Oil & Gas Science and Technology - Rev. IFP Energies nouvelles **74**, 42 (2019)

https://doi.org/10.2516/ogst/2019014

Available online at:
ogst.ifpenergiesnouvelles.fr

REGULAR ARTICLE OPEN ACCESS

On the prediction of pseudo relative permeability curves: meta-heuristics *versus* Quasi-Monte Carlo

Babak Fazelabdolabadi*, Hadi Bagherzadeh, Abbas Shahrabadi, and Sayed Ehsan Samimi

Institute of Enhanced Oil Recovery, Center for Exploration and Production Studies and Research, Research Institute of Petroleum Industry (RIPI), PO Box 14665-1998, Tehran, Iran

Received: 11 March 2018 / Accepted: 6 March 2019

Abstract. This article reports the first application of the Quasi-Monte Carlo (QMC) method for estimation of the pseudo relative permeability curves. In this regards, the performance of several meta-heuristics algorithms have also been compared *versus* QMC, including the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and the Artificial Bee Colony (ABC). The mechanism of minimizing the objective-function has been studied, for each method. The QMC has outperformed its counterparts in terms of accuracy and efficiently sweeping the entire search domain. Nevertheless, its computational time requirement is obtained in excess to the meta-heuristics algorithms.

1 Introduction

As an integral part of the economics of a field, the selection of an optimum development plan is greatly reliant on the accuracy of relative permeability information. Such information is practically provided by laboratory measurements on rock samples – to be later used as model inputs capable of capturing detailed geological heterogeneities. In practice, usage of models with length scales comparable to rock samples is hurdled; due to the computational time requirement. Consequently, effort is made to construct and use coarser models, with essentially larger block sizes, based on the information in the fine models. In this regards, the laboratory-measured relative permeabilities (rock curves) need to be adjusted, prior to applying to the coarser models. In other words, the relative permeability curves should be upscaled for the coarser models; otherwise possible errors may arouse because of disregarding the reservoir heterogeneity.

The earliest attempts on upscaling rock curves, used the concept of pseudo-functions (Coats *et al.*, 1971; Hearn, 1971; Jacks *et al.*, 1973; Kyte and Berry, 1975). Nevertheless, their further development was hindered because of two reasons. First, the dependency of pseudo-functions to the boundary conditions and the position of blocks in the coarse grid provide inevitable errors, when using them for larger models. Second, the difficulty faced in generating new pseudo-functions for each new scenario shall be practically insurmountable (Artus and Noetinger, 2004). As an alternative to the pseudo-function method, the History-Matching (HM) technique was later proposed for estimation of the pseudo relative permeability curves, for coarse models (Fayazi *et al.*, 2016; Johnson *et al.*, 1982; Tan, 1995; Wang *et al.*, 2009).

On a trial basis, the history-matching method seeks relative permeability curves (for a coarse model) that can reproduce data obtained from a corresponding model with a fine structure. Since its inception, machine search algorithms have been extensively used in the HM process, to reduce the time requirements for its implementation. In a sense, HM is a mathematical minimization problem that invokes optimization algorithms to find the appropriate pseudo relative permeability curves. For this sake, usage of optimization algorithms is rampant within the HM implementation.

The literature reports on usage of several optimization schemes for HM purposes, such as gradient-based and stochastic (Hajizadeh *et al.*, 2010; Zhang and Reynolds, 2002). In gradient-based algorithms, the search direction is determined by calculating the Hessian matrix or gradient of the objective function with respect to model parameters. Some of gradient-based algorithms include conjugate gradient, Gauss-Newton, Quasi-Newton and steepest descent. In spite of their favorable convergence, the gradient-based algorithms may be bounded to a local optimum in vicinity of the initial guess (Landa *et al.*, 2005). Moreover, these methods may not be appropriate for large-scale reservoirs, with complex objective functions (Hou *et al.*, 2012).

In comparison, stochastic methods are suitable for problems, with essentially complex objective functions. The well-known stochastic algorithms include the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony

* Corresponding author: fazelb@ripi.ir

Optimization (ACO) and Neighborhood Algorithm (NA). Finding global optimum through stochastic methods, however may require a large number of runs to reach an appropriate convergence – a potential drawback for field-scale purposes (Hou *et al.*, 2012). The application of numerical techniques, such as Ensemble Kalman Filter (EnKF) and Ensemble-based Optimization (EnOpt) have also been introduced to HM, to speed up on its convergence (Aanonsen and Naeval, 2009; Chen *et al.*, 2008).

The present article contributes to the existing literature in this field, in two venues. First, it reports on the first application of the Quasi-Monte Carlo (QMC) technique, for handling the optimization problem encountered in the HM process. Second, it provides a benchmark comparison between the QMC method and several meta-heuristics techniques.

The rest of the article is organized, as follows. The next section will mention the mathematical foundations as well as the algorithm details for the several optimization methods used. A description of our results is provided in the third section of the article, ensued by our concluding remarks.

2 Methods

The obtained pseudo-K_r value may alter significantly from the experimental counterparts, when applied to field. This is due to confluent effects of different phenomena – fine-grid heterogeneity, numerical dispersion, capillary pressure – (Fouda, 2016; Zarifi *et al.*, 2012). As such, the conventional K_r estimation technique, such as Corey method, may be ineffective in yielding accurate pseudo K_r values – deriving the need to adopt correlation-free methods with flexibility in prediction pseudo K_r values on a point-based framework. The method proposed herein possesses this feature and hence can be quite effective in capturing any complexity that arises due to field effects. The matter of finding the optimal relative permeability curves can be handled within an optimization context. In this sense, the problem can be equivalently expressed as finding the optimal value to an objective function, f, in an s-dimensional domain. In practice, the number of dimensions (to be considered) is the number of saturation points, at which data is sought for the corresponding relative permeability curves, K_{ro} and K_{rw}. In the present work, a domain of 18 dimensions was considered ($s = 18$) – a number of nine dimensions for each curve. The choice was rendered as we were interested in computing the relative permeability values at water saturations of $S_w = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$, for each curve. Consequently, a *point* may be conceived in an s-dimensional space, ω, with entries of the relative permeability curves at the considered water saturations (Eq. (1)):

$$\omega = (x_1, ..., x_s), \tag{1}$$

with $x_i = K_{ro}|_{S_{wi}}$ for $(1 \leq i \leq 9)$ and $x_i = K_{rw}|_{S_{wi}}$ for $(10 \leq i \leq s)$. As the point represents two curves simultaneously, it can be input into a reservoir simulator. Generally, the entire field production behavior including oil and water flow rates and pressure data *versus* time is used to achieve an acceptable match between the fine and coarse model during history matching (Lee and Seinfeld, 1987; Kulkarni and Datta-Gupta, 2000). Hence, output results for the Field-Oil-Production Rate (FOPR), Field-Pressure (FPR) and Field-Water-Cut-Total (FWCT), corresponding to the *point* being considered. The objective function incorporates these outputs, and sums the differences against a benchmark case (Eq. (2)),

$$\begin{aligned} f(\omega) = &\sum(|\mathrm{FOPR}_\omega - \mathrm{FOPR}_{\mathrm{benchmark}}| \\ &+ \sum(|\mathrm{FPR}_\omega - \mathrm{FPR}_{\mathrm{benchmark}}| \\ &+ \sum(|\mathrm{FWCT}_\omega - \mathrm{FWCT}_{\mathrm{benchmark}}|. \end{aligned} \tag{2}$$

Presumably, the relative permeability curves will be of *true* values, if the results of the benchmark case can be reproduced, out of the simulator. This, however, should correspond to the exact moment, when the value of the objective function reaches its global minimum. The original issue is therefore shrunk down to a minimization problem, in an s-dimensional space.

A plethora of optimization techniques were used to minimize f, as encountered in the prediction of relative permeability curves. We adopted meta-heuristics as well as QMC techniques. A description of the mathematical principle behind each method is given in the following lines.

