VIABILITY OF PHYSICS-AWARE DEEP-LEARNING BASED PROXY RESERVOIR SIMULATOR FOR USE IN WELL CONTROL OPTIMIZATION

A Thesis

by

GEORGY KOMPANTSEV

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Eduardo Gildin
Committee Members,	Misra Siddharth
	Ulisses Braga-Neto
Head of Department,	Jenn-Tai Liang

August 2022

Major Subject: Petroleum Engineering

ABSTRACT

As oil production rises, and we are forced to undertake more challenging problems, the need for accurate and efficient reservoir simulations increases. Revitalizing old reservoirs and tackling more geologically complex reserves, require increasingly complex and computationally expensive reservoir models. It is necessary to develop new, more efficient alternatives to traditional reservoir simulation workflows to keep up with the demand in computational power. This study aims to explore the potential of neural network-based proxy reservoir simulators in the context of well control optimization, one of the most computationally demanding aspects of reservoir simulation.

The research objective is tested through an implementation of a particle swarm optimizer with an algorithm called E2CO – Embed to Control and Observe, a physics-aware neural network-based proxy reservoir simulator containing model input/output matching. The obtained results are analyzed and compared against the values obtained from a traditional reservoir simulator implemented on the same optimization framework.

The results show that implementation of E2CO with particle swarm optimization is a flexible and efficient alternative to traditional well control optimization workflows. Although not ideal, the algorithm provides acceptable accuracy, while delivering the results in a fraction of the time. Pairing this proxy model optimization framework with a traditional numerical simulator yields fast and sufficient results in a realistic workflow.

The testing methodology described in this research provides a vision for what AI can achieve when implemented in a reservoir simulation environment. Quantifying the potential improvements and highlighting areas that need additional exploration.

ACKNOWLEDGMENTS

I am extremely grateful to my advisor, Dr. Eduardo Gildin, for providing guidance and academic support through all the stages of my research, as well as his understanding and patience during a difficult time in my life.

A special thanks goes out to Emilio Coutinho for providing his expertise in reservoir simulation and proxy modelling.

Lastly, I would like to express my deepest gratitude to my family. Without their love and support, I would not have been able to undertake this academic endeavor.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This thesis has been guided by a committee consisting of Professor Eduardo Gildin (advisor) and Professor Siddharth Misra of the Harold Vance Department of Petroleum Engineering and Professor Ulisses Braga-Neto of the Department of Electrical and Computer Engineering.

This thesis is based on the work done by Emilio J. R. Coutinho; Marcelo J. Aqua and Eduardo Gildin on the Physics-Aware Deep-Learning-Based Proxy Reservoir Simulation Model discussed in the thesis.

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

All other work conducted in this thesis is done by the student.

Funding Sources

Graduate study was supported by a fellowship from the Harold Vance Department of Petroleum Engineering and a Research Fellowship from Professor Eduardo Gildin.

NOMENCLATURE

E2CO	Embed to Control and Observe
CMG	Computer Modelling Group
PSO	Particle Swarm Optimization
BHP	Bottom-Hole Pressure
MOR	Model Order Reduction
POD	Proper Orthogonal Decomposition
TPWL	Trajectory Piecewise Linearization
NPV	Net Present Value
IRR	Internal Rate of Return
AI	Artificial Intelligence
MSM3	Thousand Standard Cubic Meters
kgf/cm^2	Kilogram force per sqare centimeter
m^3/day	Thousand Standard Cubic Meters
PC	Personal Computer
GPU	Graphical Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES x

1. INTRODUCTION 1

    1.1 Motivation 1
    1.2 Review 1
    1.3 Objectives and Scope 3

2. BACKGROUND 4

    2.1 Numerical Reservoir Simulation 4
    2.2 E2CO – Embed to Control and Observe 5
        2.2.1 High Level Overview of E2CO 5
    2.3 Well Control Optimization 8
    2.4 Particle Swarm Optimization 9

3. TESTING DESIGN 14

    3.1 Methodology Overview 14
    3.2 PSO-E2CO Overview 15
        3.2.1 Initialization 15
        3.2.2 Model Loading and Normalization Setup 16
        3.2.3 IMEX interaction and E2CO data Preparation 16
        3.2.4 E2CO Prediction 17
        3.2.5 Result Extraction 18
        3.2.6 Particle Update 20
    3.3 PSO-IMEX Overview 20

4. RESULTS 21

4.1 General Parameters 22
4.2 Showcase Model 23
4.3 5-spot Model 27
4.4 Inverted 5-spot Model 30
4.5 Staggered Line Drive Model 33
4.6 Training Results 35
4.7 Compiled Results 37

5. DISCUSSION AND CONCLUSION 39

5.1 Discussion 39
5.2 Conclusion 40

REFERENCES 41

APPENDIX A. ADDITIONAL FIGURES 44

APPENDIX B. IMEX .dat file Example (5-Spot) 50

LIST OF FIGURES

FIGURE Page

2.1 Adapted from Coutinho et al. - E2CO Training procedure. Grey variables are inputs to the model. Vector x_t is encoded into latent space, z_t, where the model predicts the evolution in time via the transition network. Proposition 3 highlights the evolution of well controls in latent space. The model is assessed at each generated variable via a dedicated loss function. 6

2.2 Adapted from Coutinho et al. - E2CO Prediction procedure. Grey variables are the input variables for each time step in a simulation run. Each new predicted state variable in latent space z_{t+1} becomes the base latent variable z_t for each successive time step. 7

2.3 Well Control Optimization Graphic. Well Control parameters are adjusted to achieve the maximum cumulative oil production. 8

2.4 Particle Swarm Optimization Overview. Each particle updates its position by combining its own initial velocity component with velocity towards its previous best position and the swarm best position. 11

3.1 Optimization Combined Workflow Overview 14

3.2 PSO-E2CO Overview. From initialization to finding a solution. 15

3.3 E2CO Prediction loop within PSO-E2CO (Modified from Coutinho et al.) 17

3.4 Rate Adjustment Example. Staggered-Line Drive (Validation Model 14) 19

3.5 PSO-IMEX Workflow 20

4.1 All Models Tested 22

4.2 Validation Set Cumulative Production Comparison (Showcase) 24

4.3 Example Injector BHP Match (Showcase, Validation Model 1, Injector 1) 24

4.4 Showcase Optimization Results Comparison (E2CO in Blue, IMEX in Orange) 25

4.5 Validation Set Cumulative Production Comparison (5-Spot) 28

4.6 5-Spot Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green) 28

4.7 Validation Set Cumulative Production Comparison (Inverted 5-Spot) 30

4.8 Inverted 5-Spot Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green) 31

4.9 Validation Set Cumulative Production Comparison (Staggered Line Drive) 34

4.10 Staggered Line Drive Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green) 34

4.11 Runtime Comparison. Computation time comparison between PSO-IMEX, PSO-E2CO, and PSO-E2CO Adjusted 38

A.1 Showcase model with permeability realization (Permeability Units: md) 44

A.2 5-Spot model with permeability realization (Permeability Units: md) 45

A.3 Inverted 5-spot model with permeability realization (Permeability Units: md) 46

A.4 Staggered Line Drive model with permeability realization (Permeability Units: md). 47

A.5 Encoder Neural Network Structure. Yellow: Encoding Block, Red: Residual Convolution Block, Green: Dense Layer (Adapted from Coutinho et al.) 47

A.6 Decoder Neural Network Structure. Grey: Convolutional 2D Layer, Blue: Decoding Block, Red: Residual Convolutional Block, Green: Dense Layer (Adapted from Coutinho et al.) 48

A.7 Transition Neural Network Structure. Purple: Transformation Block, Green: Dense Layer (Adapted from Coutinho et al.) 48

A.8 Transition Well Data Neural Network Structure. Purple: Transformation Block, Green: Dense Layer (Adapted from Coutinho et al.) 49

A.9 Encoding Block Structure (Adapted from Coutinho et al.) 49

A.10 Residual Convolution Block Structure (Adapted from Coutinho et al.) 49

A.11 Decoding Block Structure (Adapted from Coutinho et al.) 49

A.12 Transformation Block Structure (Adapted from Coutinho et al.) 49

LIST OF TABLES

TABLE Page

4.1 PSO parameter values used. 23

4.2 Showcase Model Parameters 23

4.3 Showcase Model Results 26

4.4 Showcase Optimization Performance 26

4.5 5-Spot Model Parameters 27

4.6 5-Spot Model Results 29

4.7 5-Spot Optimization Performance 29

4.8 Inverted 5-Spot Model Parameters 30

4.9 Inverted 5-Spot Model Results 32

4.10 Inverted 5-Spot Optimization Performance 32

4.11 Staggered Line Drive Model Parameters 33

4.12 Staggered Line Drive Model Results 35

4.13 Staggered Line Drive Optimization Performance 35

4.14 Training Results 36

4.15 PC Specifications 36

4.16 Overall Cumulative Oil Production Performance 37

4.17 Overall Optimization Runtime Performance 37

1. INTRODUCTION

1.1 Motivation

Since the inception of reservoir simulation as a discipline in petroleum engineering [1], the industry has been catching up to provide necessary computational power for the ever more complex models. Modern reservoir models rely on a large number of gridblocks for accurate representation of fluid dynamics, and the fluid flow within each gridblock needs to be solved for across potentially millions of time steps. The issue is exasperated even further in the 21st century, as complex models are necessary to simulate flow within tight reservoirs, or help revitalize old, abandoned fields. According to the company Spokesperson, BP has recently invested over 100 million USD [2] to construct a state-of-the-art high performance computing facility to aid in its processing and simulation workflow. With the rate that at which current computing necessities are increasing, the center would need to be expanded over and over as the time goes by, as BP has already done [3]. New and creative solutions are necessary to cut the computational overhead and to assist in fast decision-making.

The environmental and monetary impact of computational demands is not the only motivation, time is also of the essence. The more complex the simulation, the more time is spent waiting for results. Time wasted waiting for the results from a simulation translates to additional costs for Oil and Gas companies around the globe. On average, the majority of oil producing countries require the cost of a barrel of oil to be above around 70 USD [4], and with the cyclical nature of the industry we often see dips below that mark. Creating solutions to help optimize the simulation process can also prove invaluable.

1.2 Review

For as long as reservoir simulation existed, researchers have been trying to come up with techniques to cut down on the computational complexity of the methods used. A number of techniques ranging from sparse iterative solvers [5] to implementation of preconditioners [6] have been im-

plemented with varying degrees of success. Among these techniques, a discipline called Model Order Reduction has found good application in reservoir simulation.

Model Order Reduction (MOR) is a collection of techniques aiming at reducing the computational complexity of simulations through the reduction of a model's associated state space (dimensions). In other words, taking a model from its original (larger) space and reducing it to a new, smaller space while maintaining sufficient detail without sacrificing too much accuracy in the result. Model order reduction is its own separate discipline, and there exist multiple techniques for performing the complexity reduction [7] [8], one such relatively new technique relies on the use of neural networks to perform the operation.

In their research paper, titled "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images" [9], the authors have devised a system for model learning and control of non-linear dynamical systems via the implementation of neural network encoders and decoders. These neural networks are trained to reduce the dimensions of raw image data and are utilized in a computational loop to predict the non-linear evolution of a system in latent (reduced) space. By utilizing neural networks to reduce the size of the models, the authors were able to maintain both sufficient speed and accuracy during the prediction of the evolution of the tested systems. However, this system lacked a direct link between inputs and outputs of a simulation. The algorithm was able to predict the evolution of a non-linear system, but does not tie in a connection between the input data and output data calculated from the evolved state of the system.

In 2020, the framework developed by Watter et al. has been adapted to reservoir simulation by Jin et al. in their paper titled "Deep-learning-based surrogate model for reservoir simulation with time-varying well controls" [10]. This system allowed for prediction of the evolution of pressure and fluid saturation within a reservoir subject to imposed well controls during oil production. The system could learn the behavior of fluids in a particular reservoir model from examples produced by traditional numerical simulators, and then use this knowledge to predict the evolution of state variables (pressure and fluid saturation) across multiple time steps more efficiently than traditional approaches.

The system developed by Jin at al., was then further expanded upon by Coutinho et al., in their paper titled "Physics-Aware Deep-Learning-Based Proxy Reservoir Simulation Model Equipped with State and Well Output Prediction" [11]. Where the team introduced additional expansions to the framework, allowing the algorithm to draw a connection between the evolution of the state variables and key model outputs (oil and water production rates). The system was dubbed "E2CO" or Embed to Control and Observe, and it has bridged the gap between model input and output, allowing for the prediction of Oil and Water flow rates at producing wells based on the specified well controls across multiple time-steps.

1.3 Objectives and Scope

The goal of this work is to take the newly developed E2CO system and apply it in the context of well control optimization. Well control optimization refers to the determination of the optimal well control parameters (e.g. bottom-hole pressure for producers, and injection rate for injectors) for the operation of the reservoir. The main objective of this study is to test the ability of E2CO to achieve reliable results in well control optimization, as well as to quantify its computational efficiency compared to traditional reservoir simulators.

During the course of this study, the author will implement both E2CO and an industry standard numerical reservoir simulator, into a proprietary optimization procedure, and compare the performance of the two algorithms. The study will be conducted on a range of different reservoir models with varying physical structures and production patterns to determine the accuracy and speed of the neural network approach, as well as determine the potential areas of improvement that need to be address before E2CO can be implemented in real-world workflow. If successful, the newly proposed method can help engineers increase the speed of their workflow, and give them an ability to consider more potential reservoir development strategies.

2. BACKGROUND

2.1 Numerical Reservoir Simulation

Numerical reservoir simulation describes the process by which engineers try to predict the behavior of fluids inside real reservoirs using physically accurate, mathematical models. These models are created based on the data obtained during the exploratory stage of oil and gas production. The fluid behavior is described by partial differential equations, such as the flow equation stated [12].

$$\nabla \left[\frac{\overrightarrow{k}\left(k_{r,j}\left(S_j\right)\right)}{\mu_j} \rho_j \nabla P_j \right] + q_j - \frac{\partial}{\partial t}\left(\varphi S_j \rho_j\right) = 0$$

To accurately describe the flow of fluids inside the reservoir, the model is divided into many smaller gridblocks. The fluid behavior over time is calculated per each gridblock and complied into the overall result at each time step [12].

For each individual time step in a simulation run, the fluid flow to/from each gridblock in each of the three fundamental direction (x, y, z) is considered. The flows are organized in a matrix form, and the solution is obtained by computing the residual of a Jacobian. A Jacobian is a matrix consisting of first partial derivatives of a particular differential equation (in our case, a fluid flow equation). Calculating the residual of a Jacobian is the most computationally intensive task in numerical reservoir simulation.

