AUTOMATIC HISTORY MATCHING USING DIFFERENTIAL EVOLUTION ALGORITHM

Jianxin Wang and Jill S. Buckley
Petroleum Recovery Research Center, New Mexico Tech
Socorro, NM 87801, USA

This paper was prepared for presentation at the International Symposium of the Society of Core Analysts held in Trondheim, Norway 12-16 September, 2006

ABSTRACT

A Windows-based 1-D automatic history matching simulator utilizing the differential evolution algorithm–a global optimization algorithm–has been developed. It can be applied to calculate oil and water relative permeabilities and dynamic capillary pressure simultaneously from unsteady-state immiscible coreflooding data. The simulator has been tested for many cases with very satisfactory results. Detailed descriptions of the algorithms used in the simulator and some of the application results are presented in this paper.

INTRODUCTION

The unsteady-state coreflooding technique is widely used to measure water/oil relative permeabilities, because it is faster and it is analogous to waterflooding in a reservoir. Unlike steady-state measurements, however, interpretation of results from unsteady-state coreflooding to retrieve relative permeability is complicated. Direct interpretation with JBN method (Johnson *et al*, 1959) utilizes only the data acquired after breakthrough of the displacing phase, limiting the available data to only a few points, especially for strongly water wet cases. More often, a "history matching" simulator is used to interpret unsteady-state coreflooding tests. History matching uses relative permeabilities and capillary pressure as the adjustable input parameters to simulate oil production, saturation distribution, and the pressure gradient along the core sample, based on Darcy's law. The simulated results are compared to the experiment measurements and parameters are adjusted until a good match is reached. For a complex system such as oil/water two-phase flow in a porous medium, manually adjusting these parameters is extremely tedious, inefficient, and relies heavily on the expertise of simulation engineers. A better option is to use an automatic history matching algorithm in which the computer searches for the optimal solution. Mathematically, this can be achieved by minimizing an objective function, which takes the difference between simulation output and the experimental data.

Traditional automatic history matching algorithms that have been employed in the petroleum industry for decades are mainly gradient-based, such as Newton, Gauss-Newton, and quasi-Newton algorithms. There are two major drawbacks involved in these

algorithms. First, the calculations of gradients of the objective function with respect to the multiple adjustable parameters are very costly since these gradients can only be approached by numerical differentiations which usually require many simulation runs thus consuming lots of computer resources. Second, for a highly non-linear, complex model involving multiple parameters, gradient-based algorithms are inherently prone to finding local minima instead of a global minimum. To avoid these problems, heuristic global optimization algorithms have been developed for flow simulation, such as simulated annealing (Ouenes et al., 1992; Ucan et al., 1993) and genetic algorithms (Sun and Kishore, 2003; Tokuda et al., 2004). Although these algorithms largely solved the local minima problem, the improvement in convergence speed is less satisfactory.

A recently developed evolutionary global optimization algorithm – the differential evolution (DE) algorithm – has been shown to significantly outperform simulated annealing and other evolutionary algorithms in many applications (Storn and Price, 1995). The principle of the algorithm is straightforward and easy to implement, yet it is very powerful with respect to the reliability, robustness, and convergence speed. Moreover, it uses real numbers for parameter searching, making the incorporation of parameter constraints to the algorithm very easy. In this paper we report the first application of a DE algorithm to improve automatic history matching of coreflood data.

DIFFERENTIAL EVOLUTION ALGORITHM

The DE algorithm was first proposed by Price and Storn in 1995 in their attempt to solve the Chebychev polynomial fitting problem. Since then, the algorithm has attracted interest from many researchers and has been widely applied to solve a variety of problems, including function minimization, non-linear programming, and complex simulations (Storn and Price, 1995; Storn, 1995; Storn and Price, 1996; Storn, 1996a; Storn, 1996b; Lampinen and Zelinka, 1999; Lampinen, 2001; Karaboga and Okdem, 2003). As a member in the evolutionary algorithm family, DE is also a population-based, stochastic global optimizer. The key innovation of DE, which distinguishes it from other evolutionary algorithms, lies in its unique mutation scheme which uses vector differences to perturb the vector population to generate new generations.