2.1 Quasi-Monte Carlo

Let Ω be a separable topological space in an s-dimensional space. Clearly, any point in Ω can be described by a set of s values, $\omega = (x_1, x_2, \ldots, x_s)$ for $x_i \in \Re_{1 \leq i \leq s}$. Let f be a real-valued function on Ω, for which a global minimum is sought. Since f is assumed to hold a global minimum in the region of interest, it is bounded from below and we define its global minimum as:

$$m(f) = \min_{\omega \in \Omega} f(\omega). \tag{3}$$

Let λ be a probability measure on Ω. Furthermore, consider S as a sequence of N independent λ-distributed random samples $\omega_1, \omega_2, \ldots, \omega_N \in \Omega$. We define,

$$m_N(f; S) = \min_{1 \leq i \leq N} f(\omega_i). \tag{4}$$

The QMC method of *quasirandom search* makes use of a *deterministic* sequence of points $\omega_1, \omega_2, \ldots, \omega_N$ in Ω, in order to find the global infimum. It is proved that $m_N(f; S)$ converges to the global minimum of f with *unit* probability, if f is continuous and if a positive probability measure ($\lambda > 0$) is taken for every nonempty subset of Ω (Niederreiter, 1994),

$$\lim_{N \to \infty} m_N(f; S) = m(f). \tag{5}$$

Consider a point set $P = (\omega_1, \omega_2, \ldots, \omega_N)$. The *dispersion* of point P in Ω is defined by:

$$\mathrm{disp}_N(P; \Omega) = \max \quad \min_{1 \leq i \leq N} \quad \mathrm{disp}(\omega, \omega_i), \tag{6}$$

where,

$$\mathrm{disp}(\zeta, \gamma) = \max_{1 \leq i \leq s} |\zeta_i - \gamma_i| \qquad \text{for}$$

$$\zeta_i = (y_1, y_2, ..., y_s); \gamma_i = (z_1, z_2, ..., z_s). \quad (7)$$

The dispersion can be viewed as a measure of deviation from denseness of S in Ω (Niederreiter, 1994). Let $\Psi(f;t) = \max_{\zeta,\ \gamma\in\Omega,\ \mathrm{disp}(\zeta,\gamma)\le t}|f(\zeta)-f(\gamma)|\ t \ge 0$, be the *modulus of continuity* of f. It is proved (Niederreiter, 1994) that for any point set P of N points in Ω, with dispersion $\mathrm{disp}_N = \mathrm{disp}_N(P;\Omega)$, we have:

$$m(f) - m_N(f) \le \Psi(f;\mathrm{disp}_N), \quad (8)$$

if the metric space (Ω, disp) is bounded (*i.e.,* $\max_{\zeta,\ \gamma\in\Omega}\ \mathrm{disp}(\zeta,\gamma) < \infty$). Thus, the theorem shows point sets with *small* dispersion as being considered *suitable* for quasirandom search purposes (Niederreiter, 1994).

The fate of a quasirandom search largely depends on the condition $\mathrm{disp}_N < e$, in which e is a positive number less than 1/2 (Lei, 2002). Taking $I^s = [0, 1]^s$, an absolute low bound for dispersion of *any* N points in I^s is $(1/2)N^{-1/s}$ (Niederreiter, 1984). It later follows that N must at least be of an order of magnitude e^{-s} (Lei, 2002).

A typical point set used for the quasirandom search should also possess nice properties on its *discrepancy* – interpreted as the difference between the empirical distribution of the QMC point set and the uniform distribution (Drew and Homem-de-Mello, 2006). For a given point set $P = (\omega_1, \omega_2, ..., \omega_N)_{\omega_i\in I^s}$ and a subset $G \subset I^s$, we take $S_N(G)$ as the number of points $\omega_i \in G$, and define discrepancy as (Lei, 2002):

$$\mathrm{disp}(\omega_1, \omega_2, \ldots, \omega_N) = \max_{\tau\in I^s}\left|\frac{S_N(G_\tau)}{N} - \tau_1\tau_2, \ldots, \tau_s\right|, \quad (9)$$

where

$G_\tau = [0, \tau_1)\times,\ldots, \times[0, \tau_s)$ is a rectangular region with an s-dimensional volume equal to $\tau_1, \tau_2,\ldots, \tau_s$.

QMC deals with infinite Low Discrepancy Sequences (LDS) – possessing the additional property that for arbitrary N the initial segments have relatively small discrepancy (Lei, 2002). The merits of LDS are twofold; they provide uniform sample points avoiding large gaps or clusters. In addition, they *know* about their predecessors and fill the gaps left from the previous iterations (Kucherenko, 2006) – eliminating empty areas in which *no* information on the behavior of the underlying problem can be deducted.

The choice of LDS is therefore central to the QMC methodology. Different principles have been put to generate LDS sets (Bratley *et al.*, 1992; Niederreiter, 1984, 1987, 1994; Sobol, 1976). We adopted the methodology developed by Sobol (1976). While other theories, such as Niederreiter's (1994) result in the better asymptotic properties (Kucherenko, 2006), Sobol's LDS sets provide enhanced reliability in terms of rapid convergence in high dimensionality situations (Jäckel, 2002). A description of the Sobol's methodology for generating low-discrepancy sequences is, however, deferred to the Appendix A; to keep this text within a reasonable length.

Once an LDS set is available, the *multistart* QMC algorithm implements an inexpensive local search (such as the steepest descent) on the quasirandom points to concentrate the sample, which will subsequently be reduced by replacing the worse points (with higher function values) with the new quasirandom points. A completely new local search will then be applied to any point retained for a certain number of iterations. Two types of stopping criteria may be conceived for this algorithm. First, if no new value for the local minimum is found after several iterations (Glover and Kochenberger, 2003). Second, if the Number of Worse Stationary Points (NWSP) exceeds the Number of Stationary Points (NSP), usually by a fraction of three (Hickernell and Yuan, 1997). The reader is referred to the Appendix B of the article, for a more detailed description of the QMC algorithm.

2.2 Artificial Bee Colony

Originally inspired by the foraging behavior of a honeybee colony, the Artificial Bee Colony (ABC) algorithm consists of three fundamental components: food source, employed bees, and unemployed bees. For an explanation of the ABC method, however, we stick to the original terminology proposed by Karaboga and Akay (2009), so as to conform.

Then employed bees are those employed at, and currently exploiting, a certain food source. They carry information about the (distance, direction and the profitability) of the food source and communicate the information with other bees waiting at the hive. The unemployed bees are classified as being either a scout bee, or an onlooker one. The former randomly searches the environment to find a new (better) food source; while the latter seeks a food source by means of the information passed by an employed bee. An employed bee whose food source is depleted becomes a scout bee, and starts to search for a new food source. The method further assumes the number of employed bees in the colony, to be equal to the number of food sources (Karaboga and Akay, 2009). In practice, the position of a food source represents a possible solution to the optimization problem; whereas the amount of a food source corresponds to the quality (fitness) of the associated solution.

Once a sample of N (trial) solution points is generated within the search domain, $\omega_{i;1\le i\le N}$, its entries (food source positions) are subject to repeated ABC cycles, unless a termination criterion is not satisfied. Each ABC cycle entails tasks performed by each of the bee types, in which both global and local searches and selections are implemented. Since the tasks performed by each bee type is naturally different, they can be explained separately, as follows.

An employed bee generates a candidate solution point for the ith food source to which it has been assigned, $\omega^{\mathrm{iter}}_{\mathrm{perturb},i}$ for $1 \le i \le N$, by *perturbing* the (old) solution point, ω^{iter}_i.

$$\omega^{\mathrm{iter}}_{\mathrm{perturb},i} = \omega^{\mathrm{iter}}_i + \mathrm{rand}[-1,+1](\omega^{\mathrm{iter}}_i - \omega^{\mathrm{iterr}}_i) \quad (10)$$

where, iterr $\in \{1,\ldots, \mathrm{iter}\}$ (iter $\ne$ iterr), is a randomly-selected index, and rand $[-1, +1]$ is a random number between $[-1, +1]$, which acts as a scaling factor. The employed bee later makes a selection between $\omega^{\mathrm{iter}}_{\mathrm{perturb},i}$, and ω^{iter}_i, by the one which holds a lower function value, and refreshes its memory of the best solution point.

An onlooker bee assigns itself to a solution point (food source), with a *probabilistic* selection scheme. The scheme requires the *probability* values associated with food sources, as the input. The probability measure associated to the ith food source, prob_i, is obtained by:

$$\text{prob}_i = \frac{\text{fit}_i}{\sum_{j=1}^{\text{iter}} \text{fit}_j}, \tag{11}$$

with,

$$\text{fit}_i = \begin{cases} \dfrac{1}{1+f(\omega_i^{\text{iter}})} & (f_i \geq 0) \\ 1+\left|f(\omega_i^{\text{iter}})\right| & (f_i < 0) \end{cases}. \tag{12}$$

Assignment of an onlooker bee to a given food source (solution point) is sanctioned, provided that the probability of that solution point becomes greater than a randomly-selected number in the range [0, 1]. Once assigned, an onlooker bee should perturb its solution point (Eq. (10)), in search for an improved local minimum. Needless to mention that the perturbations are damped, as the point converges.

A scout bee should probe the s-dimensional space randomly, in search for a *global* solution point. A scout bee is unrestricted in the sense that it can adopt any location (point), as long as it remains inside the search space boundaries. This is opposed to the strategy taken by the employed or onlooker bees, because they are only allowed to select points generated *around* a previous solution point. On the condition that a local solution point cannot be improved further after a given number of iterates, it will be *abandoned*, and the functionality of the employed bees assigned to that point will be converted to a scout-type. In practice, the limit for this conversion is taken as half of the number of unemployed bees, in each dimension.

