Suggestions for further development and possible network enhancements, were discussed in greater detail in the concluding discussion chapter.

Makauskas Pijus. Baigtinių Skirtumų, Baigtinių Tūrių bei Dirbtinių Neuroninių Tinklų metodų palyginimas sprendžiant Tiesioginį Uždavinį. Magistro studijų baigiamasis projektas / vadovas doc. dr. Mayur Pal; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Taikomoji matematika (Matematikos mokslai).

Reikšminiai žodžiai: Baigtinių Skirtumų Metodas, Baigtinių Tūrių Metodas, Dirbtiniai Neuroniniai Tinklai, Puasono lygtis, nevienalytės sritys.

Kaunas, 2021. 55 p.

Santrauka

Šioje disertacijoje yra nagrinėjama tam tikro mato u pasiskirstymo nustatymo problema stipriai nevienalytėse ir daugiasluoksnėse pralaidumo erdvėse. Elipsinė Dalinė Diferencialinė Lygtis, naudojama tokiam pasiskirstymui apibūdinti pusiausvyros būsenoje, yra vadinama Puasono lygtimi, kuri yra kilusi iš Darsi dėsnio. Ši lygtis apibūdina difuziją ir neturi laiko dimensijos, todėl su pakankamomis ribų sąlygomis ji gali būti vienareikšmiškai išspręsta naudojant skaitinius metodus. Palyginus Baigtinių Skirtumų metodą (BSM) su Baigtinių Tūrių metodu (BTM) yra pasiūlytas alternatyvus sprendimo būdas pasitelkiant Dirbtinius Neuroninius Tinklus (DNN).

Abu skaitmeniniai metodai buvo įgyvendinti dvimačiu atveju ir palyginti vienas su kitu stipriai nevienalytėse ir sluoksniuotose pralaidumo erdvėse. Kadangi BTM yra laikomas konservatyviu metodu, šis pranoko BSM keliais atžvilgiais. Pastarojo sprendiniai buvo nemonotoniški bei stipriai veikiami pralaidumų srities, prastai susidorojo su aštriais pralaidumų frontais. Remiantis šiais stebėjimais, buvo padaryta išvada, kad DNN būtų mokomas tik iš BTM sugeneruotų duomenų.

Gautas DNN parodė, jog gali atkartoti BTM metodą net esant sudėtingiems pralaidumo paskirstymams duotoje erdvėje, tačiau didelė triukšmo tarša rezultatuose palieka erdvės tolesniam tinklo architektūros tobulinimui ar net pačios metodologijos performulavimui. Tai, kartu su pasiūlymais dėl tolimesnių tyrimų ir galimo neuroninio tinklo tobulinimo, buvo aptarta plačiau paskutiniame skyriuje.

Table of Contents

List of Figures

List of Abbreviations and Terminology

Abbreviations:

FDM - Finite Difference Method;

FVM - Finite Volume Method;

FEM - Finite Element Method;

ANN - Artificial Neural Network;

PINN - Physics-Informed Neural Network;

CNN - Convolutional Neural Network;

CDNN - Coordinate-Discretized Neural Network;

PDE - Partial Differential Equation;

FP - Forward Problem;

IP - Inverse Problem;

BC - Boundary Condition;

1D - One-dimensional;

2D - Two-dimensional;

LR - Logistic Regression.

Terminology:

Permeability - a general term for conductivity in various media.

Heterogeneous domain - a domain with varying permeability values.

Layered domain - a domain with sharp boundaries between permeability values.

Measure/Species u - a measure whose distribution over a certain domain is defined by Poisson's equation. It can represent voltage, pressure, gravitation, etc.

Introduction

According to the *World Health Organization* "Epilepsy is a chronic non-communicable disease that causes recurrent seizures, which are brief episodes of involuntary movement that may involve a part of the body (partial) or the entire body (generalized) and are sometimes accompanied by loss of consciousness and control of bowel or bladder function" [40]. It is a debilitating disorder that can very well lead to a premature death and cause harsh life inconveniences. Localization of the source of epilepsy is one of the steps for a possible treatment, like using laser knives for the dismantlement of affected neuron synapses. This usually includes discretizing the domain of the brain and using numerical methods (mostly discretization based methods [36]) to determine the voltage distribution among the brain volume to pinpoint the anomalies that lie within.

A similar problem could be found in a totally different field known as fluid dynamics. In subsurface flow modeling, the propagation of fluids through porous media is studied. The task is analogous - determine the pressure gradient distribution in an equilibrium state that allows for such a movement. Whether it is electrostatics, fluid dynamics or even something as Newtonian gravitational fields, a mutual principle can be observed undergoing the name of Poisson's rule (commonly referred to as a Poisson's equation).

Using the aforementioned numerical methods is computationally expensive and uses a lot of operational time. Since the digital computers are nearing the limits of the transistor size being near microscopical, a time calls for a quality over quantity improvements for determining the distribution of an unknown measure u. Investigation of different methods and studies conducted creating numerical schemes tailored for such needs have been published to encourage a fundamental shift to the proposed ones [31], however this thesis is concerned not only in comparison study, but in proposing an Artificial Neural Network approach as well.

In this thesis, Finite Difference Method (FDM) as well as Finite Volume Method (FVM) are implemented on a two-dimensional (2D) grid to solve Poisson's differential equation over a highly heterogeneous and layered permeability domains. Conclusions are drawn from the two method comparison to construct and train a proposed Artificial Neural Network (ANN). Resulting Coordinate-Discretized Neural Network (CDNN) produces impressive results managing to replicate the high-performing FVM even on trickier permeability distributions. However, high noise contamination present in the results leaves room for further network architecture improvement and even reformulating the model approach. This and much more enhancements, further suggestions for development and such were discussed in greater detail in the concluding discussion chapter.

Thesis is organized as follows: Problem description, motivation and literature review are all presented in chapter 1. Domain discretization approach in numerical methods is presented in chapter 2, where section 2 is dedicated to FDM, section 3 is dedicated to FVM and section 4 is dedicated for the design of the CDNN. Results of 2D solutions by different aforementioned methods are presented and discussed in chapter 3.

1. Problem Review

This chapter is on literature review and familiarization with the main problem at hand: domain modelling, mesh generation, method introduction as well as main formulation of the problem and assumptions/simplifications made.

1.1. Motivation

Some areas of research are more developed than others, however the implementations are in most cases analogous. So it happens that most of the industrial applications come from subsurface modeling since the industry has profited a lot from the ongoing experimentations in the field.

Although FDM and Finite Element Method (FEM) has been around for many years and has been used to solve several industrial problems [6, 5, 43], years of common practice has led some industries to reject classical FEM and FDM as they do not deal with certain properties of subsurface structures - them being layered, anisotropic and heavily heterogeneous (when talking about the permeability of the porous media). These methods in question without modification are not well fit for the problem: FDM is neither locally nor globally conservative where as classic FEM is globally conservative, however requires post processing for conservative fluxes over element interfaces to be locally conservative [17]. Then there is FVM, which is both locally and globally conservative, but suffers from monotonicity issues in case of high heterogeneity over non K-orthogonal anisotropic domains [25, 27].

Over the past two decades, methods such "Flux-continuous Control Volume Distributed Finite-Volume schemes" for determining the fluid velocity and pressure distribution in ground reservoirs were gaining popularity [12, 25]. These schemes are usually called Multi-Point Flux approximation schemes [1]. Other similar schemes concerned in preserving flux continuity have also been widely researched using Discontinuous Galerkin methods [35], Mixed methods [10], Enriched Galerkin [8], etc. These aforementioned schemes do numerically converge on both structured and unstructured grids [25, 13].

In more recent times there is an ever-growing fascination with ANNs as the computational equipment is now able to deal with enormous amounts of data processing. Advances in machine learning sparked interest among researchers to develop Machine Learning algorithms for solving engineering and mathematical problems such as Ordinary Differential Equations using neural networks [7] as well as Partial Differential Equations (PDEs) [42]. A deep learning framework has been developed for solving steady state and turbulent flow [15, 41] as well as poro-elastic problems [37]. Even Recurrent Neural Networks such as Long Short-Term Memory models have been successfully implemented for history matching and forecasting of hydrocarbon flows in heterogeneous porous media [26]. The main motivation behind the ANNs is that strongly anisotropic and heterogeneous media causes even the most conservative methods to fail at satisfying the maximum principle and thus loosing the monotonicity of solution causing spurious oscillations in the numerical solution [28]. ANN ability to mimic numerical methods that have physical laws encoded within as well as learn from experimental data [32] makes them a captivating candidate for the problem.

In a very recent paper [19] it has been suggested to solve the Poisson's equation using Dirichlet principle to transform it to an equivalent energy functional. A deep-learning network known as ResNet could be then used to tackle a resulting Stochastic PDE. This, however, is not the approach that is taken in this thesis - everything from ANN architecture to the problem formulation and results are unique. Here, FDM [11], FVM [29] and CDNN are compared in their performance and ability to deal with high heterogeneity and layering of the domain. It is important to add that even though FEM would have broadened the scope of this work (especially with all of the possible modifications), it is not treated here due to time required for Master's thesis. Given the importance of such method however, there are plans to address it in the future research.

1.2. Domain Modeling

In order to create a real world model of a domain there are usually some sort of physical measuring devices involved that help determine specific structures within. If the domain is a human head then a Magnetic Resonance Imaging technique (commonly referred as MRI) is used to capture the brain layer, skull, cerebral spinal fluid and scalp tissue formation. In the subsurface case seismic waves are used that are sent down the Earth to bounce back from different geological layers or water/oil reservoirs. These kinds of domain modelling are widely open for research even now since the current technology struggles to produce highly detailed scans required for an accurate domain model depiction. For example, since the brain imaging field is a extremely subtle and finicky there is very little room for error or uncertainty, however, sometimes the resolution can be as low as $3mm \times 3mm \times 3mm$. Due to the uncertainties in conductivity values it is "necessary to simultaneously develop, for the EEG inverse problem, regularization methods less sensitive to modelization errors with distinct origins and effects on the data" [22].

Having said that, there have recently been some great strides in the field, where the team from *Massachusetts General Hospital Laboratory for Neuro-imaging of Coma and Consciousness* managed to obtain a scan that "took 100 hours to complete and can distinguish objects as small as 0.1 millimeters across" [24]. This accuracy is actually not only impressive, but sufficient enough to consider some truly meaningful brain analysis, not only in epilepsy source localization, but treating other brain related disorders as well. This includes, but is not limited to areas such as infamous protein misfoldings that lead to other detrimental diseases such as Alzheimer's, Parkinson's, etc.

Another problem with domain modelling is determining the specifics of the domain - like permeability/conductivity values at each discretized point. For example, some geological layers may have a general permeability, but the thickness variation and even the composition of the layer itself is always present, not to mention the possible directionality of the flow. The flow itself is happening on multiple scales, meaning that it is possible to observe fluid particles sifting the pores of the media in a microscopical level, fluid penetrating through the network of cracks or it can be observed as an overall flow through a discretized heterogeneous domain. The latter approach will be considered in this thesis, which would require a sensitivity study for post-processing the results as uncertainty in permeability of highly complicated domain can produce high errors. In a head modelling study [4] it was found that some areas are more sensitive to uncertainties than other. It was also learned that some areas of the head like the skull produce higher errors if the conductivity is underestimated, where as the scalp is more sensitive to error if the conductivity is overestimated.

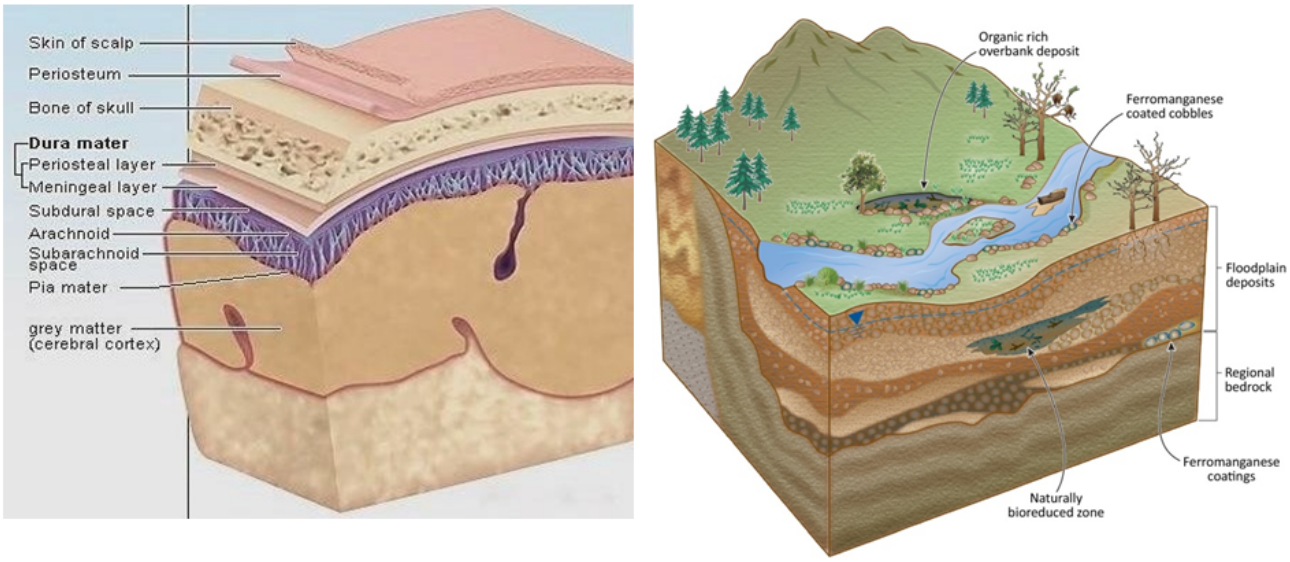


Figure 1.1: Subsurface [9] and brain [23] cross-section comparison - both of the domains bear a striking resemblance to one another visually. High heterogeneity, complicated structures, layering and anisopropy are present which require sophisticated methods and meshing techniques.

Having that in mind, the idea to work with realistic domains (figure 1.1) was abandoned due to difficulty obtaining sensitive real world data, extra time needed to program troublesome domain boundaries and the lack of clarity in demonstrating the effects of the two compared methods. The approach which is taken in this thesis is to solve the problem on a simplified 2D domain, which mimics heterogeneity observed either in a head model or subsurface modeling, but at a much smaller dimensional scale. Once the problem is solved on a smaller domain it can be generalized to large domain sizes and even 3D cases. There are two types of 2D permeability domains used in this thesis, both having 18×18 discretization nodes:

- A bi-valued 3×3 square arrangement, having $2^9 = 512$ different possible arrangements (see figure 1.2). This domain is a layering test for the methods.
- A highly heterogeneous domain generated using any smooth random noise method (like Simplex or Perlin [30]). This domain is a heterogeneity test for the methods.

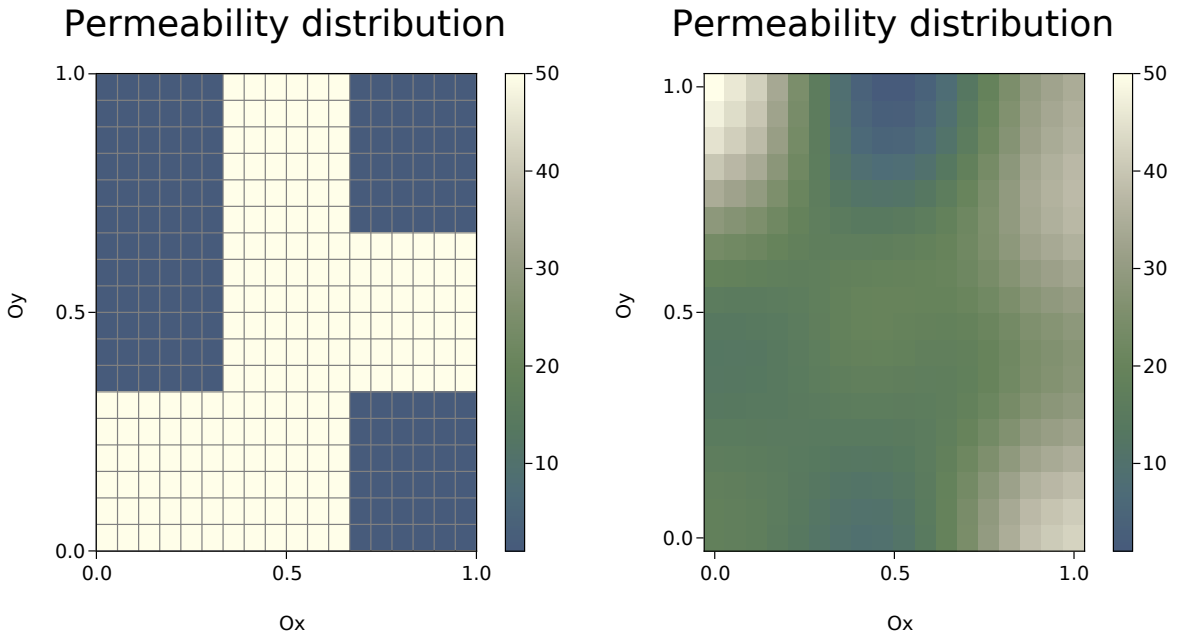
1.3. Problem Formulation

There are two problems associated with the topic:

- Forward Problem (FP) – determining field distribution of a particular measure in question assuming some driving function throughout the domain with a defined permeability. This is uniquely solved using numerical methods with a solution being relatively unresponsive to a slight change in system parameters [38].