Suppose we want to minimize an objective function f(X), where $X=(x_1, x_2, \ldots, x_D)$ is a vector with D elements (in our case the parameters for relative permeability and capillary pressure). Instead of searching from a single point in the parameter space, DE initializes a population of NP points (vectors), $X_i^{(0)}$, i=1,2,…,NP, which are randomly distributed in the D-dimension parameter space. Each individual vector represents a potential solution to the problem. The goal of evolutionary algorithms is to find a heuristic method which would produce new generations from the initial generation with the expectation that the offspring will be better solutions than their ancestors, i.e., solutions more closely approaching the global minimum. In DE, the population number NP is fixed from generation to generation. The evolutionary schemes employed in the DE algorithm, including mutation, crossover, and selection, will be described in the following sections.

Mutation

For each vector $X_i^{(G)}$ in generation G, i=1,2,..NP, a perturbed vector $V_i^{(G+1)}$ is constructed according to one of the strategies listed in Table-1:

Table-1: different mutation strategies used in DE algorithm

Strategy ID	Perturbed vector
1	$V_i^{(G+1)} = X_{r_3}^{(G)} + F(X_{r_1}^{(G)} - X_{r_2}^{(G)})$
2	$V_i^{(G+1)} = X_{best}^{(G)} + F(X_{r_1}^{(G)} - X_{r_2}^{(G)})$
3	$V_i^{(G+1)} = X_i^{(G)} + F(X_{best}^{(G)} - X_i^{(G)} + X_{r_1}^{(G)} - X_{r_2}^{(G)})$
4	$V_i^{(G+1)} = X_{best}^{(G)} + F(X_{r_1}^{(G)} - X_{r_2}^{(G)} + X_{r_3}^{(G)} - X_{r_4}^{(G)})$
5	$V_i^{(G+1)} = X_{r_5}^{(G)} + F(X_{r_1}^{(G)} - X_{r_2}^{(G)} + X_{r_3}^{(G)} - X_{r_4}^{(G)})$
6	$V_i^{(G+1)} = X_i^{(G)} + \lambda(X_{best}^{(G)} - X_i^{(G)}) + F(X_{r_1}^{(G)} - X_{r_2}^{(G)})$

where $X_{r1}^{(G)}$, $X_{r2}^{(G)}$, $X_{r3}^{(G)}$, $X_{r4}^{(G)}$, $X_{r5}^{(G)}$ are randomly selected, mutually different vectors from the population of generation G, with $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$, and $X_{best}^{(G)}$ is the best solution among all the individuals in population G. F is a mutation scaling factor which controls the magnitude of mutation and is kept constant throughout the evolution process. λ is a randomly generated factor used in strategy 6, with $\lambda \in [0,1]$.

Crossover

The $X_i^{(G)}$ and $V_i^{(G+1)}$ from mutation step are combined to reproduce a trial vector according to the following scheme (Storn, 1996b):

$$u_{i,j}^{(G+1)} = \begin{cases} v_{i,j}^{(G+1)} & \text{for } j = < n >_D, < n+1 >_D, ..., < n+L-1 >_D \\ x_{i,j}^{(G)} & \text{for all other } j \in [1, D] \end{cases}$$

where $v_{i,j}^{(G+1)}$, j=1,2,…,D, is the j-th element in vector $V_i^{(G+1)}$, $x_{i,j}^{(G)}$, j=1,2,…,D, is the j-th element in $X_i^{(G)}$, and $u_{i,j}^{(G+1)}$, j=1,2,…,D, is the j-th element in the trial vector $U_i^{(G+1)}$. Symbol $< >_D$ represents a modulo function with modulus D. n is a randomly chosen integer from [1,D]. L is the total number of elements (parameters) in vector $X_i^{(G)}$ which are to exchange with the corresponding segment from vector $V_i^{(G+1)}$ (crossover). L is drawn from interval [1,D] according to the following pseudo code:

```
L=0;
do {
        L=L+1;
} while (rand() < Cr) AND (L<D);
```

Thus, the probability for crossover to occur with L as the exchange length is controlled by a crossover factor Cr, as $Pr(L \geq \nu)=Cr^{\nu-1}$, for $\nu>0$. Like the mutation scaling factor F, Cr is also maintained constant during the evolutionary process.

Selection

Unlike other evolutionary algorithms, the selection of individuals to constitute the next generation in DE is relatively simple. The fitness between the trial vector $U_i^{(G+1)}$ and its counterpart $X_i^{(G)}$ are compared. The better one, i.e. the one with small objective function value, will be selected. It enters into the next generation, while the other one is discarded, i.e.,

$$X_i^{(G+1)} = \begin{cases} U_i^{(G+1)} & \text{if } f(U_i^{(G+1)}) < f(X_i^{(G)}) \\ X_i^{(G)} & \text{otherwise} \end{cases}$$

where $X_i^{(G+1)}$, i=1,2,…,NP, is the new vector in the generation G+1. As a result, all the individuals in the next generation (G+1) are at least as good as the current generation (G), or better.