The possibility of being trapped in a local minimum shall be minimal in ABC, as scout bees independently search the entire search space; while other bee types simultaneously probe their local candidates for the global minimum point. The algorithm to perform an ABC optimization is listed in Appendix C. For the sake of ABC optimization, we considered $N = 18$ solution points, at each iterate.

2.3 Particle Swarm Optimization

Discovered through simulation of a social model by Kennedy and Eberhart (1995), the PSO is a population-based stochastic optimization procedure. The algorithm deals with repeated operations on a *swarm* – sample of N (trial) solution points within the search domain, $\omega_{i;1\leq i\leq N}$. The points (or *particles* as named in the PSO) are continuously perturbed, in each dimension, using a *velocity* term, ω_{velocity}. The velocity term for a dimension in a given point, is updated, during each iteration, by combining the information of the (previous) velocity of that point, with the values of its current distance measured against its best-recorded location, as well as the location of the global minimum point discovered thus far (Eq. (13)):

$$\begin{aligned} \omega_{\text{velocity}_{i,j}}^{\text{iter}+1} = {} & \text{coef}_1 \times \omega_{\text{velocity}_{i,j}}^{\text{iter}} + \text{coef}_2 \times \text{rand}_1 \\ & \times \left(\omega_{\text{best}_{i,j}} - \omega_{i,j}^{\text{iter}}\right) + \text{coef}_3 \times \text{rand}_2 \\ & \times \left(\omega_{\text{FBEST}_{i,j}} - \omega_{i,j}^{\text{iter}}\right), \end{aligned} \tag{13}$$

where $\text{coef}_{i;1<i\leq 3}$ are some input parameters to PSO, $\text{rand}_{i;1\leq i\leq 2}$ are random coefficients derived from a uniform probability distribution in the range [0,1], ω_{best} is the best-recorded location for the point (particle), and ω_{FBEST} represents the point with the global minimum value, recorded up to the iterate. In equation (13), the first and second indices, i and j, refer to the point and dimension numbers, respectively. Having perturbed the solution points, the information of the best performances in the swarm is renewed, after each cycle. The algorithm stops upon satisfaction of a stopping criterion. The algorithm is explained in Appendix D. Implementation of the PSO in this article was carried out, using parameters $N = 40$, $\text{coef}_1 = 0.72134$, $\text{coef}_2 = 1.1931$, $\text{coef}_3 = 1.1931$.

2.4 Genetic Algorithm

Perhaps the mostly-used amongst the meta-heuristics methods, the Genetic Algorithm (GA) is based on two basic operations – crossover and mutation. The former feeds in two solution points (or *individuals* as named in GA) and creates a single point by mixing their features together. The latter assigns a random value to one of the features. As a result, the main parameters consist of two probability measures for each of the operations. We shall refer the reader to the available literature (Coley, 1999; Haupt and Haupt, 2004; Simon, 2013; Sivanandam and Deepa, 2007; Yu and Gen, 2010), to gain in-depth knowledge on the method. Our optimization results for the GA, were produced using a probability measure of 0.8 (for crossover), and 0.1 (for mutation). In this sake, $N = 50$ points were used.

3 Results and discussions

3.1 Model description

The fine model considered herein, is a three-dimensional, heterogeneous and anisotropic reservoir model with a total number of 9000 (30 × 30 × 10) grid blocks. Two distinct zones with different petrophysical properties have been considered in the fine grid model. The porosity and permeability are distributed randomly on the fine grid using a truncated normal distribution based on the parameters presented in Table 1. No-flow condition is assumed at the boundaries of the model. The reservoir is initially at an oil saturation of 0.75. A water flooding process in one quarter of a five spot pattern is simulated. An injection well and a production well are placed in the opposite corners of the reservoir model. Both wells are completed through all the reservoir thickness, with no mechanical skin. The black oil reservoir simulator ECLIPSE 100™ was used, for the all

Table 1. The porosity and permeability distribution parameters in the fine-grid model.

Zones	Layers	Property	Min.	Max.	Mean	Standard deviation
Top zone (1)	1–5	Porosity	0.2	0.3	0.25	0.01
		Permeability	30	80	50	5
Bottom zone (2)	6–10	Porosity	0.15	0.25	0.20	0.01
		Permeability	5	50	20	5

Table 2. The fine-grid model characteristics.

Property	Value	Property	Value
Δx (ft)	50	$P_{\text{saturation}}$ (psi)	3000
Δy (ft)	50	ρ_{oil} @ std. cond. (lb/ft^3)	49.1
Δz (ft)	10	ρ_{water} @ std. cond. (lb/ft^3)	64.79
P_{initial} (psi)	5000	μ_{oil} @ 5000 psi (cp)	1.15
		μ_{water} @ 5000 psi (cp)	0.4

simulations conducted. The fine model properties used in the simulator are enlisted in Table 2. The set of relative permeability curves used in the fine model is also shown in Figure 1 – assuming only one rock type in the model. The coarse model considered comprised of 200 grid blocks ($10 \times 10 \times 2$). The porosity of the coarse grid was estimated through the volume weighted average (Eq. (14)) – dividing the total void volume of all the fine grid blocks building the coarse grid block, by the total bulk volume of these fine grid blocks,

$$\bar{\varphi} = \frac{\sum_k \sum_j \sum_i (V\varphi)}{\sum_k \sum_j \sum_i (V)}, \tag{14}$$

where V and φ are fine grid block volume and porosity, respectively.

As for the permeability estimation, a combination of the harmonic and arithmetic averages was used. For instance, the permeability in the x-direction was estimated by equation (15):

$$\bar{k}_x = \frac{\sum_i (\Delta X)}{\sum_i \frac{\Delta X}{\left(\frac{\sum_k \sum_j (k_x A)}{\sum_k \sum_j (A)} \right)}}, \tag{15}$$

where ΔX is size of fine grid block in x-direction, k_x is fine grid block permeability in x-direction and A is cross section of fine grid block perpendicular to x direction.

Figures 2a and b represent the permeability map for the fine and the coarse grid models, respectively. The porosity map for the fine and the coarse grid models are presented in Figure 3.

The stability of a water flooding depends on the total mobility ratio evaluated across the front (M_f) which is calculated by equation (16). According to Buckley Leverett

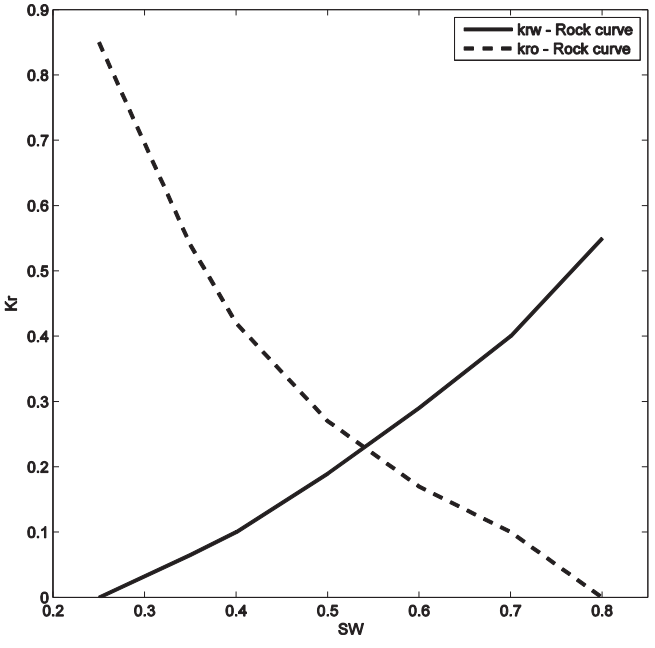

Fig. 1. The relative permeability curves used in the fine model.

calculation, $M_f = 0.876$ which means a stable displacement in the considered fine grid model. Figure 4 depicts some snapshots of saturation map given at breakthrough time at two views for the fine and coarse grid models. First one is the horizontal displacement in bottom zone. Second snapshot is vertical displacement from injection well to production well. It seems that horizontal displacement is uniform and stable which is in agreement with above-mentioned conclusion; however, vertical displacement shows some kind of instability because of contrast in top and bottom zones petrophysical properties as well as density of water.

