2017

Developing a Smart Proxy for Predicting the Fluid Dynamic in DamBreak Flow Simulation by Using Artificial Intelligence

Seyed Sina Hosseini Boosari

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Recommended Citation

Hosseini Boosari, Seyed Sina, "Developing a Smart Proxy for Predicting the Fluid Dynamic in DamBreak Flow Simulation by Using Artificial Intelligence" (2017). *Graduate Theses, Dissertations, and Problem Reports*. 5824.

<https://researchrepository.wvu.edu/etd/5824>

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Developing a Smart Proxy for predicting the fluid dynamic in DamBreak flow simulation by using Artificial Intelligence

Seyed Sina Hosseini Boosari

Thesis submitted

**to the Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University**

in partial fulfillment of the requirements for the degree of

Masters of Science in

Petroleum and Natural Gas Engineering

Ali Takkiri Borujeni, Ph. D., Committee Chairperson

Shahab D. Mohaghegh, Ph.D.

Ebrahim Fathi, Ph.D.

Department of Petroleum and Natural Gas Engineering

Morgantown, West Virginia

2017

**Keyword: CFD, DamBreak, Multiphase Flow, Dynamic flow, Artificial Neural Network &
Data Mining**

©Copyright 2017 Seyed Sina Hosseini Boosari

ABSTRACT

Developing a Smart Proxy for predicting the fluid dynamic in DamBreak flow simulation by using Artificial Intelligence

S.Sina.H.Boosari

Multiphase flow simulations are essential methods for providing information such as the evolution of phase fraction (gas, liquid and solid), velocities, pressure, temperature and flow regimes at every time during a process. Dynamic flow simulations also help reservoir, drilling, and production engineers to develop a proper well design. DamBreak problem is one of the most well-known problems in computational fluid dynamics (CFD); it is a dynamic hydraulic phenomena and the numerical simulation requires sophisticated mathematical modeling. OpenFOAM, is used to run CFD simulations in this thesis.

One of the main issues in CFD is that the simulations are time-consuming. In this work, will use artificial intelligence (AI) to predict the behavior of the system at each time-step of the process at a lower run time. DamBreak problem is defined base on a two-dimensional rectangular tank with a barrier at the bottom, the liquid column (water in this study) at the left side of the tank behind the wall. As soon as the wall collapse, the water will pour down, resulting in complicated fluid dynamics. The main data-set, generated by OpenFOAM flow simulations, is used for building the smart proxy model (SPM), using the network toolbox in MATLAB. Neural network (NN) is applied with feed-forward back propagation method and the training algorithm is Levenberg Marquardt.

Results indicate that the smart proxy can run 3 seconds of the DamBreak process, which takes 8 hours of computational time with 4 processors when is done by using OpenFOAM, takes less than 2 minutes using the developed SPM on one processor. SPM is also capable of predicting the CFD results in non-cascading condition and up to around 40 time-steps in cascading condition with acceptable error (less than %10).

Keyword: CFD, DamBreak, Multiphase Flow, Dynamic Flow, Artificial Neural Network.

NOMENCLATURE

U = Velocity

ρ = Density

g = gravitational force

μ = dynamic viscosity

μ_m = mixture viscosity

μ_1 = fluid viscosity

μ_2 = Air viscosity

k = surface curvature

σ = surface tension coefficient

P_g = Gas pressure

P_l = Liquid pressure

γ = phase fraction

ε_g = Gas fraction

ε_l = Liquid fraction

ρ_m = mixture density

ρ_1 = fluid density

ρ_2 = Air density

u_g = Velocity of gas in x direction

u_s = Velocity of solid in x direction

v_g = Velocity of gas phase

v_l = Velocity of Liquid phase

φ = Sample of property

τ = Diffusion coefficient divided by the fluid density

ρ' = Apparent Solid Density

ρ_g = Gas density

ρ_{sm} = Solid density

Acknowledgment

I would like to thank my thesis advisor *Dr. Ali Takbiri borujeni* for his supportive guidance during my research and *Professor. Shahab Mohaghegh* who gave me this opportunity to pursue my education in WVU and his motivation, and immense knowledge. I would like to express my sincere gratitude to *Professor Sam Ameri* who has consistently supported and encouraged me from the first day. I would also like to thank *Dr. Ebrahim Fathi* for his positive feedback during my thesis. Finally, I must express my very profound gratitude to my spouse for providing me with unfailing support and continuous encouragement throughout my lifetime and years of study.

Contents

Chapter 1	Introduction.....	1
1.1.	Problem statement.....	1
	Objective	2
	Literature review	3
Chapter 2	Background.....	4
2.1.	Multiphase flow.....	4
2.2.	Two-phase flow numerical solution.....	5
2.3.	OpenFOAM numerical solution.....	5
2.4.	Artificial neural network (ANN).....	7
2.5.	Type of neural network	8
	2.5.1. Feed Forward ANN.....	8
	2.5.2. Feedback ANN.....	8
Chapter 3	Methodology	9
3.1.	Data generating	9
3.2.	Model training.....	9
3.3.	Tier system	9
3.4.	Input and output of neural network.....	10
3.5.	Mesh generation	11
3.6.	Boundary condition	13
3.7.	Fluid properties	14
3.8.	Building a neural network model.....	14
3.9.	Multilayer neural network training algorithm.....	14
3.10.	Levenberg–Marquardt algorithm.....	15
3.11.	Feed-forward backpropagation.....	16

3.12.	NN data division	16
3.13.	Model building and deployment.....	18
3.14.	Cascading and non-cascading conditions	20
3.15.	Error definition	21
3.16.	Changing the data distribution.....	22
3.17.	Summary of the chapter.....	25
Chapter 4	Results and Discussion	26
4.1.	Model preparation	26
4.2.	Using single time-step for training.....	31
4.2.1.	Building a model for phase fraction based on a single time-step	31
4.2.2.	Building a model for pressure based on a single time-step	35
4.2.3.	Building a model for x-direction velocity based on a single time-step	38
4.2.4.	Building a model for y-direction velocity based on a single time-step	42
4.3.	Building a NN model based on several time-steps from significant dynamics parts.....	45
4.3.1.	Involving time-steps with highest error in training system	53
4.4.	Cascading result	58
4.5.	Cascading result for phase fraction	58
4.6.	Cascading result for pressure	61
4.7.	Cascading result for x-direction velocity	63
4.8.	Cascading result for y-direction velocity	65
4.9.	Reduction of data	67
4.9.1.	Non-cascading.....	67
4.9.2.	Cascading.....	74
4.9.3.	Make a model based on 200 time-steps	76
Chapter 5	Conclusions and Recommendations	82

5.1. Conclusions	82
5.2. Recommendations and future works	82
Bibliography	83
Appendix.....	84
Appendix 1: OpenFOAM DamBreak Code	84
Appendix 2: MATLAB Code.....	88

List of figures

Figure 1- Schematic initial conditions	1
Figure 2- Flow variations during the process.....	2
Figure 3- Gas-liquid Flow	4
Figure 4- Schematic figure of connections between neurons	7
Figure 5- Pattern of free forward ANN.....	8
Figure 6- Pattern of feedback ANN	8
Figure 7- Tiering system schematics	10
Figure 8- Input data, hidden layer & output.....	11
Figure 9- Tiering system schematic	11
Figure 10- Five areas of grid classifications	11
Figure 11- DamBreak schematic with a barrier at the bottom	11
Figure 12- Removing the barrier from the bottom.....	12
Figure 13- Grid numbering	12
Figure 15- Very fine mess with 400×400 grids	13
Figure 14- Fine mess with 200×200 grids	13
Figure 16- Medium mess with 50×50 grids	13
Figure 17- Boundary conditions	13
Figure 18- Transfer function	16
Figure 19- Example of validation stop because to overfitting phenomena.....	18
Figure 20- Number of output and input in ANN	19
Figure 21- Details of the input and output	19
Figure 22- Input, hidden layer and output of ANN.....	20
Figure 23- Schematic of non-cascading deployment	21
Figure 24- Schematic of cascading deployment	21
Figure 25- Histogram of phase fraction before and after elimination.....	23
Figure 26- Histogram of pressure before and after elimination.....	23
Figure 27- Histogram of x-direction velocity before and after elimination.....	24
Figure 28- Histogram of y-direction velocity before and after elimination.....	24
Figure 29- Schematic of input, hidden layer and output.....	26
Figure 30- 40×40 mesh, time-step 11	27
Figure 31- 40×40 mesh, time-step 12	27
Figure 32- 40×40 mesh, time-step 13	27

Figure 33- Validation performance and regression (time-step 0.004 seconds).....	28
Figure 34- Phase fraction results in time-step 575 (time-step size of 0.004 seconds)	28
Figure 35- Phase fraction results in time-step 576 (time-step size of 0.004 seconds)	29
Figure 36- Validation performance and regression (time-step size of 0.002 seconds).....	29
Figure 37- Phase fraction results in time-step 1151 (time-step size of 0.002 seconds)	30
Figure 38- Phase fraction results in time-step 1152 (time-step size of 0.002 seconds)	30
Figure 39- Phase fraction results in time-step 1153 (time-step size of 0.002 seconds)	30
Figure 40- Validation performance for phase fraction model.....	31
Figure 41- Regression results for training, validation, and test for building the phase fraction model	32
Figure 42- Non-cascading results for phase fraction at time-step 2301 when single time-step is applied .	32
Figure 43- Non-cascading results for phase fraction at time-step 2302 when single time-step is applied .	33
Figure 44- Non-cascading results for phase fraction at time-step 2303 when single time-step is applied .	33
Figure 45- Non-cascading results for phase fraction at time-step 2304 when single time-step is applied .	33
Figure 46- Non-cascading results for phase fraction at time-step 2305 when single time-step is applied .	34
Figure 47- Non-cascading results for phase fraction at time-step 2306 when single time-step is applied .	34
Figure 48- Non-cascading results for phase fraction at time-step 2307 when single time-step is applied .	34
Figure 49- Validation performance for the pressure model.....	35
Figure 50- Regression results for training, validation, and test for building the pressure model	35
Figure 51- Non-cascading results for pressure at time-step 2301 when single time-step is applied	36
Figure 52- Non-cascading results for pressure at time-step 2302 when single time-step is applied	36
Figure 53- Non-cascading results for pressure at time-step 2303 when single time-step is applied	36
Figure 54- Non-cascading results for pressure at time-step 2304 when single time-step is applied	37
Figure 55- Non-cascading results for pressure at time-step 2305 when single time-step is applied	37
Figure 56- Non-cascading results for pressure at time-step 2306 when single time-step is applied	37
Figure 57- Non-cascading results for pressure at time-step 2307 when single time-step is applied	38
Figure 58- Validation performance for x-direction velocity.....	38
Figure 59- Regression results for training, validation, and test for building the x-direction velocity model	39
Figure 60- Non-cascading results for x-direction at time-step 2301 when single time-step is applied	39
Figure 61- Non-cascading results for x-direction at time-step 2302 when single time-step is applied	40
Figure 62- Non-cascading results for x-direction at time-step 2303 when single time-step is applied	40
Figure 63- Non-cascading results for x-direction at time-step 2304 when single time-step is applied	40
Figure 64- Non-cascading results for x-direction at time-step 2305 when single time-step is applied	41
Figure 65- Non-cascading results for x-direction at time-step 2306 when single time-step is applied	41