Stop Criteria

The evolution of new generations continues, using the mutation, crossover, and selection schemes described above, until a stop criterion is reached. This stop criterion can be either 1) when a desired minimum objective function value is reached, or 2) when a maximum number of iterations (generations) is reached, or 3) when the maximum allowed computer CPU time is reached. To set a lower bound for the objective function value is probably not appropriate since the range of the objective function value may not be easily determined beforehand. In the case where CPU time is not a major concern, it is practical to set a maximum iteration number as the stop criterion. Note that the objective function value for the best solution in each new generation is either decreasing or unchanged, it is also worthwhile to check this value after several iterations and force the iteration to stop once this value is no longer decreasing or decreasing very slowly.

DEVELOPMENT OF DE AUTOMATIC HISTORY MATCHING SIMULATOR

Flow Functions

Relative permeability models

For a water/oil two-phase flow in porous media, the relative permeabilities can be expressed as a function of saturation with the well-known Corey-functions:

$$k_{rw} = k_{rw,max}\left(\frac{S_w - S_{wi}}{1 - S_{or} - S_{wi}}\right)^{n_w}$$

$$k_{ro} = k_{ro,max}\left(\frac{1-S_w-S_{or}}{1-S_{or}-S_{wi}}\right)^{n_o}$$

where S_w is water saturation, S_{wi} is initial water saturation, S_{or} is residual oil saturation, $k_{rw,max}$ is the maximum water relative permeability at $S_w=1-S_{or}$, and $k_{ro,max}$ is the maximum oil relative permeability at $S_w=S_{wi}$. If the base permeability used to calculate relative permeability is the oil permeability at $S_w=S_{wi}$, we have $k_{ro,max}=1$. All the parameters in the right-hand sides of the above equations can be measured from experiments, except for the exponents n_w and n_o. Therefore, the shapes of curves k_{rw}-S_w and k_{ro}-S_w are determined by two unknowns, n_w and n_o.

Although Corey-functions are convenient, in many cases they cannot exactly represent the relative permeabilities. A more flexible and accurate way to represent relative permeabilities is to use cubic B-spline functions. Both Corey-functions and B-spline functions will be implemented in the design of the automatic history matching simulator.

Capillary pressure model

Static capillary pressure can be measured experimentally. Static capillary pressure, however, may not represent the in-situ dynamic capillary pressure due to the influence of flow rate, possible variations of the wetting properties, and the effects of heterogeneity during coreflooding process (Bentsen and Manai, 1991; Bentsen, 1998). It is therefore desirable to estimate both relative permeabilities and capillary pressures simultaneously. Several different models have been proposed for capillary pressure (e.g., Sun and Kishore, 2003; Tokuda et al, 2004), but they used different types of functions to represent different wettability conditions, making them difficult to implement. A novel capillary pressure model is proposed in this study, which is very flexible and uses only one function to cover any wetting conditions:

$$p_c = \begin{cases} (p_{c\,max}-p_{co})\left(\dfrac{ctg(S_w)-ctg(S_{wpco})}{ctg(S_{wi})-ctg(S_{wpco})}\right)^{m_+}+p_{co} & \text{if } S_w < S_{wpco} \\ (p_{c\,min}-p_{co})\left(\dfrac{ctg(S_{wpco})-ctg(S_w)}{ctg(S_{wpco})-ctg(1-S_{or})}\right)^{m_-}+p_{co} & \text{if } S_w \geq S_{wpco} \end{cases}$$

where p_c is capillary pressure, p_{cmax} and p_{cmin} are maximum and minimum capillary pressure, respectively, within the saturation range between S_{wi} and $1-S_{or}$, p_{co} is the capillary pressure at the point of transition from imbibition curve to drainage curves, S_{wpco} is the corresponding water saturation at p_{co}, m_+ and m_- are exponents for drainage curve and imbibition curve, respectively. All six variables (p_{cmax}, p_{cmin}, p_{co}, S_{wpco}, m_+, and m_-) are adjustable parameters.