Numerical reservoir simulation demands high-end computational infrastructure. The complexity of the solution allows for reliable and mathematically sound results, but it also leads to inefficiencies. Traditional linear solvers can take up to 80% of the total runtime [13]. Even with the implementation of iterative solvers, once the simulation is large enough, the time commitment becomes hard to manage.

2.2 E2CO – Embed to Control and Observe

Embed to Control and observe is a numerical reservoir simulation proxy developed by Coutinho et. al. at Texas A&M University. E2CO builds on the work done by Jin, Liu and Durlofsky [10], where the authors have built a neural network architecture for the prediction of evolution of the state variables. In E2CO, an additional Embed to Control model is added to provide the connection between output variables (BHP and flow rate) and the state variables which allows the algorithm to predict the performance of a reservoir model with various input variables (BHP for producers, flow rate for injectors).

2.2.1 High Level Overview of E2CO

This section provides the general overview of the E2CO algorithm. As with any neural network algorithm today, the procedure is split up into training and prediction. Let's begin by describing the training portion.

For the initial preparation of the training algorithm, a set of completed reservoir simulations of varying construction (different input conditions) is necessary. The key variables need to be extracted for each time step of the simulation and arranged in vector form. The key variables are the control parameters for the wells, u^t (BHP for the producers, and injection rate for the injectors), pressure and oil saturation at each gridblock, x^t, output variables, y^t (oil and water flow rate for the producers, and injection rate for the injectors), and the time steps for the simulation, Δt^t.

$$u^t = \begin{bmatrix} \text{BHP}^t \\ q^t_{\text{inj}} \end{bmatrix} \qquad y^t = \begin{bmatrix} q^t_o \\ q^t_w \\ \text{BHP}^t \end{bmatrix}$$

$$x^t = \begin{bmatrix} p^t \\ s^t \end{bmatrix} \qquad \Delta t^t = \begin{bmatrix} t^1 \\ t^2 \\ \dots \end{bmatrix}$$

The state of the system (vector of pressures and saturations) is fed through a 2-D multi-layered neural network "Encoder" that reduces the original data, x^t, to a latent space form, z^t. The transformed data set is smaller than the original, the size reduction can be specified by the user. The variable in the latent space is then used to generate the transition matrices A_t and B_t via the "Transition" network through a deep neural analysis based on the Proper Orthogonal Decomposition - Trajectory Piece-wise Linearization (POD-TPWL) technique as described by Jin et al. The matrices are then applied to the latent space data alongside the input control variables and the time step to calculate the state variables in latent space for the next time step. Calculated data at the new time step is then fed through a decoder, which can be thought of as the reverse of the encoder originally used to generate the latent space data.

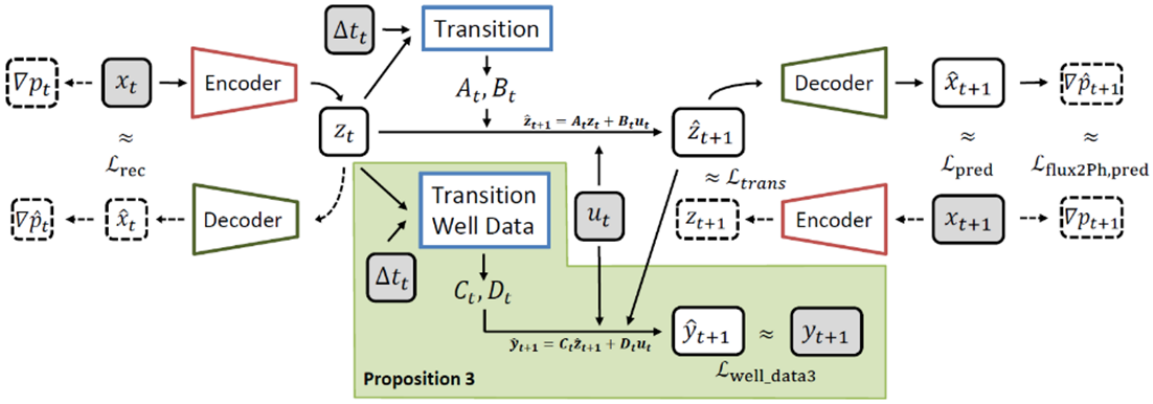

Figure 2.1: Adapted from Coutinho et al. - E2CO Training procedure. Grey variables are inputs to the model. Vector x_t is encoded into latent space, z_t, where the model predicts the evolution in time via the transition network. Proposition 3 highlights the evolution of well controls in latent space. The model is assessed at each generated variable via a dedicated loss function.

In the work done by Coutinho et al, an additional "Transition Well Data" network is added for the prediction of the output variables. The state variables in the latent space at each time step are used to generate two more transition matrices, C_t and D_t, that are used along with the input variables and time steps to calculate the output vector y_t for each well.

All the calculated variables in the algorithm are compared against their truth counterparts from

the numerical simulation results though a set of pre-defined loss functions. As with all the modern neural networks used today, the goal of the training procedure is to minimize the loss along every metric tested.

Upon the completion of the training procedure, once the loss functions are sufficiently low, the network is ready for the prediction stage. The prediction loop is a leaner, augmented version of the training procedure. In the prediction stage, only the initialized numerical model is necessary to obtain the initial state variables and the well controls. The state variables are transformed to the latent space, and the latent space variables are calculated at each time step through the same procedure used during the training. As opposed to training, during the prediction stage, the newly calculated latent state variables are used to calculate the latent space variables at each proceeding time step. The output variables are calculated simultaneously in the same manner as in the training procedure.

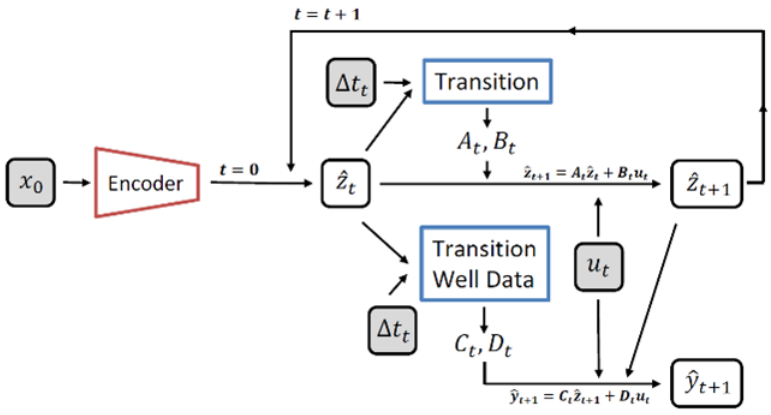

Figure 2.2: Adapted from Coutinho et al. - E2CO Prediction procedure. Grey variables are the input variables for each time step in a simulation run. Each new predicted state variable in latent space z_{t+1} becomes the base latent variable z_t for each successive time step.

The reliance on the latent space state variables allows the algorithm to quickly predict the output parameters of the simulation at a slight cost in accuracy, as will be discussed further in this thesis.

2.3 Well Control Optimization

Reservoir simulation is a powerful tool for cost-efficient development of an oil and gas field. Drilling a well is expensive, and if we can calculate which well location is going to be the most profitable, we can avoid losing potential profits. However, figuring out the location of the well is only part of the problem, we need to know how to operate that well to be profitable and sustainable, this is done through a process called well control optimization. Figure 2.3 gives a pictorial representation of well control optimization.

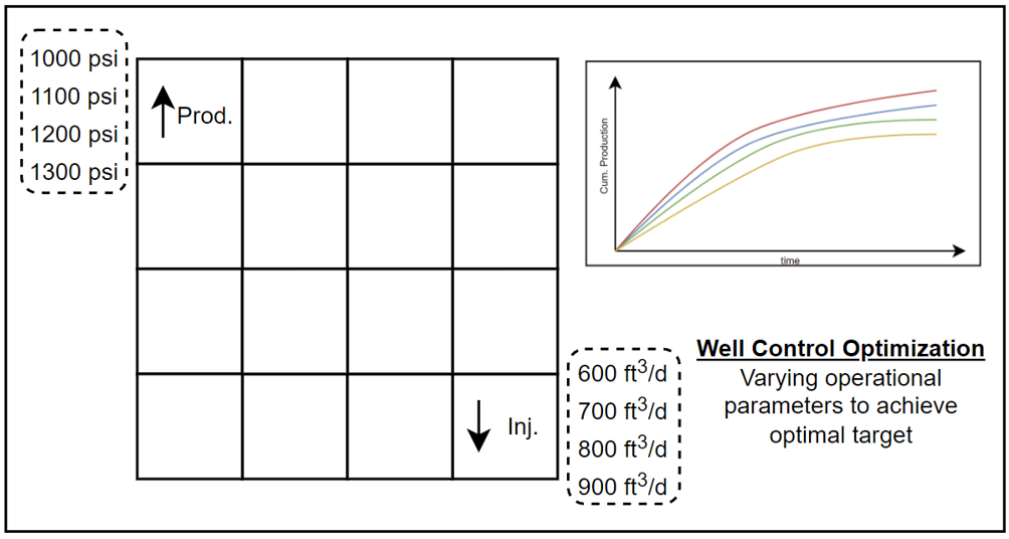

Figure 2.3: Well Control Optimization Graphic. Well Control parameters are adjusted to achieve the maximum cumulative oil production.

Well control optimization comes at the end of the reservoir simulation design, once all the critical variables have been estimated and the reservoir model is functional. However, it is often the most time-consuming step. Well control optimization refers to the determination of the optimal well control parameters (BHP/rates for producers, and injection rate for injectors) for the operation of the reservoir. The reservoir engineering team must run multiple studies on different scenarios before presenting the data to management for determining the development strategy. Each strategy employs a different objective function with various constraints, maximizing short term production,

long term production, maintaining constant BHP on the injector, maximizing NPV, maximizing IRR to name a few. Each one of those studies requires the optimization algorithm to run a full reservoir simulation model multiple times. Depending on the convergence criteria, the algorithm may need to call the simulator several hundred times to achieve a solution. When talking about modern reservoir models, with potentially billions of gridblocks, each requiring a full Jacobian solution at each time step, the computational cost stacks ups. Thus, running an optimization cycle using proxy models is a better solution.

2.4 Particle Swarm Optimization

There exist a multitude of modern optimization procedures that are currently in use across all engineering disciplines. All of those optimization algorithms achieve the same goal; To determine the best element, or a set of elements, to satisfy a predetermined criterion. In other words, given a function f with a set of elements E, determine an element $x_0 \in E$ such that $f(x_0)$ is the best element out of all the elements in E.

In order to implement well control optimization in our study, an optimization algorithm must be introduced. A number of optimization algorithms exist that fit the problem requirements [14], [15]. We have opted to go with Particle Swarm Optimization [16] for our study, due to its flexibility, ease of implementation, and availability on related platforms for comparison of results.

Particle Swarm Optimization (PSO) is a methodology in computational science used for the purpose of optimization of nonlinear functions. Initially proposed by Kennedy et al., the technique was aimed at simulation of social behavior, stemming from observations of natural processes such as bird flocking. The technique aims to mimic the behavior of organized packs of animals and apply the idea to mathematical functions of any complexity.

As the name suggests, PSO uses a set of discrete particles to search for a solution in N-dimensional space. The space that the particles occupy, represents the values of the variables of the function of interest. For each particle in the "swarm", the solution to the function at a particular point in space is the value of the function with the specific variables. The unique aspect of PSO, and the aspect that is mimicked from natural processes, is the communication between the

particles. The particles share several common variables that allow the individuals to act in unison to come up with a solution. Each particle operates with the knowledge of its current position, its solution, and the "best" or optimal solution of its previous attempts, as well as the optimal solution of the entire swarm. This allows the entire swarm of particles to narrow in on the optimal solution of the entire function.

The particle swarm algorithm is initialized by creating a set of particles, each having a random position and velocity in each of the dimensions considered. The value of the particle, or its fitness, is calculated with accordance to the function in question and stored in the memory. The fitness and the position of each particle is compared to the previously stored best (zero on the first iteration) and is overwritten once a better candidate is found. Each particle in the swarm now has access to the position and fitness of itself, it's previous best (these values are the same for the first generation), and the swarm best.

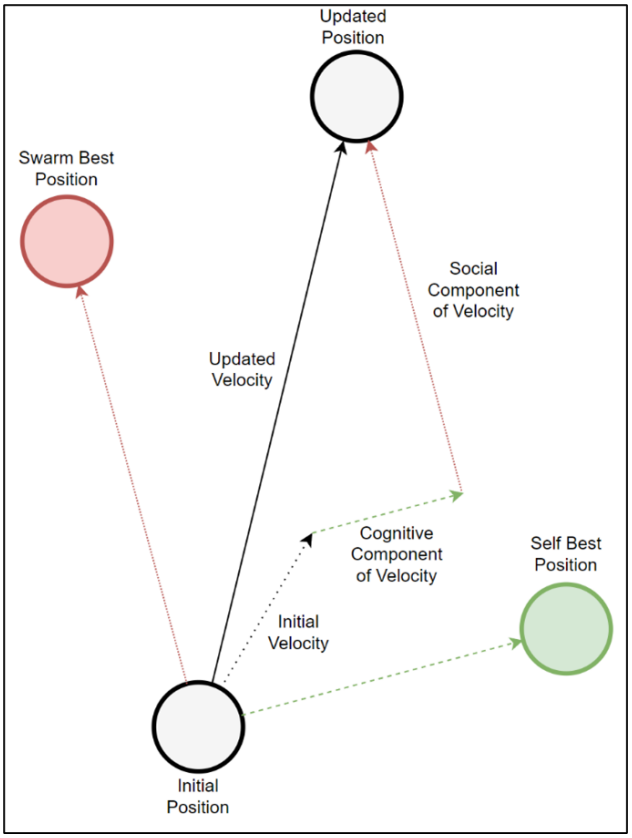

Figure 2.4: Particle Swarm Optimization Overview. Each particle updates its position by combining its own initial velocity component with velocity towards its previous best position and the swarm best position.

From the data available to each particle, a new velocity vector is calculated. The current velocity of the particle is redirected towards its previous best position, also known as cognitive acceleration, and redirected towards the swarm best position, also known as the social acceleration. Each particle is then shifted to a new position along the calculated velocity vector, the fitness is calculated once again, and the cycle continues. The algorithm continues until the maximum iteration count has been reached, or a stopping criterion has been satisfied.

