WELL PLACEMENT OPTIMIZATION USING IMPERIALIST COMPETITIVE ALGORITHM

A Dissertation

by

MOHAMMAD ABDULLAH Q. AL DOSSARY

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Hadi Nasrabadi
Committee Members,	Maria Barrufet
	Michael King
	Eduardo Gildin
Head of Department,	Daniel Hill

May 2017

Major Subject: Petroleum Engineering

ABSTRACT

An efficient and optimized field development plan is a crucial and primary aspect of maximizing well productivities and improving the recovery factors of oil and gas fields, and thereby most effectively increasing profitability. In this research, we apply a meta-heuristics algorithm known as the imperialist competitive algorithm (ICA) to determine optimal well locations for maximum well productivity.

The ICA, an evolutionary algorithm that mimics socio-political imperialist competition, uses an initial population that consists of colonies and imperialists that are assigned to several empires. The empires then compete with each other, which cause the weak empires to collapse and the powerful empires to dominate and overtake their colonies.

We compared the ICA performance with that of particle swarm optimization (PSO) and the genetic algorithm (GA) in the following four optimization scenarios: 1) a vertical well in a channeled reservoir, 2) a horizontal well in a channeled reservoir, 3) placement of multiple vertical wells, and 4) placement of multiple horizontal wells. In all four scenarios, the ICA achieved a better solution than did the PSO or GA in a fixed number of simulation runs. We also applied the ICA optimization algorithm to optimize well placement, well type (producer/injector), well configuration (vertical/directional), wellbore length, and drilling schedules for a sector of a Middle East reservoir.

In addition, we conducted sensitivity analyzes on three important parameters (revolution ratio, assimilation coefficient, and assimilation angle), and the analyses show

that the recommended ICA default parameters generally led to acceptable performances in our examples. However, to obtain optimum performance, we recommend tuning the three main ICA parameters with respect to specific optimization problems.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Hadi Nasrabadi, and my committee members, Professor Maria Barrufet, Professor Michael King, and Dr. Eduardo Gildin for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience. I also want to extend my gratitude to Saudi Aramco, which sponsored me as a PhD student.

Finally, thanks to my mother and father for their encouragement and to my wife for her patience and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Dr. Hadi Nasrabadi (chair advisor) and Professor Michael King of the Department of Petroleum Engineering and Dr. Eduardo Gilding of Department of Petroleum Engineering and Professor Maria Barrufet of the Department of Chemical Engineering.

All work for this dissertation was completed independently by the student.

Funding Source

Graduate study was supported by a sponsorship from Saudi Aramco.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

In recent years, the optimization of well placement has been the focus of much research as field developments have become more challenging and oil and gas producers have tried to respond appropriately to increasing global energy demands. As such, optimization techniques must be utilized to determine the field development plan that will yield maximum well productivity as well as increased effectiveness of the recovery factors.

Well placement optimization is a complex and challenging problem due to its non-linear nature involving various decision variables, constraints, and multiple scenarios that have both direct and indirect effects on the final optimal solution. For instance, well placement optimization problems are typically solved by optimizing the well location, configuration (vertical, horizontal, multilateral), type (producer, injector), and operation status (open, closed) (Nasrabadi et al., 2012). Decision variables with uncertainties and solution constraints lead to a very complex and large solution space, with many possible solution combinations. In addition, most optimization problems have various types of physical and economic constraints that must be heeded during the optimization process. Depending on the nature of the desired outcome of the optimization problem, the best combination of two or more conflicting objective functions (maximizing net present value (NPV) while minimizing cumulative injected water) must be found, which increases the complexity of the problem. Also, the many possible variable combinations mean that it is not sufficient to use only intuitive

engineering judgment to determine the optimum sets of problem variables. Optimization methods have also been applied to other petroleum-related problems such as history matching and inversion in reservoir simulations (Zhang, F. and Reynolds, A., 2002; Chunduru, R.K., et al., 1997; Sen, M. and Stoffa, P., 1995)

Many researchers have designed and implemented automated simulation algorithms that optimize the drilling of infill wells. Franstorm, K.L. and Litvak, M.L. (2000) developed an algorithm that optimizes the placement of new infill wells while also optimizing the recompletion design for existing wells using a full-field simulation model. Litvak, M., et al. (2007) designed and applied an automated optimization algorithm to optimize the placement of new and existing wells, waterflooding, well rates, and surface facilities. The work of these authors yielded substantial improvements with respect to oil recovery and NPV.

Typically, automatic well placement optimization for determining a global optimum solution involves several steps. The process begins with a user selecting an initial well position based on engineering judgment. Then, with the aid of an optimization algorithm, a new and improved well position is suggested. Next, a reservoir response model is applied to the newly suggested position, and the result is reported to the programed algorithm. This step is repeated until a pre-determined stopping point criterion is met.

1.1 Literature Review

Researchers have attempted to solve the well placement problem using various optimization algorithms. The first optimization algorithm used for well placement was the mixed integer programing method in which sets of pre-determined requirements are represented by mathematical linear relationships to maximize an objective function. Rosenwald and Green (1974) utilized this method to identify optimized well locations that would minimize the difference between production and scheduled demand. This method has also been used in offshore field development optimization with respect to platform placements, the number of wells in each platform, pipeline-network designs, and production constraints (Devine and Lesso, 1972; Dogru, 1987; Eeg and Herring, 1997; Garcia-Diaz et al., 1996; Hansen et al., 1992; lyer et al., 1998; Sullivan, 1982; van den Heever and Grossmann, 2000; Watson et al., 1989). The mixed integer programing method is not applicable to non-linear flow conditions and has yielded poor results in cases with a large possible solution space (Nasrabadi et al., 2012).

Gradient-based methods have also been applied to well placement optimization. In these methods, a maximum or minimum solution to an n-dimensional objective function is determined through the application of Taylor-series expansion and the calculation of an objective function gradient. There are two main approaches to the gradient calculations: the finite difference and adjoint-based methods. Gradient-based optimization has been used by several authors to determine optimum well placement and well trajectory for horizontal wells (Bangerth et al., 2006; Castineira and Alpak, 2009; Forouzanfar et al., 2010; Sarma and Chen, 2008; Viemmix et al., 2009; Wang et al.,

2007; Zandvliet, M.J., et al., 2008). Handels et al. (2007) and Wang et al. (2007) also utilized gradient-based optimization techniques. In their work, the authors proposed a workflow by which they calculate the gradient of the objective function and utilize a steepest ascent direction to direct the optimization search. They were successful in achieving a near optimal solution with fewer simulation runs. However, they only considered the optimization of vertical well placement, and they concluded that their proposed optimization workflow will not perform comparably if it is applied to more complex well configurations. In any case, a major flaw associated with the gradient-based optimization method is its high dependency on the initial guess, which may trap it in a local optimum. (Nasrabadi et al., 2012).

Another optimization technique that has been used for well placement is simultaneous perturbation stochastic approximation (SPSA), which is considered to be an approximate gradient-based algorithm. The simultaneous perturbation stochastic approximation (SPSA) algorithm is based on computing the gradient of the objective function in a random direction, which generates two new points that are then used to calculate the objective function of those two new points. If the new calculated objective function shows better value, it is used to generate two more new points and the process continues in the direction that yields a better objective function value (Spall, J.C., 1998; Spall, J.C., 2003). Bangerth et al. (2006) applied simultaneous perturbation stochastic approximation (SPSA) in the placement of multiple vertical wells and concluded that simultaneous perturbation stochastic approximation (SPSA) performed better than both the genetic algorithm (GA) and simulated annealing (SA) in their study example. One

pitfall associated with simultaneous perturbation stochastic approximation (SPSA) is the sensitivity of the chosen step size for calculating new solutions. Researchers have found that if a wrong step size is assigned, a non-feasible solution can be generated that may hinder the solution convergence. Also, simultaneous perturbation stochastic approximation (SPSA) has shown poor handling for discontinuous objective functions, as the calculated gradient might not be defined.

Many probabilistic and stochastic optimization methods have also been applied to well placement optimization. Simulated annealing, first introduced by Kirkpatrick et al. (1983), is a probabilistic method for finding a global optimum by mimicking the recrystallization process of a heated solid object until it becomes a frozen structure that corresponds to a minimum energy configuration. Beckner and Song (1995) and Norrena and Deutch (2002) applied this method for well placement optimization.

Another stochastic optimization method is particle swarm optimization (PSO), a population-based method first introduced by Kennedy and Eberhart (1995) that mimics the searching efforts of animal groups such as birds and fish. In particle swarm optimization (PSO), the algorithm starts by populating possible solutions, which are referred to as particles. Each particle is assigned a position that is updated at each iteration based on its fitness and its relative position to the other particles. The main particle swarm optimization (PSO) algorithmic operator is particle velocity, which is responsible for moving particles through the search space in an attempt to establish a better position (solution) (Helwig, S. and Wanka, R., 2008; Shi, Y. and Eberhardt, R.C., 1998; Engelbrecht, A.P., 2005). Mattot et al. (2006) used particle swarm optimization

(PSO) to minimize the cost of water remediation of produced and injected water by optimizing the number, location, and rate of injection wells. Also, Onwunalu and Durlofsky (2010) applied particle swarm optimization (PSO) to well placement optimization for single and multiple wells as well as for nonconventional well configurations. The authors concluded that in most cases particle swarm optimization (PSO) performed better than the genetic algorithm (GA).

Genetic algorithm (GA) was first introduced by Holland in 1975 when he was inspired by the idea of mimicking the process of natural evolution in a self-adaptive automated algorithm. Genetic algorithm (GA) became popular as a robust and effective algorithm for finding optimum solutions in a highly dimensional and complex solution space. The typical genetic algorithm (GA) process consists of three main operators: selection, crossover, and mutation. The genetic algorithm (GA) process starts with the generation of an initial population that consists of many individuals that correspond to possible solutions. Each individual is created by a set of parameters or variables that constitute chromosomes pertaining to the individual. Then, the generated individuals undergo a fitness evaluation whereby the chromosomes are evaluated, ranked, and selected with respect to their fitness for mating and reproduction (selection). This process is known as survival of the fittest, and the chromosomes that yield better solutions are carried into the next generation. This process continues until a pre-determined criterion is met (Goldberg, D.E., 2004; Haupt, R.L. and Haupt, S.E., 2004; Mitchell, M., 1996).

The application of genetic algorithm (GA) to oil and gas well placement optimization has yielded favorable results that have made genetic algorithm (GA) a widely used and accepted optimization algorithm for solving oil and gas industry optimization problems. For example, Bittencourt and Horne (1997) developed a hybrid binary genetic algorithm (GA) that combines the genetic algorithm (GA) with the polytope method to determine optimal well placement and well configuration (vertical/horizontal) in faulted reservoirs. The polytope method uses linear programing to generate a simplex that includes several vertices. Each vertex is evaluated and the process loops to produce solutions with improved feasibilities based on chosen pivot elements. Montes et al. (2001) applied the genetic algorithm (GA) to vertical well placement optimization and used cumulative oil production as the objective function to be maximized. The authors used two synthetic reservoir models and found that the use of elitism improved the convergence rate to the optimum solution. Emeric et al. (2009) used the genetic algorithm (GA) to determine the best optimum number, location, and trajectory for deviated producer and injector wells in field development. They introduced a handling sub-routine that discards infeasible solutions by applying a crossover between the infeasible solutions and previously determined feasible solutions.

Yeten et al. (2003) constructed a framework that utilized the genetic algorithm (GA) as a searching algorithm to identify the optimum well placement and configuration. The authors used a 3D reservoir model and included the option of having a multilateral well as one possible configuration. Several helper functions, such as artificial neural networks and the hill climber technique, were used to further enhance

the genetic algorithm (GA) optimization performance. Furthermore, the authors applied near-wellbore upscaling to account for near-well heterogeneity. Rigot (2003) further enhanced Yeten et al.'s optimization method to improve the efficiency of multilateral well placement optimization. He introduced a proxy to avoid the need to evaluate the objective function in cases where the expected well productivity is within the validity range of a proxy.

Several researchers have applied well placement optimization to waterflooding projects. Bangerth et al. (2006) applied well placement optimization to both producers and injectors. In their work, the authors compared the performances of the simultaneous perturbation stochastic approximation (SPSA), genetic algorithm (GA), finite difference gradient (FDG), and very fast simulated annealing (VFSA), and concluded that simultaneous perturbation stochastic approximation (SPSA) and very fast simulated annealing (VFSA) produced better results than genetic algorithm (GA) and finite difference gradient (FDG).

Since reservoir modeling is associated with many geological and fluid uncertainties, many researchers have accounted for these uncertainties when developing their optimization techniques by designing a robust optimization workflow. However, since considering multiple realizations in optimization problems increases the computational expense, it is prudent to develop techniques that address this challenge. One approach is to choose a set of realizations that is representative of the whole uncertainty spectrum. For example, Artus et al. (2006) used statistical proxies instead of simulation runs to find the optimal well locations for dual-lateral wells. Their method

makes it possible to find the objective functions without having to run many model simulations. Williams, G.J.J., et al. (2004) developed an automated optimization process that takes into account the effect of reservoir uncertainties on model performance prediction. The authors applied their process on both new and mature fields and reported up to a 20% increase in the overall net present value (NPV). Guyaguler et al. (2000) applied a hybrid optimization algorithm that uses both a binary genetic algorithm and a polytope technique. The authors also incorporated artificial neural networks (ANN) and Kriging techniques to generate function proxies for use in cost function evaluation. Utilizing artificial neural networks (ANN) and Kriging as a substitute for running model simulations reduces the optimization running time. Furthermore, Tupac, Y. J., et al. (2007) used function approximation models as simulator proxies and utilized quality maps to improve the optimization process while also reducing the optimization cost.

Wang et al. (2012) considered reducing the required number of realizations by utilizing a retrospective optimization framework. In their work, the authors considered a sequence of different numbers of realization sets and were able to substantially reduce the computational expense. Also, Yeten et al. (2003) included the risk of inflow control valve failure on well controls optimization. The authors also applied the optimization process to a set of five geological realizations to find the optimized expected net present value (NPV). Cameron et al. (2012) considered a set of geological realizations to find the optimal injected CO_2 rates and used particle swarm optimization (PSO) as their main algorithm engine. Yasari et al. (2013) also applied optimization to a waterflooding project and included robust optimization (RO) to account for geological uncertainty. Van

Essen et al. (2009) also applied robust optimization (RO) in their work by introducing a set of realizations to represent the field uncertainty range.

1.2 Scope of Work

In this research, we apply an imperialist competitive algorithm (ICA) to determine the optimum well location for maximum well productivity. The imperialist competitive algorithm (ICA) is an evolutionary meta-heuristic algorithm that mimics socio-political imperialist competition to search for a global optimal solution. In the imperialist competitive algorithm (ICA), an initial population consists of colonies and imperialists that are assigned to several empires. The empires then compete and as the weak empires collapse the powerful empires take over their colonies. This iterative process continues until the best solution is reached based on a pre-determined criterion. Although the imperialist competitive algorithm (ICA) has not been previously applied to the well placement optimization problem, it has been applied in the context of petroleum engineering for well flow rate predictions (Ahmadi, et al., 2013). Also, imperialist competitive algorithm (ICA) has been applied in other engineering disciplines and has shown potential superiority over other well-known optimization techniques, such as the genetic algorithm (GA) and particle swarm optimization (PSO), in terms of its convergence rate and global optima achievement (Atashpaz-Gargai, et al., 2007; Rajabioun, et al., 2008; Sepehri Rad, et al., 2008). In the first part of this research, we compare the performance of the imperialist competitive algorithm (ICA) with that of the particle swarm optimization (PSO) and the genetic algorithm (GA), with respect to well

placement optimization. In the second part, we apply the imperialist competitive algorithm (ICA) optimization algorithm on a sector of a Middle East reservoir, with the objective of identifying the optimal well placement, configuration, type, and drilling schedule for a waterflooding project while addressing the geological uncertainty associated with the reservoir.

1.3 Dissertation Outline

The objective of this work is to examine the applicability and robustness of the imperialist competitive algorithm (ICA) and benchmark its overall performance against the genetic algorithm (GA) and particle swarm optimization (PSO). In section 2, we present a detailed imperialist competitive algorithm (ICA) optimization algorithm and describe the imperialist competitive algorithm (ICA) algorithm workflow along with an overview of the genetic algorithm (GA) and particle swarm optimization (PSO) algorithms used in this work. Since the imperialist competitive algorithm (ICA) is our main optimization algorithm, we describe the algorithm workflow in detail and more briefly describe the genetic algorithm (GA) and the particle swarm optimization (PSO).

In section 3, we consider the application of the imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO) in four cases. In the first case, we optimize the well placement of a single vertical well in a 2-D channeled reservoir. In the second case, we place a single horizontal well in a 3-D channeled reservoir. In the third case, we simultaneously optimize the placement of multiple vertical wells in a 2-D channeled reservoir. In the fourth case, we consider the

optimized placement of multiple horizontal wells in a 3-D channeled reservoir. The results from these cases reveal a better overall performance by the imperialist competitive algorithm (ICA) compared to the genetic algorithm (GA) and the particle swarm optimization (PSO).

In section 4, we perform a sensitivity analysis for the three main imperialist competitive algorithm (ICA) operators: a) assimilation coefficient, b) assimilation angle coefficient, and c) revolution ratio. We conduct the sensitivity analysis by varying the value of one operator while fixing the values of the other two. We vary each operator with five different values. Our results suggest that the best imperialist competitive algorithm (ICA) operator combination depends on the problem at hand and were found the operator values suggested by the researcher who developed the imperialist competitive algorithm (ICA) algorithm to work well as a first guess.

In section 5, we apply the imperialist competitive algorithm (ICA) optimization algorithm to a sector of a Middle East reservoir. In this example, we want to optimize the number of wells, type of wells (injector/producer), configuration of wells (vertical/horizontal), length of the wellbores, and the drilling schedule for a maximum of ten wells to be drilled within a five-year window. We performed three optimization runs with a maximum of fifty iterations and one optimization run with a maximum of one hundred iterations. We achieved an approximately 5% increase in the net present value (NPV). In sections 6 and 7, we draw our conclusions and describe our plans for future work.

2. OPTIMIZATION TOOLS

Optimization techniques can be defined as a set of systematic processes that search for the set of optimization variables within the lower and upper limits of an optimization problem that yields the most efficient variable values for a pre-determined objective function. Depending on the nature of the optimization problem, the goal is to maximize or minimize the objective function. Optimization techniques can be classified into various groups (Nocedal and Wright, 1999). In this section, we present three of the main optimization categories: a) global or local, b) stochastic or deterministic, and c) constrained or unconstrained.

Local optimization techniques tend to show early convergence to local optima and are considered to be faster than global optimization techniques. Two main disadvantages of local optimization techniques are that convergence to local minima or maxima depends on the initial guess and these techniques perform poorly when dealing with non-smooth and multimodal objective functions (Abo-Hammour, 2002). Global optimization techniques, on the other hand, better handle multimodal objective functions and tend to explore the solution space in an attempt to find global optima. Imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO) are considered to be metahueristic algorithms that search for global optima and, hence, are more suitable for solving the well placement optimization problems presented in our study.

Stochastic optimization techniques are techniques that depend on utilizing

randomness operators to direct their searches to find optimum solutions. One major advantage of stochastic optimization algorithms is their ability to explore and search over a wider spectrum of the solution space through the introduction of sudden changes in the search space that may help to identify better solutions. We note that with stochastic optimization techniques, global optima are found in more than one direction path even if the same initial guess is for several runs. On the other hand, deterministic optimization techniques tend to follow the same direction path from the initial solution guess to converge on an optimum solution, regardless of how many times it is repeated (Spall, 2004). Imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO) are considered to be stochastic optimization techniques.

Optimization problems are classified as constrained or unconstrained based on the set of feasible solutions for a problem. Typically, most real physical problems require that pre-determined constraints be set that include only feasible solutions. This step is essential to avoid unnecessary optimization iterations and to produce a better converged optimum solution. To do so, penalty functions are applied to disregard the choice of infeasible solutions for further iteration. In our work, we use the penalty approach for the imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO). Other approaches for dealing with infeasible solutions can be found in the literature (Clerc, M., 2006; Zhang, W.J., et al., 2004).

The objective of this section is to present an overview of the three main algorithms used throughout this study. These three algorithm techniques are: a) the

imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO), all of which belong to the meta-heuristic algorithm group inspired by natural phenomena and all utilize stochastic search techniques. Imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO) are population-based algorithms wherein optimization starts with a population of several possible solutions, and the goal of the optimization algorithm framework is to continue to improve the population individuals to eventually evolve toward a better solution.

There are a couple of advantages associated with using population-based stochastic algorithms for well placement optimization problems. One advantage is that the algorithms can be parallelized to accelerate the process of finding optimal solutions. Another advantage is that the operating parameters include greedy and exploration parameters that search both for near-best solutions and in random directions to avoid getting trapped in local optima. They require no gradient calculations and hence they perform better with complex and discontinuous objective functions. We present detailed explanations of the imperialist competitive algorithm (ICA), genetic algorithm (GA), and particle swarm optimization (PSO) in the sections below.

2.1 Imperialist Competitive Algorithm

The imperialist competitive algorithm (ICA), first introduced by Esmaeil Atashpaz-Gargari (2007), is an evolutionary algorithm that mimics the competition

between imperialist countries to control more colonies in order to strengthen their empires through a process of imperialistic competition.

The imperialist competitive algorithm (ICA) process is similar to other evolutionary algorithms in that it begins with an initial population, which in the imperialist competitive algorithm (ICA) consists of countries. These countries are then divided into two categories: imperialists and colonies. To generate empires, colonies are distributed among the imperialists based on their relative strengths, as determined by a pre-defined criterion. The empires then compete with each other to control more colonies and expand their power. As this competition loops, stronger empires expand their power by taking possession of weak colonies from weaker empires. This process is repeated until a pre-defined stopping criterion is satisfied. A detailed description of the steps involved in this algorithm is presented in the subsection below.

2.1.1 The Initiation of Empires

The initiation of empires starts with the creation of several arrays that contain different problem variables (P_i). In imperialist competitive algorithm (ICA) terminology, these arrays are called “countries”. Imperialist competitive algorithm (ICA) countries are analogous to individuals in the genetic algorithm (GA). Any country can be defined as a $1 \times$ number of variables (N_{var}) array, for the purpose of cost function evaluation. A country can be either an imperialist or a colony.

$$Country = [P_1, P_2, \dots P_{Nvar}] \tag{1}$$

$$Cost = f(country) \tag{2}$$

Next, an initial population that consists of both imperialists N_{impr} and colonies N_{coln} is generated to form the total population N_{pop}. The formation of initial empires starts by the assignment of colonies to the imperialists, based on the relative power of the imperialists. The number of colonies an imperialist acquires is directionally proportional to its power. This is achieved by normalizing the cost of each imperialist (C_n) and then dividing this value by the total normalized cost of all the imperialists (P_n).

$$C_n = c_n - maxi(c_i) \tag{3}$$

$$P_n = \left| \frac{c_n}{\sum_{i=1}^{N_{impr}} c_i} \right| \tag{4}$$

After the determination of the imperialists' power, colonies are randomly distributed among the imperialists to create empires. The number of colonies held by each imperialist ($N.C_n$) is determined as follows:

$$N.C_n = round(P_n \times N_{coln}) \tag{5}$$

By the end of this process, several empires have been created, each with their relative imperialists and colonies, as depicted in **Fig. 1**. For instance, as shown in **Fig. 1**, five empires are represented by five different colors. The stars represent the imperialists

and the circles represent the colonies. Imperialists and colonies of the same color comprise one distinct empire. The larger the star (imperialist), the more powerful is the empire.

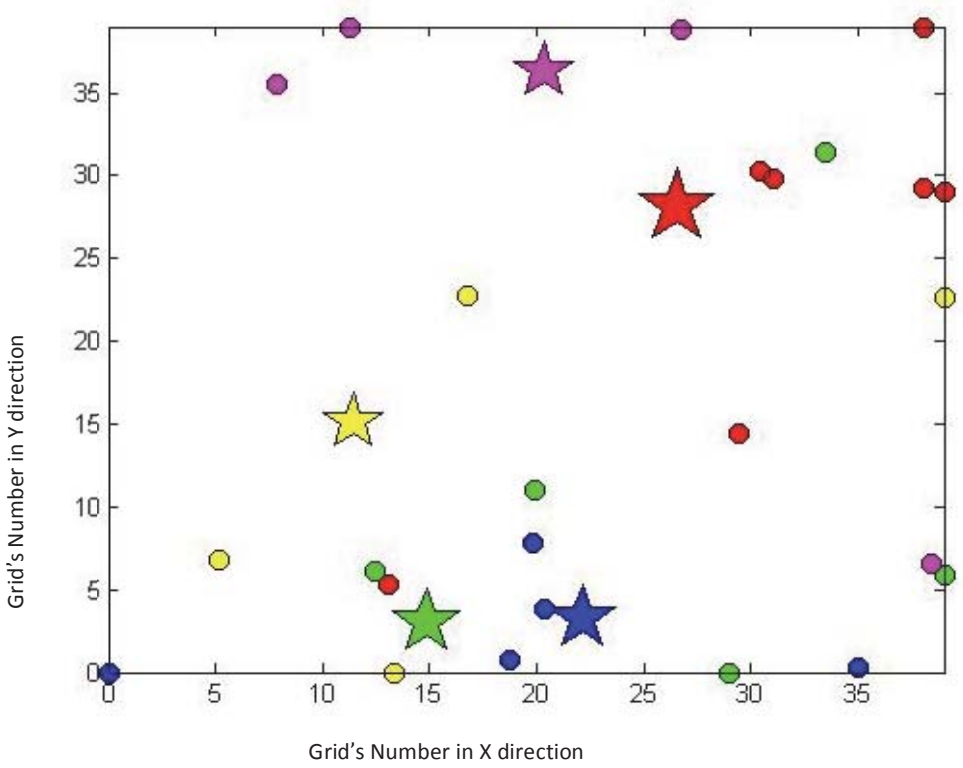

Figure 1: Generation of initial empires. Stars (imperialists) and circles (colonies) of the same color belong to one empire. The red empire is the most powerful as it has more colonies. The stars and colonies represent potential solutions for well placement in the 2-dimensional search space (X and Y are well coordinates)

2.1.2 Assimilation of Colonies

After the initiation of the empires, the next step is to move colonies toward their respective imperialists. In this step, we move all colonies toward their imperialists by moving x distance closer to the imperialist position. The distance x is chosen from a

random distribution within the interval [0, d*(assimilation.coefficient)], as shown in **Fig. 2**. Also, to make the assimilation process more robust and effective, a deviation parameter is assigned to the assimilation process to ensure a greater solution search space.

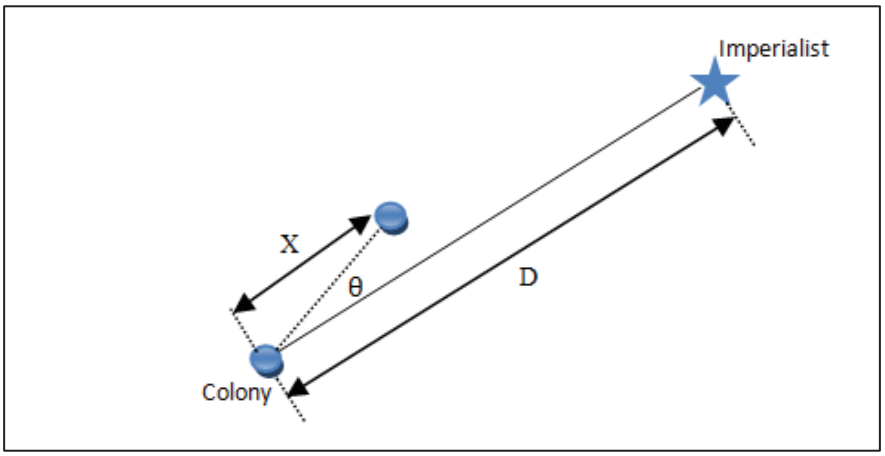

Figure 2: Assimilation process

2.1.3 Revolution

After the colonies move toward their imperialists, some colonies are chosen to participate in a revolution, which, in this case, involves a sudden change in the X and Y coordinates of the well positions. This revolution process facilitates more search and exploration activity in the solution space and thus prevents premature convergence. The revolution process is analogous to mutation in the genetic algorithm (GA).

2.1.4 Imperialist and Colonies Position Exchange

Next, the total power of each empire, which is the summation of its imperialist and colonial powers, is evaluated in order to rank empires based on their lowest cost functions. We note that an empire's power is mainly affected by the power of its imperialist and that the powers of colonies range from very low to almost negligible.

$$T.C_n = Cost(imperialist)_n + \xi \times Mean(Cost(colonies\ of\ empire)_n)) \quad (6)$$

where $T.C_n$ is the total cost of an empire and ξ is a value less than 1. The use of ξ assigns less importance to the colonies' cost and makes the empire's cost mainly dependent on the cost of its imperialist.

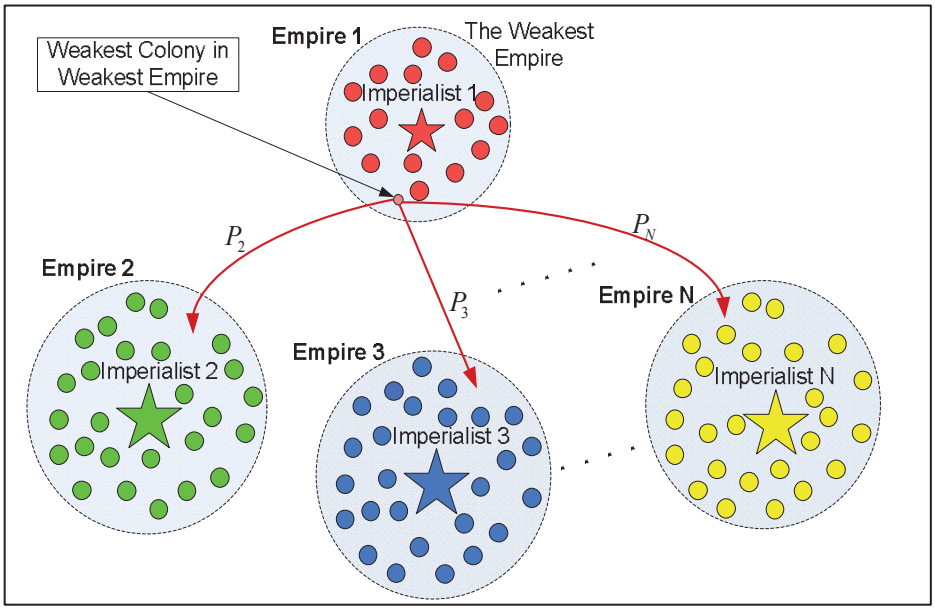

Figure 3: Imperialistic competition to take possession of the weakest colony (from Atashpaz-Gargari, 2007)

Next, the empires are ready to participate in the main imperialist competitive algorithm (ICA) process—imperialist competition. In this process, the empires compete to take possession of the weakest colonies that belong to the weakest empires, as shown in **Fig. 3**. The competition is initiated by assigning a possession probability to each empire. The possession probability increases as an empire's power increases (lowest cost). To perform this process, first, a normalized total cost ($N.T.C_n$) is evaluated for each empire. Then, each empire's total normalized cost ($N.T.C_n$) is divided by the sum of the total normalized cost of all the empires to obtain the possession probability for each empire (P_{pn}):

$$N.T.C_n = T.C_n - maxi(T.C_i) \tag{7}$$

$$P_{pn} = \left| \frac{N.T.C_n}{\sum_{i=1}^{N_{impr}} N.T.C_i} \right| \tag{8}$$

After evaluating the possession probability of each empire, three vectors are formed as follows:

$$\mathbf{P} = [\, P_{p1}, P_{p2}, \ldots., P_{pn}]$$

$$\mathbf{R} = [r_1, r_2, \ldots, r_n] \;\; ; \text{where } r \sim U(0,1)$$

$$\mathbf{D} = \mathbf{P} - \mathbf{R} = [D_1, \ldots., D_n]$$

The weakest colony is then given to the empire with the maximum D index. Further details are provided in the Appendix.

Figure 4 below summarizes the empire initiation, assimilation, revolution, and exchange processes.

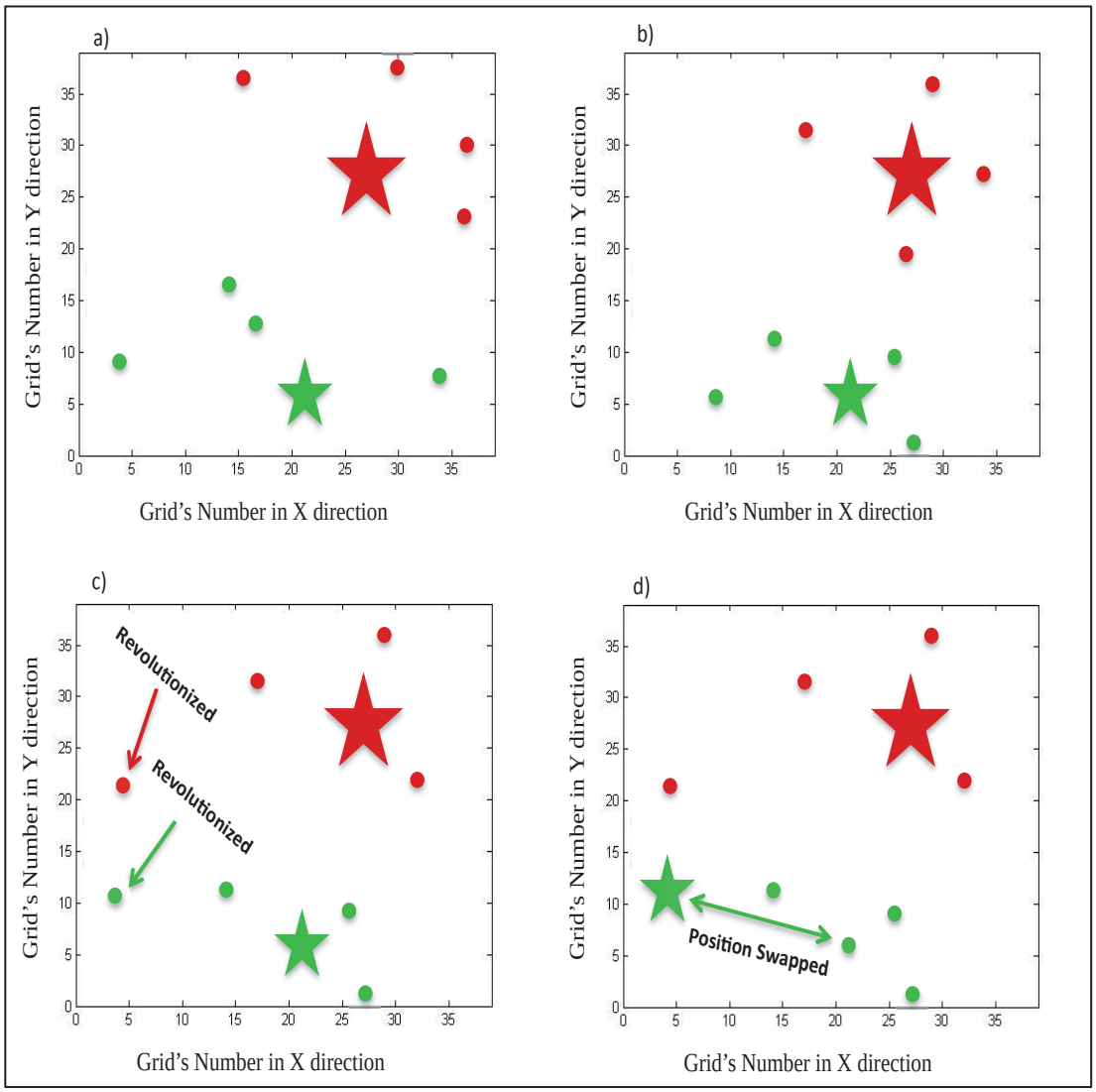

Figure 4: a) Initiation of empires, b) assimilation process, c) revolution process, and d) position swapping process

.

The competitive process repeats and iterates until meeting a pre-determined stopping criterion or until only one empire exists. **Fig. 5** shows an imperialist competitive algorithm (ICA) algorithm flowchart that summarizes the well placement process. For a more detailed description of the imperialist competitive algorithm (ICA) algorithm process, the reader is advised to refer to the original paper by Esmaeil Atashpaz-Gargari (2007).

