GLOBAL OPTIMIZATION STRATEGIES FOR WELL TEST IN SINGLE AND DUAL POROSITY RESERVOIRS

BY

ALI AHMED AL-NEMER

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

PETROLEUM ENGINEERING

December 2013

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Ali Ahmed Al-Nemer** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN PETROLEUM ENGINEERING.**

Dr. Abeeb Awotunde
(Advisor)

Dr. Abdullah Sultan
Department Chairman

Dr. Hasan Al-Hashim
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. Mohammad Issaka
(Member)

14/1/14
Date

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

DEDICATION

Fatimah, Amal & Abdulaziz. If it is not for you, it is not for anyone else.

ACKNOWLEDGMENTS

I would like to express my thanks and appreciation to my thesis supervisor Dr.Abeeb Awotunde for his support, dedication and encouragements to complete this study. I would like also to thank the committee members, Dr. Hassan Al-Hashim and Dr. Mohammad Issaka for their support, feedback directions.

I would like also to express my appreciation to the Department of Petroleum Engineering at King Fahad University of Petroleum and Minerals for their proficiency and providing knowledge.

TABLE OF CONTENTS

DEDICATION ..... **IV**

ACKNOWLEDGMENTS ..... **V**

TABLE OF CONTENTS ..... **VI**

LIST OF TABLES ..... **VIII**

LIST OF FIGURES ..... **IX**

LIST OF ABBREVIATIONS ..... **XI**

ABSTRACT ..... **XIII**

ملخص الرسالة ..... **XV**

1 CHAPTER 1 INTRODUCTION **1**

1.1 Statement of the Problem **4**

1.2 Proposed Solution **8**

1.3 Study Objective **9**

1.4 Proposed Approach **10**

2 CHAPTER 2 LITERATURE REVIEW **12**

2.1 Pressure Transient Analysis Evolution **12**

2.1.1 Dual Porosity Model **12**

2.1.2 Horizontal Well Model **14**

2.2 Type-Curve Matching Evolution **17**

3 CHAPTER 3 OPTIMIZATION ALGORITHMS **23**

3.1 Stochastic Optimization 25

3.1.1 Particle Swarm Optimization 26

3.1.2 Differential Evolution 29

3.1.3 Local Unimodal Sampling (LUS) 32

3.1.4 Many Optimization Liaisons (MOL) 34

4 CHAPTER 4 TRANSIENT PRESSURE MODELS 35

4.1 Dual Porosity Model 35

4.1.1 Model Description 35

4.1.2 Mathematical Formulation 37

4.2 Horizontal Well Model 39

4.2.1 Model Description 39

4.2.2 Mathematical Formulation 42

5 CHAPTER 5 EXPERIMENTAL WORK & FINDINGS 47

5.1 Adding Noise 47

5.2 Finding Optimum Behavioral Parameters 48

5.2.1 Dual Porosity model 49

5.2.2 Horizontal Well Model 71

5.3 Comparisons of Model Runs 82

5.3.1 Dual Porosity model 82

5.3.2 Horizontal Well model 84

6 CHAPTER 6 CONCLUSION AND RECOMENDATIONS 87

VITAE 106

LIST OF TABLES

Table 5-1 PSO algorithm recommended behavioral parameters 52
Table 5-2 DE performance summery using 30 agents and different behavioral parameters 59
Table 5-3 MOL algorithm recommended behavioral parameters..... 67
Table 6-1 Summery of best behavioral parameters 87

LIST OF FIGURES

Figure 1.1 Global and local minimum 5
Figure 1.2 Steps followed to compare stochastic optimization algorithms performance . 11
Figure 2.1 Ramey type-curve..... 18
Figure 2.2 Bourdet type curve 19
Figure 3.1 PSO Topology 28
Figure 3.2 A basic scheme of evolutionary algorithms 30
Figure 3.3 LUS algorithms 33
Figure 4.1 Warren and Root dual porosity reservoir model 35
Figure 4.2 Dual porosity flow systems 36
Figure 4.3 Dual Porosity flow regimes on loglog plot..... 37
Figure 4.4 Schematic of horizontal well in a box-shaped reservoir 40
Figure 4.5 Horizontal well flow regimes on loglog plot..... 41
Figure 4.6 Schematic of horizontal well flow regimes 42
Figure 5.1 A dual porosity model before and after adding noise..... 48
Figure 5.2 Dual Porosity model used in the test before and after adding noise 49
Figure 5.3 All PSO test runs result 50
Figure 5.4 Best 16 PSO runs parameters 51
Figure 5.5 PSO performance using different behavioral parameters and 60 particles 53
Figure 5.6 PSO Performance using different population size..... 55
Figure 5.7 Performance of dynamic and static PSO 57
Figure 5.8 DE performance using 30 agents and different behavioral parameters..... 58
Figure 5.9 DE performance summery using 30 agents and suggested behavioral parameters 60
Figure 5.10 DE performance summery using different number of agents 62
Figure 5.11 Typical behavior of LUS algorithm 63
Figure 5.12 Modified version of LUS..... 64
Figure 5.13 LUS performance using 25 particles and different configuration 65
Figure 5.14 LUS performance using different population size..... 66
Figure 5.15 MOL performance using different behavioral parameters and 60 particles.. 68
Figure 5.16 MOL Performance using different population size 70
Figure 5.17 Horizontal well model used in the test before adding noise..... 71
Figure 5.18 PSO performance using different behavioral parameters and 60 particles (Horizontal Well Model) 72
Figure 5.19 PSO performance using different population size (Horizontal Well Model) 73
Figure 5.20 DE performance summery using 30 agents and suggested behavioral parameters 75
Figure 5.21 DE performance summery using different number of agents 76
Figure 5.22 LUS performance using 25 particles and different configuration 78

Figure 5.23 LUS performance using different population size..... 79
Figure 5.24 MOL performance using different behavioral parameters and 60 particles.. 80
Figure 5.25 MOL performance using different population size 81
Figure 5.26 Comparison between PSO, DE, LUS, MOL and Levenburg-Maquart optimizing dual porosity model..... 83

LIST OF ABBREVIATIONS

a	:	Reservoir length (x-direction), ft
b	:	Reservoir width (y-direction), ft
B_g	:	Gas formation volume factor, ft^3/SCF
c	:	Compressibility, psi^{-1}
c_t	:	Total compressibility, psi^{-1}
C_D	:	Dimensionless wellbore storage
C	:	Wellbore storage, bbl/psi
a	:	Reservoir length(x-direction), ft
h	:	Reservoir thickness, ft
$k_x, k_y \& k_x$	:	Permeability in x, y & z direction
k_f	:	Fracture permeability, md
L	:	Length of well, ft
L_D	:	Dimensionless well length
P	:	Pressure, psia
P_i	:	Initial pressure, psia
P_{WD}	:	Dimensionless wellbore pressure
P_{WD}	:	Dimensionless wellbore pressure
q	:	Flow rate, STB/day
P_D	:	Dimensionless pressure
r_e	:	External boundary radius, ft
r_w	:	Wellbore radiuses, ft
r_{we}	:	Effective wellbore radiuses, ft
s	:	Skin factor

t	:	Time, hours
t_D	:	Dimensionless time
T	:	Temperature, °F
w	:	Inertia weight

Greek symbols

$\emptyset_f$	:	Fracture Porosity
$\emptyset_m$	:	Matrix Porosity
ω	:	Storativity ratio
λ	:	Interporosity flow ratio
θ	:	Angle
μ	:	Viscosity, cp
ρ	:	Dinsity, lb/ft^3

Subscripts

0	:	Starting location
1	:	Ending location
D	:	Dimensionless
f	:	Fracture
i	:	Initial
o	:	Oil
x	:	x-direction
y	:	y-direction
z	:	z-direction

ABSTRACT

Full Name : Ali Ahmed Al-Nemer

Thesis Title : Global Optimization Strategies for Well Tests in Single and Dual Porosity Reservoirs

Major Field : Petroleum Engineering

Date of Degree : 2013

This study presents an investigation of the performance of multiple stochastic optimization algorithms in performing automatic type-curve matching in pressure transient well test analysis. The primary objective is to evaluate their performance and to find the optimum behavioral parameters of each algorithm in two reservoir models which are a horizontal well model in a box-shaped reservoir and a vertical well in a dual porosity reservoir. The pressure transient response of these models was generated. A synthetic reservoir model that shows all flow regimes for both models was created. Gaussian White Noise data was added to the typical response to imitate measured data. In addition to the Levenberg-Marquardt algorithm, four stochastic algorithms were used to estimate the reservoir parameters from the noisy data. These algorithms are Deferential Evolution, Particle Swarm Optimization, Local Unimodal Sampling and Many Optimizing Liaisons. Behavioral parameters of each algorithm were investigated by comparing the performance of recommended values in the literature. Each algorithm was run for 25 realizations. The results of the runs were ordered in terms of the best achieved result. The performance was compared by comparing best 1^{st}, 7^{th}, 19^{th} and 25^{th} results of each algorithm. The result showed that the algorithms performance is affected by the model and the number of unknowns. Differential evolution algorithms showed the best performance in Dual Porosity when $\emptyset_m, \lambda, \omega, skin, r_e \; and \; k_f$ are the unknowns, and in horizontal well model when the unknowns are $k_x, k_y, k_z, skin, h$ and the length of the parallel boundary. Other stochastic algorithms also showed better performance in dual porosity model, whereas Levenberg-Marquardt algorithm performed better than Particle

Swarm Optimization, Local Unimodal Sampling and Many Optimizing Liaisons in horizontal well model.

ملخص الرسالة

الاسم الكامل:علي أحمد النمر

عنوان الرسالة: إستراتيجيات التحسين الشاملة في إختبار الآبار في المكامن الفردية والمتعددة المسامية

التخصص:هندسة بترول

تاريخ الدرجة العلمية: 2013

تسلط هذه الدراسة الضوء على فعالية مجموعة من الخوارزميات الغير قطعية في التعرف على معايير المكامن عن طريق قراءة البيانات المستخلصة من عملية اختبار موجات الضغط العابرة في آبار النفط واستخلاص المعايير منها عن طريق إعادة التنبؤ بمعايير المكامن. استخدم في هذه الدراسة نموذجي الآبار الرأسية المحفورة في مكامن ثنائية النفاذية ذات شكل اسطواني, والآبار الأفقية المحفورة في مكامن أحادية النفاذة ذات شكل متوازي المستطيلات. تمت الدراسة عن طريق استخلاص معايير لكل نموذج تبين جميع أنواع التدفقات المحتملة, ثم إضافة تشويش لبيانات كل نموذج من أجل محاكاة عمليات القياس. تم اختيار أربع خوارزميات بالإضافة إلى خوارزمية ليفينبرق-ماركوارت للقيام بعملية استخلاص المعايير من البيانات الجديدة وهي خوارزمية الطفرات الوراثية, خوارزمية الأفراد والسرب, خوارزمية العلاقات التبادلية وخوارزمية العينات المحلية أحادية الواسطة. تم تحديد أفضل المعايير التحكمية لكل من هذه الخوارزميات , عن طريق مقارنة أدائها لكل المعايير المقترحة. تم بعد ذلك مقارنة أداء هذه الخوارزميات مع بعضها البعض عن طريق تنفيذ كل خوارزمية 25 مرة ومقارنة أداء أول, سابع, تاسع, ثالث عشر, تاسع عشر وآخر أفضل أداء لكل خوارزمية. استخلصت الدراسة أن الخوارزميات الغير قطعية ذات أداء أفضل في نموذج المكامن متعددة المسامية عندما يكون البحث عن المعايير التالية:تباين القدرة التخزينية, تباين النفاذية, نصف قطر المكمن, مسامية الصخور, نفاذية الشقوق, ومعامل التلف. واستخلصت الدراسة أن خوارزمية ليفينبرق-ماركوارت تكون ذات أداء أفضل في نموذج الأبار الأفقية في المكامن أحادية المسامية عندما يكون البحث عن المعايير التالية: النفاذية السينية, النفاذية الصادية , النفاذية العامودية , معامل التلف, سماكة المكمن وطول الحدود الموازية للبئر.

CHAPTER 1

INTRODUCTION

Pressure transient analysis (PTA) is one of the most powerful tools used to characterize hydrocarbon reservoirs. It is commonly used to determine reservoir properties such as horizontal and vertical permeability, formation damage and stimulation, productivity or injectivity index, reservoir pressure, distance to boundaries and fluid fronts, reservoir volume, interwell connectivity, hydraulic fracture evaluation, heterogeneity and natural fractures and reservoir geometry. It depends on the fact that the pressure wave takes time to diffuse through the porous medium from the point of disturbance, which is the well in our case, until it reaches to the reservoir boundary. Transient pressure response is created by a temporary change in production rate. During this period, which is called transient period, the pressure behavior is affected by many well, reservoir and boundary parameters. It was found that each combination of well, reservoir and boundary model gives a distinctive behavior from which the parameters can be calculated. Finding these parameters is a reverse problem. This means that the reservoir response is used to estimate the reservoir properties. To estimate the parameters, the typical reservoir represented by the measured pressure data should be matched to the generated reservoir behavior from a computer-generated model. Computer-aided type-curve matching and nonlinear regression have become the industry standards in estimating the parameters. Since there is the possibility of multiple solutions, optimization algorithms are used to estimate the matching parameters. They start with initial solution, compare the generated model with measured data

and find the error. The model parameters are changed continually in order to minimize the error until an acceptable tolerance is attained.

Horizontal Well technology is becoming more popular in the oil and gas industry. Because of the high demand for oil and high oil prices, the high cost of drilling horizontal well has become justifiable. This technology has proved capable of enhancing oil recovery, reaching unproduced zones and producing from thin formations. Despite its advantages, horizontal well technology has introduced more complicated models that need to be addressed. Pressure transient behavior is one of these complications.

The theory of dual porosity reservoir model was first introduced in the 1960's by Warren and Root (Root *et al*., 1963). This model is used to describe naturally fractured reservoirs. In this model, two different media are involved in the flow process: a high permeability, low storativity network of connected fractures and lower permeability, higher storativity matrix that recharges the fractures with fluids. The well is connected only to the fractures and cannot flow from the matrix. All produced fluids come through the fractures. This model is similar to a formation with multilayered reservoirs.

Problem optimization is the process of finding the candidate solution with the highest quality. Considering X to be the set of candidate solutions to the problem, the optimization problem can be defined by the fitness or objective function. The objective function rates how well the candidate solutions in X fare on the given problem:

obj: $X \rightarrow R$

Optimization process aims to minimize the objective function and obtain the candidate solution $x \in X$ that fare best such that:

$$\forall y \in X : \text{obj(x)} \leq \text{obj(y)}$$

The simplest way of performing optimization is when the optimization problem can be expressed in simple formula, and then it may be possible to invert this formula to find the problem. In numerical optimization, the initial guess is selected either randomly or at what the optimum solution might be. This solution is refined until the fitness is small enough. Newton-Raphson method which was published in 1685 by John Wallis is the most classical way of numerical optimization that uses derivative, and iterates until it find the solution for a single-dimensional function. Newton-Raphson method can be generalized for multi-dimensional search space by using Quasi-Newton method.

Local search algorithms (LS) are widely used to optimize problems. These types of algorithms only accept moving to better values of the objective function. If the optimum set of the parameters are far from the optimum solution, the search algorithm most likely will get stuck in local optima. The disadvantages of LS algorithms are:

- There is high probability that these algorithms terminate in local minimum. In fact, without prior knowledge, there is always the possibility that the global optimum solution lies in unexplored region of the search space.
- Obtained local minimum depends on the initial selection of the parameters which does not follow any guideline.

Avoiding some of the disadvantages can be achieved by:

- Running the algorithm for large number of initial set of parameters.
- Using information gained from previous runs to improve the selection of parameters for the next run.
- Providing a mechanism to jump from local optimum.

Stochastic algorithm is a numerical method that involves randomness in its procedure of searching for the global optimum solution of the objective function. Unlike LS algorithms, stochastic algorithms implement the above mechanism to avoid falling in local optima. Using a random factor guarantees that a different path is followed in each iteration in order to expand the search space region by following different paths. After each iteration, prior knowledge of previous iteration is involved in the next movement. The use of stochastic algorithms has been growing rapidly over the last decades and it is becoming the standard in many industries.

1.1 Statement of the Problem

In well test pressure transient analysis, collected pressure data is used to estimate the reservoir parameters. A type-curve is fitted to the measured data by nonlinear regression. Nonlinear regression methods minimize the error between measured data points and the calculated data points from the expected model using the following formula:

$$\text{obj}(\overrightarrow{\alpha}) = \sum_0^n (P_{n\,calculated} - P_{n\,measured})^2 \qquad (1.1)$$

Where n = number of measured data points, and $\vec{\alpha}$ is the vector of parameters. Equation 1.1 is considered the objective function. Nonlinear regression is usually performed using Newton and Newton-like algorithms. These algorithms start from an initial point, and move to the closest

point where the error seems to be the minimum. This point is called *local minimum*. In each problem, more than one local minimum can exist whereas only one global minimum exists. The global minimum is the optimum solution where the error is smaller than any other point in the search space. **Figure 1.1** illustrates an example of global and local minimum. The function f(x) has local minimum at x= A, B, C and D and the global minimum at x=G.

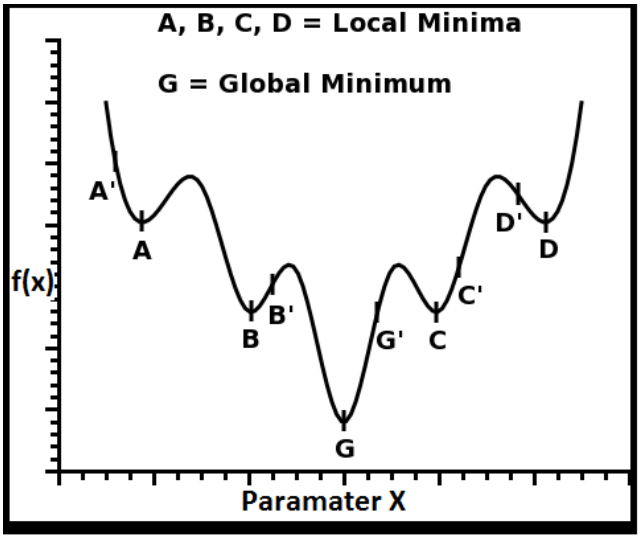


Figure 1.1Global and local minimum

Depending on the selection of the starting point, the point that the optimization algorithm finds can be a local or a global minimum. Since these algorithms start from one point and follow one path to the nearest minimum value of the objective function, there is a high probability that this point is a local minimum whereas the global minimum is at another point in the solution space. This probability increases as the number of the objective function parameters increases. Until now, there has been no way to prove that a point is a global minimum except by knowing the

value of the objective function at each point. This means that the function has to be evaluated at each point in the search space. Because some parameters may be continuously varying, this is impossible. This thesis will evaluate the performance of five stochastic optimization algorithms in the optimization of two reservoir models, a single and dual porosity model.

In well test pressure transient analysis (PTA), the objective is to determine the model parameters. Assuming that the measured data has only insignificant measurement error, the actual model parameters give the smallest error at the global minimum point. Local minimum points can be far from the global minimum. Reading reservoir parameters from local minimum can lead to significant error in determining the actual reservoir parameters. Because PTA is an inverse process, it is not necessary to have a unique solution of each problem. More than one solution can give the same result and therefore the same value of the objective function. **Appendix C** shows an example of the relation between the objective function and one of the optimized parameters. A synthetic dual porosity model was generated and noise added to the data. Five optimization algorithms were then used under the following scenarios:

Case1: The algorithms are optimizing for a problem with 6 unknowns. The range for the unknowns is relatively small.

Case 2: The algorithms are optimizing for a problem with 6 unknowns. The range for the unknowns is relatively high.

Case 3: The algorithms are optimizing for a problem with 3 unknowns. The range for the unknowns is relatively small.

They show that some factors that affect the performance of the optimization process can be:

1- **Number of the unknowns:** as the number of the unknown increases, more solution could be found. Increasing the number of the unknowns can increase the number of possible solutions, complicates the solution space geometry and therefore increase the number of local minimums. As the number of unknowns decreases, the solution space geometry becomes simpler. By comparing case 1 and case 3, the achieved fitness for all the algorithms decreased as the number of the unknowns decreased. Also, the obtained skin value became closer to the actual skin value, which is 5, when the number of unknown decreased.

2- **Range of values of the unknowns:** as the range of the unknowns increased, the solution space is increased and become more complicated. In real tests, other sources of data can be used to minimize these ranges such as drilling logs, core data and nearby wells. When comparing case 1 and case 2, it was found that increasing the difference between upper and lower limits causes an increase in difference between the obtained skin and the actual skin, which consequently increased the best achieved fitness.

The example in **Appendix C** shows also that in inverse problems, such as PTA, the value of the best achieved fitness is an indication of how the candidate solution can give a close result to the input data but it does not necessarily reflect the correctness of this solution. For example, in Case 3, DE algorithm achieved lower fitness than MOL in Case 1, but MOL estimated a skin value lower than actual skin, whereas DE estimated a skin value higher than the actual value by 0.2. This could happen when changing the value of two or more unknowns at the same time. If one of the parameters has more influence on the value of the objective function, it will give a lower fitness which consequently is considered a better solution. For example, changing the skin value can shift the whole pressure curve up or down whereas changing the wellbore storage affects only the early time period. Addressing the problem of the objective function usage as an

indication is out of the scoop of this thesis, and the objective function will be used as the means to evaluate how good the algorithm is performing.

1.2 Proposed Solution

Unlike non-linear regression algorithms, many stochastic algorithms use multiple agents to start from multiple random initial points. Each agent follows a different path to search for the minimum value of the objective function. This increases the search range and, therefore, increases the probability of finding the global minimum of the objective function. Using stochastic algorithm to do automatic type-curve matching improves the possibility of finding the global optimum parameters and therefore finding the correct model parameters.

In this study, the effectiveness of using four different stochastic algorithms to estimate the global optimum parameters were evaluated by fitting the modeled transient pressure response to measured data. A horizontal well in a box-shaped reservoir and a vertical well in a dual porosity reservoir model were used. The algorithms were run to estimate the following parameters for dual porosity reservoir model:

- Permeability of fracture k_f
- Matrix porosity,$\emptyset_m$
- Skin factor, s
- Radius of the reservoir, r_e
- Storativity ratio ω , and Interporosity flow parameter, λ

In the Horizontal well model, the algorithms were run to estimate the following parameters:

- Reservoir permeability (k_x, k_y and k_z)
- Skin factor, s
- Reservoir thickness, h
- Length of the boundaries parallel to the horizontal section, b

A comparison between all the tested algorithms in addition to a line search algorithm, Levenberg-Marquardt, which are non-linear regression algorithms, was presented.

1.3 Study Objective

The objective of this study is to evaluate the effectiveness of four stochastic optimization algorithms in estimating the global solution parameters of two reservoir models in well test analysis by performing automatic type-curve matching. The models are a horizontal well in a box-shaped homogeneous reservoir and a vertical well in a dual porosity reservoir. Deferential Evolution, Particle Swarm Optimization, Local Unimodal Sampling and Many Optimizing Liaisons algorithms were evaluated. The performances were compared by ordering the realizations from the best 1^{st} to the worst 25^{th} based on the value of the error attained. The best realization then is the one with smallest resulting error while the worst realization is the realization with the largest resulting error. The best 1^{st}, 7^{th}, 19^{th} and 25^{th} realizations of each algorithm were compared.

1.4 Proposed Approach

A computer code that models the transient pressure and pressure derivative responses of a horizontal well in a box-shaped reservoir, with no flow boundaries will be implemented. Skin effect and wellbore storage will be added to the model using numerical Laplace transformation. Another function that models the transient pressure and pressure derivative responses of a vertical well in naturally fractured reservoir will be implemented. Subsequently, a synthetic reservoir model that shows all flow regimes for both models will be generated. Simulating the real data measurement will be done by adding Gaussian White Noise. Using the stochastic algorithms (Deferential Evolution, Particle Swarm Optimization, Local Unimodal Sampling & Many Optimizing Liaisons), the model parameters will be estimated from the noisy data. For Dual Porosity, Warren and Root (1963) model were used. The parameters, $\lambda, k_f, r_e, \omega, \emptyset_m$ and $skin$ were estimated. For the horizontal well model, the parameters $k_x, k_y, k_z, skin, h$ and the length of the parallel boundary to the horizontal section of the well will be estimated. Upper and lower limits will be set for each parameter to help with convergence. For each reservoir model example, each algorithm will be run for 25 realizations. The results will be ordered from the best result to the worst. The algorithms evaluated will be compared by comparing the 1^{st}, 7^{th}, 13^{th}, 19^{th} and 25^{th} realization after ordering. **Figure 1.2** shows the steps followed in this research to compare the performance for each algorithm.

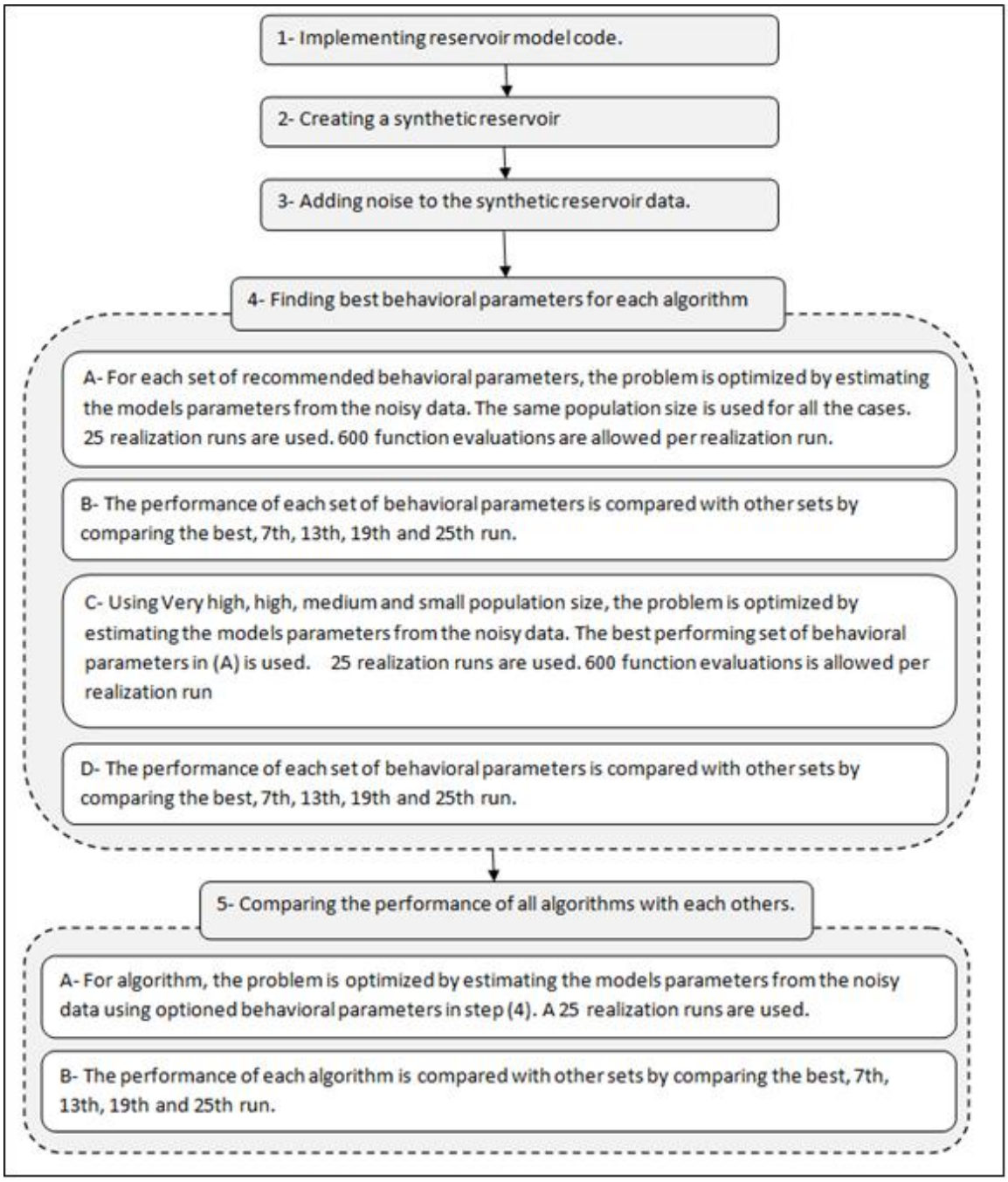


Figure 1.2 Steps followed to compare stochastic optimization algorithms performance

CHAPTER 2

LITERATURE REVIEW

The evolution of well test analysis and computer-aided type-curve matching is presented in this chapter:

2.1 Pressure Transient Analysis Evolution

2.1.1 Dual Porosity Model

Barenblatt and Zheltov (1960) were the first to present a solution to radial flow of a slightly compressible fluid in a naturally fractured reservoir. They assumed that the flow occurs only in the fracture, while the matrix block acts as a uniformly distributed source in a fractured medium.

Warren and Root (1963) developed an analytical model to characterize the behavior of a permeable medium which contains a region that has high storativity but contributes negligibly to the flow. They examined this model for a buildup and suggested a technique for analyzing the buildup data to evaluate the desired parameters. They found that ω, which is a measure of the fluid capacitance of the secondary porosity, and λ, which is related to the scale of heterogeneity that is present in the system, are very important to describe the behavior of this system. They provided an analytical derivation for flow in such systems.