Velocity update formulation:

$$v_{final} = (W * v_{init}) + (c_1 * n_{rand}) * (x_{p.best} - x_{current}) + (c_2 * n_{rand}) * (x_{s.best} - x_{current})$$

where:

v_{final} = Updated Velocity

v_{init} = Initial Velocity

$x_{current}$ = Vector of Current position

$x_{p.best}$ = Vector of Personal Best position

$x_{s.best}$ = Vector of Swarm Best position

W = Initial Velocity Weight

c_1 = Personal Influence Weight

c_2 = Swarm Influence Weight

n_{rand} = Random Value

Particle update Logic:

Algorithm 1 PSO Pseudocode

Apply Initial Parameters (c_1, c_2, W, ...)
Apply Number of Iterations (num_iter)
Apply Number of Particles per iteration ($num\_particle$)
Set Initial Fitness (Minimization = $inf.$, Maximization = 0)
Define testing function, f
Set Swarm Best Fitness = Initial Fitness
Set Swarm Best Position = $(0, 0, 0, ...)$
for $num\_particle$ **do**
  Set initial position ($x_{initial}$) = sample dist.
  Set $x_{current}$ to $x_{initial}$
  Set Personal Best Position, $x_{p.best}$ = $x_{initial}$
  Set Personal Best Fitness = Initial Fitness
  Generate Initial Velocity, v_{init}
end for
for each val in num_iter **do**
  for each val in $num\_particle$ **do**
    Calculate Current Fitness of particle at $x_{current}$ with function f
    if Personal Best Fitness < Current Fitness **then**
      Personal Best Fitness = Current Fitness
      Personal Best Position = Current Position
      if Swarm Best Fitness < Current Fitness **then**
        Swarm Best Fitness = Current Fitness
        Swarm Best Position = Current Position
      end if
    end if
  end for
  Update Velocity v_{final}
  Update Position according to Velocity
end for

3. TESTING DESIGN

3.1 Methodology Overview

To test the capability of E2CO to perform in a well control optimization scenario, a system has been developed to compare performance of E2CO against an industry standard numerical simulator. A proprietary well control optimizer was created based on particle swarm optimization. The optimizer was built in Python 3 and can interact with both E2CO, and any commercial numerical reservoir simulator currently supported on Windows. Building a single, flexible optimizer was a crucial step in providing a fair comparison ground on which to test E2CO's performance. Figure 3.1 displays the integrated workflow of E2CO and a tradition numerical simulator (IMEX in Figure 3.1) with PSO, highlighting the common Particle Swarm Optimizer at the core of this testing set-up.

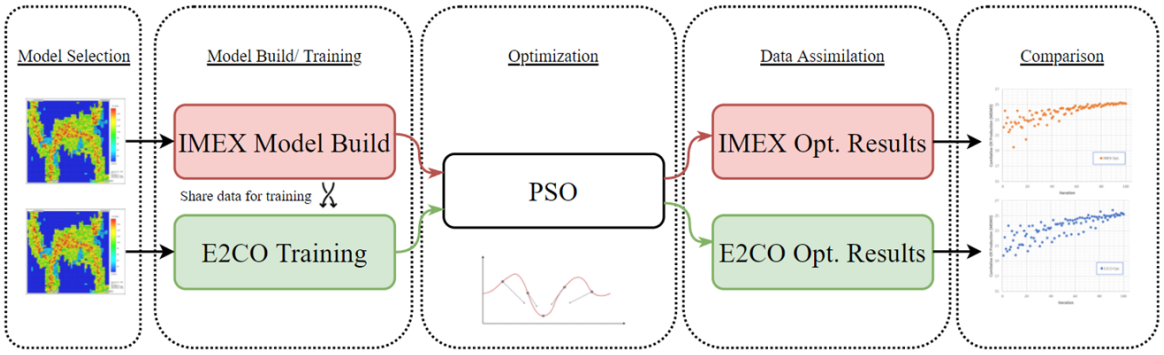

Figure 3.1: Optimization Combined Workflow Overview

E2CO provides several benefits that allow efficient implementation into and optimization algorithm, as such the design of the system will be explained as one cohesive unit. The implementation of particle swarm optimization with E2CO will be referred to as PSO-E2CO.

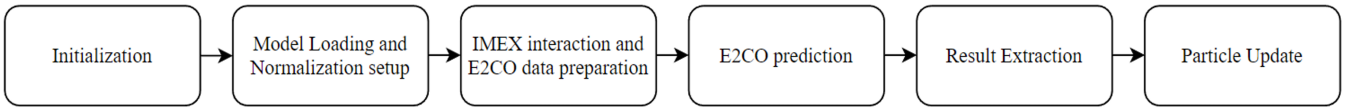

Figure 3.2: PSO-E2CO Overview. From initialization to finding a solution.

The full PSO-E2CO system can be largely broken down into six major components, as shown in Figure 3.2. A more detailed description of these components will be done in Section 3.2

The performance of E2CO will be compared with an industry-leading simulator IMEX [17], developed by the Computer Modelling Group via optimizing for cumulative oil production at the end of a simulation run. Additionally, E2CO relies on IMEX to calculate the initial realization before the start of each prediction (more on this later).

3.2 PSO-E2CO Overview

3.2.1 Initialization

The process begins by reading and applying the initial parameters to the optimizer as described in the PSO Pseudocode algorithm in Section 2.4. The initial parameters include the optimization target value (in our case it is maximization), the number of particles and iterations to use, the weights of the velocity components (i.e. how to weight social versus cognitive velocity vectors), the number of simulation time steps, etc. All the parameters used in the calculation of the velocity update described in Section 2.4. The full list of parameters used in the study can be found in the results section.

Following initial reading, the well controls are applied based on user determined ranges. The optimizer itself does not have any constrains on the values that the well controls can take; however, experience is necessary to specify reasonable ranges based on the specific model used. For example, in all the cases presented in this paper, the general guideline was to keep the final production water cut between 90% - 98%, as producing any more or any less would be wasteful from a real-world production perspective.

At this point, all the optimizer's key parameters are determined, and it is time to specify the

initial values for each of the particles. The min/max ranges for the controls specified earlier are used as bound for a distribution from which to sample the initial values. A value is sampled for each of the N wells and are applied as position of a particle in N-dimensional space. The choice of initial sampling distribution is heuristic; however, a uniform distribution is favorable for fair comparison.

Last, files are generated to save the evolution of all particle's position and fitness (final value).

3.2.2 Model Loading and Normalization Setup

At the second stage of PSO-E2CO, the data obtained during the training procedure is loaded into memory for use during the main prediction loop.

First, the generated normalizations are loaded. These normalizations are used to normalize and denormalize the data as it passes through the E2CO prediction framework. The loaded data includes the values to denormalize all the main variables in play, namely: pressure, saturation, well controls, bottom-hole pressure, injection and production rates, and flux.

Second, the Neural Networks optimized during the training procedure are loaded. These neural networks have been trained and saved as Keras models and are fixed for the duration of the well control optimization procedure. The neural networks loaded include the Encoder, the Decoder, the Transition network, as well as the Transition Well Data network. Note, the decoder is not necessary for the optimization procedure, however it was kept for debugging purposes.

3.2.3 IMEX interaction and E2CO data Preparation

In this step, the particle positional arguments are saved as well controls in formats applicable to E2CO and IMEX. The IMEX formatted data is passed along with a call to the IMEX simulator to compute the initial and first time-step of the simulation. The initial data is necessary as the initial 'x_0' state variables for the following calculations. The data for the first time-step is extra, it is saved for debugging purposes.

Once the model initialization is completed in IMEX, the data is extracted and prepared in an E2CO-friendly format for the main prediction.

3.2.4 E2CO Prediction

This step contains the core E2CO prediction loop. The data from the previous step is unpacked into four separate variables as follows (Just as described in Section 2.2.1):

- x_0: Vector of initial state variables at time t = 0.
- ∇t_t: Vector of time steps.
- u_t: Vector of well controls for each time step.
- $norm$: Normalization for final prediction result.

Networks 'Decoder', 'Transition', and 'Transition WD' were preloaded at stage 2 of the PSO-E2CO algorithm. These can be seen with respective names in Figure 3.3

The initial state variable vector x_0 is passed through the decoder network, which reduces the variables into a smaller (normalized) latent space $\hat{z}_t$. The latent (space) state variable $\hat{z}_t$ is passed into the Transition network along with the corresponding time step Δt_t. The Transition network calculates matrices A_t, B_t, that are used in conjunction with the corresponding well control u_t and latent state variable $\hat{z}_t$ to calculate the predicted latent state variable $\hat{z}_{t+1}$. The loop continues while setting the new $\hat{z}_{t+1}$ prediction to $\hat{z}_t$ for each new time step.

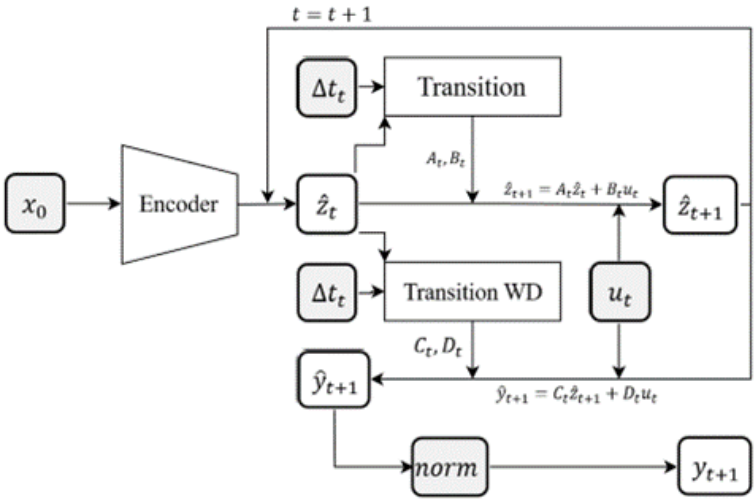

Figure 3.3: E2CO Prediction loop within PSO-E2CO (Modified from Coutinho et al.)

At each iteration of the prediction, the predicted latent state variable $\hat{z}_t$ is fed into a transition well data network along with the corresponding time step Δt_t. The transition well data network calculates matrices C_t, D_t, that are used in conjunction with the corresponding well control u_t and latent state variable prediction at the following time step $\hat{z}_{t+1}$ to calculate the normalized well outputs $\hat{y}_{t+1}$.

At the end, the output vector $\hat{y}_{t+1}$ is denormalized into y_{t+1} via the loaded normalizations.

3.2.5 Result Extraction

The results predicted by the E2CO loop require additional post-processing before they can be used. The goal with the study is to optimize for cumulative oil production at the end of a simulation. There are three key aspects that need to be addressed before cumulative oil production is determined.

First, the output vector $\hat{y}_{t+1}$ provides oil rates from the first time-step onward. To calculate the cumulative rate, we need the oil rate at time step $t = 0$ as well. Since this optimization focuses on models with constant well controls, we expect the production rate to stay constant until water break-through is observed at the producer. As such, it is safe to set the oil and water production rates at time $t = 0$ to the rates observed at $t = 1$. This can be seen in figure 3.4, where the first two rates on the top graph are identical.

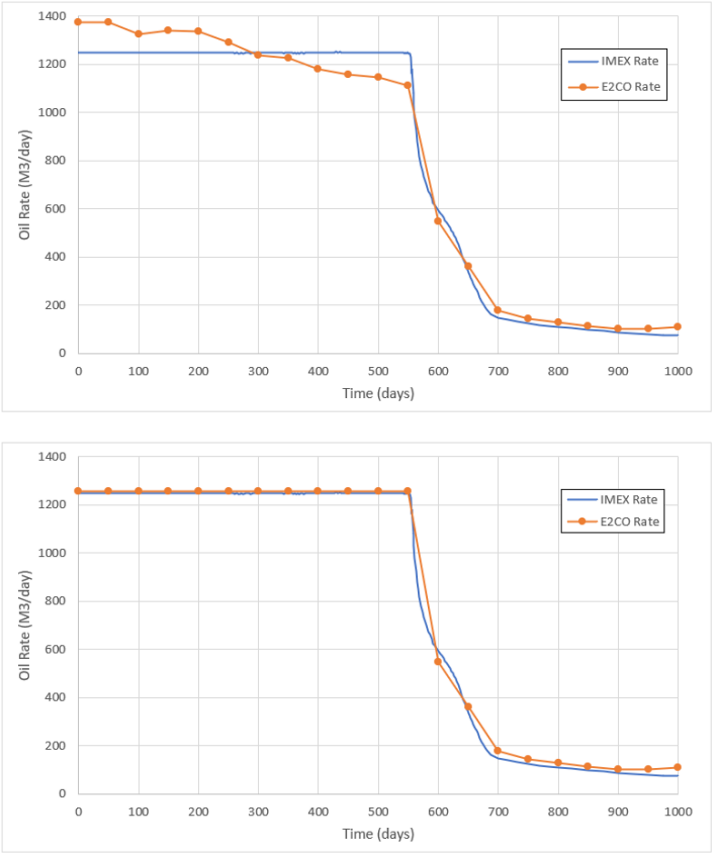

Figure 3.4: Rate Adjustment Example. Staggered-Line Drive (Validation Model 14)

Second, since E2CO was initially designed for models with varying well controls, E2CO struggles to predict constant rates that are expected before water-breakthrough. Keeping the well controls constant deprives the neural networks from some variability, and thus the results either over-predict at early times and under-predict at later times, or vice versa. To rectify this, the rates predicted before water breakthrough are set to the average of the calculated values up to that point. This modifies the values in the constant rate region to be closer to the expected results while only using information obtained from E2CO. This change can be seen in Figure 3.4, where the predicted rates from the first graph are averaged and then set constant until the point of water breakthrough.

Third, E2CO predicted well output is a vector consisting of instantaneous oil and water rate predictions at each time step of a simulation. In order to calculate the cumulative rate, the obtained values need to be numerically integrated across the entire time range.

Once the output values have been processed, the output cumulative oil and water productions are passed along to the final step in the procedure.

3.2.6 Particle Update

Finally, the results are saved, and the cumulative oil production is updated as particle fitness. Depending on the stage of the optimization procedure, the particles' personal and global best values are updated accordingly, as described in Section 2.4.

A check for termination criteria is performed, if the criteria is met, the results are saved, and the data is ready for analysis. In the case the criteria are not met, the updated particle gets passed back to the third stage of the procedure and the process is repeated until the termination criteria is satisfied.

3.3 PSO-IMEX Overview

The implementation of IMEX with the proprietary optimizer is much simpler in comparison with the E2CO implementation. Since IMEX is a polished, self-contained product, the overall PSO-IMEX algorithm only consists of three major steps as seen in Figure 3.5.

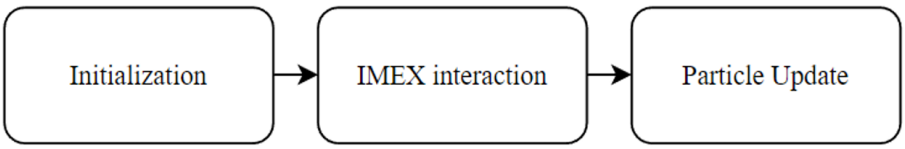

Figure 3.5: PSO-IMEX Workflow

Like with PSO-E2CO, the main PSO variables are initialized to be the same as E2CO for the sake of fair comparison. Once the particles have been assigned their corresponding values, IMEX interaction is initiated. Here, the IMEX interaction is allowed to run for the entire duration of the simulation and the cumulative oil production is extracted. The particles are updated accordingly, and the optimization procedure is conducted until the termination criteria has been met.