Figure 66- Non-cascading results for x-direction at time-step 2307 when single time-step is applied	41
Figure 67- Regression results for training, validation, and test for building the y-direction velocity model	42
Figure 68- Validation performance for y-direction velocity	42
Figure 69- Non-cascading results for y-direction at time-step 2301 when single time-step is applied	42
Figure 70- Non-cascading results for y-direction at time-step 2302 when single time-step is applied	43
Figure 71- Non-cascading results for y-direction at time-step 2303 when single time-step is applied	43
Figure 72- Non-cascading results for y-direction at time-step 2304 when single time-step is applied	43
Figure 73- Non-cascading results for y-direction at time-step 2305 when single time-step is applied	44
Figure 74- Non-cascading results for y-direction at time-step 2306 when single time-step is applied	44
Figure 75- Non-cascading results for y-direction at time-step 2307 when single time-step is applied	44
Figure 76- Significant dynamics selected for the training process	45
Figure 77- Validation performance and regression for building the phase fraction model when 16 time-steps are used	46
Figure 78- Non-cascading results for phase fraction, sample of maximum error in whole process [0.426s, 0.63s, 1.67s, 2.44s, 2.66s]	47
Figure 79- Validation performance and regression for building the pressure model when 16 time-steps are used	48
Figure 80- Non-cascading results for pressure, sample of maximum error in whole process [0.4s, 0.68s, 2.13s, 2.44s, 2.604s].....	49
Figure 81- Validation performance and regression for building the x-direction velocity model when 16 time-steps are used.....	49
Figure 82- Non-cascading results for x-direction velocity, sample of maximum error in whole process [0.01s, 0.09s, 1.46s, 2.6s, 2.94s]	51
Figure 83- Validation performance and regression for building the x-direction velocity model when 16 time-step are used	51
Figure 84- Non-cascading results for y-direction velocity, sample of maximum error in whole process [0.00s, 0. 4s, 0. 8s, 2.57s, 2.64s]	53
Figure 85- Non-cascading results for phase fraction, sample of maximum error in whole process when high error time-steps are included.....	54
Figure 86- Non-cascading results for pressure, sample of maximum error in whole process when high error time-steps are included	55
Figure 87- Non-cascading results for y-direction velocity, sample of maximum error in whole process when high error time-steps are included.....	57

Figure 88- Non-cascading results for x-direction velocity, sample of maximum error in whole process when high error time-steps are included.....	58
Figure 89- Cascading results of phase fraction at time-step 501	59
Figure 90- Cascading results of phase fraction at time-step 502	59
Figure 91- Cascading results of phase fraction at time-step 505	59
Figure 92- Cascading results of phase fraction at time-step 510	60
Figure 93- Cascading results of phase fraction at time-step 515	60
Figure 94- Cascading results of phase fraction at time-step 520	60
Figure 95- Cascading results of phase fraction at time-step 525	60
Figure 96- Cascading results of pressure at time-step 501	61
Figure 97- Cascading results of pressure at time-step 502	61
Figure 98- Cascading results of pressure at time-step 505	61
Figure 99- Cascading results of pressure at time-step 510	62
Figure 100- Cascading results of pressure at time-step 515	62
Figure 101- Cascading results of pressure at time-step 520	62
Figure 102- Cascading results of x-direction velocity at time-step 501	63
Figure 103- Cascading results of x-direction velocity at time-step 502	63
Figure 104- Cascading results of x-direction velocity at time-step 505	63
Figure 105- Cascading results of x-direction velocity at time-step 510	64
Figure 106- Cascading results of x-direction velocity at time-step 515	64
Figure 107- Cascading results of x-direction velocity at time-step 520	64
Figure 108- Cascading results of y-direction velocity at time-step 501	65
Figure 109- Cascading results of y-direction velocity at time-step 502	65
Figure 110- Cascading results of y-direction velocity at time-step 505	65
Figure 111- Cascading results of y-direction velocity at time-step 510	66
Figure 112- Cascading results of y-direction velocity at time-step 515	66
Figure 113- Cascading results of y-direction velocity at time-step 520	66
Figure 114- Eliminated grids for phase fraction	67
Figure 115- Histogram of phase fraction data after elimination	68
Figure 116- Sample of non-cascading results for phase fraction based on non-stationary grids value	68
Figure 117- Eliminated grids for pressure	69
Figure 118- Histogram of pressure data after elimination	70
Figure 119- Sample of non-cascading results for pressure based on non-stationary grids value	70
Figure 120- Eliminated grids for x-direction velocity	71

Figure 121- Histogram of x-direction data after elimination.....	72
Figure 122- Sample of non-cascading results for x-direction velocity based on non-stationary grids value	72
Figure 123- Eliminated grids for y-direction velocity	73
Figure 124- Histogram of y-direction data after elimination.....	73
Figure 125- Sample of non-cascading results for x-direction velocity based on non-stationary grids value	74
Figure 126- Cascading results of phase fraction at time-step 1001 based on non-stationary grids value...	74
Figure 127- Cascading results of phase fraction at time-step 1005 based on non-stationary grids value...	75
Figure 128- Cascading results of phase fraction at time-step 1010 based on non-stationary grids value...	75
Figure 129- Cascading results of phase fraction at time-step 1015 based on non-stationary grids value...	75
Figure 130- Cascading results of phase fraction at time-step 1020 based on non-stationary grids value...	76
Figure 131- Cascading RMSE for phase fraction when entire process (all time-steps) are used for training	76
Figure 132- Focus on every 20 time-steps	77
Figure 133- Black points are eliminated grids	77
Figure 134- Cascading results of phase fraction at time-step 1001 (focused on 200 time-steps).....	78
Figure 135- Cascading results of phase fraction at time-step 1005 (focused on 200 time-steps).....	78
Figure 136- Cascading results of phase fraction at time-step 1010 (focused on 200 time-steps).....	78
Figure 137- Cascading results of phase fraction at time-step 1015 (focused on 200 time-steps).....	78
Figure 138- Cascading results of phase fraction at time-step 1020 (focused on 200 time-steps).....	79
Figure 139- Cascading results of phase fraction at time-step 1025 (focused on 200 time-steps).....	79
Figure 140- Cascading results of phase fraction at time-step 1030 (focused on 200 time-steps).....	79
Figure 141- Cascading results of phase fraction at time-step 1035 (focused on 200 time-steps).....	79
Figure 142- Cascading results of phase fraction at time-step 1040 (focused on 200 time-steps).....	80
Figure 143- Cascading results of phase fraction at time-step 1045 (focused on 200 time-steps).....	80
Figure 144- Cascading results of phase fraction at time-step 1050 (focused on 200 time-steps).....	80
Figure 145- Cascading RMSE for phase fraction when time-steps 1000 to 1200 are used for training.....	81
Figure 146- Comparing cascading results (entire process versus 200 time-steps)	81

List of tables

Table 1- Details of the input	10
Table 2- Grid classification	12
Table 3- Fluid properties	14
Table 4- NN model characteristics	14
Table 5- NN algorithms	15
Table 6- Network specification	16
Table 7- Data partitioning	17
Table 8- Time-steps used with their corresponding real time in the training process	19
Table 9- NN characteristics, single time-step for training	31
Table 10- Time-steps with related real times used for training	45
Table 11- NN characteristics when 16 time-steps are used for training	46
Table 12- Phase fraction model specifications after elimination	67
Table 13- Pressure model specifications after elimination	69
Table 14- X-direction velocity model specifications after elimination	71
Table 15- Y-direction velocity model specifications after elimination	73

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
NN	Neural Network
CFD	Computational Fluid Dynamics
VTU	Visualization Toolkit Unstructured points data
CSV	Comma Separated Value
DM	Data Mining
KPI	Key Performance Indicator
MSE	Mean Square Error
RMSE	Root of Mean Square Error
SPM	Smart Proxy Model
VOF	Volume Of Fluid
LMA	Levenberg–Marquardt Algorithm
DLS	Damped least-Squares
GNA	Gauss–Newton Algorithm

Software

OpenFOAM

ParaView

MATLAB

Chapter 1 Introduction

In this thesis, I study two-dimensional DamBreak problem by developing a smart proxy model (SPM). CFD simulations are performed using the OpenFOAM package.

1.1. Problem statement

In the CFD simulations using the OpenFOAM package, physical properties such as pressure, velocity, and phase fraction at each time step are the main unknowns. In DamBreak problem, the numerical domain consists of a rectangular tank with a column of water located at the left side of a membrane (Figure 1). Initially the membrane is removed, which leads to the collapse of the column of water and generates a complicated flow structure, including water and captured bubbles (Figure 2).

Figure 1- Schematic initial conditions

Figure 2- Flow variations during the process

Objective

Reducing the computational time for the fluid flow simulations by developing a SPM is the main objective of this thesis. To achieve this, artificial intelligent (AI) and data mining (DM) techniques are employed during building and designing SPMs, that possess the same precision as CFD simulations.

Literature review

Artificial intelligence is widely used in industry in recent years. A. Alizadeh and M. Rahimi [1] used artificial neural network with three inputs including tube slope and velocities for gas and liquid in order to predict pressure drops in 2cm diameter and 6m length tube. Experimental setup provided main data in this study. Pressure loss calculated and the pressures inside the pipe were measured. They used three transfer functions including Hyperbolic-Tangent Sigmoid, Log- Sigmoid, and linear to select the best one. Also finding an optimum number of neurons is performed based on using eight different network.