H. Kazemi (1976) did an oil-water multi-well numerical simulation in a fractured reservoir. His work was aimed at extending the work of single phase flow equations of Warren *et al.* (1963). He simulated two conceptual models of two naturally fractured reservoirs. His work showed the

significance of imbibition in recovering oil from the reservoir rock in reservoirs with an interconnected fracture network.

De Swaan (1976) developed a complete unsteady-state theory that describes the pressure response in a naturally fractured reservoir. His theory involves flow properties and the dimensions of fractures and matrix blocks. He compared his theory with a numerical model. He found that it is possible to obtain the fracture k.h and the average product of the matrix porosity by a characteristic dimension of the matrix blocks.

Bourdet *et al*.(1980) presented type-curves for analyzing well tests with wellbore storage and skin in dual porosity reservoir. In addition to the usual reservoir parameters, volume of fissures and the size of porous blocks in the reservoir can be estimated using log-log analysis.

Aguilera (1989) presented an analytical solution for the analysis of dual-porosity systems intercepted by hydraulic vertical fractures of finite conductivity in an infinite or a bounded dual-porosity system. He identified the following four flow periods:

a) A bilinear flow period typical of finite conductivity fractures. This is recognized by a quarter slope in a conventional log-log crossplot of pressure vs. time.
b) A transition period due to flow from the matrix into the natural fractures.
c) A pseudo radial flow period recognized by a straight line in a conventional semi logarithmic plot.
d) Boundary effects which can be due to a sealed boundary or an outer boundary at constant pressure.

2.1.2 Horizontal Well Model

Robert *et al.* (1968) discussed the problem of unsteady depletion and pressure distribution in a rectangular reservoir. They used a model of single vertical well, ideal isotropic, homogenous horizontal formation and no flow boundary. They used line source solution to compute the dimensionless pressure drop if the well is not located at the center.

Daviau *et al.* (1985) presented horizontal well test design and interpretation with and without skin and wellbore storage. They described two flow regimes that may be encountered while flowing a horizontal well which are:

a) A vertical radial flow around the wellbore at early time. The vertical radial flow can be described by the following equation:

$$p_D = \frac{1}{2L_{D3}}\left(\ln\frac{t_D}{x^2{}_{wD}} + 0.809\right) \; for \; \frac{x^2{}_{wD}}{4t_D} < 10^{-2} \qquad (2.1)$$

This flow regime appears as straight line when p_D is plotted vs y_D, e_{D3} or x_{wD}. A plot of $p_D$vs $log\ t_D$ yields a straight line of slop$1.15/2\ L_{D3}$. When the pressure is measured at the end of the horizontal section, the slope is one-half of the one observed for other values. This vertical radial flow continues until one of the vertical boundaries is reached when a transition zoon starts. A horizontal pseudo radial flow occurs at late time. It can be characterized by another straight line that comes after the transition zone onp_D vs. $log\ t_D$ plot, and can be described with the following equation:

$$p_D(t_D) = 0.5(\ln t_D + \alpha) \qquad (2.2)$$

where α is the geometrical skin

b) Pseudoradial flow with the following time range: $0.8 < t_D < 3$.

They also described the solution for both constant pressure and no flow boundary.

Clonts and Ramey (1986) presented an analytical solution for the transient pressure response of a horizontal well in an anisotropic reservoir of finite thickness. Two flow regimes were discussed, depending on the effective dimensionless drainhole half-length. If $L_D < 10$, early radial flow (ERF) can be observed. If $L_D > 10$, the (ERF) ends rapidly and linear flow regime starts, which is identical to uniform flux fracture. The authors used a solution which was derived by Gringarten and Ramey (1973) to the transient response of horizontal drain hole using instantaneous source functions. The solution was verified against 3 cases that were found in the literature; Partially Penetrating Uniform Fracture, General Uniform Flux Fracture solution and Slanted Wellbore. They presented the derivation of the solution and published a set of type curves. These type curves presented typical behavior of predefined dimensionless parameters. Measured data can be fitted to one of the curves and the reservoir parameters can be obtained from the matched curve.

Goode and Thambynayagam (1987) presented an analytical solution for the pressure response during drawdown and buildup of horizontal well by solving a three-dimensional diffusion equation with successive integral transform. Simplified solutions for short, intermediate and long times that exhibit straight-line section when pressure was plotted verses time were presented. They validated the solution against a result which was generated numerically by a reservoir simulator. Straight line, segmented analysis were used to conduct the analysis.

Odeh and Babu (1990) presented four types of the flow regimes that can occur in a horizontal well these are:

a) Early radial

b) Early linear

c) Late pseudoradial

d) Late radial

They presented the pressure equation and start and end times of each flow regime in both cases of buildup and drawdown. Subsequently, they illustrated examples of solving for each flow regime using straight line slope on semilog plot and Horner plot.

Issaka and Ambastha (1992) used numerical integration to evaluate an analytical solution for horizontal well in closed, anisotropic box shaped reservoir. The study showed that numerical integration can be used to evaluate the solution with comprehensive degree of accuracy for both drawdown and buildup. New time criteria, based on the semilog pressure derivative response, were proposed for well test analysis and design purposes. The authors identified each flow regime on loglog plot of pressure derivative as following:

a) A zero-slope line for early radial flow.

b) A half-slope line for early linear flow.

c) A zero-slope line for late pseudoradial flow.

d) A half-slope line for late radial flow

e) A unit-slope line for pseudosteady state flow.

Results showed that the late linear flow period does not occur on buildup response for any case.

2.2 Type-Curve Matching Evolution

Moore *et al.* (1933) developed a 1D radial transient solution. They presented a history matching technique to estimate the formation permeability. Their test considered a drawdown situation where both rate and pressure were measured, followed by a buildup. The match was obtained manually by trial-and-error. The test was very short, 2 hr, and had only 10 data points. The test did not consider skin or wellbore storage since they were formulated later by Van Everding and Hurst (1949).

Miller *et al.* (1950) presented simple and practical method to estimate effective permeability. They showed that plotting bottom hole pressure vs. log of time function yields a straight line which has a slope of m from which the effective permeability can be calculated. This plot was later called MDH method.

D. R Horner (1951) presented the Horner plot for the first time where bottom hole pressure was plotted against logarithm of $\frac{t_p+\Delta t}{\Delta t}$. This plot yields a straight line with a slope m, from which the permeability can be calculated.

Ramey *et al*. (1970) presented a set of type curves for short-time test. These curves consider wellbore storage and skin factor. Drawdown data would be plotted as $P_i - P_{wf}$ in any convenient units vs. time in any convenient units on log-log paper of the same size cycle as in the type curve **Figure 2.1**. The curve is moved manually until the best match is obtained. Subsequently, parameters can be obtained either directly from the predefined values of the matching curve parameters, or calculated.

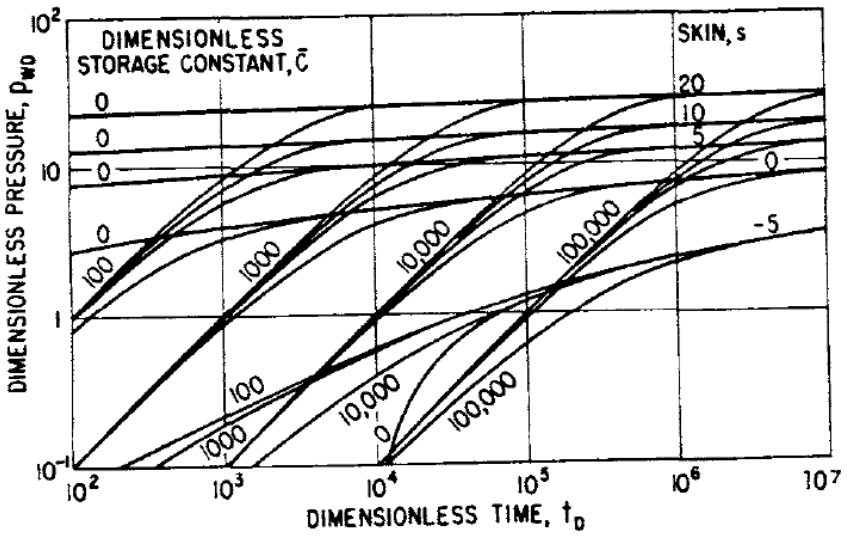


Figure 2.1 Ramey type-curve

Gringarten *et al*. (1979) presented a new type-curve for wells with wellbore storage and skin effects, which was developed according to the predefined rules. It shows a plot of dimensionless pressure P_D vs. ${t_D}/{C_D}$ and each curve is defined by value of $C_D e^{2s}$. This type-curve has been used for the analysis of many well tests.

Bourdet *et al*. (1983) presented a new type curve that combined Gringarten type curves and pressure derivative curves in one plot. The Bourdet derivative can be computed by the following expression, $\frac{dp_D}{\{d \ln[t_p \, \Delta p/ (t_p + \Delta t)]\}}$. The shape of the derivative can clearly show each flow regime and from the derivative value, the model parameters can be calculated. This method enhanced the identification of the flow regimes and improved the possibility of finding a unique match. The Bourdet type curve is shown in **Figure 2.2**.

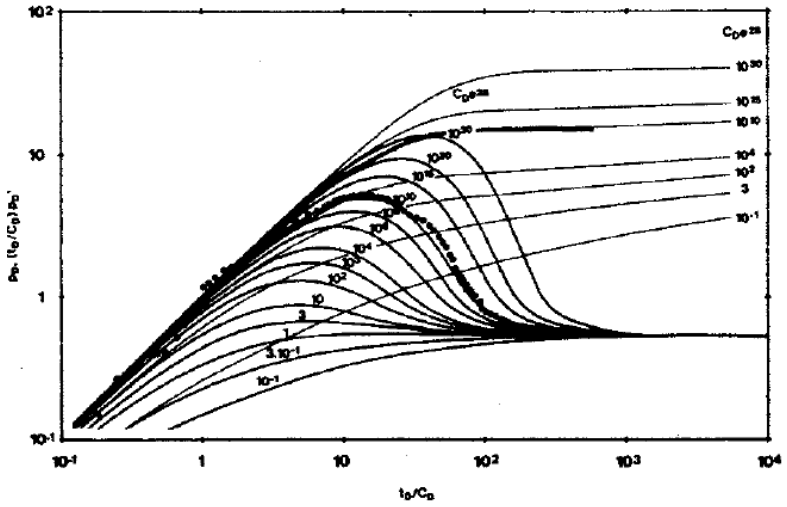


Figure 2.2 Bourdet type curve

Chang and Ershaghi (1986) proposed a methodology to select the initial guess of parameters for the nonlinear regression using derivative/gradient free algorithms. The procedure improved the direct search method, which allows pinpointing the minimum neighborhood. They developed a pre-regression algorithm for estimating a starting point in 2 or more dimensional space. They used an algorithm to compute the coordinates of uniformly distributed solution sets. The solution is then evaluated at each of this data sets. The data set that gives the minimum value is selected to be a starting point. This reduced greatly the number of runs to reach convergence.

Abbaszadeh and Medhat (1988) presented a general method for automatic well test interpretation. The method matches the field data with theoretical reservoir models using constrained, nonlinear, least-squares regression technique, coupled with superposition and numerical Laplace inversion of pressure-drawdown equations using Stehfest's inversion. The Levenberg-Marquardt algorithm was used. Bounding the parameters using physically meaningful bounds helped to converge faster. Pressure gradients were approximated by forward finite-difference scheme in time domain. The algorithm is not supposed to identify the model and

therefore the model must be known. Vertical well model with multiple reservoir model examples were used to estimate the following parameter: P_o, m', C_D, Time match, ω, λ. The result showed that the success of this method depends on:

a) Validity of selected model.

b) Initial estimate of the parameters.

c) The quality of test data.

Ohaeri (1991) developed a procedure which couples the source function approach with real space Laplace inversion algorithm to automatically fit the type-curve. The integral was evaluated by dividing the time into time steps, evaluating the pressure derivative and then using trapezoidal/Simpson rule to compute the pressure without skin or wellbore storage. Numerical Laplace inversion was then used to add skin and wellbore storage. No optimization algorithm was used, yet, the graphical approach gave better control. It gave very fast and accurate results for a wide range of wellbore and reservoir conditions.

Carvalho, Redner, Thompson, and Reynolds (1992) presented a methodology to do an automatic type-curve matching using robust procedure that minimize the non-uniqueness problem. They used a two-step regression procedure to enhance the probability of finding a unique match. The first step is to run a least squares regression. The second step is to remove the outliers using a robust algorithm followed by a least squares regression run. The typical procedure was as follows:

a) Perform the regression using initial guess for the parameters α^0 and obtain the set of parameters α^1.

b) Sort Least Square Error in ascending order and estimate preliminary standard deviation using

$$\sigma^0 = 1.4826\,(1 + \tfrac{5}{n-np})\sqrt{\underset{i}{med}\, r_i^2} \qquad (2.3)$$

where np represents the number of model parameters, and "med" denotes the median value of the residuals.

c) The standard deviation of the residuals is given by

$$\sigma^* = \sqrt{\frac{\sum_{i=1}^{n} \omega_i r_i^2}{(\sum_{i=1}^{n} \omega_i - np)}} \qquad (2.4)$$

where ω is determined using the following criteria

$$\omega_i = \begin{Bmatrix} 1 & if\ \left|\frac{r_i}{\sigma^0}\right| \le 3 \\ 0 & otherwise \end{Bmatrix} \qquad (2.5)$$

d) Outliers are detected and removed based on the following criterion. If $\left|\frac{r_i}{\sigma^0}\right| \le 3$, the data point is considered an outlier, and is removed from the set of observed data points.

e) Perform a final (re-weighted) LS regression (RLS) on the reduced set of data (i.e., $n - np$ values), using the parameters from step (a) as the initial estimate.

Statistical analysis showed 95% confidence on the estimated parameters.

Thompson and Temeng (1993) considered designing an algorithm to automatically fit type-curve of multi-rate, horizontal well pressure transient in rectangular reservoir. The authors computed the pressure derivative to be used in improving the computation of the next step in the Levenberg-Marquardt algorithm. The authors did not consider wellbore storage in the model. The method showed fast performance and can be used to add the first step in computing the

parameters. The authors also recommended using pressure, not pressure derivative because the derivative is always noisy. This solution did not eliminate the non-uniqueness of solution.

Buitrago and Gedller (1996) developed a multi-start type algorithm that combines stochastic exploration of the domain and heuristic calculation of a descent direction in order to avoid stopping the algorithm at local minimum. For selecting initial set of parameters, they run the model over large number of data points to get information about the objective function. They then selected an initial value that is close to the expected global optimum.

Sultan and Al-Kaabi (2002) used artificial neural network to automatically determine the proper horizontal well model, identify flow regimes and mark the position of identified flow regimes on the derivative plot. For each task, a set of models were generated and a separate neural network was trained to identify its assigned features. Eleven signatures were used to train the networks. Each network was trained on normalized data. This ability of identifying the flow regimes helped to identify some of the parameters.

Zakaria, Hafez, Ochi, Zaki, Loloi and Abu-Sayed (2011) applied genetic algorithm to optimize injection pressure fall-off type curve. They modeled both cases of finite and infinite fracture models. Genetic algorithm was able to automate the entire well test analysis process, eliminating the need to make an initial guess and replace it by a set of bounds assigned to the parameters.

CHAPTER 3

OPTIMIZATION ALGORITHMS

Optimization is a process of searching for values of the parameters that minimize (or sometime maximize) an *objective function*. For one variable, continuously differentiable, function $F(x)$, the minimum value can be determined at $x = x^*$, where $F'(x^*) = 0 \; and \; F''(x\hat{} *) > 0$. This condition is called *optimality condition*. If this condition is applied at any x*, then F(x*) is a *local minimum*. If f(x*) ≤ F(x) for all other x's, then F(x*) is global minimum. In practice, it is very hard to prove that F(x*) is *global minimum*. On other hand, f(x*) is at local maximum if $F'(x^*) = 0 \; and \; F''(x^*) < 0$ (Bartholomew, 1993).

The *bisection method* is one of the simplest, systematic, but less efficient ways of finding the minimum value in range a ≤ x ≤ b. It requires evaluating the function at many points between a and b to find the minimum value. This method is useful when the gradient of the objective function is undefined either because of problem discontinuity, problem changing over time or the derivative is very difficult to obtain (Magnus Erik, 1993). The *Newton method* uses the first derivatives in iterative manner to reach to the minimum. It uses the following formula for iteration until an acceptable value is achieved (Bartholomew, 1993):

$$x_{k+1} = x_k - \frac{F(x_k)}{F\prime(x_k)}. \qquad (3.1)$$

The *secant method*, on other hand considers an iterative method to find the local minimum, where F'(x) = 0, by using finite difference approximation to obtain the derivative (Bartholomew, 1993)

If the *multivariable* function's gradient is known, *steepest descent* methods can be used. It searches for the optimum solution by moving only in a direction where the function gradient is decreasing. *Newton and Newton-like* methods have an advantage of using the second derivative which can lead to better performance (Bartholomew, 1993).

The *Direct search* method is used when the function is multivariable and not differentiable. Some random points can be selected and using a statistical model, the likelihood of finding the minimum after a number of trials can be estimated. Univariate search performs a series of optimization iterations with respect to one variable. Once the minimum value is obtained, another series of the next variable optimization starts, and so on. In the iterations, bisection or linear search method can be used. The Nelder and Mead simplex method introduces a more efficient method by selecting initial points for each variable, and then updating the values using a heuristic way called *reflection* (Bartholomew, 1993).

For multivariable problems, which have more than two variables, global optimum cannot be guaranteed because the grid cannot be plotted. Moreover, there is no condition that can show the function is at global optimum where it is applied. Therefore usually the optimization result that is accepted is the result that has a high probability of being the global minimum. To reduce this problem, *multi-start* method has been developed. In this method, different starting points are used to begin the optimization iterations. This method covers a wider range of the solution space. However, it is very expensive because the many local minima it reaches may yield the same result. Moreover, it does not guarantee that one of the local optimum found is the global optimum. The performance can be enhanced by incorporating some statistics into the iterations move (Bartholomew, 1993).

3.1 Stochastic Optimization

Stochastic optimization algorithms (SOA) are derivative-free optimization algorithms that use random numbers to select the parameters values. Some of these algorithms are multi-start. These kinds of algorithms are easy to implement and do not need previous knowledge of the objective function. SOA treat the objective function as a black box that generates the *fitness* based on the input parameters. The basic idea behind SOA is that, they start with single or multiple initial guesses for the solution, called *agents*. Each agent is a collection of input parameters $[x_1, x_2, x_3, \dots, x_n]$. After evaluating the objective function value of each agent, the computed values are used to determine the next values of each agent for the next iteration. The *Genetic Algorithm* (GA) was inspired by Darwinian biological theory. In GA, initial individuals are selected randomly from the *population*. Depending on the fitness of the objective function, some members of the population survive. The other members, with less fitness, are reproduced. This alteration is called *mutation*. In this research, four algorithms were used, Differential Evolution, Particle Swarm Optimization, Local Unimodal Sampling and Many Optimization Liaisons (Magnus Erik, 1993).

3.1.1 Particle Swarm Optimization

This concept was first introduced in 1995 by Kennedy and Eberhart. Particle Swarm Optimization (PSO) was developed through simulation of a simplified social model of the movement of organisms such as in a bird flock or fish school. In this model, individual members takes advantage of the discoveries of the previous experience of all the members while searching for food, avoiding predator or optimizing environmental variable. This model was simplified and used to optimize problems (Kennedy and Eberhart, 1995). PSO has proven to be a powerful population-based stochastic algorithm. It has been applied in a variety of fields, research and application areas (Linah Mohamed and Vasily Demyanov, 2010).

When the algorithm is initiated, the agents, called particles, are selected randomly in the search space. Each agent is considered a candidate solution to the problem (Linah Mohamed and Vasily Demyanov, 2010). In addition to the objective function parameters, each particle maintains a history of its best achieved result and its velocity along each dimension after evaluating the objective function at each particle. The direction of the movement is determined by calculating the velocity. The velocity is computed by a formula which is a function of the particle's best discovered position, and the swarm's best discovered position (Magnus Erik Hvass Pedersen, 2011).

$$\vec{v} \leftarrow w\vec{v} + C_p r_p\left(\vec{p} - \overrightarrow{x_p}\right) + C_g r_g\left(\vec{g} - \overrightarrow{x_g}\right) \quad (3.2)$$

where w is inertia weight, $\overrightarrow{p_p}$ is the particle's best discovered position, $\vec{g}$ is swarm's best discovered position, C_p and C_g are behavioral parameters and r_p and r_g are random numbers generated from uniform distribution in [0,1] in each iteration. ω models the tendency of the

particle to keep moving in the same direction it has been moving. Selecting a large value for ω increases the diversity in the search space and facilitates exploring new regions, whereas small values of ω enhance local exploration. When$\omega \geq 1$, the particle velocity increases gradually, causing the swarm to diverge while selecting $\omega < 1$ causes the particle to slow down until the velocity reached 0. The variables $\mathrm{C_p}$ and $\mathrm{C_g}$ are behavioral parameters that are randomly selected. Their values are less than one and more than zero. They model the attraction to move toward the best achieved result by the particle itself and the best achieved result by the whole swarm. The next position of the practical is $\vec{x} \leftarrow \vec{x} + \vec{v}$. In each generation, the particle velocity and position is updated (Linah Mohamed and Vasily Demyanov, 2010).

Several attempts to find universal values for PSO parameters were done (Magnus Erik Hvass Pedersen, 2011 ; Onwunalu, 2010). Suggested values for the above parameters were determined after years of accumulation of experience as the following: $w = 0.729, C_p = \mathrm{C}_g = 1.49445$ and $w = 0.6, \mathrm{C}_p = \mathrm{C}_g = 1.7$ if the inertia is static (Magnus Erik, 1993; Linah Mohamed and Vasily Demyanov, 2010). In some other approaches, the initial weight is chosen to be changing dynamically. When it is set to decrease linearly, it starts from a larger value, usually 0.9, and decreases over time to a small value, usually 0.4 (Kennedy and Eberhart, 1995). This approach allows starting by exploring new region and slightly starts refining the solution. Kennedy and Elbers suggested $\mathrm{C}_p = \mathrm{C}_g = 2$ such that the product of these two by the random values have a mean of 1. The PSO algorithm works in the following way:

a) Initialize the population at random positions and velocities. Set the best particle fitness and best global fitness to be infinite positive number.
b) Evaluate the objective function for each particle.

c) For each particle, if current fitness < best particle fitness then best particle[i] = current particle and best particle fitness = current fitness.

d) For each particle, if current fitness < best global fitness, then best particle = current particle and best fitness = current fitness.

e) Change the velocity of each particle according to the following equation:

$$\vec{v} \leftarrow w\vec{v} + C_p r_p(\vec{p} - \vec{x}) + C_g r_g(\vec{g} - \vec{x})$$

f) Go to step 2 until the best global fitness ≤ tolerance value.

Two general neighborhood *topologies* can be used in PSO; global best and local best. In the global best topology, each particle is influenced by the performance of the best particle in the swarm. Therefore, each particle has access to all other particles. It is also called Star topology, as shown in **Figure 3.1 (A)**. In the local best topology, each particle is influenced and has access to only particles in the local neighborhood such as the wheel topology (as shown in **Figure 3.1(b)**) where all the information are communicated through one focal particle or in the ring topology (as shown in **Figure 3.1(c))**, where each particle has access to its two neighbor particles (Linah Mohamed and Vasily Demyanov, 2010).

In the application of this study, global best topology and static inertia weight were used.

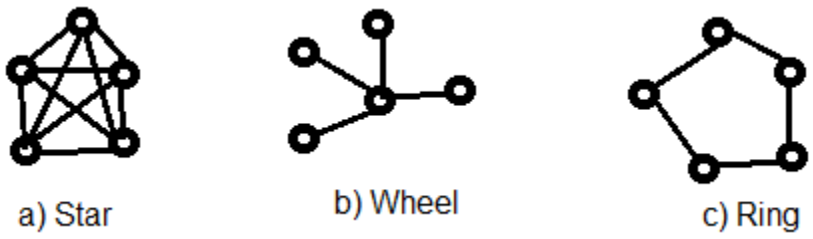


Figure 3.1 PSO Topology

3.1.2 Differential Evolution

Differential Evolution (DE) was developed by Storn and Price in 1996. DE is a stochastic search that works by creating a new potential agent-position by combining the positions of randomly chosen agents from its population, and updating the agent's current position in case of improvement to its fitness (Kennedy and Eberhart, 1995).It is proven to be efficient in searching for global optimization over continuous spaces. DE uses a population of agents, called *chromosome*, to search for the optimum solution. Better chromosomes are updated, by a process called *mutation*, after each generation while the weak chromosomes are removed. DE algorithm is influenced by three operators *mutation*, crossover and *selection*, in addition to three parameters which are population size Np, scale factor F, crossover probability CR (Youyunao *et al*., 2009). In its simplest form, DE algorithm is used to optimize an objective function of D-dimension in the following steps:

a) Initialization: an initial population is randomly created. They are assumed to be distributed uniformly. Each chromosome is a candidate solution. Chromosomes are vectors of the input parameters of the objective. The objective function is evaluated at each chromosome and this population is considered the current.

b) Selecting parents: for each chromosome, x_i, three different chromosomes are randomly selectedx_{r1}, x_{r2} and x_{r3}, such that$x_i \neq x_{r1} \neq x_{r2} \neq x_{r3}$.

c) Mutation: a random number R_{nd} is selected such that$R_{nd} \in [1, \dots, D]$. Create new trial[29,38]population X that consist of $x_{i,j} = 1, \dots, D$ according to the following equation:

$$x_{i,j}= \begin{cases} x_{i,r_3} + F \cdot (x_{i,r_1} - x_{i,r_2}) & \text{if} \quad (rand_{ij}[0,1) < Cr) \vee (Rnd = i) \\ x_{ij} & \text{otherwise} \end{cases}$$
$$i = 1, \ldots, D \qquad (3.3)$$

Where F is a behavioral parameter called the differential weight $\in [0,2]$, $R_{nd} \in \{1, \ldots, n\}$ is randomly chosen index, Cr $\in [0,1]$ is called crossover probability and $rand_{ij} \sim U(0,1)$is a uniform random number drawn for each use (Magnus Erik, 1993).

The fitness of each chromosome is calculated at each new chromosome in the new trial population.

d) The fitness of the trial population is compared with the current population. For each chromosome x_i in the trial population, if the fitness is better or equal to the corresponding chromosome in the current population, x_{ci}, then$x_{i=} x_{ci}$. The chromosome with the best fitness among the population is also selected to be thePob_{best}.

e) Steps 2 to 4 are repeated until the algorithm converges (Vitaliy Feoktistov, 2006).

Figure 3.2illustrate the behavior of DE algorithm.

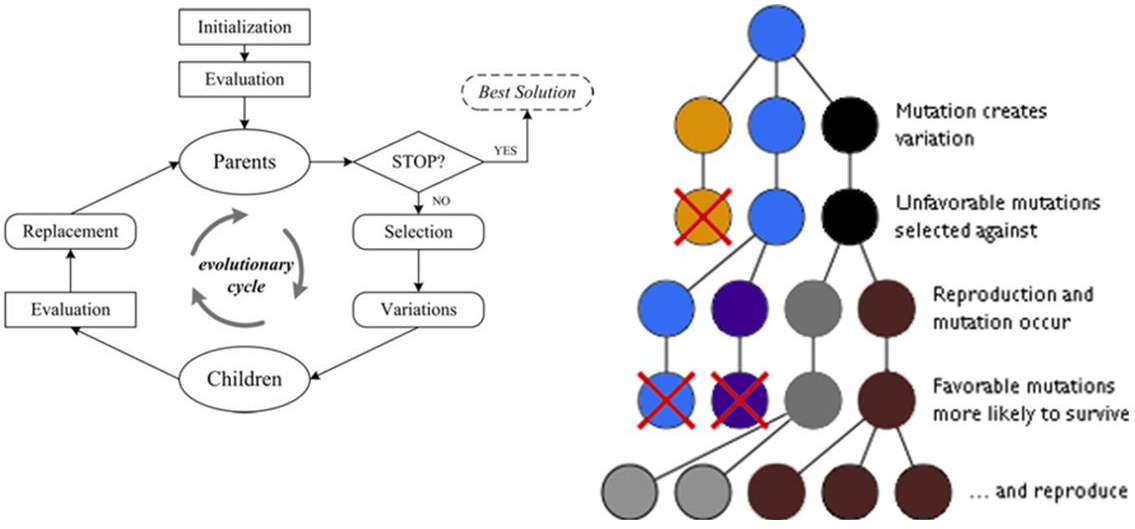


Figure 3.2 A basic scheme of evolutionary algorithms (Vitaliy Feoktistov, 2006)

Price and Storn (Vitaliy Feoktistov, 2006) suggested five strategies or schemes to improve the performance of DE algorithms which are:

- Scheme DE/rand/1 : Three random vectors are selected to construct the trial vector such that:

  $$w = x1 + F * (x2 - x3) \quad (3.4)$$

- Scheme DE/rand/2: Five random vectors are selected to construct the trial vector such that:

  $$w = x5 + F * (x1 + x2 - x3 - x4) \quad (3.5)$$

- Scheme DE/best/1: The best vector is used in addition to two random vectors, which are selected to construct the trial vector such that:

  $$w = xbest + F * (x1 - x2) \quad (3.6)$$

- Scheme DE/best/2 : The best vector is used in addition to four random vectors are, which selected to construct the trial vector such that :

  $$w = xbest + F * (x1 + x2 - x3 - x4) \quad (3.7)$$

- Scheme DE/rand-to best/1: The best vector is used in addition to three random vectors which are selected to construct the trial vector such that:

  $$w = x1 + F * (xbest - x1) + F * (x2 - x3) \quad (3.8)$$

Some studies have suggested the following values of the behavioral parameters: Np ≈ 40, F ≈ 0.6 and CR ≈ 0.9 (Magnus Erik Hvass Pedersen, 2011). Zaharie (2002) suggested that in order to find good values for F and Np, they have to satisfy the following equation:

$$(2F^2 - \frac{2}{NP} + \frac{Cr}{NP} = 0) \quad (3.9)$$

Values such as Np=50, Cr= 0.2 and F=0.1341 can perform well according to Zaharie, yet he did not mentioned the problem dimensions (Zaharie, 2002), whereas Hajizadeh et al.(2010) got the best performance of the DE when Np= 50, F=0.9 and Cr = 0.9.