4. RESULTS

As per one of the main objectives of this study, it is important to display the flexibility of the E2CO algorithm. To achieve this, the optimization performance of several different models needs to be considered. In this study, we will look at four distinct models and dissect their results. The four models are as follows:

1. 'Showcase' Model: This was the first model considered for this study. The model has a very defined permeability distribution, providing distinct high fluid flow zones for E2CO to analyze. The model contains 4 wells, 2 producers and two injectors. The injectors are placed at the bottom of the model, with producers being at the top. This provides a fairly uniform water-front to develop during production, akin to a line drive production pattern [18].

2. '5-spot' Model: The second model employs a much more homogeneous permeability distribution with a 5-spot well pattern [18]. This pattern consists of 4 injectors on the outside edges of the model, with 1 injector in the middle.

3. 'Inverted 5-spot' Model: This model mimics the set-up of the '5-spot' model with an inverted pattern, consisting of 4 producers on the outer edges of the model, with 1 injector in the middle [18].

4. "Staggered Line Drive" Model: This model employs the same underlying physical structure as the previous two models, but with a staggered line drive well set-up. 2 injectors on the right side of the model, with 1 producer on the left [18].

This collection of models was chosen to display the flexibility of the E2CO approach. With this, we hope to display the agnostic nature of E2CO in regard to the model type being used.

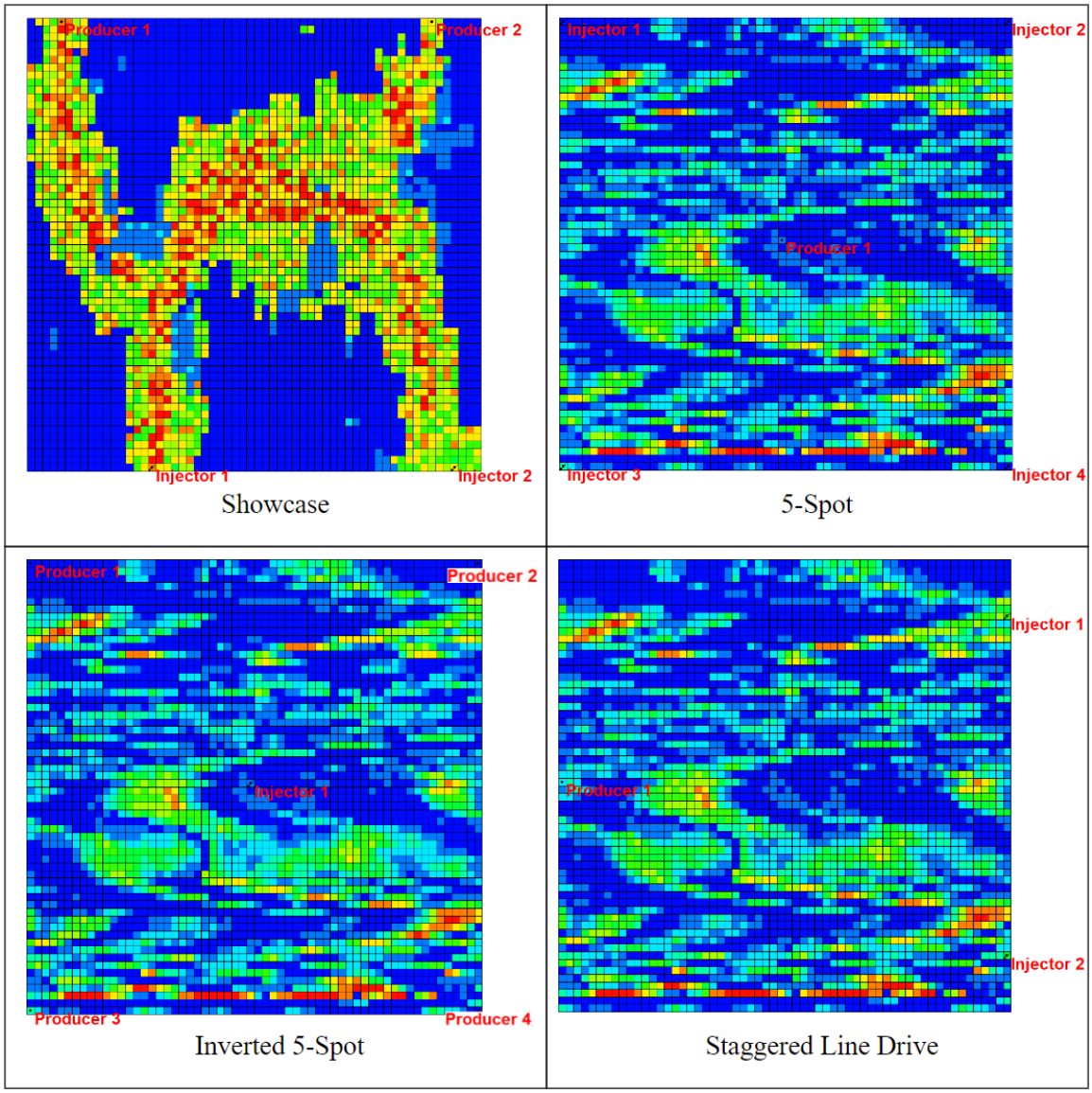

Figure 4.1: All Models Tested

4.1 General Parameters

This section contains the description of all the general common parameters used throughout the study.

All the models considered for this study are two-dimensional, two-phase, oil-water systems of varying physical properties and production patterns. Optimization is done for 10 particles, 10 iterations for all examples. Capping off overall runs to 100 provides a fair base for comparison.

Variable Name	Value
Number of Particles	10
Number of Iterations	10
Initial Velocity weight (w)	0.7298
Cognitive Acceleration weight (c_1)	1.49618
Social Acceleration weight (c_2)	1.49618
Seed	1234

Table 4.1: PSO parameter values used.

Following is a presentation of result for each of the above-mentioned models.

4.2 Showcase Model

The showcase model is the least complex of the four models considered in the study. The model is run for a short time duration of only 20 days. The short time duration was chosen for two reasons; First, the physical parameters set in this model lead to quick reservoir drainage. Second, this is another way to show the flexibility of E2CO, the ability to work with different ranges of time steps. Furthermore, having a well-defined high-permeability zone for fluids to move through translates to quick computational times.

Variable Name	Value
Number of Gridblocks	60x60x1
Gridblock sizes	5m x 5m x 10m
Number of Wells	4 Wells: 2 Inj. + 2 Prod.
Simulation Time	20 days (1 day x 20 steps)
Injector Controls	600 – 1000 m^3/day
Producer Controls	270 – 320 kgf/cm^2
Porosity	0.2 (constant)
Permeability	Refer to Appendix

Table 4.2: Showcase Model Parameters

Figure 4.2 displays the comparison of the cumulative production of oil and water between IMEX and E2CO, across 10 realizations from the validation set. Although having a small validation set, the results observed are promising, with a close match between all model realizations.

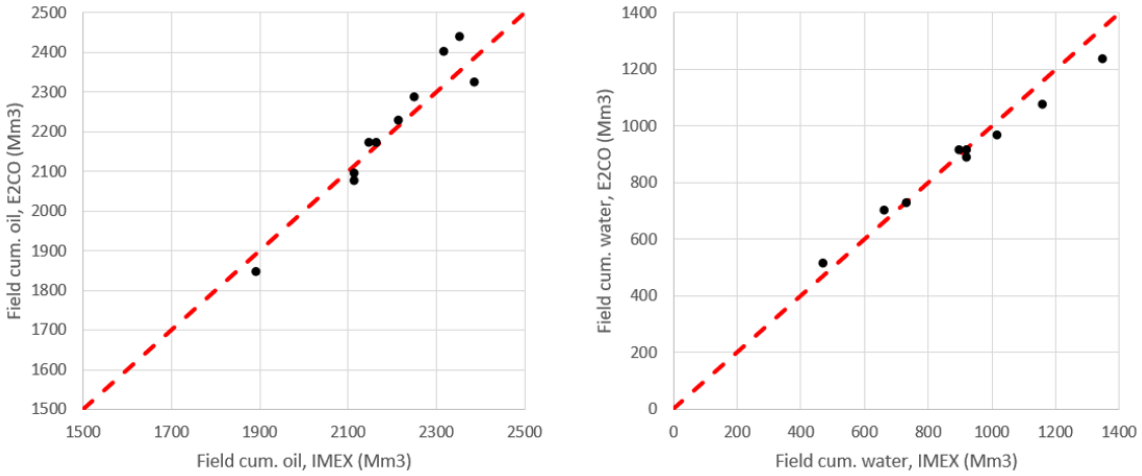

Figure 4.2: Validation Set Cumulative Production Comparison (Showcase)

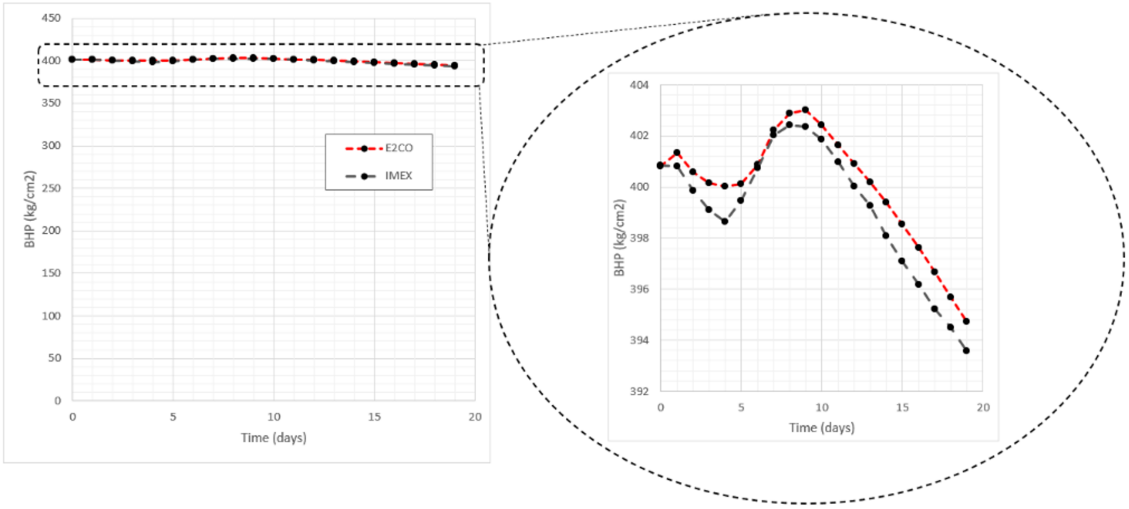

Figure 4.3: Example Injector BHP Match (Showcase, Validation Model 1, Injector 1)

The output matches on the injection bottom-hole pressure were also verified before proceeding with optimization, as seen in Figure 4.3. Seeing these matches gave confidence in pursuing this research further.

The close match obtained during the training of the model has translated to good results in the optimization procedure. The optimizer was tasked with finding the maximum possible oil production after 20 time-steps. Both E2CO and IMEX have converged to similar values after the optimization procedure. The cumulative oil production for E2CO was determined to be 25.75 MSM3, and for IMEX the production was 25.27 MSM3

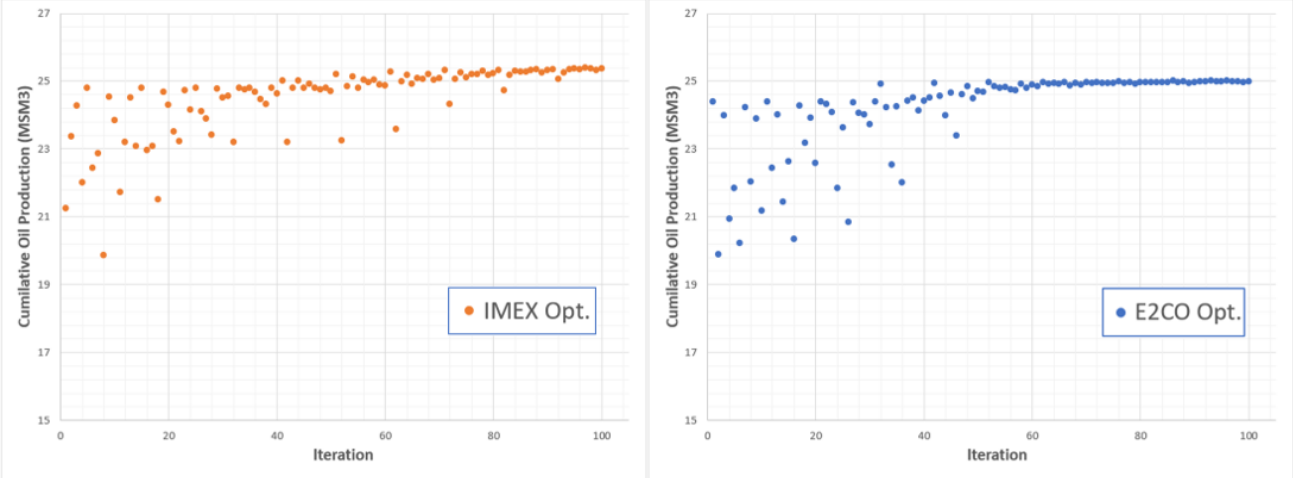

Figure 4.4: Showcase Optimization Results Comparison (E2CO in Blue, IMEX in Orange)

In this model, as well as all the others discussed, we are seeking to optimize for constant well controls (i.e. producing at set pressure/rates throughout the entire duration of the simulation). On the initial implementation, both E2CO and IMEX were able to determine values that are close to optimal well controls within 100 iterations. The true results for optimal model performance as obtained by CMG's CMOST [19] reservoir optimization software package are:

- Producer 1 = 320.00 kgf/cm^2
- Producer 2 = 270.00 kgf/cm^2

- Injector 1 = 1000.00 m^3/day
- Injector 2 = 1000.00 m^3/day

CMG's CMOST optimization platform includes their implementation of PSO, this was chosen as a fair comparison to the self-build PSO optimization algorithm.

		Well Controls			
Engine	Cumulative Production (MSM3)	Producer 1 (kgf/cm^2)	Producer 2 (kgf/cm^2)	Injector 1 (m^3/day)	Injector 2 (m^3/day)
IMEX	25.38	320.00	270.00	957.04	928.62
E2CO	25.01	311.02	270.00	1000.00	1000.00

Table 4.3: Showcase Model Results

The initial optimization runs have been conducted on an older base version of E2CO that is presented in the work of Coutinho et al., as such the time performance of E2CO suffered. The changes made to E2CO since the initial version refine some aspects of the training and prediction loops, making the algorithm more efficient. Additional features have been added as well, but these do not apply to the examples considered in this paper.