S. Esmaili and S. Mohaghegh [2] took the advantages of machine learning, data mining and pattern recognition in order to model and history-match gas production in Marcellus shale reservoirs. They used real data such as production rate, well logs and completion information and hydraulic fracturing data in order to assess hydrocarbon production. Using “hard data” which is related to the measurement during fracturing process has made this work unique.

A. Kalantari and S. Mohaghegh [3] built a model using machine learning and numerical solution for developing a SPM for regenerating methane production, which generate results in less than a second with acceptable accuracy.

Chapter 2 Background

In this chapter, multiphase flow and especially two-phase flow will be discussed. Next, the partial differential equations solved in the DamBreak problem will be presented. At the end of this chapter, application of AI for predicting the behavior of fluid properties will be discussed.

2.1. Multiphase flow

The simultaneous flow of at least two phases in a place called multi-phase flow. Predicting the behavior of each phase in multiphase flow systems are necessary and critical in many industries. There are four different forms of two-phase flow, which include:

- 1) liquid-liquid flow, consist of two immiscible liquids.
- 2) Gas-liquid flow (Figure 3).

Figure 3- Gas-liquid Flow

- 3) Liquid-Solid flow
- 4) Gas-solid flow

2.2. Two-phase flow numerical solution

Generally, there are two methods used for numerical solution of multiphase flows. Eulerian–Lagrangian approach and Eulerian–Eulerian approach. Eulerian solution is based on the locations (value of specific grids during process) during the process by using “Navier-Stocks” equation, the Lagrangian solution is based on particles movement during the process using Newton’s equation [4].

Eulerian–Lagrangian method is the combination of both Eulerian and Lagrangian solutions, it means that continuous phase is considered for Eulerian while dispersed phases considered for Lagrangian solutions [5].

2.3. OpenFOAM numerical solution

The numerical simulations of DamBreak in OpenFOAM are based on the solution of the complete set of Navier-Stokes equations and the volume of fluid (VOF) method [6]. VOF is a surface tracking technique [7]. This method is employed on DamBreak problem considering two assumptions: first, is over a flatbed and second is without any friction [8].

Two equations need to be solved: the momentum equation (Equation (1)) and the continuity equation (Equations (2), (1)). The momentum equation is:

$$\frac{\partial \rho U}{\partial t} + \nabla \cdot (\rho U U) - \nabla \cdot \mu \nabla U - \rho g = -\nabla p \quad (1)$$

where U is velocity, ρ is density, μ is dynamic viscosity, σ is the surface tension coefficient, g is gravitational force.

Pressure P and velocity U value at each cell are calculated by solving the continuity and momentum equations. The continuity equation is:

$$\nabla \cdot U = 0 \quad (2)$$

$$\frac{\partial Z}{\partial t} + \frac{\partial(hu)}{\partial t} + \frac{\partial(hv)}{\partial t} = 0 \quad (3)$$

where h is the water depth, γ is the phase fraction in a cell so the values of γ in a cell is between zero and one. $\gamma=1$ means that, the cell is completely filled with water and if it is filled with air, γ is zero. Generally, phase fraction equation is:

$$\gamma = \frac{\text{fluid volume}}{\text{cell volume}}$$

Density ρ and viscosity μ for each cell will be computed based on the phase fraction, using:

$$\rho_m = \gamma \rho_1 + (1 - \gamma) \rho_2 \quad (4)$$

where, ρ_m is mixture density, ρ_1 is water density and ρ_2 is Air density.

$$\mu_m = \gamma \mu_1 + (1 - \gamma) \mu_2 \quad (5)$$

where, μ_m is mixture viscosity, μ_1 is water viscosity and μ_2 is Air viscosity.

Apart from Equation (1) and (2) another equation that is used in the transport equation (for property ϕ) with following general form:

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla \cdot (\rho\phi U) - \nabla \cdot (\tau \nabla \phi) = S_\phi \quad (6)$$

where, first term represents the rate of change of property ϕ with time. The second term indicates the advection of property ϕ by the fluid flow and the third term represents the diffusion of property ϕ . Moreover, the last term (S_ϕ) shows the source term. (τ is Diffusion coefficient divided by the fluid density)

Transport equation with following form is used to calculate the relative volume fraction in each cell:

$$\frac{\partial \gamma}{\partial t} + \nabla \cdot (\gamma U) = 0 \quad (7)$$

By adding artificial compression term into this equation, necessary compression of the surface will be calculated:

$$\frac{\partial \gamma}{\partial t} + \nabla \cdot (\gamma U) + \nabla \cdot (\gamma (1 - \gamma) U_r) = 0 \quad (8)$$

Where, U_r is proper velocity field to compress the interface.

In this work, we use PIMPLE to solve for pressure and velocities [9].

2.4. Artificial neural network (ANN)

In the mid-19th century, Warren McCulloch and Walter Pitts designed computational model for NNs based on mathematics and algorithms. This model is considered first step in NN research. The purpose of the neural network is to solve problems like a human brain does with large interconnected neural units (Figure 4). Moreover, it works similar to the human brain in many ways, especially in process that are too complicated to design, program and solve directly such as, advanced learning, optimization, sorting, extension and extraction [10].

In many models, “*defining a simple function*” for variables or “*distribution*”, constitute the mathematical models for the artificial neural network (ANN). Essentially training a NN model is choosing one model with minimum cost criterion through the set of generated model. Many algorithms exist for training a NN. The gradient descent theory is one of the most of widespread algorithms in training ANNs [11].

Nns have consisted of multiple nodes. The input data will be generated with nodes and after doing simple mathematical operations, they will transfer to other neurons. The output of each node is called “activation” or “node value”. Each link between nodes is associated with a weight. The neural network model could be learned, by modifying weight values.

Figure 4- Schematic figure of connections between neurons

2.5. Type of neural network

There are two types of ANN based on topology [12]:

2.5.1. Feed Forward ANN

In this kind of ANN, information just moves in the forward direction. The inputs and outputs are fixed based on the initial definition and there are no loops that could be formed between units, data moves from input to hidden layer to the output (Figure 5).

Figure 5- Pattern of free forward ANN

2.5.2. Feedback ANN

In feedback ANN models, loops are formed in this system. They are connected from outputs directly to inputs and hidden layers (Figure 6).

Figure 6- Pattern of feedback ANN

Chapter 3 Methodology

In methodology section of this thesis, I will discuss the problem-solving, specification of data and how datasets are generated from OpenFOAM simulations. Also covered will be critical properties, tier system, grid classification and boundary conditions that are explained in detail. The method of NN, number of input and how the CFD model is defined in the AI will be other sections of this chapter.

3.1. Data generating

Preparing and building a comprehensive dataset is one of the most significant parts of each data-mining (DM) project. Visualizing the data set is the first step of DM; the large dataset, which is generated by OpenFOAM, could be visualized and analyzed with the ParaView package. ParaView is an open-source software with the capability to visualize large datasets, as well as converting the format of data. The OpenFOAM dataset includes the computed variables for each grid at each time-step.

3.2. Model training

Number of variables of interest are based on the type of simulation and problem definition. In this study, four variables are in need of prediction (phase fraction, pressure, x-direction velocity and y-direction velocity). The network should be trained for each variable separately.

For training the model of each variable, 24 inputs have been defined. Four variables for the cell of interest, four variable as a distance of the cell of interest from walls and also four variables for each cell that is considered as tiers (Table 1 and Figure 8).

3.3. Tier system

In this work, a method is used for accounting for the neighboring cells, which is called tiering system. The information of the neighboring cells are considered as features in the NN. This includes pressure, velocities in x and y directions and phase fraction. This 2-dimensional (2D) problem, as such the four cells surrounding the cell of interest are accounted for as tier 1. One cell at the top, one at the bottom, one to the left and one to the right side of the cell of interest.

Tier 2 includes those cells that share a corner with the cell of interest and Tier 3 contains the cells in the second layer. In this work, I only used Tier 1 data (Figure 7 and Figure 8).

Figure 7- Tiering system schematics

3.4. Input and output of neural network

The following table indicates input parameters of an ANN in detail. Each property of cell of interest could be considered as the output (target).

Table 1- Details of the input

Number of input: 24	Main cell parameter at time-step n: 4	phase Fraction
		x-direction velocity
		y-direction velocity
		pressure
	Tier cells at time-step n: 16	Phase Fraction
		x-direction velocity
		y-direction velocity
		pressure
	Location of main cell: 4	distance 1
		distance 2
		distance 3
		distance 4

Figure 9- Tiering system schematic

Figure 8- Input data, hidden layer & output

3.5. Mesh generation

“BlockMesh” module in OpenFOAM was used for mesh generation. In this module, each block has eight vertices at each corner. This problem is considered on a uniform mesh and it is assumed to be laminar flow.

DamBreak problem in OpenFOAM software in 2D consists of a square tank with a column of water inside of the tank as well as a barrier in the bottom (Figure 11). Based on this design, the model was divided into the five different areas (Figure 10 and Figure 11) [13].

Figure 11- DamBreak schematic with a barrier at the bottom

Figure 10- Five areas of grid classifications

For simplicity, the barrier has been removed from the domain and the model built with one area (Figure 12).

OpenFOAM numbers the cells from bottom left corner, the first grid in the corner is grid *number one*, the next one in x-direction is grid *number two*, and so on (Figure 13).

Figure 12- Removing the barrier from the bottom

Figure 13- Grid numbering

Three types of grid sizes are used in this project: first one is medium grid size (50×50), the second one is fine (200×200) and the third one is very fine (400×400). After running the simulation for all the grid sizes, it was decided that fine classification (200×200) would be the appropriate size for this project (Figure 15 to Figure 16). The medium grid size is rejected due to low regulation and the very fine grid size is rejected due to increasing number of records. In Table 2 grid classification and related information are listed.

Table 2- Grid classification

Grid Classification	Dimension	No. of Cells	Cell Size (cm)
Medium	50×50	2500	1.168×1.168
Fine	200×200	40,000	0.292×0.292
Very Fine	400×400	160,000	0.146×0.146

Figure 16- Medium mess with 50×50 grids

Figure 15- Fine mess with 200×200 grids

Figure 14- Very fine mess with 400×400 grids

3.6. Boundary condition

In a 2D DamBreak problem, four boundary patches are defined. Three patches are walls which are located on the bottom, left and right-hand side of the tank and one patch for the top which is exposed to atmosphere condition. It means both of the outflow or inflow are free to move up and down (Figure 17).