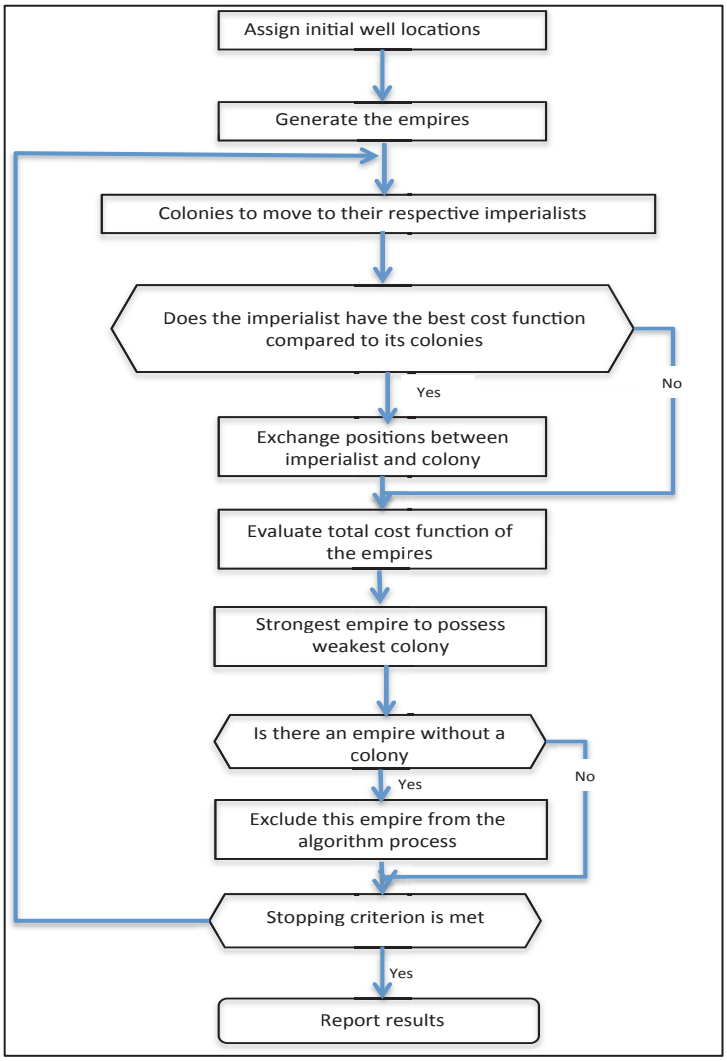

Figure 5: ICA algorithm flowchart

2.2 Genetic Algorithm

The genetic algorithm (GA) is a stochastic meta-heuristic algorithm technique that mimics the natural phenomena of population selection and evolution. The main concept of the genetic algorithm (GA) is the survival of the fittest individual. This occurs by selecting the fittest parents and then mating those parents to generate new offspring. The two operators responsible for producing the new generation are selection and reproduction. Then, some of these offspring undergo some genetic alteration, which occurs by mutation, which is the third operator. There are two main genetic algorithm (GA) types: a) binary genetic algorithm (bGA) and b) continuous genetic algorithm (cGA). The main difference between binary genetic algorithm (bGA) and continuous genetic algorithm (cGA) is that in binary genetic algorithm (bGA) the variables are coded as binary whereas in continuous genetic algorithm (cGA) the variables used are real-valued numbers. In our study, we utilized continuous genetic algorithm (cGA) as our genetic algorithm. A summary of the algorithm steps is as follows:

- Step 1: Set genetic algorithm (GA) parameters (number of individuals, crossover factor, mutation factor, etc.)
- Step 2: Initialize the first generation consisting of N individuals with Nvar chromosomes
- Step 3: Evaluate objective function for each individual
- Step 4: Select best chromosomes that will survive to next iteration (Elitism)
- Step 5: Select individuals that will be used as parents

- Step 6: Apply crossover and mutation to produce new individuals
- Step 7: Reflect changes to the new generation
- Step 8: Repeat steps 3 to 7 until a stopping criterion is met
- Step 9: Return best solution (individual) and best objective value

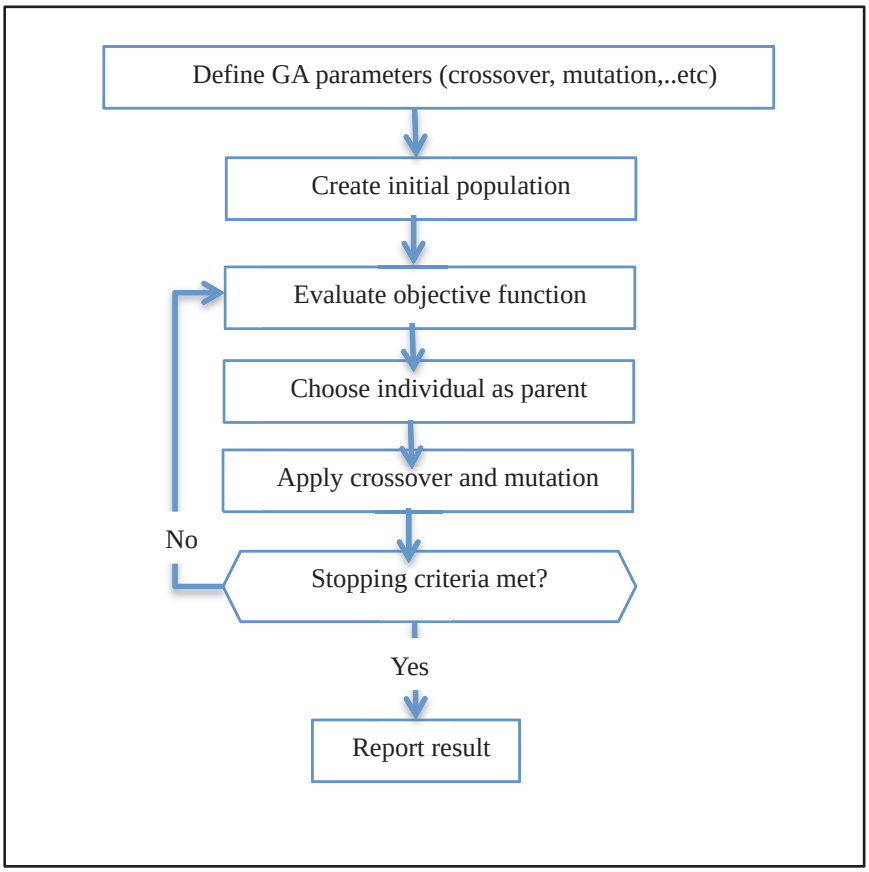

Figure 6: GA algorithm flowchart

2.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic meta-heuristic algorithm technique that mimics how biological species, such as birds and fish, interact with each other socially and cognitively. The algorithm was first developed by Eberhart and Kennedy (1995). Like the imperialist competitive algorithm (ICA) and genetic algorithm (GA), particle swarm optimization (PSO) demonstrates good capability to search for a global solution while reducing the chances of being trapped in local optima. The particle swarm optimization (PSO) optimization process starts by populating a set of possible solutions that are referred as a "swarm" and which contain individual possible solution "particles." These particles move through the search space at each iteration. The movement of the particles is based on the velocity given to each particle. The velocity of each particle is determined by three components: a) inertial (the particle's velocity in the previous iteration), b) cognitive (the component responsible for moving the particle to the best solution found by the particle itself), and c) social (the component responsible for moving the particle to the best solution of all the particles). A summary of the equations used to determine the velocity of each particle and the new position is as follows:

$$v_{i,j}(k+1) = \omega v_{i,j}(k) + c_1 r_{1,j}\left(\hat{y}_{i,j}(k) - x_{i,j}(k)\right) + c_2 r_{2,j}(y_j^*(k) - x_{i,j}(k)) \qquad (9)$$

$$x_i(k+1) = x_i(k) + v_i(k+1) \qquad (10)$$

where v is the velocity; i denotes the particle; k denotes the iteration; ω, c_1, and c_2 are the parameter weights; r_1 and r_2 are random numbers between 0 and 1; $\hat{y}$ is the best position found by the particle; y* is the best position found by all particles; and x is the particle position. Below is a summary of the particle swarm optimization (PSO) optimization workflow:

- Step 1: Set particle swarm optimization (PSO) parameters (number of population, coefficients, inertia)
- Step 2: Initialize the particle velocities and positions.
- Step 3: Evaluate objective function for each particle
- Step 4: Update best position of particle if new position yields better solution
- Step 5: Update the global best fitness value (best solution of all particles)
- Step 6: Calculate the next velocity and position of each particle
- Step 7: Repeat steps 3 to 6 until a stopping criterion is met
- Step 8: Return best position and best objective value

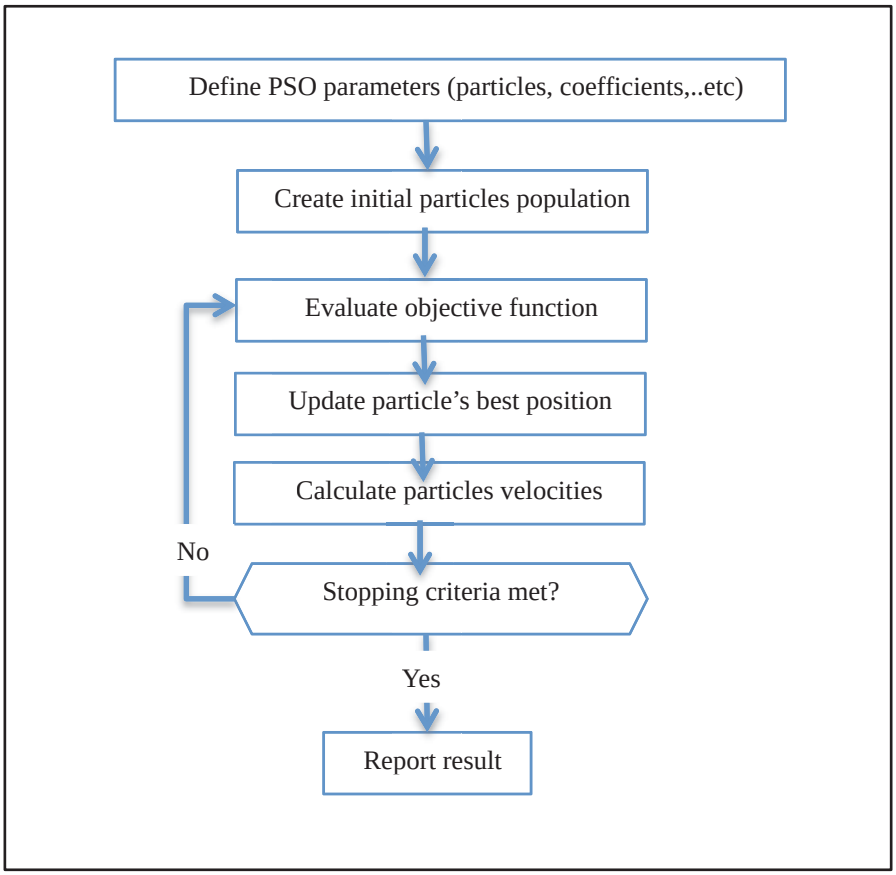

Figure 7: PSO algorithm flowchart

3. OPTIMIZATION RUNS AND RESULTS[*]

To apply the imperialist competitive algorithm (ICA) to our well placement optimization problem, we used a MATLAB code that executes the imperialist competitive algorithm (ICA) processes and interacts with a commercial simulator in order to evaluate all cost functions. For instance, the MATLAB code performs the imperialist competitive algorithm (ICA) steps until reaching the cost evaluation in which optimized input parameters are fed to the reservoir simulator to evaluate the cumulative oil production for a specific well placement position. In this part, we defined the objective function as the cumulative oil production achieved after 21 years of production.

Since the imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) are stochastic in nature; the methodology used in this part is a comparison of the performances of the imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) with somewhat similar algorithm parameters. For example, in all comparison cases, both algorithms were assigned the same initial populations (well locations), similar stopping criteria (number of simulation runs), and a similar number of optimization runs. Then, for the purpose of

---

[*] Reprinted with permission from "Well Placement Optimization Using Imperialist Competitive Algorithm" by Mohammad Al Dossary & Hadi Nasrabadi, 2016. Journal of Petroleum Science and Engineering, Vol. 147, P. 237 – 248. Copyright 2016 by Journal of Petroleum Science and Engineering.

comparison, we graphed the resulting optimization runs for each algorithm to illustrate the average, best, and worst performances.

Moreover, the imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) parameters used in our example cases were similar to those used by Atashpaz-Gargari et. al., (2007), Hosseni et. al., (2014), Onwunalu and Durlofsky (2010), Onwunalu, j. (2006), Yeten (2003) and Farshi (2008), in order to eliminate the necessity of performing a parameters sensitivity analysis, since these other authors have already done so. The imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) parameters used for all cases are summarized in **Table 1, Table 2 and Table 3**. A detailed explanation of the performance of each algorithm is presented in the results section.

ICA Parameter	Value
Number of Countries	30
Assimiliation Coefficient	2
Assimiliation Angle	0.5
Revolution Ratio	0.3
Number of Generations	50

Table 1: ICA parameters used for the optimization runs

PSO Parameter	Value
Number of Particles	30
Inertia Coefficient (ω)	0.721
Cognitive Parameter (C_1)	1.193
Social Parameter (C_2)	1.193
Number of Generations	50

Table 2: PSO parameters used for the optimization runs

GA Parameter	Value
Population Size	30
Crossover Probability	0.8
Mutation Probability	0.05
Ranking Scale	2
Number of Generations	50

Table 3: GA parameters used for the optimization runs

In this research, we used 2D and 3D synthetic channeled reservoir models. The 2D model was used for the placement of vertical wells, while the 3D model used for the placement of horizontal wells. **Table 4** and **Table 5** summarize the input parameters for each model and the pressure-volume-temperature (PVT) data used.

	Vertical Well (Channeled)	Horizontal Well (Channeled)
Number of Grids	40 × 40 × 1	40 × 40 × 3
Each Grid Block Dimension (ft)	300 × 300 × 50	300 × 300 × 50
Permeability Range (md)	1 - 100	1 - 100
Porosity	0.25	0.25
Rock Compressibility (psi^{-1})	0.000003	0.000003
Initial Pressure (psi)	4800	4800
BHP Constraint (psi)	1000	1000
Total Production Time (yrs)	21	21
Initial Water Saturation	0.2	0.2
Initial Oil Saturation	0.8	0.8
Oil Density (Ib/ft^3)	51.5	51.5

Table 4: Reservoir input parameters used in the reservoir models

P	Rs	Bo	Eg	viso	visg	co
14.7	5.11	1.04	4.89	1.74	0.0119	3.E-05
146.1	29.37	1.05	49.51	1.52	0.0120	3.E-05
277.4	58.27	1.06	95.79	1.34	0.0122	3.E-05
671.6	159.03	1.10	244.99	0.97	0.0129	3.E-05
802.9	195.80	1.11	298.22	0.89	0.0132	3.E-05
1760	493.89	1.25	724.30	0.57	0.0164	2.E-05
3280	1037.2	1.53	1321.07	0.38	0.0237	1.E-05
4040	1331.0	1.69	1525.45	0.33	0.0272	8.E-06
4800	1636.2	1.87	1682.55	0.30	0.0304	7.E-06

Table 5: PVT data used in the reservoir models

3.1 Case 1 – Single Vertical Well Placement Optimization

The first imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) optimization application was carried out with respect to the placement of a single vertical well in a channeled reservoir where the permeability distribution varies from 1– 100 md, as shown in **Fig. 8**. First, we performed an exhaustive run in order to determine the cumulative oil production for each possible well

location. With 40 × 40 grids, there were 1,600 possible well locations and hence 1,600 simulation runs. The resulting cumulative oil productions for this exhaustive run is shown in **Fig. 9**. As can be seen from the graph, the objective function surface shows many local maxima scattered throughout the model grid, which is due to the heterogeneity of the subject reservoir model. Furthermore, we found the best cumulative oil production to be 20,206 MSTB, which belongs to the grid that has x = 24 and y = 13 coordinates, **Fig. 11**. As such, we used this grid as a benchmark to test the overall ICA, PSO and GA performances.

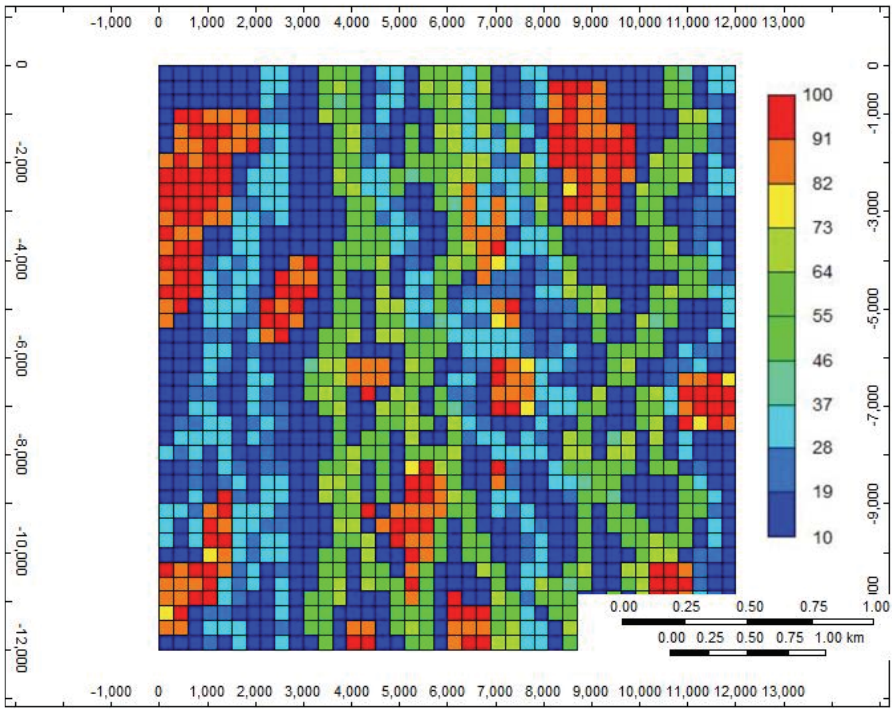

Figure 8: Permeability (md) distribution of the channeled reservoir (case 1)

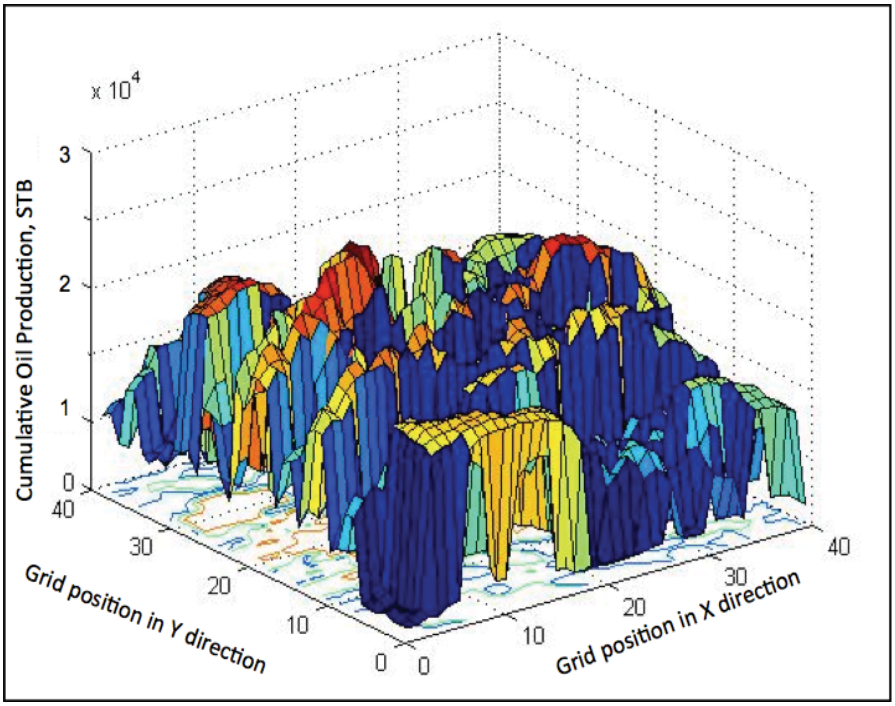

Figure 9: Objective function (cumulative oil production) surface for case 1

Fig. 10 shows a comparison of the cumulative oil production per number of simulations for the imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA). For example, the black line represents the average imperialist competitive algorithm (ICA) performance of 20 optimization runs, the blue line represents the averaged particle swarm optimization (PSO) performance over 20 optimization runs, the red line represents the averaged genetic algorithm (GA) performance of 20 optimization runs, and the green line represents the global optimum. The graph clearly shows that the overall imperialist competitive algorithm (ICA) and particle swarm optimization (PSO) performances are better than the overall genetic

algorithm (GA) performance in converging to the global optimum. Also, both the imperialist competitive algorithm (ICA) and particle swarm optimization (PSO) average performances were able to find the best optimal solution (20,206 MSTB) that was determined by the exhaustive run before reaching the stopping criterion (1000 simulation).

However, the genetic algorithm (GA) overall performance didn't reach the optimal solution by the end of 1000 simulations. The genetic algorithm (GA) resulting maximum cumulative production was 19,650 MSTB. Interestingly, particle swarm optimization (PSO) showed faster convergence rate especially in the first 100 simulations and towards the end of the simulation interval while the imperialist competitive algorithm (ICA) performance exhibits more gradual increase towards the end of the 1000 simulations. This was vividly evident when considering the individual optimization runs where imperialist competitive algorithm (ICA) showed more consistent individual runs compared to those of the particle swarm optimization (PSO) where different runs exhibits high difference as some runs showed high performance while other showed very poor performances. This differences in runs performances made the averaged run to show high jumps with longer periods of flat areas.

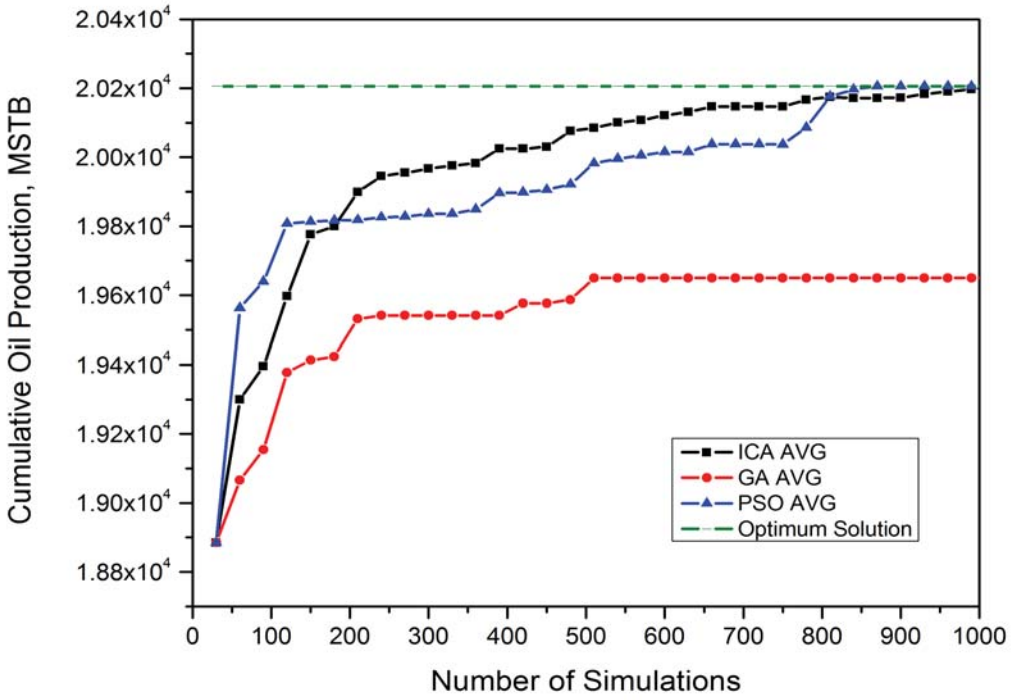

Figure 10: ICA, PSO and GA performances as a function of the number of simulations for the single vertical well case

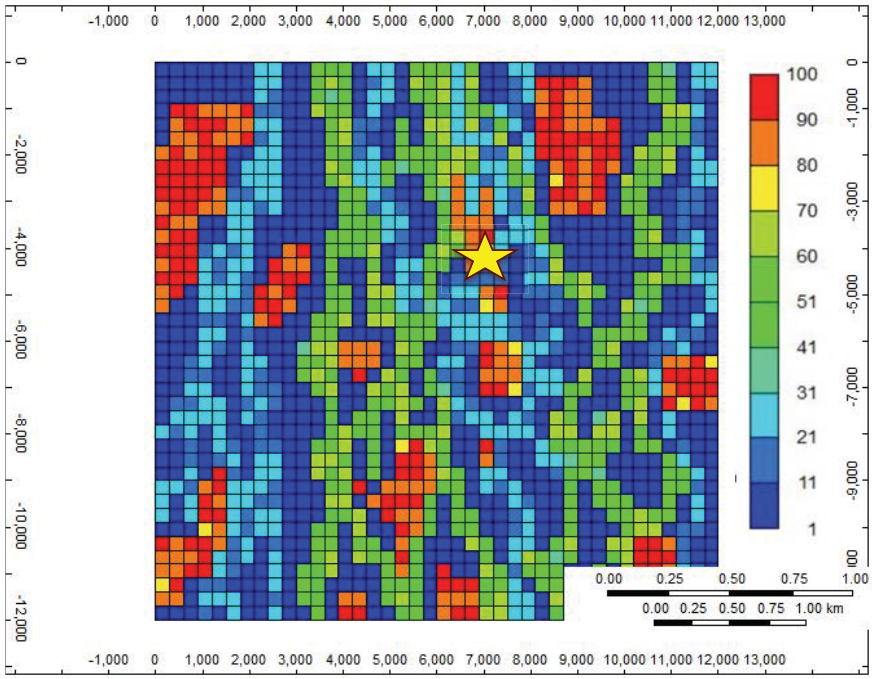

Figure 11: Optimal well location for the single vertical well (yellow star represents the well location)

3.2 Case 2 – Single Horizontal Well Placement Optimization

The second imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA) optimization application was applied to determine the optimal well placement for a horizontal well. We used the same reservoir model as in case 1 but with three horizontal layers, as depicted in **Fig. 12**. The optimized variables for this problem were the x and y well coordinates and an extra variable that dictated the well orientation. For example, in this case the length of the horizontal section was set to

be 8 grids, which could be oriented in four different perpendicular directions, as shown in **Fig. 13**. A physical constraint was induced in the algorithm so that no horizontal section will violate the grids boundaries. If any unfeasible horizontal section is encountered during the algorithm iterations, a penalty function is applied to this violating solution. An exhaustive run was performed and it was found that the best solution corresponds to x = 24, y = 21 and d (orientation) = 3 with a cumulative oil production of 78,842 MSTB, **Fig. 15**.

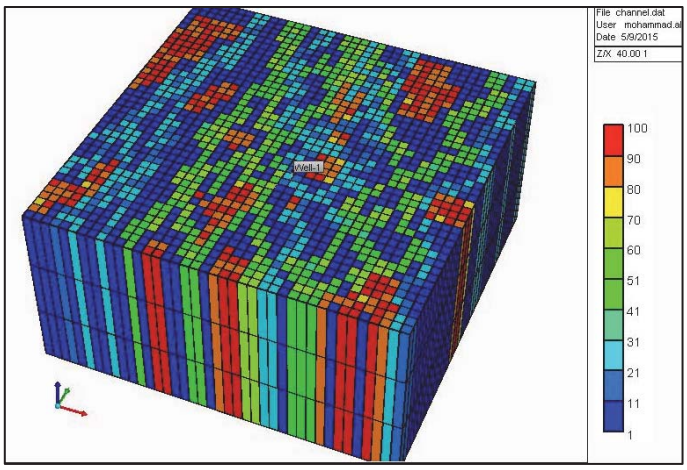

Figure 12: Permeability (md) distribution of the 3-layer channeled reservoir (case 2)

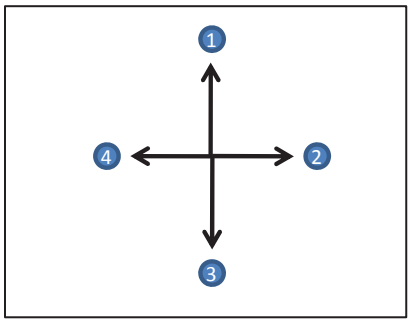

Figure 13: Well directions and corresponding values used as an optimization variable

Fig. 14 shows a comparison of the cumulative oil production per number of simulations for the imperialist competitive algorithm (ICA), particle swarm optimization (PSO), and genetic algorithm (GA). All three algorithms were run 20 times each and then the averaged performances were plotted. The graph clearly shows that among the three algorithms average performances, the overall imperialist competitive algorithm (ICA) performed better and it resulted in the highest cumulative oil production (78,083 MSTB) at the end of the 1000 simulations. Also, it is worth pointing that even though the averaged imperialist competitive algorithm (ICA) performance plotted did not reach the optimal solution that was found by the exhaustive search, three of the twenty imperialist competitive algorithm (ICA) optimization runs were able to converge to the optimal solution before reaching to the 1000 simulations. Another interesting observation that can be deduced from the graph is that the imperialist competitive algorithm (ICA) curve tends to increase in a more gradual pace, which is an indication

that the imperialist competitive algorithm (ICA) can handle the problem of premature convergence by exploring other search space, and hence continue searching with being trapped to local optimal solutions. The resulting maximum cumulative oil production for the particle swarm optimization (PSO) and genetic algorithm (GA) was 75,416 MSTB and 74,083 MSTB, respectively. Moreover, the particle swarm optimization (PSO) performed better than the genetic algorithm (GA) in terms of both convergence rate as well as final optimal solution.

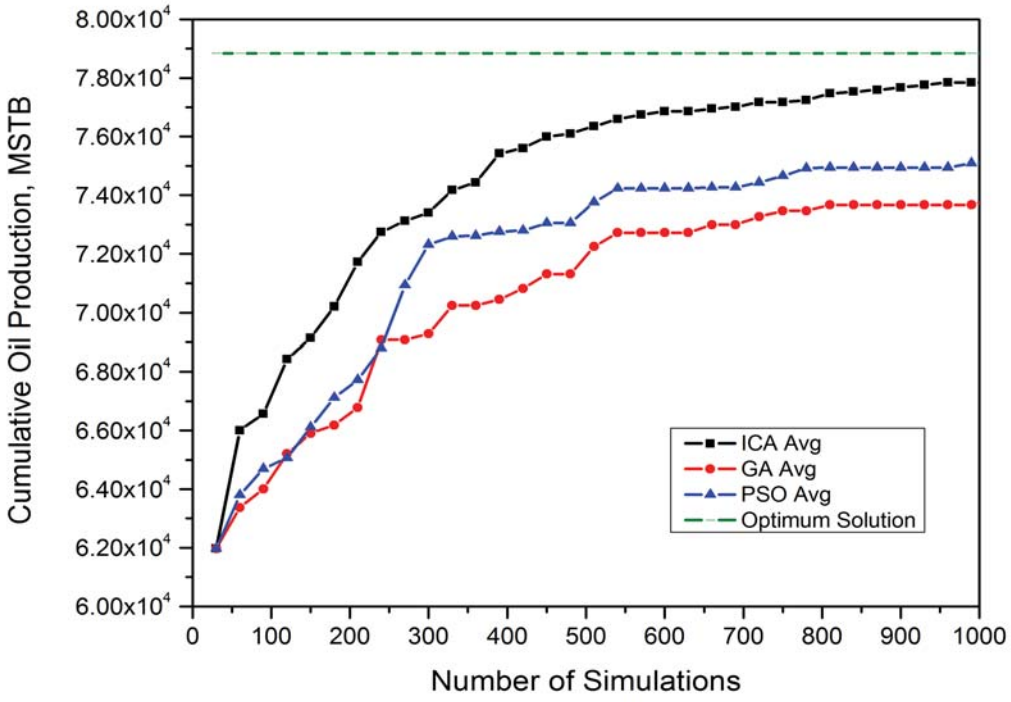

Figure 14: ICA, PSO and GA performances as a function of the number of simulations for case 2

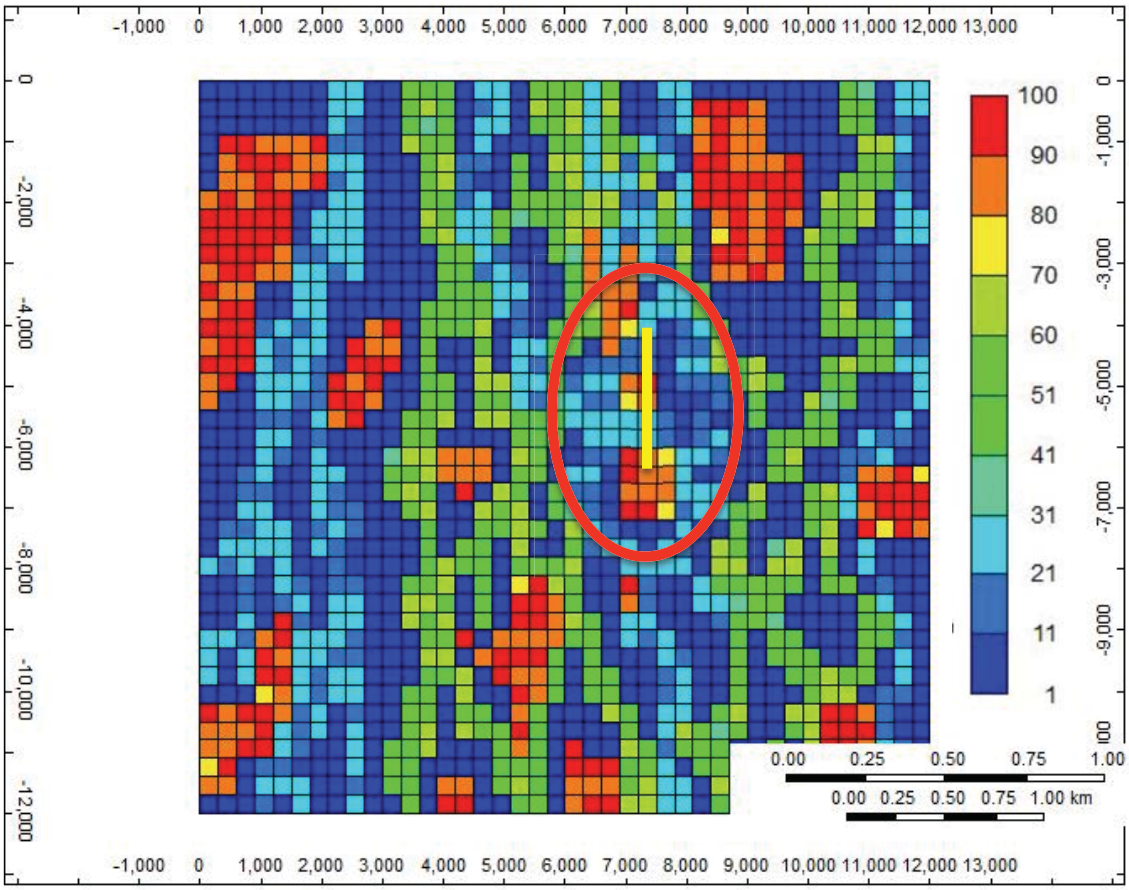

Figure 15: Optimal well location for the single horizontal well (yellow line encompassed by the red circle)

3.3 Case 3 – Multiple Vertical Wells Placement Optimization

In this case, we considered the optimization of multiple vertical wells placement. The optimization variables are x, y and n (number of wells). In this example, we set the number of wells to be from one to no more than ten wells. Since we are including the number of wells as a decision variable, we used the net present value (NPV) as the

objective function that needed to be maximized. The economical parameters used to calculate the NPV's are similar to those used by Onwunalu and Durlofsky (2010). For this case, we did not perform an exhaustive search, as we believed it would require too many simulation runs and the previous two examples were sufficient to build the confidence of the imperialist competitive algorithm (ICA) performance. Also, since we are dealing with multiple wells, a minimum well distance constraint was induced to the algorithm and a penalty function was assigned to any solution that does not honor this constraint. **Fig. 16** shows the overall performances of the three algorithms and again the imperialist competitive algorithm (ICA) showed superiority among the others. For instance, we found that for this case the imperialist competitive algorithm (ICA) provided better solutions all the way from the beginning till the end of the 1000 simulation. Also, the imperialist competitive algorithm (ICA) solution evolvement is gradual with continuous improving and no long flat periods, which may falsely suggest a convergence. The best solution that resulted in the highest NPV was found by imperialist competitive algorithm (ICA) to be three wells as can be seen in **Fig. 17**, which yielded a NPV of $\$3.73\times10^8$ compared to NPV of $\$3.05\times10^8$ for particle swarm optimization (PSO) and $\$2.85\times10^8$ for genetic algorithm (GA).