Engine Used	**Time (sec)**
IMEX	127.3
E2CO	211.0

Table 4.4: Showcase Optimization Performance

4.3 5-spot Model

The 5-spot model (as well as all proceeding models) is larger in scope than the showcase example. The model is run for a much longer duration of 1000 days. This model has a much more uniform permeability distribution, allowing more possible paths for fluid migration, adding complexity to the model. An additional well further complicates the dynamics of fluid flow in the reservoir.

Variable Name	**Value**
Number of Gridblocks	60x60x1
Gridblock sizes	10m x 10m x 20m
Number of Wells	5 Wells: 4 Inj. + 1 Prod.
Simulation Time	1000 days (50 days x 20 steps)
Injector Controls	200 – 300 m^3/day
Producer Controls	290 – 335 kgf/cm^2
Porosity	0.2 (constant)
Permeability	Refer to Appendix

Table 4.5: 5-Spot Model Parameters

Figure 4.5 displays the comparison of the cumulative production of oil and water between IMEX and E2CO, across 50 realizations from the validation set. The overall match observed is worse compared to the showcase example due to the more complicated fluid dynamics involved. Here we can notice a pattern of over-prediction of cumulative oil rates at the higher end of production, and under-prediction at the lower end of production. From this observation, as well as other empirical findings, E2CO applied to models with constant well controls tends to amplify the extremes. However, this behavior is not detrimental to the final results, as we will see. As long as the general linear trend in the cumulative comparison is maintained, the final well control values do not suffer.

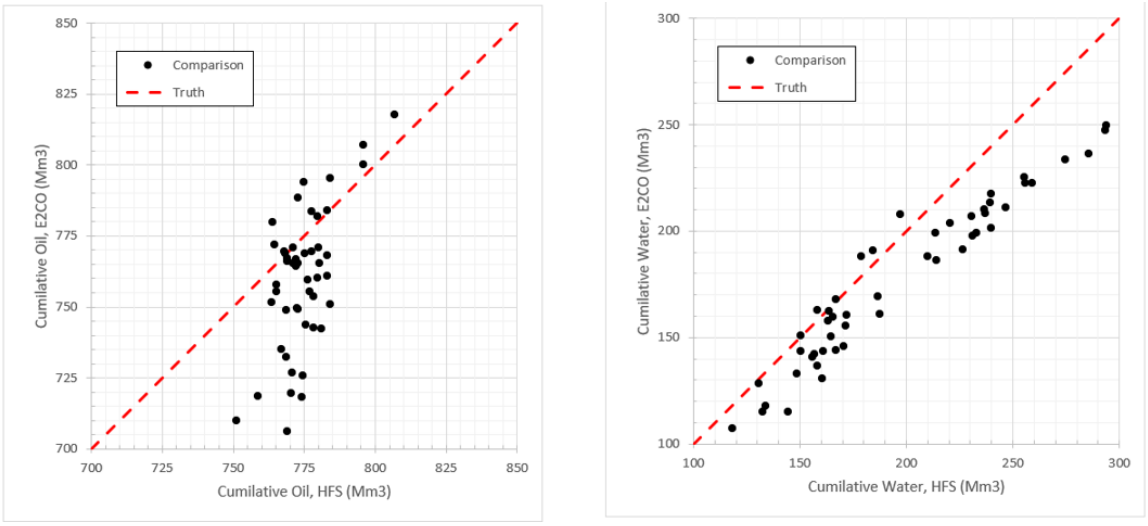

Figure 4.5: Validation Set Cumulative Production Comparison (5-Spot)

The optimizer was tasked with finding the maximum oil production after 20 time-steps. For the 5-spot model, E2CO overpredicted the final cumulative production. The cumulative oil production for E2CO was determined to be 841.05 MSM3, and for IMEX the production was 782.63 MSM3.

In this set, results from CMOST optimization engine are included in graphical form.

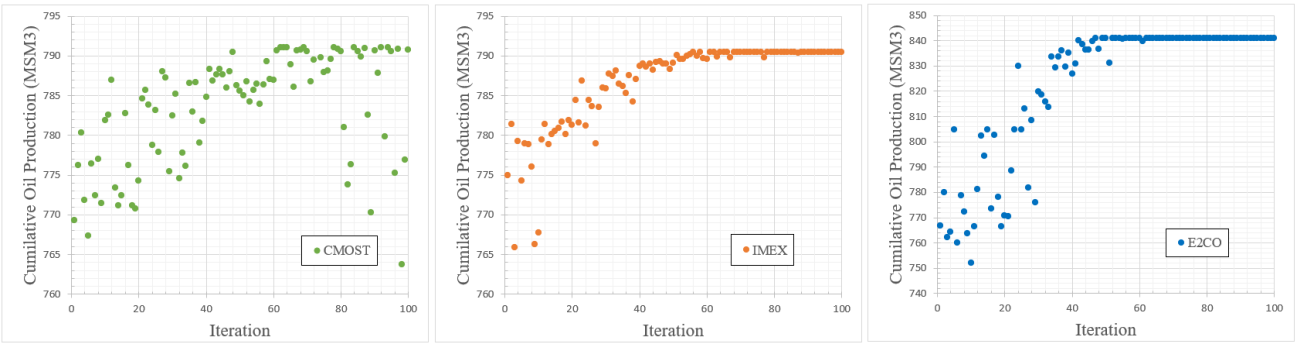

Figure 4.6: 5-Spot Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green)

As alluded to previously, the well control optimization performance did not suffer due to the overprediction in the cumulative performance. All the methods implemented were able to obtain

the desired optimal controls, as confirmed by the CMOST results.

		Well Controls				
Engine	Cumulative Production (MSM3)	Producer 1 (kgf/cm^2)	Injector 1 (m^3/day)	Injector 2 (m^3/day)	Injector 3 (m^3/day)	Injector 4 (m^3/day)
IMEX	782.63	290.00	300.00	300.00	300.00	300.00
E2CO	841.05	290.00	300.00	300.00	300.00	300.00
CMOST	782.63	290.00	300.00	300.00	300.00	300.00

Table 4.6: 5-Spot Model Results

The 5-Spot model (and the rest) run has been conducted on the updated E2CO engine, equipped with a better tuned PSO optimization algorithm. This is reflected in the runtime obtained.

Engine Used	**Time (sec)**
IMEX	246.9
E2CO	176.3
E2CO (Adjusted)	163.5

Table 4.7: 5-Spot Optimization Performance

The "Adjusted" result presented in the table 4.7 is the time for E2CO to complete an optimization run without IMEX interaction. In PSO-E2CO workflow described earlier, each optimization is run in IMEX until the first time step for debugging. The adjusted time represents the "pure" E2CO time. At the current stage of development, it is impossible to completely decouple E2CO from traditional numerical simulators due to the inherent need for primary realizations of state variables. "Adjusted" time provides a better representation of E2CO performance at scale, since by increasing the size of the considered models, the relative impact of the IMEX interaction on the time of the optimization run is minimized.

4.4 Inverted 5-spot Model

The Inverted 5-spot model takes on the same physical characteristics as the regular 5-spot. Although the physical description of the reservoir is the same, the inverted pattern displays far more complex fluid flow mechanics. For this case, all optimization algorithms struggled to find the true optimum well control configuration.

Variable Name	**Value**
Number of Gridblocks	60x60x1
Gridblock sizes	10m x 10m x 20m
Number of Wells	5 Wells: 1 Inj. + 4 Prod.
Simulation Time	1000 days (50 days x 20 steps)
Injector Controls	850 – 1050 m^3/day
Producer Controls	290 – 335 kgf/cm^2
Porosity	0.2 (constant)
Permeability	Refer to Appendix

Table 4.8: Inverted 5-Spot Model Parameters

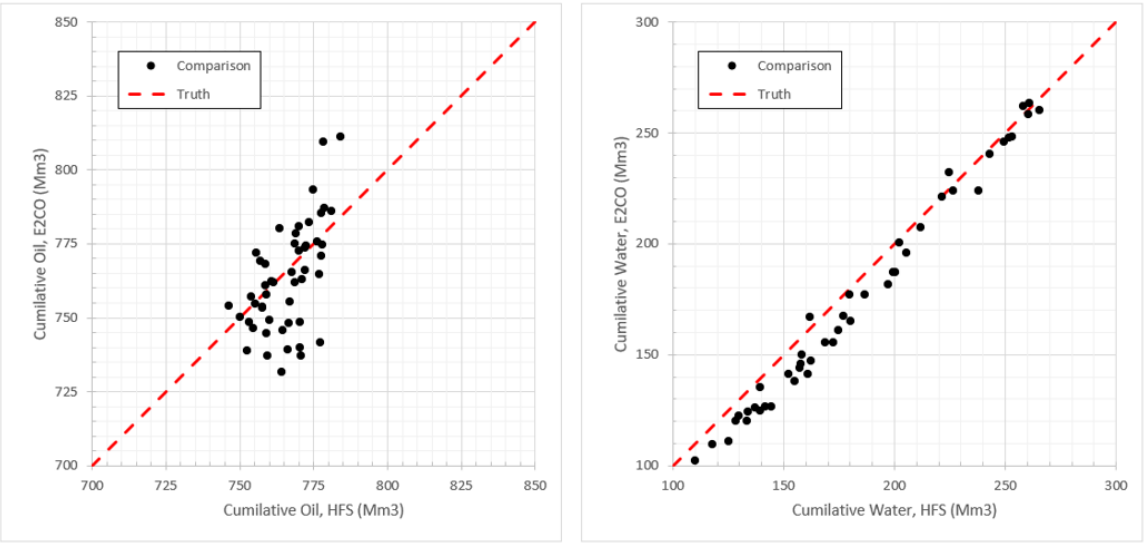

Figure 4.7: Validation Set Cumulative Production Comparison (Inverted 5-Spot)

Figure 4.7 displays the comparison of the cumulative production of oil and water between IMEX and E2CO, across 50 realizations from the validation set. The overall match observed in this example appears slightly better than the previous model. The same over/underprediction is observed at the extremes, however now there is a tighter overall clustering in the cumulative oil predictions. The complex flow nature of this drainage pattern translates to some instability in the E2CO prediction results. The cumulative water results however, are far more accurate.

The optimizer was tasked with finding the maximum possible oil production after 20 time-steps. In the case of the Inverted 5-spot model, as with the previous model, E2CO overpredicted on the final cumulative production. The cumulative oil production for E2CO was determined to be 830.90 MSM3, and for IMEX the production was 780.18 MSM3.

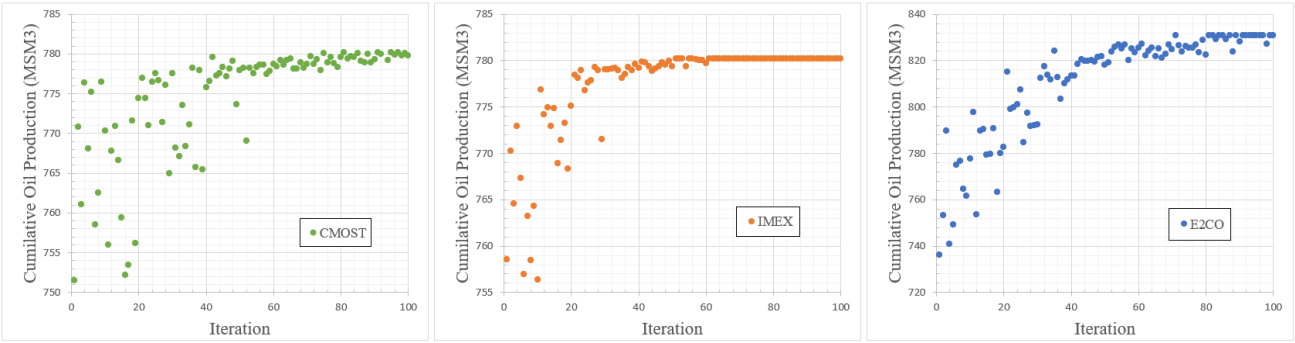

Figure 4.8: Inverted 5-Spot Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green)

The optimization results obtained for this model differ drastically from the 5-spot model. The Inverted 5-spot was much more difficult for all optimizers involved. None of the implemented methods were able to obtain the true optimal well controls after 100 iterations, including CMOST. To verify the results, additional optimization was necessary. Here, optimization results from CMG's DECE [20] are also included. DECE is CMG's proprietary optimization platform based on PSO, implementing algorithms such as Latin hypercube sampling and macro/micro optimization adjustments. DECE found the optimal results, however this took the algorithm 388 iterations.

		Well Controls				
Engine	Cumulative Production (MSM3)	Producer 1 (kgf/cm^2)	Producer 2 (kgf/cm^2)	Producer 3 (kgf/cm^2)	Producer 4 (kgf/cm^2)	Injector 1 (m^3/day)
IMEX	780.18	290.00	290.00	335.00	335.00	1050
E2CO	830.90	290.00	335.00	335.00	335.00	1050
CMOST	780.05	316.44	290.00	334.98	334.57	1050
DECE	780.41	297.20	290.00	335.00	325.78	1050

Table 4.9: Inverted 5-Spot Model Results

Both E2CO and IMEX underpredicted Producer 1 BHP as compared to the DECE results, this can be attributed to shortcomings in the implementation of PSO, however considering CMOST was unable to grasp the true solution after 100 iterations, this is up to debate. Producer 2 results show the biggest disparity between E2CO and IMEX results. This well proved difficult for E2CO to understand. Producer 4 predictions were overestimated for both IMEX and E2CO, this well was the most sensitive out of the 5. The values for Producer 3 and Injector 1 have optimized correctly.

Engine Used	**Time (sec)**
IMEX	298.7
E2CO	182.1
E2CO (Adjusted)	169.7

Table 4.10: Inverted 5-Spot Optimization Performance

4.5 Staggered Line Drive Model

The last model, Staggered Line Drive, takes on the same physical characteristics as the 5-Spot and the Inverted 5-Spot. This model displayed the longest runtimes across the board. The uneven placement of wells led to complex flow behaviors, but the reduced number of wells lead to better accuracy of results.

Variable Name	**Value**
Number of Gridblocks	60x60x1
Gridblock sizes	10m x 10m x 20m
Number of Wells	3 Wells: 2 Inj. + 1 Prod.
Simulation Time	1000 days (50 days x 20 steps)
Injector Controls	500 – 700 m^3/day
Producer Controls	290 – 335 kgf/cm^2
Porosity	0.2 (constant)
Permeability	Refer to Appendix

Table 4.11: Staggered Line Drive Model Parameters

From the validation set cumulative comparison, the overall match observed shows higher over-prediction than in previous cases. The cumulative water predictions are also a little more unstable compared to the previously discussed cases.