In the context of this study, DE algorithm with DE/best/1 scheme will be used.

3.1.3 Local Unimodal Sampling (LUS)

Local Unimodal Sampling (LUS) algorithm is a modified version of the pattern search method that improves its performance by localizing the sampling around the best known solution, and decreasing the search range exponentially when samples fail to improve the fitness of its current position. The idea is when searching for minimum of objective function $f : R^n \rightarrow R$, to choose a random vector $\vec{y}$ from the neighborhood of the current position and move to the new position if the fitness is improved. For an agent which was initially placed in position $\vec{x}$ in n-dimension problem, the next computed position is$\vec{y} = \vec{x} + \vec{a}$, where$\vec{a}$ is chosen randomly and uniformly, $\vec{a} \sim U(-\vec{d}, \vec{d}).\vec{d} = \vec{d}_{up} - \vec{d}_{lo}$where $\vec{d}_{up}$is the upper boundary and $\vec{d}_{lo}$ is the lower boundary. Upon each failure of improving the fitness, $\vec{d}$ is updated $\vec{d} \leftarrow q.\vec{d}$, where q is the *decrease factor*.

$q = \sqrt[Vn]{1/2}$. V isa predefined value and *n* is the dimension of the objective function. Selecting $0 < \frac{1}{V} < 1$ yelid to a slower decrease in the search range whereas selecting a value of $\frac{1}{V} > 1$ cause mare rapid decrease (Magnus Erik Hvass Pedersen, 2011; Pedersen *et al*, 2009).

Methods such as Pattern Search, Luus-Jaakolaand the method by Schrackand Choit have also used exponential decrease of their search-range. The method by Fermi and Metropolis used it too

and succeeded with halving the search-range one dimension at a time. In LUS, halving the search-range when all *n* dimensions are sampled concurrently has to occur after *n* failures to improve the fitness, to yield a similar combined effect. The algorithm for the LUS optimization method is shown in **Figure 3.3** (Magnus Erik Hvass Pedersen, 2011; Pedersen *et al*, 2009)**.**

- Initialize $\vec{x}$ to a random solution in the search-space:

$$\vec{x} \sim U\left(\vec{b}_{lo}, \vec{b}_{up}\right)$$

  Where $\vec{b}_{lo}$ and $\vec{b}_{up}$ are the search-space boundaries.

- Set the initial sampling range $\vec{d}$ to cover the entire search-space:

$$\vec{d} \leftarrow \vec{b}_{up} - \vec{b}_{lo}$$

- Until a termination criterion is met, repeat the following:
  - Pick a random vector $\vec{a} \sim U\left(-\vec{d}, \vec{d}\right)$
  - Add this to the current solution $\vec{x}$, to create the new potential solution $\vec{y}$:

$$\vec{y} = \vec{x} + \vec{a}$$

  - If $(f(\vec{y}) < f(\vec{x}))$ then update the solution:

$$\vec{x} \leftarrow \vec{y}$$

    Otherwise decrease the search-range by multiplication with the factor $q = \sqrt[n]{1/2}$

$$\vec{d} \leftarrow q \cdot \vec{d}$$

3.3 LUS algorithms (Pedersen et al, 2009)

3.1.4 Many Optimization Liaisons (MOL)

Many Optimization Liaisons (MOL) is a modified version of PSO, which was suggested by Kennedy (1997). It eliminates the particle best known position and updates the particle positions using the best globally found position according to the following equation:

$$\vec{v} \leftarrow w\vec{v} + C_g r_g(\vec{g} - \vec{x}) \quad (3.12)$$

The next position of the practical is $\vec{x} \leftarrow \vec{x} + \vec{v}$. It was found to be effective in several problems (Kennedy, 1997). In some cases in the literature, MOL showed a slight advantage over PSO.

CHAPTER 4

TRANSIENT PRESSURE MODELS

4.1 Dual Porosity Model

4.1.1 Model Description

In the dual porosity model, a vertical well is fully penetrating a naturally fractured reservoir of thickness h and producing a single phase fluid. This model assumes that the reservoir is made up of rock matrix blocks which have high storativity but low permeability k_m and low storativity but high permeability k_f network of fractures and fissures. The well is connected to the reservoir through the fissures only, and therefore all produced fluid is coming from these fissures. The model is simplified by Warren and Root (1963) where they assumed that the reservoir is composed of a matrix of equal size blocks separated by fractures.

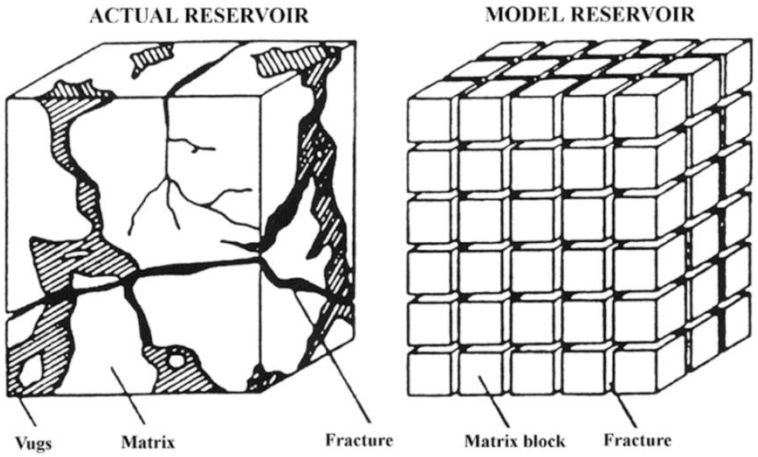
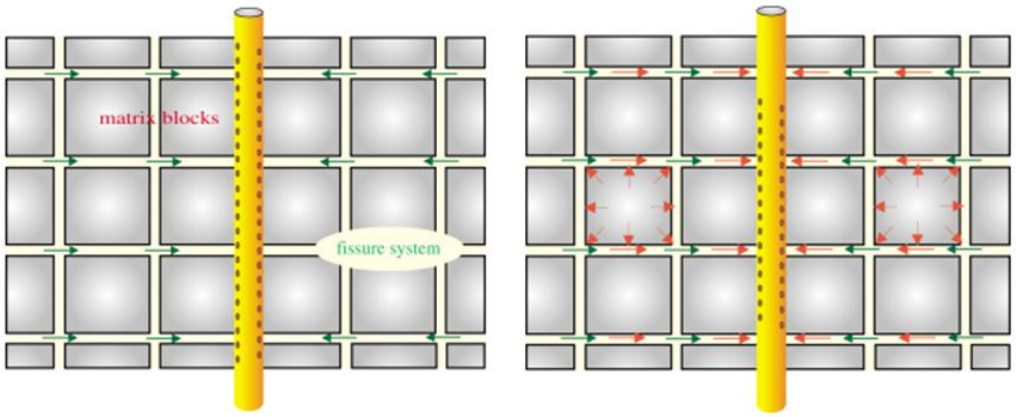


Figure 4.1 Warren and Root dual porosity reservoir model (Root et al. , 1963)

When a well is first put on production, after producing all the fluid stored in the wellbore, it starts producing the stored fluid in the fissures and develops a fissure radial flow regime



A. Fissure system production **B. Total system production**

Figure 4.2 Dual porosity flow systems (Houze, et al. 2011)

This flow regime is usually over very quickly, and is sometimes masked by wellbore storage. If not masked by wellbore storage, the fissure radial flow can be observed as a zero slope derivative right after the wellbore storage on log-log derivative plot. After sometime, the fissures start depleting, causing pressure differential between fissure and matrix blocks to provide pressure support. This pressure differential causes the matrix to start recharging the fissures with fluid. This transition occurs as a dip in the derivative. Double porosity pseudo-steady state assumes that the pressure distribution in the matrix is uniform. A schematic of dual porosity flow system is shown in **Figure 4.2**. When the pressure differential become significant, an equivalent infinite acting radial flow develops which can be seen as a zero slope derivative. This flow regime remains until the pressure transient wave reaches to the reservoir boundaries. **Figure 4.3** shows the wellbore storage, fissure radial, transition and total system radial flow regimes on a loglog plot.

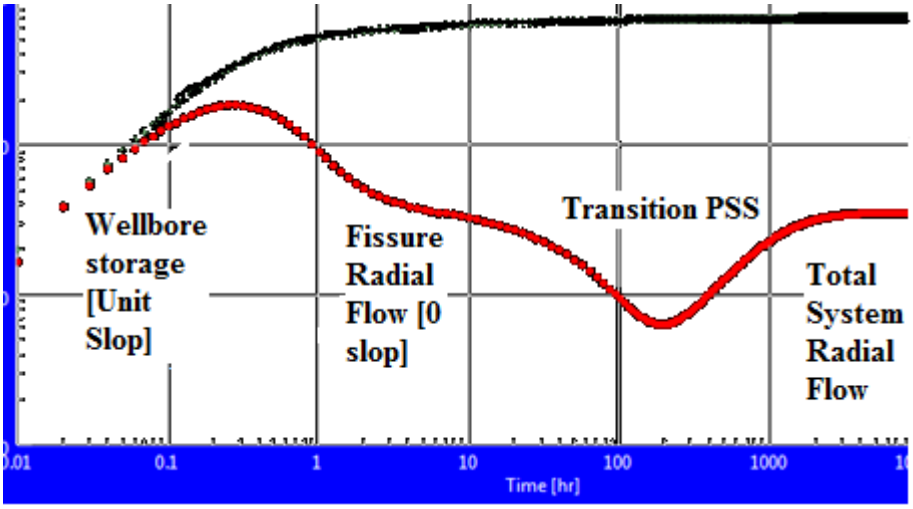


Figure 4.3 Dual Porosity flow regimes on loglog plot

4.1.2 Mathematical Formulation

The fundamental partial diffusivity equations for radial flow of single phase fluid in a, naturally-fractured, reservoir is given as (Giovanni Da Prat, 1981):

$$\frac{\partial^2 p_{Df}}{\partial r_D^2} + \frac{1}{r_D}\frac{\partial p_{Df}}{\partial r_D} = (1-\omega)\frac{\partial p_{Dm}}{\partial t_D} + \omega\frac{\partial p_{Df}}{\partial t_D} \tag{4.1}$$

$$(1-\omega)\frac{\partial p_{Dm}}{\partial t_D} = \lambda(p_{Df} - p_{Dm})$$

For the closed outer boundary, the condition is (Giovanni Da Prat, 1981):

$$\left.\frac{\partial p_{Df}}{\partial r_D}\right|_{r_D = r_{eD}} = 0 \tag{4.2}$$

The dimensionless flow rate into the wellbore is given by (Giovanni Da Prat, 1981):

$$q_D\,(t_D) = -\left(\frac{\partial p_D}{\partial r_D}\right)_{r_D=1} \tag{4.3}$$

The cumulative production is related to the flow rate by (Giovanni Da Prat, 1981):

$$Q_D = \int_0^{t_D} q_D dt_D \tag{4.4}$$

For a two-porosity system initially at constant pressure, the initial condition is given by (Giovanni Da Prat, 1981)

$$p_{Df}(r_D, 0) = 0 \tag{4.5}$$

The inner boundary condition in this case of a constant producing pressure is:

$$p_{Df} - s\left(\frac{\partial p_{Df}}{\partial t_D}\right)_{r_D=1} = 1 \tag{4.6}$$

This equation assumes pseudosteady state interporosity flow. The solution of this equation for a bounded (closed) circular reservoir, in Laplace space for dimensionless wellbore pressure, without wellbore storage and skin is (Sabet, 1991):

$$\overline{p}_{wD}(l) = \frac{K_1(r_{eD}\sqrt{lF})I_0(\sqrt{lF}) + I_1(r_{eD}\sqrt{lF})K_0(\sqrt{lF})}{l\sqrt{lF}\left[I_1(r_{eD}\sqrt{lF})K_1(\sqrt{lF}) - K_1(r_{eD}\sqrt{lF})I_1(\sqrt{lF})\right]} \tag{4.7}$$

where K & I are Bessel functions (Sabet, 1991).

$$F(l) = \frac{\omega(1-\omega)l + \lambda}{(1-\omega)l + \lambda} \tag{4.8}$$

ω is the storativity ratio = $\frac{(V\emptyset c_t)_f}{(V\emptyset c_t)_m + (V\emptyset c_t)_f}$ (4.9)

$\lambda$is the interporosity flow ratio = $\alpha r_w^2 \frac{k_m}{k_f}$ (4.10)

Accounting for wellbore storage and skin effects, the dimensionless wellbore pressure is (Sabet, 1991):

$$\overline{P}_{wD}(l) = \frac{[l\overline{p}_D + S]}{l\{1 + C_D l[l\overline{p}_D + S]\}} \tag{4.11}$$

where: $\bar{p}_D$ is the dimensionless wellbore pressure without wellbore storage and skin. $\bar{p}_D$ can be inverted from Laplace to real space using numerical inversion methods such as Stephest algorithm (Roumboutsos *et al*., 1988). The dimensionless pressure and time are defined in real space as

$$p_D = \frac{k_f h\,(\Delta p)}{141.3\; q\; \mu\; B} \tag{4.12}$$

$$t_D = \frac{0.000264\; k_f t}{\emptyset_t c_t \mu\; r_w^2} \tag{4.13}$$

C_D is defined as (Ezekwe, 2010):

$$C_D = \frac{0.894\; C}{(\emptyset_m c_m + \emptyset_f c_f) h\; r_w^2} \tag{4.14}$$

4.2 Horizontal Well Model

4.2.1 Model Description

In this model, a horizontal well of length L, in a box-shape, homogenous reservoir, with sealing boundaries, and of dimensions a in x-direction, b in y-direction and h in z-direction, is producing a single phase fluid from its horizontal section. The well is positioned parallel to the y-direction between the points (x_0, y_1, z_0) and (x_0, y_2, z_0), as shown in **Figure 4.4**.

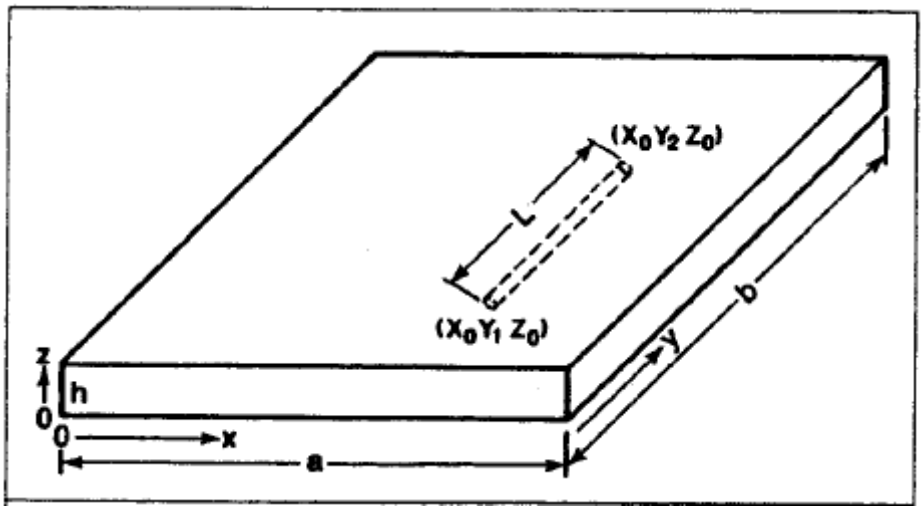


Figure 4.4 Schematic of horizontal well in a box-shaped reservoir (Issaka and Ambastha, 1992)

In this model, when the well is placed on production for the first time, it develops an *early time vertical radial flow regime* (ERF), **Figure 4.6 -1**, which is perpendicular to the horizontal section of the wellbore. This flow regime lasts until the pressure transient reaches the top and bottom boundaries. If not masked by wellbore storage, ERF can be identified by a zero slope derivative on the log-log plot. The pressure drop in ERF can be expressed as (Houze, et al. 2011):

$$\Delta p = \frac{162.6 q \mu B}{\sqrt{k_x k_z} L_e}\left[\log\left(\frac{\sqrt{k_x k_z}\, t}{\phi \mu c_t r_w^2}\right) - 3.227 + 0.868 S\right] \qquad (4.15)$$

If the well is placed closer to the top or bottom boundary, a *hemiradial* flow regime, **Figure 4.6 -2**, may develop, which can be identified by a zero slope, double derivative line. Subsequently, an *early linear flow* (ELF), **Figure 4.6-3**, may develop perpendicular to the horizontal direction of the well. On log-log plot, ELF is shown as a ½ slope derivative. Pressure drop at ELF can be calculated as (Houze, *et al*. 2011):

$$\Delta p = \frac{8.128qB}{L_e h}\sqrt{\frac{\mu t}{k_x \phi c_t}} + \frac{141.2q\mu B}{\sqrt{k_x k_z}L_e}\left(S_c + S_d\right) \tag{4.16}$$

If the test lasts long enough, a *late radial flow* (LRF), **Figure 4.6 -4**, regime may develop around the whole horizontal well, which has a signature of zero slope derivative on a log-log plot. The pressure drop can be expressed in LRF as (Houze, *et al*. 2011) :

$$\Delta p = \frac{162.6q\mu B}{\sqrt{k_x k_y}h}\left[\log\left(\frac{\sqrt{k_x k_y}\,t}{\phi\mu c_t L_e^2}\right) - 2.303\right] + \frac{141.2q\mu B}{\sqrt{k_y k_z}L_e}\left(S_c + S_d\right) \tag{4.17}$$

If the reservoir dimensions are not equal, i.e $a \neq b$, *late linear flow regime* may develop when the transient reaches the closest boundaries. Finally, when all boundaries are reached, pseudosteady state flow regime (PSS) starts, where the reservoir starts depleting. PSS can be identified by a unit slope derivative on a log-log plot.

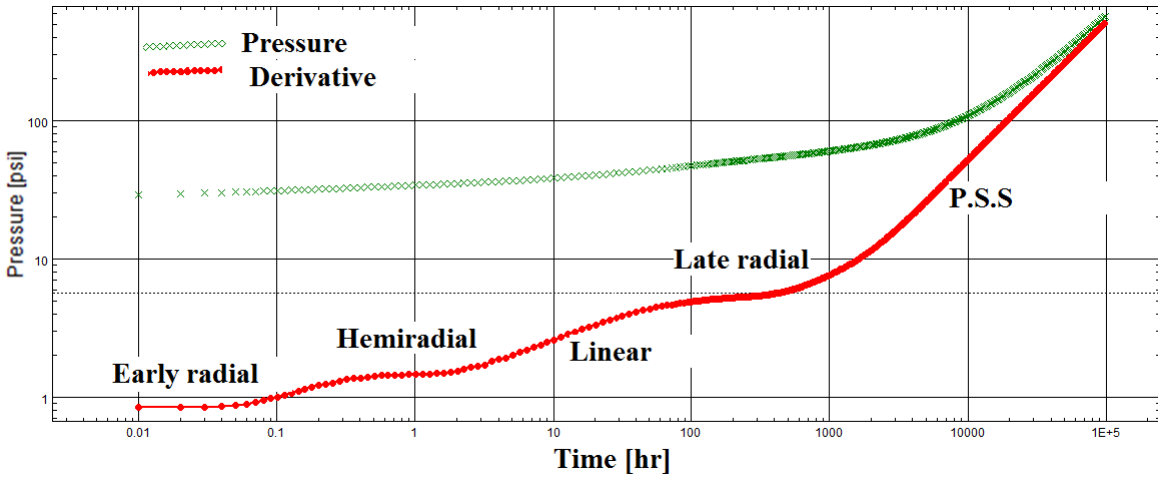


Figure 4.5 Horizontal well flow regimes on loglog plot

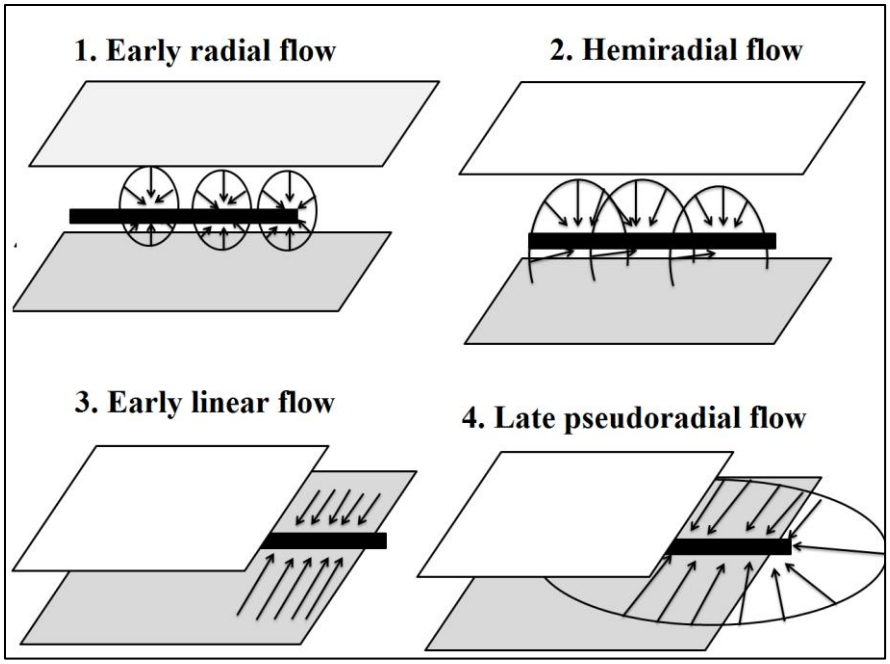


Figure 4.6 Schematic of horizontal well flow regimes

4.2.2 Mathematical Formulation

The diffusivity equation for flow in a horizontal well flow in a closed rectangular (box-shaped) reservoir is given as (Issaka, 1992):

$$k_x \frac{\partial^2 p}{\partial x^2} + k_y \frac{\partial^2 p}{\partial y^2} + k_z \frac{\partial^2 p}{\partial z^2} = \Phi \mu c_t \frac{\partial p}{\partial t} \quad (4.18)$$

where the initial condition:

$$p(x, y, z, t = 0) = p_i \quad (4.19)$$

The outer boundary condition for closed boundaries is:

$$\left(\frac{\partial p}{\partial x}\right) = \left(\frac{\partial p}{\partial y}\right) = \left(\frac{\partial p}{\partial z}\right) = 0 \quad (4.20)$$

And the inner boundary condition for uniform flux is:

$$\left(\frac{\partial p}{\partial x}\right)_{r=r_w} = \left(\frac{\partial p}{\partial y}\right)_{r=r_w} = \left(\frac{\partial p}{\partial z}\right)_{r=r_w} = Constant \tag{4.21}$$

The general solution can be obtained by analytical integration of the appropriate Green's functions. Given that the well is parallel to the y axis, i.e $x = x_0, z = z_0, y_1 \leq y \leq y_2$, the well is represented as line sink parallel to the y-axis. By defining dimensionless variables as following (Issaka and Ambastha, 1992):

$$t_D = \frac{k_x t}{\alpha L^2} \tag{4.22}$$

$$\alpha = 3790.85\ \Phi \mu c_t \tag{4.23}$$

$$p_D = \frac{0.00708 h k_x \Delta p}{B \mu q} \tag{4.24}$$

$$x_D = \frac{x}{L} \tag{4.25}$$

$$y_D = \frac{y}{L}\sqrt{\frac{k_x}{k_y}} \tag{4.26}$$

$$z_D = \frac{z}{L}\sqrt{\frac{k_x}{k_z}} \tag{4.27}$$

$$L_D = y_{2_D} - y_{1_D} \tag{4.28}$$

The diffusivity equation (4.18) becomes:

$$\frac{\partial^2 p_D}{\partial {x_D}^2} + \frac{\partial^2 p_D}{\partial {y_D}^2} + \frac{\partial^2 p_D}{\partial {z_D}^2} = \frac{\partial p_D}{\partial t_D} \tag{4.29}$$

The solution to this diffusivity equation is:

$$p_D = \frac{2\pi}{a_D b_D} \int_0^{t_D} \int_{y1_D}^{y2_D} \left(s_{1_D} . s_{2_D} . s_{3_D}\right)\, dy_{0_D}\, dt_D \tag{4.30}$$

Pressure derivative can be obtained by differentiating equation 4.30 with respect to time and therefore can be expressed as:

$$\frac{dp_D}{dt_D} = \int_{y1_D}^{y2_D} \left(s_{1_D}.s_{2_D}.s_{3_D}\right) dy_{0_D} \quad (4.31)$$

where p_D is the dimensionless pressure without skin and wellbore storage, $s_{1_D}.s_{2_D}$and s_{3_D} are the instantaneous point sink functions (Green's Function) located at x_0, y_0andz_0.

$$s_{1D} = 1 + 2\sum_{n=1}^{\infty} cos\frac{n\pi x_D}{a_D}\cos\frac{n\pi x_{oD}}{a_D}\ exp\left[-\frac{n^2\pi^2}{a_D^2}t_D\right] \quad (4.32)$$

$$s_{2D} = 1 + 2\sum_{m=1}^{\infty} cos\frac{m\pi y_D}{b_D}\cos\frac{m\pi y_{oD}}{b_D}\ exp\left[-\frac{m^2\pi^2}{b_D^2}t_D\right] \quad (4.33)$$

$$s_{3D} = 1 + 2\sum_{l=1}^{\infty} cos\frac{l\pi z_D}{h_D}\cos\frac{l\pi z_{oD}}{h_D}\ exp\left[-\frac{l^2\pi^2}{h_D^2}t_D\right] \quad (4.34)$$

Where $L = (y_2 - y_1) =$ lenght of the well in feet. To compute the pressure at the wellbore:

$$x = x_o + r_w\, cos\,\theta \quad (4.35)$$

$$z = z_o + r_w\, sin\,\theta \quad (4.36)$$

$$y = y_o\,; y_1 \leq y_o \leq y_2$$

The presented solution so far does not account for wellbore storage and skin. In order to add wellbore storage and skin (Ohaeri, 1991) used a numerical Laplace transformation to overcome the difficulty of doing direct conversion. Dimensionless pressure and time were transformed into Laplace space. The conversion was done as follows:

For the obtained values of dimensionless time and pressure:

$$\begin{matrix} T_{D1} & p(t_{D1}) & p_{D1} \\ T_{D2} & p(t_{D2}) & p_{D2} \\ T_{D3} & p(t_{D3}) & p_{D3} \\ & \vdots & \\ T_{Dn} & p(t_{Dn}) & p_{Dn} \end{matrix}$$

A straight line segment can be used to approximate p_{Di} in the interval $T_{Di-l} - T_{Di}$ such that:

$$p_{Di}(t) = p_{Di-l} + p_{Di-l}.(t - T_{Di-l}) \text{ where } T_{Di-l} \leq t \leq T_{Di} \tag{4.37}$$

$$\dot{p}_{Di-1} = \frac{p_{Di} - p_{Di-1}}{T_{Di} - T_{Di}} \tag{4.38}$$

Then $p(t)_{Di}$ can be expressed as: $\begin{Bmatrix} p_{D1}(t) & 0 \leq t < T_{Di} \\ p_{D2}(t) & T_{D2} \leq t < T_{D2} \\ p_{DN}(t) & T_{DN-1} \leq t < \infty \end{Bmatrix}$

Then, $\bar{p}(l)_D$ can be expressed as

$$\bar{p}_D(l) = \frac{\dot{p}_{D0}}{l^2}(1 - e^{-lT_{D1}}) + \sum_{i=1}^{N-2} \frac{\dot{p}_{Di}}{l^2}(e^{-lT_{Di}} - e^{-lT_{Di+1}}) + \frac{\dot{P}_{D0}}{l^2} e^{-lT_{DN-1}} \tag{4.39}$$

where $\bar{p}_D(l)$ is the dimensionless pressure in Laplace space. This inversion may encounter instability at the very early time and at the end of the period. Therefore it is recommended to add one log cycle before and after the period of interest before doing numerical Laplace transformation. These log cycle can be removed after inverting the solution back to the real space. Skin and wellbore storage can be added easily in Laplace space as following:

$$\overline{p}_{wD}(l) = \frac{\left[l\overline{p}_D + S\right]}{l\left\{1 + C_D l\left[l\overline{p}_D + S\right]\right\}} \tag{4.40}$$

The pressure can then be inversed into real space using numerical inversion methods such as the Stephest algorithm (Roumboutsos *et al*., 1988).