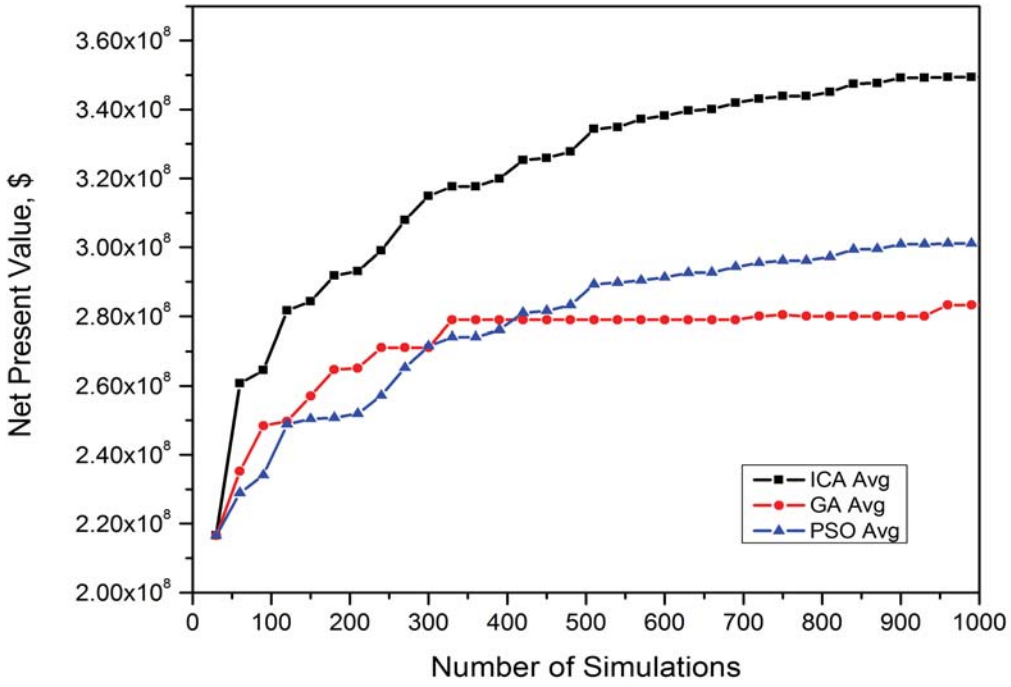

Figure 16: ICA, PSO and GA performances as a function of the number of simulations for case 3

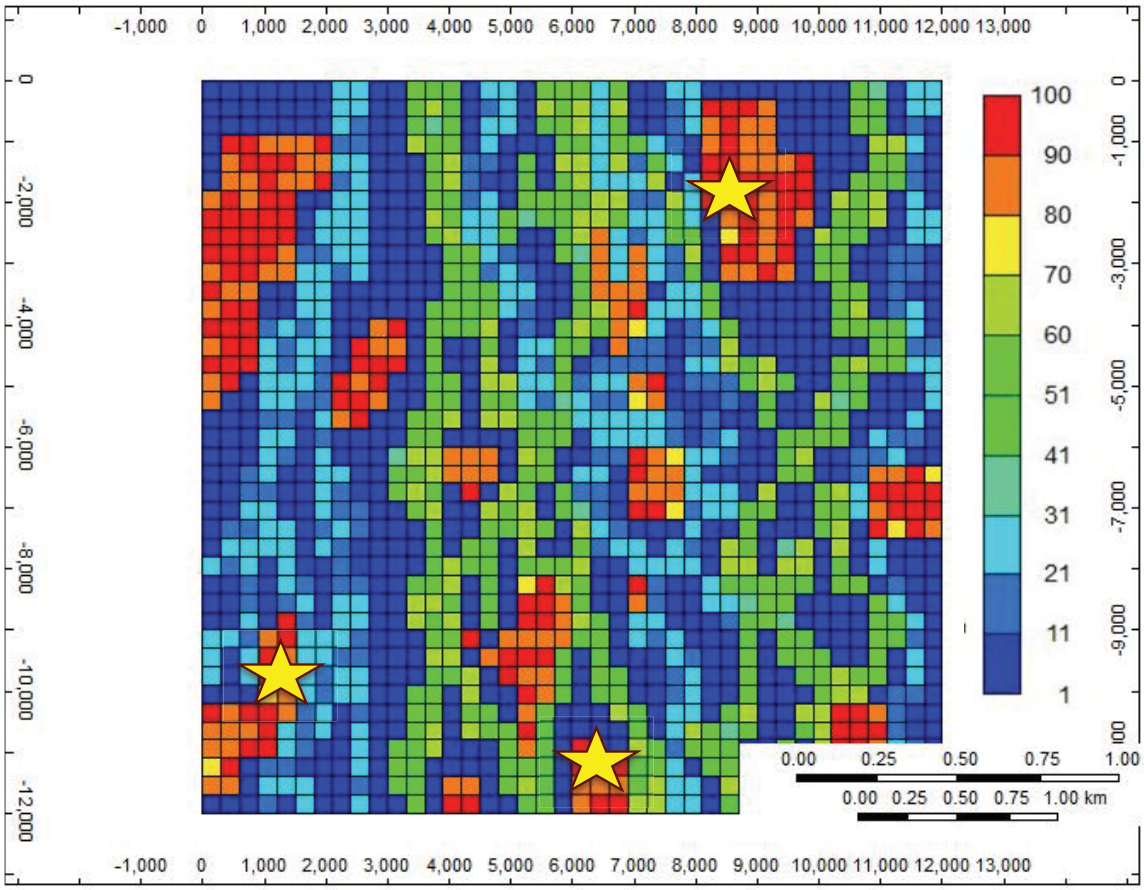

Figure 17: Optimal well locations for the multiple vertical wells case (yellow stars represent the well locations)

3.4 Case 4 – Multiple Horizontal Wells Placement Optimization

For the last case, we considered the optimization of multiple horizontal wells placement. The optimization variables are x, y, d (orientation) and n (number of wells). In this example, we set the number of wells to be from one to a maximum of six wells. The objective function used for this example is the net present value (NPV). The economical parameters used to calculate the NPV's are similar to those used by Onwunalu and Durlofsky (2010). Similar to the multiple vertical case above, we did not perform an exhaustive search for the same reasons stated in case 3. For this case, we induced three physical constraints to the algorithm. These constraints are: 1) a minimum well distance of 1640 ft, 2) no crossing between wells' mainbores, and 3) honoring the model boundaries**. Fig. 18** shows the overall performances of the three algorithms. The results show that the imperialist competitive algorithm (ICA) achieved the highest NPV of $\$1.67\times10^9$ while the particle swarm optimization (PSO) resulted in NPV of $\$1.59\times10^9$ and genetic algorithm (GA) resulted in NPV of $\$1.51\times10^9$. The best solution, which was found by the imperialist competitive algorithm (ICA), is two horizontal wells as can be seen in **Fig. 19**.

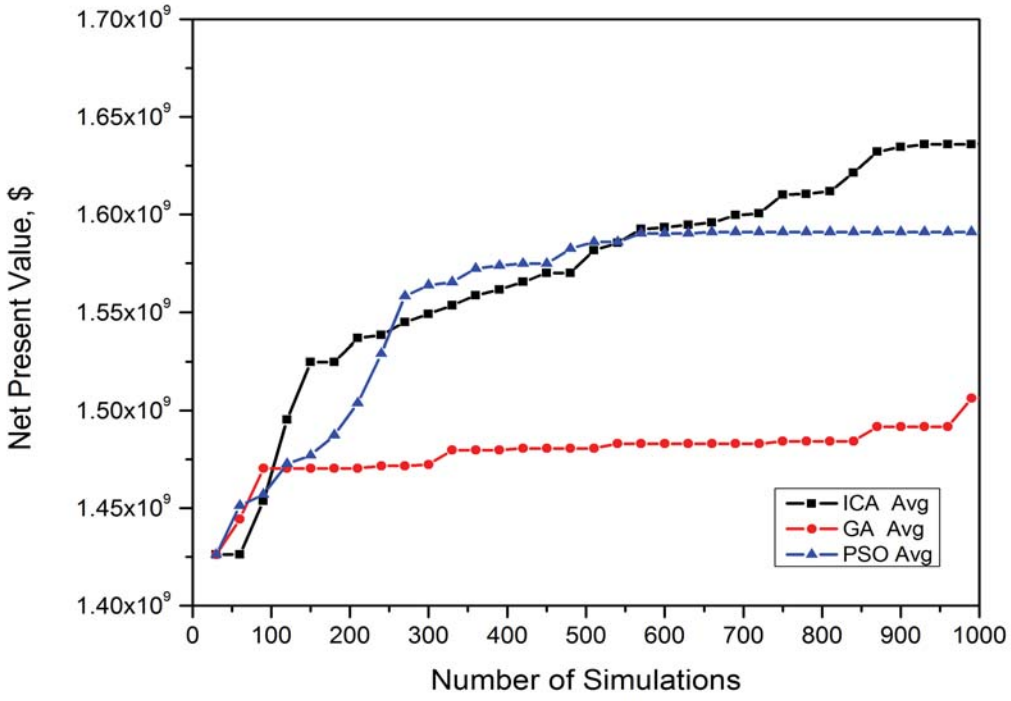

Figure 18: ICA, PSO and GA performances as a function of the number of simulations for case 4

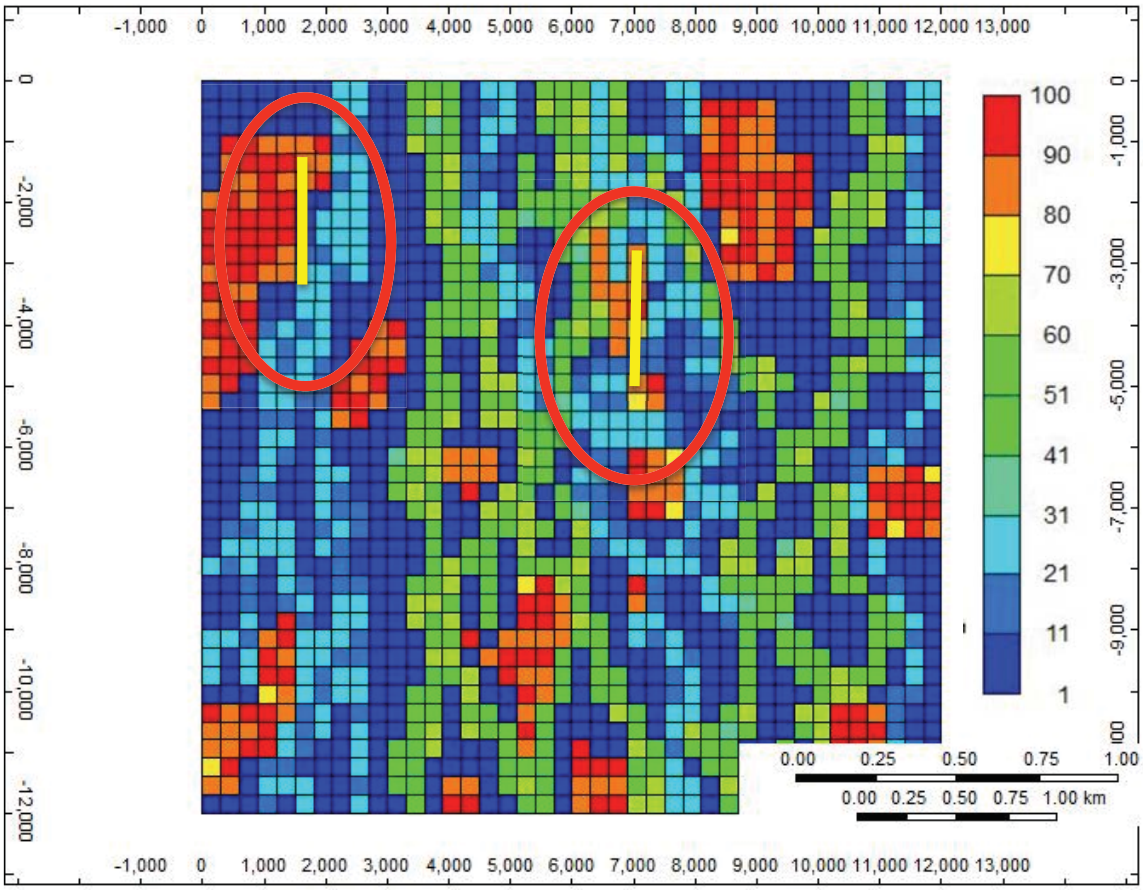

Figure 19: Optimal well locations for the multiple horizontal wells case (yellow lines encompassed by the red circles)

4. SENSITIVITY ANALYSIS OF ICA PARAMETERS*

After applying the imperialist competitive algorithm (ICA) to the above synthetic reservoir model, we performed a sensitivity analysis on the imperialist competitive algorithm (ICA) parameters to further test the effect of the imperialist competitive algorithm (ICA) parameters on the quality of the solution evolvement. We chose three of the main imperialist competitive algorithm (ICA) parameters for this sensitivity analysis, including 1) the revolution ratio, 2) the assimilation coefficient, and 3) the assimilation angle. The reason we chose these specific parameters is that they have similar functionalities compared to genetic algorithm (GA) crossover and mutation operators. In this sensitivity analysis, we performed optimization runs for each parameter while fixing the value of the others. Each parameter was divided into five sets of values and each value was used to perform five optimization runs. We averaged the results of these five optimization runs for the purpose of comparison. For example, the revolution ratio values were 0.3, 0.4, 0.5, 0.6, and 0.7. The analysis began by fixing the value of the revolution ratio to be 0.3 while holding constant the other imperialist competitive algorithm (ICA) parameters. Then, we plotted the average of the five optimization runs using 0.3 as the revolution ratio value. The same process was carried out for the other

---

imperialist competitive algorithm (ICA) parameters (assimilation coefficient and assimilation angle).

4.1 Revolution Ratio Parameter

The first imperialist competitive algorithm (ICA) parameter to be tuned, the revolution ratio, is responsible for creating a sudden change in the positions of an empire's colonies. This parameter can be considered to be analogous to the GA's mutation parameter and is an important imperialist competitive algorithm (ICA) parameter as it is responsible for broadening the exploration for new solutions and thus avoiding pre-mature solution convergence. From a survey of the literature, we found the recommended value for the revolution ratio to be between 0.2 to 0.3 (Atashpaz-Gargari et. al., 2007; Hosseni et. al., 2014). We considered five different revolution ratio values (0.3, 0.4, 0.5, 0.6, and 0.7). For each value, we performed five optimization runs in order to decrease the effect of the stochastic nature of optimization.

We then compared and ranked the parameter sensitivity test results based on the total average objective value, as shown in **Fig. 20** and in **Table 6.**

Ranking	Value
1	0.5
2	0.3
3	0.4
4	0.7
5	0.6

Table 6: Rankings of revolution ratio parameters

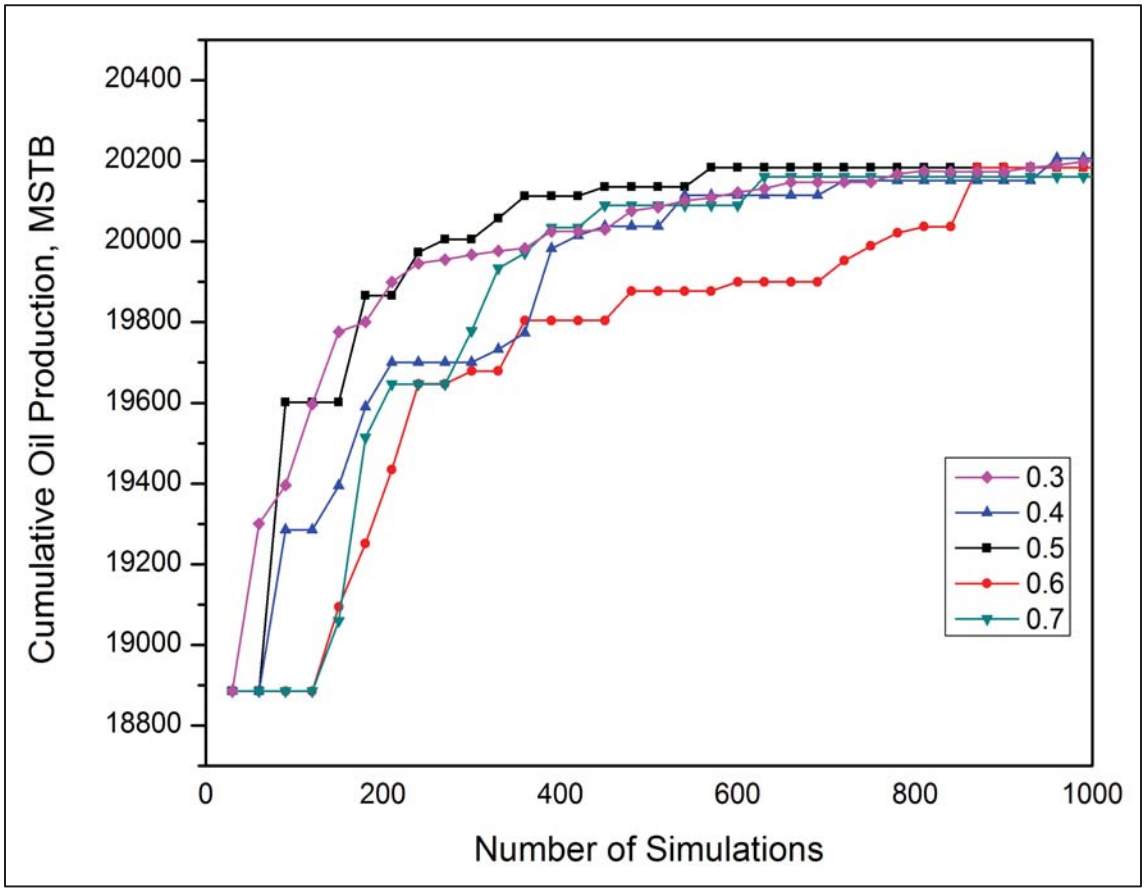

Figure 20: ICA performance with five different revolution ratios for case 1

Based on the above figure and table, we found the best revolution ratio value for 0.5. This suggests that a mid-range revolution ratio would result in a balance between exploration and exploitation of the solution space and hence better optimization performance.

4.2 Assimilation Coefficient Parameter

The second imperialist competitive algorithm (ICA) parameter to be tuned was the assimilation coefficient, which plays a significant role during the assimilation process when colonies move toward their imperialists. Based on a survey of the literature, the recommended value for the assimilation coefficient is 2 (Atashpaz-Gargari et. al., 2007; Hosseni et. al., 2014). Following the same methodology used for the revolution ratio parameter, we examined five different assimilation coefficient values (0.5, 2, 3, 4, and 5) and performed optimization runs for each value five times for the single vertical well case. We then compared and ranked the parameter sensitivity test results based on the total average objective values, as shown in **Figs. 21** and in **Table 7**. Based on the below figure and table, we found the best assimilation coefficient value to be equal to 0.5.

Ranking	Value
1	0.5
2	2
3	4
4	5
5	3

Table 7: Rankings of assimilation parameters

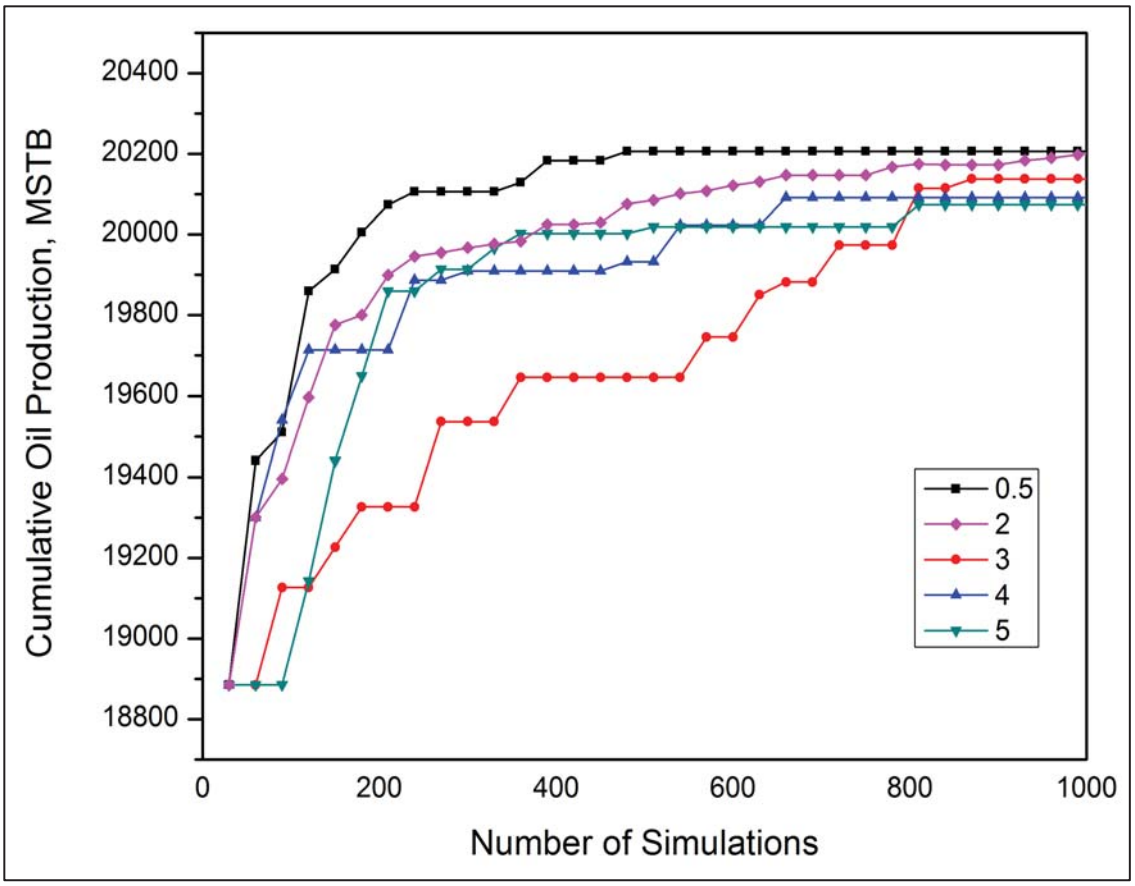

Figure 21: ICA performance with five different assimilation coefficients for case 1

4.3 Assimilation Angle Parameter

The last imperialist competitive algorithm (ICA) parameter we considered in this sensitivity analysis was the assimilation angle, which works as a deviation in the assimilation process to ensure a broad search space and hence greater search diversification. Based on a survey of the literature, the recommended value for the assimilation angle is $\pi/4$ (Atashpaz-Gargari et. al., 2007; Hosseni et. al., 2014). We examined five different assimilation angles (in radians) (0.3, 0.4, 0.5, 0.6, and 0.7) and

performed optimization runs for each value five times for the single vertical case. We then compared and ranked the parameter sensitivity test results based on the total average objective values, as shown in **Figs. 22** and in **Table 8**.

Ranking	Value
1	0.4
2	0.5
3	0.7
4	0.6
5	0.3

Table 8: Rankings of assimilation angle parameters

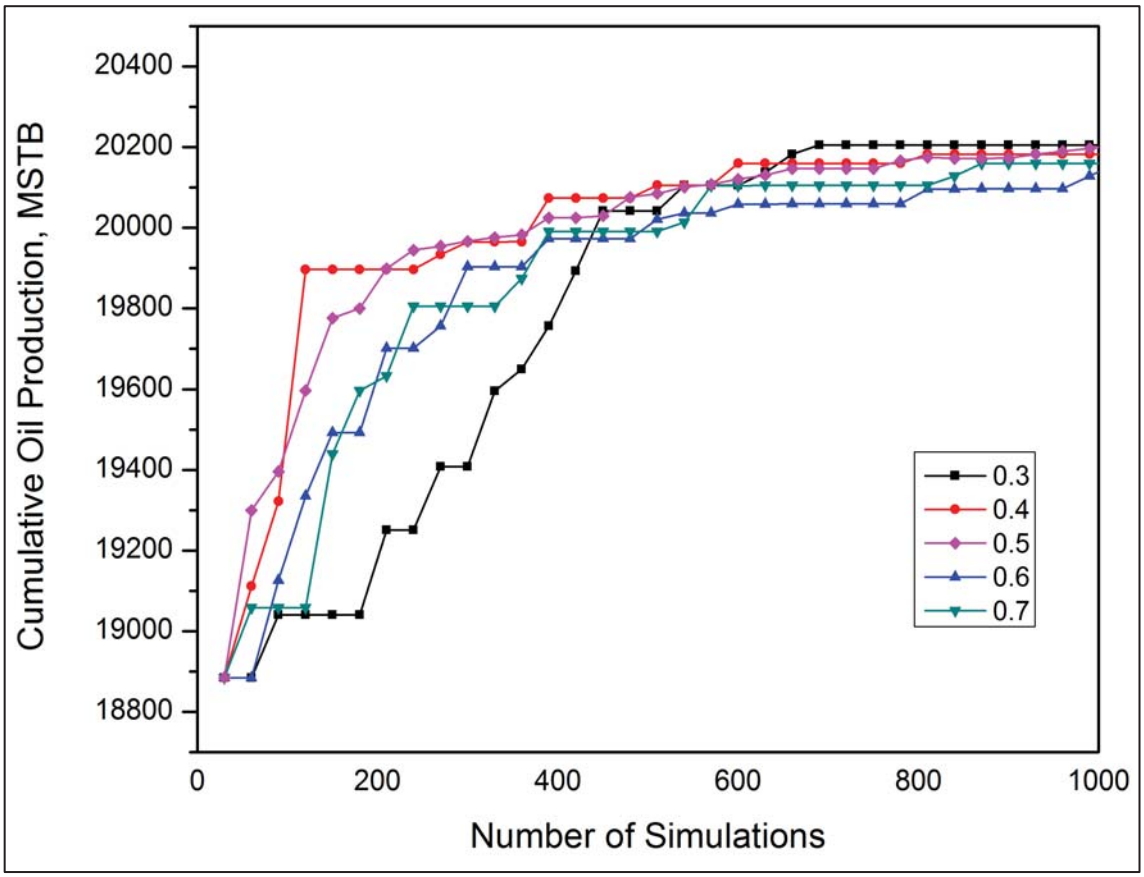

Figure 22: ICA performance with five different assimilation angles for case 1

The best assimilation angle value was found to be 0.4. Also, by examining the above figures, we note that most of the differences between the sensitivity runs occurred in the first 400 simulation runs after which the solutions tended to converge, and the differences between the last objective functions are considered very small.

5. APPLYING ICA TO A REAL FIELD EXAMPLE

After applying imperialist competitive algorithm (ICA) to multiple synthetic model cases and given the fact that imperialist competitive algorithm (ICA) generally performed better than particle swarm optimization (PSO) and genetic algorithm (GA) in these examples, the next step is to investigate and analyze imperialist competitive algorithm (ICA) performance with respect to a more complicated optimization problem that deals with real field data and more optimization variables, such as well configuration (vertical/directional), well type (producer/injector), number of wells, and drilling schedule.

5.1 Problem Description and Approach

In this part of the study, our objective was to optimize the placement of a maximum of ten wells to maximize the field production net present value (NPV). We further complicated the problem by having the imperialist competitive algorithm (ICA) optimization algorithm find the best optimization variable combination of well type (producer/injector), well configuration (directional/vertical), well position, and well drilling schedule (drill/no-drill). In this section, we first describe the reservoir model we used. Then, we detail the objective function with respect to the chosen economic parameters. Lastly, we present a detailed description of how the imperialist competitive algorithm (ICA) algorithm interacts with the commercial simulator (CMG) to generate an optimized solution.

5.2 Reservoir Model Description

The model we used for this task is specific to a sector of a Middle Eastern onshore carbonate reservoir that we hereafter refer to as ME1. ME1 extends over a 4.5 km by 1.6 km area and represents half of an anticlined structure, as shown in **Fig. 23**. The field has three existing flank water injectors and eight producers that have been in production for a total of five years. All wells are perforated in all five layers. We will continue to use these control constraints for new infill wells subject to the well placement optimization results of this study. The reservoir consists of five layers that yield a total thickness of 160 ft. The average permeability of layers 1, 4, and 5 ranges from 5 to 20 md, whereas the average permeability of layers 2 and 3 range from 150 to 600 md, as shown in **Table 9**. The porosity is correlated to the horizontal permeability, which we calculated using Equation 11. We discretized the model by 49 × 40 × 5 for a total of 9,800 grid blocks. This is a three-phase model with an initial pressure of 3,410 psi and undersaturated fluid with a bubble point pressure of 2,533 psi. The relative permeability curve corresponds to that of a single rock type, as shown in **Fig. 24**, with no capillary forces assumption. **Table 10** provides a summary of other reservoir model parameters.

$$\emptyset_i = 11.663 + 2.189\ \log(k_H) \tag{11}$$

All existing and future wells considered in this model are subject to production and injection control constraints. Producers are subject to a minimum bottomhole

pressure of 1,000 psi and a maximum liquid production rate of 15,000 STB/d, as well as control constraints. Injectors are subject to a maximum injection pressure of 3,500 psi and a maximum injection rate of 15,000 STB/d as well control constraints. **Table 11** shows the injector and producer well control constraints.

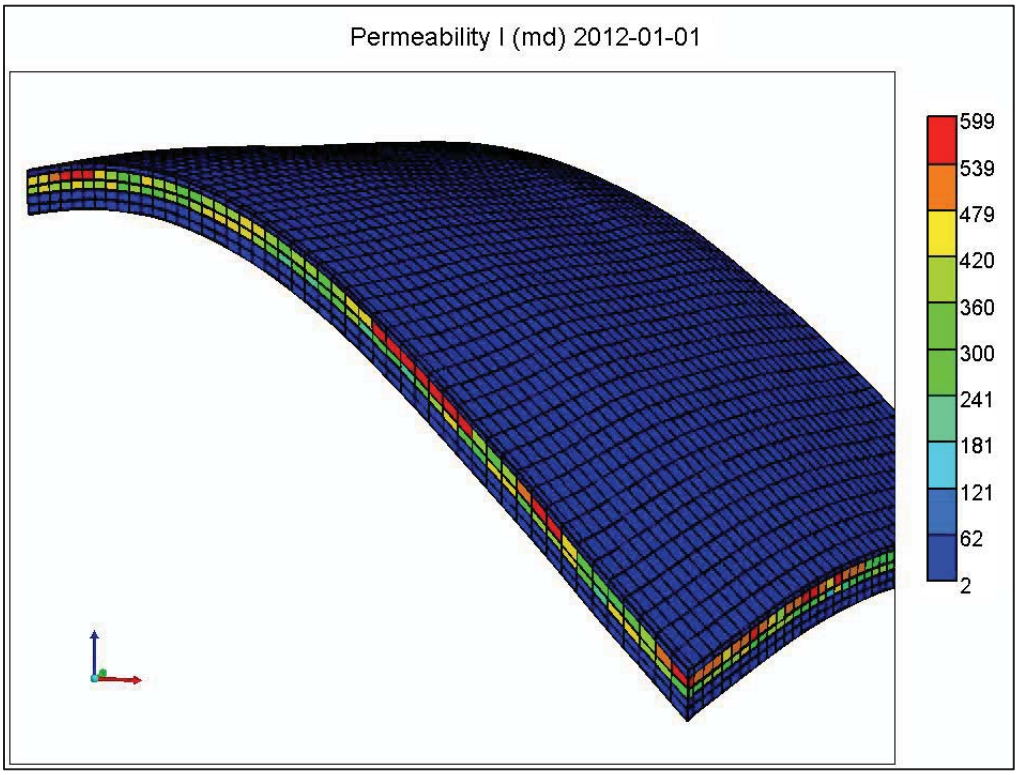

Figure 23: 3D view of ME1 reservoir model

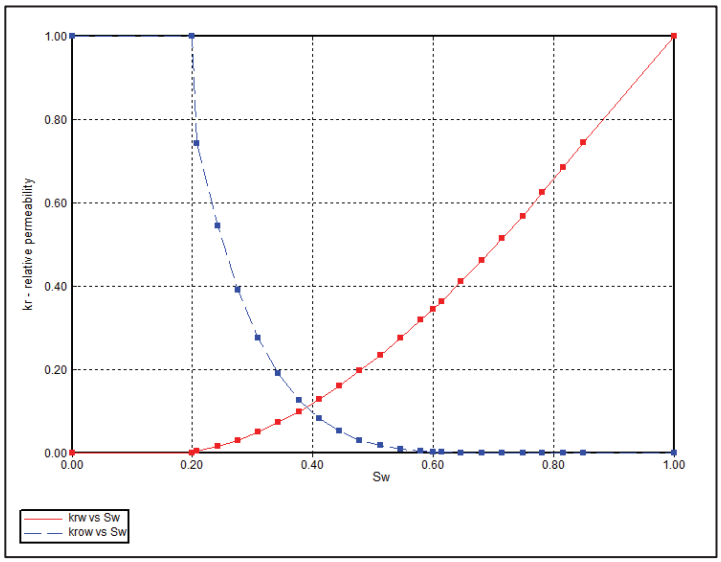

Figure 24: Relative permeability curve

Layer	Minimum (md)	Maximum (md)
1	5	20
2	250	600
3	150	450
4	2	10
5	2	10

Table 9: Permeability ranges of the ME1 layers

Parameter	Value
Grid size	$49 \times 40 \times 5$
Grid cell dimension	$300 \times 120 \times 32$ (ft)
Initial pressure	3,410 psi
Rock compressibility	0.5×10^{-5} psi^{-1}
Oil density	52.36 Ibm/ft^3
Gas density	0.061 Ibm/ft^3
Water density	71.85 Ibm/ft^3

Table 10: Reservoir model parameters

Control Type	Value
Minimum bottom hole pressure (producer)	1000 psi
Maximum liquid production rate (producer)	15,000 STB/d
Maximum injection pressure (injector)	3,500 psi
Maximum injection rate (injector)	15,000 STB/d

Table 11: Injectors and producers well control constraints

5.3 Real Field Infill Drilling Optimization

In this example, our objective is to optimize the placement of multiple producers and injectors following a six-month drilling schedule for an existing field that has been producing for five years. To implement this task, the proposed optimization variables include: a) the well's midpoint 3D coordinates, b) total well length, c) vertical well distance between heel and toe, and d) the decision to drill or not (producer/injector) every six months. The criteria for choosing these variables are that they: a) represent the most critical and important well parameters that have the greatest impact on the desired objective function (NPV), and b) are independent to reduce the number of required variables. In this study, we established variable settings similar to those introduced by Farshi (2008). The wells to be optimized are infill monobore producer/injector wells that are drilled every six months. **Fig. 25** illustrates the reservoir production time, including the optimization infill drilling period.

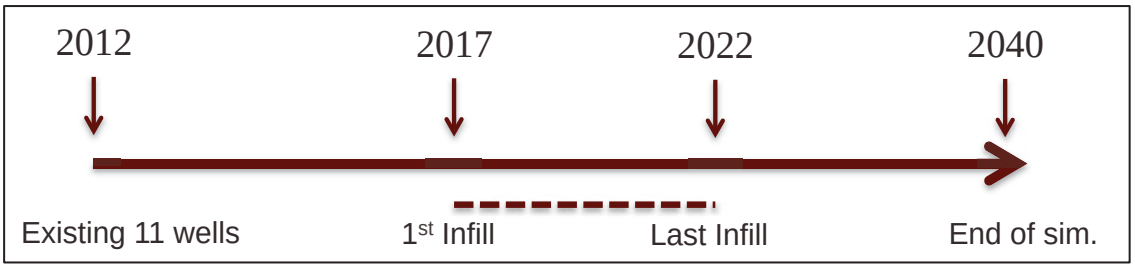

Figure 25: Timeline for major field events

5.4 Methodology For Code Implementation

To fully represent the wellbore trajectories, six variables are necessary, including the well's midpoint 3D coordinates (x_{mid}, y_{mid}, z_{mid}), the total length of the wellbore (L_{total}), the x-y rotation angle (θ), and the vertical distance between the well's heel and toe (L_z), which we use instead of the vertical azimuth angle as it is more easily controlled with respect to the vertical reservoir model constraints. **Fig. 26** depicts how a well is modeled. In addition to these six variables, we added an extra variable regarding the decision/type of the well (D). The D variable can be assigned one of three values (0,1,2), which represent (no-drill, producer, injector). Following this methodology, we can represent any well using just seven variables. With these seven variables, we can also generate other dependent parameters, such as the heel and toe coordinates, and we can then feed them, and the D variable to the simulator input file to initiate the simulation. A summary of variable transformation equations are provided below:

$$x_{heel,toe} = x_{middle}{}^{+}_{-} \frac{1}{2} L_{xy} \sin\theta \tag{12}$$

$$y_{heel,toe} = y_{middle}{}^{+}_{-} \frac{1}{2} L_{xy} \cos\theta \tag{13}$$

$$z_{heel,toe} = z_{middle}{}^{+}_{-} \frac{1}{2} L_{z} \tag{14}$$

$$L_{xy} = \sqrt{L_{total}^2 - L_z^2} \quad (15)$$

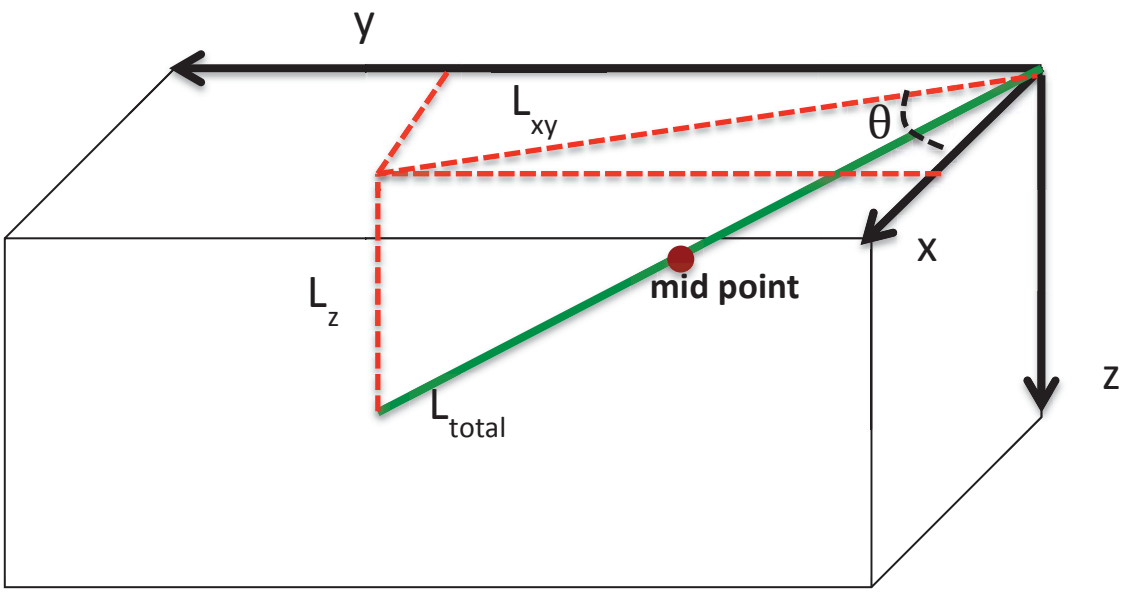

Figure 26: Well modeling representation in a 3D view

By utilizing the abovementioned seven variables, we can generate a full country representation, as used in the imperialist competitive algorithm (ICA), as follows:

Country=[$(x_{middle},y_{middle},z_{middle},L_{total},L_z,\theta, D)_{well1},..,(x_{middle},y_{middle},z_{middle},L_{total},L_z,\theta, D)_{welln}$]

Also, since the imperialist competitive algorithm (ICA) algorithm code interacts with the CMG, we must fix from the beginning the maximum possible number of wells to be considered in the optimization. This step is necessary to avoid complexity when overwriting the simulator input file in each simulation run. In this study, we considered a

maximum number of 10 wells. Hence, the total number of optimization variable for each country is as follows:

Number of variables $_{(country)}$ = Number of wells × 7 = Total of 70 variables

The code we used in this work is a modified version of that developed by Esmaeil Atashpaz Gargari (https://www.mathworks.com/matlabcentral/fileexchange/22046-imperialist-competitive-algorithm--ica-). Although the imperialist competitive algorithm (ICA) code developed by Esmaeil is typically used for solving mathematical functions, in order to use the imperialist competitive algorithm (ICA) code for our well placement optimization problem, a substantial modification was necessary. We applied most of these modifications to the *Main_ImperialistCompetitveAlgorithm.m, GenerateNewCountry.m, and CostFunction.m* files.