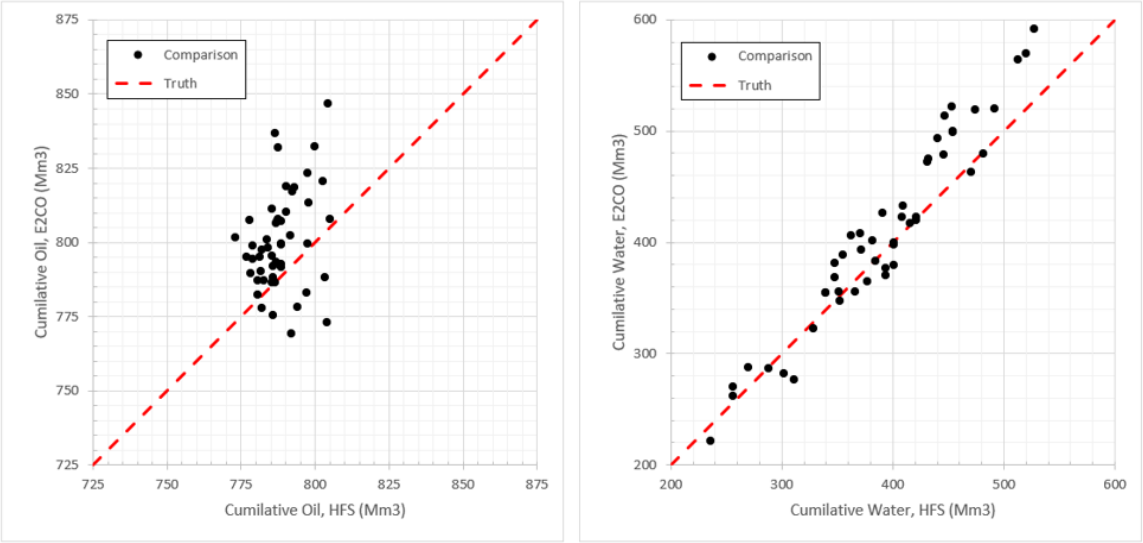

Figure 4.9: Validation Set Cumulative Production Comparison (Staggered Line Drive)

The optimizer was tasked with finding the maximum possible oil production after 20 time-steps. In the case of the Staggered Line Drive, E2CO overpredicted on the final cumulative production the most out of all the models tested. The cumulative oil production for E2CO was determined to be 873.39 MSM3, and for IMEX the production was 797.97 MSM3.

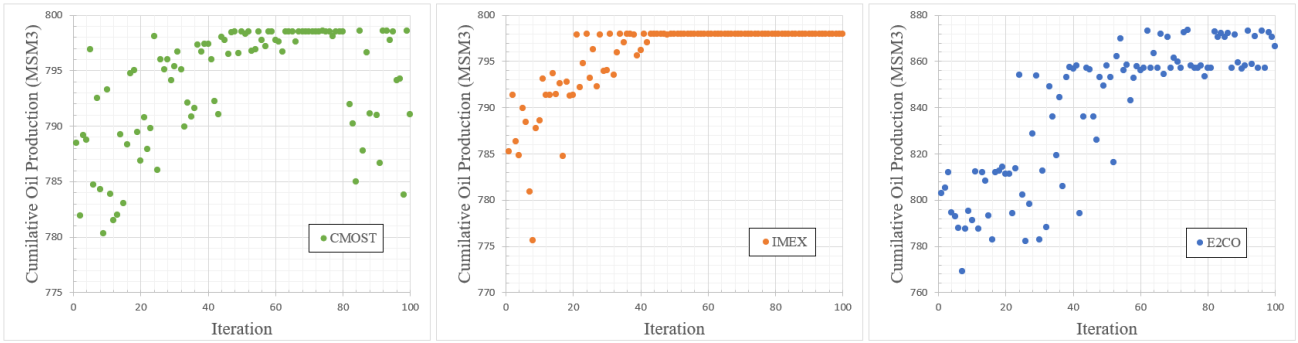

Figure 4.10: Staggered Line Drive Optimization Results Comparison (E2CO in Blue, IMEX in Orange, CMOST in Green)

E2CO was able to achieve the correct controls on 2 out of the 3 wells. Injector 2 optimal for E2CO is slightly underpredicted.

		Well Controls		
Engine	Cumulative Production (MSM3)	Producer 1 (kgf/cm^2)	Injector 1 (m^3/day)	Injector 2 (m^3/day)
IMEX	797.97	290.00	700.00	700.00
E2CO	873.39	290.00	700.00	670.62
CMOST	797.97	290.00	700.00	700.00

Table 4.12: Staggered Line Drive Model Results

Engine Used	**Time (sec)**
IMEX	322.92
E2CO	184.03
E2CO (Adjusted)	170.67

Table 4.13: Staggered Line Drive Optimization Performance

4.6 Training Results

The application of E2CO with particle swarm optimization requires a complete retraining of the underlying neural networks in order to apply the technique effectively. Each E2CO realization generated upon completion of training can be re-used indefinitely, but only for the same model.

The training times presented in Table 4.14 have all been obtained after running the training process on a personal computer. The computer specifications are provided as well.

Showcase model training times have been omitted for a fair representation of the updated E2CO algorithm. As seen in the results table, all training runs have been completed in around 110 - 120 minutes. All the training runs were set to terminate at 100 epochs, even though all have achieved their optimal settings in less epochs.

	Training Time (min)	Prep. Time (sec)
Showcase	-	-
5-spot	111.56	9.5
Inverted 5-spot	114.80	9.5
Staggered-Line Drive	122.46	9.3

Table 4.14: Training Results

Component	Applicable Metric
CPU - Ryzen 5800X	8 Core / 3.8GHz - 4.7GHz
GPU - NVIDIA RTX 3070Ti	6144 Cuda Cores
RAM	32GB DDR4

Table 4.15: PC Specifications

As it stands right now, the training algorithm lacks a proper implementation of some common features found in other neural networks, such as hyperparameter tuning and training termination criteria. As such, the training times are hurt.

It is important to point out that the training times can be significantly improved via utilization of GPU parallelization and the use of professional level hardware. As noted in the work of Coutihno et al., the primary model considered in their work is even more complex than the ones presented here, consisting of 9 wells in an unorthodox arrangement, with a larger base model and longer run times. In the paper, training on a similarly configured PC yielded a training time of 152 minutes, but utilizing 2 of NVIDIA's professional grade Tesla V100 GPUs cut this time down to only 38 minutes. In simpler terms, E2CO scales well for larger models.

4.7 Compiled Results

Table 4.16 contains the compiled results for each of the models tested. Cumulative oil production accuracy and optimization runtimes are considered.

In the table containing cumulative oil production results, a column was added titled "Re-Run". The data presented under the "Re-Run" column corresponds to the cumulative oil production obtained when the well controls determined by E2CO are used in a single run of IMEX for each corresponding model. This was added to quantify the difference between the true IMEX results and the results provided by E2CO, accounting for the overall overprediction.

	Cumulative Oil (MSM3)			**Percent Difference**	
Model	IMEX	E2CO	Re-Run	(IMEX-E2CO)	(IMEX - Re-run)
Showcase	25.38	25.01	25.76	-1.46	1.50
5-spot	782.63	841.05	782.63	7.46	0.00
Inverted 5-spot	780.18	830.90	778.01	6.50	-0.28
Staggered-Line Drive	797.97	873.39	796.65	9.45	-0.17

Table 4.16: Overall Cumulative Oil Production Performance

As seen in table 4.16, when accounting for the overprediction, the differences in the cumulative oil production between E2CO and IMEX account for around 1.5 percent on the old E2CO model, and less than 0.5 percent on the new implementation.

	Runtime (sec)			**Percent Difference**	
Model	IMEX	E2CO	Adjusted	(IMEX-E2CO)	(IMEX - Adjusted)
Showcase	127.3	211.0	-	65.75	-
5-spot	246.9	176.3	163.5	-28.59	-33.78
Inverted 5-spot	298.7	182.1	169.7	-39.04	-43.19
Staggered-Line Drive	322.9	184.0	170.7	-43.02	-47.14

Table 4.17: Overall Optimization Runtime Performance

Table 4.17 contains all runtimes, as well as their relative percentage differences to IMEX runtimes. Across all models ran on the updated versions of E2CO and PSO-E2CO, E2CO displays a clear advantage over the traditional method, with maximum observed time savings of nearly 50 percent for the most dynamically complex model. Furthermore, as the models get progressively more complex, E2CO gets further ahead of its traditional counterpart. It is evident that E2CO scales much better than traditional approaches.

Further performance increases are expected as we increase the size, complexity, and number of time steps of the models used. E2CO's scaling combined with server hardware, applied to a complex, real-world model is expected to yield far better results.

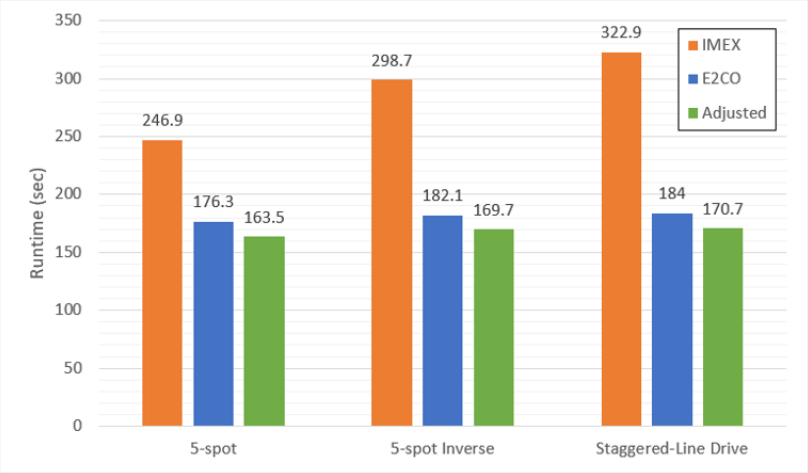

Figure 4.11: Runtime Comparison. Computation time comparison between PSO-IMEX, PSO-E2CO, and PSO-E2CO Adjusted

5. DISCUSSION AND CONCLUSION

5.1 Discussion

When interpreting the significance of the obtained results, it is important to correctly frame our perspective on the purpose of the proposed model. E2CO is a proxy reservoir simulator, as such, the model gives up on several aspects in favor of computational speed. However, E2CO was not meant to replace traditional reservoir simulators, it is meant to compliment them, to be used in tandem to achieve the best of both worlds – speed and accuracy.

From the testing performed in this study, the results indicate that E2CO can be implemented along-side traditional reservoir simulators. When pairing the two approaches, one can potentially achieve the desired results in a fraction of the time. Of course, additional tests are necessary to verify the scaling nature of E2CO displayed by this study, but as it stands, E2CO shows great promise as a tool for engineers to use to speed up their work and give them the ability to consider a greater number of possibilities. E2CO is completely non-intrusive and relies solely on simulation results for implementation, as such can be paired with any traditional reservoir simulator without the need for confidential engine parameters.

As expected, the accuracy of the results obtained depended on a good initial training fit. The closer the cumulative production validation comparison results resembled a linear trend, the closer the cumulative production obtained from E2CO resembled IMEX's output. It is important to note that E2CO was originally built for time-varying controls. The limitations of the scope of the study did not allow for integration of time-varying controls into a well control optimization set-up. Adjustments to E2CO are necessary to improve the stability and accuracy of the time-invariant control cases.

Due to E2CO's inherent reliance on initial realizations obtained from traditional reservoir simulators, it is difficult to accurately quantify its time performance. However, the analysis performed supports the claim that E2CO can outperform tradition reservoir simulators in optimization pro-

cedures. As complexity increases, the effects of coupling diminish in their significance, bringing E2CO further ahead of its traditional counterpart. However small the impact, work on optimizing the interaction is necessary to push E2CO to the next level.

The time gains displayed by E2CO are challenged by the time requirements of model training. Training a neural network is a costly, and time-consuming process. However, the relative prediction time gains inch ahead of training costs as the complexity of the models is increased. Further investigation is required to better quantify the relative time differences at scale.

Finally, the results of the study show a clear correlation between the complexity of the underlying model, and E2CO computational benefits. We expect to see a much larger increase in performance once we are ready to scale the models up. Furthermore, implementation of larger models will allow us to better utilize the potential of GPU parallelization.

5.2 Conclusion

This research aimed to determine whether a neural network-based proxy simulator approach can be used in a well control optimization workflow. Based on the results obtained, it was shown that an artificial intelligence proxy such as E2CO can be used to compliment and improve traditional workflows within reservoir engineering. It can be concluded that E2CO can be a viable addition to a reservoir engineer's tool belt, providing a faster alternative to existing frameworks.

The testing methodology described in this research provides a vision for what AI can achieve when implemented in a reservoir simulation environment. While the methodology limits the potential of E2CO by confining it to a specific task, it also provides a measurable insight into the capabilities of this technology.

The results obtained, highlight specific areas of improvement that need to be further developed to better understand the implications of this technology. It is recommended to consider work on the improvement of prediction accuracy of time-invariant control cases, hyperparameter tuning implementation, and further exploration into the scalability of the algorithm.

This research has shown that implementation of E2CO with particle swarm optimization is a flexible and efficient alternative to traditional well control optimization workflows.

REFERENCES

[1] D. W. Peaceman, *Fundamentals of numericl reservoir simulation*. Elsevier Scientific Publishing Company, 1977.

[2] K. Harrington, "Bp building new houston supercomputing center." Web Article, Dec. 2012. On BP Supercomputing Center coverage.

[3] J. Para, "Bp supercomputer in houston gets a big upgrade." Web Article, Dec. 2017. On BP Upgrade.

[4] E. Bentley, P. Minczeski, and J. Juan, "Which oil producers are breaking even?." Wall Street Journal (Online), July 2017. On Median Oil Breakeven price.

[5] W. Wu, X. Li, L. He, and D. Zhang, "Accelerating the iterative linear solver for reservoir simulation on multicore architectures," in *2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 265–272, 2014.

[6] *Development of A Fast Auxiliary Subspace Pre-conditioner for Numerical Reservoir Simulators*, vol. All Days of *SPE Reservoir Characterisation and Simulation Conference and Exhibition*, 10 2011. SPE-148388-MS.

[7] B. Beykal, F. Boukouvala, C. A. Floudas, N. Sorek, H. Zalavadia, and E. Gildin, "Global optimization of grey-box computational systems using surrogate functions and application to highly constrained oil-field operations," *Computers & Chemical Engineering*, vol. 114, pp. 99–110, 2018. FOCAPO/CPC 2017.