Figure 17- Boundary conditions

3.7. Fluid properties

The first phase is selected to be Water and the second phase is air (Table 3).

Table 3- Fluid properties

Fluid properties	Water (phase 1)	Air (phase 2)	Symbol	unit
Kinematic viscosity	1.0×10^{-6}	1.48×10^{-5}	ν	$\text{m}^2 \text{s}^{-1}$
Density	1.0×10^3	1.0	ρ	kgm^{-3}
Surface tension	0.07	0.07	δ	Nm^{-1}

3.8. Building a neural network model

NN model is built based on three sets of frameworks; the first one is the input data and called predictor. The second one is output data (forecasts or target) and the third one is the intermediate layer that is called the hidden layer. When hidden layers are not applied while building a model it is similar to linear regression, and when the hidden layers are included it is non-linear regression.

The number of inputs is defined based on the number of the properties that are effective on target parameters and problem nature. In DamBreak problem, there are four variables that are calculated in each time-step. As was mentioned previously, twenty-four parameters are defined as input of the NN. One hidden layer and 50 hidden neurons were selected for this problem. The number of hidden neurons is selected based on trial and error and the best result that was achieved using 50 hidden neurons. The last part of the model design is target definition (output) or variables that need to be predicted (Table 4), [10].

Table 4- NN model characteristics

Number of properties	Number of Inputs	Number of Hidden Neurons	Number of records	Number of Output
4	24	50	40,000	1

3.9. Multilayer neural network training algorithm

There are several algorithms used for training ANN and choosing the best algorithm depends on the nature of the problem. Many factors impact the choice of the algorithm such as problem complexity,

the size of the dataset (number of records), and type of the network (for example discriminant analysis method or regressive method). The following table lists different algorithms (Table 5), [14].

Table 5- NN algorithms [14]

Algorithm Name	Acronym	Description
Levenberg-Marquardt	LMA	-One of the fastest backpropagation algorithm. -First-choice of supervised algorithm. -more memory is required in comparison to other algorithms
BFGS Quasi-Newton	BFG	An iterative method for solving unconstrained nonlinear optimization problems.
Polak-Ribière Conjugate Gradient	CGP	It is a kind of supervised learning algorithm.
Resilient Backpropagation	RP	A proper algorithm in backpropagation techniques. One of the fastest weight update mechanisms. But it is more complex algorithm in compare to other back propagation algorithm. It is similar to levenberg-Marquardt algorithm.
One Step Secant	OSS	This method is trying to bridge the gap between the quasi-Newton (secant) algorithms and the conjugate gradient algorithms.
Scaled Conjugate Gradient	SCG	Designed for linear equations in the numerical solution of particular systems.
Variable Learning Rate Backpropagation	GDX	This algorithm is based on learning rate. The learning rate is constant during the training. The performance is highly depended and sensitive to suitable setting of learning rate.
Conjugate Gradient with Powell/Beale Restarts	CGB	Is recommended for large scale unconstrained optimization
Fletcher-Powell Conjugate Gradient	CGF	It's similar to quasi-Newton method It is used for multidimensional problems.

3.10. Levenberg–Marquardt algorithm

In 1944, Kenneth Levenberg and Donald Marquardt separately developed an algorithm based on the numerical solution in order to minimize non-linear functions, which is called LMA (Levenberg–Marquardt algorithm), or DLS (damped least-squares). Interpolation between the GNA (Gauss–

Newton algorithm) and gradient descent method is the main strategy of LMA. This algorithm is fast and suitable for problems that include relatively small sizes of data [15].

3.11. Feed-forward backpropagation

A feedforward backpropagation is a method of training that proceeds forward and computes the respective values. The computed values are then compared with the actual values and based on the error value, it goes back (back propagate) in order to reduce the error by changing and adjusting the weights.

Another important parameter in ANN is the transfer function that calculates output layers from the input. The transfer function takes the input value and returns it after converting between -1 and 1. For this model “*Tansing*”, transfer function is selected (Figure 18).

Figure 18- Transfer function

In (Table 6), the network characteristic are presented.

Table 6- Network specification

Network method	<i>Feed-forward back propagation</i>
Training algorithm	<i>Levenberg-Marquardt</i>
Error definition (performance function)	<i>Mean Square Error</i>
Transfer function	<i>TANSIG</i>

3.12. NN data division

In order to overcome “over-fitting”, data need to be divided into three parts of training, validation, and testing. The main concept behind the data division for a NN is to avoid “over- fitting” the problem.

Valuable model is not only compatible with training data, but also should predict acceptable results using the blind data.

The first part, which includes the largest amount of data, is used for learning the behaviors and training the model based on finding the optimal neuron's weights.

Validation dataset is not involved in the learning process directly and the main purpose of this data-set is to avoid over-fitting. The tuning or calibrating of a model is done by this second set of data and also used to evaluate the performance of the predictive model that was created using the training set (Figure 19). In the training part, magnitude of the error is always decreasing after each epoch from first to last. The learning process is stopped as soon as validation error increases.

The last set of data which is called testing is used just once and is not designed for the training of the model. This part of dataset works as a blind data for checking the accuracy of the best network model that was stopped during validation.

Normally the majority of data-set which is included 70% of data is used for learning, 15% of data-set is utilized for validation and 15% of data is considered for testing part, these percentages could be different amount based on problem nature. In (Table 7), the percentage of each part in this study is shown.

Table 7- Data partitioning

Data division	Data percentage used in this study
Training	70%
Validation (Calibration)	15%
Testing	15%

Figure 19- Example of validation stop because to overfitting phenomena

3.13. Model building and deployment

Each network consists of two main sections, building a model and deployment. Generally, the goal of each network is building a perfect model to predict properties. Determining the most effective variables is the first step of building a NN model. In this project, several ideas based on the physics of problem were applied to achieve an appropriate network.

Dependence of parameters to each other and the amount of dependency play a significant role in input parameters determination. On the other hand, for building a comprehensive model the user should consider all features that might be effective on project approach. In this study, four main variables are of interest, but by using just these parameters, we will not be able to make a good model.

Another important feature is using a reasonable time-step. The temporal spacing between time-steps should not be too long or too short, because the system will not learn perfectly due to large or small dynamic changes.

For learning process, time-step “n” is used for input of NN and time-step “n+1” is used as the output. As is previously mentioned, 24 variables are defined based on the problem nature (Figure 20). These variables are considered from time-step “n” as NN input and each one of main parameters (fraction, pressure, x and y-direction velocity) from time-step “n+1” could be the output of NN (Figure 21 and Figure 22). By using 30 couples of significant movements from the whole process (3000 time-steps), four NN model are built (phase fraction, pressure and velocity in x and y direction). The table of time-steps used for training with related real time is shown on (Table 8). The whole process is 3 seconds and the temporal between time-steps is 0.01 second.

Table 8- Time-steps used with their corresponding real time in the training process

Time-step	Real-time
100	0.1
500	0.5
720	0.72
835	0.835
900	0.9
1022	1.022
1300	1.3
1500	1.5
1750	1.75
1920	1.92
2000	2
2048	2.048
2250	2.25
2251	2.251
2315	2.315

2316	2.316
2465	2.465
2500	2.5
2501	2.501
2570	2.57
2571	2.571
2572	2.572
2590	2.59
2591	2.591
2702	2.702
2703	2.703
2720	2.72
2850	2.85
2851	2.851
2999	2.999

Figure 20- Number of output and input in ANN

Input from Time-step "n"

Main Cell and four tiers			
ϵ	P	V_x	V_y

Distance from boundaries for main Cell			
D ₁	D ₂	D ₃	D ₄

Output is one of parameters from time-step "n+1"

ϵ
P
V_x
V_y

Figure 21- Details of the input and output

The network works as a machine that receives data and gives out a number based on its learning quality. Two types of data feed for the deployment of the model are provided, original data and output of NN, respectively called Non-cascading and cascading conditions. The result of both deployments are presented in next chapter.

Figure 22- Input, hidden layer and output of ANN

3.14. Cascading and non-cascading conditions

After the learning process and building the model, the deployment process is begun. The machine provides output for each input dataset, and if the output of machine is used as the input for next time-step, it is called cascading , in the other words using one set of CFD data from one time-step, the rest of time-steps value could be predicted (machine feeds itself). In the other hand, non-cascading refers to conditions that the NN machine does not feed itself and uses CFD dataset for each time-step. Building a perfect Non-cascading model is necessary for cascading condition to be built. Initial useful steps are performed in this project for cascading deployment and if it is done properly and completely, it will be a great progress in the fluid dynamic science.

Since output of NN is used as the input of the next time-step, models that contain minor inaccuracies can lead to a larger scale of error. This is the most significant issue with the cascading condition. Considering very high-precision (99%) in the non-cascading part, the accuracy will be reduced after elapsing a few of time-steps. Several ideas have been applied to maximize the accuracy in non-cascading and despite the great progress; the significant changes have not been observed. Figure 23 and Figure 24 clearly show the difference between these two methods.

Figure 23- Schematic of non-cascading deployment

Figure 24- Schematic of cascading deployment

3.15. Error definition

For understanding and comparison the amount of error of NN and CFD models, we need to define appropriate errors for each parameter. For phase fraction, the error is defined as the absolute value of subtraction of CFD and NN.

$$Error = abs (CFD_{output} - NN_{output})$$

For other parameters such as pressure and velocity, the values of CFD and NN need to be normalized separately, and then error is defined:

$$CFD\ Normal = \frac{(CFD_{output} - Mean)}{(MAX_{CFD\ output} - MIN_{CFD\ output})},$$

$$NN\ Normal = \frac{(NN_{output} - Mean)}{(MAX_{CFD\ output} - MIN_{CFD\ output})}$$

$$Error = abs(CFD\ Normal - NN\ Normal) \times 100,$$

There are several definitions for the error calculation, but two of that are most prevalent for assessment of NN performance. RMSE which is Root of Mean Square Error and MSE which is Mean Square Error. RMSE is used in this project for demonstration of error.