The *Main_ImperialistCompetitveAlgorithm.m* file is the MATLAB file responsible for the implementation of the algorithm. In this file, the user enters the number of optimization variables, the maximum and minimum limits of the optimization variable values, the number of countries, the number of initial imperialists, the assimilation coefficient, the assimilation angle coefficient, the revolution parameter, and the maximum number of iterations. These pre-defined parameters are then fed to the GenerateNewCountry.m.

In the GenerateNewCountry.m file, the countries are created, and the code uses Equations 11, 12, 13, and 14 to generate the initial populations of the countries. This file

also implements a filter to check that the variables of all countries adhere to the lower and upper limits of the optimization variables. For example, if a variable exceeds the upper limit, its value will be set back to the maximum value of that specific variable. The outcome of the GenerateNewCountry.m file is a matrix with the size $Number_{countries} \times Number_{variables}$. The MATLAB code for this file is presented below.

```
function NewCountry = GenerateNewCountry(NumOfCountries,ProblemParams)


    VarMinMatrix = repmat(ProblemParams.VarMin,NumOfCountries,1);
    VarMaxMatrix = repmat(ProblemParams.VarMax,NumOfCountries,1);
    Country = round((VarMaxMatrix - VarMinMatrix) .* rand(size(VarMinMatrix)) +
VarMinMatrix);
    angle = Country(:,[6:7:end]).*(pi/180);
   Country(:,6:7:end)=angle;
    Ang=angle;
    %calculating Lxy
    Ltot=Country(:,4:7:end);
    Lz=Country(:,5:7:end);
    Lxy=round((Ltot.^2-Lz.^2).^0.5);
    %calculating x heels
    Ncol=size(Country,2);
    Nwell=Ncol/7;
```

```
xmid=Country(:,1:7:end);
xh=xmid-round((Lxy.*sin(Ang)).*0.5);
xt=xmid+round((Lxy.*cos(Ang)).*0.5);
xh(xh<5)=5;
xh(xh>35)=35;
xt(xt<5)=5;
xt(xt>35)=35;
% calculating y heels
ymid=Country(:,2:7:end);
yh=ymid-round((Lxy.*sin(Ang)).*0.5);
yt=ymid+round((Lxy.*cos(Ang)).*0.5);
yh(yh<5)=5;
yh(yh>35)=35;
yt(yt<5)=5;
yt(yt>35)=35;
% calculating zheels
zmid=Country(:,3:7:end);
zh=round(zmid-(Lz.*0.5));
zt=round(zmid+(Lz.*0.5));
zh(zh<1)=1;
zh(zh>7)=7;
zt(zt<1)=1;
```

```
    zt(zt>7)=7;
    D=Country(:,7:7:end);
    %xh=Country(:,1:6:end)-times(Country(:,4:6:end),sin((Country(:,6:6:end))));
    Ncol=size(Country,2);
    Nwell=Ncol/7;
    for i=1:Nwell
    Well{i}=[xh(:,i) yh(:,i) zh(:,i) xt(:,i) yt(:,i) zt(:,i) D(:,i)];
    end
    NewCountry=cell2mat(Well);
end
```

The cost function file is considered to be the soul of our modified imperialist competitive algorithm (ICA). In this file, the imperialist competitive algorithm (ICA) input parameters are conditioned to be made compatible for being run by CMG. After the CMG input file is created, it is fed to the CMG execution file by calling a batch file, which runs the model simulation to produce the CMG output file. Then, the code reads the cumulative oil, gas, produced water, and injected water for each year and uses these values to calculate the expected net present value (NPV). This process is repeated for all countries for the five realizations and then the countries are ranked based on their objective function values. The most difficult part of the code implementation is conditioning the CMG input file to make it readable by the CMG. Since our optimization

problem formulation consists of varying the number and length of the wells, we had to ensure that the code would seamlessly loop through the full iterations. We did so by fixing the maximum number of wells to be considered, in our work this was ten wells, and fixing the maximum length a well could have. Below, we describe the main practical implementations we used to overcome the abovementioned difficulties. We have included only a part of the code of one well input file conditioning for illustration purposes. For more code details, please refer to Appendix B.

```
%1st well input file modifications
S=floor(x(ii,[7:7:end]));
Nactive=nnz(S);
point11=x(ii,1:3);
point21=x(ii,4:6);
t=0:.1:1;
C=repmat(point11,length(t),1)';
Cbeforefinal=(point21-point11)'*t;
Cfinal=floor(C+Cbeforefinal)';
perf=unique(Cfinal,'rows','stable');
aa=size(perf,1);
bb=12-aa; % need change if you have different max number of perforations
filename = sprintf('%s%02d%s','realization',jj,'.dat'); %naming the file
fid = fopen(filename, 'r+');
```

```
Type1=x(ii,7);
if Type1==1 || Type1==0
   D='PRODUCER         ';
   E='**COMP  WATER';
   F='OPERATE  MIN  BHP  1000.0  CONT';
   O='OPERATE  MAX  STL  15000.0  CONT';
   G='FLOW-TO  ';
elseif Type1==2
   D='INJECTOR MOBWEIGHT';
   E='INCOMP  WATER';
   F='OPERATE  MAX  BHP  3500.0  CONT';
   O='OPERATE  MAX  STW  15000.0  CONT';
   G='FLOW-FROM';
end
%need for fseek and idx;
ww=fscanf(fid,'%c',Inf);
idx1 = strfind(ww, '**wl1');
fseek(fid, idx1+4, 'bof');
fprintf(fid,  '\r\nDATE  2017  1  1\r\nWELL  "Well01"\r\n%s
"Well01"\r\n%s\r\n%s\r\n%s\r\nGEOMETRY  K  0.25  0.37  0.25  0.0\r\nPERF
GEOA  "Well01"',D,E,F,O);
fprintf(fid, '\r\n %02d %02d %02d  1.0  OPEN   %s "SURFACE"', perf(1,:),G);
```

```
for i=2:aa
   fprintf(fid, '\r\n %02d %02d %02d  1.0  OPEN    %s  %02d', perf(i,:),G,i-1);
end


for i=1:bb
   fprintf(fid,'\r\n ************************************');
end
if Type1==0
   H='SHUTIN "Well01"';
else
   H='OPEN "Well01"  ';
end
fprintf(fid,'\r\n%s',H);
fclose(fid);
 % end of 1st well modifications
```

5.5 Net Present Value Formulation

As stated above, we determined the objective function (J) based on the application of a robust optimization (RO) of the net present value (NPV) in each realization. Hence, defining the economic parameters is crucial for the NPV calculations. In order to find the net present value (NPV) for each potential solution, we ran a simulation using CMG and converted the resulting oil, water, and gas production profiles to dollar values along with the associated wells drilling costs. We calculated the net present value (NPV) for each potential solution as follows:

$$NPV = \sum_{t=1}^{T} \frac{(Q_{t,o}.P_o + Q_{t,g}.P_g + Q_{t,wp}.P_{wp} + Q_{t,wi}.P_{wi})}{(1+r)^t} - C_{Dr} \quad (16)$$

$$C_{Dr} = \sum_{n=1}^{N_t} (C_n + L_n) \quad (17)$$

where T is the total production time in years; $Q_{t,o}$ is the cumulative oil production (STB) at time t; $Q_{t,g}$ is the cumulative gas production (SCF) at time t; $Q_{t,wp}$ is the cumulative produced water (STB) at time t; $Q_{t,wi}$ is the cumulative injected water (STB) at time t; P_o, P_g, P_{wp}, and P_{wi} are the oil, gas, produced water, and injected water prices ($/STB), respectively; r is the annual discount rate; and C_{Dr} is the cost to drill the wells, which consists of the cost to drill the well to the top of the reservoir (C_n) and the cost to drill the mainbore within the reservoir (L_n). We note that C_{Dr} is always taken as the expenditure at t = 0. **Table 12** summarizes the economic parameters we used to determine the net present value (NPV).

Parameter	Value
Well drilling cost	$ 12,500,000
Oil price	$ 50/STB
Gas price	$ 3.5/MSCF
Produced water cost	$ 5/STB
Injected water cost	$ 5/STB
Cost of drilling within the reservoir	$ 1000/ft

Table 12: Economics parameters used in NPV calculations

5.6 Geological Uncertainty

One important aspect to consider when utilizing optimization techniques is the extent of geological uncertainty and how it affects the optimization solution. Typically, geological uncertainties are reduced either by taking additional measurements of the uncertain parameters or by considering different scenarios that capture the parameter uncertainty ranges. One technique used to model uncertainty is the robust optimization (RO) technique. Van Essen et al. (2009) reduced the impact of uncertainty associated with field development by utilizing the concept of robust optimization. In their work, the authors chose a set of geological scenarios that reflect the geological uncertainties of the reservoir and utilized this set when calculating the net present value (NPV).

The robust optimization (RO) objective can be represented in different ways. For example, the most straightforward way is to identify the expected outcomes for a set of geological realizations. Different robust optimization (RO) objectives can take into consideration the variance in the outcomes or the worst case scenario. In our study, we were interested in establishing the expected net present value (J) over a set of five equiprobable geological realizations (θd), which we generated using a Gaussian geostatistical simulation method. This robust optimization (RO) technique takes into account the mean and standard deviation of the outcomes. Our goal is to determine the expected net present value (J) for the optimized variables (x) over the set of geological realizations (θd). Equations 18 and 19 summarize how we calculated the expected net present value (J):

$$J_{RO} = \frac{1}{N_T}\sum_{i=1}^{N_T} J(\vec{x}, \theta_i) - r.\sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}(J(\vec{x}, \theta_i) - \overline{J})\text{^}2} \quad (18)$$

$$\overline{J} = \frac{1}{N_T}\sum_{i=1}^{N_T} J(\vec{x}, \theta_i) \quad (19)$$

5.7 Real Field Example Results

As stated above, our goal in this example is to optimize the field infill drilling by finding an improved combination of producers and injectors that will yield a higher NPV. The field under optimization is an existing field that has eight producers and three injectors and has been operating for a total of five years. We made some assumptions in this case. First, we assumed that only one rig is available for drilling, that the drilling

time for drilling one well is six months, and that the rig is available only for 5 years. In other words, this rig is available to drill a well only every six months. Second, any new well will follow the same production and injection constraints of existing wells. Third, the minimum distance between any new and existing well is 500 feet. We set the starting time for drilling infill wells to January 1st, 2017 and the ending time to December 31st, 2022. The end of simulation time is January 1st, 2040.

Since we have seven variables representing each well and we assume that we can establish up to a maximum of ten wells, the total number of optimization variables is seventy. In this example, we use a total forty initial countries and five geological realizations. This equates to two hundred simulations for one algorithm iteration (decade). **Table 13** lists the imperialist competitive algorithm (ICA) parameters used in this example.

ICA Parameter	Value
Number of Countries	40
Assimiliation Coefficient	2
Assimiliation Angle	0.5
Revolution Ratio	0.3
Number of Generations	40

Table 13: ICA parameters used for this example

Fig. 27 shows the results of three optimization runs along with the average of the three. The highest yielding expected net present value (J) corresponds to the Opt 3 run, which yielded a total of $\$7.42 \times 10^9$, corresponding to a 4.4% increase from the first iteration. The Opt 3 configuration has one injector and six producers. By examining the Opt 3 well distribution, we found the injector to be more of a vertical well placed in the upper left corner of the reservoir, as shown in **Fig. 28**. This might be considered to be non-intuitive as we had expected the injector to be located in the lower flank of the reservoir. However, since our optimization problem is concerned with maximizing the NPV based on the abovementioned assumptions, this result is reasonable in light of our goal to maximize oil production for the five years of drilling time (from 2017 to 2022).

To further validate this result, **Figs. 29, 30, 31, 32, and 33** show the saturation profiles of each of the five layers. We can clearly see that layers 2 and 3 are almost 50% swept and the placement of the injector in the upper right corner yielded better sweep.

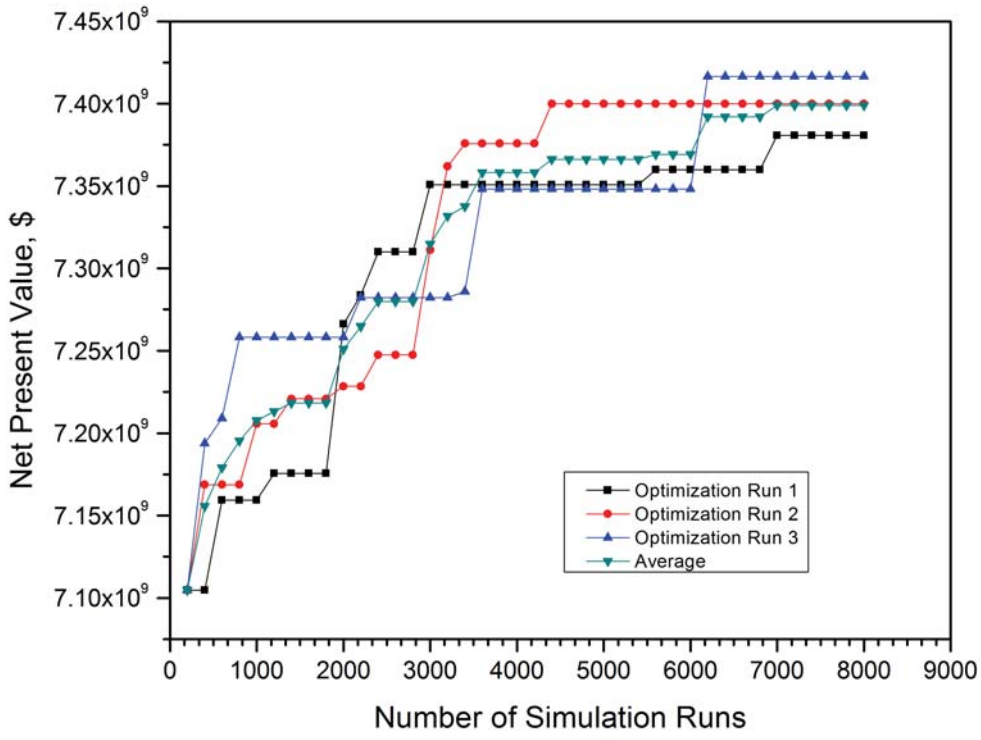

Figure 27: ICA performance of three runs and their average

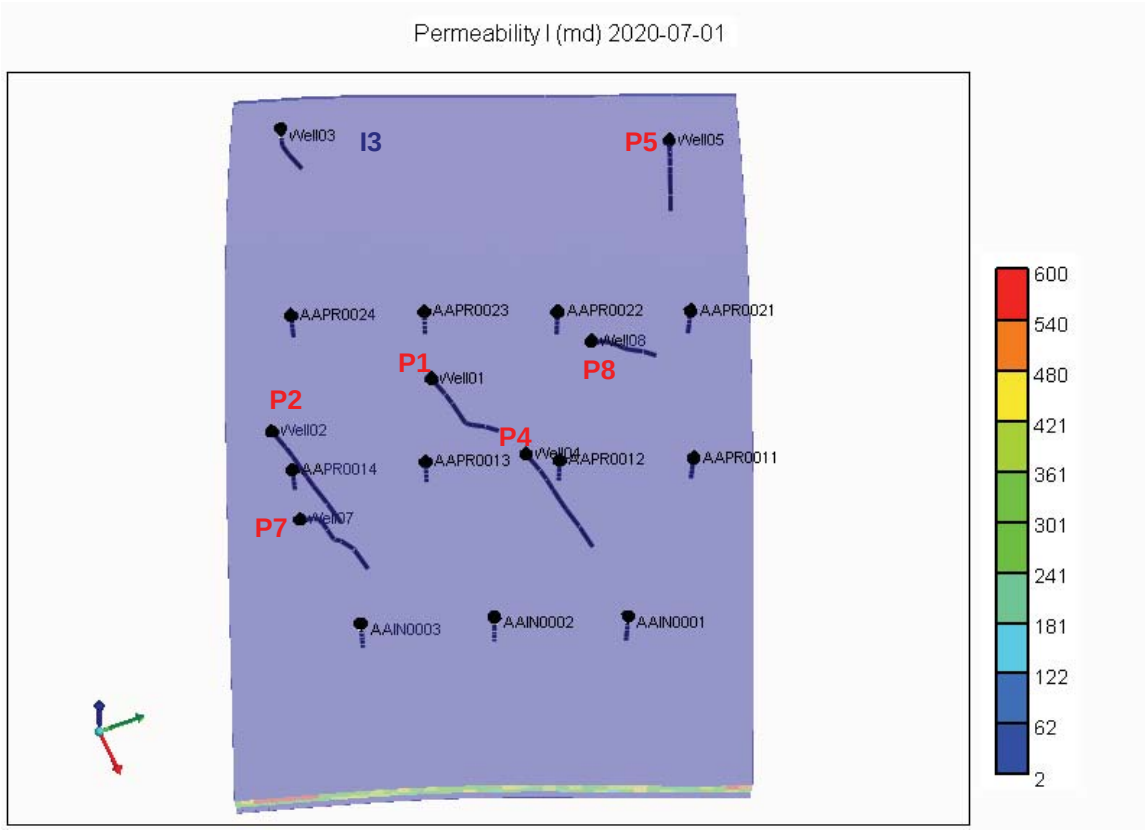

Figure 28: Final well locations and configurations for Opt 3 run. The blue font corresponds to the injector and the red to the producers. The numbers associated with I and P indicate the drilling schedule sequence

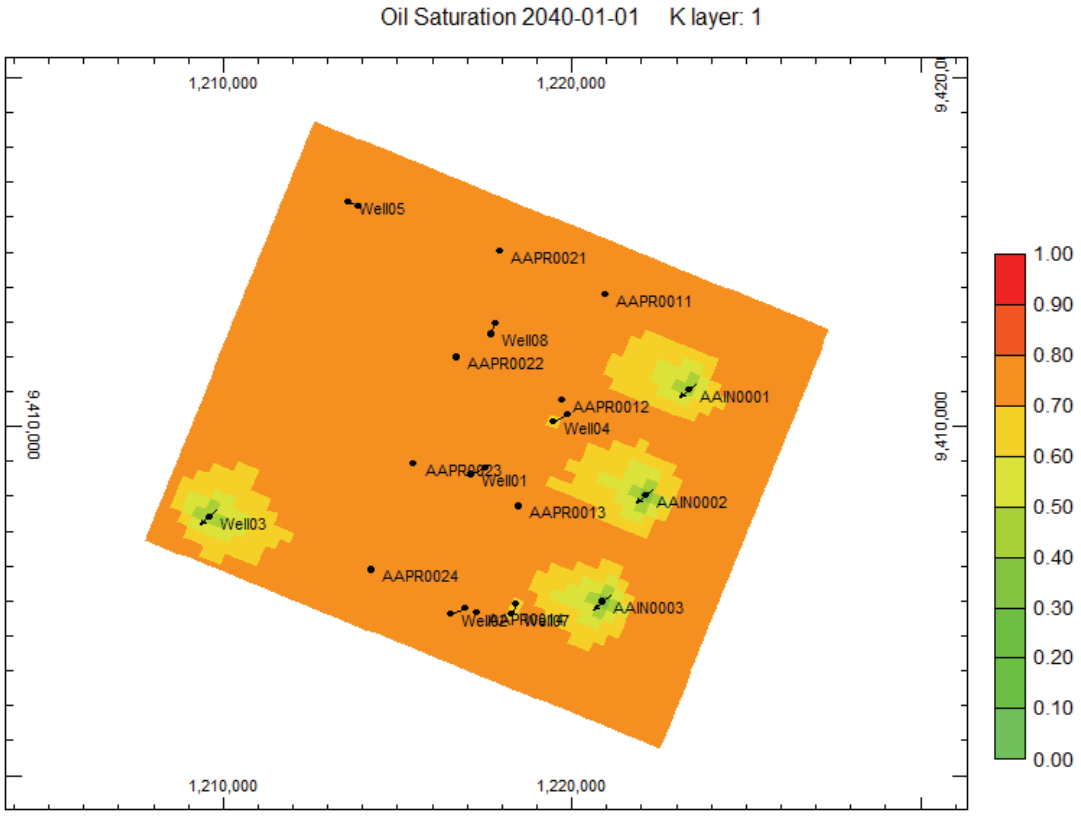

Figure 29: Oil saturation for Opt 3 at the end of the simulation for layer 1

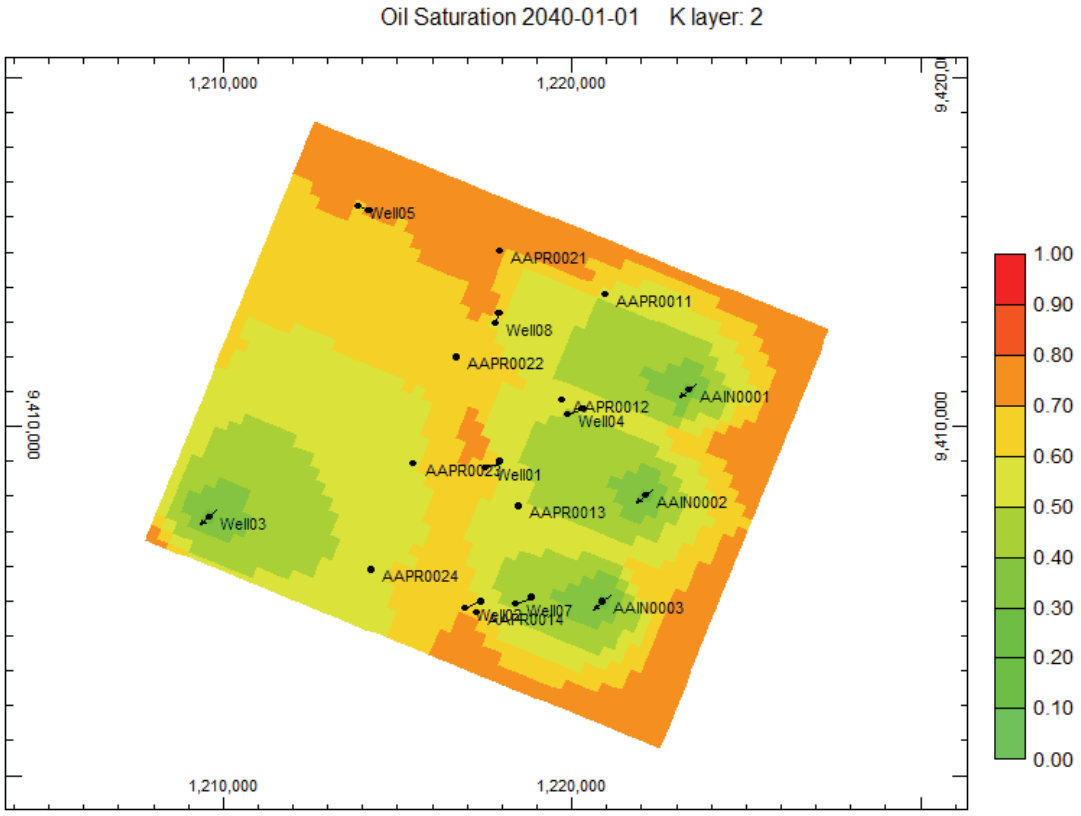

Figure 30: Oil saturation for Opt 3 at the end of the simulation for layer 2

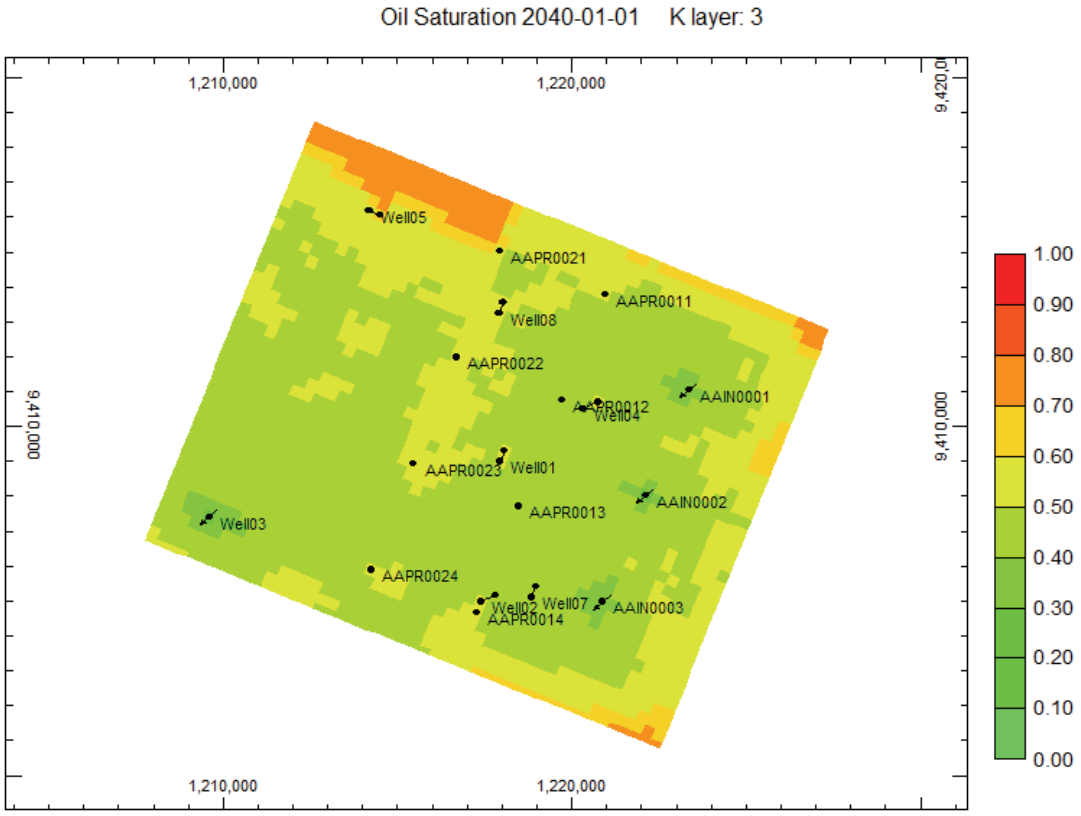

Figure 31: Oil saturation for Opt 3 at the end of the simulation for layer 3

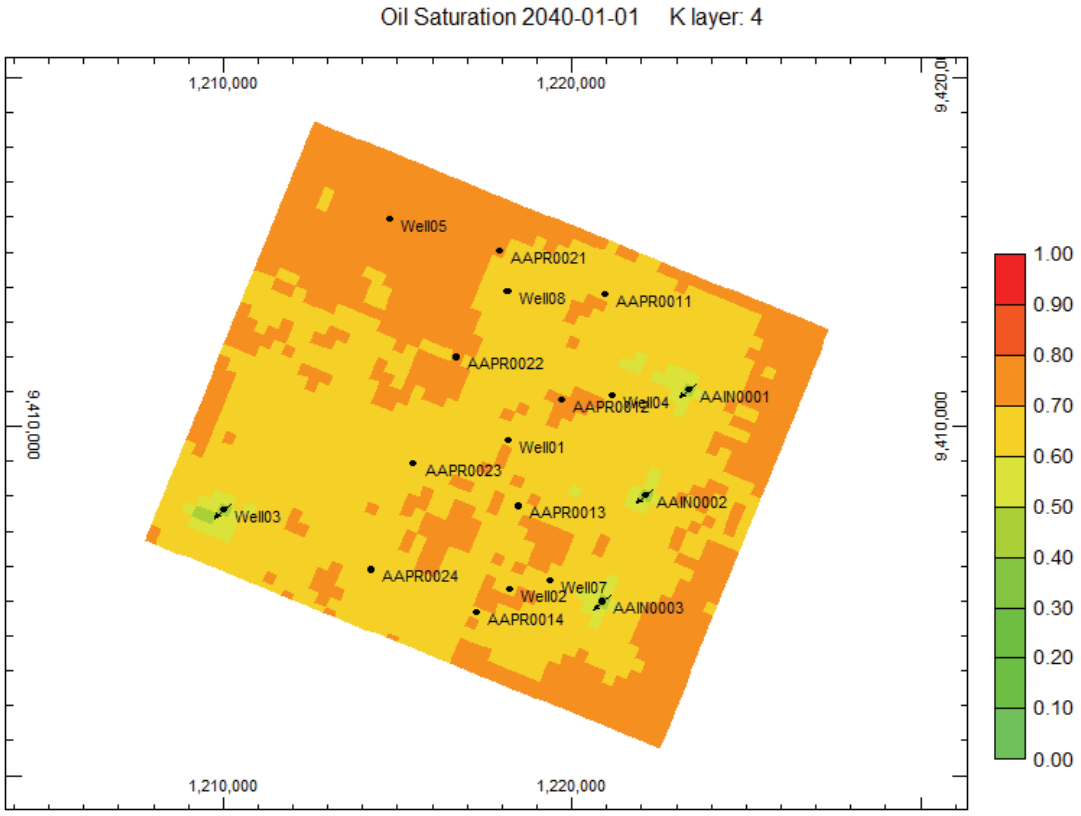

Figure 32: Oil saturation for Opt 3 at the end of the simulation for layer 4

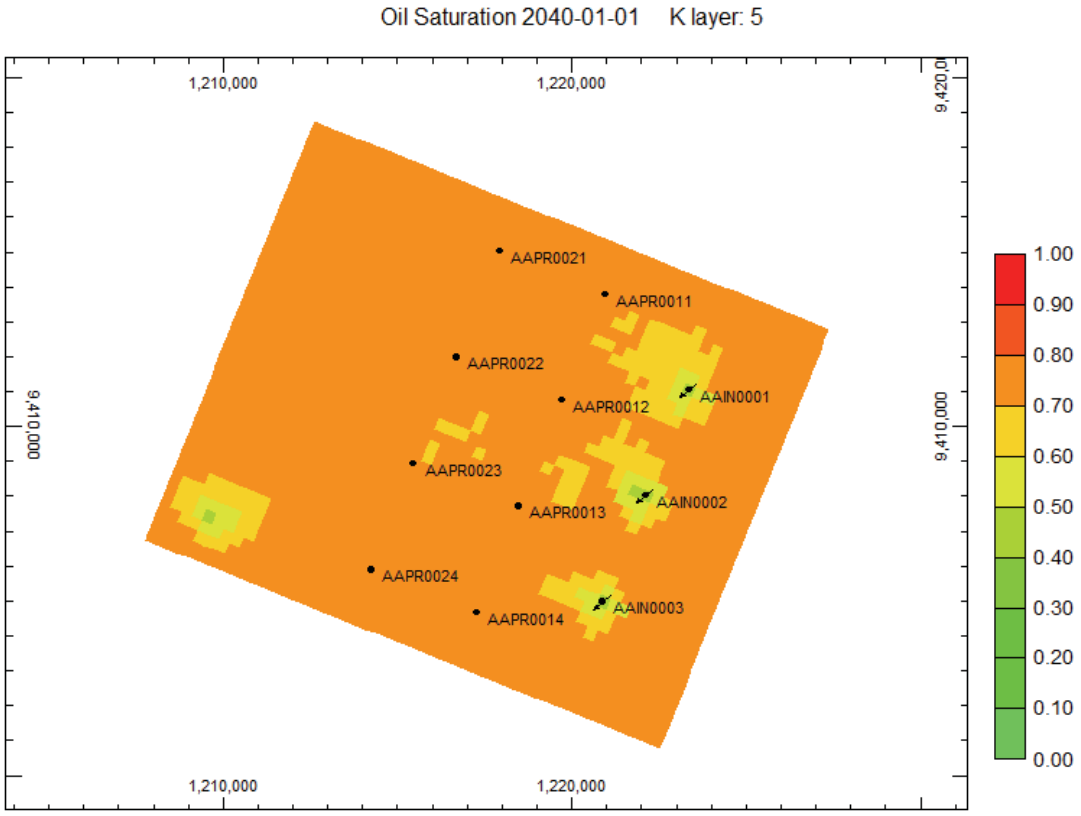

Figure 33: Oil saturation for Opt 3 at the end of the simulation for layer 5

Also, in this part, we examined the imperialist competitive algorithm (ICA) performance in a longer optimization run. In the previous example, we made a total of 8,000 simulation runs. Here, we increase the total number of simulation runs to 20,000 to yield an increase in the total expected net present value (J) of $\$7.53 \times 10^9$, which corresponds to a 5.9% increase from the first iteration, as shown in **Fig. 34**. Also, in this run, two injectors were placed in the upper part of the reservoir, following a pattern similar to that in the previous example. However, the number of producers was reduced to four mainly horizontal wells that are spread around the middle of the reservoir, as shown in **Fig. 35**. All the producers are perforated from layers 1 to 4. **Fig. 36, 37, 38, 39, and 40** show better sweep compared to the previous example

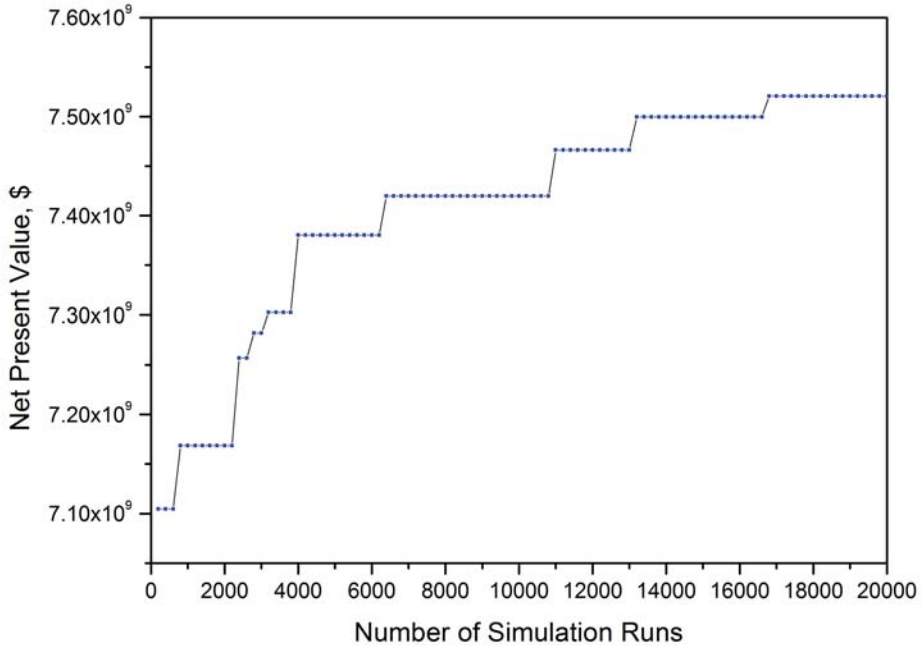

Figure 34: ICA performance for 100 iterations

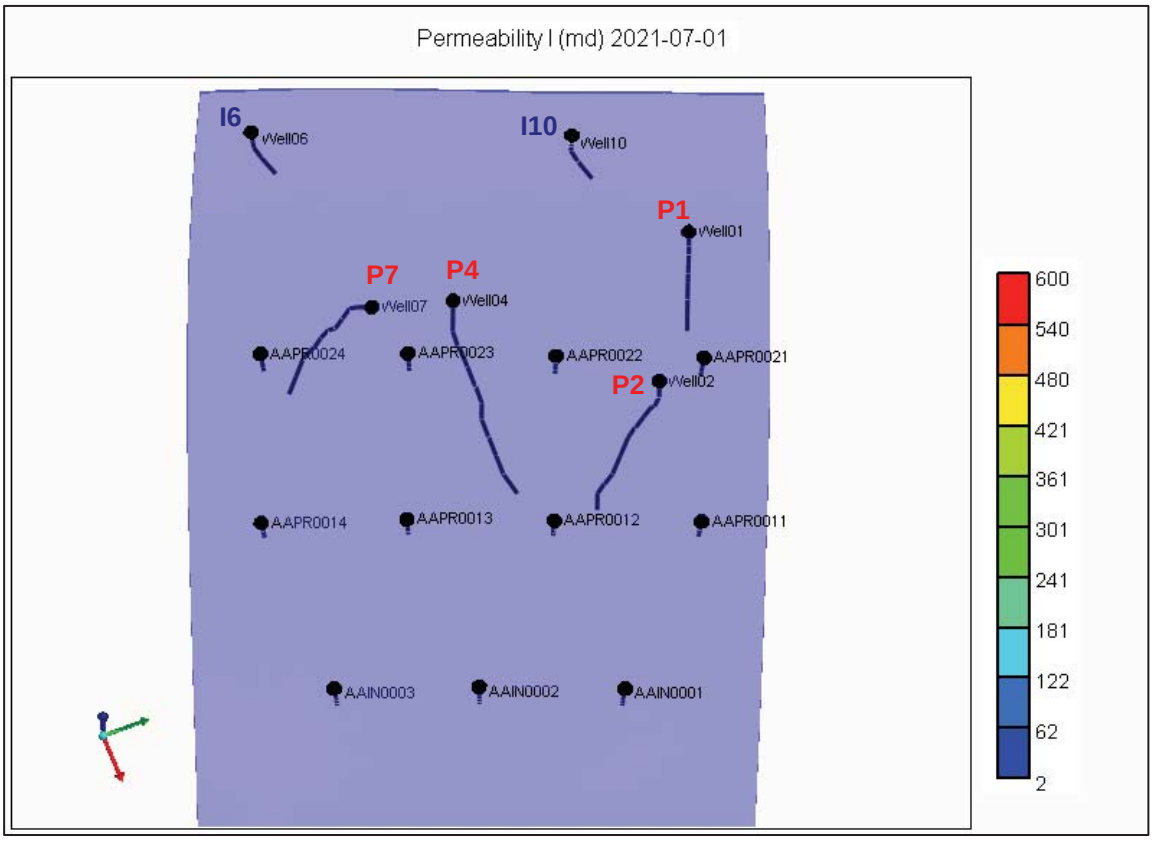

Figure 35: Final well locations and configurations for 100-iteration run. The blue font corresponds to the injector and the red to the producers. The numbers associated with I and P indicate the drilling schedule sequence