Objective Function

The first step in applying DE for automatic parameter searching is to construct an appropriate objective function. In the current situation, we want to match both oil recovery curve and pressure drop curve from coreflooding tests. Accordingly, the objective function is defined as:

$$OBJ_i = \frac{1}{2n}\sum_{l=1}^{n}\left[\frac{\left|R_{meas,l} - R_{sim,l}\right|}{R_{max}} + \frac{\left|\Delta p_{meas,l} - \Delta p_{sim,l}\right|}{\Delta p_{max}}\right]$$

where OBJ_i is the objective function value for a given parameter vector $X_i^{(G)}$, n is the number of sampling points along the oil recovery or pressure drop curve, $R_{meas,l}$ is the measured oil recovery value at point-l on the curve, $R_{sim,l}$ is the corresponding oil recovery from simulation, $\Delta p_{meas,l}$ is the measured pressure drop at point-l, and $\Delta p_{sim,l}$ is the simulated pressure drop. Since R and Δp may differ significantly in numerical values, the differences between simulation and experiment for both oil recovery and pressure drop are normalized by their maximum experimental values, R_{max} and Δp_{meas}, respectively, to give them equal weight.

Simulator Development

The automatic history matching simulator includes three major modules: a control module, a DE module, and a flow simulation module. The control module is a Windows-based application, which provides the interfaces for data input/output, graphical display, and communications with other two modules. The DE module uses the DE algorithm to automatically search for the best parameters (relative permeability and capillary pressure) that minimize the objective function. This objective function is evaluated by a 1-D flow simulation module, which takes relative permeabilities and capillary pressure provided by the DE module as input and simulates the oil recovery and pressure drop history as the output using a standard flow simulator described elsewhere (Wang, 1998). Both the DE and flow simulation modules were written in Fortran and compiled as dynamic link libraries (DLLs).

The parameters required from DE include those in both relative permeability (two adjustable parameters for Corey or 10 for B-spline) and capillary pressure (six adjustable parameters) models. For the B-spline model, five discrete points for both k_{rw} and k_{ro} are evenly distributed within the saturation range from $S_w=S_{wi}$ to $S_w=1-S_{or}$. Thus, the parameter vector used in the DE search is constructed as shown in Fig. 1.

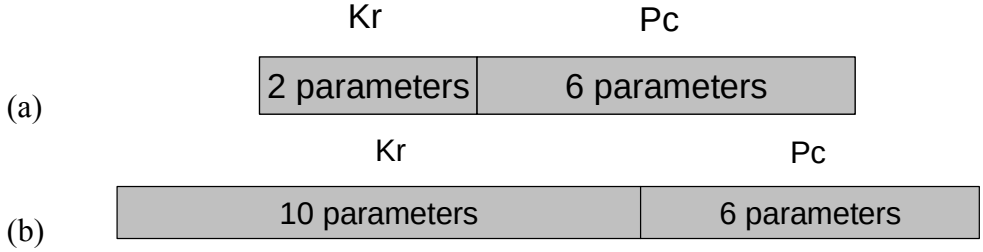


Figure-1: Parameter vectors used in DE automatic search, when relative permeability k_r is expressed as (a) Corey-function, or (b) cubic B-spline function.

Note that all the parameters are used as real numbers, without the binary encoding usually required in genetic algorithms. All the parameters are confined within their physical ranges. In order to reduce the search time, the initializations of both k_{rw} and k_{ro} are confined to values in the space below the lines connecting their end points.

The most costly step during each DE search is the call to the simulation module. The evolution iteration itself, including the mutation, crossover, and selection operations, only takes a very small fraction (<5%) of total CPU time. Since there are NP trial vectors generated in each generation, and each trial vector needs a call to the simulation module to evaluate the objective function, the total number of simulation runs that will be executed is

Number of simulation runs = (Number of total iterations + 1) * NP

CASE STUDY

Many cases have been conducted to test the performance of this DE-based automatic history matching simulator. In general, the simulator can be conveniently applied to simulate and match laboratory-measured waterflood production history to a very satisfactory level, especially if the B-spline function is used to represent relative permeability. Convergence speed is acceptable, ranging from one to several minutes on a PC with a Pentium-4 processor (3.0 GHz, 1.0 GB RAM), depending on the matching degree required. Results from two selected case studies are included here.

Coreflooding measurements

Two Berea sandstone core samples were cut from the same block. After establishing the initial water saturation (S_{wi}) by a crude oil, the cores were aged at 80°C for two weeks. Waterflooding was conducted at room temperature (22°C) with constant injection rate of 80 cm^3/hr. Both pressure drop across the core plug and the oil recovery were recorded as a function of injection brine pore volume (PV). Properties of oil, brine, and core samples are listed in Tables 2-3.