- Inverse Problem (IP) is the other way around – estimating source distribution inside the domain from the field data observed. This is even a harder problem of recovery by being ill-posed and very sensitive to parameter change. There are specific methods to reduce the number of solutions.

Since the main goal is to compare certain aspects of the methods presented earlier, this thesis only focuses on working with FP. What is more, the experiments are carried out using a simple rectangular 2D domain with equal step lengths between the discretized grid nodes.



(a) (b)

Figure 1.2: (a) A two-dimensional 18×18 permeability domain obtained by creating bi-valued 3×3 square arrangement (each tile (large square) contains 6 discretization nodes). (b) A highly heterogeneous two-dimensional 18×18 permeability domain obtained by Gaussian Noise and Moving-Mean algorithms.

Another thing to assume is the quasi-static nature of the domain. Since a lot of problems of this nature are physics-related, they are usually multi-physics ones and require two classes of differential equations to be solved numerically: parabolic as well as elliptic. The logic behind it is that the flow initialized by some source creates a reacting field, whose gradient will dictate the current flow changes which completes a cycle. This is repeated until the cycle dies out due to natural resistance forces. Quasi-static nature means abandoning the idea of induction and sticking to solving equations unrelated to one another.

In this thesis the problem chosen can be formulated as an elliptic Poisson's PDE bounded by 2D domain Ω:

$$-div(\kappa \nabla u) = \phi, \quad \Omega = [0 \leq x \leq 1, 0 \leq y \leq 1] \tag{1.1}$$

BCs:

$$\begin{gathered} u(0,0) = 1, \quad u(1,1) = -1 \\ \frac{\partial u}{\partial y}(x,0) = 0, \quad \frac{\partial u}{\partial y}(x,1) = 0, \quad 0 < x < 1 \\ \frac{\partial u}{\partial x}(0,y) = 0, \quad \frac{\partial u}{\partial x}(1,y) = 0, \quad 0 < y < 1 \end{gathered} \tag{1.2}$$

Where ∇ is the nabla operator portraying divergence, κ is a diagonal permeability tensor, $u(x,y)$ is some observed measure value at a certain point (x,y). Domain Ω is presented here as a 1×1 square in x and y directions, where all of its boundaries are zero flux except for the bottom left and top right corners which define the source function ϕ.

The Poisson's rule implies that some observed measure u (voltage, pressure, etc.) is undergoing some change of changes. The equation heavily relies on second order derivatives introduced by the ∇ operators. Given sufficient conditions this divergence defines a source/sink-like behavior at every point of the domain.

The 2×2 permeability tensor being only diagonal means that the flow is described in two perependicular directions aligned to the Cartesian plane grid of the discretized domain. The values on the diagonal can differ implying anisotropy - one direction being more permeable than other.

1.4. Heterogeneity

The impact of heterogeneity is not to be underestimated when trying to get reliable results from a model. This is backed up by considerable amount of research and experiments done in seismology. The subsurface media heterogeneity comes from the separate geological layers that have different permeabilities to fluids. Figure 1.3 shows the impact such heterogeneity has on the fluid flow pattern in presence of sources and sinks. At the source the fluid is pumped, which is heavy, so that it may create pressure difference and push the oil out, which is then collected at the sink.

Depending on how heterogeneous this system is, the flow inside the reservoir can take different paths, but in order to accurately model this, there is a need for adequate numerical methods.

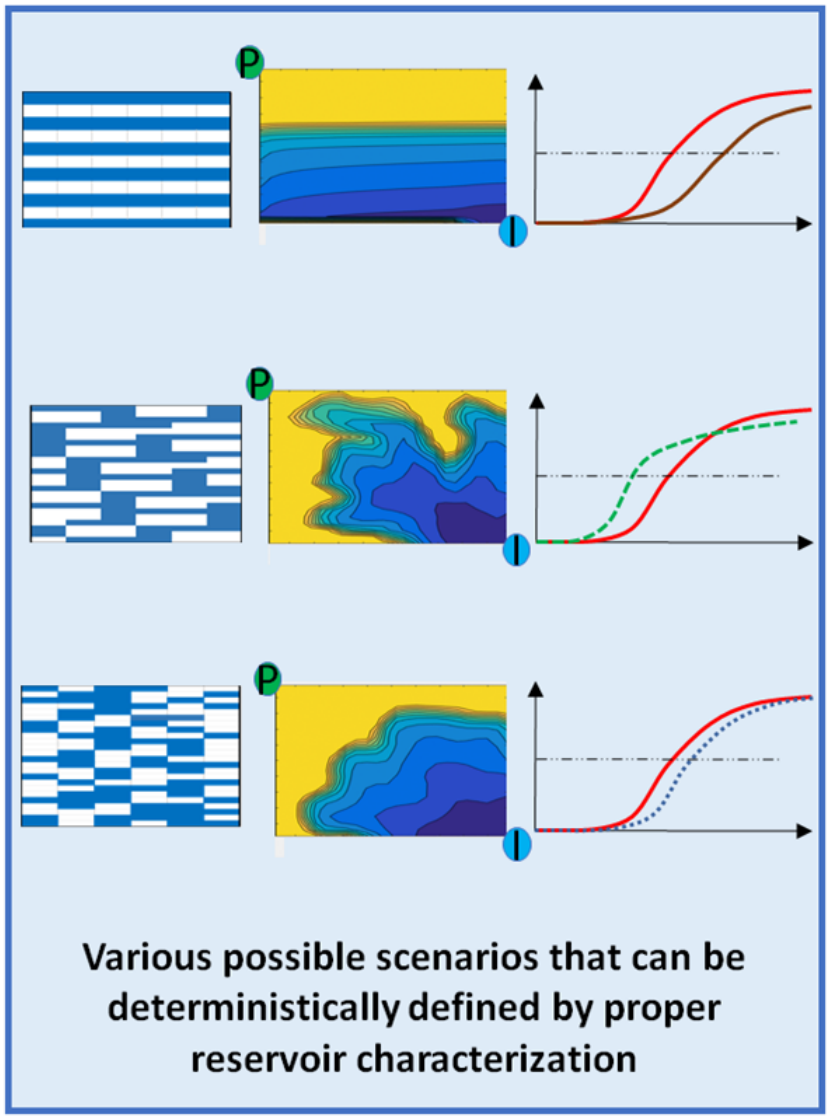


Figure 1.3: Impacts of different heterogeneities (layering included) of the domain on a numerical solution: A second phase fluid is being introduced by a source into a subsurface, where the sink collects the first phase. Different permeability structures can be seen on the left and their corresponding fluid distribution in the middle, where as the production graph of the second phase over time is depicted on the right side.

2. Methodology

In this chapter the first few sections focus on the FDM, FVM and ANN implementation, methodology and approaches used for the experiment as well as the assumptions/simplifications made.

2.1. Numerical Methods

The main idea behind the numerical methods is to discretize a given domain. In a 2D case this means that instead of having a continuous range of values in the domain from 0 to 1 in x direction, there are only a finite amount (n_x) of values introduced on the domain. The same applies for the y direction having n_y points.

While FVM can in fact adapt to any complex boundary, FDM requires a rectangular discretization aligned with the domain thus failing to capture complex structures as one presented in figure 2.1.

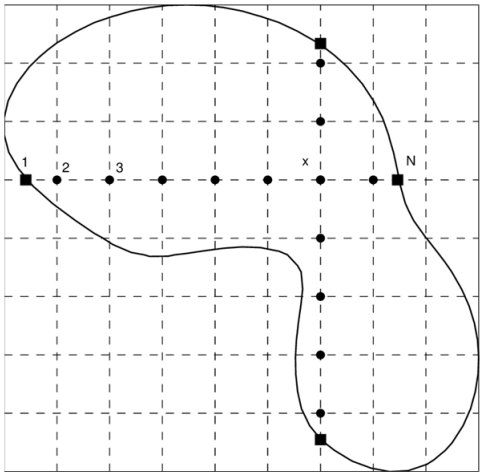


Figure 2.1: Example domain with a complex boundary that is problematic for FDM [21].

Since this thesis is focused on comparing other properties of these methods, the chosen grid is rectangular, regular and aligned with the direction of permeability (see chapter 1).

In that case, discretization points (or nodes) and steps in between them (assuming a uniform mesh) can be described as follows:

$$\begin{aligned} h_x = \frac{1}{n_x - 1}; \quad h_y = \frac{1}{n_y - 1} \\ x_1 = 0, \quad x_{n_x} = 1; \quad y_1 = 0, \quad y_{n_y} = 1 \\ x_i = x_1 + (i-1)h_x; \quad y_j = y_1 + (j-1)h_y \end{aligned} \tag{2.1}$$

Where $i = (1, ..., n_x)$ and $j = (1, ..., n_y)$. In this thesis there are no complex numbers and i is used only for indexing!

Similarly, the evaluation of u at a certain domain point will be referred like this:

$$u_{i,j} = u(x_i, y_j) \tag{2.2}$$

2.2. Finite Difference Method

In numerical analysis, Finite Difference Methods (FDM) are a class of numerical techniques for solving differential equations by approximating derivatives with finite differences:

$$v'(x_i) = \frac{v(x_i + h_x) - v(x_i)}{h_x} \tag{2.3}$$

Where $v(x)$ is some continuous and differentiable function, $v'(x)$ is its derivative, x_i is some point from the domain of $v(x)$ and h_x is a relatively small step taken along Ox direction between $v(x_i)$ and $v(x_i + h_x)$.

This is one of the simplest ways to approximate 1D first order derivate that has a truncation error of order $O(h_x)$, however it is the most intuitive way to approach FDM.

Central difference approximation is another (more accurate) way to approximate using both sides of finite differences around some central point x_i:

$$v'(x_i) = \frac{v(x_i + h_x) - v(x_i - h_x)}{2h_x} \tag{2.4}$$

This comes from the Taylor series expansion of the forward and backward differences:

$$v(x_i + h_x) = v(x_i) + h_x v'(x_i) + \frac{h_x^2}{2} v''(x_i) + ... \tag{2.5}$$

$$v(x_i - h_x) = v(x_i) + (-h_x) v'(x_i) + \frac{(-h_x)^2}{2} v''(x_i) + ... \tag{2.6}$$

Taking difference between these two differences gives central difference approximation which is of order $O(h_x^2)$, since the second order derivative terms cancel out:

$$v(x_i + h_x) - v(x_i - h_x) = v(x_i) - v(x_i) + h_x v'(x_i) + h_x v'(x_i) + \frac{h_x^2}{2} v''(x_i) - \frac{h_x^2}{2} v''(x_i) + ... \tag{2.7}$$

For second order derivatives a central difference approximation is also used, where the main idea comes from applying finite difference of forward and backward differences:

$$\frac{\partial^2 v}{\partial x^2}(x_i) = \frac{d}{dx}\left(\frac{dv}{dx}\right)(x_i) = \frac{\frac{v_{i+1} - v_i}{h_x} - \frac{v_i - v_{i-1}}{h_x}}{h_x} = \frac{v_{i+1} - 2v_i + v_{i-1}}{h_x^2} \tag{2.8}$$

Getting back to the main problem PDE (equation 1.1):

$$\nabla \cdot (\kappa \nabla u) = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \cdot \left(\begin{pmatrix} \xi & 0 \\ 0 & \eta \end{pmatrix} \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} \right) = \xi \frac{\partial^2 u}{\partial x^2} + \eta \frac{\partial^2 u}{\partial y^2} = -\phi \tag{2.9}$$

Where ξ stands for permeability along x direction and η is permeability along y direction.

Using central differences:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2}, \quad \frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2} \tag{2.10}$$

Put everything back to PDE (eq. 2.9):

$$\frac{u_{i-1,j}\xi_{i-1,j} - 2u_{i,j}\xi_{i,j} + u_{i+1,j}\xi_{i+1,j}}{h_x^2} + \frac{u_{i,j-1}\eta_{i,j-1} - 2u_{i,j}\eta_{i,j} + u_{i,j+1}\eta_{i,j+1}}{h_y^2} = -\phi_{i,j} \tag{2.11}$$

For spacial visualization it can rewritten as such:

$$\begin{array}{ccccc} 0 \cdot u_{i-1,j+1} & & \frac{\eta_{i,j+1}}{h_y^2} \cdot u_{i,j+1} & & 0 \cdot u_{i+1,j+1} \\ & + & + & + & \\ \frac{\xi_{i-1,j}}{h_x^2} \cdot u_{i-1,j} & + & -2\frac{\xi_{i,j}}{h_x^2} \cdot u_{i,j} - 2\frac{\eta_{i,j}}{h_y^2} \cdot u_{i,j} & + & \frac{\xi_{i+1,j}}{h_x^2} \cdot u_{i+1,j} = -\phi_{i,j} \\ & + & + & + & \\ 0 \cdot u_{i-1,j-1} & & \frac{\eta_{i,j-1}}{h_y^2} \cdot u_{i,j-1} & & 0 \cdot u_{i+1,j-1} \end{array} \tag{2.12}$$

Every point takes its u value into consideration as well as its closest neighbors'. For every single $u_{i,j}$ point there is a need for a solution of the equation 2.12. Given some boundary conditions, this can be done by solving a system of $(n_x - 2) \cdot (n_y - 2)$ such equations:

$$\begin{aligned}
&\frac{u_{1,2}\xi_{1,2} - 2u_{2,2}\xi_{2,2} + u_{3,2}\xi_{3,2}}{h_x^2} + \frac{u_{2,1}\eta_{2,1} - 2u_{2,2}\eta_{2,2} + u_{2,3}\eta_{2,3}}{h_y^2} = -\phi_{2,2} \\
&\frac{u_{2,2}\xi_{2,2} - 2u_{3,2}\xi_{3,2} + u_{4,2}\xi_{4,2}}{h_x^2} + \frac{u_{3,1}\eta_{3,1} - 2u_{3,2}\eta_{3,2} + u_{3,3}\eta_{3,3}}{h_y^2} = -\phi_{3,2} \\
&\frac{u_{3,2}\xi_{3,2} - 2u_{4,2}\xi_{4,2} + u_{5,2}\xi_{5,2}}{h_x^2} + \frac{u_{4,1}\eta_{4,1} - 2u_{4,2}\eta_{3,2} + u_{4,3}\eta_{4,3}}{h_y^2} = -\phi_{4,2} \\
&\cdots \\
&\frac{u_{1,3}\xi_{1,3} - 2u_{2,3}\xi_{2,3} + u_{3,3}\xi_{3,3}}{h_x^2} + \frac{u_{2,2}\eta_{2,2} - 2u_{2,3}\eta_{2,3} + u_{2,4}\eta_{2,4}}{h_y^2} = -\phi_{2,3} \\
&\frac{u_{2,3}\xi_{2,3} - 2u_{3,3}\xi_{3,3} + u_{4,3}\xi_{4,3}}{h_x^2} + \frac{u_{3,2}\eta_{3,2} - 2u_{3,3}\eta_{3,3} + u_{3,4}\eta_{3,4}}{h_y^2} = -\phi_{3,3} \\
&\frac{u_{3,3}\xi_{3,3} - 2u_{4,3}\xi_{4,3} + u_{5,3}\xi_{5,3}}{h_x^2} + \frac{u_{4,2}\eta_{4,2} - 2u_{4,3}\eta_{4,3} + u_{4,4}\eta_{4,4}}{h_y^2} = -\phi_{4,3} \\
&\cdots
\end{aligned} \tag{2.13}$$

This system of linear equations can be rewritten in a matrix form. Say, the grid is 4×3 with equal distances between the nodes. Give every node its number from left to right and then from bottom to top:

$$\begin{matrix} u_{1,3} & u_{2,3} & u_{3,3} & u_{4,3} \\ u_{1,2} & u_{2,2} & u_{3,2} & u_{4,2} \\ u_{1,1} & u_{2,1} & u_{3,1} & u_{4,1} \end{matrix} \rightarrow \begin{pmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{4,1} \\ \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{4,2} \\ \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \\ u_{4,3} \end{pmatrix} \tag{2.14}$$

This giant $n_x \cdot n_y$ element vector represents the u distribution over the domain Ω, so we call it **u**. The goal is to find **u** such that:

$$A\mathbf{u} = -\boldsymbol{\phi} \tag{2.15}$$

The numerical stencil says not only what elements are being considered, but also their weights (equation 2.11). These weights are put inside a banded matrix A, so that when they are multiplied with vector **u**, the product is the source vector $-\boldsymbol{\phi}$.