CHAPTER 5

EXPERIMENTAL WORK & FINDINGS

A computer code was written in c#.Net programming language that generates the pressure response model described in Chapter 3. All the models are single phase flow models. The code implementation is shown in appendix A.

5.1 Adding Noise

To simulate real data, synthetic models are generated. Then, Gaussian white noise data is added to the pressure data by using a normally distributed random number generator. For pressure drop Δp, the noise is added such that

$$\Delta p = random.Gauss(\Delta p, \varepsilon) \quad (5.1)$$

where ε is the deviation and ΔP is the average of the noisy data. **Figures 5.1** and **5.2** show a dual porosity log-log plot before and after adding the noise. A value of $\varepsilon = 0.3$ was used because it gave moderate and reasonable noise.

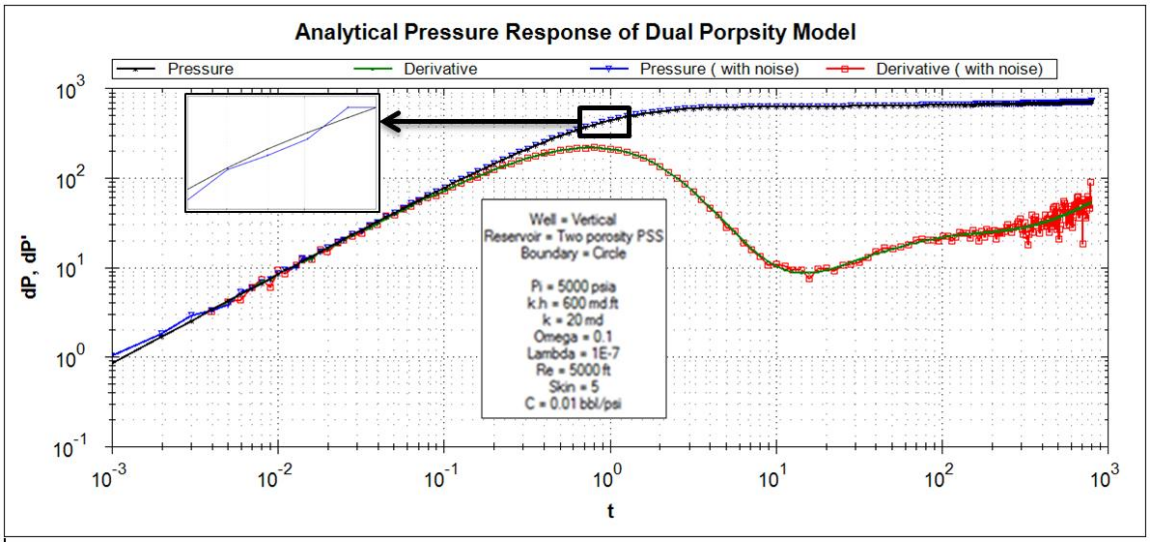


Figure 5.1 A dual porosity model before and after adding noise

5.2 Finding Optimum Behavioral Parameters

Before starting to evaluate the algorithms performance, initial runs were performed. The objectives of these runs are:

1- To find good behavioral parameters values for each algorithm.

2- To determine the behavior of each algorithm against the number of iterations.

Parameters determination was done in two steps:

1- Comparing the algorithm performance using recommended behavioral parameters from the literature. The same population size or agents is used for each behavioral parameters combination. Each algorithm is run for 25 times. The best, 7th, 13th, 19th and 25th best perform run.

2- After finding the best behavioral parameters, the same algorithm is run with these parameters but using different population size (population size). The best performing population size is used.

5.2.1 Dual Porosity model

The dual porosity model used in this test encounters all the flow regimes including wellbore storage, fissure radial, transition and equivalent infinite acting radial flow. Reservoir parameters used are as follows:

$c_f = 3E-6\,, c_m = 1.5E-5,\ skin = 5,\ C_D = 0.1,\ S_o = 1,\ \lambda = 1\mathrm{E}-6,\ \omega = 0.03,\ r_w = 0.25\ ft\,,\ r_e = 2000\ ft,\ \mu_o = 1,\ B_o = 1.25,\ k_f = 350,\ \emptyset_f = 0.3,\ \emptyset_m = 0.2,\ q_o = 800\ bbl, h = 400ft$. After generating the model, noise was added to the obtained pressure response using a standard deviation of 0.3. **Figure 5.2** shows the model before and after adding the noise.

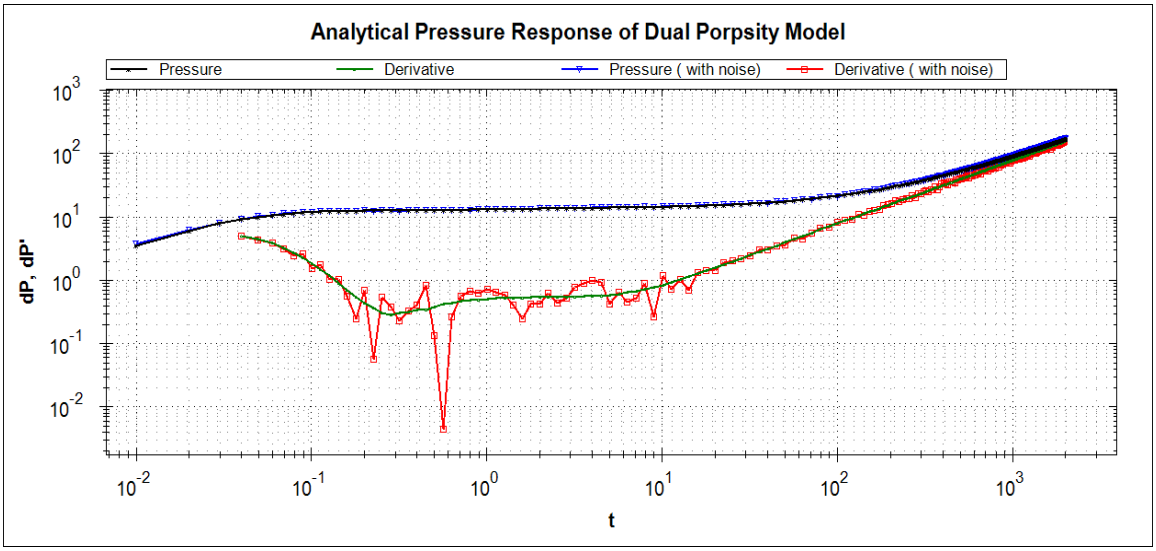


Figure 5.2 Dual Porosity model used in the test before and after adding noise

5.2.1.1 PSO

At the beginning, the PSO algorithm was run against 3 models using combinations of random values for the parameters. The objective of this step was to find out if the algorithm is following

any trend such that the algorithm performance is better in areas where the behavioral parameters have certain values. The maximum number of function evaluations was set to 300. **Figure 5.3** shows all the runs colored by the best achieved fitness. Green color is best fitness. No specific trend was identified. Another approach to allocate the best behavioral parameters was followed by selecting the best optimization run and comparing it with the algorithm performance when using recommended behavioral parameters in literature.

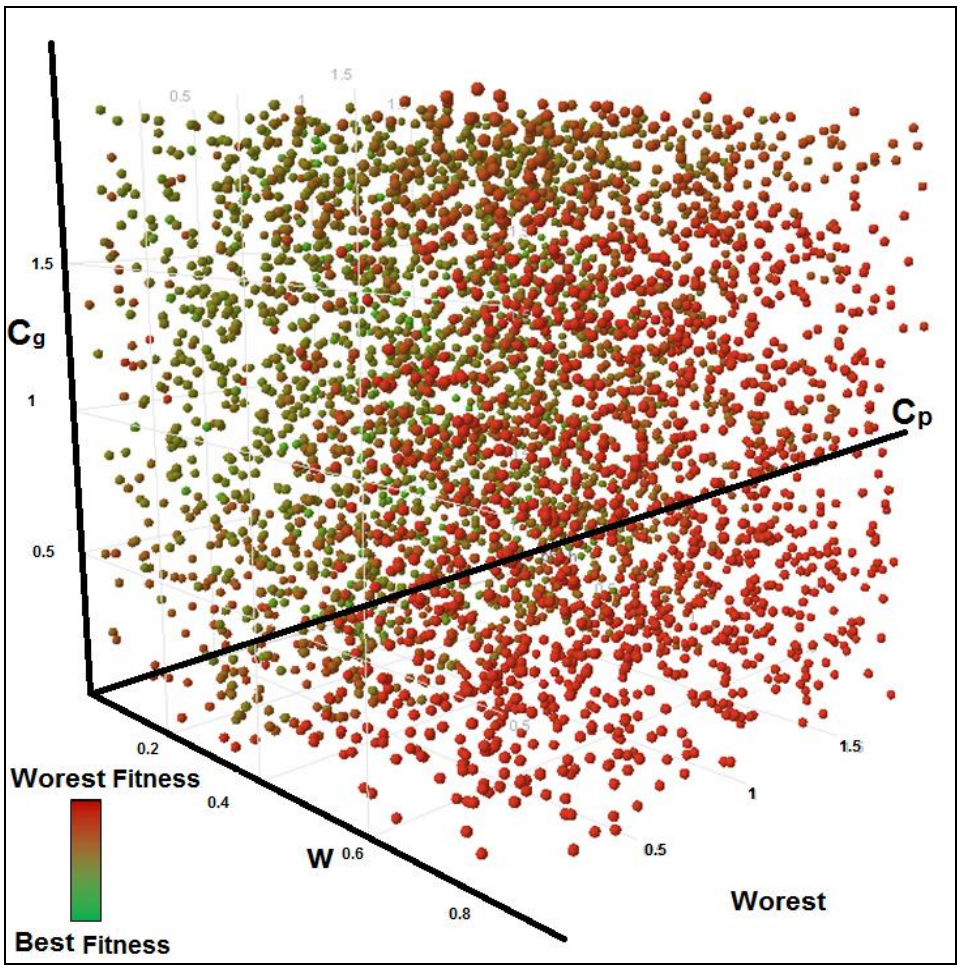


Figure 5.3 All PSO test runs result

Subsequently, the Best 16 runs were selected to be analyzed as shown in **Figure 5.3**.

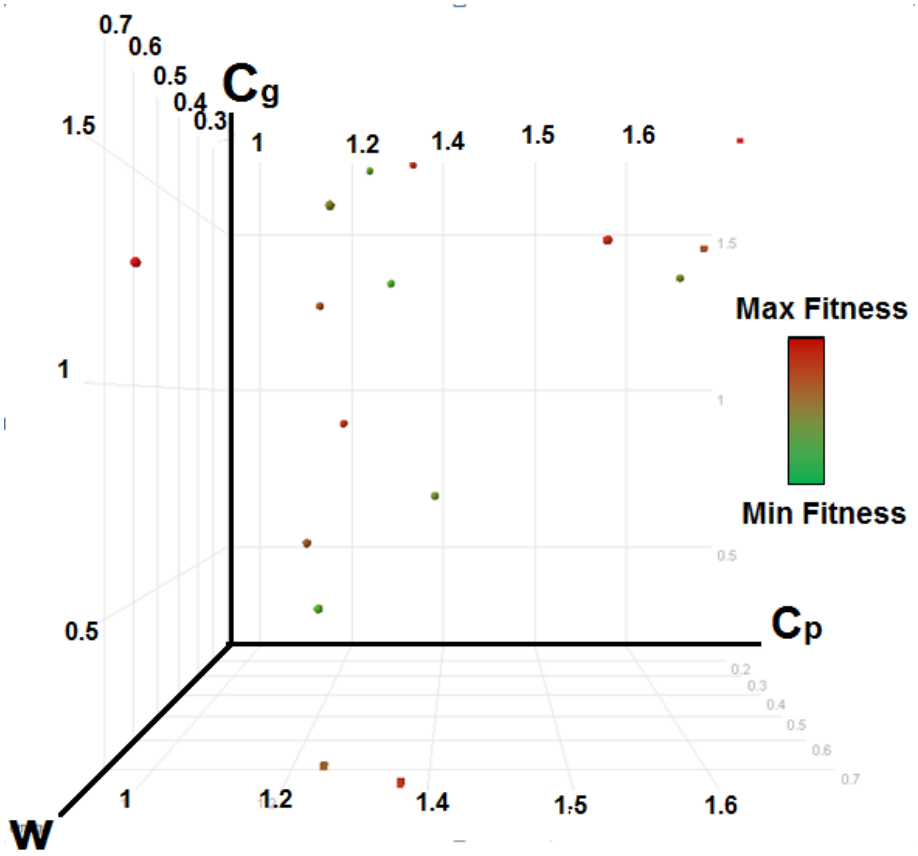


Figure 5.4 Best 16 PSO runs parameters

The test showed that at 300 function evaluations $w = 0.28$, $C_p = 1.31$ and $C_g = 1.14$ are the best performing combination. To validate this result, performance of the algorithm using the estimated parameters was compared with the performance of the algorithm using the recommended behavioral parameters from the literature.

In order to find good behavioral parameters of the PSO algorithm, the recommended values in the literature were compared. First, different values for w, C_p and C_g were used with the same population size $NP = 60$. The compared values are shown in **Table 5-1**. The total number of iterations per run was chosen to be 600.

Table 5-1 PSO algorithm recommended behavioral parameters

	w	C_p	C_g
Setting 1	0.28	1.31	1.14
Setting 2	0.729	1.49	1.49
Setting 3	0.7	1	1
Setting 4	0.6	1.7	1.7

With each set of behavioral parameters, the algorithm was used to perform an optimization for the dual porosity model. The model with noisy data was used for all the behavioral parameters. The result is then compared as shown in **Figure 5.5**.

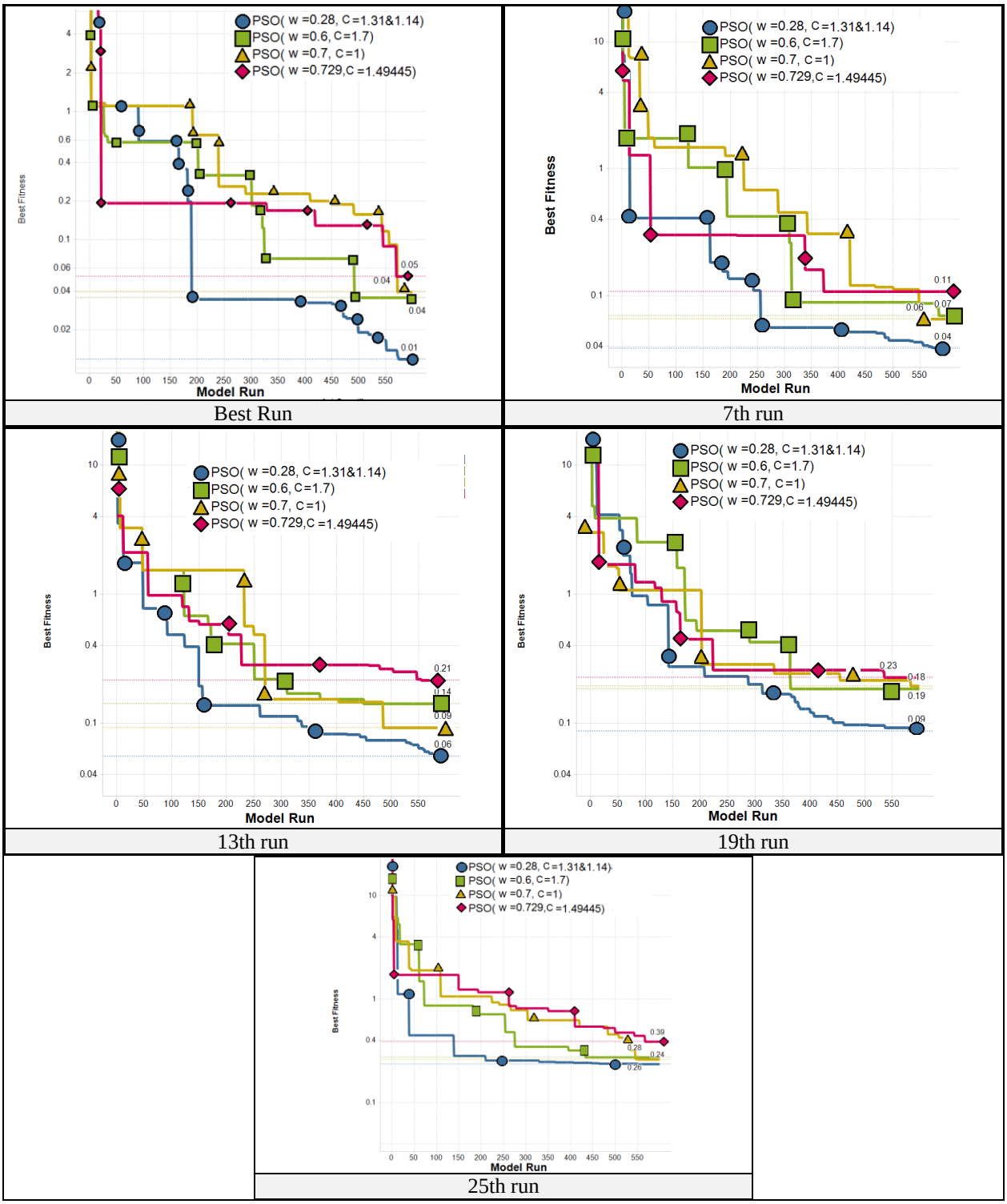
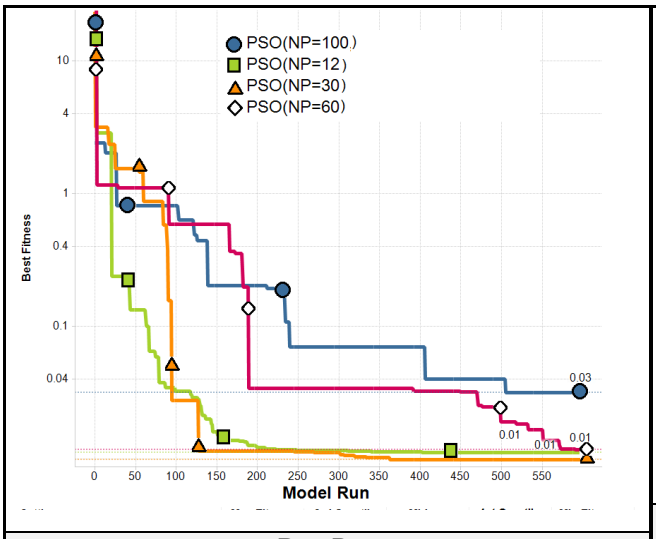


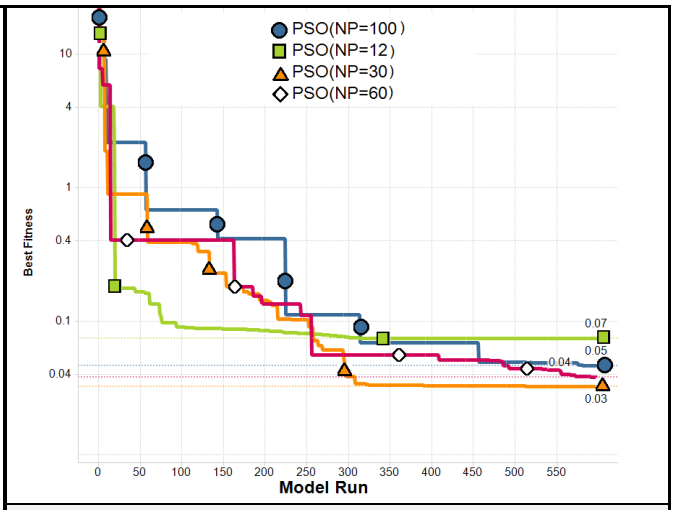
Figure 5.5 PSO performance using different behavioral parameters and 60 particles

The performance of the founded behavioral parameters in this experiment is better than the behavioral parameters recommended in the literature. The best performance was found at $w =$

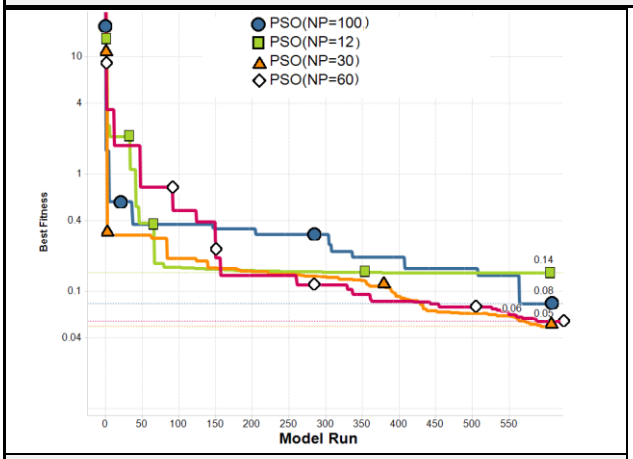
0.28, $C_p = 1.31$, and$C_g = 1.14$, which were found in the random runs test. The performance of the PSO algorithm was also compared base on the population size. Different population sizes were selected for comparison. The sizes were 100, 60, 30 and 12. **Figure 5.6** shows a comparison between the four population sizes.



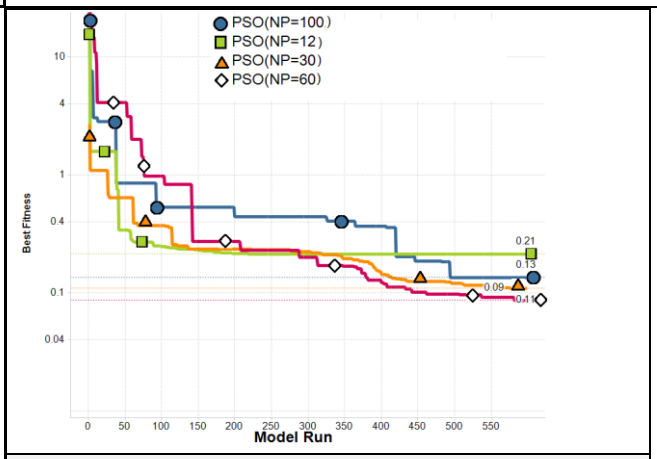
Best Run



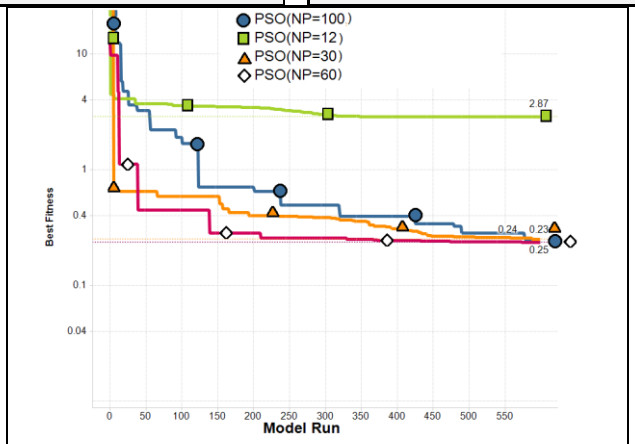
7th run



13th run



19th run



25th run

Figure 5.6 PSO Performance using different population size

Figure 5.6 shows that the population size of 30 performed the best. The algorithm performance, when using 60 particles, also is good and the progress continues as the number of function evaluations increases. Reducing the size of the population causes the algorithm to stop improving after about 300 iterations.

A final comparison was performed between dynamic w and static w strategies. The value of w was allowed to decrease linearly from 0.7 to 0.28. **Figure 5.7** shows the performance of both static and dynamic PSO. In the case of the dual porosity model, linear decrease of w worsens the PSO performance. The algorithm stopped improving after around 200 iterations.

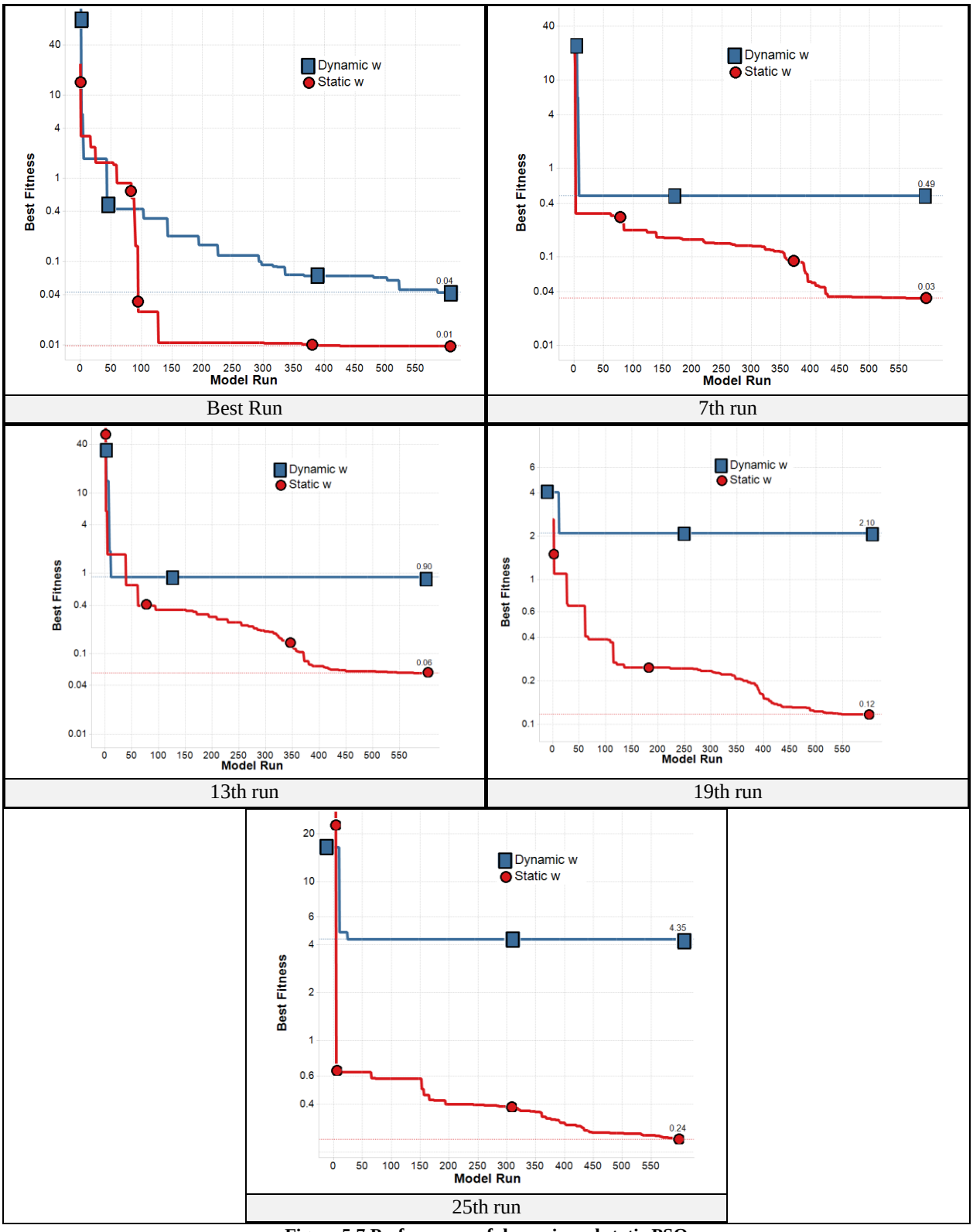


Figure 5.7 Performance of dynamic and static PSO

5.2.1.2 DE

The performances of 16 set of DE configurations were compared with the recommended behavioral parameters, which is$Cr = 0.9$ and $F = 0.9$. Thirty agents were used to perform 600 function evaluations in 25 realizations. Series of random behavioral parameters selection was generated to be compared. Sixteen combinations of Cr and F were used. Then, the best performing combination was compared and the best three performing combinations were compared in detail with the recommended behavioral parameters as shown in **Figure 5.8**. The performance is compared using the best and median of the average best achieved fitness.