Table-2 Properties of Berea sandstone core samples

Sample	Length (cm)	Diameter (cm)	Porosity (%)	K_{N2}* (md)	S_{wi} (%)	$K_{o@Swi}$** (md)
B35	7.21	3.79	22	803	21	501
B15	5.73	3.59	22	786	20	483

* K_{N2}: nitrogen permeability;
** $K_{o@Swi}$: oil permeability at initial water saturation.

Table-3 Properties of oil and brine at 22°C

	Density (g/cm^3)	Viscosity (mPa.s)
Crude oil	0.83	5.55
Brine	1.15	1.72

Matching results

Figure-2 shows the oil recovery curve from one case study. It is matched quite well by the Corey function from automatic history matching, and almost completely matched by the B-spline function under similar conditions. The pressure drop is matched very well with either Corey or B-spline functions (**Figure-3**). Another case study is shown in **Figure-4** (oil recovery) and **Figure-5** (pressure drop). In this case, the B-spline results are significantly better than those using the Corey model.

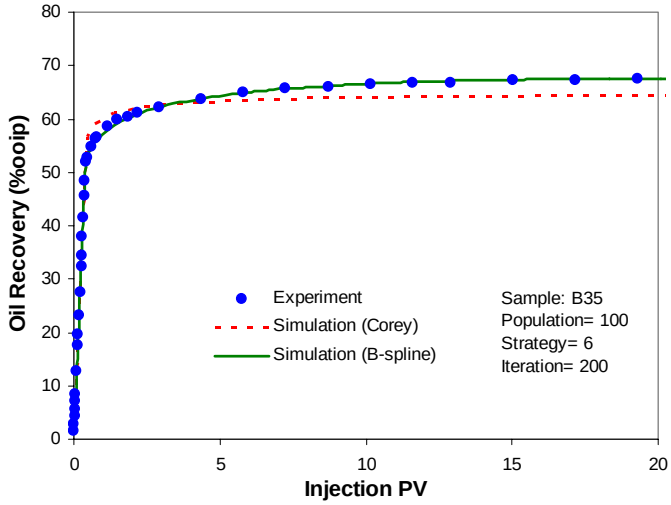


Figure-2: Oil recovery from automatic history matching using different k_r models (case-1)

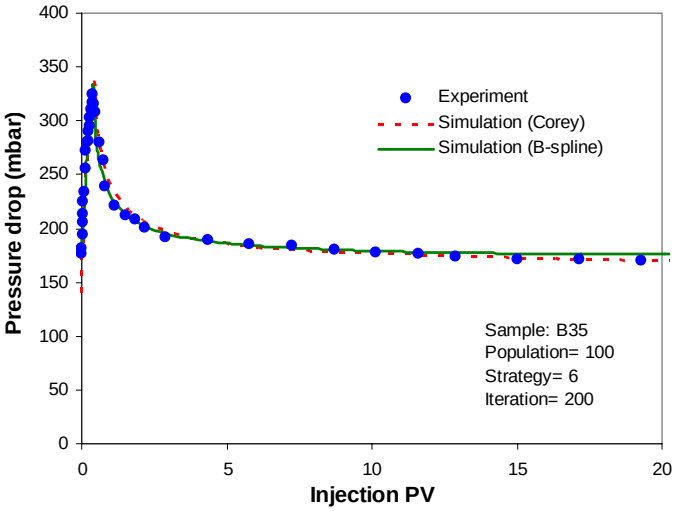


Figure-3: Pressure drop from automatic history matching using different k_r models (case-1)

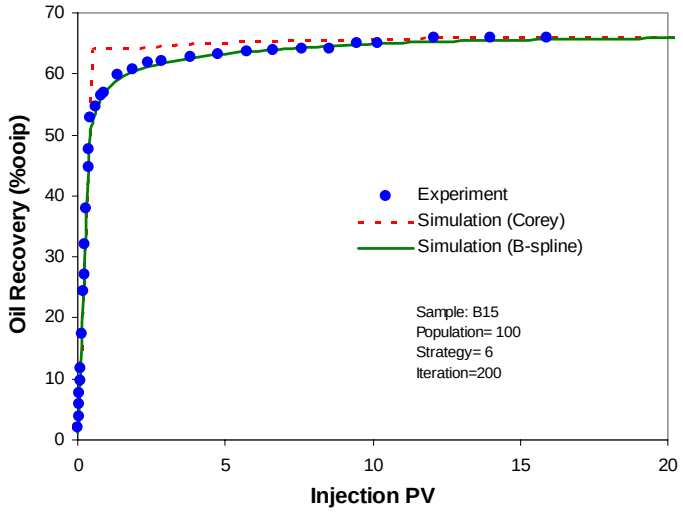


Figure-4: Oil recovery from automatic history matching using different k_r models (case-2)