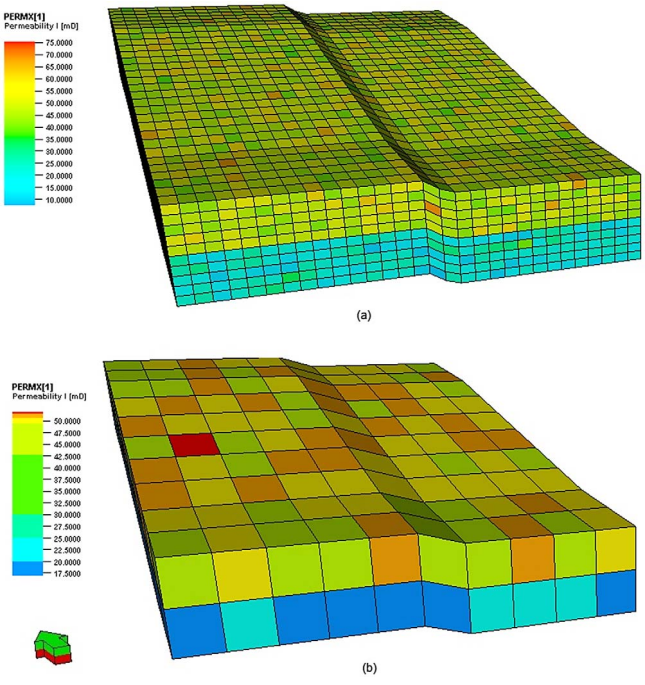

Fig. 2. The permeability map for the (a) fine-grid and (b) coarse-grid models.

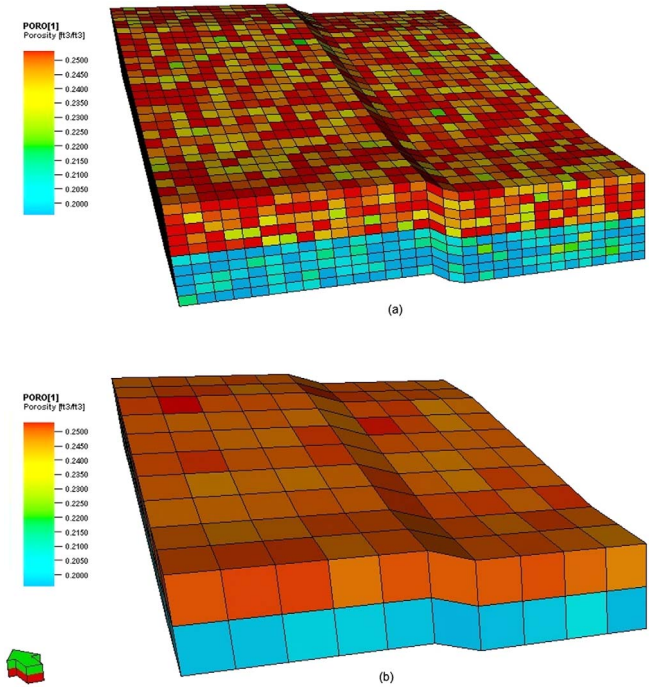

Fig. 3. The porosity map for the (a) fine-grid and (b) coarse-grid models.

$$M_f = \frac{\lambda(S_f)}{\lambda(S_{\min})} = \frac{\mu_o k_{\mathrm{rw}}(S_f) + \mu_w k_{\mathrm{ro}}(S_f)}{\mu_o k_{\mathrm{rw}}(S_{\min}) + \mu_w k_{\mathrm{ro}}(S_{\min})}. \quad (16)$$

3.2 Results

The results obtained from simulations of the fine-grid system are compared against the results obtained from the corresponding coarse-grid system, using different numerical algorithms adopted (GA, PSO, ABC, QMC). The results for the oil production rate, reservoir pressure/water-cut *versus* time are used for comparison. Figure 5 shows the computed oil production rate in both systems, over a period of 15 years. Clearly, all the methods tested, have been capable of accurately reproducing the field's oil production rate, as compared with the fine-grid case. The situation is somewhat different for the reservoir pressure estimations (Fig. 6). As it can be noticed, the PSO outputs are the closest to the fine-grid reservoir pressures; while the ABC results show the most deviations. With an exemption of ABC, all the methods have estimated satisfactory results for the water-cut, over the time span considered (Fig. 7).

The pseudo relative permeability curves estimated by each technique are presented in Figure 8. The deviation of the pseudo relative permeability curves from the rock curves is expected, owing to the altered heterogeneities and numerical dispersion in the coarse-grid system. On the other hand, the pseudo relative permeability curves estimated by methods adopted are seemingly identical – pinpointing the uniqueness of the relative permeability curves for the coarse-grid system considered. This finding should be logical, as the underlying physics related to the problem, should be independent of the type of the numerical algorithm being employed. The uniqueness of the generated pseudo relative permeability curves can also be interpreted in terms of the algorithms having successfully reached the global minimum in their entities.

In this work, the choice of a (new) trial relative permeability curve was made *directly*, by randomly selecting an entry from a pool of plausible choices. This pool consisted of nearly 500 ECLIPSE-acceptable curves generated in vicinity of the last optimum curve detected, during each iterate. This renders model representation of relative permeability curves rather unnecessary. Our methodology also makes no use of the transformation of the control points, in the B-spline process (Chen *et al.*, 2008) – obviating the need to solve a system of linear equations.

As the size of the system (to be simulated) may increase in the field situation, the time requirement on the simulations may become an issue. Therefore, one may wish to find out the order of proximity towards the global optimum within each iterate. For this sake, understanding the mechanism of target approach is essential. To complement this work, we have also reported this mechanism for the numerical methods adopted (GA, PSO, ABC, QMC) in Figure 9. As evident from the figure, the QMC outputs have plummeted to the global optimum within few iterates, followed by the GA. On the other hand, the approaching mechanism of the PSO/ABC has been rather steadily decreasing. This behavior of the QMC method stems from its capability of efficiently searching the entire search space. The only drawback to QMC, however, remains in its time requirement

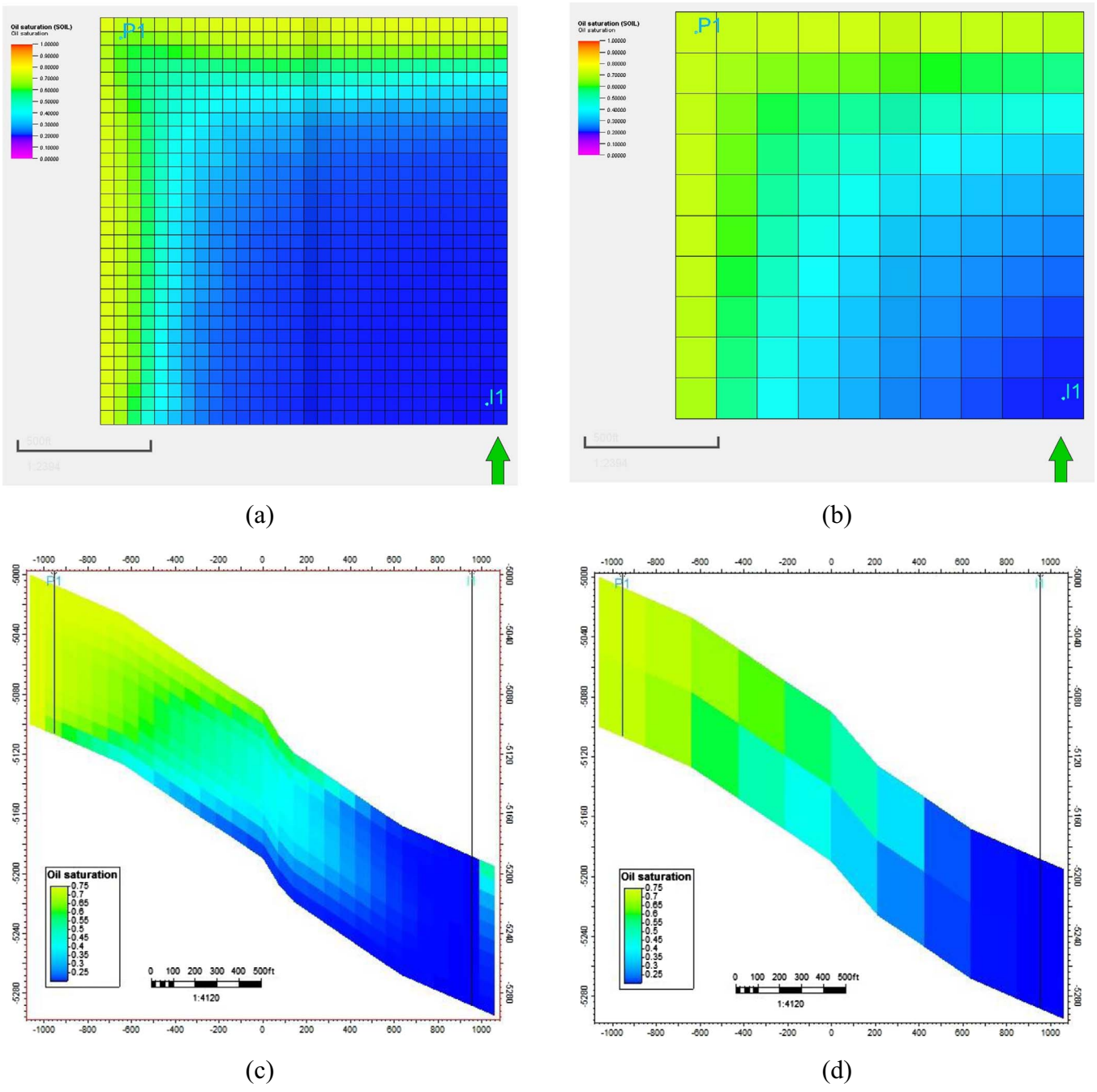

Fig. 4. The snapshots of saturation map for (a) horizontal displacement in fine grid, (b) horizontal displacement in coarse grid, (c) vertical displacement from injection to production well in fine grid and (d) vertical displacement from injection to production well in coarse grid.