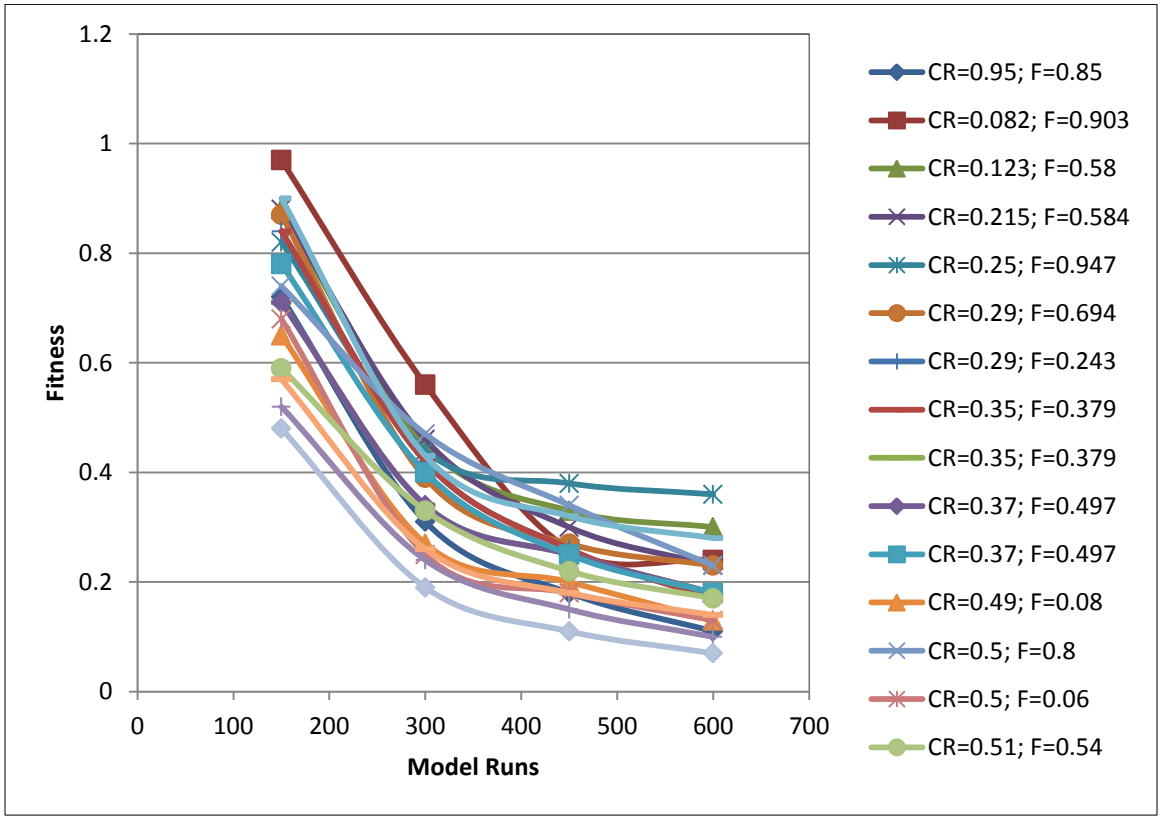


Figure 5.8 DE performance using 30 agents and different behavioral parameters

Table 5-2 DE performance summery using 30 agents and different behavioral parameters

CR	F	3rd Quartile	Median	1st Quartile	Best Fitness
0.95	0.85	0.72	0.31	0.18	0.11
0.082	0.903	0.97	0.56	0.26	0.24
0.123	0.58	0.88	0.45	0.33	0.3
0.215	0.584	0.88	0.46	0.3	0.23
0.25	0.947	0.82	0.44	0.38	0.36
0.29	0.694	0.87	0.39	0.27	0.23
0.29	0.243	0.84	0.42	0.26	0.17
0.35	0.379	0.71	0.34	0.25	0.18
0.37	0.497	0.78	0.4	0.25	0.18
0.49	0.08	0.65	0.27	0.2	0.13
0.5	0.8	0.74	0.47	0.34	0.23
0.5	0.06	0.68	0.25	0.18	0.13
0.51	0.54	0.59	0.33	0.22	0.17
0.56	0.18	0.52	0.24	0.15	0.1
0.69	0.03	0.9	0.43	0.32	0.28
0.86	0.19	0.57	0.26	0.18	0.14
0.99	0.45	0.48	0.19	0.11	0.07

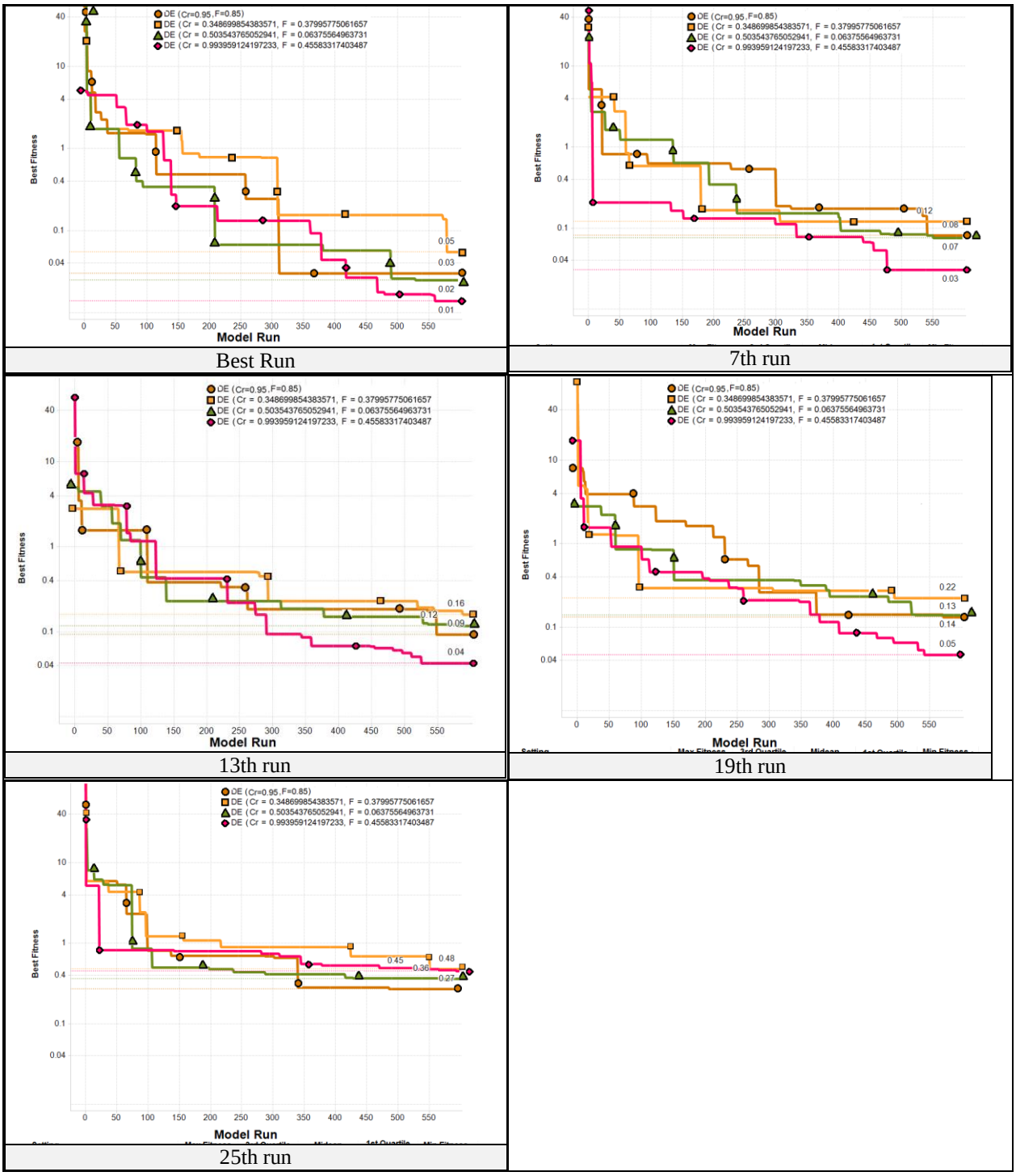


Figure 5.9 DE performance summery using 30 agents and suggested behavioral parameters

The test shows that the performance can be better by using $Cr = 0.99$ and $F = 0.45$. Like the PSO, another test was conducted to select the population size. Four population sizes; 100, 60, 30 and 12 were used to study the effect of population size on the performance of DE.

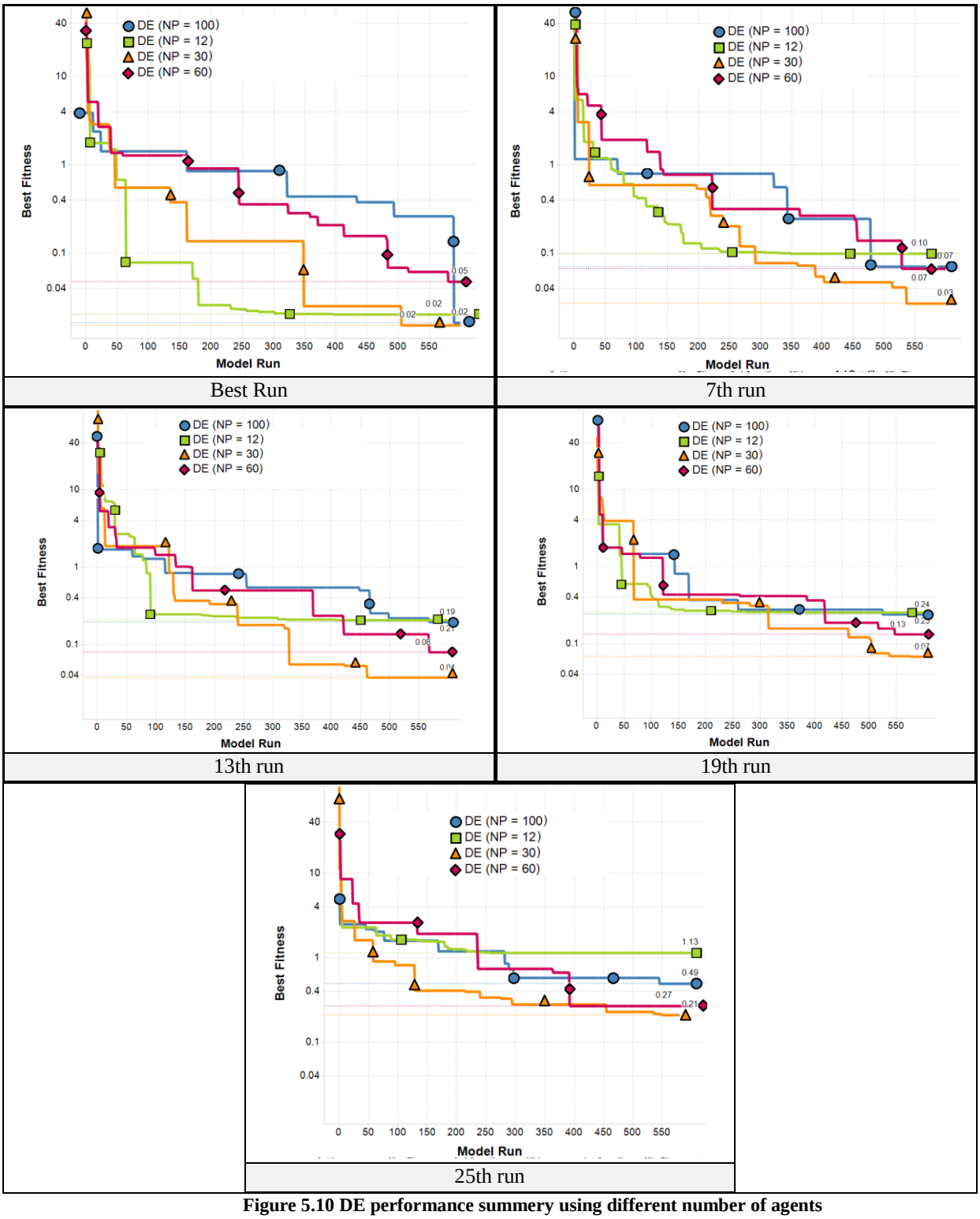


Figure 5.10 DE performance summery using different number of agents

The best performance was achieved when $NP = 30$, $Cr = 0.99$ and $F = 0.45$. Reducing the population size shows an early improvement in the best realization runs. After the early improvement, the algorithm stops improving which indicate that small population size is suitable for optimizing expensive function in which a lot of resources are needed to run the model. If the function is inexpensive, a moderately larger population size can be used. Given that only 600 function evaluations is allowed, 30 particles were enough to discover reasonable area of the solution space and to do evolve and improve the solution.

5.2.1.3 LUS

Initial runs showed that LUS does not progress significantly after the first 45-50 iteration. To fully utilize all the iterations, a new realization run was initiated after each 50 function evaluations. By doing this, each run is treated as an agent in comparison to other algorithms. This increases the chance of finding the global minimum by covering a wider search space.

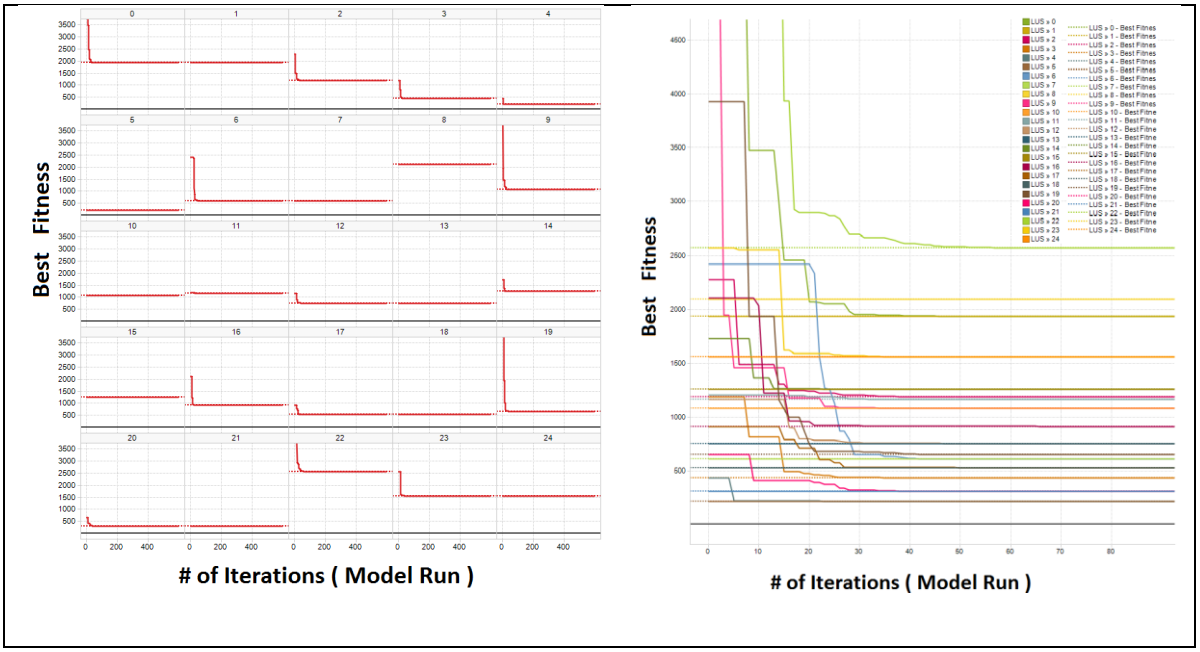


Figure 5.11 Typical behavior of LUS algorithm

A modification was done to the algorithm by re-initializing the algorithm after a predefined number of iterations. Each realization run was equivalent to one member of the population that is searching for a solution. Since each optimization run in this study is set to 600 model evaluation, each LUS agent can iterate for 600 divided by number of agents. **Figure 5.12** shows how this strategy improved the performance of the algorithm. **Figure 5.12** shows an optimization of the same model shown in **Figure 5.11**.

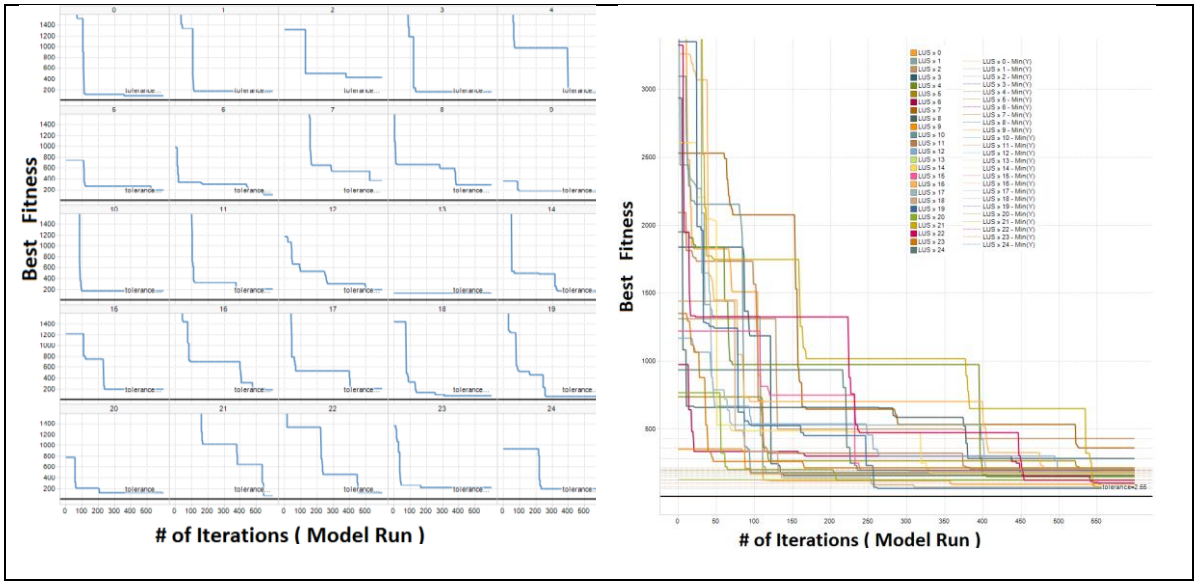


Figure 5.12 Modified version of LUS

To select the optimum configuration for the LUS algorithm, the two parameters that need to be searched are V and population size (Np). **Figure 5.13** shows a comparison of the following values of $V = 0.25,\ 0.33,\ 0.5$ and 0.66.

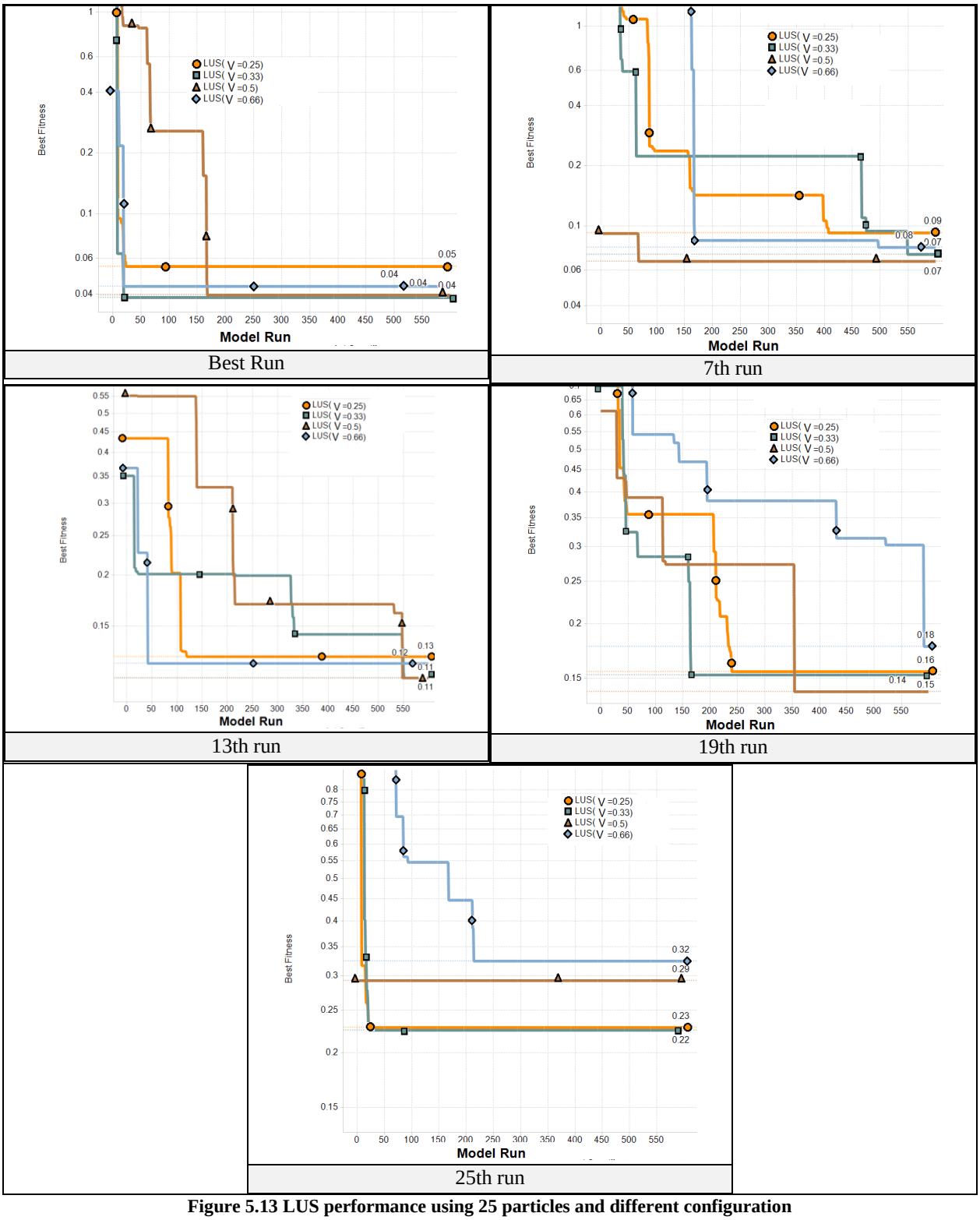


Figure 5.13 LUS performance using 25 particles and different configuration

The test shows that the LUS gave the best result when $V = 0.33$. While $V = 0.33$ showed a good performance in some cases, $V = 0.33$ also shows lower values for median and quartile.

Overall, the trend is the same for all where the algorithm start by significant improvement, and then the improvement became less significant.The next test was to select the best population size that lead to the best performance of the LUS algorithm.

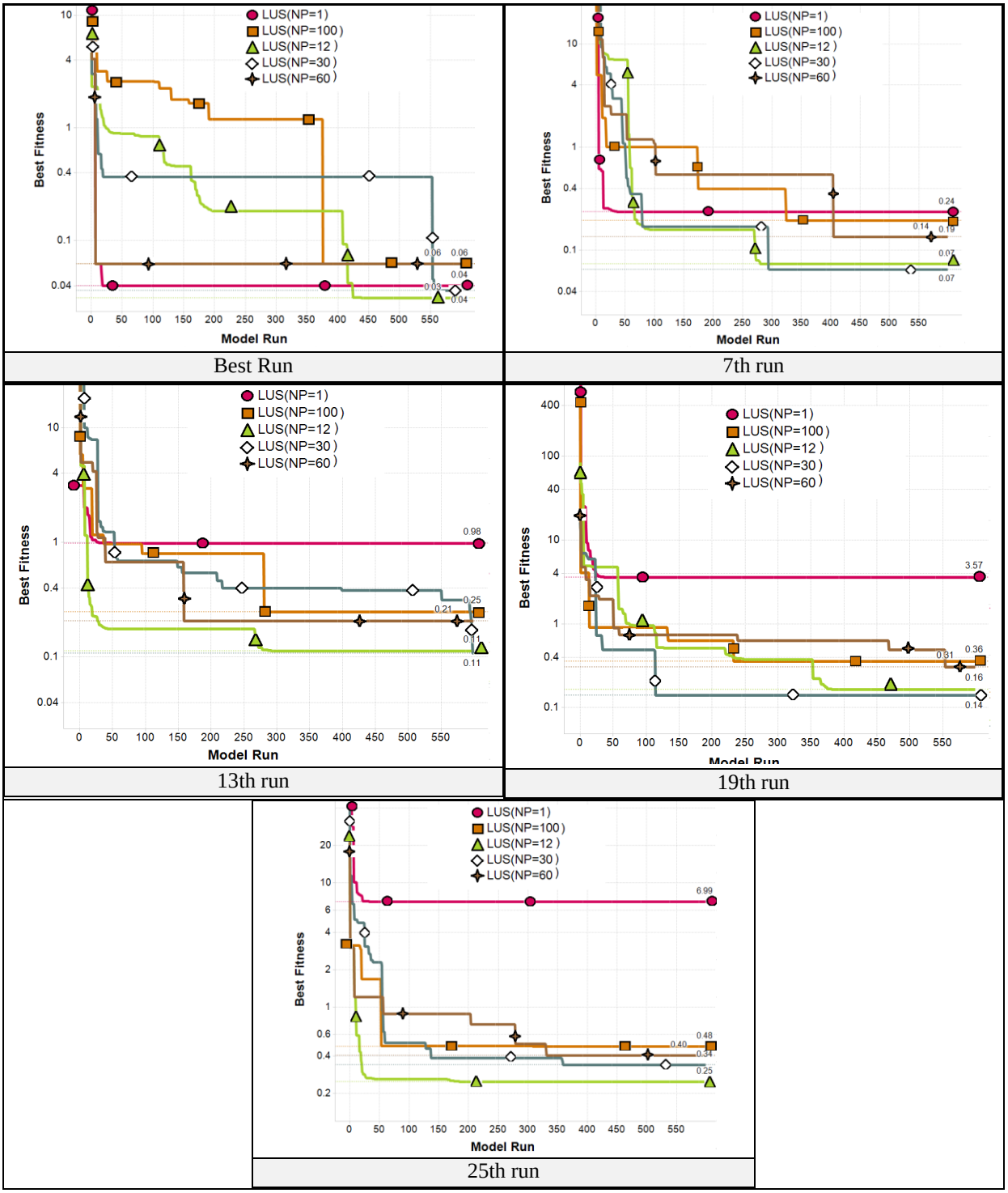


Figure 5.14 LUS performance using different population size

The performances of Np = 21 and Np = 30 are very close but the performance of Np = 12 particles is better when comparing second, median and first obtained fitness as shown in **Figure 5.14**. Therefore, using 12 particles can give a better result in less number of model evaluations.

5.2.1.4 MOL

Since MOL is a modified version of PSO, the same test that was conducted to select PSO best parameters was also conducted for the MOL using behavioral parameters shown in **Table 5.3**.

Table 5-3 MOL algorithm recommended behavioral parameters

	w	C_g
Setting 1	0.28	1.14
Setting 2	0.729	1.49445
Setting 3	0.7	1
Setting 4	0.6	1.7

As shown in **Figure 5.15**, the results are similar to those of PSO, where PSO with $w = 0.28$ and $C_g = 1.14$giving the best performance. The next test is to determine the best population size to be used in the optimization process.

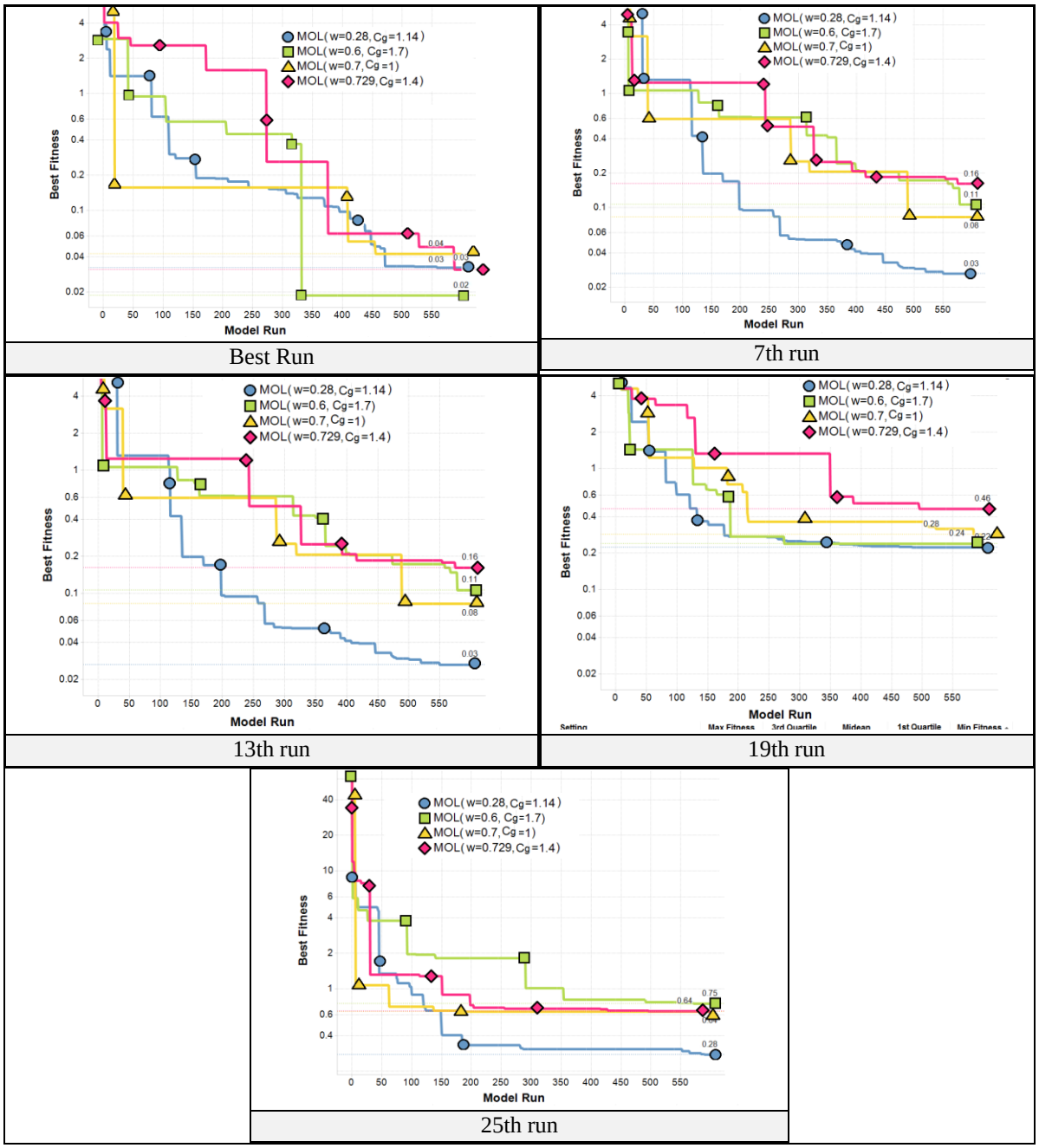


Figure 5.15 MOL performance using different behavioral parameters and 60 particles

The test showed that 30 particles performed slightly better than 60 particles in the MOL algorithm as shown in **Figure 5.16**. This is expected as MOL is a modified version of PSO and most likely it will behave similar to it. In addition, the performance of MOL using one single

particle was evaluated. It showed how the algorithm loses it is efficiency and how bad the performance can be due to the easy falling in local minimum.