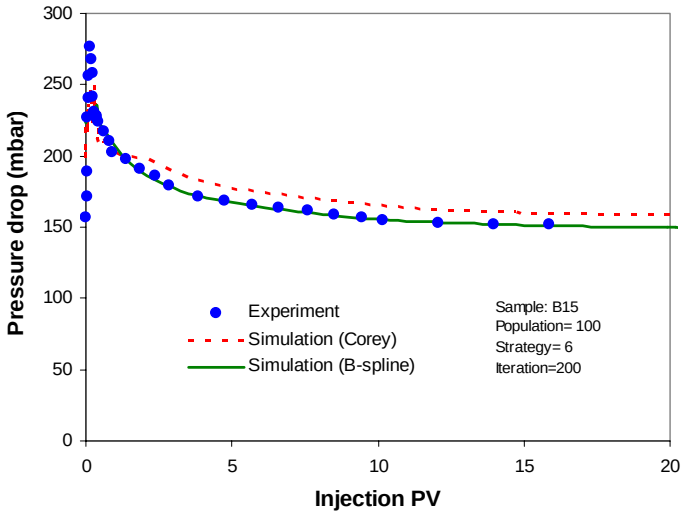
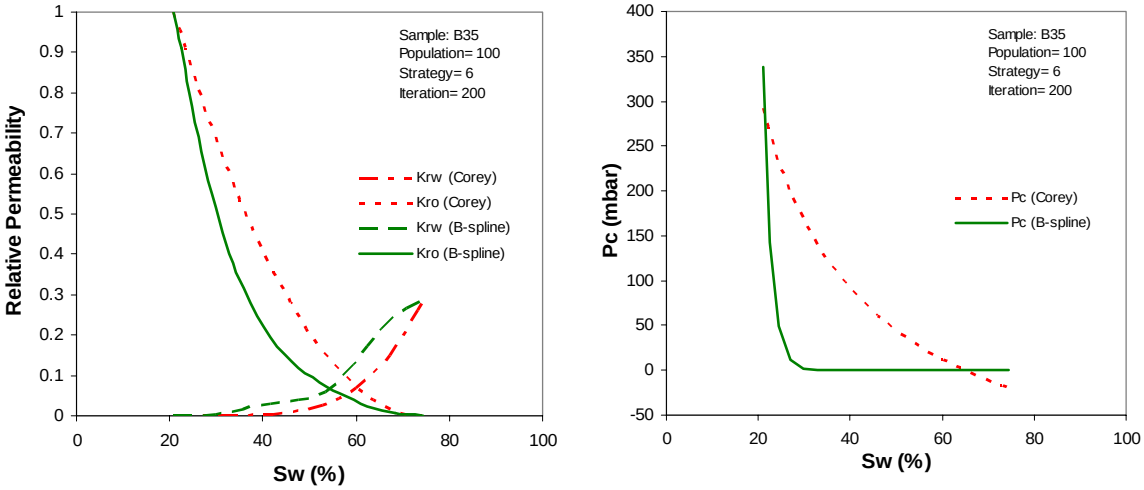


Figure-5: Pressure drop from automatic history matching using different k_r models (case-2)