(Fig. 10). For a given number of iterates, our results indicate that the QMC has taken more time to implement, compared to the meta-heuristics algorithms. We have also measured the maximum error encountered in each method, against the corresponding fine-grid results (Fig. 11). Since the computed reservoir pressure curve looks to have the most deviation – in all methods considered – this maximum error was practically taken as the percentage change in result in the worst situation, when compared with the corresponding pressure value from the fine-grid model. The QMC has again gained supremacy over the meta-heuristics methods used, in this regards.

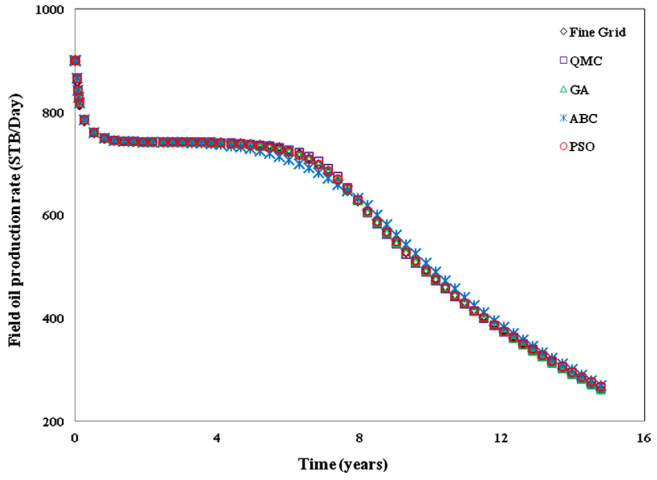

Fig. 5. The estimated oil production rates *versus* time.

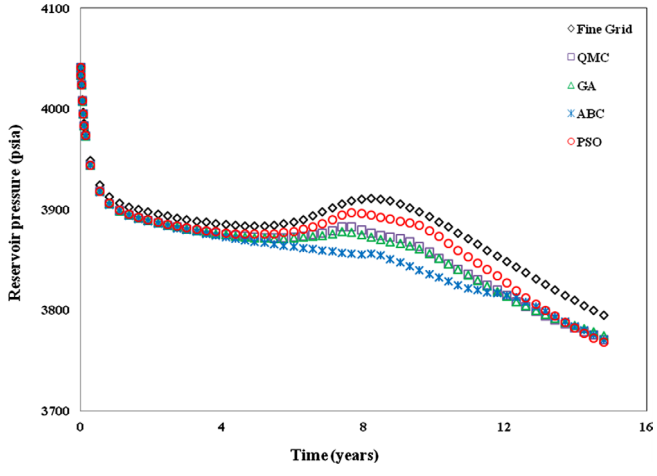

Fig. 6. The estimated reservoir pressures *versus* time.

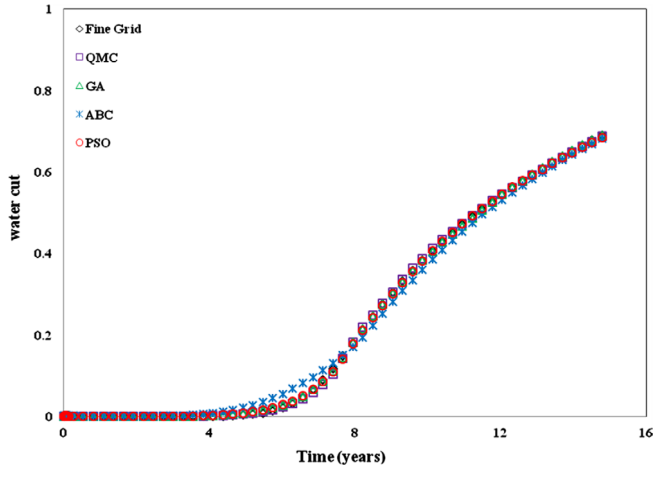

Fig. 7. The estimated water-cuts *versus* time.

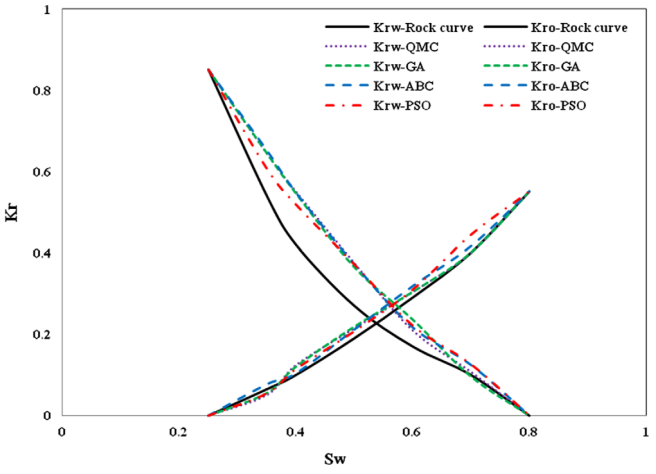

Fig. 8. The estimated pseudo relative permeability curves *versus* the rock curves.

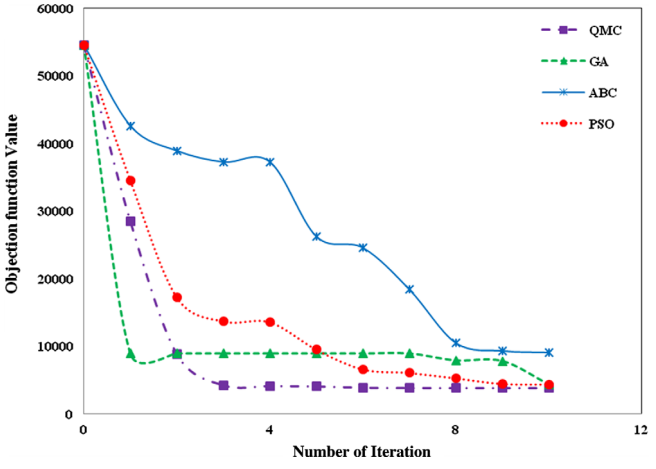

Fig. 9. The mechanism of objective-function minimization in the numerical methods adopted.

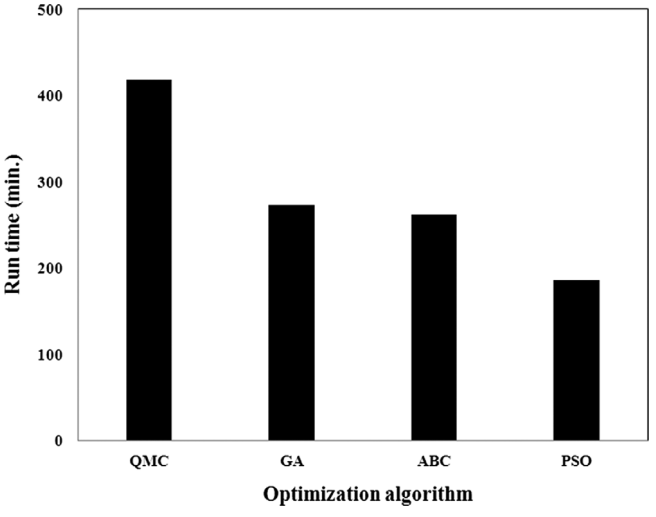

Fig. 10. The run-time of the numerical methods adopted.

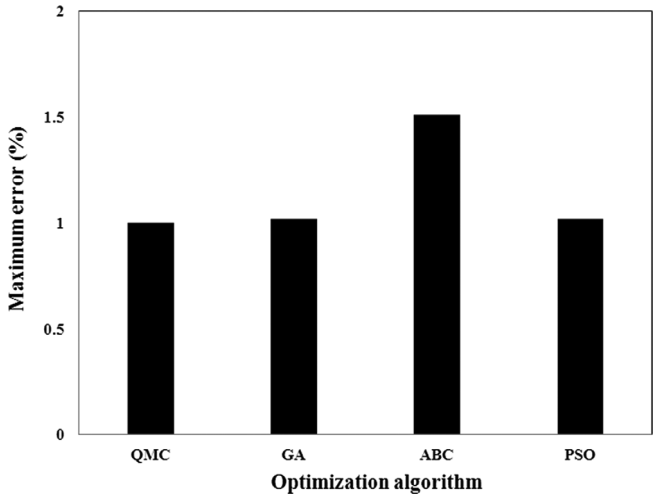

Fig. 11. Maximum error encountered in each numerical method, compared against the fine-grid results.