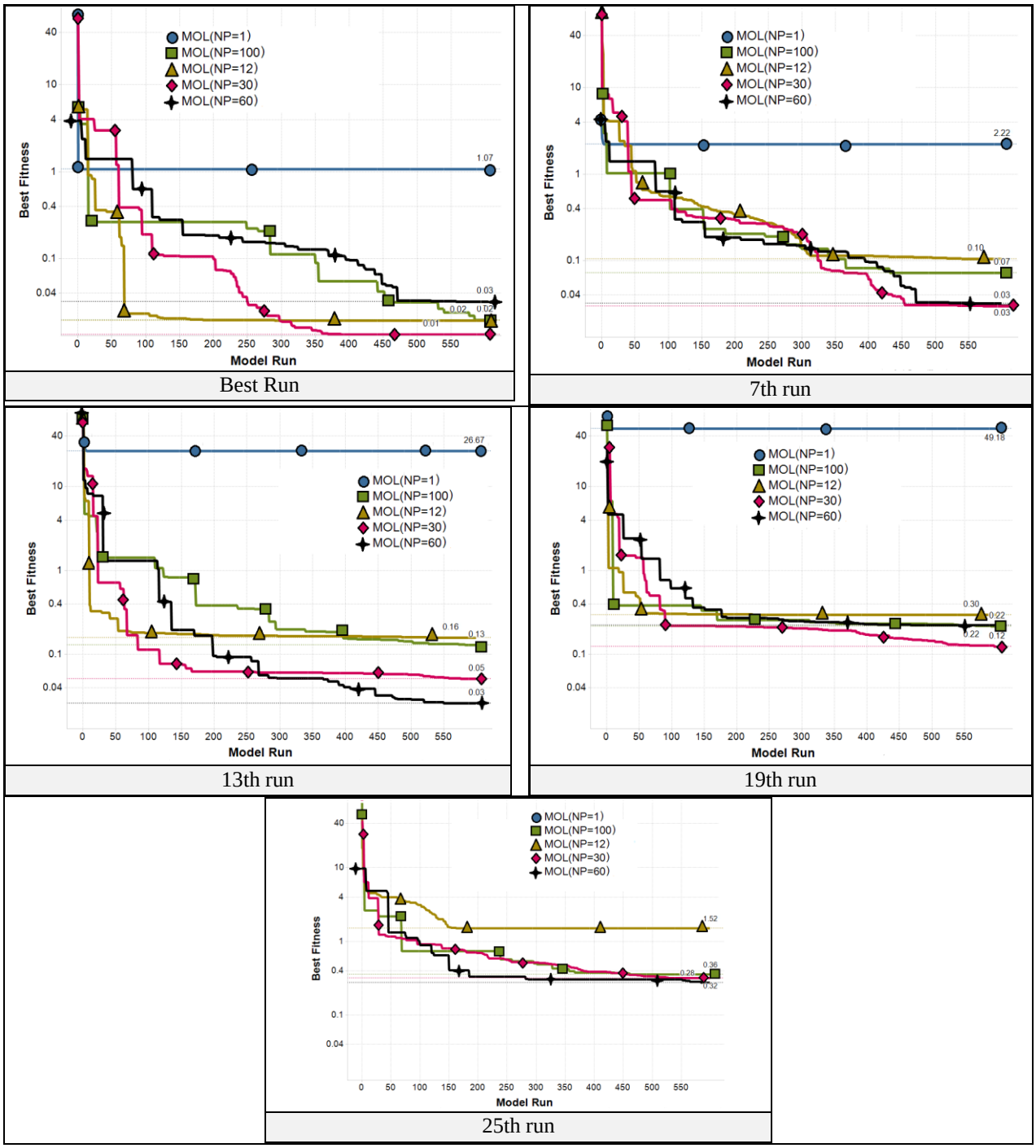


Figure 5.16 MOL Performance using different population size

5.2.2 Horizontal Well Model

Since each model has different geometry of the solution space, another study was conducted to find out whether the best performing behavioral parameters which were found in section 5.1 also apply to the horizontal well model. The same test was repeated, but using Horizontal well model.

The horizontal well model used in this test encounters all the flow regimes and has the following parameters:

$P_i = 5000\ psi$, $\emptyset = 0.25$, $L = 300\ ft$, $a = 1000\ ft$, $b = 7532\ ft$,$c_t = 3e-6$,$B_o = 1.25$,$\mu_o = 1$,$r_w = 0.25\ ft$,$q_o = 800\ bbl$,$x_0 = 500\ ft$,$y_0 = 5000\ ft$,$z_0 = 40\ ft$,$h = 200\ ft$,$k_x = 50\ md$,$k_y = 50\ md$,$k_z = 5\ md$,$skin = 2$,$C = 1e-3$. The unknowns to be optimized are:$k_x, k_y, k_z, skin, h$ and the length of the parallel boundary.

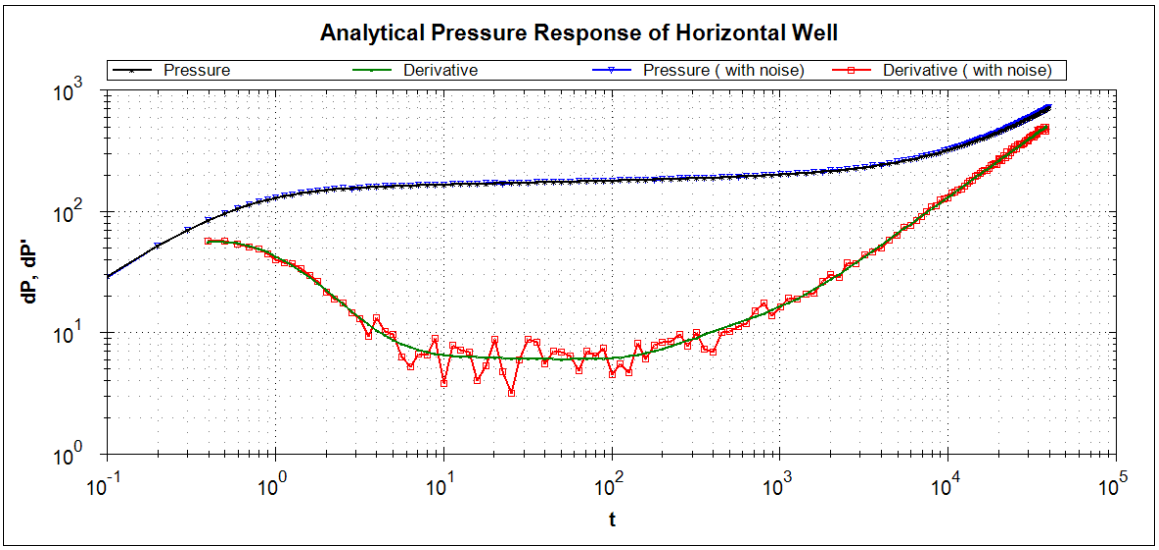


Figure 5.17 Horizontal well model used in the test before adding noise

5.2.2.1 PSO

The same behavioral parameters shown in **Table 5-1** were used with PSO algorithm to optimize horizontal well problem. The result is shown in **Figure 5.17**.

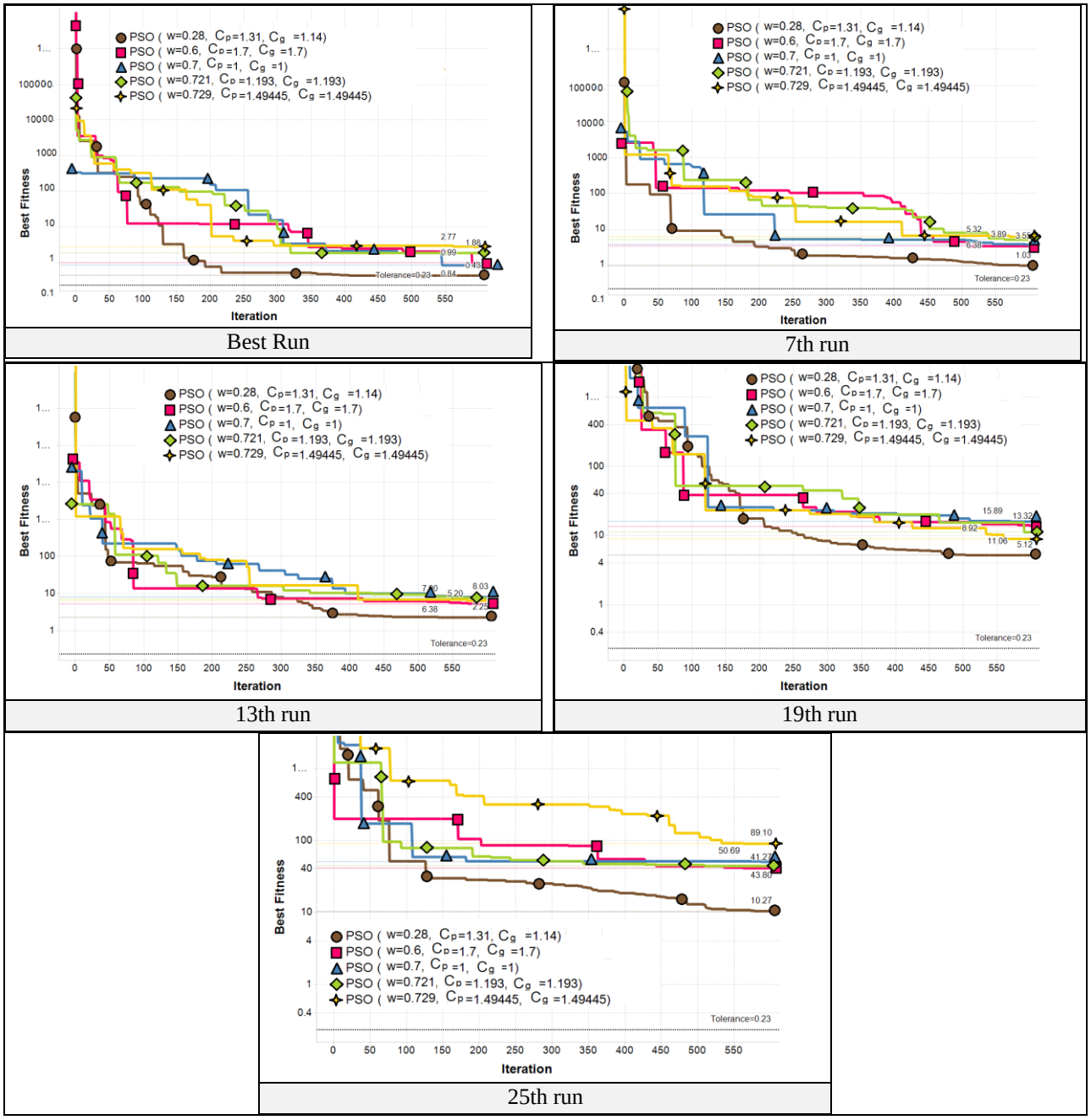


Figure 5.18 PSO performance using different behavioral parameters and 60 particles (Horizontal Well Model)

The test showed that the set $w = 0.28$, $C_p = 1.31$ and$C_g = 1.14$ performed the best. Again, different population sizes of 100, 60, 30 and 12 were used in order to determine the optimum population size. The test result is shown in **Figure 5.18**.

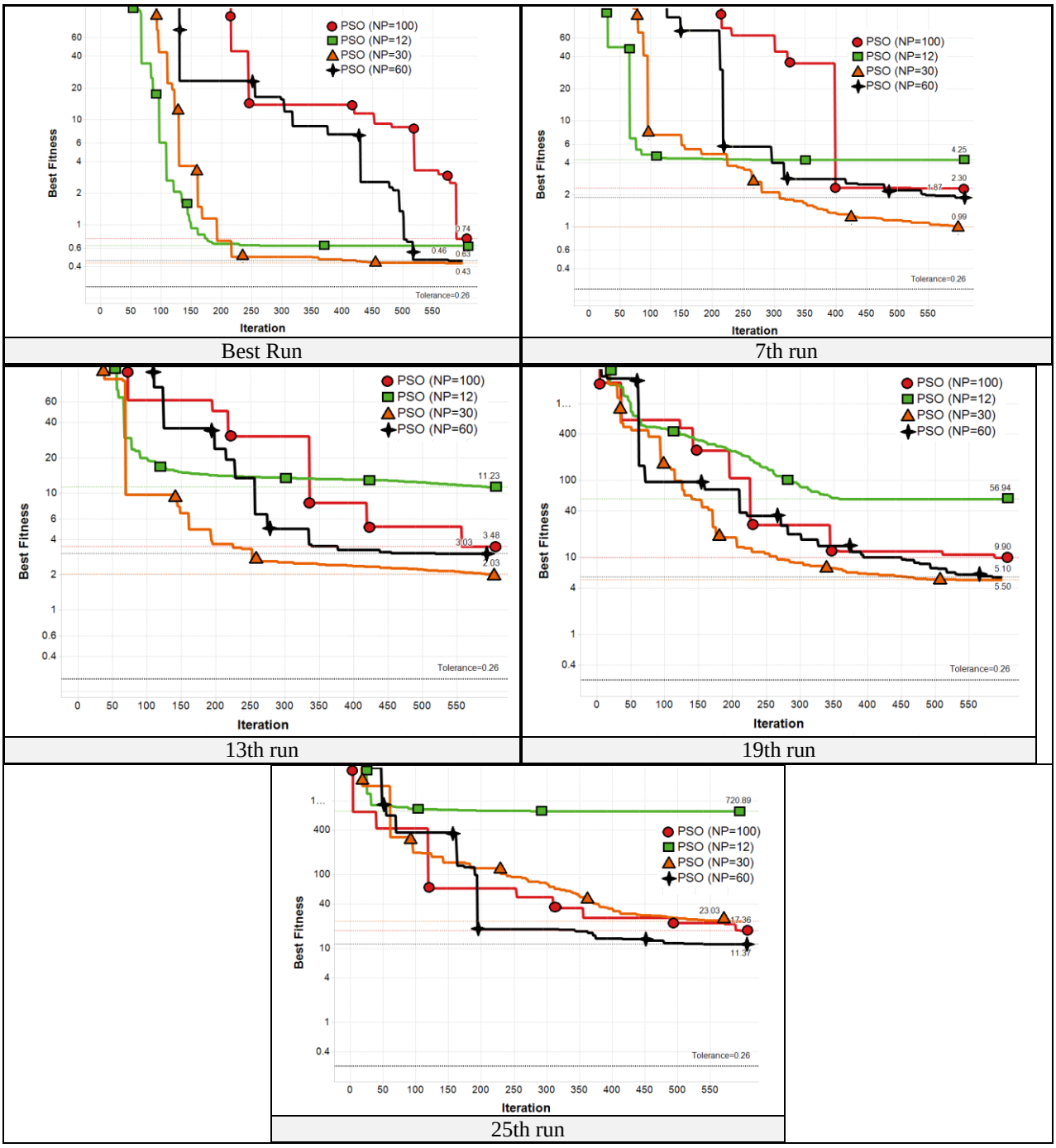


Figure 5.19 PSO performance using different population size (Horizontal Well Model)

As can be seen in **Figure 5.18** and **Figure 5.19**, the same PSO behavioral parameters and population size that gave the best performance for the dual porosity model also gave the best performance in horizontal well model.

5.2.2.2 DE

Figure 5.20 shows the same comparison that is conducted for DE algorithm to optimize dual porosity model but for horizontal well model. The same behavioral parameters that were used in dual model are used here.

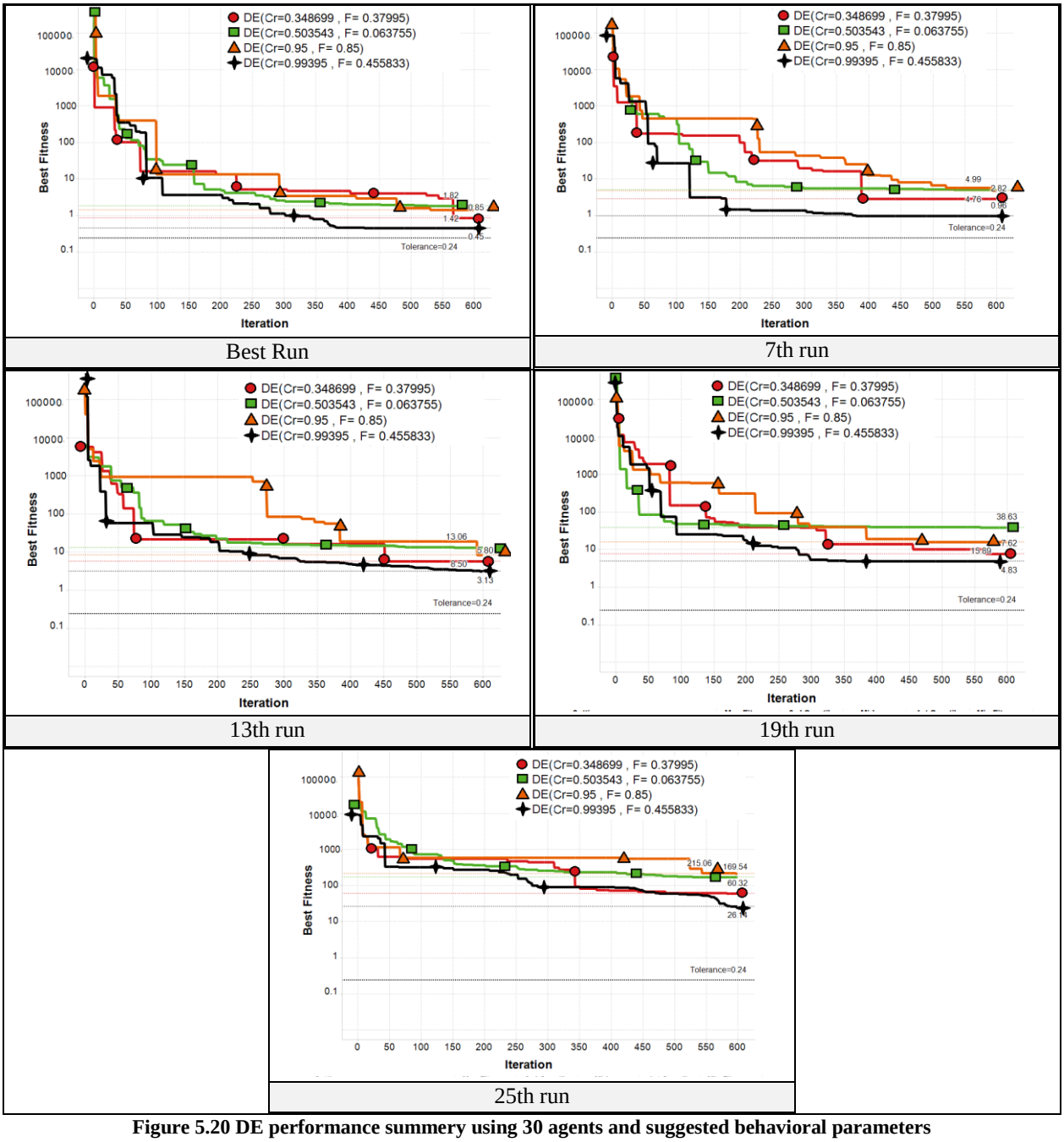


Figure 5.20 DE performance summery using 30 agents and suggested behavioral parameters

The result shows that the best performance can be achieved when $Np = 30, Cr = 0.99$and $F = 0.45$.

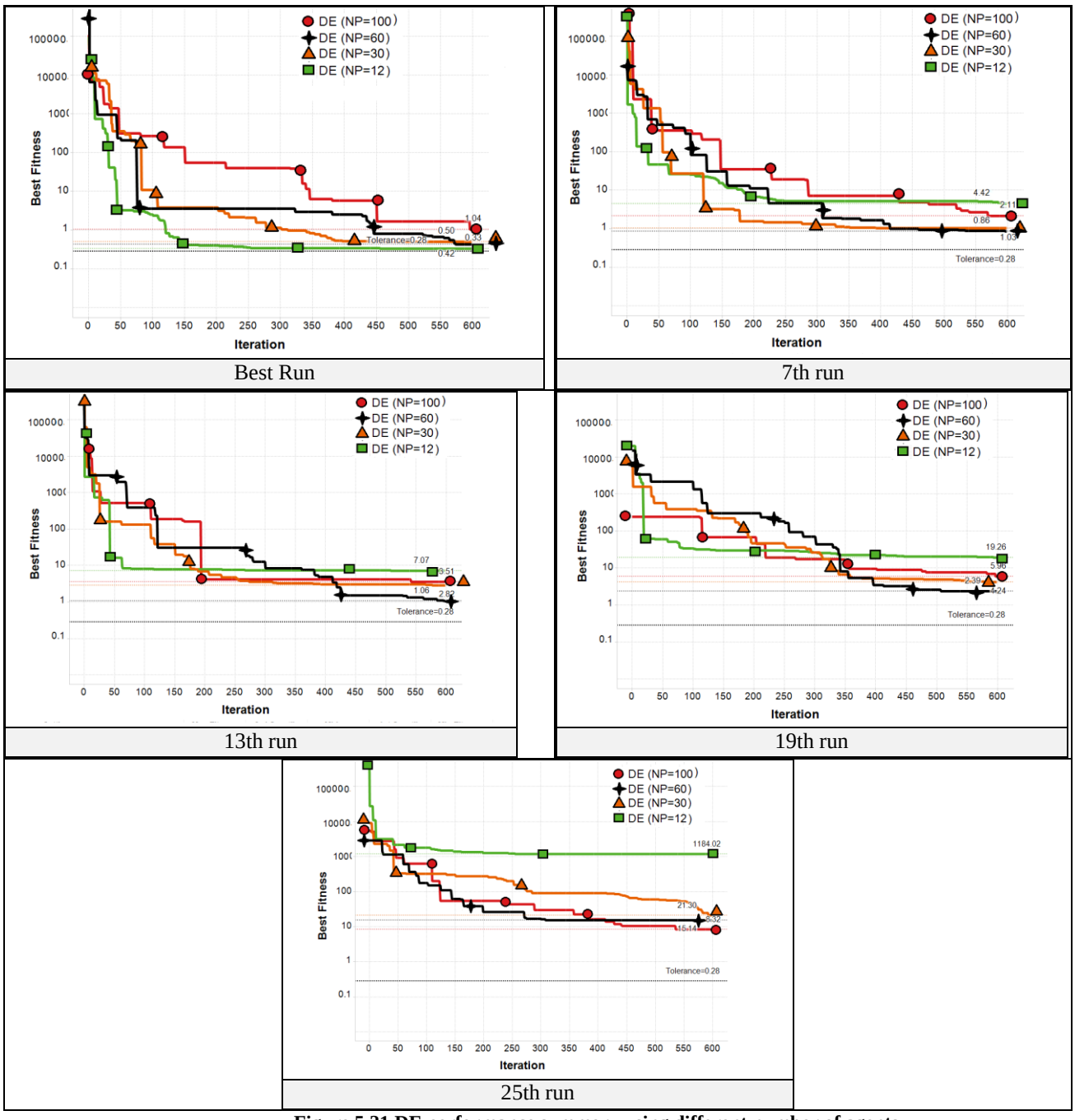


Figure 5.21 DE performance summery using different number of agents

Unlike in the Dual porosity model, DE gave the best performance when $Np = 60, Cr = 0.99$ and $F = 0.45$, as shown in **Figures5.20** and**5.21**. Also, when *Np= 30* is also acceptable.

5.2.2.3 LUS

The same procedure that was used to select the best behavioral parameters for LUS algorithm to optimize dual porosity model was also used in horizontal well model. **Figure 5.22** shows a comparison between different values of V. Unlike dual porosity model, the value of V=0.66 is performing better in the horizontal well problem as can be seen in **Figure 5.22**.

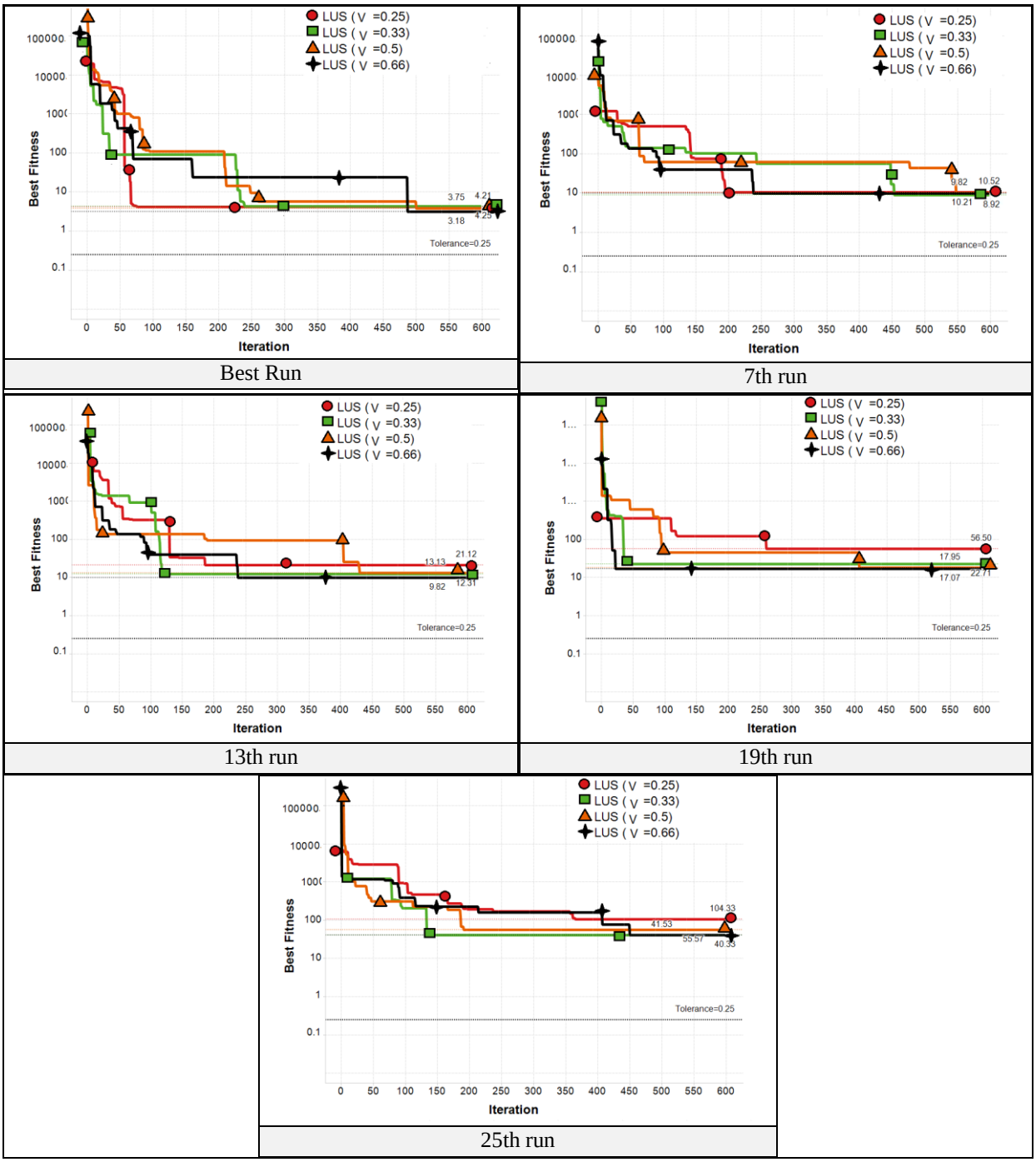
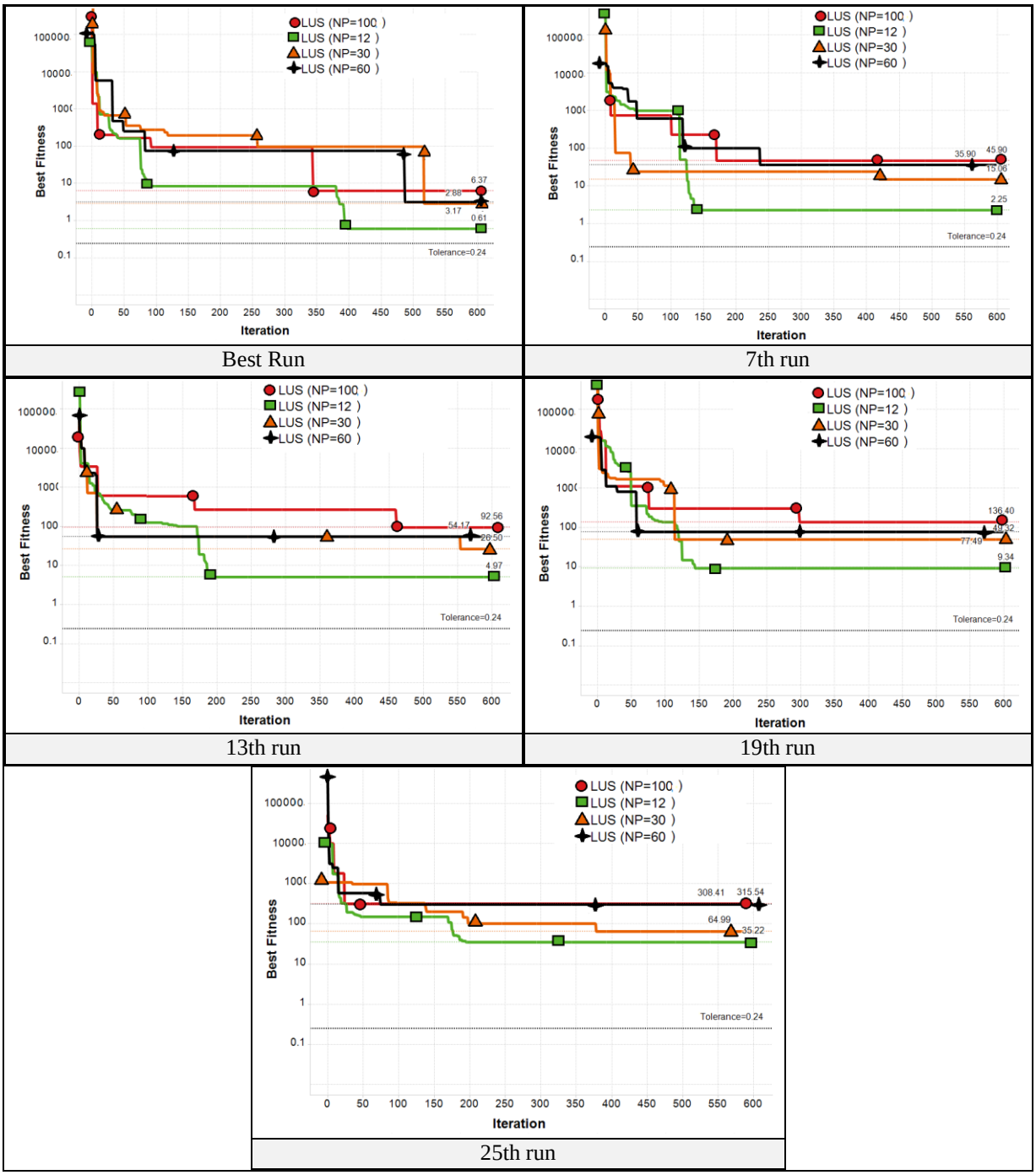


Figure 5.22 LUS performance using 25 particles and different configuration

Figure 5.23compare the performance of LUS algorithm when using population size. In all the realizations, Np=12 gave the best performance.