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_{CFD} - y_{NN})^2},$$

$$MSE = \frac{1}{n} \sum_{j=1}^n (y_{CFD} - y_{NN})^2$$

3.16. Changing the data distribution

By looking at the CFD results, it is found that values in many grids have not changed significantly over large number of time steps. One of the ideas that was performed in order to increase the accuracy of model was changing the distribution of data. For instance, majority of phase fraction values remain constant from beginning to end of the simulation and were either zero or one. By eliminating these data, a new model has been trained with smaller amount of data. Process of elimination the data is different for each variables. Data distributions are shown by histogram graphs before and after elimination in following figures. The number of constant grids and percentage of elimination for each variable as well as schematic figures of eliminated grids will be shown separately in the next chapter. (Table 12 to Table 15 and Figure 114 to Figure 125).

The histogram graphs of phase fraction for a time-step before and after elimination indicate that the frequency of “0” have been decreased from almost 34,000 to 21,000 and frequency of “1” decreased from 5,000 to 4,000 (Figure 25).

Figure 25- Histogram of phase fraction before and after elimination

Due to elimination, pressure distribution is also changed significantly. Most of the data are zero and the frequency decreased to 12,000 (Figure 26).

Figure 26- Histogram of pressure before and after elimination

The following histogram graphs indicate the distribution of x and y-direction velocity. For x-direction velocity the maximum frequency reduced from 12000 to 3600 and frequency of y-direction velocity, decreased from 6500 to 5000 (Figure 27 and Figure 28).

Figure 27- Histogram of x-direction velocity before and after elimination

Figure 28- Histogram of y-direction velocity before and after elimination

3.17. Summary of the chapter

In this chapter, NN model generation for building the model and ways of deployment were explained. Preparing the data and visualizing them by applying related software were mentioned as well. All effective variables were also discussed. In the next chapter, the obtained results from various models will be presented in detail.

Chapter 4 Results and Discussion

In the first section of this chapter, the ideas that were applied for improving the non-cascading models are presented. The ideas contain several approaches such as: using different time-steps for training, elimination of constant grids, and concentration on a hundred consecutive time-steps. In the second part of this chapter, cascading result are demonstrated.

In order to compare and visualize the result of NN model prediction to CFD data, for each time-step, a 2D figure is generated. Based on the grid values, each figure contains three subplots: left side subplot shows OpenFOAM results, the middle one represents the prediction data made by the NN model, and the last one is on the right demonstrates the error percentage.

4.1. Model preparation

In this section before referring to the main result, the beginning steps such as “gridding classification” and “selecting temporal time-step” will be presented. Grid size selection depends on the numerical accuracy, resolution of the computed variables, and number of records generated.

Another significant selection is temporal time-step as a system cannot learn correctly when temporal time-step is too big or too small. In Figure 30 , Figure 31 and Figure 32, large grids (low resolution (40×40)) and high dynamic movement (0.05 s) in three consecutive time-steps are shown.

In order to achieve the best gridding size, several sizes such as (40×40), (200×200) and (400×400) have been tested and (200×200) was selected.

Figure 29- Schematic of input, hidden layer and output

Figure 30- 40×40 mesh, time-step 11

Figure 31- 40×40 mesh, time-step 12

Figure 32- 40×40 mesh, time-step 13

As mentioned before, selection of temporal time-step should be reasonable in terms of dynamic changes. The two following models has been trained with temporal time-steps of $0.002s$ and $0.004s$ for phase fraction as a sample. These intervals between time-steps are more appropriate to enable training of the network. Result of 0.004 second, temporal time-step are shown in Figure 34 and Figure 35. Time-step 575 is used as input and 576 as an output. The process stopped after 3 epochs due to validation error (Figure 33).

Figure 33- Validation performance and regression (time-step 0.004 seconds)

Figure 34- Phase fraction results in time-step 575 (time-step size of 0.004 seconds)

Figure 35- Phase fraction results in time-step 576 (time-step size of 0.004 seconds)

Result of 0.002 second temporal time-step (Figure 37 to Figure 39):

Figure 36- Validation performance and regression (time-step size of 0.002 seconds)

The red points in error distribution indicate at least 10% error. Time-step 1150 is used as an input and time-step 1151 as an output.

Figure 37- Phase fraction results in time-step 1151 (time-step size of 0.002 seconds)

Figure 38- Phase fraction results in time-step 1152 (time-step size of 0.002 seconds)

Figure 39- Phase fraction results in time-step 1153 (time-step size of 0.002 seconds)

4.2. Using single time-step for training

After selecting proper time-step and grid sizes, the model has been trained with time-step 2300 as the input and 2301 as the output. It is understood that using one time-step is not a good idea for all process prediction. This network is able just to predict one or two time-step reasonably and after that, error increases.

4.2.1. Building a model for phase fraction based on a single time-step

As is shown in the validation graph, the blue line indicates training improvement, the green line shows the validation error and the red line demonstrates the test error base on MSE. The training error line (MSE) is decreasing steadily, but it not an evidence that the best model is built. The model must be accurate with applying blind set of data. The process stopped at epochs 28 due to six consecutive validation error (Figure 40 and Figure 41).

Figure 40- Validation performance for phase fraction model

Table 9- NN characteristics, single time-step for training

Number of parameters	24
Number of Hidden Neurons	50
Number of output layer	1
Number of output	1
Number of rows	40,000

Figure 41- Regression results for training, validation, and test for building the phase fraction model

The related result is shown in Figure 42 to Figure 48.

Figure 42- Non-cascading results for phase fraction at time-step 2301 when single time-step is applied

Figure 43- Non-cascading results for phase fraction at time-step 2302 when single time-step is applied

Figure 44- Non-cascading results for phase fraction at time-step 2303 when single time-step is applied

Figure 45- Non-cascading results for phase fraction at time-step 2304 when single time-step is applied

Figure 46- Non-cascading results for phase fraction at time-step 2305 when single time-step is applied

Figure 47- Non-cascading results for phase fraction at time-step 2306 when single time-step is applied

Figure 48- Non-cascading results for phase fraction at time-step 2307 when single time-step is applied

4.2.2. Building a model for pressure based on a single time-step

The model has been trained for pressure prediction. Variables in time-step 2300 were used as an input and pressure in 2301 is used as the target. Validation performance and regression figures are shown in following figures. The model stopped after 217 epochs to six consecutive validation error (Figure 49 and Figure 50).

Figure 49- Validation performance for the pressure model

Figure 50- Regression results for training, validation, and test for building the pressure model

The result of pressure training is shown in following figures. The error percentage is calculated after each time-step. The red points in error distribution indicate at least 10% error (Figure 51 to Figure 57).

Figure 51- Non-cascading results for pressure at time-step 2301 when single time-step is applied

Figure 52- Non-cascading results for pressure at time-step 2302 when single time-step is applied

Figure 53- Non-cascading results for pressure at time-step 2303 when single time-step is applied

Figure 54- Non-cascading results for pressure at time-step 2304 when single time-step is applied

Figure 55- Non-cascading results for pressure at time-step 2305 when single time-step is applied

Figure 56- Non-cascading results for pressure at time-step 2306 when single time-step is applied

Figure 57- Non-cascading results for pressure at time-step 2307 when single time-step is applied

4.2.3. Building a model for x-direction velocity based on a single time-step

The network has been trained for x-direction velocity at time-step 2300 as an input and x-direction velocity at time-step 2301 as an output. The model stopped after 100 epochs and the MSE of validation is less than 0.01. The validation performance and regression of x-direction velocity are presented in following figures (Figure 58 and Figure 59).

Figure 58- Validation performance for x-direction velocity

Figure 59- Regression results for training, validation, and test for building the x-direction velocity model

The results of deployment from time-step 2301 to 2307 for x-direction velocity are presented in following figures. (Figure 60 to Figure 66).

Figure 60- Non-cascading results for x-direction at time-step 2301 when single time-step is applied

Figure 61- Non-cascading results for x-direction at time-step 2302 when single time-step is applied

Figure 62- Non-cascading results for x-direction at time-step 2303 when single time-step is applied

Figure 63- Non-cascading results for x-direction at time-step 2304 when single time-step is applied

Figure 64- Non-cascading results for x-direction at time-step 2305 when single time-step is applied

Figure 65- Non-cascading results for x-direction at time-step 2306 when single time-step is applied

Figure 66- Non-cascading results for x-direction at time-step 2307 when single time-step is applied

4.2.4. Building a model for y-direction velocity based on a single time-step

The network has been trained for y-direction velocity at time-step 2300 as an input and y-direction velocity at time-step 2301 as an output. The model stopped after 146 epochs and the MSE of validation is less than 0.01. The validation performance and regression of y-velocity are presented in Figure 67 and Figure 68.

Figure 68- Validation performance for y-direction velocity

Figure 67- Regression results for training, validation, and test for building the y-direction velocity model

The result of deployment from time-step 2301 to 2307 is presented below (Figure 69 to Figure 75).

Figure 69- Non-cascading results for y-direction at time-step 2301 when single time-step is applied

Figure 70- Non-cascading results for y-direction at time-step 2302 when single time-step is applied

Figure 71- Non-cascading results for y-direction at time-step 2303 when single time-step is applied

Figure 72- Non-cascading results for y-direction at time-step 2304 when single time-step is applied

Figure 73- Non-cascading results for y-direction at time-step 2305 when single time-step is applied

Figure 74- Non-cascading results for y-direction at time-step 2306 when single time-step is applied

Figure 75- Non-cascading results for y-direction at time-step 2307 when single time-step is applied

4.3. Building a NN model based on several time-steps from significant dynamics parts

In previous approach, we found that using a single time-step will not lead to a comprehensive model for entire the process, so all variables from the most significant dynamics parts have been selected for training procedure. Figure 76 and Table 10 include all time-steps used in this approach and number of rows as are mentioned in Table 11 increased to 640,000.

Figure 76- Significant dynamics selected for the training process

Table 10- Time-steps with related real times used for training

Time-step	Real time (Sec)	Time-step	Real time (Sec)
100	0.1	2200	2.2
500	0.5	2500	2.5
720	0.72	2501	2.51
900	0.9	2720	2.72
1300	1.3	2721	2.72
1500	1.5	2850	2.85
1750	1.75	2851	2.85
1920	1.92	3000	3

Table 11- NN characteristics when 16 time-steps are used for training

Number of parameters	24
Number of Hidden layer	50
Number of output layer	1
Number of out put	1
Number of rows	640,000

Phase fraction result when 16 time-steps are used for training are presented in Figure 78 and the validation performance and regression of phase fraction are shown in Figure 77.