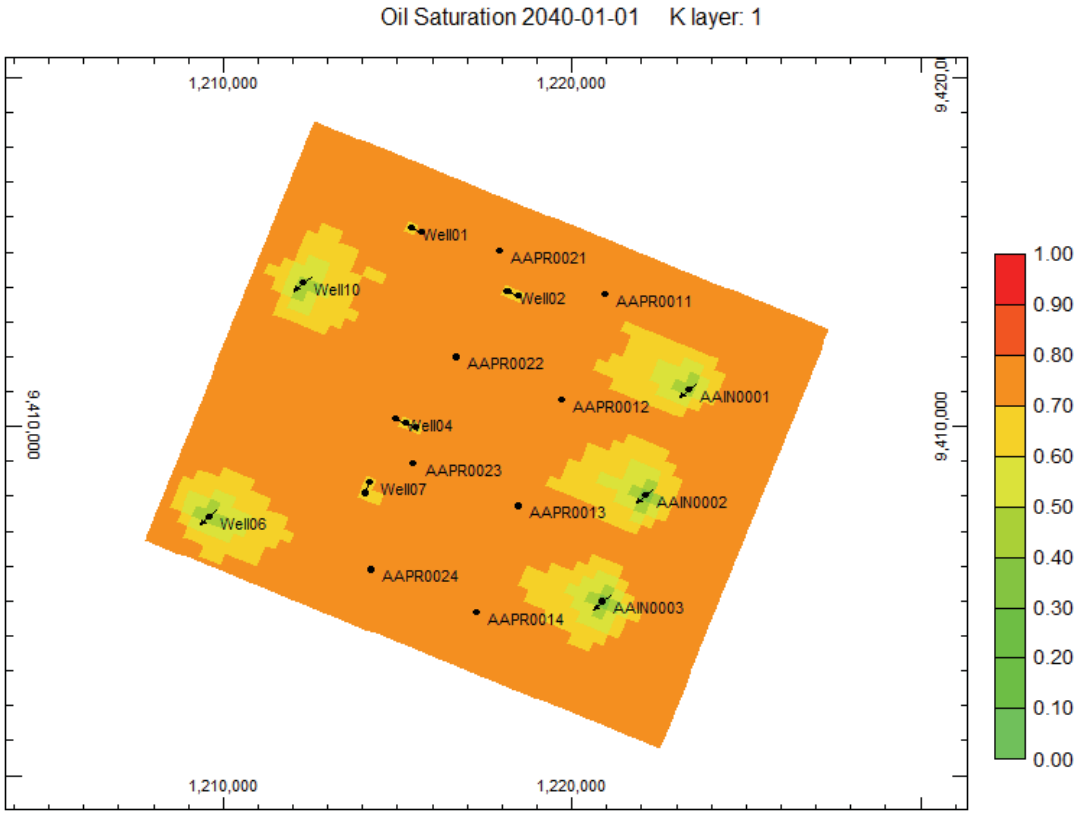

Figure 36: Oil saturation for the 100-iteration run at the end of the simulation for layer 1

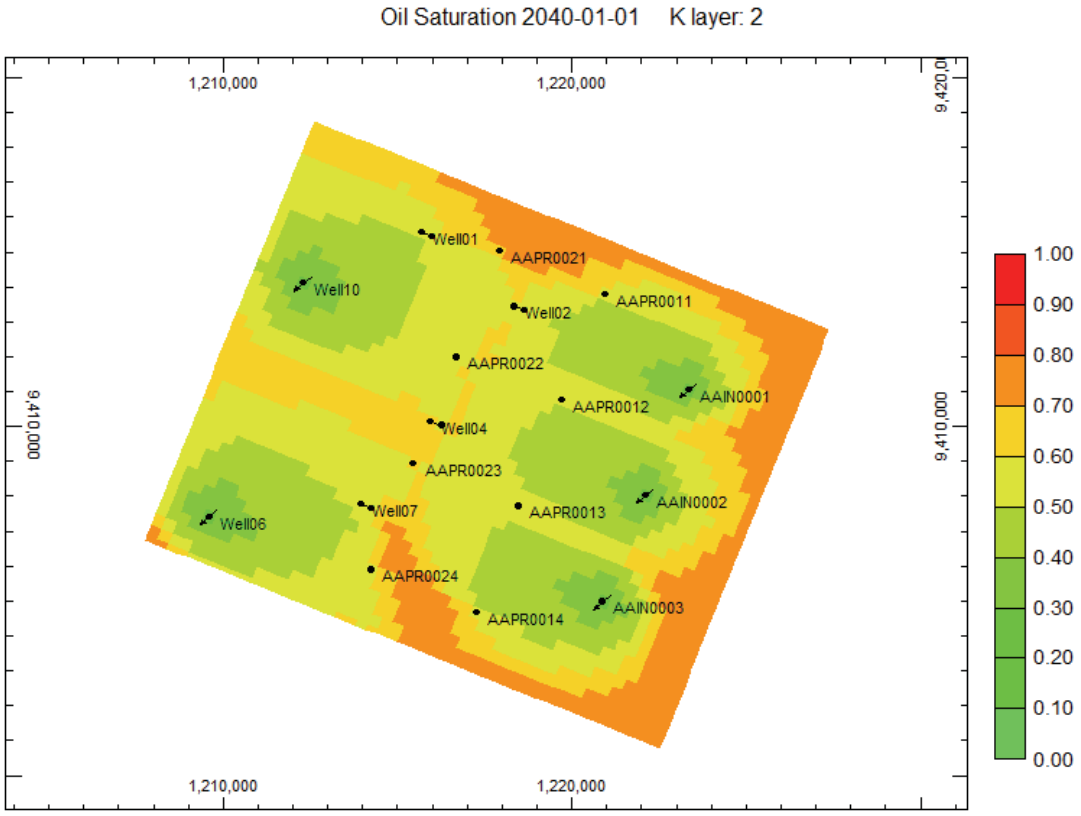

Figure 37: Oil saturation for the 100-iteration run at the end of the simulation for layer 2

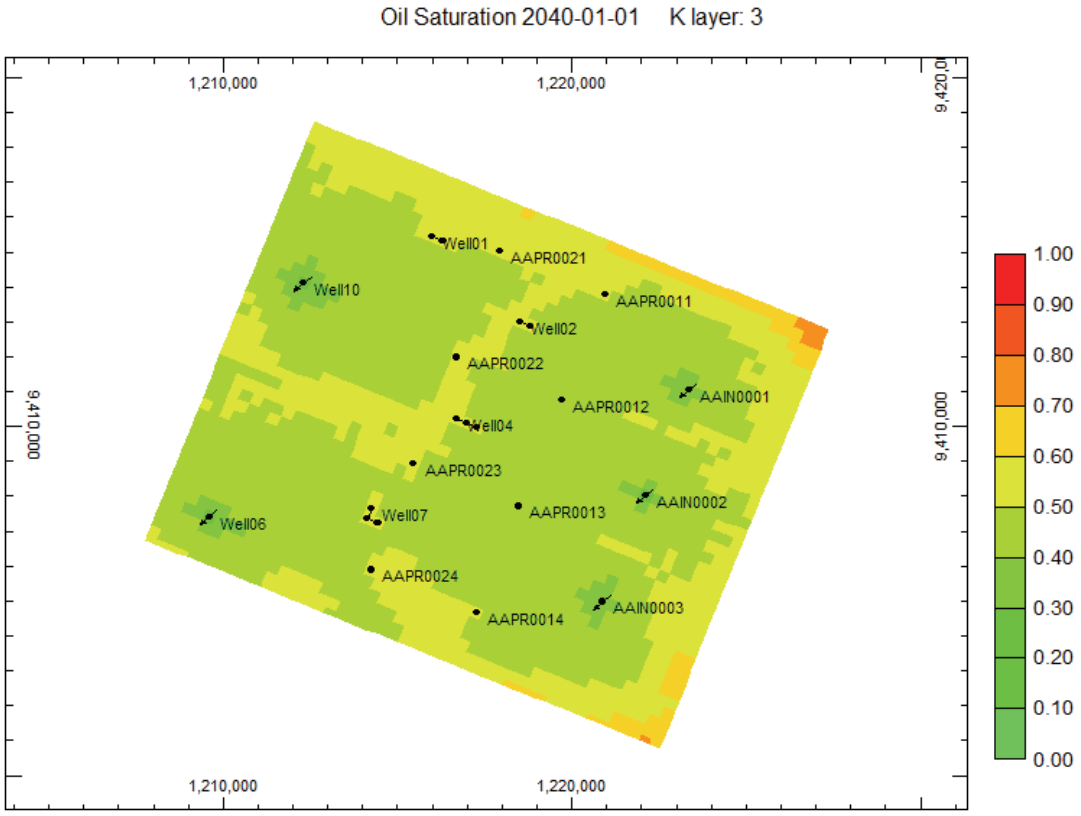

Figure 38: Oil saturation for the 100-iteration run at the end of the simulation for layer 3

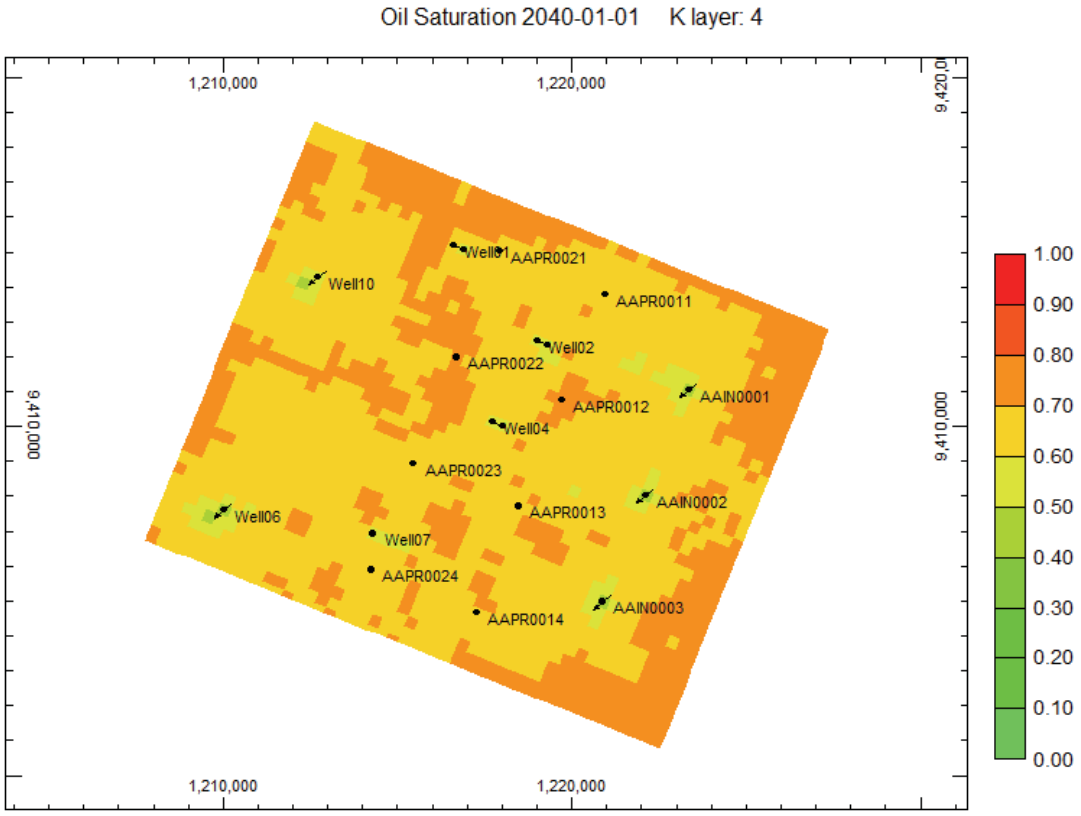

Figure 39: Oil saturation for the 100-iteration run at the end of the simulation for layer 4

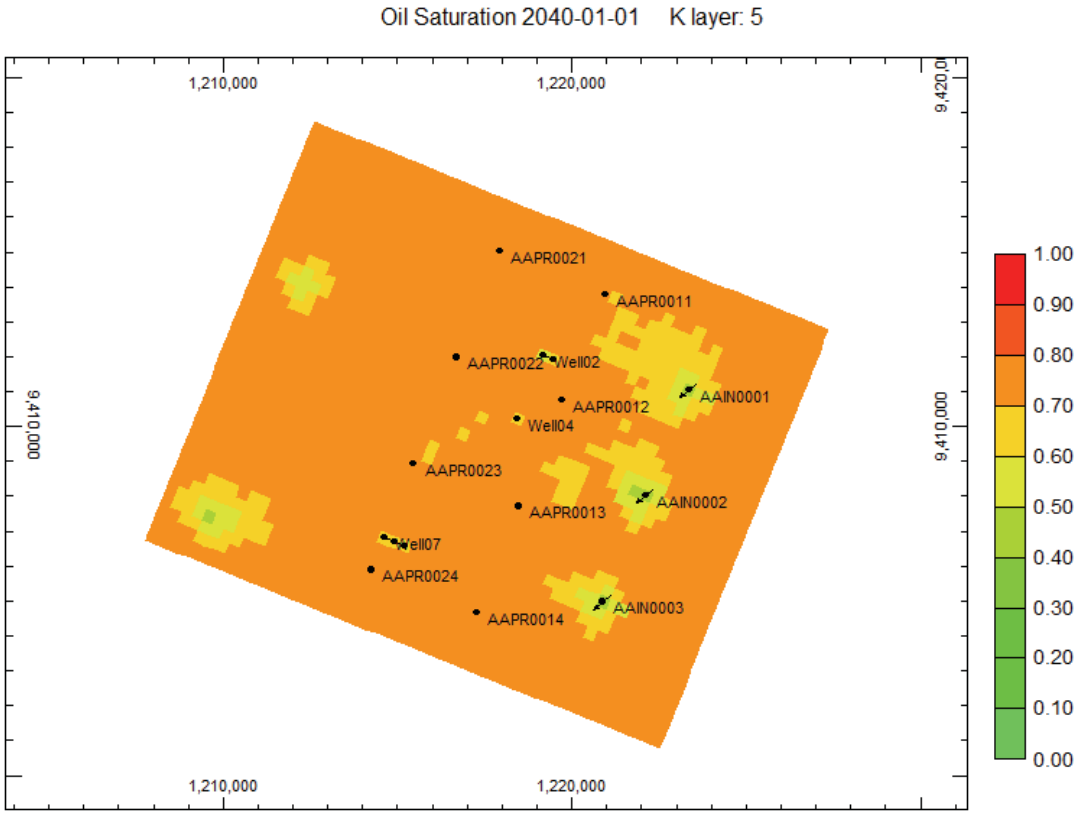

Figure 40: Oil saturation for the 100-iteration run at the end of the simulation for layer 5

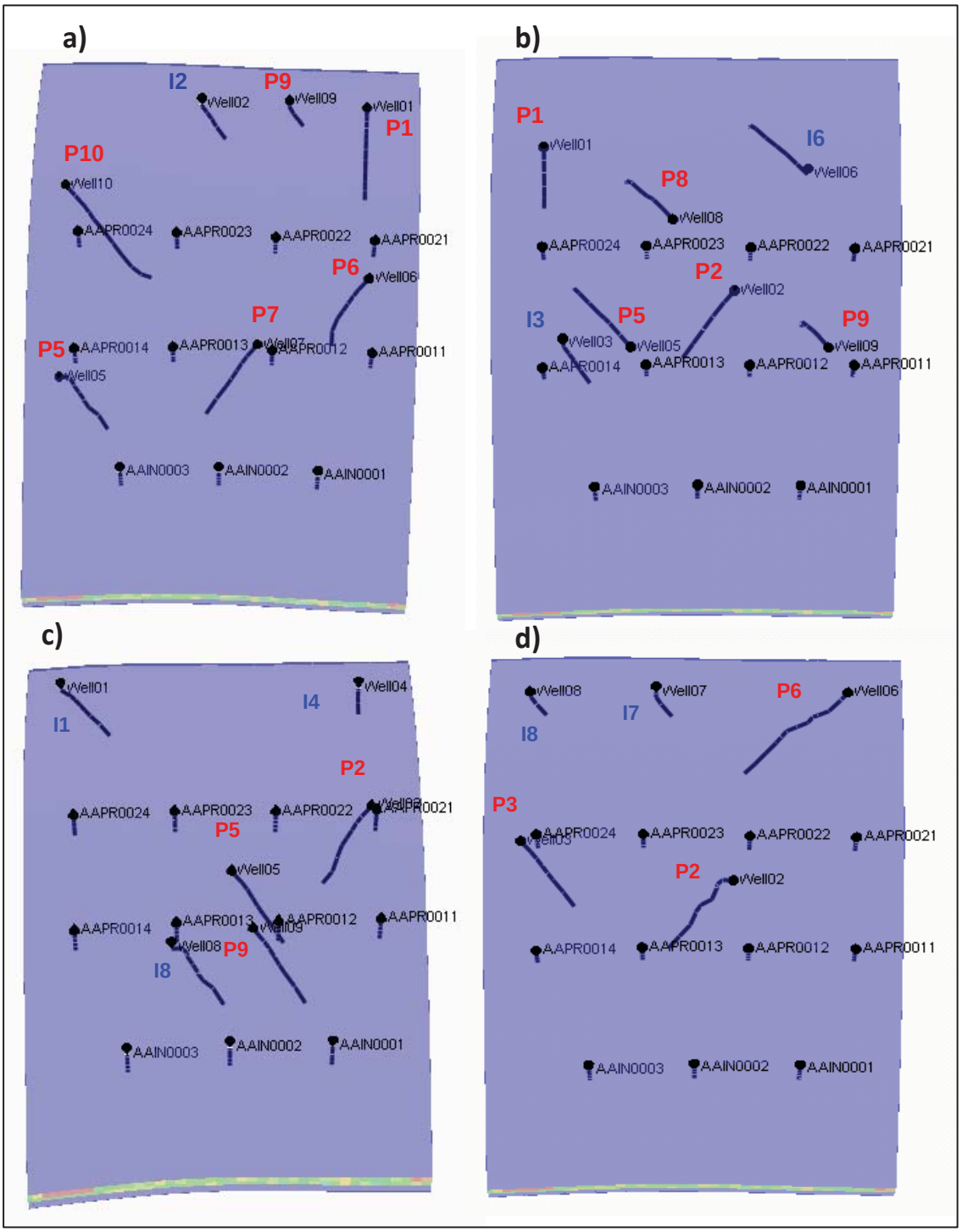

Figure 41: Optimum well configurations at various optimization iterations: a) iteration 20, b) iteration 40, c) iteration 60, and d) iteration 80

As can be seen from **Fig. 41**, at iteration 20 the optimization algorithm suggested the drilling of one injector and seven producers. By closely examining the well configurations and distribution through the field, we can deduce that the wells show an almost equal spacing between new wells and existing ones. Also, the injector is placed at the top of the reservoir away from the center of the reservoir where most of the producer wells are placed. At iteration 40, the optimization algorithm improved the expected net present value (NPV) by placing two injectors and five producers. The wells distribution resulted of this iteration showed increased spacing between new and existing wells. Also, the two injectors were placed far from each other in different reservoir areas which improves oil sweep efficiency in these areas. At iteration 60, the algorithm suggested drilling three injectors and three producers. Here, we can see that total number of wells has been reduced to six wells compared to seven wells. Injectors I1 and I4 were placed at the top of the reservoir that helps push more oil from that area towards the new and existing production wells. Although, injector I8 was placed near the middle of the reservoir next to existing and new wells (P5 and P9), the overall expected net present value (NPV) was improved. This may be attributed to the fact that the other injectors (I1 and I4) were placed far from the existing and new producers that resulted in delayed water breakthroughs in the other producers. At iteration 80, the optimization algorithm suggested an improved expected net present value (NPV) with a total of five wells (two injectors and three producers). At this iteration, we can see that the injectors were placed in the upper part of the reservoir. By examining this iteration's well distribution, it is evident that the injectors are placed away from the producers that delayed water

breakthrough. Also, the new producers (P2, P3 and P6) are placed far from each other. This reduces the interference effect between the producers. At the end of the optimization algorithm, the final wells distribution was similar to the wells distribution at iteration 80 in terms of total number of new wells (five: two injectors and three producers). However, the well distribution at the 100th iteration clearly shows better well spacing between the injectors and the producers. **Fig. 42** shows during the iterations the optimization algorithm kept improving the placement of the wells in the most permeable zones, while placing the injectors in the upper part of the reservoir far away from the existing and the four new producer wells. This resulted in the most optimal solution at iteration 100.

Throughout the optimization process, the algorithm is trying to find a case that maintains reservoir pressure, maximize oil sweep and reduce water cycling while minimizing the number of wells required. The location of injectors seems to be critical to this process. As the number of iterations increases, the idea of placing injectors in mid-field are abandoned and the optimization process seem to favor the top portion of the reservoir for the location of injectors. This makes sense because by injecting water away from the producers we are restricting the decline of reservoir pressure while delaying water cut. As for producers, the optimization process seems to favor long horizontal wells placed in the most permeable top layers of the reservoir to increase oil production while reducing water encroachment (see kh variation in **Fig. 42**). The optimized locations are found to be either in the middle of reservoir flanked by injectors from both

the top and the bottom of the reservoir or in the unswept and undeveloped areas of the reservoir.

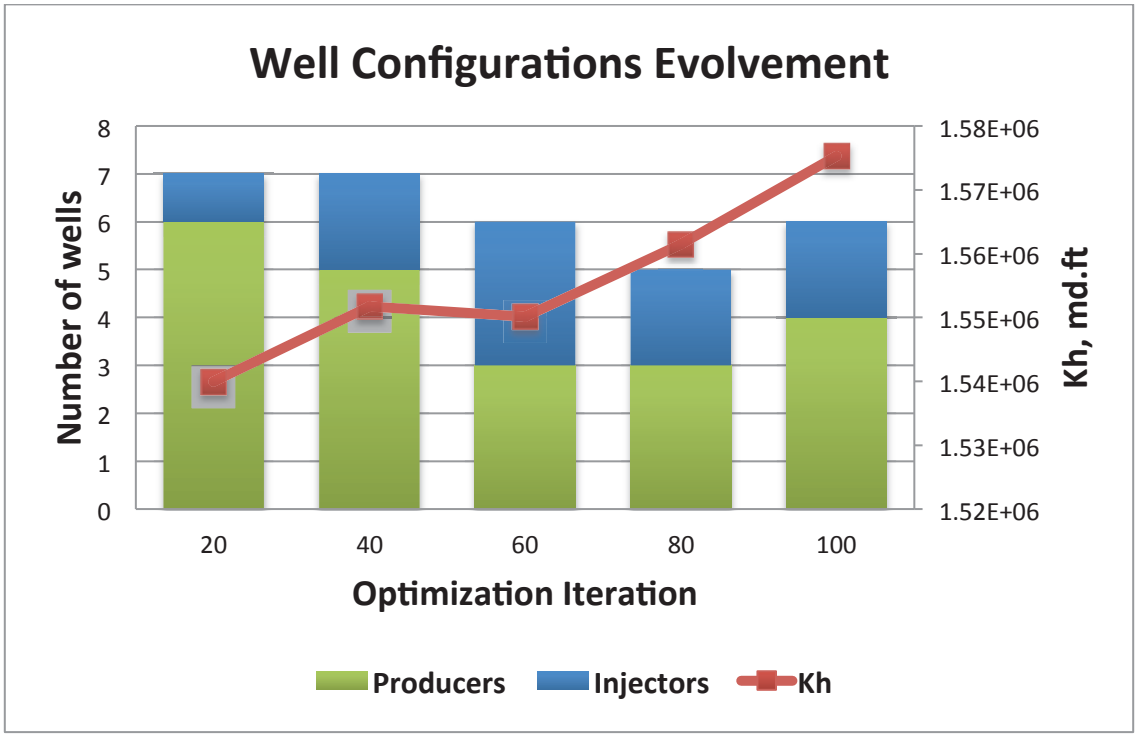

Figure 42: Change in the optimum number/type of wells and total kh along the well path at various optimization iterations

6. SUMMARY AND CONCLUSION

As the demand for oil has continued to grow, over the past decade many researchers have worked to identify new ways to effectively develop oil and gas fields. One important aspect of this effort has been to identify effective and robust optimization techniques to solve well placement optimization problems. Stochastic optimization techniques have been applied to oil field development and show promising results. In this study, we examined the effectiveness of applying imperialist competitive algorithm (ICA) as an optimization tool by comparing its results with those of the well-known genetic algorithm (GA) and particle swarm optimization (PSO) techniques. Our results show that, overall, imperialist competitive algorithm (ICA) performed better than both the genetic algorithm (GA) and particle swarm optimization (PSO) in four synthetic reservoir models in which we had optimized the well placement and number of producers to be drilled to maximize the NPV. In addition, to further test the ICA optimization technique, we applied the ICA to a sector of a Middle East reservoir (ME1) to optimize the well types (producer/injector) and well configuration (vertical/horizontal). We present our major findings and conclusions below:

- The results show that the imperialist competitive algorithm (ICA) achieved better convergence to the global optimum, thereby supporting the utilization of the imperialist competitive algorithm (ICA) as a viable option for well placement optimization.

- The results of our sensitivity analyzes with respect to three important parameters (revolution ratio, assimilation coefficient, and assimilation angle) show that the recommended imperialist competitive algorithm (ICA) default parameters in our examples generally led to acceptable performances. However, to obtain optimum performance, we recommend tuning the three main imperialist competitive algorithm (ICA) parameters to address specific optimization problems.
- We accounted for geological uncertainty by utilizing a set of five geological realizations and took the expected net present value (J) as the objective function. By applying a robust optimization (RO) formula, we calculated the expected net present value (J), which can represent the optimization performance when applied to any of the five realizations.
- We performed three optimization runs for the real field example and presented the best optimized solution, which suggested drilling with one injector and six producers. We further examined the effectiveness of the final optimized solution by referring to the saturation profile of the ME1 reservoir, which indicates that this well distribution would yield good sweep, especially in the more permeable layers (layers 2 and 3).
- In a final optimization run for the ME1 field, we ran the optimization algorithm for a longer simulation time (20,000 runs), which yielded a better final optimized solution by the use of two injectors and four producers for a better expected net present value (J).

- By analyzing the evolvement of optimization iterations, it is clear that the optimization algorithm is searching for an optimized well configuration that maintains reservoir pressure, maximize oil sweep and reduce water injection while minimizing the number of wells required. This was achieved by placing two injectors at the top of the reservoir (away from the producers) and placing four long horizontal producers in the middle of the reservoir in areas that exhibit high permeability.

7. FUTURE WORK

- To further enhance the imperialist competitive algorithm (ICA) optimization performance in field development, we recommend that an optimization workflow be generated that takes into account real-time data acquisition coupled with the optimization method to achieve a more realistic optimum solution. One approach could be to apply a closed-loop field development strategy whereby the models are updated each time a well is drilled.

- To reduce the simulation computation time, we suggest the utilization of surrogate models to replace the required simulation model runs.

- To further enhance the overall performance of the imperialist competitive algorithm (ICA) optimization framework with respect to well placement, it would be beneficial for the framework to couple well placement with the optimization of well control parameters (flow rates/bottomhole pressures).

- To generate a multiobjective Pareto surface, this work can be extended to multiobjective problems that include additional objectives such as watercut and the recovery factor.

- In this work, we considered the well placement of producers and injectors in a waterflood project. In future work, other recovery methods such as steam-assisted gravity drainage and gas injection could be considered.

REFERENCES

Abo-Hammour, Z.S., 2002, “Advanced Continuous Genetic Algorithms and their applications in the motion planning of robot manipulators and the numerical solution of boundary value problems”, Ph.D. thesis, Quaid-i-Azam University, Islamabad, Pakistan.

Al Dossary, M. & Nasrabadi, H., 2016. Well Placement Optimization Using Imperialist Competitive Algorithm. Journal of Petroleum Science and Engineering, Vol. 147, P. 237 – 248.

Atashpaz-Gargari, E., & Lucas, C., 2007. Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition. IEEE Congress on Evolutionary Computation, 4661–4667.

Bangerth, W., Klie, H., Wheeler, M.F., Stoffa, P.L., Sen, M.K., 2006. On Optimization Algorithms for the Reservoir Oil Well Placement Problem. Comput. Geosci. 10, 303-319.

Beckner, B.L., Song, X., 1995. Field Development Planning Using Simulated Annealing - Optimal Economic Well Scheduling and Placement, SPE Annual Technical Conference and Exhibition Dallas, Texas.

Bittencourt, A.C., Horne, R.N., 1997. Reservoir Development and Design Optimization, SPE Annual Technical Conference and Exhibition San Antonio, Texas.

Castineira, D., Alpak, F.O., 2009. Automatic Well Placement Optimization in a Channelized Turbidite Reservoir Using Adjoint Based Sensitivities, SPE Reservoir Simulation Symposium The Woodlands, Texas.

Chunduru, R.K., Sen, M., Stoffa, P.: Hybrid optimization methods for geophysical inversion. Geophysics **62**, 1196–1207 (1997)

Clerc, M.: Particle Swarm Optimization. iSTE, London (2006)

D. A. Cameron and L. J. Durlofsky. Optimization of well placement, CO_2 injec- tion rates, and brine cycling for geological carbon sequestration. International Journal of Greenhouse Gas Control, 10:100–112, 2012.

Devine, M.D., Lesso, W.G., 1972. Models for Minimum Cost Development of Offshore Oil Fields. Manage. Sci. B Appl. 18, B378-B387.

Dogru, S., 1987. Selection of Optimal Platform Locations. SPE Drilling Eng. 2. 382 – 386.

Eberhardt, R.C., Kennedy, J.: A new optimizer using parti- cle swarm theory. In: Proceedings of the 6th International Symposium on Micromachine and Human Science, pp. 39–43 (1995)

Eeg, O.S., Herring, T., 1997. Combining Linear Programming and Reservoir Simulation to Optimize Asset Value, SPE Production Operations Symposium OklahomaCity, Oklahoma.

Emerick, A.A., Silva, E., Messer, B., Almeida, L.F., Szwarcman, D., Pacheco, M.A.C., Vellasco, M.M.B.R., 2009. Well Placement Optimization Using a Genetic Algorithm with Nonlinear Constraints, SPE Reservoir Simulation Symposium The Woodlands, Texas, USA.

Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley, West Sussex (2005)

Farshi, M.M., 2008. Improving Genetic Algorithms for Optimum Well Placement. Master's Thesis, Stanford University.

Forouzanfar, F., Li, G., Reynolds, A.C., 2010. A Two-Stage Well Placement Optimization Method Based on Adjoint Gradient, SPE Annual Technical Conference and Exhibition Florence, Italy.

Franstrom, K.L., Litvak, M.L.: Automatic simulation algo- rithm for appraisal of future infill development potential of Prudhoe Bay. Paper SPE 59374 presented at the 2000 SPE/DOE Improved Oil Recovery Symposium, Tulsa, 3–5 April 2000

Garcia-Diaz, J.C., Startzman, R., Hogg, G.L., 1996. A New Methodology for Minimizing Investment in the Development of Offshore Fields. SPE Prod. Facil. 11. 22 – 29.

Goldberg, D.E.: Genetic Algorithms in Search, Optimiza- tion, and Machine Learning. Addison-Wesley, Reading (2004)

Guyaguler, B., Horne, R.N.: Uncertainty assessment of well placement optimization. SPE Reservoir Evaluation & Engi- neering, pp. 24–32 (2004)

H. Wang, D. Echeverr´ıa Ciaurri, L. J. Durlofsky, and A. Cominelli. Optimal well placement under uncertainty using a retrospective optimization framework. SPE Journal, 17(1):112–121, 2012.

Handels, M., Zandvliet, M.J., Brouwer, D.R., Jasen, J.D. Adjoint-based well placement optimization under production constraints. SPE Reservoir Simulation Symposium. The Woodlands, TX, 2007.

Hansen, P., Pedrosa, E.D., Ribeiro, C.C., 1992. Location and Sizing of Offshore Platforms for Oil-exploration. Eur. J. Oper. Res. 58, 202-214.

Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms, 2nd edn. Wiley, New York (2004)

Helwig, S., Wanka, R.: Theoretical analysis of initial par- ticle swarm behavior. In: Proceedings of the 10th Interna- tional Conference on Parallel Problem Solving from Nature (PPSN08), pp. 889–898. Springer, Dortmund (2008)

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, Ann Arbor.

Hosseini, S., Al Khaled, A. A Survey on the Imperialist Competitive Algorithm Metaheuristic: Implementation in Engineering Domain and Directions for Future Research. Applied Soft Computing 24 (2014), 1078-1094.

Iyer, R.R., Grossmann, I.E., Vasantharajan, S., Cullick, A.S., 1998. Optimal Planning and Scheduling of Offshore Oil Field Infrastructure Investment and Operations. Ind. Eng. Chem. Res. 37, 1380-1397.

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. In: 1995 IEEE International Conference on Neural Networks Proceedings, vols. 1-6, 1942-1948.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by Simulated Annealing. Science 220, 671-680.

Litvak, M., Gane, B., Williams, G., Mansfield, M., Angert, P., Macdonald, C., McMurray, L., Skinner, R., Walker, G.J.: Field development optimization technology. Paper SPE 106426 presented at the 2007 SPE Reservoir Simulation Sym- posium, Houston, 26–28 February 2007

Lizon, C., D'Ambrosio, C., Liberti, L., Le Ravalec, M., and Sinoquet, D. (2014). A mixed-integer nonlinear optimization approach for well placement and geometry. In ECMOR XIV-14th European conference on the mathematics of oil recovery.

M.A. Ahmadi, M. Ebadi, A. Shokrollahi, S.M.J. Majidi, 2013. Evolving artificial neural network and imperialist competitive algorithm for prediction oil flow rate of the reservoir. Appl Soft Comput, 13, pp. 1085–1098

Mattot, L.S., Rabideau, A.J., Craig J.R.: Pump-and-treat optimization using analytic element method flow models. Adv. Water Resour. **29**, 760–775 (2006)

Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press (1996)

Montes, G., Bartolome, P., Udias, A.L., 2001. The Use of Genetic Algorithms in Well Placement Optimization, SPE Latin American and Caribbean Petroleum Engineering Conference Buenos Aires, Argentina.

Nasrabadi, H., Morales, A., Zhu, D. 2012. Well Placement Optimization: A survey with Special Focus on Application for Gas/Gas-condensate Reservoirs. Journal of Natural Gas Science and Engineering, Vol. 5, 6-16.

Nocedal, J. and Wright, S.J., 1999, "Numerical Optimization", Springer, New York.

Norrena, K.P., Deutsch, C.V., 2002. Automatic Determination of Well Placement Subject to Geostatistical and Economic Constraints, SPE International Thermal Operations and Heavy Oil Symposium and International Horizontal Well Technology Conference Calgary, Alberta, Canada.

Onwunalu, J., 2006. Optimization of Nonconventional Well Placement Using Genetic Algorithms and Statistical Proxy. Master's Thesis, Stanford University.

Onwunalu, J.E., Durlofsky, L.J., 2010. Application of a Particle Swarm Optimization Algorithm for Determining Optimum Well Location and Type. Comput. Geosci. 14, 183-198.

Rajabioun, R., Atashpaz-Gargari, E., & Lucas, C., 2008. Colonial Competitive Algorithm as a Tool for Nash Equilibrium Point Achievement. Lecture Notes in Computer Science, 5073, 680–695.