Best Run | 7th run

13th run | 19th run

25th run

Figure 5.23 LUS performance using different population size

5.2.2.4 MOL

The MOL was tested on a model horizontal well problem using the same set of algorithm parameters (C_g and w) utilized in testing the PSO. The performance of the MOL using the

different sets of behavioral parameters is shown in **Figure 5.24**. The best performance was obtained when $w = 0.28$ and $C_g = 1.14$.

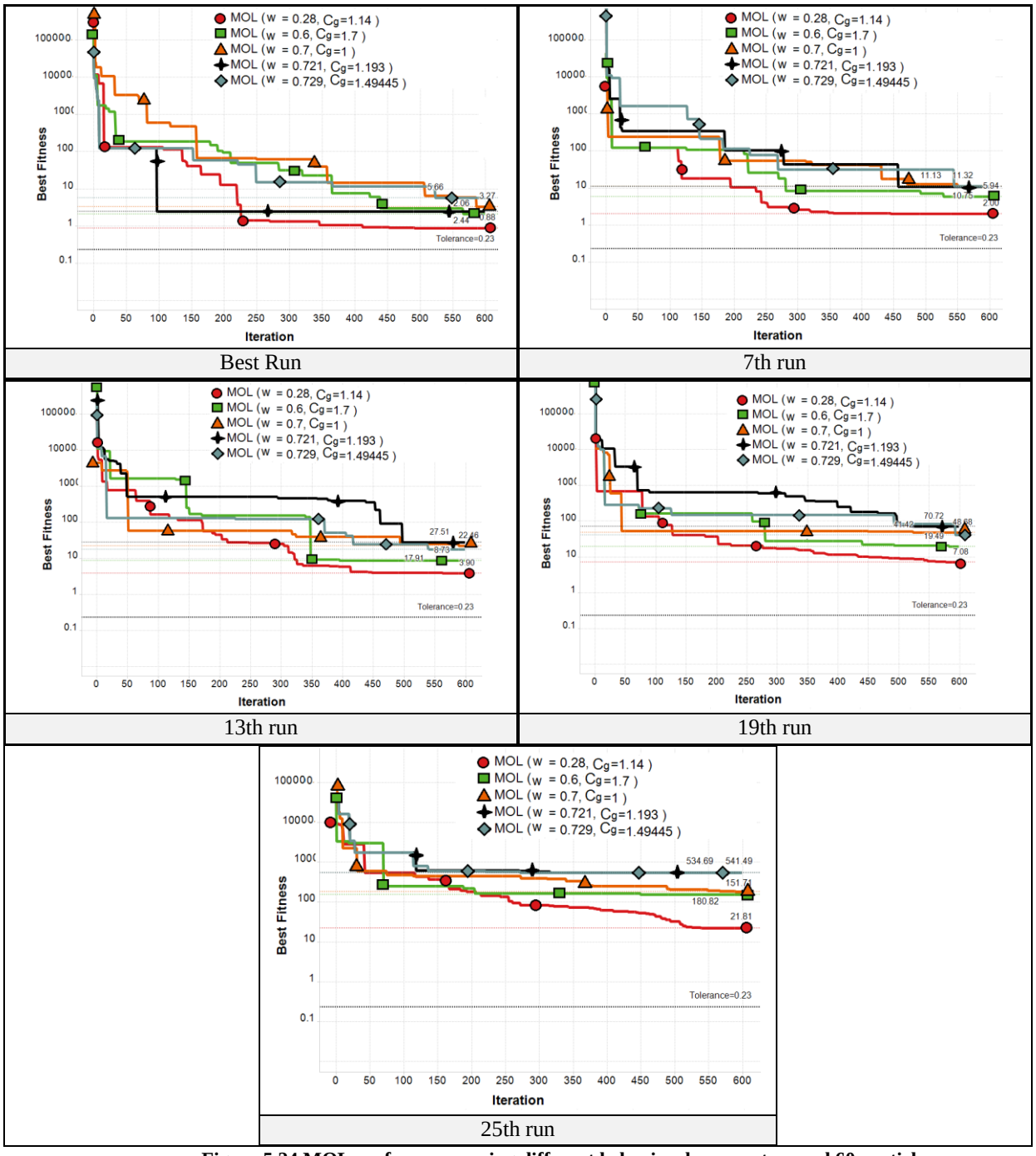
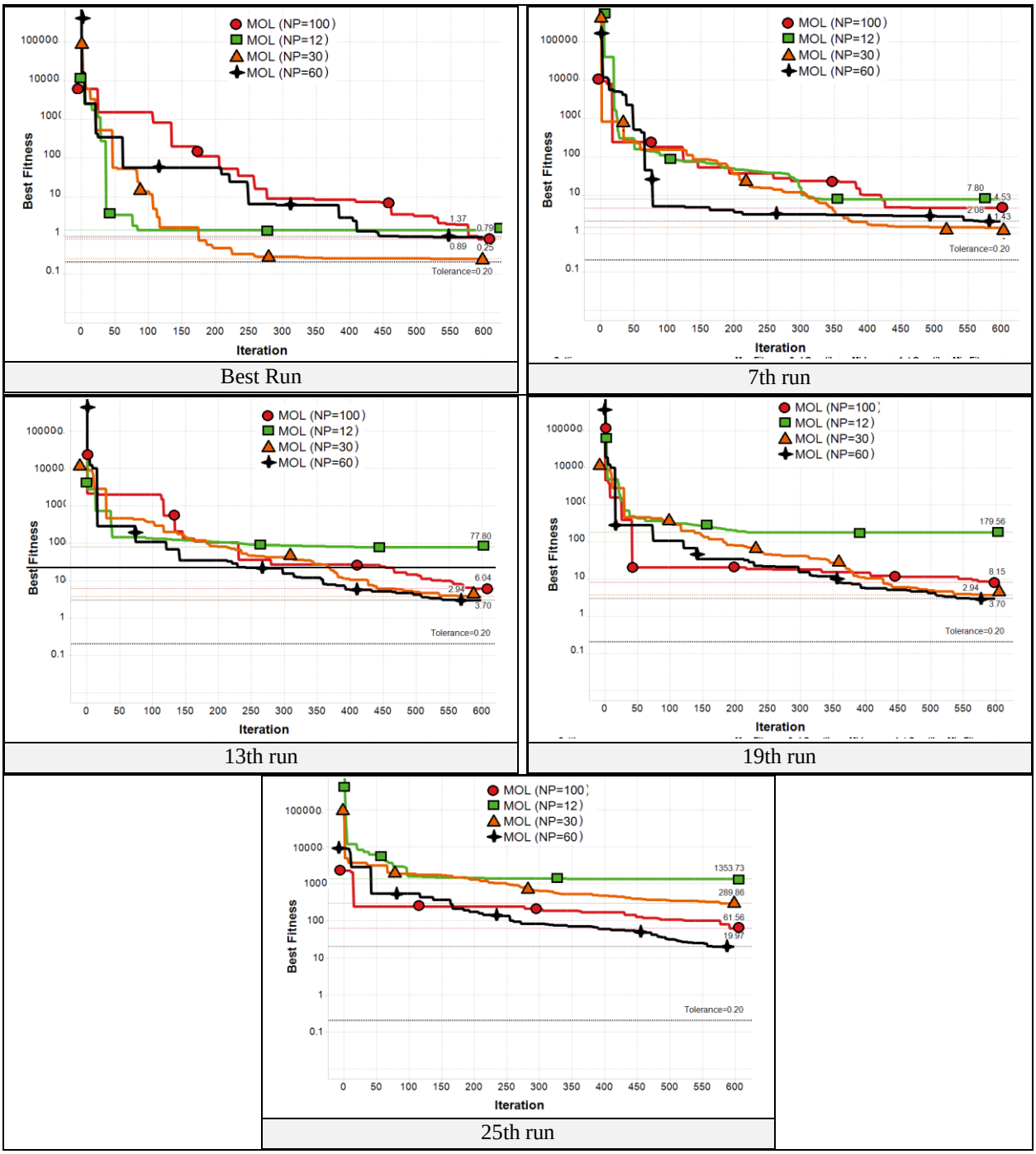


Figure 5.24 MOL performance using different behavioral parameters and 60 particles

Figure 5.25 shows the performance of MOL on the horizontal well model for different population sizes. It was observe that a population size of 30 gave the best performance.



Best Run | 7th run

13th run | 19th run

25th run

Figure 5.25 MOL performance using different population size

5.3 Comparisons of Model Runs

For fair comparison between the optimization algorithms; each algorithm was used to optimize the problem 25 times; each time with a different set of random numbers. the problem here refers to the estimation of reservoir and well parameters from each of the models (dual porosity model and horizontal well model). The 25 runs from each algorithm are arranged in ascending order, with the one with largest error. The 7th, 3th (median) and 19th realizations are also compared. This statistical analysis was performed because one run of each algorithm will not give an adequate picture of the performance of the algorithm over a range of different random numbers. Such analysis has been used in the Congress of Evolutionary Computation.

5.3.1 Dual Porosity model

In the case of the dual porosity model, the algorithm was run to estimate $k_m, k_f, r_e, \omega, \emptyset_m \; and \; skin.$ The best population size and the best set of behavioral parameters obtained in Section 2 of Chapter 4 were used in fitting the model. A total of 900 function evaluations were performed in each realization. Therefore, the algorithm can go through a generation as follows:

$$Number\ of\ generations = \frac{Number\ of\ function\ evaluations}{Population\ size}$$

In each generation, each particle is updated once.

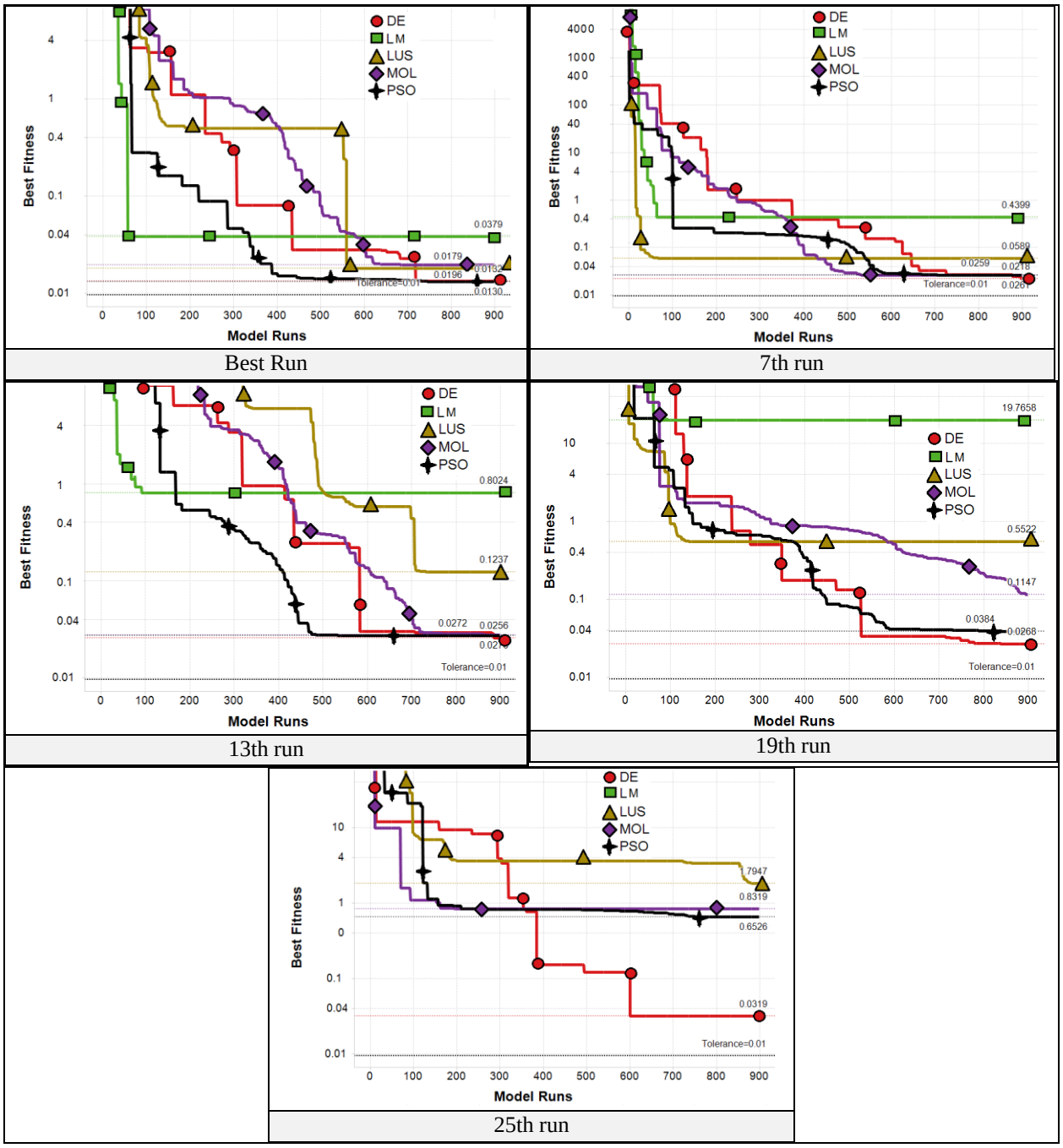


Figure 5.26 Comparison between PSO, DE, LUS, MOL and Levenburg-Maquart optimizing dual porosity model

As can be seen from **Figure 5.26**, DE performed the best in 7th, 13th and 19th realization. At the early generations, many of the other algorithms outperformed the DE. However, at later generations, the performance of the DE was much better than many of the other algorithms. In general, the performance of DE improved with increasing number of function evaluation. The

second best performing algorithm is PSO. The PSO performs better than DE during the early generations, indicating that when only a few functions evaluations can be afforded, the PSO serve as better alternative to the DE. The third best performing algorithm is the MOL, which behaves like the PSO. In comparison with Levenberg-Marquardt, LM, stochastic algorithms performed better. LM's performance is highly influenced by initial starting point. Because six parameters are being optimized, LM algorithm can easily get a trapped in a local minimum. **Appendix C** shows a comparison between the actual model parameters and the obtained parameters by each algorithm in the best realization run of dual porosity model.

5.3.2 Horizontal Well model

In the case of the horizontal well model, the algorithm was run to estimate $k_x,\ k_y,\ k_z\ , h, b\ and\ s$. A population size and behavioral parameters were selected as found in Section 2 of Chapter 4 to try fitting the model to the data using a total of 900 function evaluations per algorithm realization run. It was found that 900 function evaluations were not enough to show the behavior of the algorithms in optimizing horizontal well model. In all the runs Levenberg-Marquardt algorithm performed the best. Therefore, the number of function evaluations was increased to 1800.

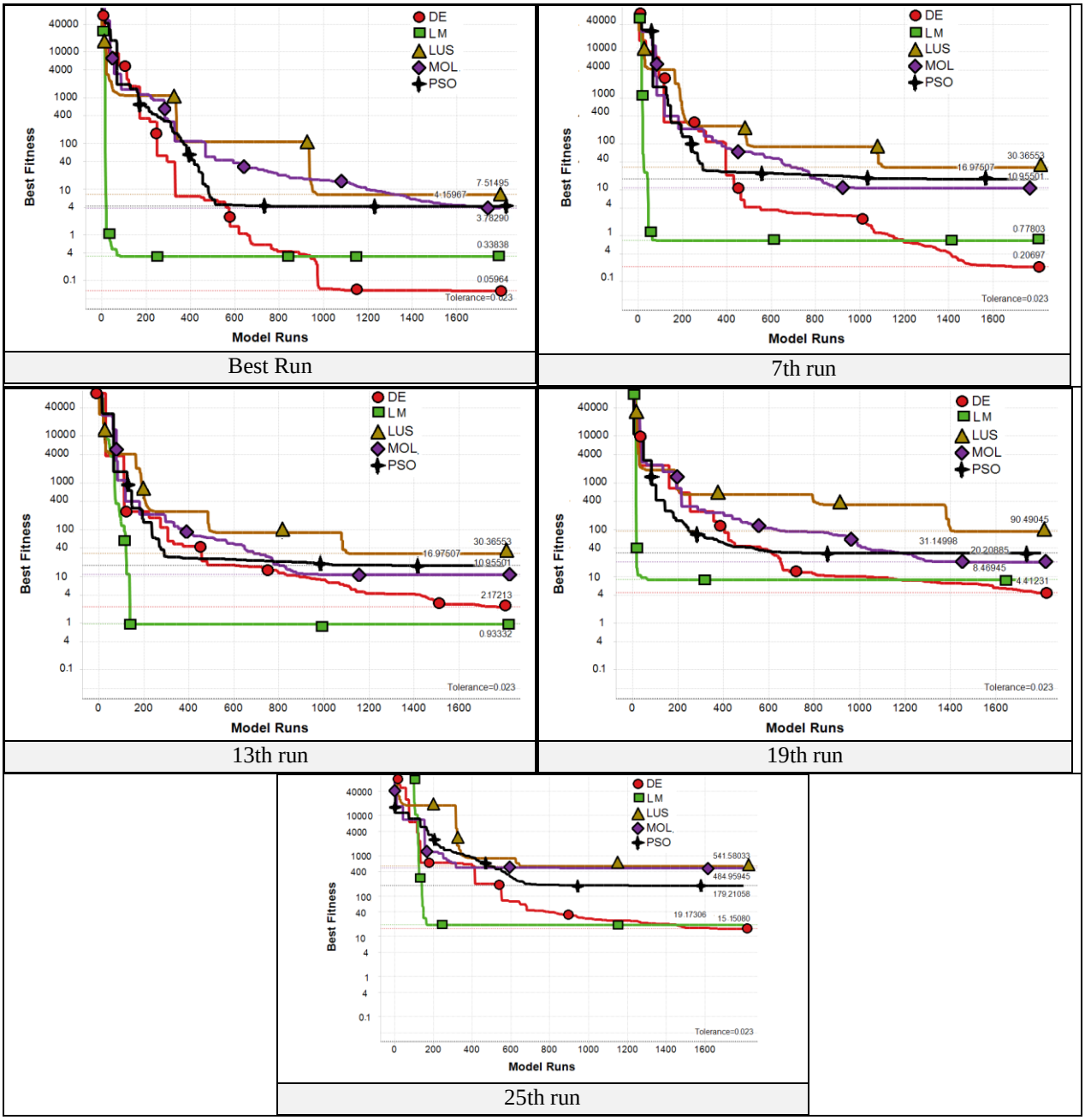


Figure 5.28Comparison between PSO, DE, LUS, MOL and Levenburg-Maquart optimizing horizontal well model

By comparing the performance of the algorithms, it was found that Levenberg-Marquardt performs the best in all the realizations when the number of function evaluations is around 1000. **Figure 5.28** shows that DE is the best performing algorithm as it continued to reduce the error as more functions evaluation were performed. It has a steady performance and it keeps giving

smaller fitness as the number of function evaluations increases. The MOL performance is the second best among stochastic algorithms. MOL and PSO showed good performance and progress until around 900 evaluations, where the improvement is insignificant. The modified version of the LUS performance was the worst among all the stochastic algorithms. **Appendix E** shows a comparison between the actual model parameters and the obtained parameters by each algorithm in the best realization run of horizontal well model.

CHAPTER 6

CONCLUSION AND RECOMENDATIONS

In this study, four stochastic algorithms and the Levenberg-Marquardt algorithm were used to estimate the parameters of the drawdown pressure transient responses of two models: a vertical well in a dual porosity reservoir with cylindrical boundary, and a horizontal well in a box-shaped homogenous reservoir. A code to generate the reservoir response was implemented. Gaussian White Noise data was added to the typical response to simulate measured data. Subsequently, the best behavioral parameters for each algorithm were searched. These are summarized in **Table 6-1**.

Table 6-1 Summery of best behavioral parameters

	Dual Porosity model	**Horizontal Well Model**
Genetic Algorithm (Differential Evolution)	$Cr = 0.9939$ $F = 0.455833$	
	$NP = 30$	$NP = 60$
Particle Swarm optimization algorithm	$w = 0.28, \quad C_p = 1.31 \ \& C_g = 1.14$ $NP = 30\ particles$	
Many Optimizing Liaisons (MOL)	$w = 0.28$ $C_g = 1.14$ $NP = 60$	
Local Unimodal Sampling	$v = 0.33$ $NP = 12$	$v = 0.66$ $NP = 12$

Each algorithm was run for 25 realizations. The results of the runs were ordered in terms of the best achieved result. The performances were compared by considering the best 1^{st}, 7^{th}, 13^{th}, 19^{th} and 25^{th} results of each algorithm. After running all the optimization algorithms, the following observations were made:

- The Levenberg-Marquardt algorithm made significant improvement within the first few function evaluations. If finite difference method was used instead of analytical derivative, the algorithm must make an improvement after each n runs, where n is the number of the unknowns. When the improvement is less than the tolerance value, the algorithm is terminated.
- DE algorithm showed steady improvement with increasing number of function evaluations. The more the number of function evaluations, the more improvement the algorithm achieved.
- The performances of MOL and PSO were always close to each other. This is due to the similarity in their social behavior. However, the PSO's performance was better than the MOL in optimizing dual porosity model, while MOL performed better in the horizontal well model. These algorithms obtained their best performance in half the number of given runs after which they got stalled.
- Before modifying the LUS algorithm, its performance was the worst among all. When the LUS was modified, such as that it was restarted from another random location when its convergence stalled; its performance improved.

For the dual porosity model with a relatively large number of function evaluations and the following 6 unknowns, $\emptyset_m, \lambda, \omega, skin, r_e$ and k_f, the stochastic algorithms were always better than the Levenberg-Marquardt algorithm. In this case, the stochastic algorithms

attained error as low as half of the error attained by the Levenberg-Marquardt algorithm. This can be explained by the geometry of the solution space. The solution space of this model, with these unknowns, has a more complicated geometry that has many local minima. This shape (geometry) of the solution space can cause Levenberg-Marquardt algorithms to get stuck easily in a local minimum. Therefore, the Levenberg-Marquardt algorithms' performance depends on the initial guess of the solution, which is usually selected randomly, and the maximum/minimum values of the unknown, if set. It is recommended to use Chang and Ershaghi (1986) method which scans the solution space and selects the initial guess to be in the minimum neighborhood. Population based stochastic algorithms can avoid this problem since they start from multiple points rather than one. If the objective function is not expensive to run, the DE algorithm is the best to be used since its performance is improved in a large number of runs. On other hand, the PSO algorithm is the best to use when the objective function is expensive. In optimizing the horizontal well model problem where $k_x,\ k_y,\ k_z\ , h, b$ and $skin$ are the unknowns, the stochastic algorithm also showed better performance in minimizing the error. The DE algorithm had the best performance, while the Levenberg-Marquardt performed the second best. This can be explained by the simplicity of the solution space of the problem, such that when following the direction of decreasing gradient can lead to significant reduction in the fitness of the objective function. Knowing that the horizontal well model is expensive to generate in terms of computation time, due to the double integrals of the summation in Equation 4.26, the Levenberg-Marquardt algorithm can be used to optimize horizontal well model if a high number of computation cannot be tolerated. Otherwise, the DE algorithm is the best to use in performing the optimization.

Above scenarios and conclusions are problem specific. They can change from one problem to another. They may also change by changing the dimension of the problem or upper and lower limit. Yet, they reflect the general behavior of the optimization algorithms.

References

[1] Van Everdingen, A.F., and W. Hurst (1949): "The Application of the Laplace Transformation to Flow Problems in Reservoirs," Journal of Petroleum Technology, Volume 1, Number 12. December 1949

[2] Earlougher JR., Robert C, Ramey JR., H.J., Miller, F.G., Mueller, T.D. (1968) :"Pressure Distributions in Rectangular Reservoirs," Journal Paper ,Journal of Petroleum Technology, Volume20, Number 2, February 1968. 1968.

[3] Daviau, F., Mouronval, G., Bourdarot, G., Curutchet, P., Franlab (1985): "Pressure Analysis for Horizontal Wells," Journal Paper, SPE Formation Evaluation, Volume 3, Number 4, December 1988. 1985 SPE.

[4] Clonts, M.D., Ramey Jr., H.J. (1986):" Pressure Transient Analysis for Wells With Horizontal Drainholes," Paper SPE 15116, SPE California Regional Meeting, 2-4 April 1986, Oakland, California

[5] Goode, P.A., Thambynayagam, R.K.M. (1987):" Pressure Drawdown and Buildup Analysis of Horizontal Wells in Anisotropic Media," Journal Paper, SPE Formation Evaluation, Volume 2, Number 4, December 1987.

[6] Abbaszadeh, Maghsood, Kamal, Medhat M. (1988): "Automatic Type, Curve Matching for Well Test Analysis," Journal Paper, SPE Formation Evaluation, Volume 3, Number 3, September 1988.

[7] Aziz S. Odeh, and D.K. Babu (1990): "Transient Flow Behavior of Horizontal Wells: Pressure Drawdown and Buildup Analysis," Journal Paper, Volume 5, Number 1, March 1990.

[8] Thompson, L.G. and Jelmert: "Efficient Algorithms for Computing the Bounded Reservoir Horizontal Well Pressure Response," Paper SPE 21827, Presented at "Low Permeability Reservoirs Symposium", 15-17 April 1991, Denver, Colorado.

[9] Ohaeri, C.U. (1991): "Practical Solutions for Interactive Horizontal Well Test Analysis," Paper SPE 22729, Presented at "SPE Annual Technical Conference and Exhibition", 6-9 October 1991, Dallas, Texas

[10] Issaka, M.B., Ambastha, A.K. (1992): "Drawdown and Buildup Pressure Derivative Analyses for Horizontal Wells," Paper SPE 24323, Presented at SPE Rocky Mountain Regional Meeting, 18-21 May 1992, Casper, Wyoming.

[11] Carvalho, R.S., Redner, R.A. and , L.G., Reynolds. (1992):" Robust Procedures for Parameter Estimation by Automated Type-Curve Matching," Paper SPE 24732. Presented at SPE Annual Technical Conference and Exhibition, 4-7 October 1992, Washington, D.C.

[12] Thompson, L.G. and Temeng, K.O. (1993):" tomatic Type-Curve Matching for Horizontal Wells," Paper SPE 25507, Presented at SPE Production Operations Symposium, 21-23 March 1993, Oklahoma City, Oklahoma.

[13] Buitrago, S., Gedler, G., Intevep, S.A.(1996):" Automatic Optimization of Parameters in Horizontal Well Tests Analysis," Paper SPE 37070, Presented at International Conference on Horizontal Well Technology, 18-20 November 1996, Calgary, Alberta, Canada.

[14] Mir Asif Sultan and Abdulaziz U. Al-Kaabi. (2002):" Application of Neural Network to the Determination of Well-Test Interpretation Model for Horizontal Wells," Paper SPE 77878, Presented at SPE Asia Pacific Oil and Gas Conference and Exhibition, 8-10 October 2002, Melbourne, Australia.

[15] Ahmed S. Zakaria, SPE, Mohamed M. Hafez, Jalel Ochi, Karim S. Zaki, Mehdi Loloi, Ahmed Abou-Sayed. (2001):"Application of Genetic Algorithms to the Optimization of Pressure Transient Analysis of Water Injectors using Type Curves," Paper SPE 143386, Presented at SPE European Formation Damage Conference, 7-10 June 2011, Noordwijk, The Netherlands.