The band of a large matrix A can be thought of as consisting of three mini matrices in itself:

1. Diagonal matrix on the left-side:

$$\begin{bmatrix} \frac{\eta_{1,j-1}}{h_y^2} & 0 & 0 & 0 \\ 0 & \frac{\eta_{2,j-1}}{h_y^2} & 0 & 0 \\ 0 & 0 & \frac{\eta_{3,j-1}}{h_y^2} & 0 \\ 0 & 0 & 0 & \frac{\eta_{4,j-1}}{h_y^2} \end{bmatrix} \tag{2.16}$$

   Which represents the coefficients associated with nodes $(i, j-1)$ (that are located below the central nodes (i, j));

2. Tridiagonal matrix in the middle:

$$\begin{bmatrix} -2\frac{\xi_{1,j}}{h_x^2} - 2\frac{\eta_{1,j}}{h_y^2} & \frac{\xi_{2,j}}{h_x^2} & 0 & 0 \\ \frac{\xi_{1,j}}{h_x^2} & -2\frac{\xi_{2,j}}{h_x^2} - 2\frac{\eta_{2,j}}{h_y^2} & \frac{\xi_{3,j}}{h_x^2} & 0 \\ 0 & \frac{\xi_{2,j}}{h_x^2} & -2\frac{\xi_{3,j}}{h_x^2} - 2\frac{\eta_{3,j}}{h_y^2} & \frac{\xi_{4,j}}{h_x^2} \\ 0 & 0 & 2\frac{\xi_{3,j}}{h_x^2} & -2\frac{\xi_{4,j}}{h_x^2} - 2\frac{\eta_{4,j}}{h_y^2} \end{bmatrix} \tag{2.17}$$

   Which represents the stencils' middle three coefficients of the central node (i, j) and the nodes to the left $(i-1, j)$ and right $(i+1, j)$;

3. Diagonal matrix on the right-side:

$$\begin{bmatrix} \frac{\eta_{1,j+1}}{h_y^2} & 0 & 0 & 0 \\ 0 & \frac{\eta_{2,j+1}}{h_y^2} & 0 & 0 \\ 0 & 0 & \frac{\eta_{3,j+1}}{h_y^2} & 0 \\ 0 & 0 & 0 & \frac{\eta_{4,j+1}}{h_y^2} \end{bmatrix} \tag{2.18}$$

   Which represents the coefficients assosiated with nodes $(i, j+1)$ (that are located above the central nodes (i, j));

In order to be able to visualise the large matrix A, the stencil constants will be addressed accordingly:

1. Central node constants will be denoted as $c = -2\frac{\xi_{i,j}}{h_x^2} - 2\frac{\eta_{i,j}}{h_y^2}$;
2. Left and right node constants will be denoted as $l = \frac{\xi_{i-1,j}}{h_x^2}$ and $r = \frac{\xi_{i+1,j}}{h_x^2}$ respectively;
3. Below and above node constants will be denoted as $b = \frac{\eta_{i,j-1}}{h_y^2}$ and $a = \frac{\eta_{i,j+1}}{h_y^2}$ respectively;

Equation 2.19 presents matrix A in a segmented way to help visualise the diagonal and tridiagonal matrices mentioned earlier.

$$\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ l & c & r & 0 & 0 & 2u & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & l & c & r & 0 & 0 & 2u & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2l & c & 0 & 0 & 0 & 2u & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & c & 2r & 0 & 0 & u & 0 & 0 & 0 \\ 0 & b & 0 & 0 & l & c & r & 0 & 0 & u & 0 & 0 \\ 0 & 0 & b & 0 & 0 & l & c & r & 0 & 0 & u & 0 \\ 0 & 0 & 0 & b & 0 & 0 & 2l & c & 0 & 0 & 0 & u \\ 0 & 0 & 0 & 0 & 2b & 0 & 0 & 0 & c & 2r & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2b & 0 & 0 & l & c & r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2b & 0 & 0 & l & c & r \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{4,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{4,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \\ u_{4,3} \end{bmatrix} \tag{2.19}$$

Every line of the matrix A is dedicated to writing an equation for each $(i,j)^{th}$ element which are located on the main diagonal. Left and right diagonal matrices deal with elements included in the stencil (equation 2.11) that are sided below and above (from the domain perspective), where the tridiagonal matrix are about the elements on the left and right.

Matrix multiplication of A and **u** will now yield $-\boldsymbol{\phi}$. Implementing the Dirichlet BCs alters the first and last rows of matrix A to only include the main node and for it to be equal to some number:

$$u_{0,0} = 1, \quad u_{n_x,n_y} = -1 \tag{2.20}$$

The 1 and -1 are added to the source term $-\boldsymbol{\phi}$:

$$-\boldsymbol{\phi} = (1, 0, 0, ..., -1)^T \tag{2.21}$$

Implementing the Neumann BCs alters:

1. The first and last rows of the tridiagonal mini matrices (situated at the diagonal):
   Since the $\begin{bmatrix} l & c & r \end{bmatrix}$ band going through the center cannot be fully included at the matrix's corners they are cut off like such: $\begin{bmatrix} c & r & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 & l & c \end{bmatrix}$. However, the l and r values are doubled. This comes from the fact that the nodes at the left boundary lack the neighbors to their left so they compensate by doubling the value of the right neighbors'. This is because of the Neumann condition implies, that the missing left (or 0^{th}) node should behave like the right one:

$$\begin{gathered} \frac{\partial u}{\partial x}(0,y) = \frac{u_{2,j} - u_{0,j}}{2h_x} = 0 \\ u_{0,j} = u_{2,j} \end{gathered} \tag{2.22}$$

   Analogous conclusion can be derived for the right boundary:

$$\begin{gathered} \frac{\partial u}{\partial x}(1,y) = \frac{u_{n_{x+1},j} - u_{n_{x-1},j}}{2h_x} = 0 \\ u_{n_{x+1},j} = u_{n_{x-1},j} \end{gathered} \tag{2.23}$$

2. The first and last diagonal mini matrices inside matrix A are doubled. This comes from the fact that the nodes at the bottom boundary lack the neighbors below, so they compensate by doubling the value of the neighbors above:

$$\frac{\partial u}{\partial y}(x,0) = \frac{u_{i,2} - u_{i,0}}{2h_y} = 0$$
$$u_{i,0} = u_{i,2} \tag{2.24}$$

Analogous conclusion can be derived for the right boundary:

$$\frac{\partial u}{\partial y}(x,1) = \frac{u_{i,n_{y+1}} - u_{i,n_{y-1}}}{2h_y} = 0$$
$$u_{i,n_{y+1}} = u_{i,n_{y-1}} \tag{2.25}$$

Now the equation 2.15 can be solved for **u**:

$$\mathbf{u} = A^{-1} \times (-\boldsymbol{\phi}) \tag{2.26}$$

Since A is usually large and sparse, a Gaussian elimination method can be used for approximating the solution:

$$\mathbf{u} = A \backslash (-\boldsymbol{\phi}) \tag{2.27}$$

Vector **u** is then reconstructed back into the discretized grid in order to see the solution in its primary 2D form.

2.3. Finite Volume Method

Finite Volume Method is also numerical and requires defining control volumes around the points of discretization (figure 2.2) that conserve the flux. This control is what makes FVM superior over the FDM in this case, as it allows the flux propagation to be directional and uneven. Having control volumes also allows for a recreation of a much more profound boundary of the domain space, which was absent in FDM.

As mentioned before, FVM requires conservation of species u among the domain which is achieved by integrating over Ω:

$$\iint_\Omega -div(\kappa \nabla u) dS = \iint_\Omega \phi \, dS \tag{2.28}$$

This equation can be quite easily expanded as:

$$\iint_\Omega \xi \frac{\partial^2 u}{\partial x^2} \, dS + \iint_\Omega \eta \frac{\partial^2 u}{\partial y^2} \, dS = - \iint_\Omega \phi \, dS \tag{2.29}$$

The idea is to take each term and integrate over each of the finite volumes rather than the whole domain. If the flux of u is conserved with every finite volume, then it will also be conserved over all of the domain. Since there are $n_x \cdot n_y$ nodes, there will be the same amount of finite volumes and thus - equations, which will be solvable for the u distribution over Ω. Let us zoom in on a point C

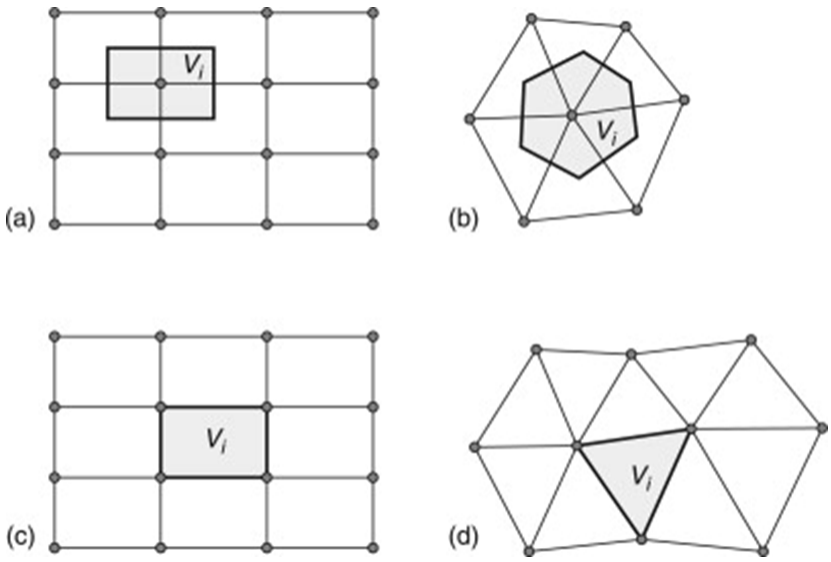


Figure 2.2: Depiction of FVM's control volumes: a, b are vertex centered and c, d are cell centered, where a, c are defined on a rectangular and b, d are on a triangular grid [2].

with its corresponding finite volume, where nodes W, E, S and N are the ones surrounding C for a five-point stencil scheme (figure 2.3). To begin with, integrate the first term containing the second order derivative with respect to x over the finite volume and discretize to finite differences:

$$\begin{aligned}\int_{y_s}^{y_n}\int_{x_w}^{x_e}\frac{\partial}{\partial x}\left(\xi\frac{\partial u}{\partial x}\right)dxdy = \int_{y_s}^{y_n}\left(\xi\frac{\partial u}{\partial x}\bigg|_{x_e} - \xi\frac{\partial u}{\partial x}\bigg|_{x_w}\right)dy \approx \\ \left(\xi(x_e)\frac{u(E)-u(C)}{\delta x_e} - \xi(x_w)\frac{u(C)-u(W)}{\delta x_w}\right)\Delta y\end{aligned} \tag{2.30}$$

Here $\xi(x_w)$ marks the permeability estimate at the left boundary of the finite volume, where $\xi(x_e)$ marks the permeability estimate at the right boundary.

Now, integrate the second term containing the second order derivative with respect to y over the finite volume and discretize to finite differences:

$$\begin{aligned}\int_{x_w}^{x_e}\int_{y_s}^{y_n}\frac{\partial}{\partial y}\left(\eta\frac{\partial u}{\partial y}\right)dydx = \int_{x_w}^{x_e}\left(\eta\frac{\partial u}{\partial y}\bigg|_{y_n} - \eta\frac{\partial u}{\partial y}\bigg|_{y_s}\right)dx \approx \\ \left(\eta(y_n)\frac{u(N)-u(C)}{\delta y_n} - \eta(y_s)\frac{u(C)-u(S)}{\delta y_s}\right)\Delta x\end{aligned} \tag{2.31}$$

Here $\eta(y_s)$ marks the permeability estimate at the bottom boundary of the finite volume, where $\eta(y_n)$ marks the permeability estimate at the top boundary.

Lastly, integrate the third term containing the driving function over the finite volume:

$$-\int_{y_s}^{y_n}\int_{x_w}^{x_e}\phi\,dxdy = -\int_{x_w}^{x_e}\int_{y_s}^{y_n}\phi\,dydx = -\phi(C)\Delta x\Delta y \tag{2.32}$$

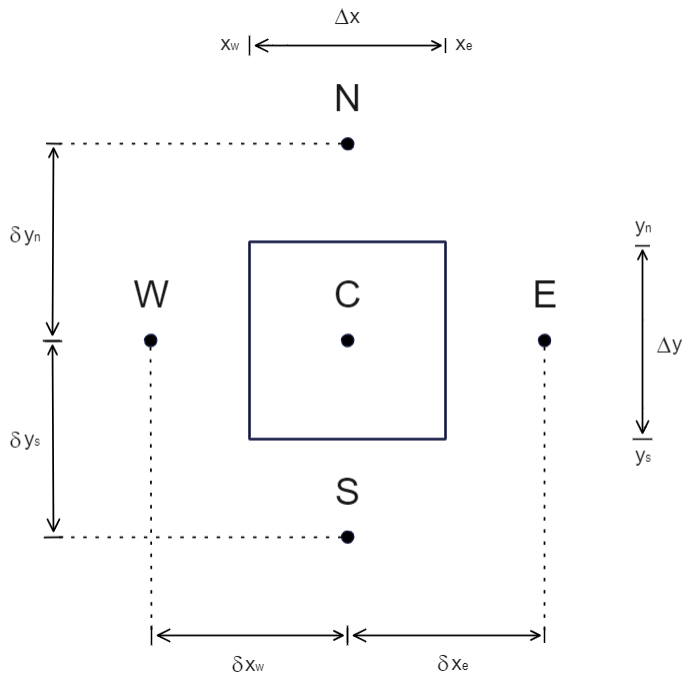


Figure 2.3: FVM control volume representation with respect to the discretization nodes around.

Also, looking into figure 2.3 it is clear to see that:

$$\Delta y = (y_n - y_s) \quad \Delta x = (x_e - x_w) \tag{2.33}$$

Substituting equations 2.30, 2.31, and 2.32 into equation 2.29 produces:

$$\begin{gathered}\left(\xi(x_e)\frac{u(E)-u(C)}{\delta x_e} - \xi(x_w)\frac{u(C)-u(W)}{\delta x_w}\right)\Delta y + \\ \left(\eta(y_n)\frac{u(N)-u(C)}{\delta y_n} - \eta(y_s)\frac{u(C)-u(S)}{\delta y_s}\right)\Delta x = \\ -\phi(C)\Delta x \Delta y\end{gathered} \tag{2.34}$$

Permeability coefficients at the interfaces of the finite volumes ($\xi(x_w)$, $\xi(x_e)$, $\eta(y_s)$ and $\eta(y_n)$) are not given and must be somehow evaluated. This can be done by linear interpolation, however, paper [33] suggests preserving the continuity of the flux at the interfaces between the finite volumes of the nodes that are next to each other. This comes with an assumption, that the permeability is uniform inside the control volume, which is not true, but completely reasonable. Since permeabilities are given at each node and the discretization scheme is defined as a rectangular one - control volumes can be defined in little rectangles that bound the permeability specified in the middle. Assumption becomes true as control volume size tends to zero.

The idea that the rate of change (for example in x direction) of species u over the finite volume (e.g. of the node W) with a given uniform permeability ($\xi(W)$) has to be the same as the rate of change

(again, in x direction) of species u over the finite volume (i.e. of the node C) with a given uniform permeability ($\xi(C)$), it can be mathematically expressed as such (first see equation 2.41):

$$\xi(W)\frac{u(w)-u(W)}{\frac{\Delta_x}{2}} = \xi(C)\frac{u(C)-u(x_w)}{\frac{\Delta_x}{2}} \tag{2.35}$$

This can be tied to the unknown $\xi(x_w)$ by thinking about the overall change from point W to C which also has to be of the same magnitude. In mathematical notation this becomes:

$$\begin{aligned}\xi(x_w)\frac{u(C)-u(W)}{\Delta x} &= \xi(W)\frac{u(x_w)-u(W)}{\frac{\Delta_x}{2}} \\ \xi(x_w)\frac{u(C)-u(W)}{\Delta x} &= \xi(C)\frac{u(C)-u(x_w)}{\frac{\Delta_x}{2}}\end{aligned} \tag{2.36}$$

Note that the term $u(x_w)$ is not a rigorous notation and is only used to notate u value right in between the nodes W and C.