[8] Y. Yang, M. Ghasemi, E. Gildin, Y. Efendiev, and V. Calo, "Fast Multiscale Reservoir Simulations With POD-DEIM Model Reduction," *SPE Journal*, vol. 21, pp. 2141–2154, 06 2016.

[9] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *Advances in Neural Information Processing Systems*, 06 2015.

[10] Z. L. Jin, Y. Liu, and L. J. Durlofsky, "Deep-learning-based surrogate model for reservoir simulation with time-varying well controls," *Journal of Petroleum Science and Engineering*, vol. 192, p. 107273, 2020.

[11] E. Coutinho, M. Dall'Aqua, and E. Gildin, "Physics-aware deep-learning-based proxy reservoir simulation model equipped with state and well output prediction," *Frontiers in Applied Mathematics and Statistics*, vol. 7, 09 2021.

[12] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*. Gulf Professional Publishing, 1979.

[13] S. Klevtsov, "Linear solution techniques for reservoir simulation with fully coupled geomechanics," mathesis, Stanford University, Mar. 2017.

[14] J. Jansen, "Adjoint-based optimization of multi-phase flow through porous media – a review," *Computers & Fluids*, vol. 46, no. 1, pp. 40–51, 2011. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[15] V. Lai, Y. F. Huang, C. H. Koo, A. N. Ahmed, and A. El-Shafie, "A review of reservoir operation optimisations: from traditional models to metaheuristic algorithms," *Archives of Computational Methods in Engineering*, 2022.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, 1995.

[17] CMG, *IMEX*. Computer Modelling Group, 2019.

[18] S. C. Rose, J. F. Buckwatter, and R. J. Woodhall, *The design engineering aspects of waterflooding*. Society of Petroleum Engineers, 1 1989.

[19] CMG, *CMOST*. Computer Modelling Group, 2019.

[20] CMG, *DECE*. Computer Modelling Group, 2019.

[21] J. Hou, K. Zhou, X.-S. Zhang, X.-D. Kang, and H. Xie, "A review of closed-loop reservoir management," *Petroleum Science*, vol. 12, no. 1, pp. 114–128, 2015.

APPENDIX A

ADDITIONAL FIGURES

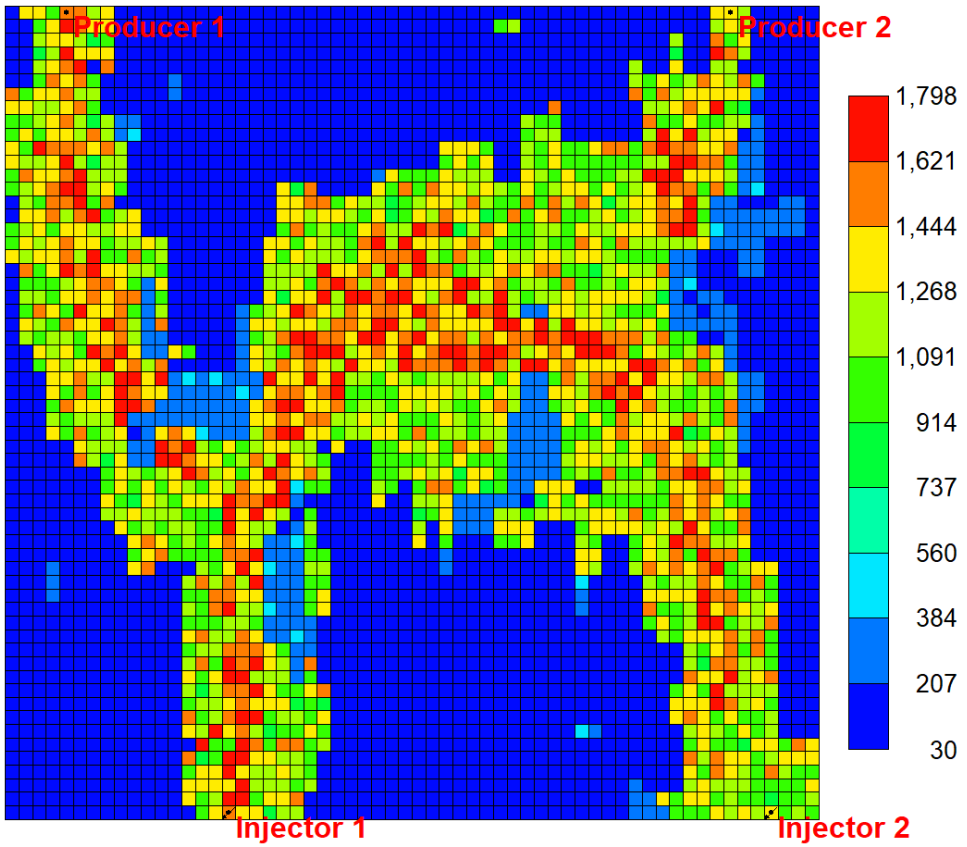

Figure A.1: Showcase model with permeability realization (Permeability Units: md)

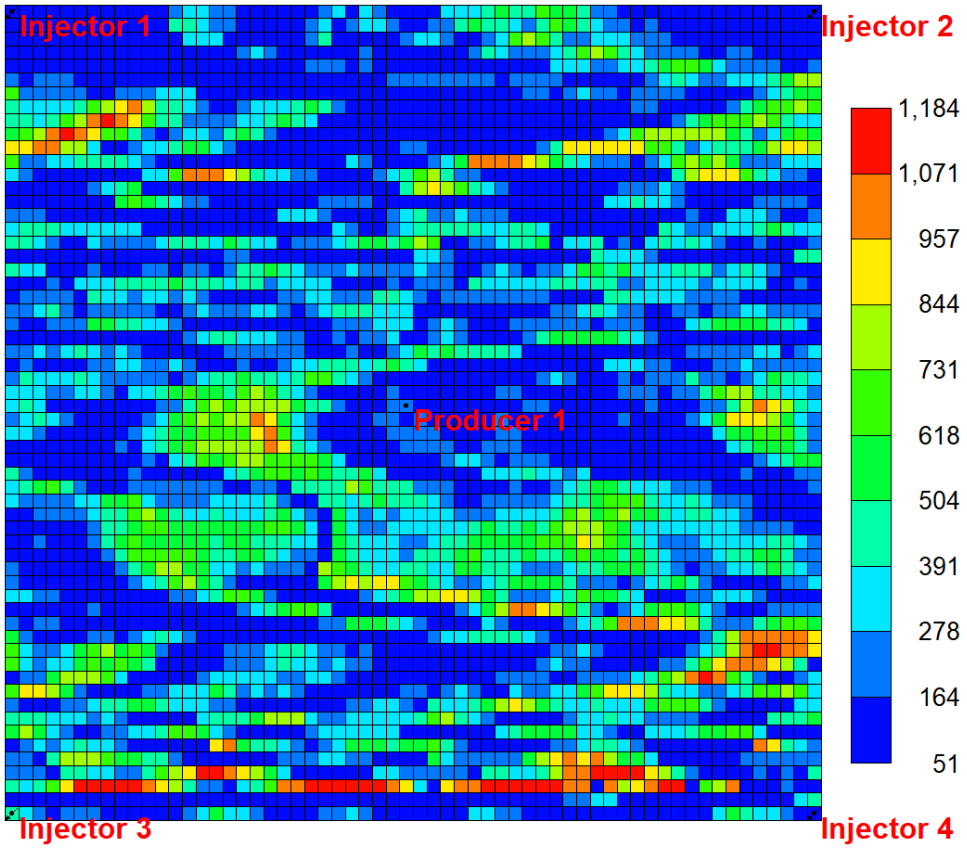

Figure A.2: 5-Spot model with permeability realization (Permeability Units: md)

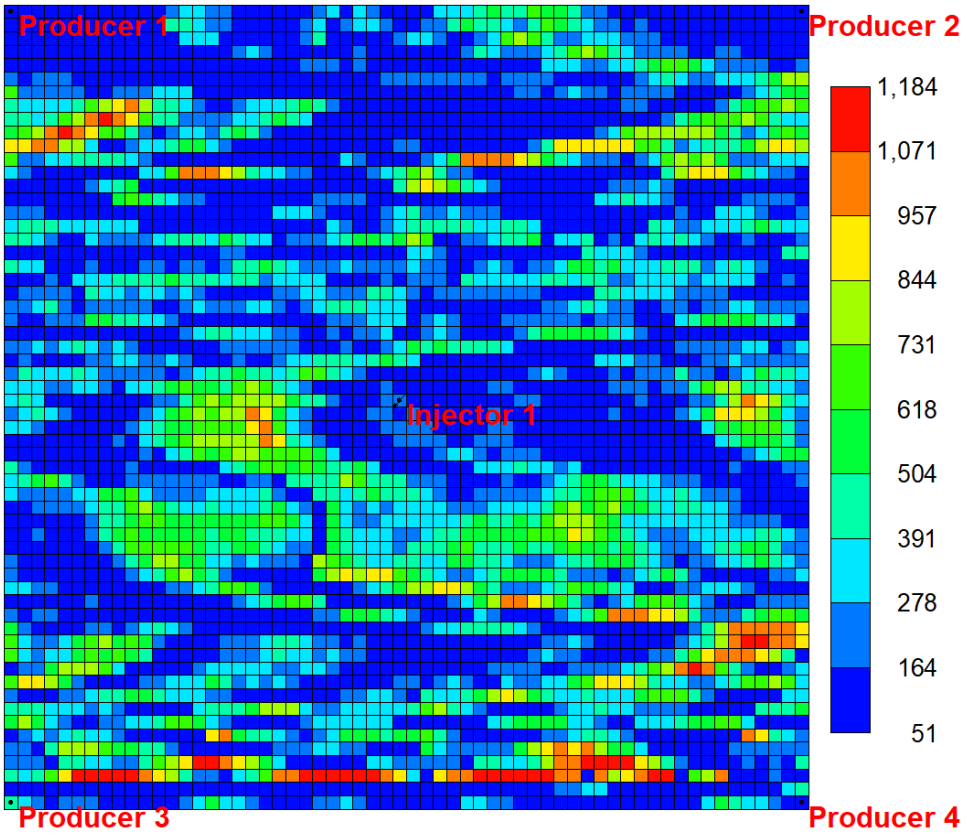

Figure A.3: Inverted 5-spot model with permeability realization (Permeability Units: md)

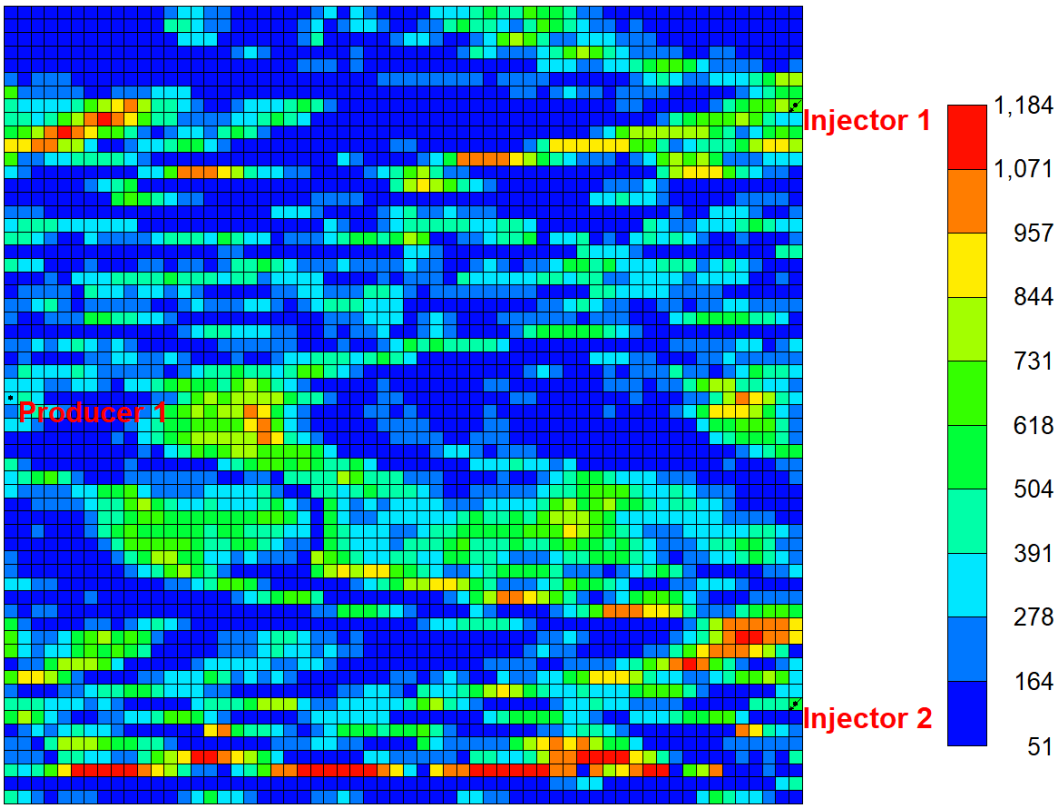

Figure A.4: Staggered Line Drive model with permeability realization (Permeability Units: md)

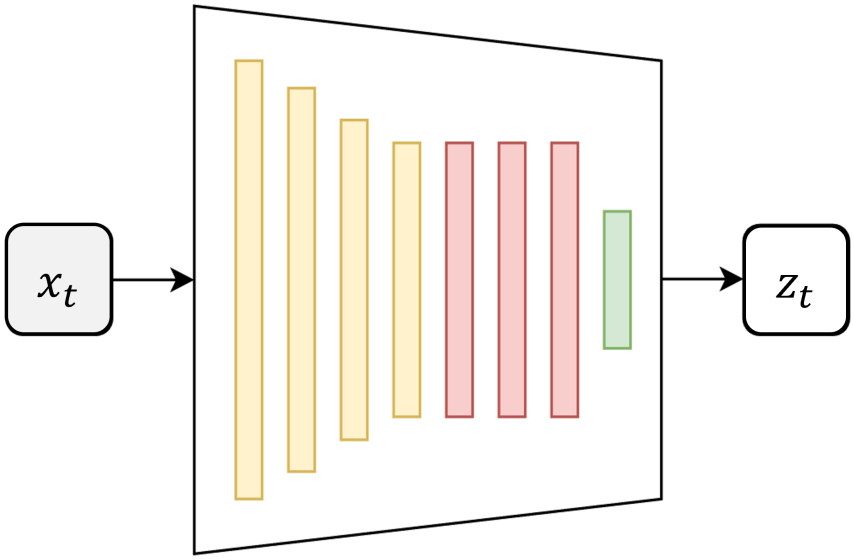

Figure A.5: Encoder Neural Network Structure. Yellow: Encoding Block, Red: Residual Convolution Block, Green: Dense Layer (Adapted from Coutinho et al.)