Rigot, V., 2003. New Well Optimization in Mature Fields. Master's Thesis. Stanford University.

Rosenwald, G.W., Green, D.W., 1974. A Method for Determining the Optimum Location of Wells in a Reservoir Using Mixed-integer Programming. SPE J. 14, 44-54.

Sarma, P., Chen, W.H., 2008. Efficient Well Placement Optimization with Gradient-based Algorithms and Adjoint Models, Intelligent Energy Conference and Exhibition Amsterdam, The Netherlands.

Sen, M., Stoffa, P.: Global Optimization Methods in Geo- physical Inversion. Elsevier (1995)

Sepehri Rad, H., & Lucas, C., 2008. Application of Imperialistic Competition Algorithm in Recommender Systems. In 13th international CSI computer conference (CSICC'08), Kish Island, Iran.

Shi, Y., Eberhardt, R.C.: A modified particle swarm opti- mizer. In: Proceedings of the 1998 IEEE International Con- ference on Evolutionary Computation, pp. 69–73. IEEE, New York (1998)

Spall, J.C., 2004, “Stochastic Optimization”, in Handbook of Computational Statistics (J. Gentle, W. Hardle, and Y. Mori, eds.), Springer−Verlag, New York, pp. 169−197.

Spall, J.C., Introduction to Stochastic Search and Optimiza- tion: Estimation, Simulation and Control. Wiley, New Jersey (2003)

Spall, J.C.: An overview of the simultaneous perturbation method for efficient optimization. John Hopkins APL Tech. Dig. **19**(4), 482–492 (1998)

Sullivan, J., 1982. A Computer-model for Planning the Development of an Offshore Gas-field. J. Petrol. Technol. 34, 1555-1564.

Tupac, Y.J., Faletti, L., Pacheco, M.A.C., Vellasco, M.M.B.R.: Evolutionary optimization of oil field development. Paper SPE 107552 presented at the 2007 SPE Digital Energy Con- ference and Exhibition, Houston, 11–12 April 2007

V. Artus, L. J. Durlofsky, J. E. Onwunalu, and K. Aziz. Optimization of non-conventional wells under uncertainty using statistical proxies. Computational Geosciences, 10(4):389–404, 2006.

Van den Heever, S.A., Grossmann, I.E., 2000. An Iterative Aggregation/Disaggregation Approach for the Solution of a Mixed-integer Nonlinear Oilfield Infrastructure Planning Model. Ind. Eng. Chem. Res. 39, 1955-1971.

Van Essen, G.M., Zandvliet, M.J., Van den Hof, P.M.J., Bosgra, O.H., Jasen, J.D. Robust waterflooding optimization of multiple geological scenarios. SPE-102913-PA. SPE Journal, Vol.14, 202-210, 2009.

Vlemmix, S., Joosten, G.J.P., Brouwer, R., Jansen, J.-D., 2009. Adjoint-Based Well Trajectory Optimization, EUROPEC/EAGE Conference and Exhibition Amsterdam, The Netherlands.

Wang, C., Li, G., Reynolds, A.C., 2007. Optimal Well Placement for Production Optimization, Eastern Regional Meeting Lexington, Kentucky USA.

Watson Jr., W.S., Mahaffey, D.W., Still, J.P., Taylor, R.D., 1989. PLATLOC: a Program for Optimizing Offshore Platform Locations, Petroleum Computer Conference San Antonio, Texas.

Williams, G.J.J., Mansfield, M., Macdonald, D.G., Bush, M.D.: Top-down reservoir modeling. Paper SPE 89974 pre- sented at the 2004 SPE Annual Technical Conference and Exhibition, Houston, 26–29 September 2004

Yasari, E., Pishvaie, M.R., Khorasheh, F., Salahshoor, K. Application of multi-criterion robust optimization in water-flooding of oil reservoir. Journal of Petroleum Science and Engineering 109, 1-11, 2013.

Yeten, B., Durlofsky, L.J., Aziz, K., 2003. Optimization of Nonconventional Well Type, Location, and Trajectory. SPE J. 8., 200 – 210.

Zandvliet, M.J., Handels, M., van Essen, G.M., Brouwer, D.R., Jansen, J.D.: Adjoint-

based well-placement optimization under production constraints. SPE J. **13**(4), 392–399 (2008)

Zhang, F., Reynolds, A.: Optimization algorithms for auto- matic history matching of production data. In: 8th European Conference on the Mathematics of Oil Recovery, ECMOR, Freiberg, Germany, EAGE, September 2002

Zhang, W.J., Xie, X.F., Bi, D.C.: Handling boundary con- straints for numerical optimization by particle swarm flying in periodic search space. In: IEEE Congress on Evolution- ary Computation, vol. 2, pp. 2307–2311. IEEE, Piscataway (2004)

APPENDIX A

DETAILS OF ICA APPLICATION FOR WELL PLACEMENT

In this section, we show step-by-step how the ICA was utilized for well placement optimization and further clarify how a solution evolves through the algorithm process. Regarding methodology, here we apply the ICA in a simplified reservoir model to provide the reader with a clear explanation and to minimize any confusion. We detail the ICA implementation for well placement optimization below:

A.1 The Initiation of Empires

The first step is to define the countries, which in this example refer to the well positions. For example, we start by having ten countries (two of which are imperialists and the remaining eight colonies), which consist of two variables to be optimized (the x and y coordinates). The initial positions of the countries are as follows:

$$\text{Country} = \begin{bmatrix} Country1 \\ Country2 \\ . \\ . \\ . \\ . \\ . \\ Country10 \end{bmatrix} = \begin{bmatrix} 14 & 9 \\ 5 & 34 \\ 8 & 15 \\ 5 & 7 \\ 36 & 32 \\ 22 & 3 \\ 15 & 3 \\ 21 & 1 \\ 32 & 3 \\ 11 & 12 \end{bmatrix}$$

The next step is to calculate the cost of each country by calling the commercial simulator from the MATLAB code to calculate the cumulative oil production for each well's position (country). Then, the countries are sorted in descending order, and the two countries with the highest cumulative oil productions are chosen as the imperialists.

$$\text{Cost} = F(\text{Country}) = \begin{bmatrix} f(8,15) \\ f(22,3) \\ f(15,3) \\ f(32,3) \\ f(5,34) \\ f(36,32) \\ f(21,1) \\ f(11,12) \\ f(14,9) \\ f(5,7) \end{bmatrix} = \begin{bmatrix} -23,715 \\ -22,480 \\ -21,452 \\ -21,206 \\ -20,744 \\ -20,348 \\ -20,049 \\ -19,216 \\ -16,308 \\ -13,733 \end{bmatrix}$$

In this case, the imperialists are the countries that have the (X ,Y) coordinates of (8, 15) and (22,3). The remaining countries (colonies) are distributed between the two empires based on the relative power of each. The number of colonies an empire can possess is directly proportional to its power. In this example, there are only two imperialists. The distribution of colonies among the imperialists begins by calculating the costs of the imperialists and then determining the normalized costs using Equation (3), as provided in the main text of this dissertation:

$N_{imp} = 2$, $N_{col} = 8$, $c_1 = -23{,}715$, $c_2 = -22{,}480$;

C_1 = -23,715 – (-22,480) = -1,235 ; C_2 = -22,480 – (-22,480) = 0

Next, the power of each imperialist is calculated based on Equation (4):

$p_1 = \left|\frac{-23{,}715}{-23{,}715-22{,}480}\right| = 0.51$; $p_2 = \left|\frac{-22{,}480}{-23{,}715-22{,}480}\right| = 0.49$

After determining of the power of the two imperialists, the remaining eight colonies are randomly distributed between them to create two empires. The number of colonies each of the imperialists possess is determined by Equation (5):

$N.C_1$ = round(0.51*8) = 4 ; $N.C_2$ = round(0.49*8) = 4

The results indicate that each empire possesses four colonies. The initiation of the empires is now complete and the algorithm is ready for the next step, which is the assimilation process.

A.2 Assimilation of Colonies

After the initiation of the empires, the next step is to move colonies toward their respective imperialists. In this step, we move all colonies toward their imperialists by moving an x distance closer to the imperialist position. The distance x is chosen from a random distribution within an interval of [0, d*(assimilation.coefficient)]. Also, to make

the assimilation process more robust and effective, a deviation parameter is assigned to the assimilation process, to ensure a greater solution search space.

A.3 Revolution

After the colonies move toward their imperialists, some colonies are chosen for revolution, which, in this case, involves a sudden change in the X and Y coordinates of the well positions. This revolution process provides for more search and exploration of the solution space and thus prevents pre-mature convergence. The revolution process is analogous to mutation in the GA.

A.4 Imperialist and Colonies Position Exchange:

After the colonies complete the assimilation and revolution processes, the new colonies' positions (wells' X and Y coordinates) are used to calculate their new cost functions (cumulative oil production). Then, the cost functions (cumulative oil production) of the colonies are compared with those of their relative imperialist. If any colony is found to have a better cost function than its imperialist, then the imperialist exchanges its position with that of the colony. **Fig.15** below summarizes the empire initiation, assimilation, revolution, and exchange processes.

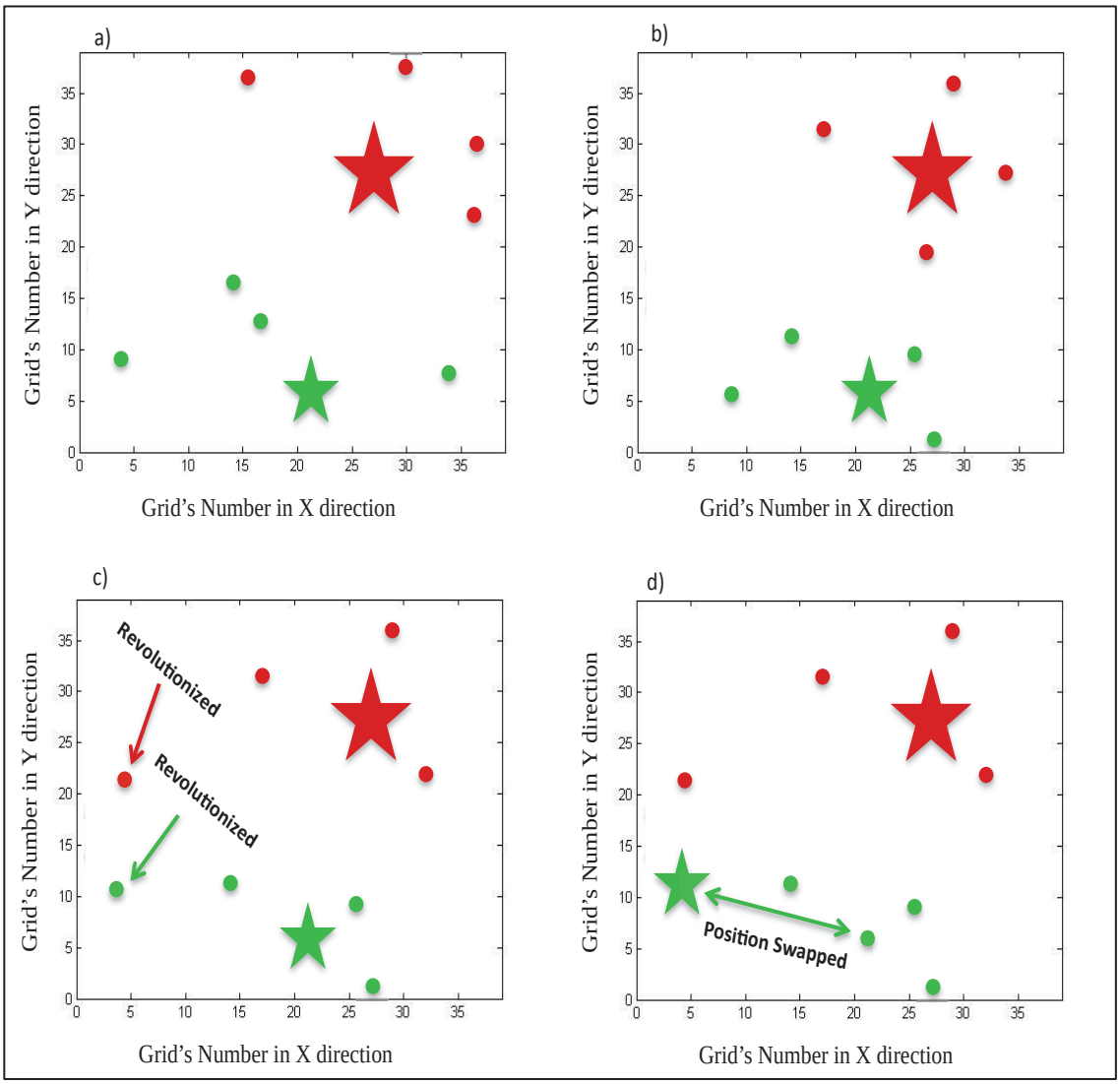

Figure A-1: a) Initiation of empires, b) assimilation process, c) revolution process, and d) position swapping process

A.5 Competition between Empires:

In this step, competition is initiated between the empires, whereby the imperialists compete in order to take possession of the weakest colonies of the weakest empires. This process begins by calculating the power of each empire, which is predominantly a function of its imperialist power but is also a small function of the colonies' mean power, as shown by Equation (6). Next, the competition begins between the empires to overtake the weakest colony of the weakest empire. To do so, the normalized total cost (NTCn) of each empire is calculated using Equation (7). Then, the probability of each empire taking possession of this weakest colony is calculated using Equation (8). Finally, the empire with the greatest power and possession probability conquers this colony. Below, we provide a simplified example showing the steps just described.

1) The total cost of each empire is calculated using Equation (6) and the resulting total costs are shown as follows:

Total cost of empire 1 (TC_1) = -24511

Total cost of empire 2 (TC_2) = -22568

Total cost of empire 3 (TC_3) = -21033

2) The normalized cost of each empire is calculated using Equation (7) as follows:

Normalized cost of empire 1 (NTC_1) = -24511-(-21033) = -3478

Normalized cost of empire 2 (NTC_2) = -22568-(-21033) = -1535

Normalized cost of empire 3 (NTC_3) = -21033-(-21033) = 0

3) The possession probability of each empire is calculated using equation (8):

Possession probability of empire 1 (P_{p1}) = $\left|\frac{-3478}{-3478-1535-0}\right|$ = 0.69

Possession probability of empire 2 (P_{p2}) = $\left|\frac{-1535}{-3478-1535-0}\right|$ = 0.31

Possession probability of empire 2 (P_{p3}) = $\left|\frac{0}{-3478-1535-0}\right|$ = 0.00

Next, the P, R, and D vectors (as defined in the main text of this dissertation) are determined to assign the weakest colony to the empire with the highest D index. Here, vector **P** = [0.69, 0.31, 0], vector **R** = [0.53, 0.11, 0.79], and vector **D** = [0.16, 0.2, -0.79]. So, in this example, the colony will be possessed by empire number 2. **Fig. 16** shows the collapse of the weakest colony and its possession by the second strongest empire (green empire) as follows:

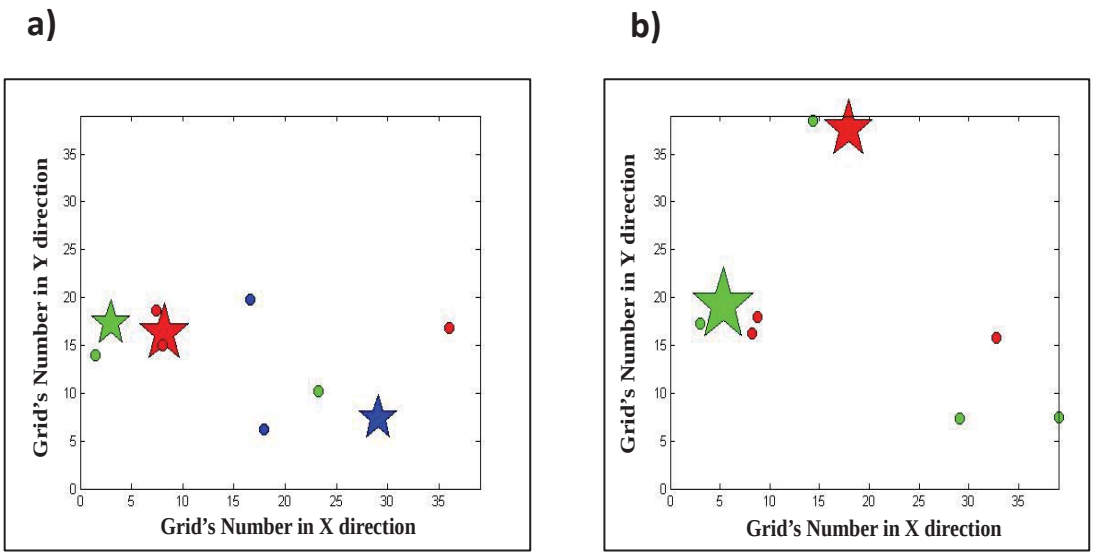

Figure A-2: Collapse of weakest empire and possession of colonies by stronger empire: a) before the possession process, b) after the possession process.

APPENDIX B

The full ICA code that was used in this work was implemented through MATLAB 2014 version. The full ICA code is included in a folder and it is intended to be allowed for public usage. Should you need to get a copy of the entire files that were associated with the ICA code, please do not hesitate to contact the authors.

Mohammad Al Dossary

mohada@tamu.edu

Dr. Hadi Nasrabadi

Hadi.nasrabadi@tamu.edu

APPENDIX C

INPUT FILE USED FOR MULTIPLE HORIZONTAL WELL PLACEMENT

```
INUNIT FIELD

WSRF WELL 1

WSRF GRID TIME

WSRF SECTOR TIME

OUTSRF WELL LAYER NONE

OUTSRF RES ALL

OUTSRF GRID SO SG SW PRES OILPOT BPP SSPRES WINFLUX

WPRN GRID 0

OUTPRN GRID NONE

OUTPRN RES NONE

**$  Distance units: ft

RESULTS XOFFSET          0.0000

RESULTS YOFFSET          0.0000

**$  (DEGREES)

RESULTS ROTATION          0.0000  **  (DEGREES)

RESULTS AXES-DIRECTIONS 1.0 -1.0 1.0
```

```
**$
*************************************************************************
****
**$ Definition of fundamental cartesian grid
**$
*************************************************************************
****
GRID VARI 40 40 3
KDIR DOWN
DI IVAR
 40*300
DJ JVAR
 40*300
DK ALL
 4800*50
DTOP
 1600*2000
PERMJ EQUALSI
PERMK EQUALSI
**$ Property: NULL Blocks  Max: 1  Min: 1
**$  0 = null block, 1 = active block
NULL CON           1
```

POR ALL

0.1134959 2*0.08857616 0.1134959 3*0.08857616 0.2028321 0.1997039

0.1585346 0.1464379 0.23393 0.2292194 0.2246236 0.1529927 0.2070665

0.1997039 0.1585346 0.2313142 0.2332937 0.2306294 0.208374 0.23393

0.1464379 0.1675697 0.2108544 0.2096356 0.1529927 0.1713576 2*0.1675697

0.1529927 0.1675697 0.1529927 0.1585346 0.2246236 0.221197 0.1585346

0.201302 0.1997039 2*0.08857616 2*0.1134959 0.08857616 0.1134959 0.08857616

0.2057097 0.208374 0.1464379 0.1675697 0.2326459 0.2363684 0.228493

0.1633353 0.1962773 0.1997039 0.1675697 0.2380944 0.2319862 0.2397413

0.2108544 0.236953 0.1713576 0.1529927 0.1904345 0.2096356 0.2541391

0.2511414 0.2507485 0.2478708 0.1675697 0.1464379 0.1585346 0.1529927

0.2313142 0.2262217 0.1464379 2*0.208374 2*0.1134959 3*0.08857616

2*0.1134959 0.2096356 0.1904345 0.1464379 0.1529927 0.2254315 0.221197

0.2269949 0.1529927 0.2096356 0.1997039 0.1713576 0.228493 0.2313142

0.2386518 0.2070665 0.2332937 0.1713576 0.1464379 0.1904345 0.2096356

0.2470045 0.2507485 0.2474402 0.2491324 2*0.1529927 0.1675697 0.1585346

0.2299313 0.2229511 0.1675697 0.1859347 0.1633353 0.08857616 2*0.2515301

0.2465633 0.2474402 0.2487168 0.2519146 0.1904345 0.1944332 0.1633353

0.1464379 0.2380944 0.2306294 0.2351699 0.1633353 0.2070665 0.1464379

0.1585346 0.1633353 0.2220848 0.2254315 0.2096356 0.2319862 0.2277518

0.1997039 0.1924894 0.1713576 0.2470045 0.2515301 0.2495433 0.2470045

0.1713576 0.1633353 3*0.1464379 0.2397413 0.2392007 0.2096356 0.1675697

0.1134959 0.2491324 0.2507485 0.2541391 0.2534128 0.2495433 0.2465633

0.2070665 0.2096356 0.1675697 0.1529927 0.1633353 0.2351699 0.2397413

0.188255 0.1962773 0.1713576 0.2277518 0.1529927 0.2375282 0.236953

0.1997039 0.2392007 0.1944332 0.2237971 0.201302 0.1713576 2*0.2526715

0.2470045 0.2474402 0.2487168 0.2519146 0.1464379 0.1585346 0.1713576

0.23393 0.2306294 0.1859347 0.1713576 0.1134959 0.2465633 0.2503512

0.2507485 0.2499495 0.2487168 0.2526715 0.1962773 0.2035733 0.1675697

0.1713576 0.1529927 0.2345553 0.2397413 0.1859347 0.1585346 0.1633353

0.2292194 0.1675697 0.2254315 0.2386518 0.2108544 0.2035733 0.2096356

0.2363684 0.1464379 0.2096356 0.2515301 0.2534128 0.253044 2*0.2515301

0.2507485 0.1585346 0.1633353 0.2319862 0.2254315 0.2220848 0.1962773

0.1464379 0.2470045 0.2511414 0.2470045 0.2534128 0.2515301 0.08857616

0.1134959 0.1980314 0.2042997 0.1713576 0.1633353 0.1464379 0.23393

0.2262217 0.1962773 0.2108544 0.1464379 0.2357741 0.2351699 0.2326459

0.2292194 0.2397413 0.2380944 0.1904345 0.1944332 0.236953 0.2096356

0.2541391 0.2537777 0.2519146 0.2495433 0.2534128 0.2537777 0.1585346

0.2397413 0.2357741 0.2386518 0.1633353 0.208374 0.1529927 0.2511414

0.2537777 0.2515301 0.2537777 0.252295 0.08857616 0.1980314 0.2070665

0.2050116 3*0.1134959 0.236953 0.2357741 2*0.2096356 0.2332937 0.221197

0.2229511 0.2220848 2*0.2363684 0.2345553 0.201302 0.2096356 0.2299313

0.1997039 0.1134959 0.2487168 0.2519146 0.2491324 0.2519146 0.2495433

0.08857616 0.2351699 0.2306294 0.2375282 0.08857616 2*0.1997039 0.2515301

2*0.2511414 0.2519146 0.2534128 0.08857616 0.2096356 0.2108544 0.2057097

3*0.1134959 0.2313142 0.2380944 0.1997039 0.2042997 0.2292194 0.08857616

0.1134959 0.228493 0.2375282 0.2491324 0.2042997 0.1859347 0.2035733

0.2237971 0.2345553 0.08857616 0.2461167 0.2511414 0.2482963 0.2507485

0.1134959 0.2375282 2*0.2220848 0.08857616 0.1134959 2*0.208374 0.2519146

0.2526715 2*0.2537777 0.2470045 0.1924894 0.1904345 0.2057097 0.2063945

4*0.1134959 0.2262217 0.2332937 0.2070665 0.1962773 0.08857616 0.1134959

0.2351699 0.221197 0.2491324 0.2070665 0.2503512 0.2035733 0.1944332

0.2380944 0.1134959 0.2507485 0.2526715 0.2487168 0.2541391 0.1134959

0.2269949 0.2313142 2*0.08857616 0.188255 0.1859347 0.1134959 0.2495433

0.252295 0.2519146 0.2495433 0.2470045 0.2028321 0.2070665 0.1962773

0.2070665 4*0.1134959 0.2345553 0.2246236 0.188255 0.2028321 0.1904345

0.1134959 0.08857616 0.2269949 0.2491324 0.2108544 0.2495433 0.2035733

0.1859347 0.2237971 0.1134959 0.2503512 0.2470045 0.253044 0.2487168

0.08857616 0.2313142 0.2306294 0.1134959 0.08857616 0.1944332 0.1962773

0.08857616 0.2537777 0.2487168 0.2495433 0.08857616 0.1134959 0.201302

0.2057097 6*0.1134959 0.2229511 2*0.2220848 0.2070665 0.2057097 2*0.08857616

0.2269949 2*0.2491324 0.2495433 0.08857616 0.1997039 0.2096356 0.2292194

0.1134959 2*0.08857616 3*0.1134959 0.23393 0.2306294 0.08857616 0.1944332

0.1134959 0.08857616 0.2499495 0.2511414 0.2507485 2*0.08857616 0.201302

0.2108544 5*0.1134959 0.2254315 0.2332937 0.2229511 0.2319862 0.2108544

0.1924894 2*0.1134959 0.2357741 0.2397413 0.2491324 0.253044 2*0.2057097

0.2035733 0.2306294 0.1134959 0.08857616 0.1134959 0.08857616 0.1134959 0.08857616 0.2237971 0.2397413 0.08857616 0.1924894 0.208374 0.08857616 0.2526715 0.2511414 0.2507485 0.1924894 0.2096356 0.1904345 0.2042997 2*0.1134959 0.2499495 0.2511414 0.1134959 0.2326459 0.2277518 0.2220848 0.2380944 0.08857616 0.1962773 0.08857616 0.2262217 0.2363684 0.2319862 0.2491324 0.2461167 0.201302 0.08857616 0.1962773 0.2028321 0.2326459 0.08857616 4*0.1134959 2*0.2345553 0.2229511 0.2277518 0.208374 0.1134959 0.2499495 0.2511414 0.2507485 0.1980314 0.1944332 0.2070665 0.201302 0.1134959 0.2511414 2*0.2470045 0.1134959 0.2299313 2*0.1134959 0.23393 0.1134959 0.1997039 0.2057097 0.2277518 0.2246236 0.08857616 0.2491324 0.1904345 0.1944332 0.08857616 0.1962773 0.1924894 0.2380944 0.1134959 0.08857616 0.1134959 3*0.08857616 0.1134959 0.2220848 0.2332937 0.1962773 0.1944332 0.2534128 0.2511414 0.2507485 0.1997039 0.1859347 0.2042997 0.1859347 0.1134959 0.2515301 0.2526715 0.253044 0.1134959 0.2306294 2*0.08857616 0.2375282 0.08857616 0.1997039 0.1924894 0.2319862 0.08857616 2*0.1904345 0.1944332 0.1924894 0.1134959 0.2057097 0.201302 0.2397413 0.2269949 0.2380944 3*0.08857616 0.2345553 0.2386518 0.2319862 0.2332937 0.2028321 0.2070665 0.2503512 0.2511414 0.08857616 0.2028321 0.201302 0.2108544 0.1859347 0.253044 0.2495433 0.2515301 2*0.1134959 0.2254315 0.23393 0.1134959 0.2392007 0.2363684 0.2057097 0.208374 0.2326459 0.1904345 0.1997039 0.2042997 0.2503512 0.2515301 0.08857616 0.1944332 0.188255 0.1904345 0.2269949 0.2363684 0.2237971 2*0.08857616 0.228493