Dividing both equations by Δx and moving permeability constants to the left yields:

$$\begin{aligned}u(C)-u(W)\frac{\xi(x_w)}{\xi(W)} &= 2(u(x_w)-u(W)) \\ u(C)-u(W)\frac{\xi(x_w)}{\xi(C)} &= 2(u(C)-u(x_w))\end{aligned} \tag{2.37}$$

Add the two equations to cancel the $u(x_w)$ terms:

$$(u(C)-u(W))\left(\frac{\xi(x_w)}{\xi(W)}+\frac{\xi(x_w)}{\xi(C)}\right) = 2(u(C)-u(W)) \tag{2.38}$$

Divide by common factor and solve for $\xi(x_w)$:

$$\xi(x_w) = 2\cdot\left(\frac{1}{\xi(W)}+\frac{1}{\xi(C)}\right)^{-1} = 2\cdot\frac{\xi(W)\xi(C)}{\xi(C)+\xi(W)} \tag{2.39}$$

Analogously, it can be derived for the remaining three constants $\xi(x_e)$, $\eta(y_s)$ and $\eta(y_n)$:

$$\begin{aligned}\xi(x_w) &= 2\cdot\frac{\xi(W)\xi(C)}{\xi(C)+\xi(W)} \qquad & \xi(x_e) &= 2\cdot\frac{\xi(C)\xi(E)}{\xi(E)+\xi(C)} \\ \eta(y_s) &= 2\cdot\frac{\eta(S)\eta(C)}{\eta(C)+\eta(S)} \qquad & \eta(y_n) &= 2\cdot\frac{\eta(C)\xi(N)}{\eta(N)+\eta(C)}\end{aligned} \tag{2.40}$$

One last step before generalizing everything to a stencil is to remember that all of the finite volumes in this thesis are defined as equally big squares around the nodes, so it can be said that:

$$\begin{aligned}\delta x_w = \delta x_e = \Delta x = x_e - x_w = \frac{x_{i+1}-x_{i-1}}{2} \\ \delta y_s = \delta y_n = \Delta y = y_n - y_s = \frac{y_{j+1}-y_{j-1}}{2}\end{aligned} \tag{2.41}$$

Now, let us look at the bigger picture where the C node is now (i, j), so W is $(i-1, j)$, E is $(i+1, j)$, S is $(i, j-1)$ and N is $(i, j+1)$, then equation 2.34 can be expressed in a following way:

$$\begin{gathered}(y_{j+1}-y_{j-1})\left(\frac{\xi_{i,j}\xi_{i+1,j}}{\xi_{i+1,j}+\xi_{i,j}}\frac{u_{i+1,j}-u_{i,j}}{x_{i+1}-x_i}-\frac{\xi_{i-1,j}\xi_{i,j}}{\xi_{i,j}+\xi_{i-1,j}}\frac{u_{i,j}-u_{i-1,j}}{x_i-x_{i-1}}\right)+\\(x_{i+1}-x_{i-1})\left(\frac{\eta_{i,j}\eta_{i,j+1}}{\eta_{i,j+1}+\eta_{i,j}}\frac{u_{i,j+1}-u_{i,j}}{y_{j+1}-y_j}-\frac{\eta_{i,j-1}\eta_{i,j}}{\eta_{i,j}+\eta_{i,j-1}}\frac{u_{i,j}-u_{i,j-1}}{y_j-y_{j-1}}\right)=\\-\phi_{i,j}\frac{x_{i+1}-x_{i-1}}{2}\frac{y_{j+1}-y_{j-1}}{2}\end{gathered}\tag{2.42}$$

After rearranging the terms, so that the values of u at every node would be expressed and accounting for equal distances between the nodes, the stencil looks like this:

$$\begin{gathered}\frac{y_{j+1}-y_{j-1}}{x_i-x_{i-1}}\frac{\xi_{i-1,j}\xi_{i,j}}{\xi_{i,j}+\xi_{i-1,j}}\cdot\boldsymbol{u_{i-1,j}}+\frac{y_{j+1}-y_{j-1}}{x_{i+1}-x_i}\frac{\xi_{i,j}\xi_{i+1,j}}{\xi_{i+1,j}+\xi_{i,j}}\cdot\boldsymbol{u_{i+1,j}}-\\\frac{y_{j+1}-y_{j-1}}{x_i-x_{i-1}}\left(\frac{\xi_{i-1,j}\xi_{i,j}}{\xi_{i,j}+\xi_{i-1,j}}+\frac{\xi_{i,j}\xi_{i+1,j}}{\xi_{i+1,j}+\xi_{i,j}}\right)\cdot\boldsymbol{u_{i,j}}-\\\frac{x_{i+1}-x_{i-1}}{y_j-y_{j-1}}\left(\frac{\eta_{i,j-1}\eta_{i,j}}{\eta_{i,j}+\eta_{i,j-1}}+\frac{\eta_{i,j}\eta_{i,j+1}}{\eta_{i,j+1}+\eta_{i,j}}\right)\cdot\boldsymbol{u_{i,j}}+\\\frac{x_{i+1}-x_{i-1}}{y_j-y_{j-1}}\frac{\eta_{i,j-1}\eta_{i,j}}{\eta_{i,j}+\eta_{i,j-1}}\cdot\boldsymbol{u_{i,j-1}}+\frac{x_{i+1}-x_{i-1}}{y_{j+1}-y_j}\frac{\eta_{i,j}\eta_{i,j+1}}{\eta_{i,j+1}+\eta_{i,j}}\cdot\boldsymbol{u_{i,j+1}}=\\-\phi_{i,j}\frac{x_{i+1}-x_{i-1}}{2}\frac{y_{j+1}-y_{j-1}}{2}\end{gathered}\tag{2.43}$$

Similar to the FDM case, this numerical stencil can be used to create the matrix A for working out the *u* distribution in vector form **u** (equation 2.27).

2.4. Neural Network Method

Machine learning is a field of mathematics-computer science for artificial intelligence that analyzes algorithms that solve various problems without giving "precise" instructions. These algorithms can be loosely classified into supervised and non-supervised learning, however here the focus is on the former. Each time a machine solves a particular problem from a given input, some relevant feedback is obtained about its output by comparing it to a "true" result. At first epochs the computer is untrained and the errors are high, however, with time it manages to learn by re-selecting a set of parameters automatically for more accurate results.

Artificial Neural Network is a very renowned type of machine learning algorithms that is used for many difficult problems such as applications for image and language processing, forecasting, risk assessment, market research, etc. The idea behind one is trying to recognize an underlying relationship between some input and output vectors. This is done by mimicking a phenomenon inside the brain where nerve cells (called neurons) create intricate relationships of varying strength between one another that resembles a net, thus such algorithm earned its name as a "neural network". ANN is known for being an universal function approximator - this means that it may, in theory, given sufficient amount of data that is processed for a specific problem and fed accordingly, imitate any possible relationship (where there is one).

There are two types of problems ANN can deal with: classification and regression. Classification usually outputs some kind of probability estimate of a defined occurrence, whereas regression outputs values of a function at a certain input. The main focus in this thesis is on the latter, however it is extremely important to understand the part that is usually taught in classification approach - Logistic Regression (LR). This is because LR is in a way a building block of NN as it defines what a neuron is.

2.4.1. Logistic Regression and Neural Networks

The idea behind LR is to map independent variable space to some extent of the activation function (sigmoid if talking strictly LR). Let us say, that input values range from 0 to 1, by adding some constant term (called bias) and multiplying by a scalar value (called weight) it is possible to modify the input space $[0, 1]$ to any $[a, b]$, $a, b \in \mathbb{R}$ space already. That way it is possible to take only a certain range of activation function values to create a non-linearity. If bias $b = 2$ and weight $w = 3$, where activation function is sigmoid, the input space $[0, 1]$ is firstly mapped to $[2, 5]$ linear space and then to $[0.88, 0.99]$ space of a non-linear nature:

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$

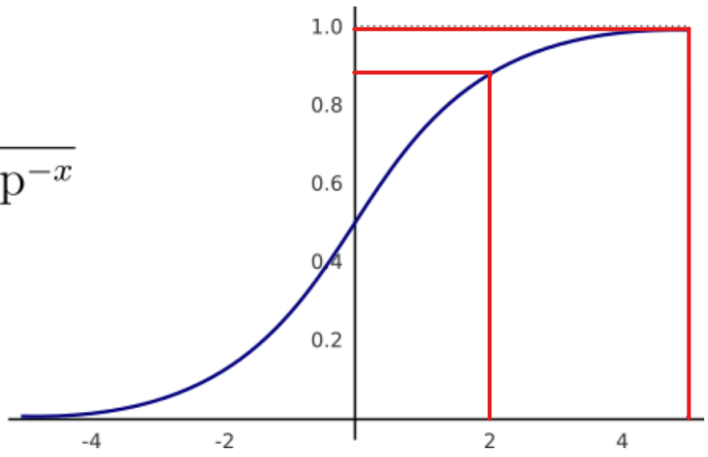


Figure 2.4: Illustration of adding non-linearity to via sigmoidal curve.

Using one LR approach will only yield strictly monotonous solutions (either increasing in value or decreasing), however if two or more LRs are used on the same input - they can be added to construct almost any shape imaginable much like Fourier series do. That is what the idea behind basic feed-forward ANN is - stacking LR approach multiple times and repeatedly feeding modified input spaces into sequential modifiers. However, the weights are guided to produce a wanted output (called target) using back-propagation and optimization algorithms. If weights are not optimal - the output of the ANN is far from a target by some error calculated by a cost function (e.g. Mean Square Error). This cost function has to be minimized by weights (e.g. using Gradient Descent algorithm) so that the error would be as small as possible. A lot of minimization algorithms make use of cost function gradient which is very computationally expensive to compute, so a back-propagation algorithm is usually used [16] to approximate the gradient.

2.4.2. Physics-Informed Neural Networks

Turning back to the problem at hand (equation 1.1), the way recent research suggest to solve PDE related problems is by specifically constructed ANNs called Physics-Informed Neural Networks (PINNs) [32].

Real world phenomena like subsurface flows tend to go by some laws of nature, for example - Darcy's Law, from which the Poisson's equation arises. These laws are great in general, however, the real world data paints a similar, but not the exact same picture as the numerically computed one as there are always numerous other factors that are in effect. PINNs try to learn those other factors additionally and make the model a bit more accurate.

The crux of that kind of NN is to construct two separate loss functions with differing weights. One loss function is only for learning the numerical stencil like FDM where the other one is a typical data driven loss function. This thesis is only concerned in creating ANN that would replicate FDM results for the Poisson's equation, meaning that only the former loss function is used. It was hypothesized in the beginning of this research that the network will outperform the FDM even if it learns from its generated data, which is something that will be elaborated later on.

Since the described Poisson's equation problem of using differing permeability domains was never practically solved, it was decided to start from 1D cases. Working in only one dimension made computations fast and the problem easier to work with. Instead of looking at a 2D plane of u distribution, a 1D "rod" is considered with differing permeabilities along its length.

Input for a conventional 1D PINN is usually a single neuron that represents any real x coordinate, where the output is also a single neuron that returns a u value at that particular x coordinate. This is great as the neural network can evaluate the u distribution at any point of the Ox domain, where numerical methods would require some fiddling with the discretization thus being not as flexible in that regard. The drawback is that it is not possible to use differing permeability domains when teaching a PINN meaning that it can only learn on one specific domain at a time which is hardcoded in the loss function.

Another point to note is the implementation of the boundary conditions. In this case Dirichlet BCs are considered, which define some fixed u values at the both ends of the domain. As for the permeability domain itself - a fixed one was used for training all of the PINN variations (figure 2.5). For this, a very simple architecture was used containing only one dense layer with 32 neurons (figure 2.6).

In 1D case, the Poisson's problem would translate to:

$$\begin{gathered} \frac{\partial^2 u}{\partial x^2} \cdot \kappa(x) = 0 \\ u(0) = \mathfrak{U}_0,\ u(1) = \mathfrak{U}_1 \end{gathered} \tag{2.44}$$

Idea was to train the neural network $NN(x)$ such that:

$$NN(x) \approx u(x) \tag{2.45}$$

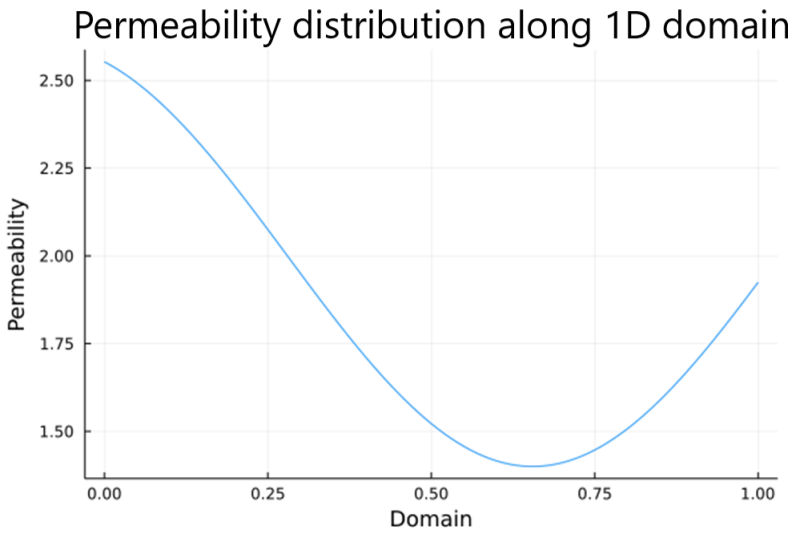


Figure 2.5: Permeability distribution underneath the u solutions of the 1:32:1 PINN.

Putting equation 2.45 into equation 2.44 yields:

$$\frac{\partial^2 NN}{\partial x^2} \cdot \kappa(x) = 0 \tag{2.46}$$

Since the expression on the left has to be zero, the network can actually try to minimize its square. This means that this expression in itself can be a loss function:

$$Loss = \left(\frac{\partial^2 NN}{\partial x^2} \cdot \kappa(x) \right)^2 = 0 \tag{2.47}$$

The second derivative term can be discretized using central differences equation 2.10, giving:

$$Loss = E\left[\left(\frac{NN(x_{i-1})\kappa(x_{i-1}) - 2NN(x_i)\kappa(x_i) + NN(x_{i+1})\kappa(x_{i+1})}{\Delta x_i^2} \right)^2 \right] \tag{2.48}$$

The most straightforward way of implementing boundary conditions is to put them directly inside of the loss function as such:

$$\begin{aligned} Loss = E\left[\left(\frac{NN(x_{i-1})\kappa(x_{i-1}) - 2NN(x_i)\kappa(x_i) + NN(x_{i+1})\kappa(x_{i+1})}{\Delta x_i^2} \right)^2 \right] \\ +(NN(0) - \mathfrak{U}_0)^2 + (NN(1) - \mathfrak{U}_1)^2 \end{aligned} \tag{2.49}$$

However, this way the neural network becomes much more unstable either blowing up the solution to extremely high values or disobeying BCs and falling in a local minimum producing poor results (figure 2.7).

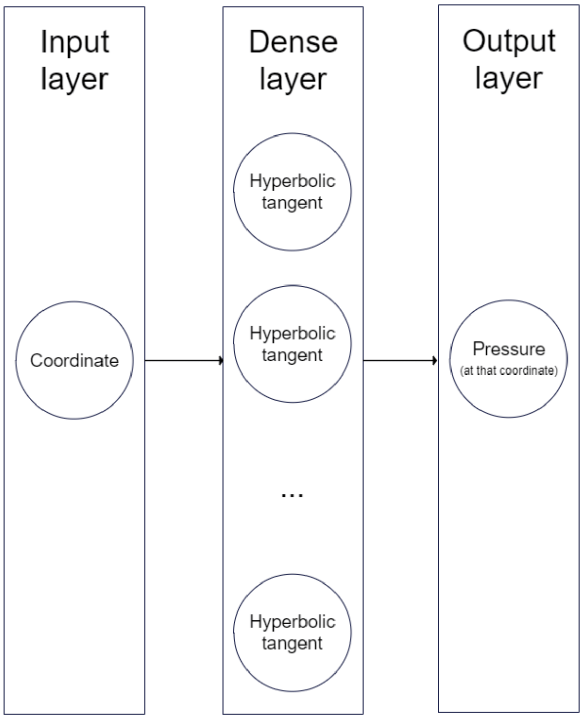


Figure 2.6: Architecture of a simple feed forward network with one dense layer consisting out of 32 hidden neurons (1:32:1). This network was used for the classic PINN formulation for the Poisson equation early in the thesis.

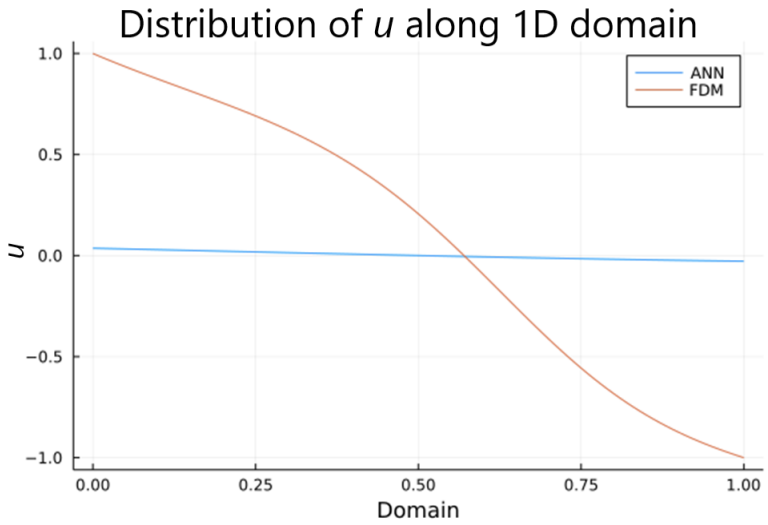


Figure 2.7: Distribution of u obtained from FDM and 1:32:1 PINN with boundary conditions declared inside the loss function.

A flexible method of dealing with this is to artificially introduce weights $\lambda_0, \lambda_1 \approx 100$ on the boundary conditions:

$$Loss = E\left[\left(\frac{NN(x_{i-1})\kappa(x_{i-1}) - 2NN(x_i)\kappa(x_i) + NN(x_{i+1})\kappa(x_{i+1})}{\Delta x_i^2}\right)^2\right] \\ +\lambda_0(NN(0) - \mathfrak{U}_0)^2 + \lambda_1(NN(1) - \mathfrak{U}_1)^2 \tag{2.50}$$

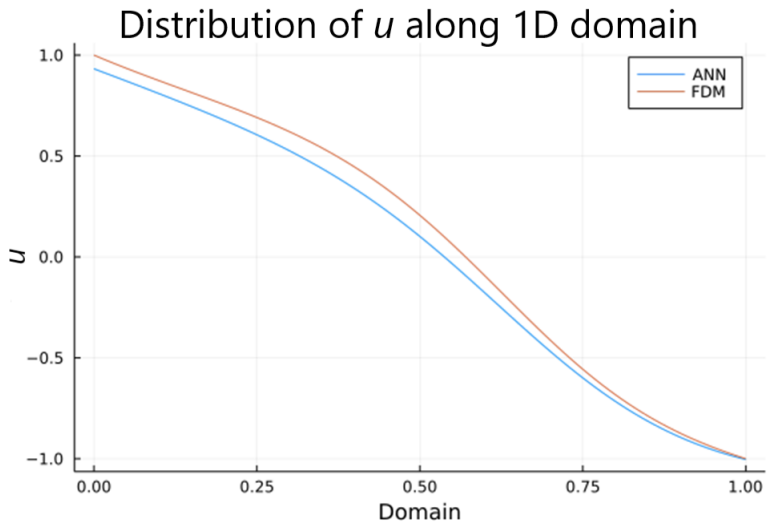


Figure 2.8: u distribution obtained from FDM and the same 1:32:1 PINN mentioned above with boundary conditions declared inside the loss function that had additional penalty weights assigned for their violation.