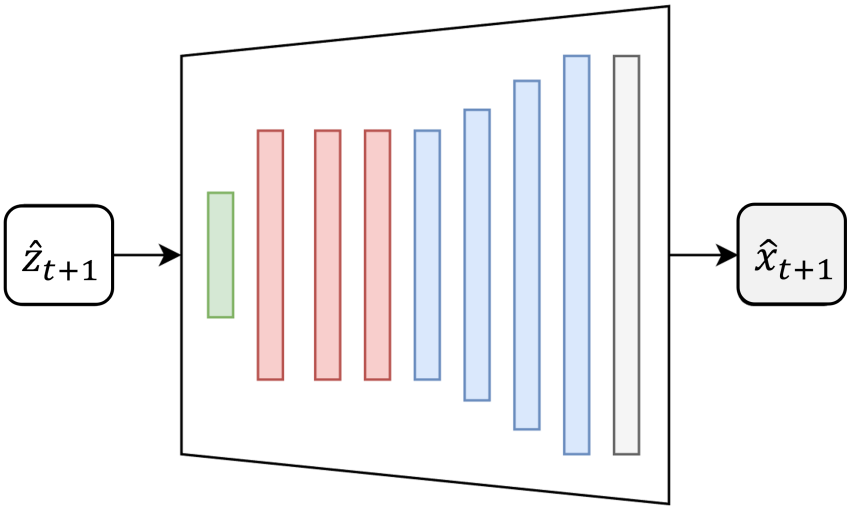

Figure A.6: Decoder Neural Network Structure. Grey: Convolutional 2D Layer, Blue: Decoding Block, Red: Residual Convolutional Block, Green: Dense Layer (Adapted from Coutinho et al.)

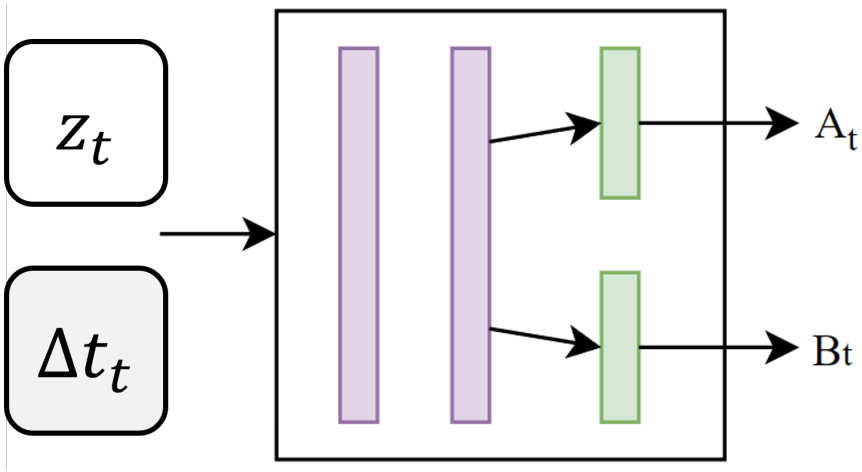

Figure A.7: Transition Neural Network Structure. Purple: Transformation Block, Green: Dense Layer (Adapted from Coutinho et al.)

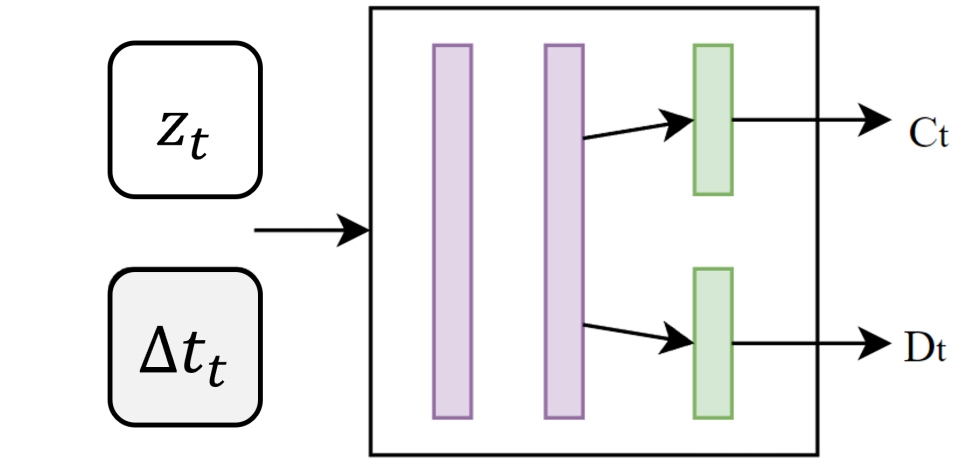

Figure A.8: Transition Well Data Neural Network Structure. Purple: Transformation Block, Green: Dense Layer (Adapted from Coutinho et al.)

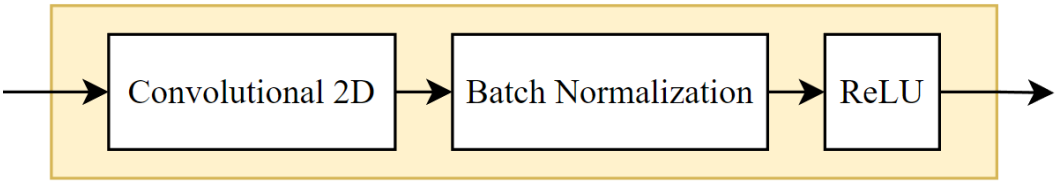

Figure A.9: Encoding Block Structure (Adapted from Coutinho et al.)

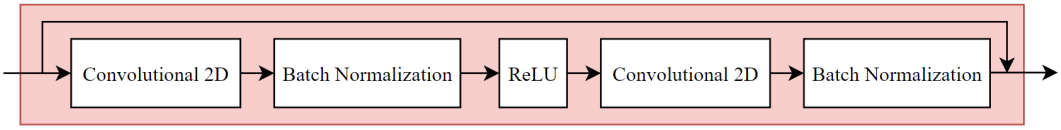

Figure A.10: Residual Convolution Block Structure (Adapted from Coutinho et al.)

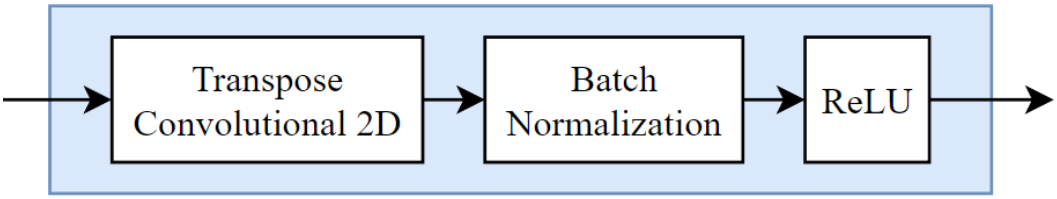

Figure A.11: Decoding Block Structure (Adapted from Coutinho et al.)

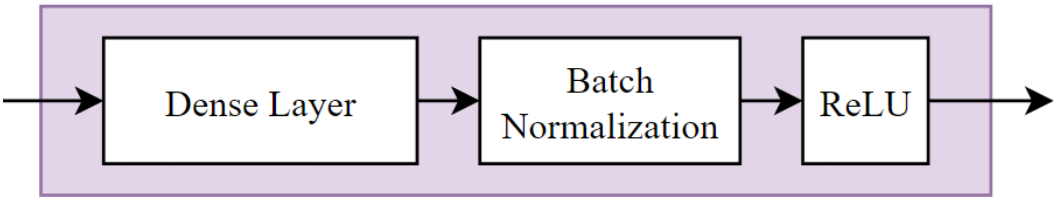

Figure A.12: Transformation Block Structure (Adapted from Coutinho et al.)

APPENDIX B

IMEX .dat file Example (5-Spot)

```
INUNIT MODSI
WSRF WELL TIME
WSRF GRID TIME
WSRF SECTOR TIME
OUTSRF WELL LAYER NONE
OUTSRF RES ALL
OUTSRF GRID SO SW PRES KRO KRW VISO VISW BO
WPRN GRID 0
OUTPRN GRID NONE
OUTPRN RES NONE
**  Distance units: m
RESULTS XOFFSET           0.0000
RESULTS YOFFSET           0.0000
RESULTS ROTATION          0.0000  **  (DEGREES)
RESULTS AXES-DIRECTIONS 1.0 -1.0 1.0
*************************************************
**    Definition of fundamental cartesian grid    **
*************************************************
GRID VARI 60 60 1
KDIR DOWN
DI IVAR
60*10
DJ JVAR
60*10
DK ALL
3600*20
DTOP
3600*3000
**  0 = null block, 1 = active block
NULL CON            1
INCLUDE 'permi.inc'
PERMJ  EQUALSI
PERMK  EQUALSI
**  0 = pinched block, 1 = active block
PINCHOUTARRAY CON            1
POR CON          0.2
PRPOR 350
CPOR 1E-6
PVCUTOFF 0
***************************************
**              Fluid                **
***************************************
```

```
MODEL OILWATER
TRES 100
**      p        Rs        Bo        Zg      viso      visg
PVT ZG 1
      1.0       1.0    0.9999         1      0.71    0.0001
    300.0       1.2       1.0         1      0.71    0.0001
    380.0       1.4    1.0001         1      0.71    0.0001
BWI 1
CO 0
CVW 0
CW 0
DENSITY OIL 800
DENSITY WATER 1000
REFPW 350
VWI 0.81
GRAVITY GAS 1
****************************************
**          Rock-Fluid                **
****************************************
ROCKFLUID
KROIL STONE1 SO
RPT 1
** Sw        krw         krow
SWT
0.1000000000 0.0000000000 1.0000000000
0.1035000000 0.0000000026 0.9821166882
0.1070000000 0.0000000315 0.9644655077
0.1140000000 0.0000003825 0.9298520920
0.1210000000 0.0000016467 0.8961449120
0.1280000000 0.0000046386 0.8633292176
0.1350000000 0.0000103577 0.8313903495
0.1525000000 0.0000445852 0.7552839186
0.1700000000 0.0001255943 0.6843417468
0.2050000000 0.0005406280 0.5570678985
0.2400000000 0.0015229232 0.4478411055
0.3100000000 0.0065555106 0.2769191208
0.3800000000 0.0184665588 0.1589807353
0.4500000000 0.0412346222 0.0824692444
0.5200000000 0.0794903677 0.0369331176
0.5900000000 0.1384595604 0.0131110211
0.6600000000 0.2239205527 0.0030458463
0.6950000000 0.2785339492 0.0010812560
0.7300000000 0.3421708734 0.0002511886
0.7475000000 0.3776419593 0.0000891704
0.7650000000 0.4156951747 0.0000207153
0.7720000000 0.4316646088 0.0000092772
0.7790000000 0.4480724560 0.0000032933
0.7860000000 0.4649260460 0.0000007651
0.7930000000 0.4822327539 0.0000000631
```

```
0.7965000000 0.4910583441 0.0000000052
0.8000000000 0.5000000000 0.0000000000
**************************************
**       Initial Conditions           **
**************************************
INITIAL
**VERTICAL BLOCK_CENTER WATER_OIL
USER_INPUT
SW CON         0.10
PRES CON          350
PB CON            0
**************************************
**            Numerical               **
**************************************
NUMERICAL
NCUTS 100
** DTMAX 0.1
** DTMIN 0.01
RUN
**************************************
**             Wells                  **
**************************************
DATE 2020 1 1
**.
**
WELL  'Injector 1'
INJECTOR MOBWEIGHT 'Injector 1'
INCOMP  WATER
OPERATE  MAX  STW  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.0762  0.37  0.5  0.0
      PERF      GEOA  'Injector 1'
** UBA             ff          Status  Connection
    1 1 1         1.0  OPEN    FLOW-FROM  'SURFACE'
**
**
WELL  'Injector 2'
INJECTOR MOBWEIGHT 'Injector 2'
INCOMP  WATER
OPERATE  MAX  STW  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.0762  0.37  0.5  0.0
      PERF      GEOA  'Injector 2'
** UBA             ff          Status  Connection
    60 1 1         1.0  OPEN    FLOW-FROM  'SURFACE'
**
**
WELL  'Injector 3'
INJECTOR MOBWEIGHT 'Injector 3'
```

```
INCOMP  WATER
OPERATE  MAX  STW  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.0762  0.37  0.5  0.0
      PERF      GEOA  'Injector 3'
** UBA             ff          Status  Connection
    1 60 1         1.0  OPEN    FLOW-FROM  'SURFACE'
**
**
WELL  'Injector 4'
INJECTOR MOBWEIGHT 'Injector 4'
INCOMP  WATER
OPERATE  MAX  STW  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.0762  0.37  0.5  0.0
      PERF      GEOA  'Injector 4'
** UBA             ff          Status  Connection
    60 60 1         1.0  OPEN    FLOW-FROM  'SURFACE'
**
**
WELL  'Producer 1'
PRODUCER 'Producer 1'
OPERATE  MIN  BHP  50.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.0762  0.37  0.5  0.0
      PERF      GEOA  'Producer 1'
** UBA              ff          Status  Connection
    30 30 1         1.0  OPEN    FLOW-TO  'SURFACE'
*INCLUDE 'schedule.inc'
STOP

RESULTS SPEC 'Permeability K'
RESULTS SPEC SPECNOTCALCVAL -99999
RESULTS SPEC REGION 'All Layers (Whole Grid)'
RESULTS SPEC REGIONTYPE 'REGION_WHOLEGRID'
RESULTS SPEC LAYERNUMB 0
RESULTS SPEC PORTYPE 1
RESULTS SPEC EQUALSI 0 1
RESULTS SPEC SPECKEEPMOD 'YES'
RESULTS SPEC STOP
RESULTS SPEC 'Oil Saturation'
RESULTS SPEC SPECNOTCALCVAL -99999
RESULTS SPEC REGION 'All Layers (Whole Grid)'
RESULTS SPEC REGIONTYPE 'REGION_WHOLEGRID'
RESULTS SPEC LAYERNUMB 0
RESULTS SPEC PORTYPE 1
RESULTS SPEC CON 0.9
RESULTS SPEC SPECKEEPMOD 'YES'
RESULTS SPEC STOP
```

```
RESULTS SPEC 'Permeability J'
RESULTS SPEC SPECNOTCALCVAL -99999
RESULTS SPEC REGION 'All Layers (Whole Grid)'
RESULTS SPEC REGIONTYPE 'REGION_WHOLEGRID'
RESULTS SPEC LAYERNUMB 0
RESULTS SPEC PORTYPE 1
RESULTS SPEC EQUALSI 0 1
RESULTS SPEC SPECKEEPMOD 'YES'
RESULTS SPEC STOP
```