0.2262217 0.08857616 2*0.1134959 0.1997039 0.2503512 2*0.08857616

0.2070665 0.2028321 0.1962773 0.1924894 0.2519146 0.2526715 0.2465633

2*0.1134959 0.2351699 0.2277518 0.1134959 0.2357741 0.2220848 0.2096356

0.1859347 0.221197 0.2096356 0.1962773 0.1944332 0.2461167 0.2487168

2*0.1134959 0.08857616 0.188255 0.08857616 0.2269949 2*0.1134959 0.2386518

0.221197 0.2277518 0.1134959 2*0.08857616 0.2096356 0.1134959 2*0.08857616

0.1962773 0.1924894 0.1134959 0.2070665 0.2482963 0.2519146 2*0.1134959

0.2345553 0.2351699 0.2380944 0.1134959 0.236953 0.2386518 0.2237971

2*0.2292194 0.2096356 0.2108544 2*0.201302 0.2108544 0.1944332 0.08857616

0.188255 0.08857616 0.1134959 0.228493 0.08857616 0.1134959 0.2237971

0.2254315 3*0.08857616 0.2042997 2*0.1134959 2*0.08857616 0.2028321

0.2042997 0.1134959 0.1962773 4*0.1134959 0.2357741 0.2332937 4*0.08857616

0.2332937 0.2246236 0.2332937 0.1134959 0.1980314 0.2028321 0.2070665

0.1980314 0.1924894 0.08857616 0.188255 0.2108544 0.08857616 0.1134959

0.2269949 0.23393 0.2299313 0.23393 0.08857616 0.2070665 0.1962773

0.188255 2*0.1134959 2*0.08857616 0.2108544 0.1962773 0.1134959 0.2070665

3*0.1134959 0.221197 0.236953 0.2363684 0.2503512 0.2491324 0.2495433

0.08857616 0.2397413 0.2357741 0.1134959 0.08857616 0.1944332 0.1924894

0.2534128 0.2474402 0.2461167 0.1980314 0.2096356 0.1904345 0.08857616

0.1134959 0.228493 0.236953 2*0.1134959 0.1904345 0.1962773 0.1980314

2*0.08857616 0.1134959 0.08857616 0.1859347 0.2070665 2*0.1134959

0.2042997 0.1980314 2*0.1134959 0.2269949 0.2306294 0.2363684 0.2503512

0.2487168 0.2470045 0.1134959 0.2299313 0.2345553 0.08857616 0.1980314

0.2070665 0.08857616 0.2515301 0.2487168 0.2465633 0.208374 0.08857616

0.1134959 0.08857616 0.1134959 2*0.2319862 0.08857616 0.1134959 0.188255

0.2478708 0.2503512 0.2515301 0.2461167 0.1134959 0.08857616 0.1944332

0.1859347 2*0.1134959 0.1980314 0.1859347 2*0.1134959 0.2357741 0.2363684

0.2237971 0.08857616 0.253044 0.1134959 0.2375282 0.2277518 0.08857616

0.2246236 0.1962773 0.1924894 0.1134959 0.2465633 0.2491324 0.2474402

0.2042997 0.1980314 0.2096356 0.08857616 0.2363684 0.08857616 0.1134959

0.2306294 0.2392007 0.2057097 0.2511414 2*0.2534128 0.2537777 0.1134959

0.08857616 0.2096356 0.201302 2*0.1134959 0.2042997 0.1904345 0.1944332

2*0.1134959 0.2306294 0.236953 0.1134959 0.08857616 0.2375282 0.23393

0.2397413 0.08857616 0.2269949 0.221197 0.1944332 0.08857616 0.2526715

0.252295 0.2461167 0.1944332 0.2096356 0.2042997 0.2220848 0.2313142

2*0.08857616 0.23393 0.2326459 0.1980314 0.2487168 0.253044 0.2515301

0.2526715 0.1134959 0.1944332 0.1924894 0.2070665 0.2028321 0.1134959

0.1924894 0.2028321 0.188255 3*0.1134959 0.2292194 0.1134959 0.08857616

0.2313142 0.2246236 0.2363684 0.2357741 0.08857616 0.2397413 0.2057097

0.1904345 0.08857616 2*0.1134959 0.2057097 2*0.08857616 0.2357741

3*0.08857616 0.2220848 0.2375282 0.08857616 0.2526715 0.2461167 0.2537777

0.2487168 0.1134959 0.201302 2*0.2096356 0.2108544 0.1134959 0.2057097

0.1904345 4*0.1134959 0.2237971 0.08857616 0.23393 0.2229511 0.08857616

0.236953 2*0.1134959 0.2357741 0.1962773 0.1980314 0.2108544 0.08857616

0.2108544 0.1859347 0.1134959 0.236953 0.2345553 0.1134959 0.221197

0.2363684 0.1134959 0.236953 0.08857616 0.1997039 0.08857616 0.1134959

0.08857616 0.2070665 0.1944332 0.208374 0.1134959 0.2028321 2*0.1134959

0.1944332 4*0.1134959 0.2254315 0.2363684 0.2229511 0.2254315 0.08857616

0.2254315 0.2351699 0.2319862 0.2375282 2*0.1134959 0.2028321 0.2042997

0.2108544 0.2096356 0.08857616 0.2380944 0.2386518 0.08857616 0.2380944

0.2386518 0.08857616 0.2292194 0.08857616 0.208374 2*0.08857616 3*0.1134959

0.188255 0.1134959 0.201302 0.1944332 0.1134959 2*0.1904345 2*0.1134959

0.228493 0.2357741 0.2375282 0.08857616 0.2262217 0.1134959 0.2461167

0.2511414 0.2386518 0.23393 0.08857616 0.1134959 0.252295 0.1944332

2*0.08857616 0.2057097 2*0.2254315 0.2299313 0.2357741 0.2220848 0.1134959

0.2332937 0.2345553 0.08857616 0.1904345 2*0.08857616 2*0.1134959

0.1904345 0.1134959 0.1924894 0.2070665 0.1134959 0.2042997 0.208374

2*0.1134959 0.2375282 0.2397413 0.2269949 0.1134959 0.2262217 0.1134959

0.2537777 0.2526715 0.2495433 0.2277518 0.1134959 0.08857616 0.2487168

0.1924894 2*0.08857616 2*0.201302 0.221197 0.2313142 0.2332937 2*0.1134959

2*0.08857616 0.2332937 0.2070665 0.1134959 0.08857616 2*0.1134959

0.2042997 0.1134959 0.253044 0.1924894 0.1134959 0.2057097 0.1997039

2*0.1134959 0.2363684 0.228493 0.2397413 0.1134959 0.2306294 0.08857616

0.2515301 0.2541391 0.2478708 0.2375282 0.2254315 0.1134959 0.2042997

0.1980314 0.08857616 2*0.1134959 0.1904345 0.2269949 0.2306294 0.221197

0.1134959 2*0.08857616 0.1134959 0.2269949 0.2229511 0.2096356 0.1134959

0.201302 0.1980314 0.1904345 0.2499495 0.2507485 0.2108544 0.201302

0.2042997 0.1904345 3*0.1134959 0.2319862 0.2326459 0.2526715 0.1134959

0.2491324 2*0.2495433 0.2482963 0.228493 0.2345553 0.08857616 0.2042997

0.1944332 0.2057097 2*0.1134959 0.08857616 0.2237971 0.08857616 0.2357741

0.2246236 3*0.1134959 0.2319862 0.2254315 0.1859347 0.08857616 0.1980314

0.1134959 0.2108544 0.2515301 0.2482963 2*0.2057097 0.1962773 0.1904345

3*0.1134959 0.2386518 0.236953 0.1134959 0.2491324 0.2534128 0.252295

0.2541391 0.2220848 0.08857616 0.2229511 2*0.08857616 0.1134959 0.2108544

0.1904345 2*0.1134959 0.23393 0.08857616 0.2277518 0.2299313 0.2237971

2*0.08857616 0.2351699 0.08857616 0.2070665 0.1134959 0.2028321 0.1134959

0.2042997 0.2526715 0.253044 0.2057097 0.1904345 0.2096356 0.201302

3*0.1134959 0.2262217 0.2220848 0.236953 0.1134959 0.253044 2*0.2470045

0.2220848 0.1134959 0.2326459 0.2254315 2*0.1134959 0.1962773 0.2070665

2*0.08857616 0.2319862 0.08857616 0.1134959 0.2269949 0.23393 0.2380944

0.2345553 0.23393 0.1134959 0.2096356 4*0.1134959 0.2461167 0.2478708

0.2057097 0.1944332 0.201302 0.2028321 2*0.1134959 0.2254315 0.2357741

0.1134959 0.2363684 2*0.1134959 0.2519146 0.2495433 0.2229511 0.1134959

0.228493 0.2299313 2*0.1134959 0.1859347 0.201302 2*0.08857616 0.23393

3*0.08857616 0.2313142 2*0.2397413 0.08857616 0.1859347 0.188255 0.08857616

0.253044 0.2495433 0.253044 0.2526715 0.2499495 0.1859347 0.1997039

0.1134959 0.1924894 2*0.1134959 0.2345553 0.23393 0.1134959 0.2319862

0.2277518 0.1134959 0.2499495 0.2306294 2*0.1134959 0.08857616 0.2363684

0.2254315 0.1134959 0.1980314 0.2042997 0.1134959 0.2277518 0.2375282

0.08857616 0.1134959 0.2386518 0.2375282 0.2515301 0.2526715 0.2470045

0.1859347 0.1134959 0.08857616 0.2474402 0.2495433 0.2482963 0.2507485

0.2070665 0.188255 0.201302 0.1134959 0.2028321 2*0.1134959 0.2306294

0.2375282 0.1134959 0.2397413 0.2357741 0.2332937 0.252295 0.228493

0.1134959 0.08857616 2*0.1134959 0.2299313 0.08857616 0.2070665 0.2375282

0.2319862 0.221197 0.2357741 0.08857616 0.1134959 0.2269949 0.2363684

0.2515301 0.2526715 0.2507485 0.1962773 0.1134959 0.08857616 2*0.2482963

0.2541391 0.2515301 0.2042997 0.1944332 2*0.1134959 0.208374 0.1134959

0.2254315 0.2292194 0.2380944 3*0.1134959 0.2351699 0.2478708 0.2220848

0.08857616 0.2526715 2*0.1134959 0.2363684 0.2254315 0.1962773 0.2313142

0.2332937 0.2237971 0.08857616 0.1134959 0.2351699 0.1134959 0.2397413

0.2515301 0.2526715 0.2465633 0.2096356 0.1904345 0.08857616 0.2461167

0.2519146 0.1134959 0.1944332 0.1904345 3*0.1134959 0.1980314 0.1134959

0.2357741 0.2269949 4*0.1134959 0.2220848 0.23393 0.2262217 0.1134959

0.2537777 0.2515301 0.2537777 0.2254315 0.2299313 0.201302 0.1134959

0.2262217 0.2319862 0.2246236 0.236953 0.2246236 0.1134959 0.2534128

0.2461167 0.2470045 0.2511414 0.252295 0.1944332 0.08857616 0.2465633

0.2470045 0.1134959 0.2070665 0.208374 0.2096356 2*0.1134959 0.1859347

0.1134959 0.2277518 0.2237971 0.1134959 0.2495433 0.2503512 0.1134959

0.2351699 0.236953 0.2246236 0.08857616 0.2465633 0.2491324 0.2465633

0.2299313 0.2363684 0.2057097 0.1980314 0.08857616 0.2380944 0.2392007

0.2313142 2*0.2478708 0.2503512 0.2507485 0.2482963 0.2507485 0.2503512

0.2108544 0.1859347 3*0.1134959 0.1924894 0.1134959 0.2108544 2*0.1134959

0.2042997 2*0.1134959 0.2332937 0.1134959 0.2534128 0.2515301 2*0.1134959

0.2292194 0.23393 0.08857616 0.2491324 0.2519146 0.253044 2*0.2363684

0.201302 0.2042997 0.08857616 0.2269949 2*0.2277518 0.2478708 0.1134959

0.2537777 0.253044 0.08857616 0.2541391 0.2491324 0.2028321 0.1944332

0.1134959 2*0.08857616 0.1134959 3*0.08857616 0.2028321 0.1997039

0.1585346 0.1464379 0.23393 0.2292194 0.2246236 0.1529927 0.2070665

0.1997039 0.1585346 0.2313142 0.2332937 0.2306294 0.208374 0.23393

0.1464379 0.1675697 0.2108544 0.2096356 0.1529927 0.1713576 2*0.1675697

0.1529927 0.1675697 0.1529927 0.1585346 0.2246236 0.221197 0.1585346

0.201302 0.1997039 2*0.08857616 2*0.1134959 0.08857616 0.1134959 0.08857616

0.2057097 0.208374 0.1464379 0.1675697 0.2326459 0.2363684 0.228493

0.1633353 0.1962773 0.1997039 0.1675697 0.2380944 0.2319862 0.2397413

0.2108544 0.236953 0.1713576 0.1529927 0.1904345 0.2096356 0.2541391

0.2511414 0.2507485 0.2478708 0.1675697 0.1464379 0.1585346 0.1529927

0.2313142 0.2262217 0.1464379 2*0.208374 2*0.1134959 3*0.08857616

2*0.1134959 0.2096356 0.1904345 0.1464379 0.1529927 0.2254315 0.221197

0.2269949 0.1529927 0.2096356 0.1997039 0.1713576 0.228493 0.2313142

0.2386518 0.2070665 0.2332937 0.1713576 0.1464379 0.1904345 0.2096356

0.2470045 0.2507485 0.2474402 0.2491324 2*0.1529927 0.1675697 0.1585346

0.2299313 0.2229511 0.1675697 0.1859347 0.1633353 0.08857616 2*0.2515301

0.2465633 0.2474402 0.2487168 0.2519146 0.1904345 0.1944332 0.1633353

0.1464379 0.2380944 0.2306294 0.2351699 0.1633353 0.2070665 0.1464379

0.1585346 0.1633353 0.2220848 0.2254315 0.2096356 0.2319862 0.2277518

0.1997039 0.1924894 0.1713576 0.2470045 0.2515301 0.2495433 0.2470045

0.1713576 0.1633353 3*0.1464379 0.2397413 0.2392007 0.2096356 0.1675697

0.1134959 0.2491324 0.2507485 0.2541391 0.2534128 0.2495433 0.2465633

0.2070665 0.2096356 0.1675697 0.1529927 0.1633353 0.2351699 0.2397413

0.188255 0.1962773 0.1713576 0.2277518 0.1529927 0.2375282 0.236953

0.1997039 0.2392007 0.1944332 0.2237971 0.201302 0.1713576 2*0.2526715

0.2470045 0.2474402 0.2487168 0.2519146 0.1464379 0.1585346 0.1713576

0.23393 0.2306294 0.1859347 0.1713576 0.1134959 0.2465633 0.2503512

0.2507485 0.2499495 0.2487168 0.2526715 0.1962773 0.2035733 0.1675697

0.1713576 0.1529927 0.2345553 0.2397413 0.1859347 0.1585346 0.1633353

0.2292194 0.1675697 0.2254315 0.2386518 0.2108544 0.2035733 0.2096356

0.2363684 0.1464379 0.2096356 0.2515301 0.2534128 0.253044 2*0.2515301

0.2507485 0.1585346 0.1633353 0.2319862 0.2254315 0.2220848 0.1962773

0.1464379 0.2470045 0.2511414 0.2470045 0.2534128 0.2515301 0.08857616

0.1134959 0.1980314 0.2042997 0.1713576 0.1633353 0.1464379 0.23393

0.2262217 0.1962773 0.2108544 0.1464379 0.2357741 0.2351699 0.2326459

0.2292194 0.2397413 0.2380944 0.1904345 0.1944332 0.236953 0.2096356

0.2541391 0.2537777 0.2519146 0.2495433 0.2534128 0.2537777 0.1585346

0.2397413 0.2357741 0.2386518 0.1633353 0.208374 0.1529927 0.2511414

0.2537777 0.2515301 0.2537777 0.252295 0.08857616 0.1980314 0.2070665

0.2050116 3*0.1134959 0.236953 0.2357741 2*0.2096356 0.2332937 0.221197

0.2229511 0.2220848 2*0.2363684 0.2345553 0.201302 0.2096356 0.2299313

0.1997039 0.1134959 0.2487168 0.2519146 0.2491324 0.2519146 0.2495433

0.08857616 0.2351699 0.2306294 0.2375282 0.08857616 2*0.1997039 0.2515301

2*0.2511414 0.2519146 0.2534128 0.08857616 0.2096356 0.2108544 0.2057097

3*0.1134959 0.2313142 0.2380944 0.1997039 0.2042997 0.2292194 0.08857616

0.1134959 0.228493 0.2375282 0.2491324 0.2042997 0.1859347 0.2035733

0.2237971 0.2345553 0.08857616 0.2461167 0.2511414 0.2482963 0.2507485

0.1134959 0.2375282 2*0.2220848 0.08857616 0.1134959 2*0.208374 0.2519146

0.2526715 2*0.2537777 0.2470045 0.1924894 0.1904345 0.2057097 0.2063945

4*0.1134959 0.2262217 0.2332937 0.2070665 0.1962773 0.08857616 0.1134959

0.2351699 0.221197 0.2491324 0.2070665 0.2503512 0.2035733 0.1944332

0.2380944 0.1134959 0.2507485 0.2526715 0.2487168 0.2541391 0.1134959

0.2269949 0.2313142 2*0.08857616 0.188255 0.1859347 0.1134959 0.2495433

0.252295 0.2519146 0.2495433 0.2470045 0.2028321 0.2070665 0.1962773

0.2070665 4*0.1134959 0.2345553 0.2246236 0.188255 0.2028321 0.1904345

0.1134959 0.08857616 0.2269949 0.2491324 0.2108544 0.2495433 0.2035733

0.1859347 0.2237971 0.1134959 0.2503512 0.2470045 0.253044 0.2487168

0.08857616 0.2313142 0.2306294 0.1134959 0.08857616 0.1944332 0.1962773

0.08857616 0.2537777 0.2487168 0.2495433 0.08857616 0.1134959 0.201302

0.2057097 6*0.1134959 0.2229511 2*0.2220848 0.2070665 0.2057097 2*0.08857616

0.2269949 2*0.2491324 0.2495433 0.08857616 0.1997039 0.2096356 0.2292194
0.1134959 2*0.08857616 3*0.1134959 0.23393 0.2306294 0.08857616 0.1944332
0.1134959 0.08857616 0.2499495 0.2511414 0.2507485 2*0.08857616 0.201302
0.2108544 5*0.1134959 0.2254315 0.2332937 0.2229511 0.2319862 0.2108544
0.1924894 2*0.1134959 0.2357741 0.2397413 0.2491324 0.253044 2*0.2057097
0.2035733 0.2306294 0.1134959 0.08857616 0.1134959 0.08857616 0.1134959
0.08857616 0.2237971 0.2397413 0.08857616 0.1924894 0.208374 0.08857616
0.2526715 0.2511414 0.2507485 0.1924894 0.2096356 0.1904345 0.2042997
2*0.1134959 0.2499495 0.2511414 0.1134959 0.2326459 0.2277518 0.2220848
0.2380944 0.08857616 0.1962773 0.08857616 0.2262217 0.2363684 0.2319862
0.2491324 0.2461167 0.201302 0.08857616 0.1962773 0.2028321 0.2326459
0.08857616 4*0.1134959 2*0.2345553 0.2229511 0.2277518 0.208374 0.1134959
0.2499495 0.2511414 0.2507485 0.1980314 0.1944332 0.2070665 0.201302
0.1134959 0.2511414 2*0.2470045 0.1134959 0.2299313 2*0.1134959 0.23393
0.1134959 0.1997039 0.2057097 0.2277518 0.2246236 0.08857616 0.2491324
0.1904345 0.1944332 0.08857616 0.1962773 0.1924894 0.2380944 0.1134959
0.08857616 0.1134959 3*0.08857616 0.1134959 0.2220848 0.2332937 0.1962773
0.1944332 0.2534128 0.2511414 0.2507485 0.1997039 0.1859347 0.2042997
0.1859347 0.1134959 0.2515301 0.2526715 0.253044 0.1134959 0.2306294
2*0.08857616 0.2375282 0.08857616 0.1997039 0.1924894 0.2319862 0.08857616
2*0.1904345 0.1944332 0.1924894 0.1134959 0.2057097 0.201302 0.2397413
0.2269949 0.2380944 3*0.08857616 0.2345553 0.2386518 0.2319862 0.2332937

0.2028321 0.2070665 0.2503512 0.2511414 0.08857616 0.2028321 0.201302

0.2108544 0.1859347 0.253044 0.2495433 0.2515301 2*0.1134959 0.2254315

0.23393 0.1134959 0.2392007 0.2363684 0.2057097 0.208374 0.2326459

0.1904345 0.1997039 0.2042997 0.2503512 0.2515301 0.08857616 0.1944332

0.188255 0.1904345 0.2269949 0.2363684 0.2237971 2*0.08857616 0.228493

0.2262217 0.08857616 2*0.1134959 0.1997039 0.2503512 2*0.08857616

0.2070665 0.2028321 0.1962773 0.1924894 0.2519146 0.2526715 0.2465633

2*0.1134959 0.2351699 0.2277518 0.1134959 0.2357741 0.2220848 0.2096356

0.1859347 0.221197 0.2096356 0.1962773 0.1944332 0.2461167 0.2487168

2*0.1134959 0.08857616 0.188255 0.08857616 0.2269949 2*0.1134959 0.2386518

0.221197 0.2277518 0.1134959 2*0.08857616 0.2096356 0.1134959 2*0.08857616

0.1962773 0.1924894 0.1134959 0.2070665 0.2482963 0.2519146 2*0.1134959

0.2345553 0.2351699 0.2380944 0.1134959 0.236953 0.2386518 0.2237971

2*0.2292194 0.2096356 0.2108544 2*0.201302 0.2108544 0.1944332 0.08857616

0.188255 0.08857616 0.1134959 0.228493 0.08857616 0.1134959 0.2237971

0.2254315 3*0.08857616 0.2042997 2*0.1134959 2*0.08857616 0.2028321

0.2042997 0.1134959 0.1962773 4*0.1134959 0.2357741 0.2332937 4*0.08857616

0.2332937 0.2246236 0.2332937 0.1134959 0.1980314 0.2028321 0.2070665

0.1980314 0.1924894 0.08857616 0.188255 0.2108544 0.08857616 0.1134959

0.2269949 0.23393 0.2299313 0.23393 0.08857616 0.2070665 0.1962773

0.188255 2*0.1134959 2*0.08857616 0.2108544 0.1962773 0.1134959 0.2070665

3*0.1134959 0.221197 0.236953 0.2363684 0.2503512 0.2491324 0.2495433

0.08857616 0.2397413 0.2357741 0.1134959 0.08857616 0.1944332 0.1924894
0.2534128 0.2474402 0.2461167 0.1980314 0.2096356 0.1904345 0.08857616
0.1134959 0.228493 0.236953 2*0.1134959 0.1904345 0.1962773 0.1980314
2*0.08857616 0.1134959 0.08857616 0.1859347 0.2070665 2*0.1134959
0.2042997 0.1980314 2*0.1134959 0.2269949 0.2306294 0.2363684 0.2503512
0.2487168 0.2470045 0.1134959 0.2299313 0.2345553 0.08857616 0.1980314
0.2070665 0.08857616 0.2515301 0.2487168 0.2465633 0.208374 0.08857616
0.1134959 0.08857616 0.1134959 2*0.2319862 0.08857616 0.1134959 0.188255
0.2478708 0.2503512 0.2515301 0.2461167 0.1134959 0.08857616 0.1944332
0.1859347 2*0.1134959 0.1980314 0.1859347 2*0.1134959 0.2357741 0.2363684
0.2237971 0.08857616 0.253044 0.1134959 0.2375282 0.2277518 0.08857616
0.2246236 0.1962773 0.1924894 0.1134959 0.2465633 0.2491324 0.2474402
0.2042997 0.1980314 0.2096356 0.08857616 0.2363684 0.08857616 0.1134959
0.2306294 0.2392007 0.2057097 0.2511414 2*0.2534128 0.2537777 0.1134959
0.08857616 0.2096356 0.201302 2*0.1134959 0.2042997 0.1904345 0.1944332
2*0.1134959 0.2306294 0.236953 0.1134959 0.08857616 0.2375282 0.23393
0.2397413 0.08857616 0.2269949 0.221197 0.1944332 0.08857616 0.2526715
0.252295 0.2461167 0.1944332 0.2096356 0.2042997 0.2220848 0.2313142
2*0.08857616 0.23393 0.2326459 0.1980314 0.2487168 0.253044 0.2515301
0.2526715 0.1134959 0.1944332 0.1924894 0.2070665 0.2028321 0.1134959
0.1924894 0.2028321 0.188255 3*0.1134959 0.2292194 0.1134959 0.08857616
0.2313142 0.2246236 0.2363684 0.2357741 0.08857616 0.2397413 0.2057097

0.1904345 0.08857616 2*0.1134959 0.2057097 2*0.08857616 0.2357741

3*0.08857616 0.2220848 0.2375282 0.08857616 0.2526715 0.2461167 0.2537777

0.2487168 0.1134959 0.201302 2*0.2096356 0.2108544 0.1134959 0.2057097

0.1904345 4*0.1134959 0.2237971 0.08857616 0.23393 0.2229511 0.08857616

0.236953 2*0.1134959 0.2357741 0.1962773 0.1980314 0.2108544 0.08857616

0.2108544 0.1859347 0.1134959 0.236953 0.2345553 0.1134959 0.221197

0.2363684 0.1134959 0.236953 0.08857616 0.1997039 0.08857616 0.1134959

0.08857616 0.2070665 0.1944332 0.208374 0.1134959 0.2028321 2*0.1134959

0.1944332 4*0.1134959 0.2254315 0.2363684 0.2229511 0.2254315 0.08857616

0.2254315 0.2351699 0.2319862 0.2375282 2*0.1134959 0.2028321 0.2042997

0.2108544 0.2096356 0.08857616 0.2380944 0.2386518 0.08857616 0.2380944

0.2386518 0.08857616 0.2292194 0.08857616 0.208374 2*0.08857616 3*0.1134959

0.188255 0.1134959 0.201302 0.1944332 0.1134959 2*0.1904345 2*0.1134959

0.228493 0.2357741 0.2375282 0.08857616 0.2262217 0.1134959 0.2461167

0.2511414 0.2386518 0.23393 0.08857616 0.1134959 0.252295 0.1944332

2*0.08857616 0.2057097 2*0.2254315 0.2299313 0.2357741 0.2220848 0.1134959

0.2332937 0.2345553 0.08857616 0.1904345 2*0.08857616 2*0.1134959

0.1904345 0.1134959 0.1924894 0.2070665 0.1134959 0.2042997 0.208374

2*0.1134959 0.2375282 0.2397413 0.2269949 0.1134959 0.2262217 0.1134959

0.2537777 0.2526715 0.2495433 0.2277518 0.1134959 0.08857616 0.2487168

0.1924894 2*0.08857616 2*0.201302 0.221197 0.2313142 0.2332937 2*0.1134959

2*0.08857616 0.2332937 0.2070665 0.1134959 0.08857616 2*0.1134959

0.2042997 0.1134959 0.253044 0.1924894 0.1134959 0.2057097 0.1997039

2*0.1134959 0.2363684 0.228493 0.2397413 0.1134959 0.2306294 0.08857616

0.2515301 0.2541391 0.2478708 0.2375282 0.2254315 0.1134959 0.2042997

0.1980314 0.08857616 2*0.1134959 0.1904345 0.2269949 0.2306294 0.221197

0.1134959 2*0.08857616 0.1134959 0.2269949 0.2229511 0.2096356 0.1134959

0.201302 0.1980314 0.1904345 0.2499495 0.2507485 0.2108544 0.201302

0.2042997 0.1904345 3*0.1134959 0.2319862 0.2326459 0.2526715 0.1134959

0.2491324 2*0.2495433 0.2482963 0.228493 0.2345553 0.08857616 0.2042997

0.1944332 0.2057097 2*0.1134959 0.08857616 0.2237971 0.08857616 0.2357741

0.2246236 3*0.1134959 0.2319862 0.2254315 0.1859347 0.08857616 0.1980314

0.1134959 0.2108544 0.2515301 0.2482963 2*0.2057097 0.1962773 0.1904345

3*0.1134959 0.2386518 0.236953 0.1134959 0.2491324 0.2534128 0.252295

0.2541391 0.2220848 0.08857616 0.2229511 2*0.08857616 0.1134959 0.2108544

0.1904345 2*0.1134959 0.23393 0.08857616 0.2277518 0.2299313 0.2237971

2*0.08857616 0.2351699 0.08857616 0.2070665 0.1134959 0.2028321 0.1134959

0.2042997 0.2526715 0.253044 0.2057097 0.1904345 0.2096356 0.201302

3*0.1134959 0.2262217 0.2220848 0.236953 0.1134959 0.253044 2*0.2470045

0.2220848 0.1134959 0.2326459 0.2254315 2*0.1134959 0.1962773 0.2070665

2*0.08857616 0.2319862 0.08857616 0.1134959 0.2269949 0.23393 0.2380944

0.2345553 0.23393 0.1134959 0.2096356 4*0.1134959 0.2461167 0.2478708

0.2057097 0.1944332 0.201302 0.2028321 2*0.1134959 0.2254315 0.2357741

0.1134959 0.2363684 2*0.1134959 0.2519146 0.2495433 0.2229511 0.1134959

0.228493 0.2299313 2*0.1134959 0.1859347 0.201302 2*0.08857616 0.23393

3*0.08857616 0.2313142 2*0.2397413 0.08857616 0.1859347 0.188255 0.08857616

0.253044 0.2495433 0.253044 0.2526715 0.2499495 0.1859347 0.1997039

0.1134959 0.1924894 2*0.1134959 0.2345553 0.23393 0.1134959 0.2319862

0.2277518 0.1134959 0.2499495 0.2306294 2*0.1134959 0.08857616 0.2363684

0.2254315 0.1134959 0.1980314 0.2042997 0.1134959 0.2277518 0.2375282

0.08857616 0.1134959 0.2386518 0.2375282 0.2515301 0.2526715 0.2470045

0.1859347 0.1134959 0.08857616 0.2474402 0.2495433 0.2482963 0.2507485

0.2070665 0.188255 0.201302 0.1134959 0.2028321 2*0.1134959 0.2306294

0.2375282 0.1134959 0.2397413 0.2357741 0.2332937 0.252295 0.228493

0.1134959 0.08857616 2*0.1134959 0.2299313 0.08857616 0.2070665 0.2375282

0.2319862 0.221197 0.2357741 0.08857616 0.1134959 0.2269949 0.2363684

0.2515301 0.2526715 0.2507485 0.1962773 0.1134959 0.08857616 2*0.2482963

0.2541391 0.2515301 0.2042997 0.1944332 2*0.1134959 0.208374 0.1134959

0.2254315 0.2292194 0.2380944 3*0.1134959 0.2351699 0.2478708 0.2220848

0.08857616 0.2526715 2*0.1134959 0.2363684 0.2254315 0.1962773 0.2313142

0.2332937 0.2237971 0.08857616 0.1134959 0.2351699 0.1134959 0.2397413

0.2515301 0.2526715 0.2465633 0.2096356 0.1904345 0.08857616 0.2461167

0.2519146 0.1134959 0.1944332 0.1904345 3*0.1134959 0.1980314 0.1134959

0.2357741 0.2269949 4*0.1134959 0.2220848 0.23393 0.2262217 0.1134959

0.2537777 0.2515301 0.2537777 0.2254315 0.2299313 0.201302 0.1134959

0.2262217 0.2319862 0.2246236 0.236953 0.2246236 0.1134959 0.2534128

0.2461167 0.2470045 0.2511414 0.252295 0.1944332 0.08857616 0.2465633

0.2470045 0.1134959 0.2070665 0.208374 0.2096356 2*0.1134959 0.1859347

0.1134959 0.2277518 0.2237971 0.1134959 0.2495433 0.2503512 0.1134959

0.2351699 0.236953 0.2246236 0.08857616 0.2465633 0.2491324 0.2465633

0.2299313 0.2363684 0.2057097 0.1980314 0.08857616 0.2380944 0.2392007

0.2313142 2*0.2478708 0.2503512 0.2507485 0.2482963 0.2507485 0.2503512

0.2108544 0.1859347 3*0.1134959 0.1924894 0.1134959 0.2108544 2*0.1134959

0.2042997 2*0.1134959 0.2332937 0.1134959 0.2534128 0.2515301 2*0.1134959

0.2292194 0.23393 0.08857616 0.2491324 0.2519146 0.253044 2*0.2363684

0.201302 0.2042997 0.08857616 0.2269949 2*0.2277518 0.2478708 0.1134959

0.2537777 0.253044 0.08857616 0.2541391 0.2491324 0.2028321 0.1944332

0.1134959 2*0.08857616 0.1134959 3*0.08857616 0.2028321 0.1997039

0.1585346 0.1464379 0.23393 0.2292194 0.2246236 0.1529927 0.2070665

0.1997039 0.1585346 0.2313142 0.2332937 0.2306294 0.208374 0.23393

0.1464379 0.1675697 0.2108544 0.2096356 0.1529927 0.1713576 2*0.1675697

0.1529927 0.1675697 0.1529927 0.1585346 0.2246236 0.221197 0.1585346

0.201302 0.1997039 2*0.08857616 2*0.1134959 0.08857616 0.1134959 0.08857616

0.2057097 0.208374 0.1464379 0.1675697 0.2326459 0.2363684 0.228493

0.1633353 0.1962773 0.1997039 0.1675697 0.2380944 0.2319862 0.2397413

0.2108544 0.236953 0.1713576 0.1529927 0.1904345 0.2096356 0.2541391

0.2511414 0.2507485 0.2478708 0.1675697 0.1464379 0.1585346 0.1529927

0.2313142 0.2262217 0.1464379 2*0.208374 2*0.1134959 3*0.08857616

2*0.1134959 0.2096356 0.1904345 0.1464379 0.1529927 0.2254315 0.221197

0.2269949 0.1529927 0.2096356 0.1997039 0.1713576 0.228493 0.2313142

0.2386518 0.2070665 0.2332937 0.1713576 0.1464379 0.1904345 0.2096356

0.2470045 0.2507485 0.2474402 0.2491324 2*0.1529927 0.1675697 0.1585346

0.2299313 0.2229511 0.1675697 0.1859347 0.1633353 0.08857616 2*0.2515301

0.2465633 0.2474402 0.2487168 0.2519146 0.1904345 0.1944332 0.1633353

0.1464379 0.2380944 0.2306294 0.2351699 0.1633353 0.2070665 0.1464379

0.1585346 0.1633353 0.2220848 0.2254315 0.2096356 0.2319862 0.2277518

0.1997039 0.1924894 0.1713576 0.2470045 0.2515301 0.2495433 0.2470045

0.1713576 0.1633353 3*0.1464379 0.2397413 0.2392007 0.2096356 0.1675697

0.1134959 0.2491324 0.2507485 0.2541391 0.2534128 0.2495433 0.2465633

0.2070665 0.2096356 0.1675697 0.1529927 0.1633353 0.2351699 0.2397413

0.188255 0.1962773 0.1713576 0.2277518 0.1529927 0.2375282 0.236953

0.1997039 0.2392007 0.1944332 0.2237971 0.201302 0.1713576 2*0.2526715

0.2470045 0.2474402 0.2487168 0.2519146 0.1464379 0.1585346 0.1713576

0.23393 0.2306294 0.1859347 0.1713576 0.1134959 0.2465633 0.2503512

0.2507485 0.2499495 0.2487168 0.2526715 0.1962773 0.2035733 0.1675697

0.1713576 0.1529927 0.2345553 0.2397413 0.1859347 0.1585346 0.1633353

0.2292194 0.1675697 0.2254315 0.2386518 0.2108544 0.2035733 0.2096356

0.2363684 0.1464379 0.2096356 0.2515301 0.2534128 0.253044 2*0.2515301

0.2507485 0.1585346 0.1633353 0.2319862 0.2254315 0.2220848 0.1962773

0.1464379 0.2470045 0.2511414 0.2470045 0.2534128 0.2515301 0.08857616

0.1134959 0.1980314 0.2042997 0.1713576 0.1633353 0.1464379 0.23393

0.2262217 0.1962773 0.2108544 0.1464379 0.2357741 0.2351699 0.2326459

0.2292194 0.2397413 0.2380944 0.1904345 0.1944332 0.236953 0.2096356

0.2541391 0.2537777 0.2519146 0.2495433 0.2534128 0.2537777 0.1585346

0.2397413 0.2357741 0.2386518 0.1633353 0.208374 0.1529927 0.2511414

0.2537777 0.2515301 0.2537777 0.252295 0.08857616 0.1980314 0.2070665

0.2050116 3*0.1134959 0.236953 0.2357741 2*0.2096356 0.2332937 0.221197

0.2229511 0.2220848 2*0.2363684 0.2345553 0.201302 0.2096356 0.2299313

0.1997039 0.1134959 0.2487168 0.2519146 0.2491324 0.2519146 0.2495433

0.08857616 0.2351699 0.2306294 0.2375282 0.08857616 2*0.1997039 0.2515301

2*0.2511414 0.2519146 0.2534128 0.08857616 0.2096356 0.2108544 0.2057097

3*0.1134959 0.2313142 0.2380944 0.1997039 0.2042997 0.2292194 0.08857616

0.1134959 0.228493 0.2375282 0.2491324 0.2042997 0.1859347 0.2035733

0.2237971 0.2345553 0.08857616 0.2461167 0.2511414 0.2482963 0.2507485

0.1134959 0.2375282 2*0.2220848 0.08857616 0.1134959 2*0.208374 0.2519146

0.2526715 2*0.2537777 0.2470045 0.1924894 0.1904345 0.2057097 0.2063945

4*0.1134959 0.2262217 0.2332937 0.2070665 0.1962773 0.08857616 0.1134959

0.2351699 0.221197 0.2491324 0.2070665 0.2503512 0.2035733 0.1944332

0.2380944 0.1134959 0.2507485 0.2526715 0.2487168 0.2541391 0.1134959

0.2269949 0.2313142 2*0.08857616 0.188255 0.1859347 0.1134959 0.2495433

0.252295 0.2519146 0.2495433 0.2470045 0.2028321 0.2070665 0.1962773

0.2070665 4*0.1134959 0.2345553 0.2246236 0.188255 0.2028321 0.1904345

0.1134959 0.08857616 0.2269949 0.2491324 0.2108544 0.2495433 0.2035733

0.1859347 0.2237971 0.1134959 0.2503512 0.2470045 0.253044 0.2487168

0.08857616 0.2313142 0.2306294 0.1134959 0.08857616 0.1944332 0.1962773

0.08857616 0.2537777 0.2487168 0.2495433 0.08857616 0.1134959 0.201302

0.2057097 6*0.1134959 0.2229511 2*0.2220848 0.2070665 0.2057097 2*0.08857616

0.2269949 2*0.2491324 0.2495433 0.08857616 0.1997039 0.2096356 0.2292194

0.1134959 2*0.08857616 3*0.1134959 0.23393 0.2306294 0.08857616 0.1944332

0.1134959 0.08857616 0.2499495 0.2511414 0.2507485 2*0.08857616 0.201302

0.2108544 5*0.1134959 0.2254315 0.2332937 0.2229511 0.2319862 0.2108544

0.1924894 2*0.1134959 0.2357741 0.2397413 0.2491324 0.253044 2*0.2057097

0.2035733 0.2306294 0.1134959 0.08857616 0.1134959 0.08857616 0.1134959

0.08857616 0.2237971 0.2397413 0.08857616 0.1924894 0.208374 0.08857616

0.2526715 0.2511414 0.2507485 0.1924894 0.2096356 0.1904345 0.2042997

2*0.1134959 0.2499495 0.2511414 0.1134959 0.2326459 0.2277518 0.2220848

0.2380944 0.08857616 0.1962773 0.08857616 0.2262217 0.2363684 0.2319862

0.2491324 0.2461167 0.201302 0.08857616 0.1962773 0.2028321 0.2326459

0.08857616 4*0.1134959 2*0.2345553 0.2229511 0.2277518 0.208374 0.1134959

0.2499495 0.2511414 0.2507485 0.1980314 0.1944332 0.2070665 0.201302

0.1134959 0.2511414 2*0.2470045 0.1134959 0.2299313 2*0.1134959 0.23393

0.1134959 0.1997039 0.2057097 0.2277518 0.2246236 0.08857616 0.2491324

0.1904345 0.1944332 0.08857616 0.1962773 0.1924894 0.2380944 0.1134959

0.08857616 0.1134959 3*0.08857616 0.1134959 0.2220848 0.2332937 0.1962773

0.1944332 0.2534128 0.2511414 0.2507485 0.1997039 0.1859347 0.2042997

0.1859347 0.1134959 0.2515301 0.2526715 0.253044 0.1134959 0.2306294

2*0.08857616 0.2375282 0.08857616 0.1997039 0.1924894 0.2319862 0.08857616

2*0.1904345 0.1944332 0.1924894 0.1134959 0.2057097 0.201302 0.2397413

0.2269949 0.2380944 3*0.08857616 0.2345553 0.2386518 0.2319862 0.2332937

0.2028321 0.2070665 0.2503512 0.2511414 0.08857616 0.2028321 0.201302

0.2108544 0.1859347 0.253044 0.2495433 0.2515301 2*0.1134959 0.2254315

0.23393 0.1134959 0.2392007 0.2363684 0.2057097 0.208374 0.2326459

0.1904345 0.1997039 0.2042997 0.2503512 0.2515301 0.08857616 0.1944332

0.188255 0.1904345 0.2269949 0.2363684 0.2237971 2*0.08857616 0.228493

0.2262217 0.08857616 2*0.1134959 0.1997039 0.2503512 2*0.08857616

0.2070665 0.2028321 0.1962773 0.1924894 0.2519146 0.2526715 0.2465633

2*0.1134959 0.2351699 0.2277518 0.1134959 0.2357741 0.2220848 0.2096356

0.1859347 0.221197 0.2096356 0.1962773 0.1944332 0.2461167 0.2487168

2*0.1134959 0.08857616 0.188255 0.08857616 0.2269949 2*0.1134959 0.2386518

0.221197 0.2277518 0.1134959 2*0.08857616 0.2096356 0.1134959 2*0.08857616

0.1962773 0.1924894 0.1134959 0.2070665 0.2482963 0.2519146 2*0.1134959

0.2345553 0.2351699 0.2380944 0.1134959 0.236953 0.2386518 0.2237971

2*0.2292194 0.2096356 0.2108544 2*0.201302 0.2108544 0.1944332 0.08857616

0.188255 0.08857616 0.1134959 0.228493 0.08857616 0.1134959 0.2237971

0.2254315 3*0.08857616 0.2042997 2*0.1134959 2*0.08857616 0.2028321

0.2042997 0.1134959 0.1962773 4*0.1134959 0.2357741 0.2332937 4*0.08857616

0.2332937 0.2246236 0.2332937 0.1134959 0.1980314 0.2028321 0.2070665
0.1980314 0.1924894 0.08857616 0.188255 0.2108544 0.08857616 0.1134959
0.2269949 0.23393 0.2299313 0.23393 0.08857616 0.2070665 0.1962773
0.188255 2*0.1134959 2*0.08857616 0.2108544 0.1962773 0.1134959 0.2070665
3*0.1134959 0.221197 0.236953 0.2363684 0.2503512 0.2491324 0.2495433
0.08857616 0.2397413 0.2357741 0.1134959 0.08857616 0.1944332 0.1924894
0.2534128 0.2474402 0.2461167 0.1980314 0.2096356 0.1904345 0.08857616
0.1134959 0.228493 0.236953 2*0.1134959 0.1904345 0.1962773 0.1980314
2*0.08857616 0.1134959 0.08857616 0.1859347 0.2070665 2*0.1134959
0.2042997 0.1980314 2*0.1134959 0.2269949 0.2306294 0.2363684 0.2503512
0.2487168 0.2470045 0.1134959 0.2299313 0.2345553 0.08857616 0.1980314
0.2070665 0.08857616 0.2515301 0.2487168 0.2465633 0.208374 0.08857616
0.1134959 0.08857616 0.1134959 2*0.2319862 0.08857616 0.1134959 0.188255
0.2478708 0.2503512 0.2515301 0.2461167 0.1134959 0.08857616 0.1944332
0.1859347 2*0.1134959 0.1980314 0.1859347 2*0.1134959 0.2357741 0.2363684
0.2237971 0.08857616 0.253044 0.1134959 0.2375282 0.2277518 0.08857616
0.2246236 0.1962773 0.1924894 0.1134959 0.2465633 0.2491324 0.2474402
0.2042997 0.1980314 0.2096356 0.08857616 0.2363684 0.08857616 0.1134959
0.2306294 0.2392007 0.2057097 0.2511414 2*0.2534128 0.2537777 0.1134959
0.08857616 0.2096356 0.201302 2*0.1134959 0.2042997 0.1904345 0.1944332
2*0.1134959 0.2306294 0.236953 0.1134959 0.08857616 0.2375282 0.23393
0.2397413 0.08857616 0.2269949 0.221197 0.1944332 0.08857616 0.2526715