Figure 77- Validation performance and regression for building the phase fraction model when 16 time-steps are used

Figure 78- Non-cascading results for phase fraction, sample of maximum error in whole process [0.426s, 0.63s, 1.67s, 2.44s, 2.66s]

Pressure result when 16 time-steps are used for training are presented in Figure 80 , also the validation performance and regression of pressure are shown in Figure 79.

Figure 79- Validation performance and regression for building the pressure model when 16 time-steps are used

Figure 80- Non-cascading results for pressure, sample of maximum error in whole process [0.4s, 0.68s, 2.13s, 2.44s, 2.604s]

X-direction velocity result when 16 time-steps are used for training presented in Figure 82, also the validation performance and regression of x-direction velocity are shown in Figure 81.

Figure 81- Validation performance and regression for building the x-direction velocity model when 16 time-steps are used

Figure 82- Non-cascading results for x-direction velocity, sample of maximum error in whole process [0.01s, 0.09s, 1.46s, 2.6s, 2.94s]

Y-direction velocity result when 16 time-steps are used for training presented in Figure 84, also the validation performance and regression of y-direction velocity are shown in Figure 83.

Figure 83- Validation performance and regression for building the x-direction velocity model when 16 time-step are used

Figure 84- Non-cascading results for y-direction velocity, sample of maximum error in whole process [0.00s, 0. 4s, 0. 8s, 2.57s, 2.64s]

From this section using 16 time-steps for building a model, it is concluded that using significant movements is a great enhancement in order to have a perfect model for the entire process. In some time-steps, especially in pressure and velocity, the noticeable error observed (Figure 80, Figure 82, Figure 84). In order to improve accuracy, more time-steps from the parts where with highest error occurred were included.

4.3.1. Involving time-steps with highest error in training system

As it was observed in previous section, the model was not good enough through the entire process. In this section, a new model for each parameter was built by involving high error time- steps. This consideration improved the non-cascading model. In previous section, 16 time-steps are used for NN training process, but this number of time-steps were increased up to 30 for each model. The result of this idea will be presented in following sections (Figure 85 to Figure 88).

4.3.1.1. Phase fraction result including high error time-steps

Figure 85- Non-cascading results for phase fraction, sample of maximum error in whole process when high error time-steps are included

4.3.1.2. Pressure result by using high error time-steps

Figure 86- Non-cascading results for pressure, sample of maximum error in whole process when high error time-steps are included

4.3.1.3. Y-direction velocity result by using high error time-steps

Figure 87- Non-cascading results for y-direction velocity, sample of maximum error in whole process when high error time-steps are included

4.3.1.4. X-direction velocity model by using high error time-steps

Figure 88- Non-cascading results for x-direction velocity, sample of maximum error in whole process when high error time-steps are included

4.4. Cascading result

Non-cascading results which were presented before prove that NN model has this capability to learn the entire process (3000 time-steps) by using 30 time-steps. It means that by selecting any time-step within the whole process and consider the CFD related value as an input the model is able to predict the next target value with high accuracy. In the other words, the input is coming from CFD and the output is the next time-step predicted by ANN. For cascading deployment, just the data from one time step from CFD is necessary. Now, what will happen if the output of the NN model is used as an input of the next time-step. In this type of deployment, there is no dependency on CFD data for each time-step.

Since the purpose of the cascading is the removal of any dependency on CFD data, four models are needed (phase fraction, pressure, x-direction velocity and y-direction velocity) for deploying each parameter. For starting the deployment of cascading, each of the variables from CFD could be used. For example, we started the deployment from time-step 500 to see how many time-steps could be predicted. In the following steps, the result of cascading for each parameter are shown.

4.5. Cascading result for phase fraction

Phase fraction prediction up to 25 time-steps in cascading condition are shown in following figures (Figure 89 to Figure 95). Time-step 500 is used to begin the deployment afterward the model predicted time-step 501 to 525. The error is propagating after each time-step and the prediction is not acceptable after a few time-steps. The red points in right-hand side of figure indicate at least 10% error.

Figure 89- Cascading results of phase fraction at time-step 501

Figure 90- Cascading results of phase fraction at time-step 502

Figure 91- Cascading results of phase fraction at time-step 505

Figure 92- Cascading results of phase fraction at time-step 510

Figure 93- Cascading results of phase fraction at time-step 515

Figure 94- Cascading results of phase fraction at time-step 520

Figure 95- Cascading results of phase fraction at time-step 525

4.6. Cascading result for pressure

Pressure prediction after 20 time-steps in cascading condition are shown in following figures (Figure 96 to Figure 101). Time-step 500 is used to begin the deployment afterward the model predicts time-step 501 to 520. The error is propagating after each time-step and the prediction is not acceptable after couples of time-steps. The red points in right-hand side of figure indicates at least 10% error.

Figure 96- Cascading results of pressure at time-step 501

Figure 97- Cascading results of pressure at time-step 502

Figure 98- Cascading results of pressure at time-step 505

Figure 99- Cascading results of pressure at time-step 510

Figure 100- Cascading results of pressure at time-step 515

Figure 101- Cascading results of pressure at time-step 520

4.7. Cascading result for x-direction velocity

X-direction velocity prediction after 20 time-steps in cascading condition are shown in following figures (Figure 102 to Figure 107). Time-step 500 is used to begin the deployment afterward the model predicts time-step 501 to 520. The error is propagating after each time-step and the prediction is not accurate enough after couples of time-steps. The red points in right-hand side of figure indicate at least 10% error.

Figure 102- Cascading results of x-direction velocity at time-step 501

Figure 103- Cascading results of x-direction velocity at time-step 502

Figure 104- Cascading results of x-direction velocity at time-step 505

Figure 105- Cascading results of x-direction velocity at time-step 510

Figure 106- Cascading results of x-direction velocity at time-step 515

Figure 107- Cascading results of x-direction velocity at time-step 520

4.8. Cascading result for y-direction velocity

Y-direction velocity prediction after 20 time-steps by ANN in cascading condition are shown in following figures (Figure 108 to Figure 113). Time-step 500 is used to begin the deployment afterward the model predicts time-step 501 to 520. The error is propagating after each time-step and the prediction is not acceptable after couples of time-steps. The red points in right-hand side of figure indicate at least 10% error.

Figure 108- Cascading results of y-direction velocity at time-step 501

Figure 109- Cascading results of y-direction velocity at time-step 502

Figure 110- Cascading results of y-direction velocity at time-step 505

Figure 111- Cascading results of y-direction velocity at time-step 510

Figure 112- Cascading results of y-direction velocity at time-step 515

Figure 113- Cascading results of y-direction velocity at time-step 520

4.9. Reduction of data

As is mentioned previously, cascading results after a few time-steps are associated with notiable amount of errors. It is tried to change the data distribution by removing the similar data in the learning process. Non-cascading and cascading result as well as histogram of data before and after elimination are presented in following steps.

4.9.1. Non-cascading

4.9.1.1.Phase-fraction

If the amount of changes for each grid, between the two consecutive time-steps in the entire process is less than 0.001, the data are removed from training process (Table 12).

Table 12- Phase fraction model specifications after elimination

Difference	< 0. 001
Number of total grids	40,000
Constant girds (eliminated)	13327
Non-stationary grids	26673
% of elimination	%34

The black shadows indicate eliminated grids, which are approximately constant during entire process (Figure 114).

Figure 114- Eliminated grids for phase fraction

Figure 115- Histogram of phase fraction data after elimination

Non-cascading result is still good and the error is always less than 5%. Some sample of non-cascading result are shown in following figures (Figure 116).

Figure 116- Sample of non-cascading results for phase fraction based on non-stationary grids value

4.9.1.2. Pressure

If the amount of change for each grid, between the two consecutive time-steps in the entire process is less than 50 Pascal (% 0.1), that grids information are removed from training process. (Table 13)

Table 13- Pressure model specifications after elimination

Difference	< 0.001 or <50 Pascal
Number of total grids	40,000
Constant grids (eliminated)	17235
Non-stationary grids	22765
% of elimination	%43

The black shadows indicate those eliminated grids, which are almost constant during entire process (Figure 117).

Figure 117- Eliminated grids for pressure

Figure 118- Histogram of pressure data after elimination

The non-cascading result for pressure is acceptable as well (Figure 119).

Figure 119- Sample of non-cascading results for pressure based on non-stationary grids value

4.9.1.3. X-direction velocity

If the amount of changes for each grid, between the two consecutive time-steps in the entire process is less than 0.15 m/s (%0.1), that grids information are removed from training process. (Table 14).

Table 14- X-direction velocity model specifications after elimination

Difference	< 0.001 or <0.15 m/s
Number of total grids	40,000
Constant grids (eliminated)	10336
Non-stationary grids	29664
% of elimination	%26

The black points indicate those eliminated grids, which are almost constant during entire process (Figure 120).

Figure 120- Eliminated grids for x-direction velocity

Figure 121- Histogram of x-direction data after elimination

The non-cascading result for x-direction velocity is acceptable as well (Figure 122).

Figure 122- Sample of non-cascading results for x-direction velocity based on non-stationary grids value

4.9.1.4. Y-direction velocity

If the amount of changes for each grid, between the two consecutive time-steps in the entire process is less than 0.15 m/s ($\%0.1$), that grid's information are removed from training process. (Table 15).

Table 15- Y-direction velocity model specifications after elimination

Difference	< 0.001 or < 0.15 m/s
Number of total grids	40,000
Constant grids (eliminated)	10214
Non-stationary grids	29786
% of elimination	%26

The black points indicate those eliminated grids, which are almost constant during entire process (Figure 123).

Figure 123- Eliminated grids for y-direction velocity

Figure 124- Histogram of y-direction data after elimination

The non-cascading result for y-direction velocity is acceptable as well (Figure 125).

Figure 125- Sample of non-cascading results for x-direction velocity based on non-stationary grids value

4.9.2. Cascading

The cascading result of phase fraction is shown as a showcase and RMSE graphs will be presented in order to compare the amount of error after each time-step. Cascading deployment is started from time-step 1000 to time-step 1020. The figures are presented every 5 time-steps (Figure 126 to Figure 130).