4 Conclusion

The application and performance of several meta-heuristics methods (GA, PSO, ABC) were investigated *versus* the QMC technique, for forecasting the pseudo relative permeability curves in a synthetic model. The QMC method outperforms in the meta-heuristics counterparts in the accuracy and the efficient search strategy of visiting the entire problem domain. Our methodology to randomly select the new trial relative permeability curve from a pool of simulator-acceptable curves in vicinity of the last optimum curve detected, has rendered the model representation of curves rather unnecessary. In addition, this strategy has obviated the need to solve a system of linear equations – commonly encountered in the method of transformation of the control points, in the B-spline scenario. The mechanism of the QMC approaching the global minimum – dropping suddenly to the proximity of the solution – should be taken as its other advantage. The only drawback to QMC though, resides in its computational time requirement, being higher than the meta-heuristics methods, as our results indicate.

References

Aanonsen S.I., Naeval G. (2009) The ensemble Kalman filter in reservoir engineering-a review, *SPE J.* **14**, 393–412. doi: 10.2118/117274-PA.

Artus V., Noetinger B. (2004) Up-scaling two-phase flow in heterogeneous reservoirs: current trends, *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles* **59**, 185–195.

Bratley P., Fox B.L., Niederreiter H. (1992) Implementation and tests of low discrepancy sequences, *ACM. T. Model. Comput. Simul.* **2**, 195–213. doi: 10.1145/146382.146385.

Chen Y., Oliver D.S., Zhang G.M. (2008) Efficient ensemble-based closed-loop production optimization, *SPE J.* **14**, 20–23. doi: 10.2118/112873-PA.

Coats K.H., Dempsey J.R., Henderson J.H. (1971) The use of vertical equilibrium in two-dimensional simulation of three-dimensional reservoir performance, *SPE J.* **11**, 63–71.

Coley D. (1999) *An introduction to genetic algorithms for scientists and engineers*, World Scientific Pub. Co., Inc., Singapore.

Drew S., Homem-de-Mello T. (2006) Quasi-Monte Carlo strategies for stochastic optimization, in: *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA, pp. 774–782.

Fayazi A., Bagherzadeh H., Shahrabadi A. (2016) Estimation of pseudo relative permeability curves for a heterogeneous reservoir with a new automatic history matching algorithm, *J. Pet. Sci. Eng.* **140**, 154–163. doi: 10.1016/j.petrol.2016.01.013.

Fouda M.A.G. (2016) Relative permeability upscaling for heterogeneous reservoir models, *PhD Dissertation*, Heriot-Watt University, Edinburgh, UK.

Glover F., Kochenberger G. (2003) *Handbook of metaheuristics*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

Hajizadeh Y., Demyanov V., Mohamed L., Christie M. (2010) Comparison of evolutionary and swarm intelligence methods for history matching and uncertainty quantification in petroleum reservoir models, in: *Intelligent Computational Optimization in Engineering*, Springer, Berlin/Heidelberg, Germany, pp. 209–240.

Haupt R.L., Haupt S.E. (2004) *Practical genetic algorithms*, 2nd edn., John Wiley & Sons, New York, NY.

Hearn C.L. (1971) Simulation of stratified water flooding by pseudo relative permeability curves, *J. Pet. Technol.* **23**, 805–813.

Hickernell F.J., Yuan Y. (1997) A simple multistart algorithm for global optimization, *OR Trans.* **1**, 2.

Hou J., Wang D., Luo F., Zhang Y.H. (2012) A review on the numerical inversion methods of relative permeability curves, *Procedia Eng.* **29**, 375–380. doi: 10.1016/j.proeng.2011.12.726.

Jäckel P. (2002) *Monte Carlo methods in finance*, John Wiley and Sons, New York, NY.

Jacks H.H., Smith O.J.E., Mattax C.C. (1973) The modeling of a three-dimensional reservoir with a two-dimensional reservoir simulator – the use of dynamic pseudo functions, *SPE J.* **13**, 175–185.

Johnson J.B., Nanney M.M., Killough J.E., Lin Y.T. (1982) The Kuparuk River field: a regression approach to pseudo-relative permeabilities, *SPE* **10531**. doi: 10.2118/10531-MS.

Karaboga D., Akay B. (2009) A comparative study of Artificial Bee Colony algorithm, *Appl. Math. Comput.* **214**, 1, 108–132. doi: 10.1016/j.amc.2009.03.090.

Kennedy J., Eberhart R. (1995) Particle swarm optimization, *Proc. IEEE Int. Conf. Neural Netw.* **IV**, 1942–1948. doi: 10.1109/ICNN.1995.488968.

Kucherenko S. (2006) Application of Quasi Monte Carlo methods in global optimization, *Glob. Optim.* **5**, 111–133. doi: 10.1007/0-387-30528-9_5.

Kulkarni K.N., Datta-Gupta A. (2000) Estimating relative permeability from production data: a streamline approach, *SPE J.* **5**, 4, 402–411. doi: 10.2118/66907-PA.

Kyte J.R., Berry D.W. (1975) New pseudo functions to control numerical dispersion, *SPE J.* **15**, 269–276.

Landa J., Kalia R.K., Nakano A., Nomura K., Vashishta P. (2005) History match and associated forecast uncertainty analysis-practical approaches using cluster computing, *International Petroleum Technology Conference*, 21–23 November, Doha, Qatar. *IPTC*-**10751**-MS. doi: 10.2523/IPTC-10751-MS.

Lee T., Seinfeld J.H. (1987) Estimation of absolute and relative permeabilities in petroleum reservoirs, *Inverse Probl.* **3**, 4, 711–728. doi: 10.1088/0266-5611/3/4/015.

Lei G. (2002) Adaptive random search in Quasi-Monte Carlo methods for global optimization, *Comput. Math. Appl.* **43**, 747–754. doi: 10.1016/S0898-1221(01)00318-2.

Niederreiter H. (1984) On the measure of denseness for sequences, in: *Topics of classical number theory, Colloquia Math. Sot. Janos Bolyai.* **34**, North Holland, Amsterdam, 1163–1208.

Niederreiter H. (1987) Point sets and sequences with small discrepancy, *Monatsch. Math.* **104**, 273–337. doi: 10.1007/bf01294651.

Niederreiter H. (1994) *Random number generation and Quasi-Monte Carlo methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Simon D. (2013) *Evolutionary optimization algorithms*, John Wiley & Sons, Hoboken, New Jersey.

Sivanandam S., Deepa S. (2007) *Introduction to genetic algorithms*, Springer-Verlag, Berlin, Heidelberg, Germany.

Sobol I.M. (1976) Uniformly distributed sequences with an additional uniform property, *U.S.S.R. Comput. Maths. Math.* **16**, 236–242. doi: 10.1016/0041-5553(76)90154-3.

Tan T.B. (1995) Estimating two and three dimensional pseudo-relative permeabilities with non-linear regression, Reservoir Simulation Symposium, San Antonio, Texas. doi: 10.2118/29129-MS.

Wang K., Killough J.E., Sepehrnoori K. (2009) A new upscaling method of relative permeability curves for reservoir simulation, *SPE Annual Technical Conference and Exhibition*, 4–7 October, New Orleans, LA. doi: 10.2118/124819-MS.

Yu X., Gen M. (2010) *Introduction to evolutionary algorithms*, Springer-Verlag, Berlin, Heidelberg, Germany.

Zarifi A.H., Sahraei E., Parvizi H. (2012) Pseudo relative permeability compensation for numerical dispersion, *Pet. Sci. Technol.* **30**, 15, 1529–1538. doi: 10.1080/10916466.2010.503217.

Zhang F., Reynolds A.C. (2002) Optimization algorithms for automatic history matching of production data, *Proceedings of 8th European Conference on the Mathematics of Oil Recovery*, Freiberg, Germany. doi: 10.3997/2214-4609.201405958.

Appendix A

The construction of Sobol's low-discrepancy sequences

The initial stage in generating a Sobol LDS set deals with operation on a set of *integers* in the interval $[1, 2^b-1]$, where b represents the number of bits in an unsigned integer on the operating computer (typically $b = 32$). Let x_{nk} be the nth draw of one of Sobol's integers in dimension k.