This makes the model behave much more decently (figure 2.8), however, it makes the computations even more unstable and the learning rate had to be reduced tenfold making the computations significantly slower (500 that iterations lasted a couple of seconds).

Even though the results were good, a better method is usually to hard-code the BCs in by using a dummy function that represents the u distribution, say g(x). Since ANNs can approximate any function, it is possible to arrange g(x) such that it would always satisfy the left boundary condition:

$$g(x) = \mathfrak{U}_0 + xNN(x) \tag{2.51}$$

When $x = 0$, then $g(0) = \mathfrak{U}_0$. Right BC would require similar approach and so a bit more sophisticated function was introduced:

$$g(x) = \mathfrak{U}_0 + (x^2 - x)NN(x) + x(\mathfrak{U}_1 - \mathfrak{U}_0) \tag{2.52}$$

Again, $g(0) = \mathfrak{U}_0$, but at the same time $g(1) = \mathfrak{U}_1$. The rest is taken care by $NN(x)$ itself which can take up any shape of any function. The resulting loss function is shown in equation 2.53.

$$Loss = E\left[\left(\frac{g(x_{i-1})\kappa(x_{i-1}) - 2g(x_i)\kappa(x_i) + g(x_{i+1})\kappa(x_{i+1})}{\Delta x_i^2}\right)^2\right] \tag{2.53}$$

Enormous flexibility of working with loss functions and feeding the network with data not as a coupled input-label data set, but inside the loss function itself was made convenient by Julia programming language [18].

The resulting trained neural network contained very high similarity with FDM that it tried to learn from (figure 2.9).

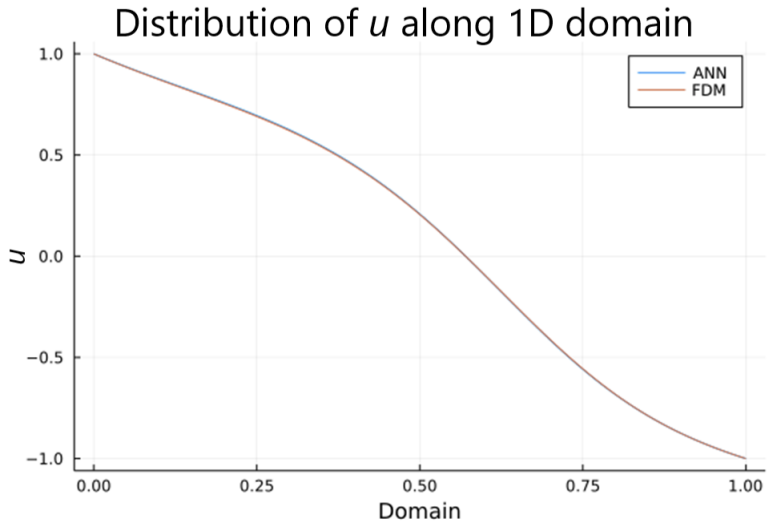


Figure 2.9: Distribution of u obtained from FDM and the same 1:32:1 PINN mentioned above with boundary conditions now hard-coded inside the u distribution function $g(x)$ that is being approximated.

This was impressive for the few seconds it needed (same 500 iterations) to train having such a simple network architecture. It could also take additional real world data to accustom for what the Poisson model lacked and have any Dirichlet condition defined. However, this network could only be trained with one permeability distribution at a time which is not ideal.

2.4.3. Coordinate-Discretized Neural Networks

The idea developed in this thesis is called Coordinate-Discretized Neural Networks (CDNNs) which suggests treating the input neurons as discrete coordinates themselves instead of having one single input neuron for coordinate insertion. That way it would have an input slot for every permeability value of the discretized domain. Such network would also produce an output of the same size (equal to the disrcretization node count n_x).

Before going any further, it is important to address the permeability vector $\vec{\kappa}$ generation and the approaches that have been taken. In the 1D examples above, a simple sinusoidal function was chosen to

represent permeability distribution throughout the domain, however a real world heterogeneity is usually much more complex. In order to push the network to its limits, a more sensible approach is to use smooth random noise methods like Perlin or Simplex to generate complicated permeability domains [30]. In this thesis a Gaussian Noise [39] that was glossed over with a Moving Mean algorithm [20] was used to produce the permeability domains.

In order to have an input slot for every permeability value of the discretized domain it was decided to look at the matrix form of FDM solution (equation 2.27), such that $NN(x)$ would behave as $\mathbf{u}$:

$$NN(\vec{\kappa}) \approx A \backslash (-\boldsymbol{\phi}) \tag{2.54}$$

Where on the right hand side is a solved u distribution by FDM $\mathbf{u_{FDM}}$. This will act as the regression "labels" for the CDNN such that:

$$NN(\vec{\kappa}) - \mathbf{u}_{FDM} = 0 \tag{2.55}$$

This can be used in creating a loss function:

$$Loss = E(\|\mathbf{u}_{FDM} - NN(\vec{\kappa})\|) \tag{2.56}$$

Symbol $\|\cdot\|$ is a norm which squares each element of the vector inside and adds them together. The expected value $E(\cdot)$ can be omitted and the optimization would not change, but it gives a better understanding of the error for different discretizations. However, since the problem is computationally expensive - computation of the mean was excluded.

The CDNN algorithm was developed so that it mimics the FDM method. Idea bbeing that this way CDNN would behave like FDM unsupervised, which implies that you only have to provide $\vec{\kappa}$ data for it to teach itself, which was the aim of the thesis.

Since the input and output layers were extended, it was decided to augment the architecture of the network and an additional 32 neuron hidden dense layer was added (figure 2.10).

While the results were mostly satisfactory, the model sometimes failed to keep up with steep "hills" and "valleys" on certain permeability distributions as well as made the solution very rough (figure 2.11).

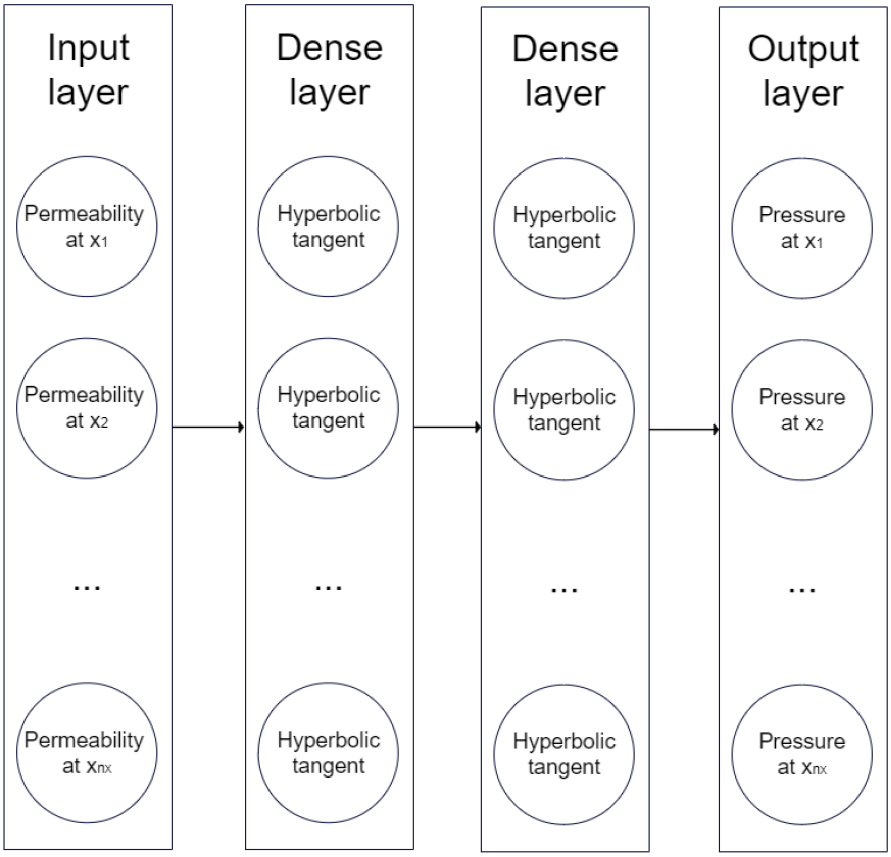


Figure 2.10: Architecture of a bit more complex feed forward network with two dense layers both consisting out of 32 hidden neurons $(n_x : 32 : 32 : n_x)$. This model was used for feeding the network permeabilities at each discretized point in the domain while getting the u values as an output.

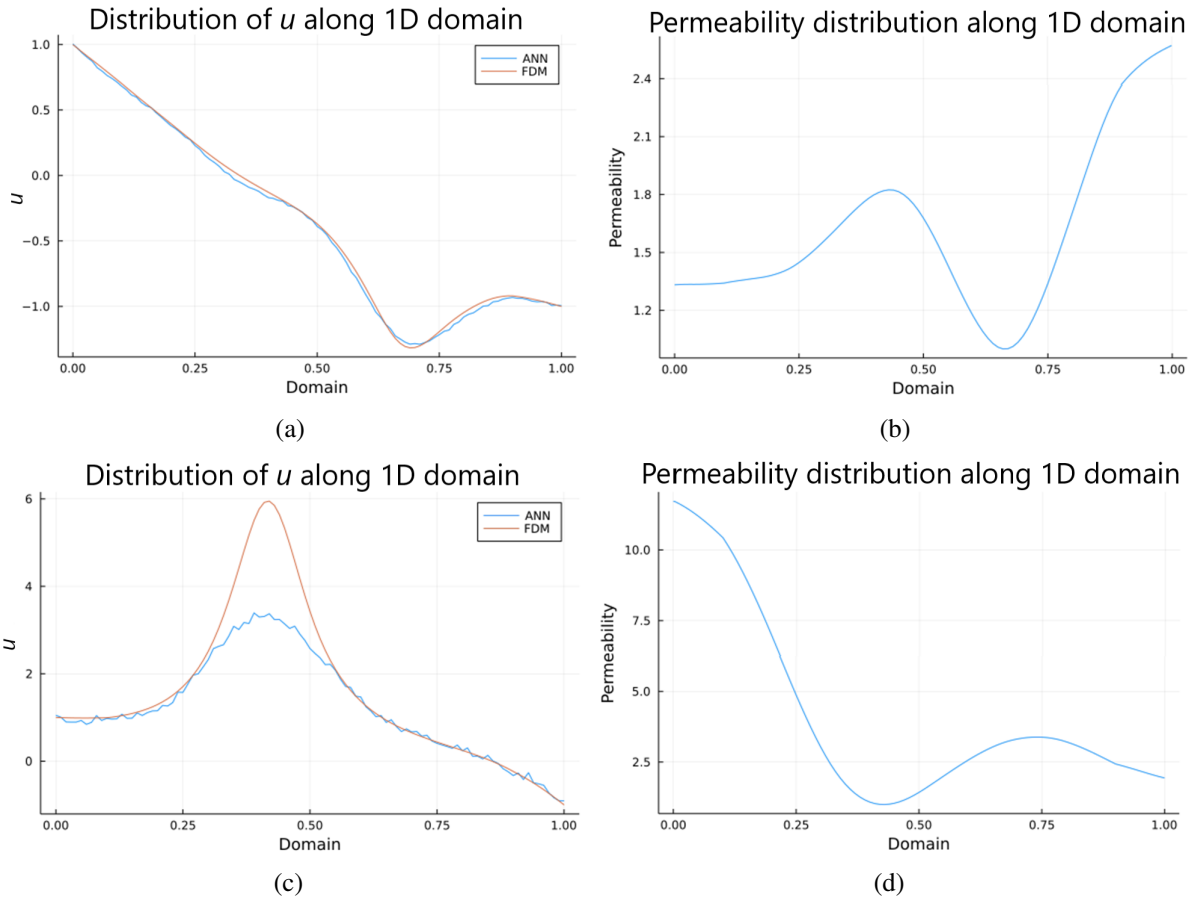


Figure 2.11: (a) Distribution of u obtained using FDM as well as $n_x : 32 : 32 : n_x$ CDNN whose boundary conditions are declared inside the loss function. (b) Corresponding permeability distribution. (c) Distribution of u obtained using FDM as well as the same CDNN. This subfigure depicts CDNN failing to replicate the FDM solution (case with the highest loss). (d) Corresponding permeability distribution.

This is when it was decided to implement convolutions in order to help the network adapt to a more sudden changes in permeability. Convolution operation can be approached as evaluating changes (which act as a derivative analog) of permeabilities at every point. This meant that the network could learn from shapes and edges rather then intricately interconnected raw data. Network architectures were kept simple but sensible nevertheless (figure 2.12).

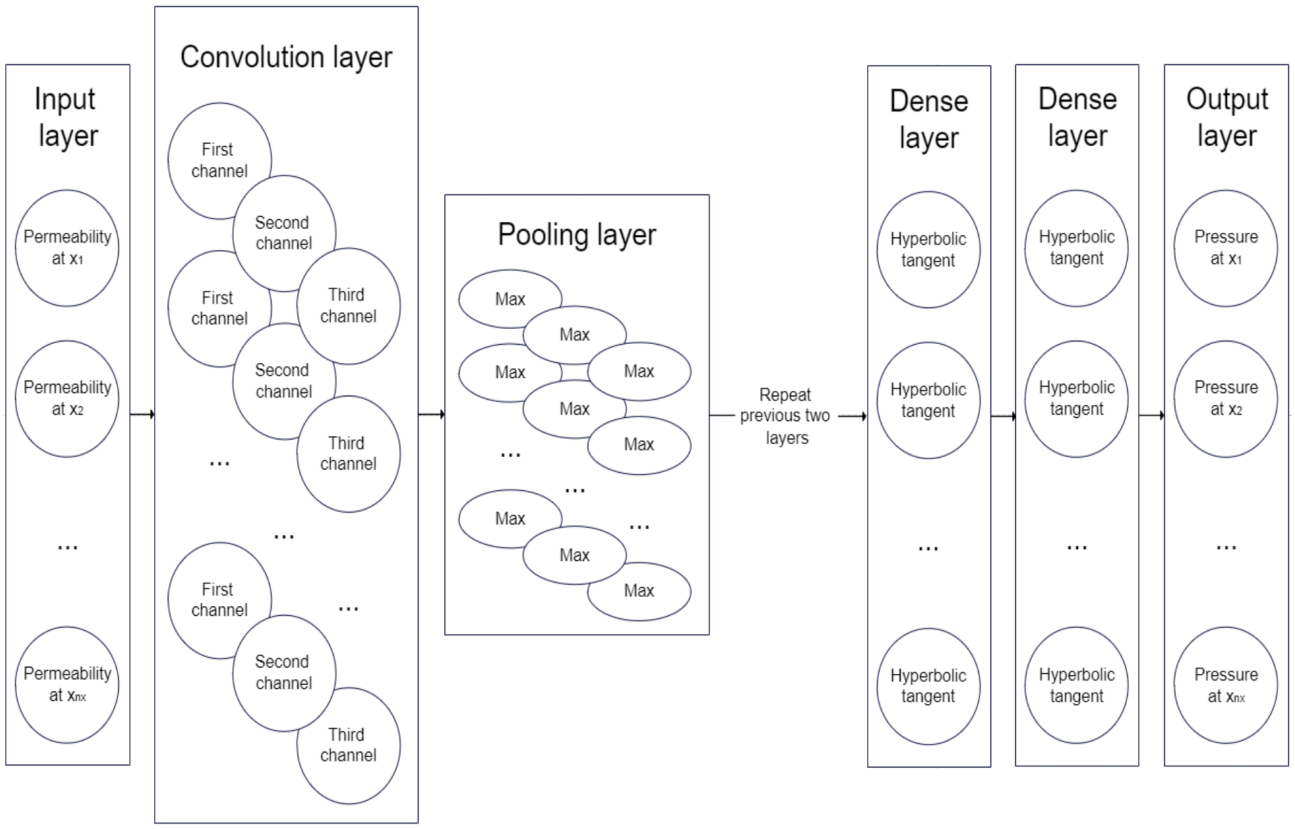
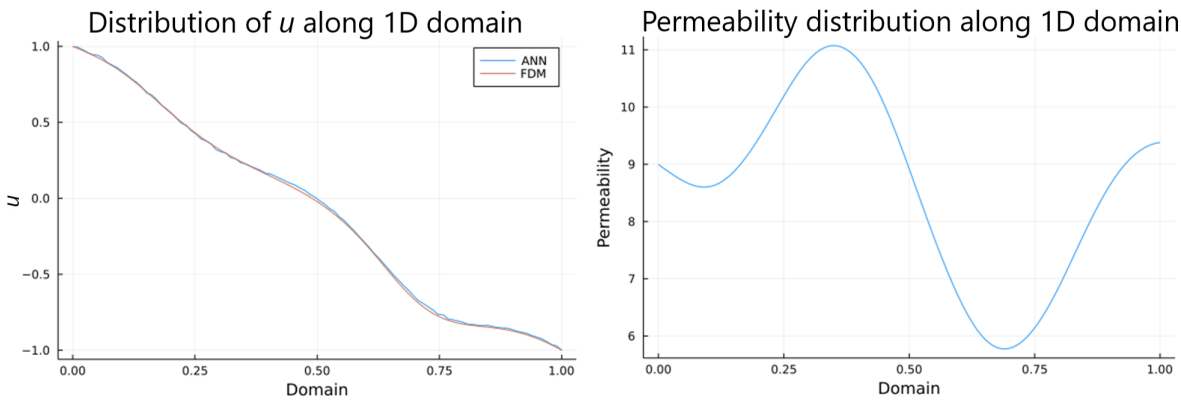


Figure 2.12: Architecture of a convolutional neural network (CNN) used with two convolutional layers that triple the channels for feature extraction, two max pooling layers after each convolution for reduction of the amount of neurons by two and two dense layers $\frac{9}{4}\ n_x : 32$ for loss minimization.