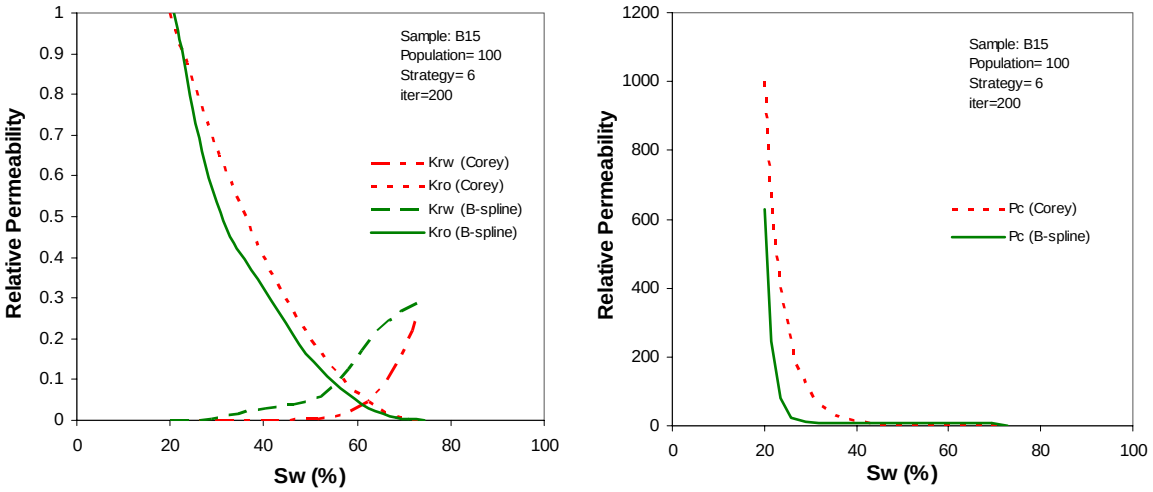
Calculated k_r and p_c

Figure-6 shows the corresponding relative permeabilities and capillary pressure from automatic history matching in case-1. Although only a slight difference in oil recovery was obtained from the two different matches (Figure-2), oil relative permeability (k_{ro}), water relative permeability (k_{rw}), and capillary pressure obtained from the Corey-function match are very different from those obtained using a B-spline model. Similar trends are observed in the case-2 study (**Figure-7**). It is evident that small deviations from the experimental data can have a big influence on the flow functions calculated in the history matching process.



(a) kr (b) Pc

Figure-6: k_r and p_c obtained from automatic history matching (case-1).



(a) kr (b) Pc

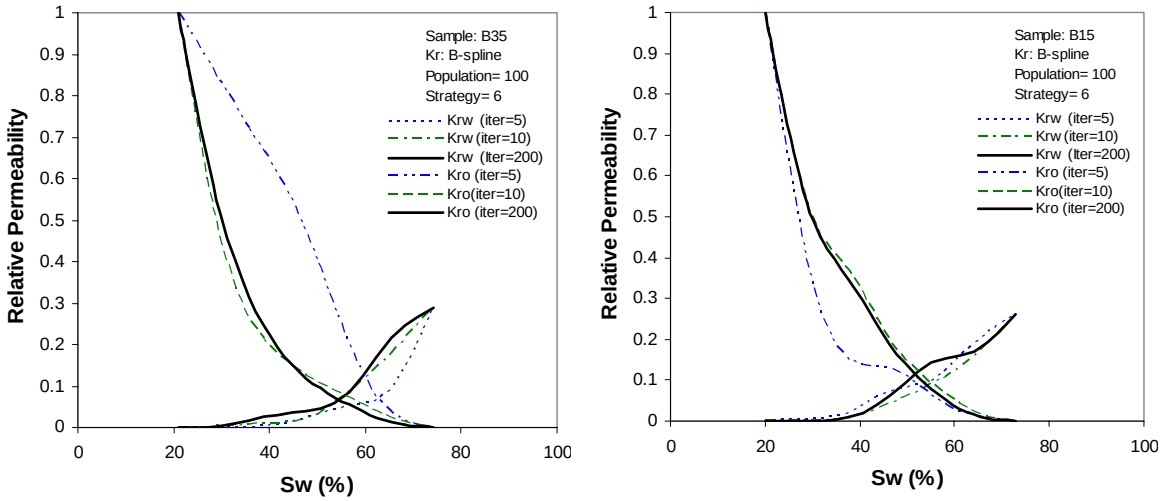
Figure-7: k_r and p_c obtained from automatic history matching (case-2).

Convergence Speed

The convergence speed is mainly determined by the size of the population (NP) in the DE algorithm, and the functional form chosen for the k_r model. It is also influenced by the mutation scaling factor, F, and the crossover factor, Cr. The suggested values for NP, F, and Cr are based on rule of thumb (Storn, 1996b). NP should be at least 10 times as big as the vector length (total number of parameters). F is usually chosen from [0.5,1]. $Cr \in [0,1]$ should be considerably lower than 1. Based on preliminary testing, default values of NP=100, F=0.8, and Cr=0.5 were selected, although they can be adjusted.

Examination of the OBJ shows that it decreases very rapidly for the first several iterations and reaches an equilibrium after about a dozen iterations. The influence of population size (NP=100, 160, and 200, were tested) is very small.