[16] Warren, J.E., Root, P.J. (1963):" The Behavior of Naturally Fractured Reservoirs," Journal Paper SPE 426, SPE Journal, Volume 3, Number 3, September 1963.

[17] Kazemi, H., Merrill JR., L.S., Porterfield, K.L., Zeman, P.R. (1976): "Numerical Simulation of Water-Oil Flow in Naturally Fractured Reservoirs," Journal Paper SPE 5719, SPE Journal , Volume 16, Number 6, December 1976.

[18] Mohammed Ben Issaka, (1992) Horizontal Well Testing Under Thermal and Non-thermal Situation. M.S, University of Alberta.

[19] D. R.HORNER. (1951):" Pressure Build-up in Wells," Conference Paper, 3rd World Petroleum Congress, May 28 - June 6, 1951 , The Hague, the Netherlands

[20] F.J Kuchuk, M. Omur, f. Hollaender.:"Pressure Transient Formation And Well Testing," First Ed.

[21] C.C. Miller, A.B. Dyes and C.A. Hutchinson Jr. (1950):" The Estimation of Permeability and Reservoir Pressure From Bottom Hole Pressure Build-Up Characteristics," Journal Paper, Journal of Petroleum Technology, Volume 2, Number4, April 1950.

[22] Agarwal, Ram G., Al-Hussainy, Rafi, and Ramey Jr., H.J. (1970):" An Investigation of Wellbore Storage and Skin Effect in Unsteady Liquid Flow: I. Analytical Treatment," Journal Paper, SPE Journal, Volume 10, Number 3, September 1970.

[23] Gringarten A. C et al. (1979):" A Comparison Between Different Skin and Wellbore Storage Type-curves for Early-time Transient Analysis," Paper SPE 8205 presented at the 1979 SPE, Annual Technical Conference and Exhibition, Las Vegas, Sept. 23-26.

[24] Bourdet et al. (1989):" Use of Pressure Derivative in Well Test Interpretation," Journal Paper, SPE Formation Evaluation, Volume 4, Number 2, June 1989.

[25] Munson, J. K.; Rubin, A. I.; (1959):"Optimization by Random Search on the Analog Computer," *Electronic Computers, IRE Transactions on* , vol.EC-8, no.2, pp.200-203, June,1959 doi: 10.1109/TEC.1959.5219522

[26] Bartholomew-biggs, M. (1993). :"Nonlinear Optimizationwith Engineering Applications," Springer ScienceBusiness Media, LLC.

[27] Magnus Erik Hvass Pedersen.(2010).: "Tuning & Simplifying Heuristical Optimization," PhD. University of Southampton.

[28] Kennedy, J.; Eberhart, R.; "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on* , vol.4, no., pp.1942-1948 vol.4,Nov/Dec 1995, doi: 10.1109/ICNN.1995.488968

[29] Rainer Storn and Kenneth Proce. "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," Journal Paper, Journal of Global Optimization, 19 November 1996.

[30] Magnus Erik Hvass Pedersen. (2011):"SwarmOps for C#,"January 2011.

[31] Kennedy, J.; , "The particle swarm: social adaptation of knowledge," *Evolutionary Computation, 1997., IEEE International Conference on* , vol., no., pp.303-308, 13-16 Apr 1997 doi: 10.1109/ICEC.1997.592326

[32] Aguilera, R..:" Well Test Analysis of Dual-Porosity Systems, Intercepted by Hydraulic Vertical Fractures of Finite Conductivity," Conference Paper, SPE 18948. Low Permeability Reservoirs Symposium, 6-8 March 1989, Denver, Colorado.

[33] De Swaan, O.A. "Analytical Solutions for Determining Naturally Fractured Reservoir Properties By Well Testing," Soc . Pet. Eng. IT. (June 1976).

[34] Bourdet, D. and Gringarten, A.C. "Determination of Fissure Volume and Block Size in Fractured Reservoirs by Type-Curve Analysis," paper SPE Technical Conference and Exhibition, Dallas (Sept. 21-24, 1980).

[35] R.C. Eberhart and Y. Shi. : "Particle swarm optimization: developments, applications and resources," In Proceedings of the Congress on Evolutionary Computation, volume 1, pages 81 - 86, 2001.

[36] Linah Mohamed, Vasily Demyanov.:" Reservoir Model History Matching with Particle Swarms: Variants Study," Conference Paper, SPE 129152. SPE Oil and Gas India Conference and Exhibition, 20-22 January 2010, Mumbai, India.

[37] Youyunao; Hongqin Chi, "Experimental Study on Differential Evolution Strategies," Intelligent Systems, 2009. GCIS '09. WRI Global Congress on , vol.2, no., pp.19,24, 19-21 May 2009

[38] Vitaliy Feoktistov.(2006):" Differential Evolution In Search of Solutions," United States of America

[39] D. Zaharie. (2002):"Critical values for the control parameters of differential evolution algorithms", 8th International Mendel Conference on Soft Computing, pages 62-67, Bruno, 2002.

[40] Yasin Hajizadeh, Mike Christie and Vasily Demyanov. (2010): "History Matching with Differential Evolution Approach; a Look at New Search Strategies," Conference Paper, SPE130253. SPE EUROPEC/EAGE Annual Conference and Exhibition, 14-17 June 2010, Barcelona, Spain.

[41] Roumboutsos, A., Stewart, G., Heriot-Watt U. (1988): "A Direct Deconvolution or Convolution Algorithm for Well Test Analysis," Paper SPE 18517, 1988.

[42] Nnaemeka Ezekwe (2010).: "Petroleum reservoir engineering practice," Boston: ISBN 0-13-715283-3

[43] Chatas, a.t., Iranian oil exploration and producing co. (1966).:" Unsteady Spherical Flow in Petroleum Reservoirs," Paper SPE 1305,1966. SPE Journal, Volume 6, Number 2

[44] Jerome Onwunalu. (2010)."Optimization of field development using particle swarm optimization and new well pattern descriptions". Published PhD thesis. Stanford University

[45] Olivier Houze, Didier Virturat and Ole S. Fjaere. (2008) "Dynamic Data Analysis". Kappa Engineering.

[46] M.A Sabet(1991). "Well Test Analysis". Houston, Texas. Gulf Publisher Company.

[47] Magnus Erik Hvass Pedersen and Andrew John Chippereld (2009)."Simplifying Particle Swarm Optimization". Applied Soft Computing, Volume, March 2010.

[48] Giovanni Da Prat. (1981)."Well Test Analysis For Naturally-Fractured Reservoir". Published Phddissertation. Stanford University, July 1981.

APPENDEX A

The following code is written in c#.net programming language can be used to generate to dual porosity model.

```
privatevoidGenerateDualPorosityModel(doubleqo,
doubleuo,
double Bo,
double h,
double re,
doublef_phi,
doublem_phi,
doublePr,
double w_,
double lambda,
double cm,
doublecf,
doublerw,
doubleti,
double So,
doublekf,
double WBS,
double skin,
outList<double> t, outList<double> td,
outList<double>Pd, outList<double>dP,
outList<double>Pdp, outList<double>dPdtd,
outList<double>tdpdt)
        {
dPdtd         =         newList<double>();
td            =newList<double>();
t             =      Functions.GenerateLogSpace(double.Parse(this.t_from_txt.Text),
double.Parse(this.t_to_txt.Text), int.Parse(dt_txt.Text));
Pd            =      newList<double>();
dP            =      newList<double>();
Pdp           =      newList<double>();
tdpdt         =      newList<double>();
tdpdt_time    =      newList<double>();
InitStehfest(16);
//--------------Compute dimensionless parameters-------------------%
doublereD     =      re / rw;
double CD     =      (WBS * 0.894) / ((m_phi * cm + f_phi * cf)*h*rw*rw);
//-------------------Beginning of Computation-----------------------%
foreach (doublethetin t)
            {
      doubletD      = (0.0002637*kf*thet)/( (f_phi*cf + m_phi*cm ) * uo*rw*rw);
      td.Add(tD);
      double PD_     = InverseTransform(PD, tD, CD, skin, lambda, w_, reD);
      doubledPDdTd_= tD* InverseTransform(dPDdTd, tD, CD, skin, lambda, w_, reD);
      Pd.Add(PD_);
      dPdtd.Add(dPDdTd_);
      dP.Add((141.2 * qo * uo * Bo * PD_) / (h * kf));
      }
//---------------- Convert to dimention ----------------------%
            int smoothing = int.Parse(this.smothing_txt.Text);
for (int xx = smoothing; xx <dP.Count - smoothing; xx++)
            {
tdpdt_time.Add(
```

```
this.t_model[xx]);
double dt1 = t_model[xx] - t_model[xx - smoothing];
double dt2 = t_model[xx + smoothing] - t_model[xx];
double dp1 = dP[xx] - dP[xx - smoothing];
double dp2 = dP[xx + smoothing] - dP[xx];
tdpdt.Add(tdpdt_time[xx - smoothing] * ((((dp1 / dt1) * dt2) + ((dp2 / dt2) * dt1)) / (dt1 + dt2)));
            }
        }
privatestaticdouble[] V;
privatestaticdouble ln2;
staticvoidInitStehfest(int N)
        {
            ln2 = Math.Log(2.0);
int N2 = N / 2;
int NV = 2 * N2;
            V = newdouble[NV];
int sign = 1;
if ((N2 % 2) != 0)
sign = -1;
for (int i = 0; i < NV; i++)
            {
intkmin = (i + 2) / 2;
intkmax = i + 1;
if (kmax> N2)
kmax = N2;
                V[i] = 0;
sign = -sign;
for (int k = kmin; k <= kmax; k++)
                {
                    V[i] = V[i] + (Math.Pow(k, N2) / Factorial(k)) * (Factorial(2 * k)
                        / Factorial(2 * k - i - 1)) / Factorial(N2 - k) / Factorial(k - 1)
                        / Factorial(i + 1 - k);
                }
                V[i] = sign * V[i];
            }
        }
publicstaticdouble Factorial(int N)
        {
double x = 1;
if (N > 1)
            {
for (int i = 2; i <= N; i++)
                    x = i * x;
            }
return x;
        }

publicstaticdoubleInverseTransform(FunctionDelegate f, double t, double CD, double S, double lambda,
double w, doublereD)
        {
double ln2t = ln2 / t;
double x = 0;
double y = 0;
for (int i = 0; i <V.Length; i++)
            {
                x += ln2t;
                y += V[i] * f(x, CD, S, lambda, w, reD);
            }
return ln2t * y;
        }
```

```
privatedoubledPDdTd(double l, double CD, double S, double lambda, double w, doublereD)
        {
doublepD = PD(l, CD, S, lambda, w, reD);
return (l * pD);
        }
//======= PD with wellborestorage and skin
privatedouble PD(double l, double CD, double S, double lambda, double w, doublereD)
        {

doublepD = PD_Closed(l, CD, S, lambda, w, reD);
return ((l * pD + S) / (l * (1.0 + CD * l * (l * pD + S))));        }

privatedoublePD_Closed(double l, double CD, double S, double lambda, double w, doublereD)
        {
double F = ((w * (1 - w) * l) + lambda) /
                        (((1 - w) * l) + lambda);
double x = reD * Math.Sqrt(l * F);
double y = Math.Sqrt(l * F);
double a =  alglib.besselk1(x) * alglib.besseli0(y);
double b = alglib.besseli1(x) * alglib.besselk0(y);
double c = alglib.besseli1(x) * alglib.besselk1(y);
double d = alglib.besselk1(x) * alglib.besseli1(y);
double e = l * Math.Sqrt(l * F) * (c - d);
return ((a + b) / e);


        }
```

APPENDEX B

The following code is written in c#.net programing language can be used to generate to horizontal well in homogenous boxed shape reservoir model.

```
privatevoidGenerateHorizontalWellModel(doublePr, double phi, double L, double a, double b,
doublect, double Bo,
doubleuo, doublerw, doubleqo, doubleti,
double So, double x0, double y0, double z0, double h, doublekx, doubleky, doublekz, double s,
doublewbs,
outList<double> t, outList<double> td,
outList<double>Pd, outList<double>dP,
outList<double>Pdp, outList<double>dPdtd,
outList<double>tdpdt)
        {
this.Invoke((MethodInvoker)delegate
            {
this.compare_txt.Text = this.compare_txt.Text + this.run_num_lbl.Text + ":\t" + h + "\t" + kx
+ "\t" + b + "\r\n";
            });
dPdtd =
newList<double>();
td =
newList<double>();
            t =
newList<double>();
Pd =
newList<double>();
dP =
newList<double>();
Pdp =
newList<double>();
tdpdt =
newList<double>();
tdpdt_time =
newList<double>();
double NN = 100;
doubleteeta = Math.Atan(Math.Pow((kz / kx), (1 / 4)));
double x = x0 + rw * Math.Cos(teeta);
double z = z0 + rw * Math.Sin(teeta);
double y = y0;
List<double>tt = newList<double>();
tdd =
newList<double>();
List<double>ydd = newList<double>();
//-----------------Compute dimensionless parameters----------------------%
doublexd = x / L; double x0d = x0 / L; double ad = a / L;
doubleyd = y * (Math.Sqrt(kx / ky) / L); double y0d = y0 * (Math.Sqrt(kx / ky) / L); double
y2d = (y0 + L) * (Math.Sqrt(kx / ky) / L);
doublebd = b * (Math.Sqrt(kx / ky) / L);
doublezd = z * (Math.Sqrt(kx / kz) / L); double z0d = z0 * (Math.Sqrt(kx / kz) / L); doublehd
= h * (Math.Sqrt(kx / kz) / L);
// double kd = Math.Sqrt(kx / ky);
//-----------------------Beginning of Computation----------------------%
int i = 0;
doubledtt = double.Parse(dt_txt.Text);
            t =
Functions.GenerateLogSpace(dtt, double.Parse(t_to_txt.Text), 300);
```

```
foreach (doublethetin t)
            {

doubletD = (kx * thet) / (3790.85 * uo * phi * ct * L * L);
td.Add(tD);
//===== Computing S1D and S3D ===== //
double sumS1 = 0; double sumS3 = 0;
for (int n = 1; n < NN; n++)
                {
double S1 = Math.Exp(-(n * n * Math.PI * Math.PI * (tD) / Math.Pow(ad, 2))) * Math.Cos(n *
Math.PI * xd / ad) * Math.Cos(n * Math.PI * x0d / ad);
double S3 = Math.Exp(-(n * n * Math.PI * Math.PI * (tD) / Math.Pow(hd, 2))) * Math.Cos(n *
Math.PI * zd / hd) * Math.Cos(n * Math.PI * z0d / hd);
                    sumS1 = sumS1 + S1; sumS3 = sumS3 + S3;
                }
double S1d = 1 + 2 * sumS1;
double S3d = 1 + 2 * sumS3;
//===== Computing S2D ===== //
double M = 10; ydd.Add(y0d); doubleddy = ((y2d - y0d) / M);
// double y1d = (y + L / 2) / L;
List<double> SS2d = newList<double>();
for (int m = 0; m < M; m++)
                {
ydd.Add(ydd[m] + ddy);
double y1d = ydd[m];
double sumS2 = 0;
for (int n = 1; n <= NN; n++)
                    {
double S2 = Math.Exp(-(n * n * Math.PI * Math.PI * (tD) / Math.Pow(bd, 2))) * Math.Cos(n *
Math.PI * yd / bd) * Math.Cos(n * Math.PI * y1d / bd);
                        sumS2 = sumS2 + S2;
                    }
SS2d.Add(1 + 2 * sumS2);
                }
double S2d = ddy * trapz(SS2d);
Pdp.Add((2 * Math.PI / (ad * bd)) * S1d * S2d * S3d);
            }
TimeEnd:

for (int j = 0; j <t.Count; j++)
            {
dPdtd.Add(Pdp[j] * td[j]);
////// dimentionless derivative
List<double> _td = td.GetRange(0, j);
List<double> _Pdp = Pdp.GetRange(0, j);
Pd.Add(
Functions.trapz(_td, _Pdp));
dP.Add((141.2 * qo * uo * Bo * Pd[j]) / (h * kx));
            }
//======== Adding Skin & WBS ============
InitStehfest(16);
double skin = s;
double CD = (4 * wbs * 5.615) / (2 * Math.PI * phi * ct * h * uo * L * L);
List<double>Pd_s_wbs = newList<double>();
if (this.AddSkinWBS_chbx.Checked)
            {
dP.Clear();
foreach (doubletDin td)
                {
doubledeltaPd = InverseTransform(Functions.AddSkinWBSinLaplase, Pd, td, tD, skin, CD);
Pd_s_wbs.Add(deltaPd);
```

```
dP.Add((141.2 * qo * uo * Bo * deltaPd) / (h * kx));
                }
Pd.Clear();
//======== copy dimentionlesspressuer with skin &wbs ======//
foreach (doublepdinPd_s_wbs)
Pd.Add(pd);
            }
//======== making deravitivetdpdt ======
int smoothing = int.Parse(this.smothing_txt.Text);
for (int xx = smoothing; xx <dP.Count - smoothing; xx++)
            {
tdpdt_time.Add(
this.t_model[xx]);
double dt1 = t_model[xx] - t_model[xx - smoothing];
double dt2 = t_model[xx + smoothing] - t_model[xx];
double dp1 = dP[xx] - dP[xx - smoothing];
double dp2 = dP[xx + smoothing] - dP[xx];
tdpdt.Add(tdpdt_time[xx - smoothing] * ((((dp1 / dt1) * dt2) + ((dp2 / dt2) * dt1)) / (dt1 +
dt2)));
            }
        }
```

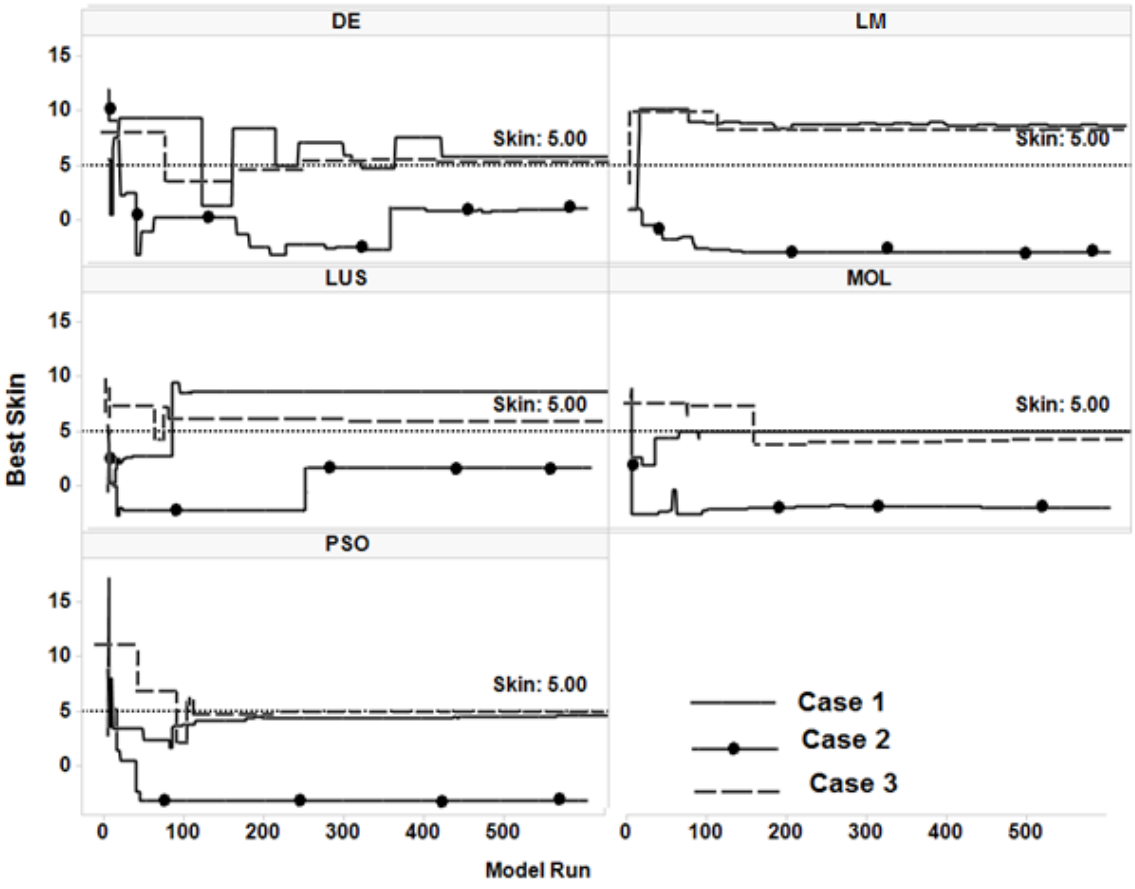
APPENDEX C

Dual porosity reservoir model:

$c_f = 3E-6$, $c_m = 1.5E-5$,$Skin = 5$,$C = 0.1$,$S_o = 1$,$\lambda = 1\text{E}-6$, $\omega = 0.03$,$r_w = 0.25\ ft$,$r_e = 2000\ ft$,$\mu_o = 1$,$B_o = 1.25$,$k_f = 350$,$\emptyset_f = 0.2$,$\emptyset_m = 0.2$,$q_o = 800\ bbl$, $h = 400ft$.

Number of function evaluations per realization: 600.

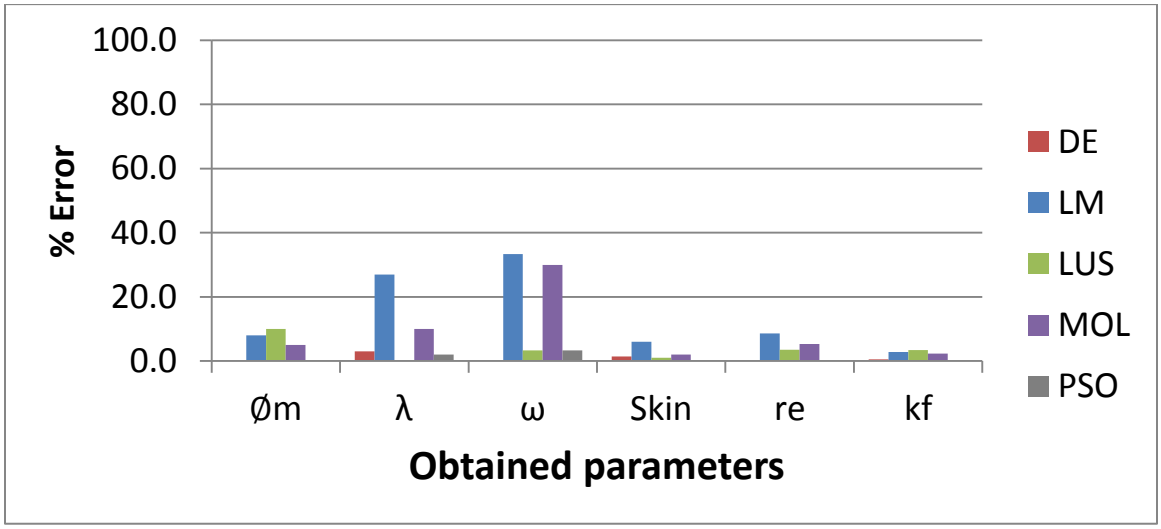
Case	Number of Unknowns	Upper limits		Lower limits		Best Fitness		Obtained Skin	
1	6	λ	1e-5	λ	1e-7	DE	0.091	DE	5.77
		r_e	10000	r_e	1000	PSO	0.092	PSO	4.64
		ω	0.05	ω	0.01	MOL	0.09	MOL	5
		k_f	500	k_f	100	LUS	0.11	LUS	8.6
		$\emptyset_m$	0.25	$\emptyset_m$	0.001	LB	0.14	LB	9.3
		$Skin$	10	$Skin$	0				
2	6	λ	1e-4	λ	1e-8	DE	5.62	DE	1.3
		r_e	20000	r_e	1000	PSO	5.66	PSO	1.1
		ω	0.05	ω	0.01	MOL	6.17	MOL	-1.5
		k_f	1500	k_f	50	LUS	7.23	LUS	-1.8
		$\emptyset_m$	0.3	$\emptyset_m$	0.0001	LB	16.84	LB	-2.8
		$Skin$	20	$Skin$	-3				
3	3	k_f	500	k_f	100	DE	0.05	DE	5.2
		$\emptyset_m$	0.25	$\emptyset_m$	0.001	PSO	0.057	PSO	5
		$Skin$	10	$Skin$	0	MOL	0.07	MOL	3.8
						LUS	0.09	LUS	6.4
						LB	0.12	LB	9


DE
LM
LUS
MOL
PSO
Skin: 5.00
Best Skin
Model Run
Case 1
Case 2
Case 3
0
5
10
15
100
200
300
400
500

APPENDEX D

A comparison between the actual model parameters and the obtained parameters by each algorithm in the best realization run of dual porosity model.

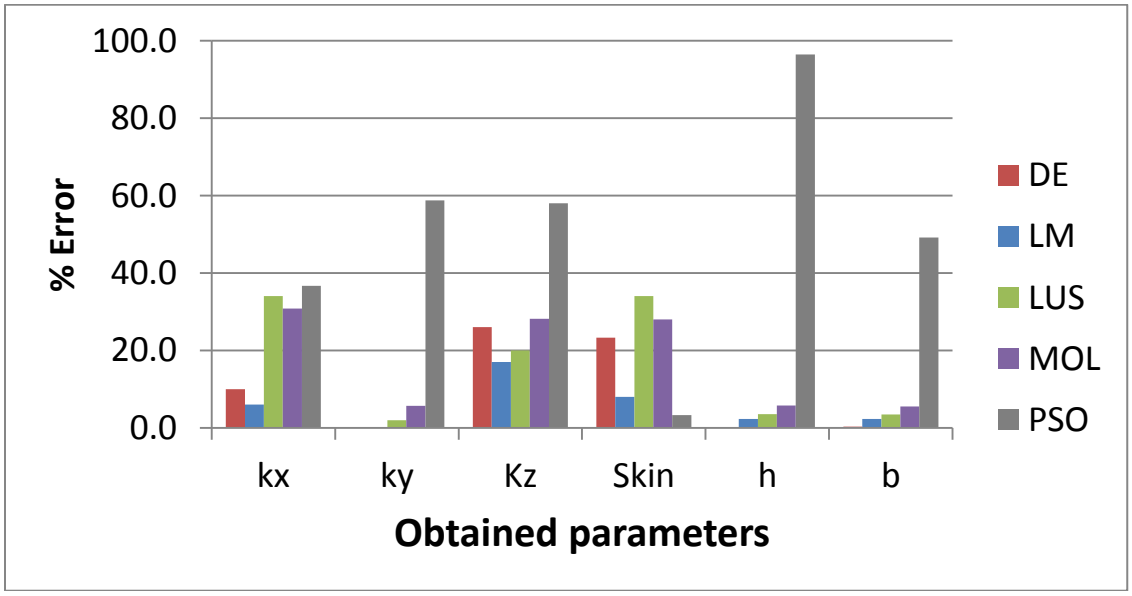
	Actual parameters	DE		LM		LUS		MOL		PSO	
		Obtained	% error	Obtained	% error	Obtained	% error	Obtained	% error	Obtained	% error
Øm	0.2	0.20	0.0	0.22	8.0	0.18	10.0	0.19	5.0	0.20	0.0
λ	1.00E-06	1.03E-06	3.0	7.30E-07	27.0	1.00E-06	0.3	9.00E-07	10.0	1.02E-06	2.0
ω	0.03	0.030	0.3	0.040	33.3	0.029	3.3	0.039	30.0	0.031	3.3
Skin	5	4.930	1.4	5.300	6.0	4.950	1.0	5.100	2.0	5.000	0.0
re	2000	2006	0.3	1828	8.6	2070	3.5	1893	5.4	2004	0.2
kf	350	352	0.6	360	2.9	362	3.4	358	2.3	349	0.3



APPENDEX E

A comparison between the actual model parameters and the obtained parameters by each algorithm in the best realization run of horizontal well model.

	Actual parameters	DE		LM		LUS		MOL		PSO	
		Obtained	% error	Obtained	% error	Obtained	% error	Obtained	% error	Obtained	% error
Kx	50.0	55.0	10.0	47.0	6.0	67.0	34.0	34.6	30.8	31.7	36.7
Ky	50.0	50.0	0.0	50.0	0.0	51.0	2.0	52.9	5.7	20.6	58.8
Kz	5.0	3.7	26.0	5.9	17.0	6.0	20.0	6.4	28.2	2.1	58.0
Skin	2.0	2.5	23.3	1.8	8.0	2.7	34.0	1.4	28.0	2.1	3.3
h	150.0	150.0	0.0	146.6	2.3	144.7	3.5	158.7	5.8	294.6	96.4
b	7532.0	7558.0	0.3	7707.0	2.3	7791.4	3.4	7114.0	5.5	3828.9	49.2



Vitae

Name : Ali Ahmed Alnemer

Nationality : Saudi Arabia

Date of Birth :11/18/1982

Email : ali.alnemer.1@aramco.com

Address : Dammam- Saudi Arabia

Academic Background :

BS. Computer Science from King Fahad University of Petroleum and Minerals; 2005.

Ms. Petroleum Engineering from King Fahad University of Petroleum and Minerals; 2014.

Work Experience :

Coop Student: Saudi Aramco – Research & Development Center. 2004-2005.

System Analyst: Saudi Aramco – Petroleum Engineering Services Department. 2005-Present.