Generation of numbers in the Sobol's method, is based on a set of *direction integers*. A distinct direction integer is considered for each of the b bits in the binary integer representation. Let v_{kl} denote the lth direction integer for dimension k. In order to construct Sobol's numbers, one needs to evaluate the direction integers, first. This process involves the binary coefficients of a selected primitive modulo two for each dimension (Jäckel, 2002). Take p_k as the primitive polynomial modulo two for dimension k with the degree g_k (defined by equation (A-1)). We assume $a_{k0}\ldots a_{kg}$ representing the coefficients of p_k, with a_{k0} being the coefficient of the highest monomial term.

$$p_k(z) = \sum_{j=0}^{g_k} a_{kj} z^{gk-j} \tag{A-1}$$

In each dimension, the first g_k direction integers v_{kl} for $l = 1\ldots g_k$ are allowed to be freely chosen for the associated p_k of the dimension, provided that two conditions are met. First, the lth leftmost bit of the direction integer v_{kl} must be set. Second, only the l leftmost bits can be non-zero, where the leftmost bit refers to the most significant one in a bit field representation. All subsequent direction integers are calculated from a recurrence relation (A-2) (Jäckel, 2002):

$$v_{kl} = \frac{v_k(l-g_k)}{2^{g_k}} \oplus_2 \sum_{j=1}^{g_k} {}^{\oplus_2} a_{kj} v_{k(1-j)} \quad \text{for } l > g_k\,. \tag{A-2}$$

Hereby, $\oplus_2$ represents the binary addition of integers modulo two (often referred to in the computer science literature as the XOR gate), and $\sum^{\oplus_2}$ stands for a set of XOR operations. The procedure is to right-shift the direction integer $v_k\,(l-g_k)$ by g_k bits, and then performing the XOR operation with a selection of the un-shifted direction integers $v_k\,(l-j)$ for $j = 1,\ldots,g_k$. The summation is performed analogous to the conventional $\sum$ summation operator.

The only remaining requirement for the algorithm is the *generating integer* of the nth draw. For this sake, the natural choice appears to be the draw number itself, n. Nevertheless, any other sequence with a unique integer for each new draw is equally useful (Jäckel, 2002). Once all the preliminaries are set, the Sobol's integers, for the s dimensions of interest, are generated by (Jäckel, 2002);

$$x_{nk} = \sum_{j=1}^{s} {}^{\oplus_2} v_{kj}\,1. \tag{A-3}$$

In which the jth bit of the generating integer is set (counting from the right).

Jäckel (2002) has provided tabulated initialization numbers for generating Sobol's integers, up to a dimension of 32 (Table A-1). The generated sequence, using these initialization numbers, possesses the property; such that for any binary segment of the s-dimensional sequence of length 2^s there is exactly one draw in each of the 2^s hypercubes which result from subdividing the unit hypercube along each of its unit length extensions into half (Jäckel, 2002).

Once generated, conversion of Sobol's integers to other scales is fairly straightforward. For example, they can be converted to the [0, 1] scale by dividing the integers by 2^b.

Table A-1. An instance of the initialisation numbers for generating Sobol's LDS, up to a dimension of 32 (Jäckel, 2002).

k	g_k		v_{kl} for $l = 1,\ldots,10$									
1	0	1	1.2^{31}	1.2^{30}	1.2^{29}	1.2^{28}	1.2^{27}	1.2^{26}	1.2^{25}	1.2^{24}	1.2^{23}	1.2^{22}
2	1	11	1.2^{31}	3.2^{30}	5.2^{29}	15.2^{28}	17.2^{27}	51.2^{26}	85.2^{25}	255.2^{24}	257.2^{23}	771.2^{22}
3	2	111	1.2^{31}	1.2^{30}	7.2^{29}	11.2^{28}	13.2^{27}	61.2^{26}	67.2^{25}	79.2^{24}	465.2^{23}	721.2^{22}
4	3	1011	1.2^{31}	3.2^{30}	7.2^{29}	5.2^{28}	7.2^{27}	43.2^{26}	49.2^{25}	147.2^{24}	439.2^{23}	1013.2^{22}
5	3	1101	1.2^{31}	1.2^{30}	5.2^{29}	3.2^{28}	15.2^{27}	51.2^{26}	125.2^{25}	141.2^{24}	177.2^{23}	759.2^{22}
6	4	10011	1.2^{31}	3.2^{30}	1.2^{29}	1.2^{28}	9.2^{27}	59.2^{26}	25.2^{25}	89.2^{24}	321.2^{23}	835.2^{22}
7	4	11001	1.2^{31}	1.2^{30}	3.2^{29}	7.2^{28}	31.2^{27}	47.2^{26}	109.2^{25}	173.2^{24}	181.2^{23}	949.2^{22}
8	5	100101	1.2^{31}	3.2^{30}	3.2^{29}	9.2^{28}	9.2^{27}	57.2^{26}	43.2^{25}	43.2^{24}	225.2^{23}	113.2^{22}
9	5	101001	1.2^{31}	3.2^{30}	7.2^{29}	7.2^{28}	21.2^{27}	61.2^{26}	55.2^{25}	19.2^{24}	59.2^{23}	761.2^{22}
10	5	101111	1.2^{31}	1.2^{30}	5.2^{29}	11.2^{28}	27.2^{27}	53.2^{26}	69.2^{25}	25.2^{24}	103.2^{23}	615.2^{22}
11	5	110111	1.2^{31}	1.2^{30}	7.2^{29}	3.2^{28}	29.2^{27}	51.2^{26}	47.2^{25}	97.2^{24}	233.2^{23}	39.2^{22}
12	5	111011	1.2^{31}	3.2^{30}	7.2^{29}	13.2^{28}	3.2^{27}	35.2^{26}	89.2^{25}	9.2^{24}	235.2^{23}	929.2^{22}
13	5	111101	1.2^{31}	3.2^{30}	5.2^{29}	1.2^{28}	15.2^{27}	19.2^{26}	113.2^{25}	115.2^{24}	411.2^{23}	157.2^{22}
14	6	1000011	1.2^{31}	1.2^{30}	1.2^{29}	9.2^{28}	23.2^{27}	37.2^{26}	97.2^{25}	97.2^{24}	353.2^{23}	169.2^{22}
15	6	1011011	1.2^{31}	1.2^{30}	3.2^{29}	13.2^{28}	11.2^{27}	7.2^{26}	37.2^{25}	101.2^{24}	463.2^{23}	657.2^{22}
16	6	1100001	1.2^{31}	3.2^{30}	3.2^{29}	5.2^{28}	19.2^{27}	33.2^{26}	3.2^{25}	197.2^{24}	329.2^{23}	983.2^{22}
17	6	1100111	1.2^{31}	1.2^{30}	7.2^{29}	13.2^{28}	25.2^{27}	5.2^{26}	27.2^{25}	71.2^{24}	377.2^{23}	719.2^{22}
18	6	1101101	1.2^{31}	1.2^{30}	1.2^{29}	3.2^{28}	13.2^{27}	39.2^{26}	7.2^{25}	23.2^{24}	391.2^{23}	389.2^{22}
19	6	1110011	1.2^{31}	3.2^{30}	5.2^{29}	11.2^{28}	7.2^{27}	11.2^{26}	43.2^{25}	25.2^{24}	187.2^{23}	825.2^{22}
20	7	10000011	1.2^{31}	3.2^{30}	1.2^{29}	7.2^{28}	3.2^{27}	23.2^{26}	79.2^{25}	65.2^{24}	451.2^{23}	321.2^{22}
21	7	10001001	1.2^{31}	3.2^{30}	1.2^{29}	15.2^{28}	17.2^{27}	63.2^{26}	13.2^{25}	113.2^{24}	147.2^{23}	881.2^{22}
22	7	10001111	1.2^{31}	3.2^{30}	3.2^{29}	3.2^{28}	25.2^{27}	17.2^{26}	115.2^{25}	17.2^{24}	179.2^{23}	883.2^{22}
23	7	10010001	1.2^{31}	3.2^{30}	7.2^{29}	9.2^{28}	31.2^{27}	29.2^{26}	17.2^{25}	121.2^{24}	363.2^{23}	783.2^{22}
24	7	10011101	1.2^{31}	1.2^{30}	3.2^{29}	15.2^{28}	29.2^{27}	15.2^{26}	41.2^{25}	249.2^{24}	201.2^{23}	923.2^{22}
25	7	10100111	1.2^{31}	3.2^{30}	1.2^{29}	9.2^{28}	5.2^{27}	21.2^{26}	119.2^{25}	53.2^{24}	319.2^{23}	693.2^{22}
26	7	10101011	1.2^{31}	1.2^{30}	5.2^{29}	5.2^{28}	1.2^{27}	27.2^{26}	33.2^{25}	253.2^{24}	341.2^{23}	385.2^{22}
27	7	10111001	1.2^{31}	1.2^{30}	3.2^{29}	1.2^{28}	23.2^{27}	13.2^{26}	75.2^{25}	29.2^{24}	181.2^{23}	895.2^{22}
28	7	10111111	1.2^{31}	1.2^{30}	7.2^{29}	7.2^{28}	19.2^{27}	25.2^{26}	105.2^{25}	173.2^{24}	509.2^{23}	75.2^{22}
29	7	11000001	1.2^{31}	3.2^{30}	5.2^{29}	5.2^{28}	21.2^{27}	9.2^{26}	7.2^{25}	143.2^{24}	157.2^{23}	959.2^{22}
30	7	11001011	1.2^{31}	1.2^{30}	1.2^{29}	15.2^{28}	5.2^{27}	49.2^{26}	59.2^{25}	71.2^{24}	31.2^{23}	111.2^{22}
31	7	11010011	1.2^{31}	3.2^{30}	5.2^{29}	15.2^{28}	17.2^{27}	19.2^{26}	21.2^{25}	227.2^{24}	413.2^{23}	727.2^{22}
32	7	11010101	1.2^{31}	1.2^{30}	6.2^{29}	11.2^{28}	13.2^{27}	29.2^{26}	3.2^{25}	15.2^{24}	279.2^{23}	17.2^{22}