Since the results are only for experimentation purposes iteration count and the time taken were omitted. As the network structures become more intricate and input data more complex, CDNN will require more iterations, however the goal is to demonstrate the results received and compare them in a somewhat considerate manner as there is no real sensible way to account for the progression of complexity.

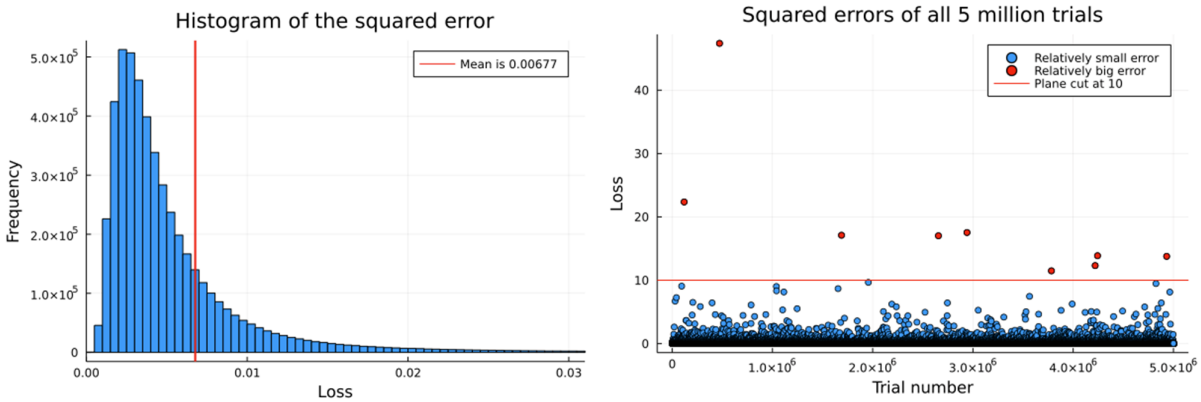
With the new architecture resulting u distributions were much smoother and more consistent overall. This held true even with sharper fronts in permeability domain (figure 2.13).



(a) (b)

Figure 2.13: (a) Distribution of u obtained using FDM as well as convolutional CDNN (a common case). (b) Corresponding permeability distribution.

A histogram was produced showing how much of the training samples every bin of squared errors received (figure 2.14). Error reaches all the way up until ~ 47, which is very high compared to the mean. In the same figure a statistic of the losses of all 5 million trials used in training is presented as well.

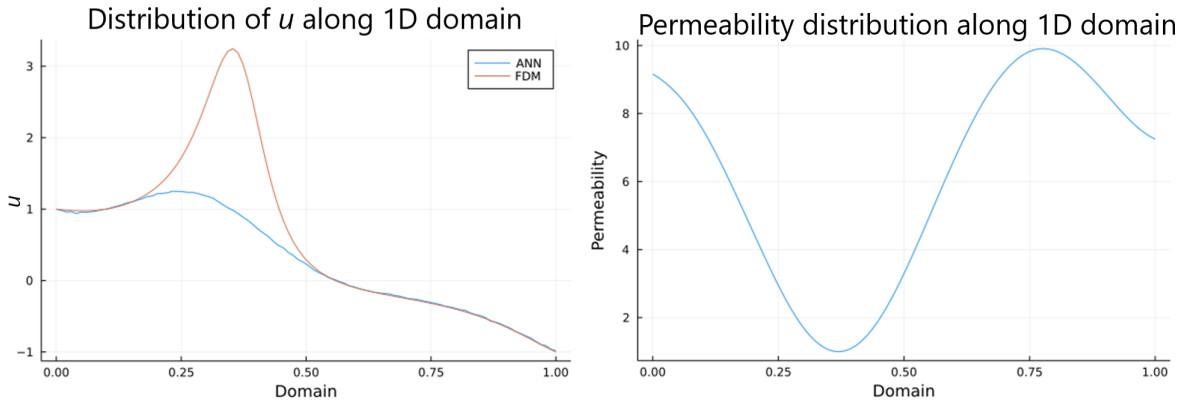


(a) (b)

Figure 2.14: (a) The histogram shows how much of the training samples every bin of squared errors received. Ox axis goes all the way up to ~ 47, but the frequencies become vanishingly small quite soon. (b) The graph shows every squared error of every trial. Since most of the errors are clumped at around the value of 0.003 - a high density of points can be seen at the bottom of the graph. The red line indicates the level of the errors that is phenomenally high, so the red points are a subject of a further investigation.

Out of 5 million trials only a handful of them had a "significantly" large loss. After separating the worst cases and checking how were they different compared to the whole population an interesting observation arose - they were the cases where FDM failed to depict an accurate solution giving what is called a non-monotonous solution (figure 2.15). These solutions are numerical artefacts created by

the method's inability to deal under certain circumstances leading to such u values that $u \notin [\mathfrak{U}_0, \mathfrak{U}_1]$. After an enormous amount of training performed by CDNN, it managed to learn from the majority of cases that u values throughout the domain must be bounded by the values at the boundaries.



(a) (b)

Figure 2.15: (a) Distribution of u obtained from FDM and CDNN (case with the highest loss). (b) Corresponding permeability distribution.

3. Results

Results were achieved by implementing and then comparing FDM with FVM solutions, drawing insights on what a better method for the particular problem is and then training CDNN on the chosen method's generated data. Two main cases were considered: layered and heterogeneous domains. This led to a training of two separate neural networks that have identical architecture for each of the domain type. While this approach is not tailoring for each of the domain type specific needs, it certainly is simpler for modelling purposes and enough to draw conclusions for the thesis. For possible future advancement ideas see the discussion chapter.

3.1. Layering

The main focus in this section was to observe the ability of FDM, FVM and then CDNN to cross a layered domain. Such domain is a merge of two separate sub-domains of different permeabilities that create a sharp and distinguishable boundary in between. An example domain chosen to illustrate the numerical method differences is shown in figure 3.1.

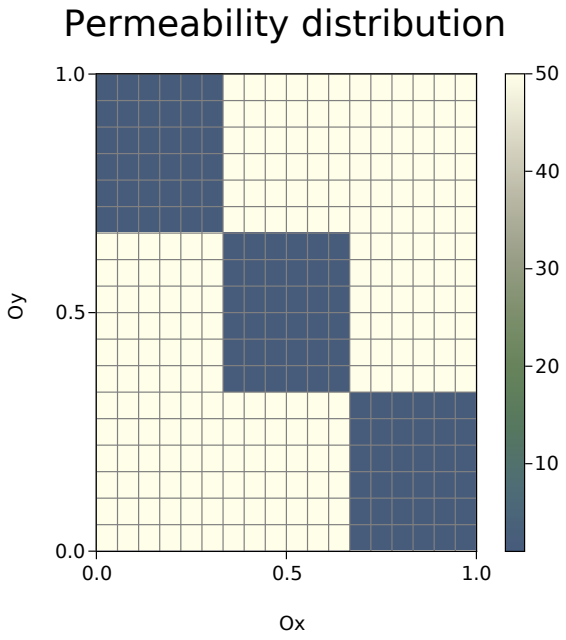
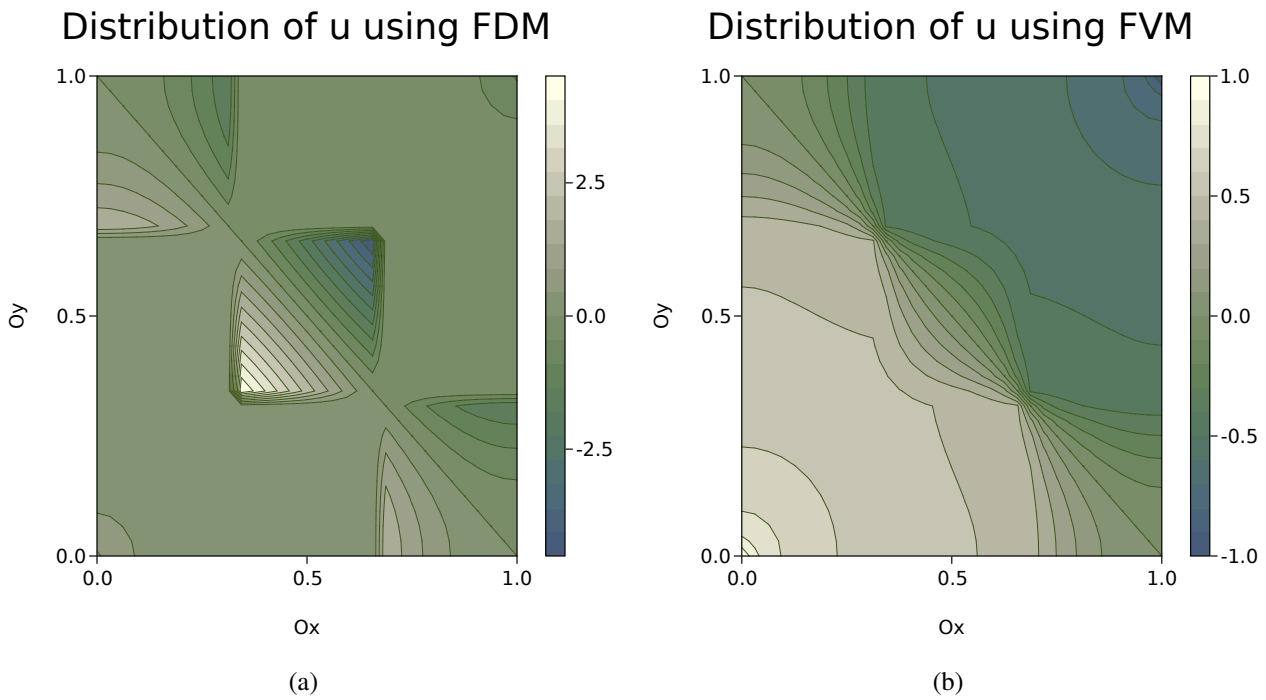


Figure 3.1: A bi-valued permeability space with sharp boundaries (tiled domain). Dark area corresponds to low permeability where light tiles mark high permeability areas.



(a) (b)

Figure 3.2: Contour plot of u distribution crossing a sharp permeability boundary. Solution u of the Poisson's equation (a) using FDM (b) using FVM

Applying FDM in such case (figure 3.2) creates an unrealistic distribution of u as it acts more on differing permeability tiles separately than considering the global picture and recognizing the neighboring sub-domain. This is not present in FVM case, as introduction of the control volumes regulates the flux propagation even along the sharp boundary between the two sub-domains. This conservation of the flux is what allows the solution to remain smooth throughout all of the domain.

Since FDM is unable to adequately represent the distribution of u in 2D case, CDNN was trained on the FVM solely. All of the possible solutions were segmented in three main categories where the first one was looking at the overall distribution of u (figure 3.3). In these graphs the main point is to notice that CDNN manages to deal with large areas of uniformity in permeability data and understand what overall "color" should the graph undertake. The CDNN picture has a considerable amount of noise contamination and is something what should be dealt in the future. In theory, this could be reduced by implementing deeper network structures and more sophisticated network architectures.

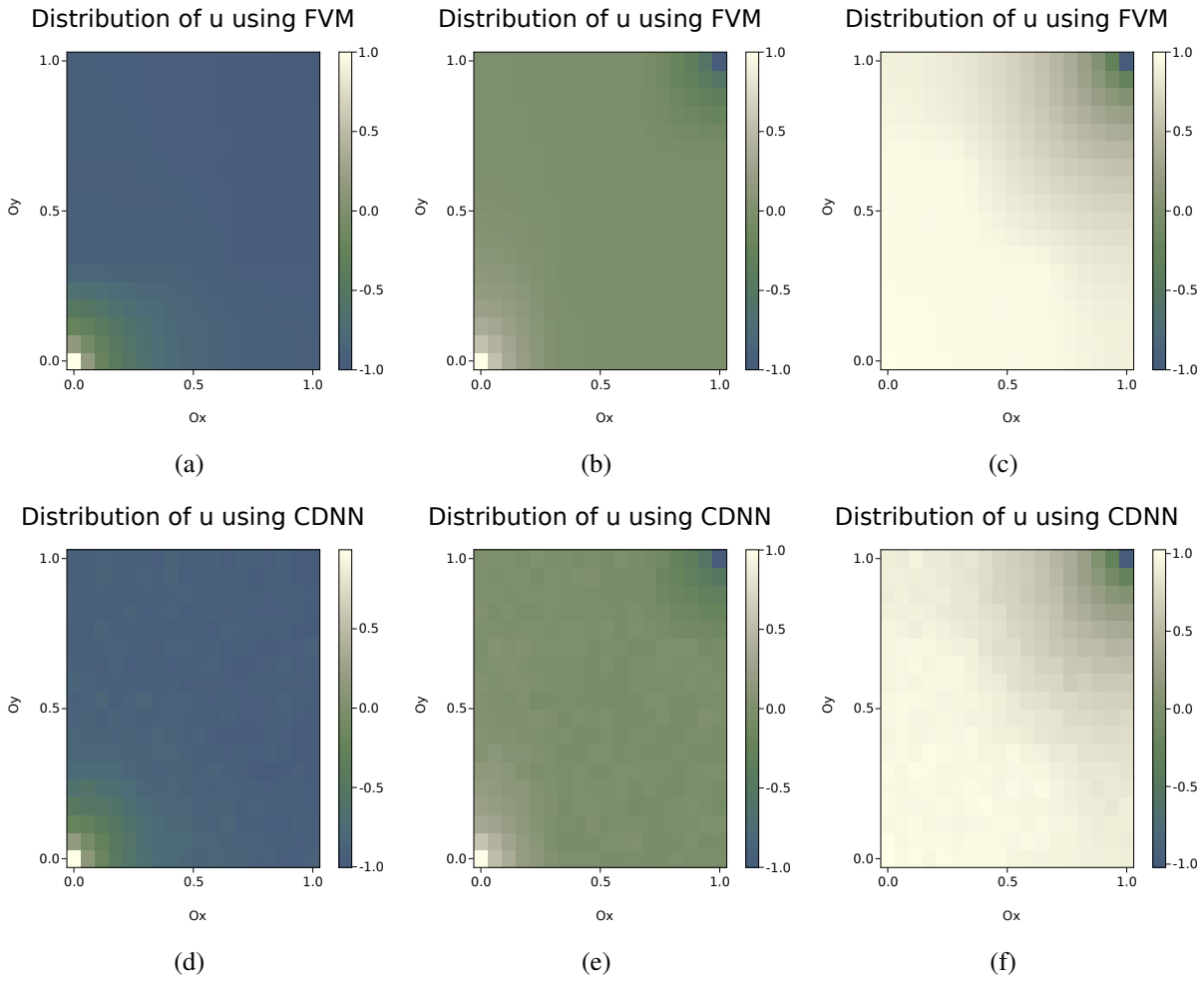
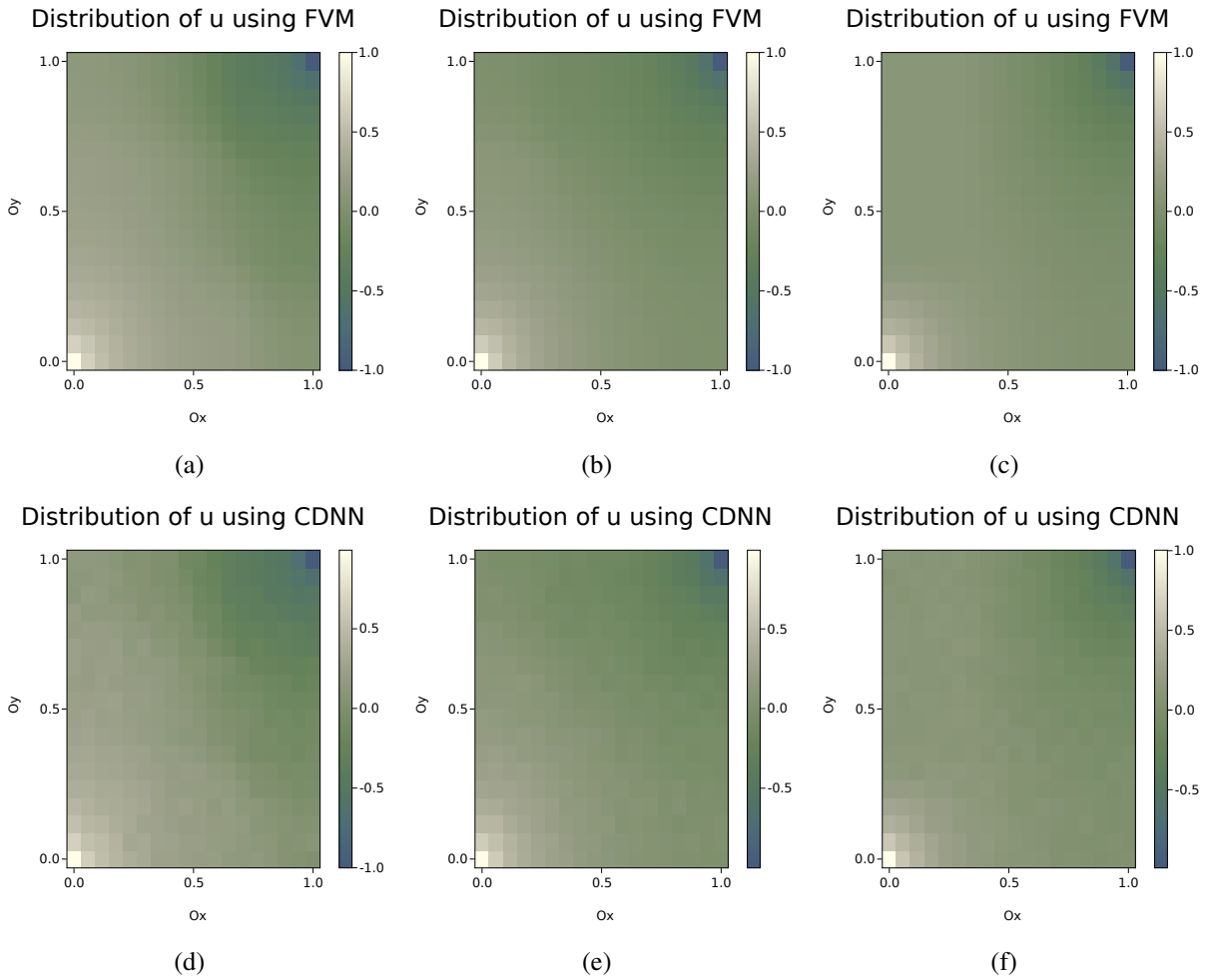


Figure 3.3: Graph depicting CDNN's ability to deal with overall distribution of u on a tiled permeability domain. (a)-(c) Solutions using FVM (d)-(f) Solutions using CDNN.