0.252295 0.2461167 0.1944332 0.2096356 0.2042997 0.2220848 0.2313142

2*0.08857616 0.23393 0.2326459 0.1980314 0.2487168 0.253044 0.2515301

0.2526715 0.1134959 0.1944332 0.1924894 0.2070665 0.2028321 0.1134959

0.1924894 0.2028321 0.188255 3*0.1134959 0.2292194 0.1134959 0.08857616

0.2313142 0.2246236 0.2363684 0.2357741 0.08857616 0.2397413 0.2057097

0.1904345 0.08857616 2*0.1134959 0.2057097 2*0.08857616 0.2357741

3*0.08857616 0.2220848 0.2375282 0.08857616 0.2526715 0.2461167 0.2537777

0.2487168 0.1134959 0.201302 2*0.2096356 0.2108544 0.1134959 0.2057097

0.1904345 4*0.1134959 0.2237971 0.08857616 0.23393 0.2229511 0.08857616

0.236953 2*0.1134959 0.2357741 0.1962773 0.1980314 0.2108544 0.08857616

0.2108544 0.1859347 0.1134959 0.236953 0.2345553 0.1134959 0.221197

0.2363684 0.1134959 0.236953 0.08857616 0.1997039 0.08857616 0.1134959

0.08857616 0.2070665 0.1944332 0.208374 0.1134959 0.2028321 2*0.1134959

0.1944332 4*0.1134959 0.2254315 0.2363684 0.2229511 0.2254315 0.08857616

0.2254315 0.2351699 0.2319862 0.2375282 2*0.1134959 0.2028321 0.2042997

0.2108544 0.2096356 0.08857616 0.2380944 0.2386518 0.08857616 0.2380944

0.2386518 0.08857616 0.2292194 0.08857616 0.208374 2*0.08857616 3*0.1134959

0.188255 0.1134959 0.201302 0.1944332 0.1134959 2*0.1904345 2*0.1134959

0.228493 0.2357741 0.2375282 0.08857616 0.2262217 0.1134959 0.2461167

0.2511414 0.2386518 0.23393 0.08857616 0.1134959 0.252295 0.1944332

2*0.08857616 0.2057097 2*0.2254315 0.2299313 0.2357741 0.2220848 0.1134959

0.2332937 0.2345553 0.08857616 0.1904345 2*0.08857616 2*0.1134959

0.1904345 0.1134959 0.1924894 0.2070665 0.1134959 0.2042997 0.208374

2*0.1134959 0.2375282 0.2397413 0.2269949 0.1134959 0.2262217 0.1134959

0.2537777 0.2526715 0.2495433 0.2277518 0.1134959 0.08857616 0.2487168

0.1924894 2*0.08857616 2*0.201302 0.221197 0.2313142 0.2332937 2*0.1134959

2*0.08857616 0.2332937 0.2070665 0.1134959 0.08857616 2*0.1134959

0.2042997 0.1134959 0.253044 0.1924894 0.1134959 0.2057097 0.1997039

2*0.1134959 0.2363684 0.228493 0.2397413 0.1134959 0.2306294 0.08857616

0.2515301 0.2541391 0.2478708 0.2375282 0.2254315 0.1134959 0.2042997

0.1980314 0.08857616 2*0.1134959 0.1904345 0.2269949 0.2306294 0.221197

0.1134959 2*0.08857616 0.1134959 0.2269949 0.2229511 0.2096356 0.1134959

0.201302 0.1980314 0.1904345 0.2499495 0.2507485 0.2108544 0.201302

0.2042997 0.1904345 3*0.1134959 0.2319862 0.2326459 0.2526715 0.1134959

0.2491324 2*0.2495433 0.2482963 0.228493 0.2345553 0.08857616 0.2042997

0.1944332 0.2057097 2*0.1134959 0.08857616 0.2237971 0.08857616 0.2357741

0.2246236 3*0.1134959 0.2319862 0.2254315 0.1859347 0.08857616 0.1980314

0.1134959 0.2108544 0.2515301 0.2482963 2*0.2057097 0.1962773 0.1904345

3*0.1134959 0.2386518 0.236953 0.1134959 0.2491324 0.2534128 0.252295

0.2541391 0.2220848 0.08857616 0.2229511 2*0.08857616 0.1134959 0.2108544

0.1904345 2*0.1134959 0.23393 0.08857616 0.2277518 0.2299313 0.2237971

2*0.08857616 0.2351699 0.08857616 0.2070665 0.1134959 0.2028321 0.1134959

0.2042997 0.2526715 0.253044 0.2057097 0.1904345 0.2096356 0.201302

3*0.1134959 0.2262217 0.2220848 0.236953 0.1134959 0.253044 2*0.2470045

0.2220848 0.1134959 0.2326459 0.2254315 2*0.1134959 0.1962773 0.2070665

2*0.08857616 0.2319862 0.08857616 0.1134959 0.2269949 0.23393 0.2380944

0.2345553 0.23393 0.1134959 0.2096356 4*0.1134959 0.2461167 0.2478708

0.2057097 0.1944332 0.201302 0.2028321 2*0.1134959 0.2254315 0.2357741

0.1134959 0.2363684 2*0.1134959 0.2519146 0.2495433 0.2229511 0.1134959

0.228493 0.2299313 2*0.1134959 0.1859347 0.201302 2*0.08857616 0.23393

3*0.08857616 0.2313142 2*0.2397413 0.08857616 0.1859347 0.188255 0.08857616

0.253044 0.2495433 0.253044 0.2526715 0.2499495 0.1859347 0.1997039

0.1134959 0.1924894 2*0.1134959 0.2345553 0.23393 0.1134959 0.2319862

0.2277518 0.1134959 0.2499495 0.2306294 2*0.1134959 0.08857616 0.2363684

0.2254315 0.1134959 0.1980314 0.2042997 0.1134959 0.2277518 0.2375282

0.08857616 0.1134959 0.2386518 0.2375282 0.2515301 0.2526715 0.2470045

0.1859347 0.1134959 0.08857616 0.2474402 0.2495433 0.2482963 0.2507485

0.2070665 0.188255 0.201302 0.1134959 0.2028321 2*0.1134959 0.2306294

0.2375282 0.1134959 0.2397413 0.2357741 0.2332937 0.252295 0.228493

0.1134959 0.08857616 2*0.1134959 0.2299313 0.08857616 0.2070665 0.2375282

0.2319862 0.221197 0.2357741 0.08857616 0.1134959 0.2269949 0.2363684

0.2515301 0.2526715 0.2507485 0.1962773 0.1134959 0.08857616 2*0.2482963

0.2541391 0.2515301 0.2042997 0.1944332 2*0.1134959 0.208374 0.1134959

0.2254315 0.2292194 0.2380944 3*0.1134959 0.2351699 0.2478708 0.2220848

0.08857616 0.2526715 2*0.1134959 0.2363684 0.2254315 0.1962773 0.2313142

0.2332937 0.2237971 0.08857616 0.1134959 0.2351699 0.1134959 0.2397413

0.2515301 0.2526715 0.2465633 0.2096356 0.1904345 0.08857616 0.2461167

0.2519146 0.1134959 0.1944332 0.1904345 3*0.1134959 0.1980314 0.1134959

0.2357741 0.2269949 4*0.1134959 0.2220848 0.23393 0.2262217 0.1134959

0.2537777 0.2515301 0.2537777 0.2254315 0.2299313 0.201302 0.1134959

0.2262217 0.2319862 0.2246236 0.236953 0.2246236 0.1134959 0.2534128

0.2461167 0.2470045 0.2511414 0.252295 0.1944332 0.08857616 0.2465633

0.2470045 0.1134959 0.2070665 0.208374 0.2096356 2*0.1134959 0.1859347

0.1134959 0.2277518 0.2237971 0.1134959 0.2495433 0.2503512 0.1134959

0.2351699 0.236953 0.2246236 0.08857616 0.2465633 0.2491324 0.2465633

0.2299313 0.2363684 0.2057097 0.1980314 0.08857616 0.2380944 0.2392007

0.2313142 2*0.2478708 0.2503512 0.2507485 0.2482963 0.2507485 0.2503512

0.2108544 0.1859347 3*0.1134959 0.1924894 0.1134959 0.2108544 2*0.1134959

0.2042997 2*0.1134959 0.2332937 0.1134959 0.2534128 0.2515301 2*0.1134959

0.2292194 0.23393 0.08857616 0.2491324 0.2519146 0.253044 2*0.2363684

0.201302 0.2042997 0.08857616 0.2269949 2*0.2277518 0.2478708 0.1134959

0.2537777 0.253044 0.08857616 0.2541391 0.2491324 0.2028321 0.1944332

**$ Property: Permeability I (md) Max: 100 Min: 1

PERMI ALL

2	1	1	2	1	1	1	24	22	7	5	57
	50	44	6	27	22	7	53	56	52	28	57
	5	9	30	29	6	10	9	9	6	9	6
	7	44	40	7	23	22					

1	1	2	2	1	2	1	26	28	5	9	55
	61	49	8	20	22	9	64	54	67	30	62
	10	6	17	29	100	92	91	84	9	5	7
	6	53	46	5	28	28					
2	2	1	1	1	2	2	29	17	5	6	45
	40	47	6	29	22	10	49	53	65	27	56
	10	5	17	29	82	91	83	87	6	6	9
	7	51	42	9	15	8					
1	93	93	81	83	86	94	17	19	8	5	64
	52	59	8	27	5	7	8	41	45	29	54
	48	22	18	10	82	93	88	82	10	8	5
	5	5	67	66	29	9					
2	87	91	100	98	88	81	27	29	9	6	8
	59	67	16	20	10	48	6	63	62	22	66
	19	43	23	10	96	96	82	83	86	94	5
	7	10	57	52	15	10					
2	81	90	91	89	86	96	20	24.5	9	10	6
	58	67	15	7	8	50	9	45	65	30	24.5
	29	61	5	29	93	98	97	93	93	91	7
	8	54	45	41	20	5					
82	92	82	98	93	1	2	21	25	10	8	5
	57	46	20	30	5	60	59	55	50	67	64

	17	19	62	29	100	99	94	88	98	99	7
	67	60	65	8	28	6					
92	99	93	99	95	1	21	27	25.5	2	2	2
	62	60	29	29	56	40	42	41	61	61	58
	23	29	51	22	2	86	94	87	94	88	1
	59	52	63	1	22	22					
93	92	92	94	98	1	29	30	26	2	2	2
	53	64	22	25	50	1	2	49	63	87	25
	15	24.5	43	58	1	80	92	85	91	2	63
	41	41	1	2	28	28					
94	96	99	99	82	18	17	26	26.5	2	2	2
	2	46	56	27	20	1	2	59	40	87	27
	90	24.5	19	64	2	91	96	86	100	2	47
	53	1	1	16	15	2					
88	95	94	88	82	24	27	20	27	2	2	2
	2	58	44	16	24	17	2	1	47	87	30
	88	24.5	15	43	2	90	82	97	86	1	53
	52	2	1	19	20	1					
99	86	88	1	2	23	26	2	2	2	2	2
	2	42	41	41	27	26	1	1	47	87	87
	88	1	22	29	50	2	1	1	2	2	2
	57	52	1	19	2	1					

89	92	91	1	1	23	30	2	2	2	2	2
	45	56	42	54	30	18	2	2	60	67	87
	97	26	26	24.5	52	2	1	2	1	2	1
	43	67	1	18	28	1					
96	92	91	18	29	17	25	2	2	89	92	2
	55	48	41	64	1	20	1	46	61	54	87
	80	23	1	20	24	55	1	2	2	2	2
	58	58	42	48	28	2					
89	92	91	21	19	27	23	2	92	82	82	2
	51	2	2	57	2	22	26	48	44	1	87
	17	19	1	20	18	64	2	1	2	1	1
	1	2	41	56	20	19					
98	92	91	22	15	25	15	2	93	96	97	2
	52	1	1	63	1	22	18	54	1	17	17
	19	18	2	26	23	67	47	64	1	1	1
	58	65	54	56	24	27					
90	92	1	24	23	30	15	97	88	93	2	2
	45	57	2	66	61	26	28	55	17	22	25
	90	93	1	19	16	17	47	61	43	1	1
	49	46	1	2	2	22					
90	1	1	27	24	20	18	94	96	81	2	2
	59	48	2	60	41	29	15	40	29	20	19

	80	86	2	2	1	16	1	47	2	2	65
	40	48	2	1	1	29					
2	1	1	20	18	2	27	85	94	2	2	58
	59	64	2	62	65	43	50	50	29	30	23
	23	30	19	1	16	1	2	49	1	2	43
	45	1	1	1	25	2					
2	1	1	24	25	2	20	2	2	2	2	60
	56	1	1	1	1	56	44	56	2	21	24
	27	21	18	1	16	30	1	2	47	57	51
	57	1	27	20	16	2					
2	1	1	30	20	2	27	2	2	2	40	62
	61	90	87	88	1	67	60	2	1	19	18
	98	83	80	21	29	17	1	2	49	62	2
	2	17	20	21	1	1					
2	1	15	27	2	2	25	21	2	2	47	52
	61	90	86	82	2	51	58	1	21	27	1
	93	86	81	28	1	2	1	2	54	54	1
	2	16	84	90	93	80					
2	1	19	15	2	2	21	15	2	2	60	61
	43	1	97	2	63	48	1	44	20	18	2
	81	87	83	25	21	29	1	61	1	2	52
	66	26	92	98	98	99					

2 1 29 23 2 2 25 17 19 2 2 52
62 2 1 63 57 67 1 47 40 19 1
96 95 80 19 29 25 41 53 1 1 57
55 21 86 97 93 96
2 19 18 27 24 2 18 24 16 2 2 2
50 2 1 53 44 61 60 1 67 26 17
1 2 2 26 1 1 60 1 1 1 41
63 1 96 80 99 86
2 23 29 29 30 2 26 17 2 2 2 2
43 1 57 42 1 62 2 2 60 20 21
30 1 30 15 2 62 58 2 40 61 2
62 1 22 1 2 1
27 19 28 2 24 2 2 19 2 2 2 2
45 61 42 45 1 45 59 54 63 2 2
24 25 30 29 1 64 65 1 64 65 1
50 1 28 1 1 2
2 2 16 2 23 19 2 17 17 2 2 49
60 63 1 46 2 80 92 65 57 1 2
95 19 1 1 26 45 45 51 60 41 2
56 58 1 17 1 1
2 2 17 2 18 27 2 25 28 2 2 63
67 47 2 46 2 99 96 88 48 2 1

	86	18	1	1	23	23	40	53	56	2	2
	1	1	56	27	2	1					
2	2	25	2	97	18	2	26	22	2	2	61
	49	67	2	52	1	93	100	84	63	45	2
	25	21	1	2	2	17	47	52	40	2	1
	1	2	47	42	29	2					
23	21	17	89	91	30	23	25	17	2	2	2
	54	55	96	2	87	88	88	85	49	58	1
	25	19	26	2	2	1	43	1	60	44	2
	2	2	54	45	15	1					
21	2	30	93	85	26	26	20	17	2	2	2
	65	62	2	87	98	95	100	41	1	42	1
	1	2	30	17	2	2	57	1	48	51	43
	1	1	59	1	27	2					
24	2	25	96	97	26	17	29	23	2	2	2
	46	41	62	2	97	82	82	41	2	55	45
	2	2	20	27	1	1	54	1	2	47	57
	64	58	57	2	29	2					
2	2	2	80	84	26	19	23	24	2	2	45
	60	2	61	2	2	94	88	42	2	49	51
	2	2	15	23	1	1	57	1	1	1	53
	67	67	1	15	16	1					

97	88	97	96	89	15	22	2	18	2	2	58
	57	2	54	48	2	89	52	2	2	1	61
	45	2	21	25	2	48	63	1	2	65	63
	93	96	82	15	2	1					
83	88	85	91	27	16	23	2	24	2	2	52
	63	2	67	60	56	95	49	2	1	2	2
	51	1	27	63	54	40	60	1	2	47	61
	93	96	91	20	2	1					
85	85	100	93	25	19	2	2	28	2	45	50
	64	2	2	2	59	84	41	1	96	2	2
	61	45	20	53	56	43	1	2	59	2	67
	93	96	81	29	17	1					
80	94	2	19	17	2	2	2	21	2	60	47
	2	2	2	2	41	57	46	2	99	93	99
	45	51	23	2	46	54	44	62	44	2	98
	80	82	92	95	19	1					
81	82	2	27	28	29	2	2	15	2	48	43
	2	88	90	2	59	62	44	1	81	87	81
	51	61	26	21	1	64	66	53	84	84	90
	91	85	91	90	30	15					
2	2	2	18	2	30	2	2	25	2	2	56
	2	98	93	2	2	50	57	1	87	94	97

	61	61	23	25	1	47	48	48	84	2	99
	97	1	100	87	24	19					
2	1	1	2	1	1	1	24	22	7	5	57
	50	44	6	27	22	7	53	56	52	28	57
	5	9	30	29	6	10	9	9	6	9	6
	7	44	40	7	23	22					
1	1	2	2	1	2	1	26	28	5	9	55
	61	49	8	20	22	9	64	54	67	30	62
	10	6	17	29	100	92	91	84	9	5	7
	6	53	46	5	28	28					
2	2	1	1	1	2	2	29	17	5	6	45
	40	47	6	29	22	10	49	53	65	27	56
	10	5	17	29	82	91	83	87	6	6	9
	7	51	42	9	15	8					
1	93	93	81	83	86	94	17	19	8	5	64
	52	59	8	27	5	7	8	41	45	29	54
	48	22	18	10	82	93	88	82	10	8	5
	5	5	67	66	29	9					
2	87	91	100	98	88	81	27	29	9	6	8
	59	67	16	20	10	48	6	63	62	22	66
	19	43	23	10	96	96	82	83	86	94	5
	7	10	57	52	15	10					

2	81	90	91	89	86	96	20	24.5	9	10	6
	58	67	15	7	8	50	9	45	65	30	24.5
	29	61	5	29	93	98	97	93	93	91	7
	8	54	45	41	20	5					
82	92	82	98	93	1	2	21	25	10	8	5
	57	46	20	30	5	60	59	55	50	67	64
	17	19	62	29	100	99	94	88	98	99	7
	67	60	65	8	28	6					
92	99	93	99	95	1	21	27	25.5	2	2	2
	62	60	29	29	56	40	42	41	61	61	58
	23	29	51	22	2	86	94	87	94	88	1
	59	52	63	1	22	22					
93	92	92	94	98	1	29	30	26	2	2	2
	53	64	22	25	50	1	2	49	63	87	25
	15	24.5	43	58	1	80	92	85	91	2	63
	41	41	1	2	28	28					
94	96	99	99	82	18	17	26	26.5	2	2	2
	2	46	56	27	20	1	2	59	40	87	27
	90	24.5	19	64	2	91	96	86	100	2	47
	53	1	1	16	15	2					
88	95	94	88	82	24	27	20	27	2	2	2
	2	58	44	16	24	17	2	1	47	87	30

	88	24.5	15	43	2	90	82	97	86	1	53
	52	2	1	19	20	1					
99	86	88	1	2	23	26	2	2	2	2	2
	2	42	41	41	27	26	1	1	47	87	87
	88	1	22	29	50	2	1	1	2	2	2
	57	52	1	19	2	1					
89	92	91	1	1	23	30	2	2	2	2	2
	45	56	42	54	30	18	2	2	60	67	87
	97	26	26	24.5	52	2	1	2	1	2	1
	43	67	1	18	28	1					
96	92	91	18	29	17	25	2	2	89	92	2
	55	48	41	64	1	20	1	46	61	54	87
	80	23	1	20	24	55	1	2	2	2	2
	58	58	42	48	28	2					
89	92	91	21	19	27	23	2	92	82	82	2
	51	2	2	57	2	22	26	48	44	1	87
	17	19	1	20	18	64	2	1	2	1	1
	1	2	41	56	20	19					
98	92	91	22	15	25	15	2	93	96	97	2
	52	1	1	63	1	22	18	54	1	17	17
	19	18	2	26	23	67	47	64	1	1	1
	58	65	54	56	24	27					

90	92	1	24	23	30	15	97	88	93	2	2
	45	57	2	66	61	26	28	55	17	22	25
	90	93	1	19	16	17	47	61	43	1	1
	49	46	1	2	2	22					
90	1	1	27	24	20	18	94	96	81	2	2
	59	48	2	60	41	29	15	40	29	20	19
	80	86	2	2	1	16	1	47	2	2	65
	40	48	2	1	1	29					
2	1	1	20	18	2	27	85	94	2	2	58
	59	64	2	62	65	43	50	50	29	30	23
	23	30	19	1	16	1	2	49	1	2	43
	45	1	1	1	25	2					
2	1	1	24	25	2	20	2	2	2	2	60
	56	1	1	1	1	56	44	56	2	21	24
	27	21	18	1	16	30	1	2	47	57	51
	57	1	27	20	16	2					
2	1	1	30	20	2	27	2	2	2	40	62
	61	90	87	88	1	67	60	2	1	19	18
	98	83	80	21	29	17	1	2	49	62	2
	2	17	20	21	1	1					
2	1	15	27	2	2	25	21	2	2	47	52
	61	90	86	82	2	51	58	1	21	27	1

	93	86	81	28	1	2	1	2	54	54	1
	2	16	84	90	93	80					
2	1	19	15	2	2	21	15	2	2	60	61
	43	1	97	2	63	48	1	44	20	18	2
	81	87	83	25	21	29	1	61	1	2	52
	66	26	92	98	98	99					
2	1	29	23	2	2	25	17	19	2	2	52
	62	2	1	63	57	67	1	47	40	19	1
	96	95	80	19	29	25	41	53	1	1	57
	55	21	86	97	93	96					
2	19	18	27	24	2	18	24	16	2	2	2
	50	2	1	53	44	61	60	1	67	26	17
	1	2	2	26	1	1	60	1	1	1	41
	63	1	96	80	99	86					
2	23	29	29	30	2	26	17	2	2	2	2
	43	1	57	42	1	62	2	2	60	20	21
	30	1	30	15	2	62	58	2	40	61	2
	62	1	22	1	2	1					
27	19	28	2	24	2	2	19	2	2	2	2
	45	61	42	45	1	45	59	54	63	2	2
	24	25	30	29	1	64	65	1	64	65	1
	50	1	28	1	1	2					

2 2 16 2 23 19 2 17 17 2 2 49
60 63 1 46 2 80 92 65 57 1 2
95 19 1 1 26 45 45 51 60 41 2
56 58 1 17 1 1
2 2 17 2 18 27 2 25 28 2 2 63
67 47 2 46 2 99 96 88 48 2 1
86 18 1 1 23 23 40 53 56 2 2
1 1 56 27 2 1
2 2 25 2 97 18 2 26 22 2 2 61
49 67 2 52 1 93 100 84 63 45 2
25 21 1 2 2 17 47 52 40 2 1
1 2 47 42 29 2
23 21 17 89 91 30 23 25 17 2 2 2
54 55 96 2 87 88 88 85 49 58 1
25 19 26 2 2 1 43 1 60 44 2
2 2 54 45 15 1
21 2 30 93 85 26 26 20 17 2 2 2
65 62 2 87 98 95 100 41 1 42 1
1 2 30 17 2 2 57 1 48 51 43
1 1 59 1 27 2
24 2 25 96 97 26 17 29 23 2 2 2
46 41 62 2 97 82 82 41 2 55 45

	2	2	20	27	1	1	54	1	2	47	57
	64	58	57	2	29	2					
2	2	2	80	84	26	19	23	24	2	2	45
	60	2	61	2	2	94	88	42	2	49	51
	2	2	15	23	1	1	57	1	1	1	53
	67	67	1	15	16	1					
97	88	97	96	89	15	22	2	18	2	2	58
	57	2	54	48	2	89	52	2	2	1	61
	45	2	21	25	2	48	63	1	2	65	63
	93	96	82	15	2	1					
83	88	85	91	27	16	23	2	24	2	2	52
	63	2	67	60	56	95	49	2	1	2	2
	51	1	27	63	54	40	60	1	2	47	61
	93	96	91	20	2	1					
85	85	100	93	25	19	2	2	28	2	45	50
	64	2	2	2	59	84	41	1	96	2	2
	61	45	20	53	56	43	1	2	59	2	67
	93	96	81	29	17	1					
80	94	2	19	17	2	2	2	21	2	60	47
	2	2	2	2	41	57	46	2	99	93	99
	45	51	23	2	46	54	44	62	44	2	98
	80	82	92	95	19	1					

81	82	2	27	28	29	2	2	15	2	48	43
	2	88	90	2	59	62	44	1	81	87	81
	51	61	26	21	1	64	66	53	84	84	90
	91	85	91	90	30	15					
2	2	2	18	2	30	2	2	25	2	2	56
	2	98	93	2	2	50	57	1	87	94	97
	61	61	23	25	1	47	48	48	84	2	99
	97	1	100	87	24	19					
2	1	1	2	1	1	1	24	22	7	5	57
	50	44	6	27	22	7	53	56	52	28	57
	5	9	30	29	6	10	9	9	6	9	6
	7	44	40	7	23	22					
1	1	2	2	1	2	1	26	28	5	9	55
	61	49	8	20	22	9	64	54	67	30	62
	10	6	17	29	100	92	91	84	9	5	7
	6	53	46	5	28	28					
2	2	1	1	1	2	2	29	17	5	6	45
	40	47	6	29	22	10	49	53	65	27	56
	10	5	17	29	82	91	83	87	6	6	9
	7	51	42	9	15	8					
1	93	93	81	83	86	94	17	19	8	5	64
	52	59	8	27	5	7	8	41	45	29	54

	48	22	18	10	82	93	88	82	10	8	5
	5	5	67	66	29	9					
2	87	91	100	98	88	81	27	29	9	6	8
	59	67	16	20	10	48	6	63	62	22	66
	19	43	23	10	96	96	82	83	86	94	5
	7	10	57	52	15	10					
2	81	90	91	89	86	96	20	24.5	9	10	6
	58	67	15	7	8	50	9	45	65	30	24.5
	29	61	5	29	93	98	97	93	93	91	7
	8	54	45	41	20	5					
82	92	82	98	93	1	2	21	25	10	8	5
	57	46	20	30	5	60	59	55	50	67	64
	17	19	62	29	100	99	94	88	98	99	7
	67	60	65	8	28	6					
92	99	93	99	95	1	21	27	25.5	2	2	2
	62	60	29	29	56	40	42	41	61	61	58
	23	29	51	22	2	86	94	87	94	88	1
	59	52	63	1	22	22					
93	92	92	94	98	1	29	30	26	2	2	2
	53	64	22	25	50	1	2	49	63	87	25
	15	24.5	43	58	1	80	92	85	91	2	63
	41	41	1	2	28	28					

94	96	99	99	82	18	17	26	26.5	2	2	2
	2	46	56	27	20	1	2	59	40	87	27
	90	24.5	19	64	2	91	96	86	100	2	47
	53	1	1	16	15	2					
88	95	94	88	82	24	27	20	27	2	2	2
	2	58	44	16	24	17	2	1	47	87	30
	88	24.5	15	43	2	90	82	97	86	1	53
	52	2	1	19	20	1					
99	86	88	1	2	23	26	2	2	2	2	2
	2	42	41	41	27	26	1	1	47	87	87
	88	1	22	29	50	2	1	1	2	2	2
	57	52	1	19	2	1					
89	92	91	1	1	23	30	2	2	2	2	2
	45	56	42	54	30	18	2	2	60	67	87
	97	26	26	24.5	52	2	1	2	1	2	1
	43	67	1	18	28	1					
96	92	91	18	29	17	25	2	2	89	92	2
	55	48	41	64	1	20	1	46	61	54	87
	80	23	1	20	24	55	1	2	2	2	2
	58	58	42	48	28	2					
89	92	91	21	19	27	23	2	92	82	82	2
	51	2	2	57	2	22	26	48	44	1	87

	17	19	1	20	18	64	2	1	2	1	1
	1	2	41	56	20	19					
98	92	91	22	15	25	15	2	93	96	97	2
	52	1	1	63	1	22	18	54	1	17	17
	19	18	2	26	23	67	47	64	1	1	1
	58	65	54	56	24	27					
90	92	1	24	23	30	15	97	88	93	2	2
	45	57	2	66	61	26	28	55	17	22	25
	90	93	1	19	16	17	47	61	43	1	1
	49	46	1	2	2	22					
90	1	1	27	24	20	18	94	96	81	2	2
	59	48	2	60	41	29	15	40	29	20	19
	80	86	2	2	1	16	1	47	2	2	65
	40	48	2	1	1	29					
2	1	1	20	18	2	27	85	94	2	2	58
	59	64	2	62	65	43	50	50	29	30	23
	23	30	19	1	16	1	2	49	1	2	43
	45	1	1	1	25	2					
2	1	1	24	25	2	20	2	2	2	2	60
	56	1	1	1	1	56	44	56	2	21	24
	27	21	18	1	16	30	1	2	47	57	51
	57	1	27	20	16	2					

2 1 1 30 20 2 27 2 2 2 40 62
61 90 87 88 1 67 60 2 1 19 18
98 83 80 21 29 17 1 2 49 62 2
2 17 20 21 1 1
2 1 15 27 2 2 25 21 2 2 47 52
61 90 86 82 2 51 58 1 21 27 1
93 86 81 28 1 2 1 2 54 54 1
2 16 84 90 93 80
2 1 19 15 2 2 21 15 2 2 60 61
43 1 97 2 63 48 1 44 20 18 2
81 87 83 25 21 29 1 61 1 2 52
66 26 92 98 98 99
2 1 29 23 2 2 25 17 19 2 2 52
62 2 1 63 57 67 1 47 40 19 1
96 95 80 19 29 25 41 53 1 1 57
55 21 86 97 93 96
2 19 18 27 24 2 18 24 16 2 2 2
50 2 1 53 44 61 60 1 67 26 17
1 2 2 26 1 1 60 1 1 1 41
63 1 96 80 99 86
2 23 29 29 30 2 26 17 2 2 2 2
43 1 57 42 1 62 2 2 60 20 21

	30	1	30	15	2	62	58	2	40	61	2
	62	1	22	1	2	1					
27	19	28	2	24	2	2	19	2	2	2	2
	45	61	42	45	1	45	59	54	63	2	2
	24	25	30	29	1	64	65	1	64	65	1
	50	1	28	1	1	2					
2	2	16	2	23	19	2	17	17	2	2	49
	60	63	1	46	2	80	92	65	57	1	2
	95	19	1	1	26	45	45	51	60	41	2
	56	58	1	17	1	1					
2	2	17	2	18	27	2	25	28	2	2	63
	67	47	2	46	2	99	96	88	48	2	1
	86	18	1	1	23	23	40	53	56	2	2
	1	1	56	27	2	1					
2	2	25	2	97	18	2	26	22	2	2	61
	49	67	2	52	1	93	100	84	63	45	2
	25	21	1	2	2	17	47	52	40	2	1
	1	2	47	42	29	2					
23	21	17	89	91	30	23	25	17	2	2	2
	54	55	96	2	87	88	88	85	49	58	1
	25	19	26	2	2	1	43	1	60	44	2
	2	2	54	45	15	1					

21	2	30	93	85	26	26	20	17	2	2	2
	65	62	2	87	98	95	100	41	1	42	1
	1	2	30	17	2	2	57	1	48	51	43
	1	1	59	1	27	2					
24	2	25	96	97	26	17	29	23	2	2	2
	46	41	62	2	97	82	82	41	2	55	45
	2	2	20	27	1	1	54	1	2	47	57
	64	58	57	2	29	2					
2	2	2	80	84	26	19	23	24	2	2	45
	60	2	61	2	2	94	88	42	2	49	51
	2	2	15	23	1	1	57	1	1	1	53
	67	67	1	15	16	1					
97	88	97	96	89	15	22	2	18	2	2	58
	57	2	54	48	2	89	52	2	2	1	61
	45	2	21	25	2	48	63	1	2	65	63
	93	96	82	15	2	1					
83	88	85	91	27	16	23	2	24	2	2	52
	63	2	67	60	56	95	49	2	1	2	2
	51	1	27	63	54	40	60	1	2	47	61
	93	96	91	20	2	1					
85	85	100	93	25	19	2	2	28	2	45	50
	64	2	2	2	59	84	41	1	96	2	2

	61	45	20	53	56	43	1	2	59	2	67
	93	96	81	29	17	1					
80	94	2	19	17	2	2	2	21	2	60	47
	2	2	2	2	41	57	46	2	99	93	99
	45	51	23	2	46	54	44	62	44	2	98
	80	82	92	95	19	1					
81	82	2	27	28	29	2	2	15	2	48	43
	2	88	90	2	59	62	44	1	81	87	81
	51	61	26	21	1	64	66	53	84	84	90
	91	85	91	90	30	15					
2	2	2	18	2	30	2	2	25	2	2	56
	2	98	93	2	2	50	57	1	87	94	97
	61	61	23	25	1	47	48	48	84	2	99
	97	1	100	87	24	19					

**$ Property: Pinchout Array Max: 1 Min: 1

**$ 0 = pinched block, 1 = active block

PINCHOUTARRAY CON 1

PRPOR 4800

CPOR 3e-6

MODEL BLACKOIL

TRES 140

PVT EG 1

**$ p	Rs	Bo	Eg	viso	visg	co
14.696	5.10911	1.03681	4.89173	1.74377	0.0118885	3e-005
80.3829	16.4175	1.04107	27.0004	1.63191	0.0119408	3e-005
146.07	29.3692	1.046	49.5142	1.52382	0.0120096	3e-005
211.757	43.4031	1.05142	72.4411	1.42512	0.0120897	3e-005
277.444	58.2658	1.05723	95.7885	1.33659	0.0121792	3e-005
343.131	73.8084	1.06337	119.563	1.25762	0.0122775	3e-005
408.818	89.9316	1.06983	143.77	1.1872	0.012384	3e-005
474.505	106.564	1.07656	168.414	1.12425	0.0124985	3e-005
540.191	123.651	1.08356	193.498	1.06781	0.0126209	3e-005
605.878	141.15	1.09081	219.022	1.017	0.0127513	3e-005
671.565	159.027	1.09829	244.986	0.971078	0.0128896	3e-005
737.252	177.253	1.106	271.387	0.929409	0.0130359	3e-005
802.939	195.803	1.11393	298.217	0.891451	0.0131904	3e-005
868.626	214.657	1.12206	325.469	0.856743	0.0133531	3e-005
934.313	233.798	1.1304	353.13	0.824896	0.0135242	3e-005
1000	253.209	1.13893	381.184	0.795574	0.0137038	3e-005
1760	493.893	1.25026	724.3	0.572387	0.0163979	2.45755e-005
2520	757.168	1.38146	1053.67	0.45555	0.0199521	1.54197e-005
3280	1037.28	1.52935	1321.07	0.383003	0.0237034	1.09554e-005

4040 1331.01 1.69188 1525.45 0.333177 0.0272185 8.36273e-006

4800 1636.27 1.86761 1682.55 0.296617 0.0303767 6.68949e-006

GRAVITY GAS 0.7

REFPW 4800

DENSITY WATER 62.6005

BWI 1.00377

CW 2.74599e-006

VWI 0.516363

CVW 0

**$ Property: PVT Type Max: 1 Min: 1

PTYPE CON 1

DENSITY OIL 51.4561

ROCKFLUID

RPT 1

**$ Sw krw krow

SWT

0 0 1

0.2 0 1

1 1 0

**$ Sl krg krog

SLT

0 1 0

```
        0.2         1         0
          1         0         1
INITIAL
USER_INPUT
**$ Property: Pressure (psi)   Max: 4800  Min: 4800
PRES CON         4800
**$ Property: Bubble Point Pressure (psi)   Max: 0  Min: 0
PB CON           0
**$ Property: Oil Saturation  Max: 0.8  Min: 0.8
SO CON         0.8
**$ Property: Water Saturation  Max: 0.2  Min: 0.2
SW CON         0.2
NUMERICAL
RUN
DATE 2012 1 1


**$
WELL  'Well-1'
PRODUCER 'Well-1'
OPERATE  MIN  BHP  1000.  CONT
** UBA      ff   Status  Connection
**          rad  geofac  wfrac  skin
```

```
GEOMETRY  K  0.25  0.37  1.0  0.0
PERF  GEOA  'Well-1'
** UBA      ff   Status  mohada1
36 28 1  1.0  OPEN    FLOW-TO  'SURFACE' REFLAYER
36 29 2  1.0  OPEN    FLOW-TO  1
36 30 2  1.0  OPEN    FLOW-TO  2
36 31 2  1.0  OPEN    FLOW-TO  3
36 32 2  1.0  OPEN    FLOW-TO  4
36 33 2  1.0  OPEN    FLOW-TO  5
36 34 2  1.0  OPEN    FLOW-TO  6
36 35 2  1.0  OPEN    FLOW-TO  7
36 36 2  1.0  OPEN    FLOW-TO  8
**$
**
WELL  'Well-2'
PRODUCER 'Well-2'
OPERATE  MIN  BHP  1000.0  CONT
**         rad  geofac  wfrac  skin
GEOMETRY  K  0.25  0.37  1.0  0.0
PERF  GEOA  'Well-2'
** UBA      ff   Status  mohada2
06 10 1  1.0  OPEN    FLOW-TO  'SURFACE' REFLAYER
```

```
06 11 2  1.0  OPEN    FLOW-TO  1
06 12 2  1.0  OPEN    FLOW-TO  2
06 13 2  1.0  OPEN    FLOW-TO  3
06 14 2  1.0  OPEN    FLOW-TO  4
06 15 2  1.0  OPEN    FLOW-TO  5
06 16 2  1.0  OPEN    FLOW-TO  6
06 17 2  1.0  OPEN    FLOW-TO  7
06 18 2  1.0  OPEN    FLOW-TO  8
**$
**
WELL  'Well-3'
PRODUCER 'Well-3'
OPERATE  MIN  BHP  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.25  0.37  1.0  0.0
PERF  GEOA  'Well-3'
** UBA       ff   Status  mohada3
06 36 1  1.0  OPEN    FLOW-TO  'SURFACE' REFLAYER
07 36 2  1.0  OPEN    FLOW-TO  1
08 36 2  1.0  OPEN    FLOW-TO  2
09 36 2  1.0  OPEN    FLOW-TO  3
10 36 2  1.0  OPEN    FLOW-TO  4
```

```
11 36 2  1.0  OPEN    FLOW-TO  5
12 36 2  1.0  OPEN    FLOW-TO  6
13 36 2  1.0  OPEN    FLOW-TO  7
14 36 2  1.0  OPEN    FLOW-TO  8
**$
**
WELL  'Well-4'
PRODUCER 'Well-4'
OPERATE  MIN  BHP  1000.0  CONT
**          rad  geofac  wfrac  skin
GEOMETRY  K  0.25  0.37  1.0  0.0
PERF  GEOA  'Well-4'
** UBA       ff   Status  mohada4
28 06 1  1.0  OPEN    FLOW-TO  'SURFACE' REFLAYER
29 06 2  1.0  OPEN    FLOW-TO  1
30 06 2  1.0  OPEN    FLOW-TO  2
31 06 2  1.0  OPEN    FLOW-TO  3
32 06 2  1.0  OPEN    FLOW-TO  4
33 06 2  1.0  OPEN    FLOW-TO  5
34 06 2  1.0  OPEN    FLOW-TO  6
35 06 2  1.0  OPEN    FLOW-TO  7
36 06 2  1.0  OPEN    FLOW-TO  8
```

DATE 2032 1 1.00000

STOP