Appendix B

The algorithm to perform Quasi-Monte Carlo minimization

Assume ω_i^{iter} to represent the best solution for ith point at the $iter$th iteration, also consider $FBEST$ as the best (minimum) value of f, recorded up to the $iter$th iteration. A detailed description of the QMC procedure is then ensued as follows (Hickernell and Yuan, 1997):

Step-0 Initialize

Input the number of initial points, N, the number of points with best (lowest) objective function values to retain in each iteration, N_{best}, and the desired number of iterations to be done for local search on each of the points, $Niter_{local.search}$.

Set the number of iterations, $iter = 0$

Set $NSP = 0$; $NSWP = 0$; $NTIX(j) = 0$ for $(i \leq j \leq N)$

Step-1 Concentrate

Obtain a new point set, by applying $Niter_{local.search}$ iteration(s) of an inexpensive local search to each of ω_i^{iter} points $(1 \leq i \leq N)$

Step-2 Reduce

Find $\Xi(iter) \subset \{1,\ldots,N\}$ such that $\Xi(iter)$ has N_{best} elements and that $f(\omega_i^{iter}) \leq f(\omega_j^{iter})$ $\forall i \in \Xi(iter)$ and $\forall j \notin \Xi(iter)$

If $j \in \Xi(iter)$, set $NTIX\ (j) = NTIX\ (j) + 1$

If $j \notin \Xi(iter)$, set $NTIX\ (j) = 0$

Step-3 Find local minimum

For $j = 1,\ldots, N$ such that $NTIX\ (j) \geq 2$

Set $NTIX\ (j) = 0$
If $NSP = 0$ or $f(\omega_j^{iter}) \leq FBEST + 10^{-4}$ then

Starting from ω_j^{iter}, perform a local optimization search, to obtain the local minimize of the point, $\omega_{j,local.\min}^{iter}$.

If $f\left(\omega_{j,local.\min}^{iter}\right) < FBEST$ then

Set $NSP = NSP + 1$; $NSWP = 0$; $FBEST = f\left(\omega_{j,local.\min}^{iter}\right)$

Else
Set $NSWP = NSWP + 1$

End
Else
If $\frac{NSWP}{NSP} \geq 3$ then stop (success)

Step-4 Sample additional points
For $j = 1, 2, \ldots, N$
If $NTIX\ (j) = 0$ then
Generate ω_j^{iter+1} by the Sobol's LDS technique

Else
Set $\omega_j^{iter+1} = \omega_j^{iter}$

End
Set $iter = iter + 1$
If the total number of function calls reached then stop (failure).
Go to *Step-1*.

Appendix C

The algorithm to perform the Artificial Bee Colony (ABC) minimization

Assume ω_i^{iter} to represent the best solution for ith point at the $iter$th iteration, also consider $FBEST$ as the best (minimum) value of the function, f, recorded up to the $iter$th iteration. The ABC optimization algorithm then works in the following steps:

Step-0 Initialize
Input the number of initial points, N, the required relative tolerance of solution as a stopping criteria, e, and the maximum number of cycles to be performed, $Niter_{\max}$.

Step-1 Find initial minimum
Set $iter = 1$
Generate N solution points, $\omega_{i\ 1\leq i\leq N}^{iter}$, by (uniform) random selection within the s-dimensional domain.
Set $FBEST = \min\ (f\ (\omega_i^{iter}))$ for $1 \leq i \leq N$.

Step-2 Repeat
Employed bees
Perturb each solution point, based on its latest solution history (Eq. (10)), obtain $\omega_{perturb,i;1\leq i\leq N}^{iter}$ solutions set.

For $i = 1, \ldots, N$
If $f\left(\omega_{perturb,i}^{iter}\right) < f(\omega_i^{iter})$ then

$\omega_i^{iter} = \omega_{perturb,i}^{iter}$

End
Abandon the ith employed bee, if its solution cannot be improved, and convert it to a scout-type bee.

Onlooker bees
Compute the probability of each solution point, based on its corresponding solution history (eq.), obtain $prob_{i_{1\leq i\leq N}}$.

For $i = 1,\ldots, N$
If $rand[0, 1] \leq prob_i$ then
Assign the ith onlooker bee to the ith solution set.
Else
Assign the ith onlooker bee to a different solution set, if the probabilistic measure is met.
End
Perturb each solution point of an onlooker bee, based on its latest history (Eq. (10)), obtain $\omega'^{iter}_{perturb,i_{1\leq i\leq N}}$ solutions set.

For $i = 1, \ldots, N$
If $f\left(\omega'^{iter}_{perturb,i}\right) < f(\omega'^{iter}_i)$ then
$\omega'^{iter}_i = \omega'^{iter}_i$
End

Scout bees
Randomly select $N_{scout.bees}$ solution points within the s-dimensional domain, obtain $\omega''^{iter}_{i\ 1\leq i\leq N}$.

If $\min(\omega''^{iter}_i)_{1\leq iN} < FBEST$ then

$FBEST = \min\ (\omega''^{iter}_i)_{1\leq i\leq N}$

End
Set $FBEST = \min\ (\omega, \omega', \omega'')$
If $(iter > Niter_{\max})$ then stop.
If $\left(\left|FBEST^{iter} - FBEST^{iter-1}\right| < \varepsilon\right)$ then stop.
Set $iter = iter + 1$
Go to *Step-2*.

Appendix D

The algorithm to perform the Particle Swarm Optimization (PSO) minimization

Assume ω_i^{iter} to represent the solution for ith point at the $iter$th iteration. Take ω_{best_i} as the *best* solution for ith point recorded up to the $iter$th iteration, and $\omega_{velocity_{i,j}}$ as the velocity of the ith point in the jth dimension, with the length Δ_j. Also, consider $FBEST$ as the best (minimum) value of the function, f (recorded up to the $iter$th iteration), and U as the random uniform distribution. The PSO optimization algorithm is mentioned below.

Step-0 Initialize
Input the number of initial points, N, the required relative tolerance of solution as a stopping criteria, the

maximum number of cycles to be performed, $Niter_{\max}$, and the three coefficients in the velocity equation (Eq. (1)), $coef_1$, $coef_2$, $coef_3$.

Step-1 Find initial minimum

Set $iter = 1$

Generate N solution points, $\omega_{i\ 1\le i\le N}^{iter}$, by (uniform) random selection within the s-dimensional domain.

Set $FBEST = \min\left(f\left(\omega_i^{iter}\right)\right.$ for $i = 1, \ldots, N$.

For $i = 1, \ldots, N$

$\omega_{best_i} = \omega_i^{iter}$

For $j = 1, \ldots, s$

$\omega_{velocity_{i,j}=U(-\Delta_j,+\Delta_j)}$

Step-2 Repeat

For $i = 1, \ldots, N$

For $j = 1,\ldots, s$

$rand_1 = U(0,1)$; $rand_2 = U(0,1)$

$$\omega_{velocity_{i,j}}^{iter} = coef_1 \times \omega_{velocity_{i,j}}^{iter} + coef_2 \times rand_1 \times (\omega_{best_{i,j}} = \omega_{i,j}^{iter}) + coef_3 \times rand_2 \times (\omega_{FBEST_{i,j}} = \omega_{i,j}^{iter})$$

$$\omega_i^{iter} = \omega_i^{iter} + \omega_{velocity_i}^{iter}$$

If $f(\omega_i^{iter}) < f(\omega_{best_i})$ then

$\omega_{best_i} = \omega_i^{iter}$

If $f(\omega_{best_i}) < FBEST$ then

$FBEST = f(\omega_{best_i})$

$\omega_{FBEST} = \omega_i^{iter}$

If $(iter > Niter_{\max})$ then stop.

If $\left(\left|FBEST^{iter} - FBEST^{iter-1}\right| < \varepsilon\right)$ then stop.

Set $iter = iter + 1$

$\omega^{iter} = \omega^{iter-1}$

$\omega_{velocity}^{iter} = \omega_{velocity}^{iter-1}$

Go to *Step-2*.