Figure 126- Cascading results of phase fraction at time-step 1001 based on non-stationary grids value

Figure 127- Cascading results of phase fraction at time-step 1005 based on non-stationary grids value

Figure 128- Cascading results of phase fraction at time-step 1010 based on non-stationary grids value

Figure 129- Cascading results of phase fraction at time-step 1015 based on non-stationary grids value

Figure 130- Cascading results of phase fraction at time-step 1020 based on non-stationary grids value

The RMSE graphs illustrates that the error in cascading deployment is increasing linearly after each time-step (Figure 131).

Figure 131- Cascading RMSE for phase fraction when entire process (all time-steps) are used for training

4.9.3. Make a model based on 200 time-steps

In this idea, it is focused just on 200 time-steps instead of entire process in order to achieve a better non-cascading model. So in every 20 time-steps from time-step 1000 to 1200, constant grids have been eliminated. 11 time-steps selected for non-cascading training (Figure 132).

Figure 132- Focus on every 20 time-steps

The phase fraction results are shown Figure 131. The following is the procedure used for phase fraction. For the other variables, the procedure is the same.

- Constant grids are eliminated.
- Number of total grid= 40,000 grids
- Constant grids (eliminated)=18898 grids
- Non-stationary grids= 21102 grids
- Percentage of elimination = %47

Figure 133- Black points are eliminated grids

The cascading deployment of phase fraction are started from time-step 1000 to 1050. Results in every 5 time-step are presented in Figure 134 to Figure 144.

Figure 134- Cascading results of phase fraction at time-step 1001 (focused on 200 time-steps)

Figure 135- Cascading results of phase fraction at time-step 1005 (focused on 200 time-steps)

Figure 136- Cascading results of phase fraction at time-step 1010 (focused on 200 time-steps)

Figure 137- Cascading results of phase fraction at time-step 1015 (focused on 200 time-steps)

Figure 138- Cascading results of phase fraction at time-step 1020 (focused on 200 time-steps)

Figure 139- Cascading results of phase fraction at time-step 1025 (focused on 200 time-steps)

Figure 140- Cascading results of phase fraction at time-step 1030 (focused on 200 time-steps)

Figure 141- Cascading results of phase fraction at time-step 1035 (focused on 200 time-steps)

Figure 142- Cascading results of phase fraction at time-step 1040 (focused on 200 time-steps)

Figure 143- Cascading results of phase fraction at time-step 1045 (focused on 200 time-steps)

Figure 144- Cascading results of phase fraction at time-step 1050 (focused on 200 time-steps)

The RMSE graph is presented in Figure 145 from time-step 1000 to 1050.

Figure 145- Cascading RMSE for phase fraction when time-steps 1000 to 1200 are used for training

The following graph (Figure 146) indicates that the result is more accurate when the model is made based on 200 time-step compared to one based on entire process in cascading deployment.

Figure 146- Comparing cascading results (entire process versus 200 time-steps)

Chapter 5 Conclusions and Recommendations

5.1. Conclusions

The main objective for this work is to predict the physical parameters such as velocity, pressure and phase fraction during the process along with reducing the computational time. DamBreak problem, which is one of the most well-known CFD problems, is selected in order to examine whether the neural network will be capable of predicting the properties. OpenFOAM package is used for simulating the problem and generating the data for building the neural network model. After performing several ideas, the last model designed based on 24 parameters and 40,000 number of grids.

Two scenarios are considered: first one which is called Non-cascading, was to build a model for each property that is able to predict the results at any time steps using the information from the previous time-step. The second scenario which is called cascading, was to build a model that is able to use CFD results in a time step and predict several following time-steps.

For the first scenario, the results indicated that neural network model is able to predict the results with high accuracy and is perfectly matched with the CFD results. For the cascading, the model is able to predict up to around 40 time-steps after input time-step with acceptable error (10%).

Reducing the time process compared to CFD was the main objective of the research, so this process takes 8 hours with four CPUs when it is done by CFD and takes 2 minutes by smart proxy with one CPU.

5.2. Recommendations and future works

Better results in cascading condition will be gained by a better non-cascading model. The non-cascading result was much closer to the CFD data and it is acceptable for that condition, but for cascading result after several time-steps will fail due to error propagation. Building a perfect non-cascading model leads to a better cascading model.

In the following of this context, focusing on cascading deployment, changing the geometry and applying the model with different fluid properties are some of the future works could be performed.

Bibliography

- [1] A. Alizadehdakhel, M. Rahimi, J. Sanjari, and A. Abdulaziz, "CFD and artificial neural network modeling of two-phase flow pressure drop ☆," vol. 36, pp. 850–856, 2009.
- [2] S. Esmaili and S. D. Mohaghegh, "Geoscience Frontiers Full field reservoir modeling of shale assets using advanced data-driven analytics," *Geosci. Front.*, vol. 7, no. 1, pp. 11–20, 2016.
- [3] A. Kalantari-dahaghi, S. Mohaghegh, and S. Esmaili, "Journal of Natural Gas Science and Engineering Coupling numerical simulation and machine learning to model shale gas production at different time resolutions," *J. Nat. Gas Sci. Eng.*, vol. 25, pp. 380–392, 2015.
- [4] N. A. Patankar and D. D. Joseph, "Modeling and numerical simulation of particulate flows by the Eulerian ± Lagrangian approach," vol. 27, pp. 1659–1684, 2001.
- [5] Z. Chen and G. Huan, *No TComputational Methods for Multiphase Flows in Porous Media* title. 2006.
- [6] P. Higuera, J. L. Lara, and I. J. Losada, "Three-dimensional interaction of waves and porous coastal structures using OpenFOAM ® . Part I : Formulation and validation," *Coast. Eng.*, vol. 83, pp. 243–258, 2014.
- [7] H. Ozmen-Cagatay and S. Kocaman, "Dam-break flow in the presence of obstacle: Experiment and CFD simulation," *Eng. Appl. Comput. Fluid Mech.*, vol. 5, no. 4, pp. 541–552, 2011.
- [8] C. Biscarini, S. Di Francesco, and P. Manciola, "Sciences CFD modelling approach for dam break flow studies," pp. 705–718, 2010.
- [9] I. E. Barton, "COMPARISON OF SIMPLE- AND PISO-TYPE ALGORITHMS FOR TRANSIENT FLOWS," vol. 26, no. February 1997, pp. 459–483, 1998.
- [10] S. ; Mohaghegh, Shahab D; Ameri, "SPE 29220 Artificial Neural Network As A Valuable Tool For Petroleum Engineers," *SPE Pet. Comput. Conf.*, 1995.
- [11] S. Mohaghegh and W. V. U, "Virtual-Intelligence Applications in Petroleum Engineering : Part 1 — Artificial Neural Networks," *Soc. Pet. Eng.*, no. September, pp. 64–71, 2000.
- [12] E. Alpaydin, *Introduction to Machine Learning Second Edition*. Massachusetts Institute of Technology, 2010.
- [13] P. W. Hogg, X. Gu, and D. R. Emerson, "AN IMPLICIT ALGORITHM FOR CAPTURING SHARP FLUID INTERFACES IN THE VOLUME OF FLUID ADVECTION METHOD," pp. 1–20, 2006.
- [14] K. Madsen and H. B. Nielsen, "Introduction to Optimization and Data Fitting," no. August, 2010.
- [15] S. Dean, "Journal of Natural Gas Science and Engineering Reservoir simulation and modeling based on artificial intelligence and data," *J. Nat. Gas Sci. Eng.*, vol. 3, no. 6, pp. 697–705, 2011.

Appendix

Appendix 1: OpenFOAM DamBreak Code

```
17  convertToMeters 0.146;
18
19  vertices
20  (
21      (0 0 0)
22      (2 0 0)
23      (2.16438 0 0)
24      (4 0 0)
25      (0 0.32876 0)
26      (2 0.32876 0)
27      (2.16438 0.32876 0)
28      (4 0.32876 0)
29      (0 4 0)
30      (2 4 0)
31      (2.16438 4 0)
32      (4 4 0)
33      (0 0 0.1)
34      (2 0 0.1)
35      (2.16438 0 0.1)
36      (4 0 0.1)
37      (0 0.32876 0.1)
38      (2 0.32876 0.1)
39      (2.16438 0.32876 0.1)
40      (4 0.32876 0.1)
41      (0 4 0.1)
42      (2 4 0.1)
43      (2.16438 4 0.1)
44      (4 4 0.1)
45  );
46
47  blocks
48  (
49      hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1)
50      hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1)
51      hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1)
52      hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1)
```



```

53     hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1)
54 );
55
56 edges
57 (
58 );
59
60 boundary
61 (
62     leftWall
63     {
64         type wall;
65         faces
66         (
67             (0 12 16 4)
68             (4 16 20 8)
69         );
70     }
71     rightWall
72     {
73         type wall;
74         faces
75         (
76             (7 19 15 3)
77             (11 23 19 7)
78         );
79     }
80     lowerWall
81     {
82         type wall;
83         faces
84         (
85             (0 1 13 12)
86             (1 5 17 13)
87             (5 6 18 17)
88             (2 14 18 6)
89             (2 3 15 14)
90         );
91     }
92     atmosphere
93     {

```

```

94         type patch;
95         faces
96         (
97             (8 20 21 9)
98             (9 21 22 10)
99             (10 22 23 11)
100         );
101     }
102 );
103
104 mergePatchPairs
105 (
106 );
107
setFields
18 defaultFieldValues
19 (
20     volScalarFieldValue alpha.water 0
21 );
22
23 regions
24 (
25     boxToCell
26     {
27         box (0 0 -1) (0.1461 0.292 1);
28         fieldValues
29         (
30             volScalarFieldValue alpha.water 1
31         );
32     }
33 );
34
Discretisation schemes
ddtSchemes
19 {
20     default          Euler;
21 }
22
23 gradSchemes
24 {

```

```

25     default          Gauss linear;
26 }
27
28 divSchemes
29 {
30     div(rhoPhi,U)  Gauss linearUpwind grad(U);
31     div(phi,alpha)  Gauss vanLeer;
32     div(phirb,alpha) Gauss linear;
33     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
34 }
35
36 laplacianSchemes
37 {
38     default          Gauss linear corrected;
39 }
40
41 interpolationSchemes
42 {
43     default          linear;
44 }
45
46 snGradSchemes
47 {
48     default          corrected;
49 }
50