In fact, the calculated relative permeabilities after 10 iterations (~ 1.5 minutes) are already very close to those obtained after 200 iterations (**Figure-8**). This was found to be true for most cases. Therefore, very satisfactory results can be achieved generally within a few minutes, though even better results are available if computing time is not restricted.



(a) kr for B35 (b) kr for B15

Figure-8: Effect of number of iterations on k_r obtained from automatic history matching.

CONCLUSIONS

1. A differential evolution algorithm to automate a 1-D history matching simulator has been developed for interpretation of relative permeability and dynamic capillary pressure from unsteady-state immiscible coreflooding experimental data. The simulator is robust, fast, and easy to use, with only a few control parameters that can be adjusted by the user.

2. Simulated production history can satisfactorily match the experimental measurements. The convergence speed is fast, generally within a few minutes with a PC.

REFERENCES

Bentsen, R.G., and Manai, A.A.: "Measurement of cocurrent and countercurrent relative permeability curves using the steady-state method," *AOSTRA J. Res.*, (1991) **7**, 169-181.

Bentsen, R.G.: "Influence of Hydrodynamic Forces and Interfacial Momentum transfer on the Flow of Two Immiscible Phases," *J. Petr. Sci. Eng.*, (1998) **19**, 177-190.

Johnson, E.F., Bossler, D.P., and Naumann, V.O.: "Calculation of Relative Permeability from Displacement Experiments," SPE 1023-G, Petr. Trans., AIME, Vol **216**, 1959, 370-372.

Karaboga, D., and Okdem, S.: "A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm," Intl. XII Turkish Symp. On Artificial Intelligence and Neural Networks–TAINN 2003.

Lampinen, J.: "Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution," In: Radek Matousek and Pavel Osmera (eds.) 2001. Proceedings of *MENDEL 2001, 7th Intl. Conference on Soft Computing*, June 6-8 2001, Brno, Czech Republic, pp.50-57. ISBN 80-214-1894-X.

Lampinen, J., and Zelinka, I.: "Mixed Integer-Discrete-Continuous Optimization by Differential Evolution Part 1: the Optimization Method," In: Osmera, Pavel (ed.) 1999. Proceedings of MENDEL'99, 5th Intl. Mendel Conference on Soft Computing, June 9-12 1999, Brno, Czech Republic, pp. 71-76. ISBN 80-214-1131-7.

Ouenes, A., Fasanino, G., Lee, R.L.: "Simulated Annealing for Interpreting Gas/Water Laboratory Corefloods," SPE ATCE, 4-7 October, 1992, Washington, D.C.

Storn, R. and Price, K.: "Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," Technical Report TR-95-012, ICSI, March 1995.

Storn, R.: "Differential Evolution Design of an IIR-Filter with Requirements for Magnitude and Group Delay," IEEE International Conference on Evolutionary Computation ICEC 96, pp. 268 - 273, Technical Report TR-95-026, ICSI, May 1995.

Storn, R. and Price, K.: "Minimizing the real functions of the ICEC'96 contest by Differential Evolution," IEEE Conference on Evolutionary Computation, Nagoya, 1996, pp. 842–844.

Storn, R.: "System Design by Constraint Adaptation and Differential Evolution," Technical Report TR-96-039, ICSI, November 1996a.

Storn, R.: "On the Usage of Differential Evolution for Function Optimization," NAFIPS 1996b, Berkeley, pp. 519 - 523.

Sun, X., and Kishore, K.M.: "Estimation of Flow Functions During Drainage Using Genetic Algorithm," paper SPE 84548, presented at the SPE Annual Conference and Exhibition held in Denver, Colorado, USA, 5-8 Oct. 2003.

Tokuda, N., Takahashi, S., Watanada, M.: "Application of Genetic Algorithm to History Matching for Core Flooding," paper SPE 88621, presented at the SPE Asia Pacific Oil and Gas Conference and Exhibition held in Perth, Australia, 18-20 Oct. 2004.

Ucan, S., Civan, F., and Evans, R.D.: "Simulated Annealing for Relative Permeability and Capillary Pressure From Unsteady-State Non-Darcy Displacement," paper SPE 26670, SPE ATCE, 3-6 October, 1993, Houston, Texas.

Wang, J.X.: "Effects of Wettability on Rate-Dependence of Relative Permeability End-Points," MS thesis, New Mexico Institute of Mining and Technology, Socorro, New Mexico, USA, April, 1998.