Figure 3.4, on the other hand, depicts how well CDNN reacts to very subtle changes of the species u distribution. Even FVM solution varies only a little going from the leftmost image towards the right one, however analogous shift of colours can be spotted in the CDNN images as well. This shows the neural network's ability to replicate finer details on top of aforementioned overall u distribution.



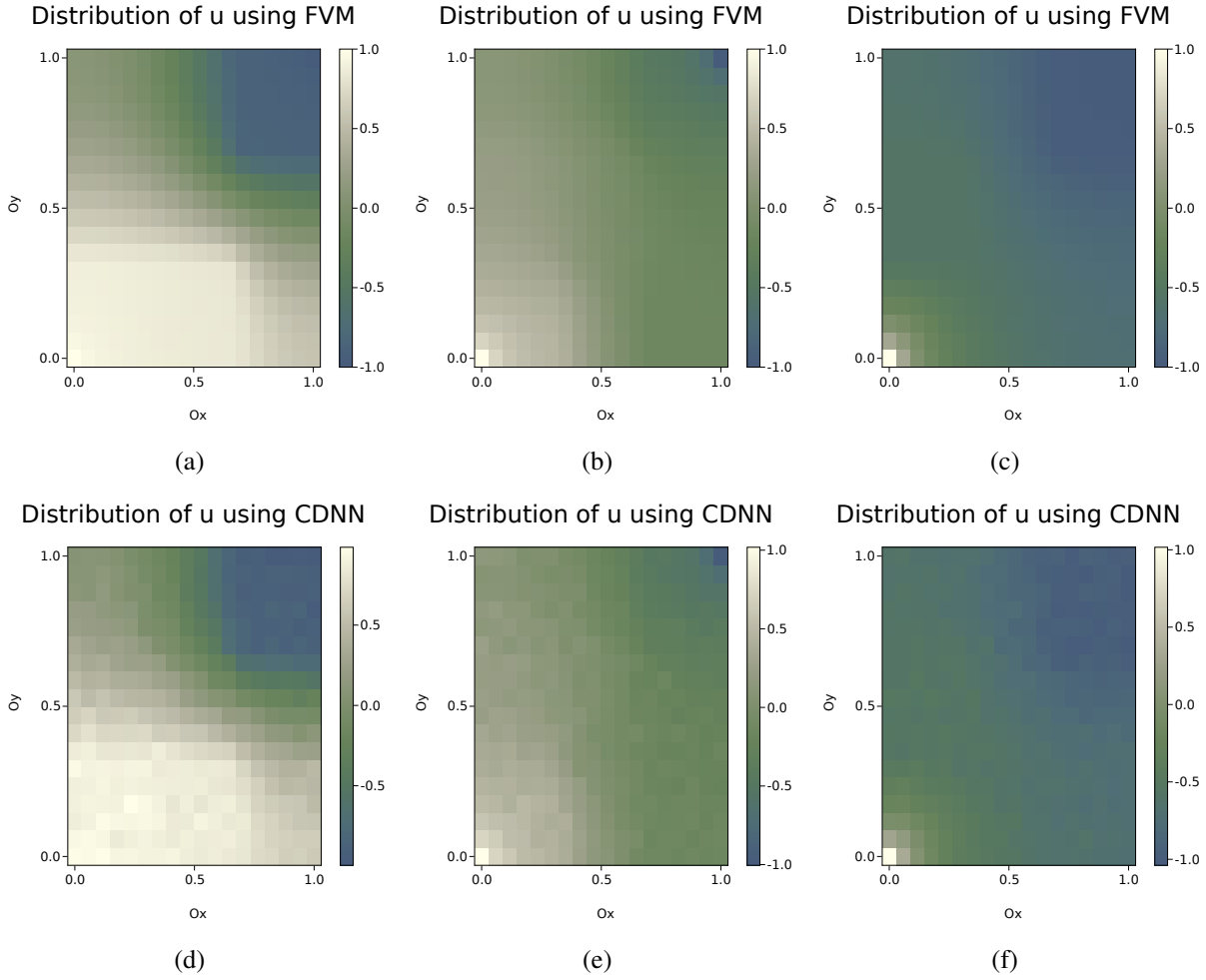
(a) (b) (c)

(d) (e) (f)

Figure 3.4: Graph depicting CDNN's ability to mimic FVM with more subtle u distribution changes on a tiled permeability domain. (a)-(c) Solutions using FVM (d)-(f) Solutions using CDNN.

The last series of images (figure 3.5) represent the toughest cases with a lot of activity that can be seen - there being darker spots as well as lighter regions, sharper edges and shapes emerging from the underlying permeability domain, also uniform regions as well as sharp transitions in the same domain. Nevertheless, CDNN managed to keep up with these obstacles fairly well - it is easy to distinguish which CDNN image corresponds with which FVM solution to the point where even the finer details are somewhat matching.

The obvious issue is that the noise contamination is most severe with these cases in its amplitude *and* the amount. Some stochastic jumps in value break the desired uniformity of the solution as well as the conservation of the flux which can be reduced by implementing smoothing functions. While this approach would decrease the error of the solution, overall it is not desirable as the standard smoothing functions do not take into account the underlying permeability domain.



(a) (b) (c)

(d) (e) (f)

Figure 3.5: Graph depicting CDNN's ability to deal with even more challenging distributions of u on a tiled permeability domain. (a)-(c) Solutions using FVM (d)-(f) Solutions using CDNN.

3.2. Heterogeneity

The main focus in this section is to observe the ability of FDM, FVM and then CDNN to deal with a strong heterogeneity of the domain which is commonly present in real world problems. In order to clearly illustrate the effects it has on both methods, a Gaussian Noise coupled with Moving-Mean algorithm were used to generate domains, which would mimic the ones commonly found in nature (figure 3.6).

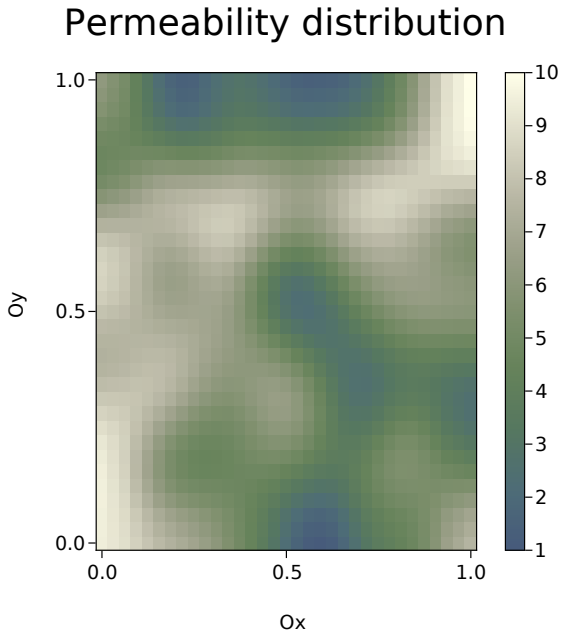


Figure 3.6: A highly heterogeneous permeability space generated by Gaussian noise function coupled with Moving-mean algorithm. It somewhat resembles the permeability structure of what can be found in the brain, Earth's subsurface, various porous media, etc.

While these domains are unpredictable, they do have a certain "frequency" feel to them - a general idea of a number of wave-like structures being repeated across the domain. This is one of the hyper-parameters that can be controlled while generating such permeability domains.

The aforementioned waves have ever-changing amplitudes and even frequencies, which makes this problem even more difficult. Layered as well as highly heterogeneous media are discussed in this thesis, however there are more types of domains to be considered like cracked surfaces and non-circular heterogeneity.

The solution of FDM (figure 3.7) over a heterogeneous domain (figure 3.6) gets strongly impacted by the discontinuities of the underlying media which is favored only to an extent as the actual distribution of u should be smoother (based on the real world observations). The permeability values range only from 1 to 10 (where on the tiled domain the range was $[1, 50]$) as FDM not only starts producing non-monotonous solutions, but breaks down completely blowing up the solution at wider permeability intervals. Even with this example domain it is visible, that FDM produces u values lower than -1. FVM, on the other hand, provides a much more consistent picture of how u should really be distributed along the given domain and is bounded by interval $[-1, 1]$.

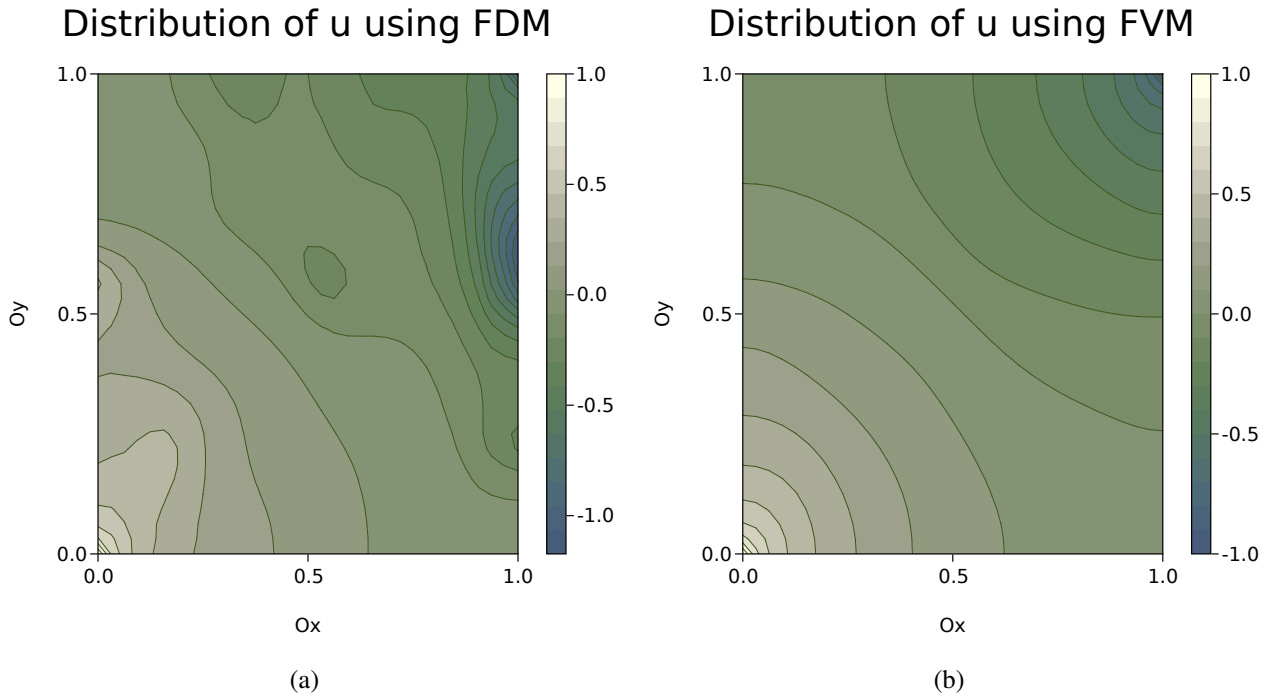
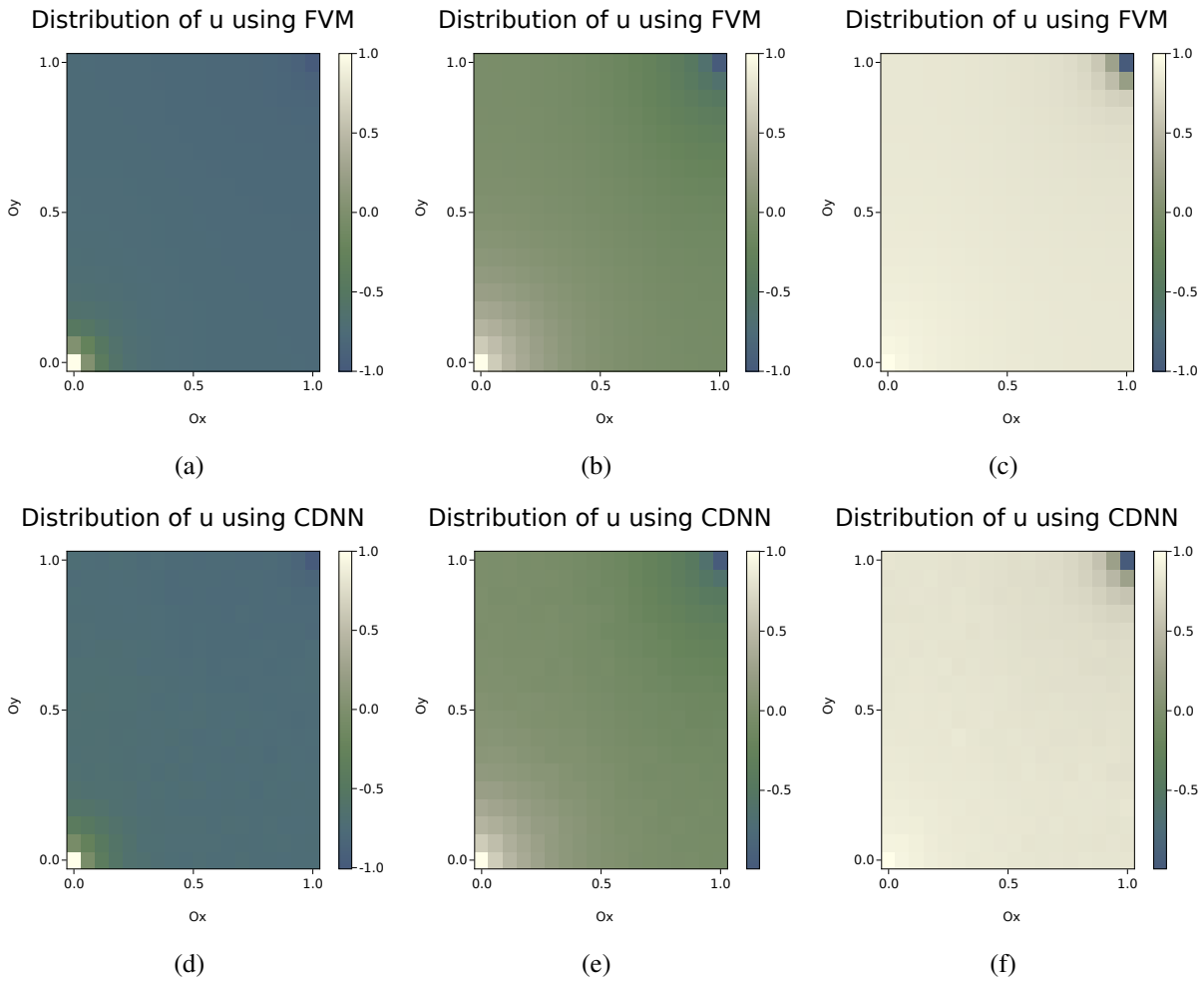


Figure 3.7: Contour plot of u distribution over a highly heterogeneous domain. Solution u of the Poisson's equation (a) using FDM (b) using FVM

Because of the aforementioned reasons, CDNN was again trained solely on the FVM. The permeability domain can again range from 1 to 50, creating more interesting solutions to showcase. The first figure 3.8 (analogously to the cases in the previous section) depicts the overall distribution of u. This time around images can be almost entirely coloured in the same colour given favourable permeability distributions as the permeability change, be it not as sharp - is much finer.

As for the heterogeneous domain, the CDNN images have unexpectedly low amounts of noise contamination, which could be because the network training was much longer (overnight) than in the tiled case (40 minutes), resulting in a similar amount of noise. I believe that training the network any longer would have not produced any significant changes to the quality of the solution and that any further improvements should be left to the design of a more effective network architecture or on the tweaks in the problem model approach in general.