```

Blocks definition

```

blocks
(
    hex (0 1 5 4 12 13 17 16) (46 10 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (40 10 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (46 76 1) simpleGrading (1 2 1)
    hex (5 6 10 9 17 18 22 21) (4 76 1) simpleGrading (1 2 1)
    hex (6 7 11 10 18 19 23 22) (40 76 1) simpleGrading (1 2 1)
);

```

Appendix 2: MATLAB Code

Training code

```
clc
% clear;
close all;
%% Input and Output Time-step

Time_Step_in=2999;
Time_Step_out=Time_Step_in+1;

%% Defining the geometry
imax = 200;           % cell in x-direction
kmax = 1;             % cell in z-direction
jmax = 200;           % cell in y-direction
Dim_x = 4;            %length
Dim_y = 4;            %height
Dim_z = .1;           %width
Dim_cell = Dim_x/imax; %dimension of cell,
CellNo=imax*jmax*kmax; %total number of cells

%% Loading the CSV files from time-step start to final

data_input = ReadFiles( Time_Step_in,CellNo );
data_output = ReadFiles( Time_Step_out,CellNo );

%% Initialize the matrices
location = zeros(CellNo,2);
CellID = zeros(CellNo,1); % ID of each cell
tier1 = zeros(CellNo,4); % tier system with of order 1
(bottom,top,west,north,east,south)
DisToBoundary = zeros(CellNo,4); % distance to the boundaries
(bottom,top,west,north,east,south)
counter = 1;

%% Filling the geomtry matrices
for k = 1:kmax
    for j = 1:jmax
        for i = 1:imax
            location(counter,:) = [i j];
            CellID(counter) = (j-1)*imax+i;
            %cell ID of tier cells(left,up,right,down)
            tier1(counter,:) = [(j-1)*imax+i-1 (j)*imax+i (j-1)*imax+i+1 (j-
2)*imax+i];
            %distance of center of each cell to the
boundaries(left,up,right,down)
            DisToBoundary(counter,:) = [(i-0.5)*Dim_cell Dim_y-(j-0.5)*Dim_cell
Dim_x-(i-0.5)*Dim_cell (j-0.5)*Dim_cell ];
            %% filling zero at for the tiers for the cells at the boundary
            if i==1
                tier1(counter,1) = 0;
            elseif i==imax
                tier1(counter,3) = 0;
            end
        end
    end
end
```

```

        if j==1
            tier1(counter,4) = 0;
        elseif j==jmax
            tier1(counter,2) = 0;
        end

        counter=counter+1;
    end
end
end

tierData=zeros(CellNo,5*4);
for i = 1:CellNo
    tierData(i,1:4)=data_input(i,:);
    for p = 1:4
        index = tier1(i,p);
        if index==0
            t = zeros(1,4);
        else
            t = data_input(index,:);
        end
        tierData(i,4*p+1:4*(p+1)) = t;
    end
end

NN_input=[DisToBoundary tierData];
NN_input=NN_input';

alpha_water=data_output(:,1);
p_rgh=data_output(:,2);
Velocity_x=data_output(:,3);
Velocity_y=data_output(:,4);

alpha_water= alpha_water';
p_rgh=p_rgh';
Velocity_x=Velocity_x';
Velocity_y=Velocity_y';

```

time-step selecting

```
%% run one time
NN_input4=[];
Velocity_x4=[];
%% run each time-step
NN_input4=[NN_input4 NN_input];
Velocity_x4=[Velocity_x4 Velocity_x];

function [ data ] = ReadFiles( start,CellNo )

FileName=strcat('200.',num2str(start),'.csv');
data= csvread(FileName, 1, 0,[1,0,CellNo,3]);

end
```

Deployment code

```
for      d=1000:1:1050

    %% Input and Output Time-step

    Time_Step_in=d;
    Time_Step_out=d+1;

    %% Defining the geometry
    imax = 200;
    kmax = 1;
    jmax = 200;
    Dim_x = 4;
    Dim_y = 4;
    Dim_z = .1;
    Dim_cell = Dim_x/imax;      CellNo=imax*jmax*kmax;

    %% Loading the CSV files from time-step start to final

    data_input = ReadFiles( Time_Step_in,CellNo );
    data_output = ReadFiles( Time_Step_out,CellNo );

    location = zeros(CellNo,2);      %(x,y,z) of each cell
    CellID = zeros(CellNo,1);        % ID of each cell
    tier1 = zeros(CellNo,4);         % tier system
    (bottom,top,west,north,east,south)
    DisToBoundary = zeros(CellNo,4); (bottom,top,west,north,east,south)
    counter = 1;
```

```

%% Filling the geomtry matrices
for k = 1:kmax
    for j = 1:jmax
        for i = 1:imax
            location(counter,:) = [i j];
            CellID(counter) = (j-1)*imax+i;
            %cell ID of tier cells(left,up,right,down)
            tier1(counter,:) = [(j-1)*imax+i-1 (j)*imax+i (j-1)*imax+i+1 (j-
2)*imax+i];
            %distance of center of each cell to the
boundaries(left,up,right,down)
            DisToBoundary(counter,:) = [(i-0.5)*Dim_cell Dim_y-(j-0.5)*Dim_cell
Dim_x-(i-0.5)*Dim_cell (j-0.5)*Dim_cell ];
            %% filling zero at for the tiers for the cells at the boundary
            if i==1
                tier1(counter,1) = 0;
            elseif i==imax
                tier1(counter,3) = 0;
            end

            if j==1
                tier1(counter,4) = 0;
            elseif j==jmax
                tier1(counter,2) = 0;
            end

            counter=counter+1;
        end
    end
end

tierData=zeros(CellNo,5*4);
for i = 1:CellNo
    tierData(i,1:4)=data_input(i,:);
    for p = 1:4
        index = tier1(i,p);
        if index==0
            t = zeros(1,4);
        else
            t = data_input(index,:);
        end
        tierData(i,4*p+1:4*(p+1)) = t;
    end
end

NN_input=[DisToBoundary tierData];
NN_input=NN_input';

alpha_water=data_output(:,1);

```

```

p_rgh=data_output(:,2);
Velocity_x=data_output(:,3);
Velocity_y=data_output(:,4);

alpha_water= alpha_water';
p_rgh=p_rgh';
Velocity_x=Velocity_x';
Velocity_y=Velocity_y';

% property=input('Property: (1=alpha water, 2=p_rgh, 3=Velocity_x,
4=Velocity_y,):' );
property=1
if property==1
    Title='alpha water';
    load('network_alphawater.mat') %network name
    output_real=alpha_water; % your name
    tic
    output_NN=network_alphawater(NN_input);
    toc
elseif property==2
    Title='Pressure';
    load('network_pressure.mat')
    output_real=p_rgh;
    tic
    output_NN=network_pressure(NN_input);
    toc
elseif property==3
    Title='X_velocity';
    load('network_xvelocity.mat')
    output_real=Velocity_x;
    tic
    output_NN=network_xvelocity(NN_input);
    toc

elseif property==4
    Title='Y_velocity';
    load('network_yvelocity.mat')
    output_real = Velocity_y;
    tic
    output_NN=network_yvelocity(NN_input);
    toc

end

%% Error

if property==1
    error=abs((output_real-output_NN))*100;
else
    output_real_normal=(output_real-mean(output_real))/(max(output_real)-
min(output_real));
    output_NN_normal=(output_NN-mean(output_NN))/(max(output_NN)-
min(output_NN));

```



```

        error=abs((output_real_normal-output_NN_normal))*100;
        mse=mean(error.^2);
end

%% Plotting

t=zeros(imax,jmax);
CellID_locator=zeros(imax,jmax);
subplot(1,3,1)
for i=1:jmax
    for j=1:imax
        CellID_locator(i,j)=(i-1)*jmax+j;
        t(i,j)=output_real(CellID_locator(i,j));
    end
end

t = [[t nan*zeros(size(t,1),1)] ; nan*zeros(1,size(t,2)+1)];
pcolor(t)
xlabel('x-division')
ylabel('y-division')
title('OpenFOAM result')
colorbar('eastoutside')
shading flat
Low=min(t(:));
Up=max(t(:));

t=zeros(imax,jmax);
CellID_locator=zeros(imax,jmax);
subplot(1,3,2)
for j=1:imax
    for i=1:jmax
        CellID_locator(i,j)=(k-1)*jmax*imax+(i-1)*jmax+j;
        t(i,j)=output_NN(CellID_locator(i,j));
    end
end

t = [[t nan*zeros(size(t,1),1)] ; nan*zeros(1,size(t,2)+1)];
pcolor(t)
xlabel('x-division')
ylabel('y-division')
title('Smart Proxy Result')
colorbar('eastoutside')
shading flat
Low=min(min(t(:)),Low);
Up=max(max(t(:)),Up);
caxis([Low, Up])

subplot(1,3,1)
caxis([Low, Up])

t=zeros(imax,jmax);
CellID_locator=zeros(imax,jmax);

```

```

subplot(1,3,3)
for i=1:imax
    for j=1:jmax
        CellID_locator(i,j)=(k-1)*jmax*imax+(i-1)*jmax+j;
        t(i,j)=error(CellID_locator(i,j));
    end
end

t = [[t nan*zeros(size(t,1),1)] ; nan*zeros(1,size(t,2)+1)];
pcolor(t)
xlabel('x-division')
ylabel('y-division')
title('Error Distribution (%)')
colorbar('eastoutside')
shading flat
caxis([0, 10])
colormap(jet); % color of the graphs
%subtitle(strcat(Title,' of Layer= ',num2str(plane_no(counter)),' at time-step= ',num2str(Time_Step_out)));
% print(strcat(Title,' at time-step= ',num2str(Time_Step_out)),'-dpng')
fig = gcf;
fig.PaperUnits = 'inches';
fig.PaperPosition = [0 0 14 3];
print(strcat(Title,' at time-step= ',num2str(Time_Step_out)),'-dpng','-r0')

% flippedJetColorMap = flipud(jet(256));
% colormap(flippedJetColorMap);

End

reading the file

function [ data ] = ReadFiles( start,CellNo )

FileName=strcat('200.',num2str(start),'.csv');
data= csvread(FileName, 1, 0,[1,0,CellNo,3]);

end

```