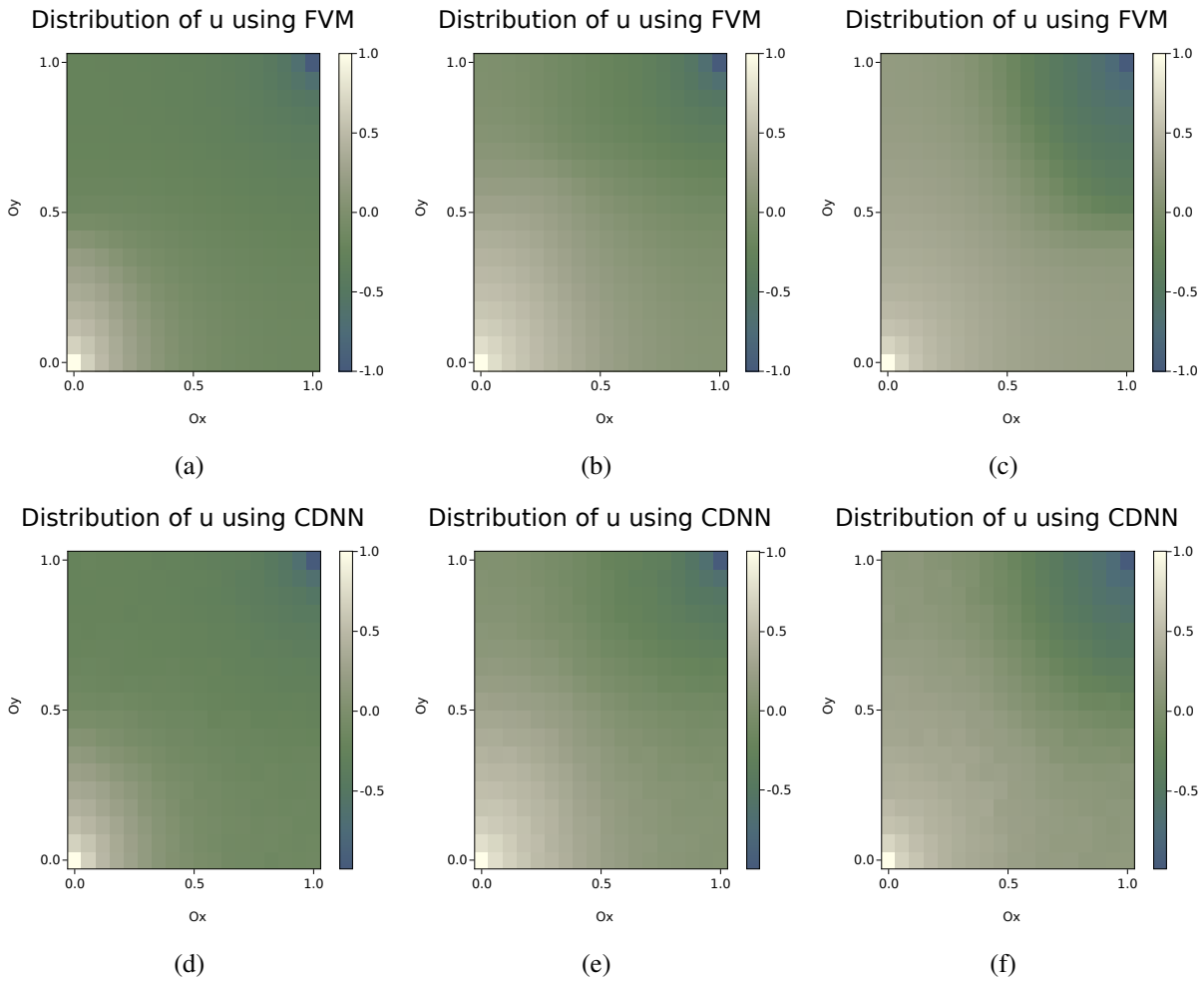


(a) (b) (c)

(d) (e) (f)

Figure 3.8: Graph depicting CDNN's ability to deal with overall distribution of u on a highly heterogeneous permeability domain. (a)-(c) Solutions using FVM (d)-(f) Solutions using CDNN.

In the tiled permeability case there was a graph for smoother, more gentle solutions as well as a graph for edgy, shapely solutions. However, since the generated heterogeneity was smooth, those two cases on heterogeneous media became a merge of the two and produced solutions that are represented by images in figure 3.9. These images depict some of the most challenging cases, where the CDNN had trouble replicating FVM. Even though it is possible to distinguish between the graphs, CDNN tends to smooth out areas where the FVM considered to have sharper boundaries.



(a) (b) (c)

(d) (e) (f)

Figure 3.9: Graph depicting CDNN's ability to deal with some challenging distributions of u on a highly heterogeneous permeability domain. (a)-(c) Solutions using FVM (d)-(f) Solutions using CDNN.

3.3. Error Overview

A bar chart 3.10 depicts an average of each case Mean Squared Error (MSE). This accuracy evaluation is slightly deceiving - since the average of errors is so low, it could be thought that the CDNN performs better than it actually does. It is extremely important to investigate the influential observations on the statistical properties first in order not to be misguided by such summary statistic [3].

This can be seen very clearly in effect, where the "subtle" column of the tiled permeability domain is much lower than the other ones. Since subtle distributions of u deviate so little from the overall distribution, the average error estimate is relatively lower then of other types. Heterogeneous case has no subtle case, but a merge of both subtle and challenging cases, as discussed before. This led to using same error for both cases, thus having a higher error compared to tiled permeability.

Overall, there are no significant error differences between the two cases, which means, either the CDNN architecture is not optimized for any of the permeability distributions, or that the architecture does not make any difference for different permeability types. Further investigation is required.

Figure 3.10: Bar chart depicting an average of each u distribution case MSE between FVM and CDNN. For each permeability (tiled/heterogeneous) there were three distinct u distribution types - a type for somewhat uniform u distribution over the domain where one colour dominates over all, a type for more subtle shapes in the solution and a type for challenging cases.

Conclusions

The goal of solving the Poisson's Equation by implementing FDM, FVM and ANN on a layered and highly heterogeneous media was completed to the extent where the code was written and every method was derived, constructed and tested properly. This was elaborated more in detail in chapter 2. All of the parameters as well as domain distributions were picked carefully out of hundreds of cases to illustrate possible drawbacks of the methods, so that the reader would not be left mislead and indifferent. For example FDM inability to deal with sudden high permeability fluctuations and CDNN's noise contamination with specific challenging permeability distributions.

The construction and motivation behind using convolutions as well as specific neural network architecture can be found in chapter 2 section 3. Some of the hyper-parameters for the proposed CDNN were chosen purely by trial and error and could be optimized for a much better performance. The construction of the network was motivated by PINN approach as well as CNNs and was solved for a 1D domain case where it was illustrated that CDNN can learn to avoid non-monotonous solutions even when learning from methods such as FDM that produce such solutions.

It was also shown in chapter 3 section 1, that between the two numerical methods, FVM has the advantage when dealing with sharp edges in-between the differing permeabilities of neighboring subdomains by creating smooth solutions. There were three main cases of the distribution of u that could have manifested on a tiled permeability domain showing an overall spread of colours along the domain, a more subtle change of values and some challenging cases in general. A constructed CDNN had the ability to mimic such FVM solutions albeit being noisy.

Moreover, in chapter 3 section 2 it was shown that FVM dealt with high heterogeneity much more consistently than FDM did, leading to the training of the network again being solely on FVM. The resulting CDNN had trouble capturing more subtle areas of the species u distribution, but managed to deal with the overall spread of values to produce results that were worth attention.

I strongly believe that there could be real benefits in using the proposed CDNN in solving Forward problem and even Inverse Problems in the future.

Discussion

There are many issues with the proposed ANN and the presented work should only be taken as a naive attempt at a complicated problem. One of the most obvious drawbacks of the method is the inability to specify the sources and sinks. In paper [19] researchers implemented additional two input nodes for specific x and y coordinates and had only two output nodes for species u value. That way the approach becomes extremely similar to PINN in the way that the boundary conditions as well as sources and sinks could be specified inside of the loss function. This approach also reduces the number of output nodes significantly which helps to deal with the "curse of dimensionality" in ANN construction - another drawback of CDNN that is highly apparent in this thesis. However, there is no clear way to separate the permeability and coordinate input and they serve very different purposes inside of the ANN. This thesis was my attempt to help ANN train more intuitively by throwing the coordinate inputs out. This unfortunately comes to a higher cost especially if the problem scales to finer grids or worse - higher dimensions.

Some of the positive sides with the proposed CDNN method is that the real world data can also be very easily included and even enforced, which would lead to a more profound model that is able to learn the non-linearities and additional factors (not considered by Poisson's equation model) present in the real world. Another positive is that after sacrificing some time for offline network training, the online performance outshines any numerical method without competition. Computational swiftness on the go is highly desired in industry and is one of the most valued factors. One of the drawbacks is the pre-set discretization of the grid, which dictates how many nodes can the network be fed for it to return the solution. This is a minor inconvenience though, as permeability re-scaling is also a huge area for research [34, 14] which I and the thesis supervisor are also interested in and most likely will be publishing AI assisted re-scaling paper soon.

There are some future plan to conduct a more in-depth study at the improvement on the network architecture as in create deeper structures, experiment with broader range of activation functions, gradient descent parameters, batching, etc. Also, every permeability domain type would require different network architecture - using differing frequency oscillations in heterogeneous permeability values and simultaneously having layered structures inside the domain would require network branching to be implemented. This could also be considered for investigating the multi-scale nature of the problem.

Next strides for the research itself would be to conduct a method stability as well as sensitivity analysis - see how reliable is the CDNN approach, does it produce the adequate results with any given permeability domain and finally merge layering with heterogeneity together.

There are also some future ideas on the improvement on the methodology itself. First one is an extension of the proposed method using a very specific way to convolve residuals of the FVM solution to get a unique solution for the convolved ANN output by using layer freezing. This approach would tailor exactly to the idea behind the Poisson's equation and thus leads me to believe that it would help to reduce errors of ANN.

Another idea is to scratch this methodology entirely and create an ANN as an analog to inverse matrix A. That way it would be possible to feed the network some driving function ϕ and let ANN would do the heavy lifting that is done by methods like Gaussian elimination. This would, again, increase the online performance significantly compared to numerical methods and could have sinks and sources specified, however, this approach would require u and κ to be separable, meaning that only FDM could

be used on training such ANN.

There are even more ideas like implementing Poisson's equation independent of any numerical scheme by minimising the difference of differences by using dual number automatic differentiation, but these topics are very weakly formulated and only being considered as of the time this thesis is being written. In the last few years there has been a noticeable increase of PINN related research which requires a broader literature review and new approaches.

Bibliography

[1] Ivar Aavatsmark. An introduction to multipoint flux approximations for quadrilateral grids. *Computational Geosciences*, 6:405–432, 09 2002.

[2] Dragan Aleksendrić and Pierpaolo Carlone. *Soft computing techniques*, pages 39–60. 12 2015.

[3] F.J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[4] Kassem Awada, David Jackson, Stephen Baumann, Jeffery Williams, Donald Wilton, Patrick Fink, and Brian Prasky. Effect of conductivity uncertainties and modeling errors on eeg source localization using a 2-d model. *IEEE transactions on bio-medical engineering*, 45:1135–45, 10 1998.

[5] Rimantas Barauskas and Ausra Abraitiene. Multi-resolution finite element models for simulation of the ballistic impact on non-crimped composite fabric packages. *Composite Structures*, 104:215–229, 2013.

[6] Rimantas Barauskas, Antanas Gulbinas, Tomas Vanagas, and Giedrius Barauskas. Finite element modeling of cooled-tip probe radiofrequency ablation processes in liver tissue. *Computers in Biology and Medicine*, 38(6):694–708, 2008.

[7] Tian Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 06 2018.

[8] Jinhyun Choo and Sanghyun Lee. Enriched galerkin finite elements for coupled poromechanics with local mass conservation. *Computer Methods in Applied Mechanics and Engineering*, 341:311–332, 2018.

[9] A U.S. Department of Energy National Laboratory Managed by the University of California Diana Swantek. New berkeley lab subsurface sfa 2.0 project explores uncharted environmental frontier of subsurface ecogenomics, 2013.

[10] Louis J. Durlofsky. A triangle based mixed finite element—finite volume technique for modeling two phase flow through porous media. *Journal of Computational Physics*, 105(2):252–266, 1993.

[11] Lennart Edsberg. *Introduction to Computation and Modeling for Differential Equations*. Wiley-Interscience, USA, 2008.

[12] Michael Edwards. Unstructured, control-volume distributed, full-tensor finite-volume schemes with flow based grids. *Computational Geosciences*, 6:433–452, 09 2002.

[13] G. Eigestad and Runhild Klausen. On the convergence of the multi-point flux approximation o-method: Numerical experiments for discontinuous permeability. *Numerical Methods for Partial Differential Equations*, 21:1079 – 1098, 11 2005.

[14] C. Farmer. Upscaling: A review. *International Journal for Numerical Methods in Fluids*, 40:63 – 78, 09 2002.

[15] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. pages 481–490, 08 2016.

[16] M.T. Hagan, H.B. Demuth, and M.H. Beale. Neural network design. *Pws Pub, Boston*, 2:2–14, 01 1996.

[17] Thomas J.R. Hughes, Gerald Engel, Luca Mazzei, and Mats G. Larson. The continuous galerkin method is locally conservative. *Journal of Computational Physics*, 163(2):467–488, 2000.

[18] Viral B. Shah Jeff Bezanson, Alan Edelman and Stefan Karpinski. The julia programming language, 2009.

[19] Sharmila Karumuri, Rohit Tripathy, Ilias Bilionis, and Jitesh Panchal. Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *Journal of Computational Physics*, 404:109120, 11 2019.

[20] Brenhin Keller. Nanstatistics.jl, 2021.

[21] Phong Le, Nam Mai-Duy, Thanh Tran-Cong, and Graham Baker. A cartesian-grid collocation technique with integrated radial basis functions for mixed boundary value problems. *International Journal for Numerical Methods in Engineering*, 82:435 – 463, 01 2009.

[22] Gildas Marin, Christophe Guérin, Sylvain Baillet, Line Garnero, and Gerard Meunier. Influence of skull anisotropy for the forward and inverse problem in eeg: Simulation studies using fem on realistic head models. *Human Brain Mapping*, 6:250 – 269, 02 1998.

[23] Inc. MedicineNet. Medical illustrations: Picture of brain layers, 2010.

[24] Science News. A 100-hour mri scan captured the most detailed look yet at a whole human brain, 2019.

[25] Mayur Pal. *Families of Control-Volume Distributed CVD(MPFA) Finite Volume Schemes for the Porous Medium Pressure Equation on Structured and Unstructured Grids*. PhD thesis, 09 2007.

[26] Mayur Pal. On application of machine learning method for history matching and forecasting of times series data from hydrocarbon recovery process using water flooding. *Petroleum Science and Technology*, 39:1–31, 07 2021.

[27] Mayur Pal and Michael Edwards. Non-linear flux-splitting schemes with imposed discrete maximum principle for elliptic equations with highly anisotropic coefficients. *International Journal for Numerical Methods in Fluids*, 66:299 – 323, 05 2011.

[28] Mayur Pal and Michael Edwards. The effects of control-volume distributed multi-point flux approximation (cvd-mpfa) on upscaling—a study using the cvd-mpfa schemes. *International Journal for Numerical Methods in Fluids*, 68:18 – 35, 01 2012.

[29] Suhas Patankar. Numerical heat transfer and fluid flow. *Series in Computational Methods in Mechanics and Thermal Sciences*, 67, 01 1980.

[30] Ken Perlin. *An image synthesizer*, pages 147–156. 07 1998.

[31] G Pruis, Brian Gilding, and M Peters. A comparison of different numerical methods for solving the forward problem in eeg and meg. *Physiological measurement*, 14 Suppl 4A:A1–9, 12 1993.

[32] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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, 2019.

[33] Gerald Recktenwald. The control-volume finite-difference approximation to the diffusion equation. 2014.

[34] Philippe Renard and G. Marsily. Calculating equivalent permeability: A review. *Advances in Water Resources*, 20:253–278, 10 1997.

[35] Thomas Russell and Mary Wheeler. 2. finite element and finite difference methods for continuous flows in porous media. *Frontiers in Applied Mathematics*, 1, 01 1983.

[36] Marc Thevenet, Olivier Bertrand, Francois Perrin, and Jacques Pernier. Finite element method for a realistic head model of electrical brain activities. pages 2024 – 2025, 12 1992.

[37] Maria Vasilyeva and Aleksey Tyrylgin. Machine learning for accelerating effective property prediction for poroelasticity problem in stochastic media, 10 2018.

[38] Marko Vauhkonen, Tanja Tarvainen, and Timo Lähivaara. *Inverse Problems*, pages 207–227. 07 2016.

[39] Felix Wechsler. Noise.jl, 2020.

[40] World Health Organization WHO. Epilepsy, 2022.

[41] Jinlong Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3:074602, 01 2018.

[42] Kirill Zubov, Zoe McCarthy, Yingbo Ma, Francesco Calisto, Valerio Pagliarino, Simone Azeglio, Luca Bottero, Emmanuel Luján, Valentin Sulzer, Ashutosh Bharambe, Nand Vinchhi, Kaushik Balakrishnan, Devesh Upadhyay, and Chris Rackauckas. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations, 07 2021.

[43] Dalia Čalnerytė, Rimantas Barauskas, Daiva Milašienė, Rytis Maskeliunas, Audrius Neciunas, Armantas Ostreika, Martynas Patasius, and Andrius Krisciunas. Multi-scale finite element modeling of 3d printed structures subjected to mechanical loads. *Rapid Prototyping Journal*, 24:00–00, 11 2017.