Comparing Genetic Algorithms and Newton-Like Methods for the Solution of the History Matching Problem

Elisa Portes dos Santos[1,*],Carolina Ribeiro Xavier[1,*],
Paulo Goldfeld[2], Flavio Dickstein[2],
and Rodrigo Weber dos Santos[1]

[1] Dept. of Computer Science, Federal University of Juiz de Fora
Juiz de Fora, MG, Brazil
[2] Dept. of Applied Mathematics, Federal University of Rio de Janeiro
Rio de Janeiro, RJ, Brazil
* First Authors in Alphabetical Order
rodrigo.weber@ufjf.edu.br

Abstract. In this work we presents a comparison of different optimization methods for the automatic history matching problem of reservoir simulation. The history matching process is an inverse problem that searches a set of parameters that minimizes the difference between the model performance and the historical performance of the field. This model validation process is essential and gives credibility to the predictions of the reservoir model. Derivative-based methods are compared to a free-derivative algorithm. In particular, we compare the Quasi-Newton method, non-linear Conjugate-Gradient, Steepest-Descent and a Genetic Algorithm implementation. Several tests are performed and the preliminary results are presented and discussed.

Keywords: Reservoir simulation, History Matching, Optimization.

1 Introduction

Reservoir simulation is a powerful tool that has been extensively used in reservoir engineering. It combines physics, mathematics, reservoir engineering and computer programming. One of the main goals of the models deals with the ability to predict the behavior of a reservoir. Unfortunately, the computational models depend on many parameters and features of the reservoir and the prediction's performance of a model depends on good estimations of some physical properties, such as the permeability distribution of the reservoir. Several difficulties arise during the validation of a model, since most of the oil reservoirs are inconveniently buried beneath thousands of feet of overburden. Direct observations of the reservoir are available only at well locations that are often hundreds of meters [1].

An alternative for model validation is the estimation of the relevant properties by History Matching[2]. The History Matching process is an inverse problem

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 377–386, 2009.

that utilizes reservoir simulation to find a set of parameters that minimizes the difference between the model performance and the historical performance of the field. This process can be made manually or automatically.

Traditional Newton-like methods have been used before[3]. In addition, free-derivative methods based on Genetic Algorithms were proposed in[4]. This paper presents a comparison of different optimization methods used to perform the automatic history matching in a 2D flow model. In particular we compare the Quasi-Newton method, Conjugate-Gradient, Steepest-Descent and a Genetic Algorithm implementation.

The paper is organized as follows: Section 2 introduces the direct problem formulation and implementation. Section 3 introduces the inverse problem theory and the methods implemented. Section 4 presents the methods and the computer platform used for the tests. Section 5 and 6 present the results and conclusion of this work, respectively.

2 Forward Problem

2.1 Theory

The problem treated in this paper is a two dimensional two-phase (water/oil) incompressible and immiscible porous media flow in a gravity-free environment [12]. The system of partial differential equations which governs this flow is derived from the *law of mass conservation* and the *Darcy Law*. The law of mass conservation for both phases is written as $\phi\partial_t(\rho_\alpha s_\alpha) + \nabla.(\rho_\alpha v_\alpha) = Q_\alpha$, where $\alpha = w$ denotes the water phase, $\alpha = o$ denotes the oil phase, ϕ is the porosity of the porous medium, and ρ_α, s_α, v_α and Q_α are, respectively, the density, saturation, volumetric velocity and flow rate in wells of the α-phase. The volumetric velocity (v_α) is given by the Darcy law: $v_\alpha = \frac{Kk_{r\alpha}(s_\alpha)}{\mu_\alpha}\nabla p_\alpha$, where K is the effective permeability of the porous medium, $k_{r\alpha}$ is the relative permeability of α-phase, which is a function that depends on saturation, and μ_α and p_α are, respectively, viscosity and pressure of the α-phase. In this work we consider that the capillary pressure is null, that is, $p_w = p_o$. So, from now on we will refer to pressure simply as p. We also have that $s_w + s_o = 1$. We introduce the phase mobility and transmissibility functions, respectively: $\lambda_\alpha(s) = \frac{k_{r\alpha}(s)}{\mu_\alpha}$, $T_\alpha(s) = K\lambda_\alpha$, where $s = s_w$ from now on. The volumetric velocity can then be written as $v_\alpha = -T_\alpha\nabla p$. We assume that the phases density and viscosity are constant and get

$$\begin{cases} \phi\rho_w\partial_t s_w + \rho_w\nabla v_w = Q_w \\ \phi\rho_o\partial_t s_o + \rho_o\nabla v_w = Q_o\ . \end{cases} \tag{1}$$

Now we can divide the equations in 1 by ρ_α and sum both and get

$$\begin{cases} \phi\partial_t s + \nabla v_w = q_w \\ \nabla v_t = q_t\ . \end{cases} \tag{2}$$

where $q_\alpha = \frac{Q_\alpha}{\rho_\alpha}$ is the flow rate density of α-phase, $q_t = q_w + q_o$ and $v_t = v_w + v_o$. Defining total mobility as $\lambda_t = \lambda_w + \lambda_o$ we introduce the fractional flow functions as $f(s) = \frac{T_w}{T_t} = \frac{\lambda_w}{\lambda_t}$. System 1 is then rewritten as

$$\begin{cases} \phi \partial_t s - \nabla(f(s)T_t(s)\nabla p) = q_w \\ -\nabla(T_t(s)\nabla p) = q_t \ . \end{cases} \tag{3}$$

To complete the model the boundary conditions must be specified. In this paper we consider no flow boundary condition, $v_\alpha.\nu = 0, x \in \partial\Omega$, where ν is the outer unit normal to the boundary $\partial\Omega$ of the domain Ω. Finally we define the initial condition given by $s(x,0) = s_0(x), x \in \Omega$.

The forwad problem treated on this paper is the system of partial differential equations given by 3 with the boundary and initial conditions given above.

2.2 Implementation

The differential equations described in Sect. 2.1 are nonlinear and coupled. In this work the method used to solve these equations is the so called IMPES. Our implementation of the IMPES methods adopts an adaptive time step scheme. The basic idea of the IMPES method is to separate the computation of pressure from that of saturation. The coupled system is split into a pressure equation and a saturation equation, and the pressure and saturation equations are solved using implicit and explicit time approximation approaches, respectively. Decoupling the system 3 we get an elliptic equation for pressure given by (4) and a nonlinear hyperbolic equation for saturation, given by (5).

$$-\nabla(T_t(s)\nabla p) = q_t \ . \tag{4}$$

$$\phi \partial_t s - \nabla(f(s)T_t(s)\nabla p) = q_w \ . \tag{5}$$

For the pressure computation, the saturation s in (4) is supposed to be known and 4 is solved implicitly for p. In this work, the finite volume method was used for spatial discretization[5]. As mentioned before, the saturation equation given by 5 is solved explicitly. The IMPES method goes as follows: given s^0; for $n = 0, 1, ...$ we use (4) and s^n to evaluate p^n; next we use (5), s^n and p^n to evaluate s^{n+1}. To guaranty the stability of this equation the time step Δt must be sufficiently small which is an expensive requirement. To minimize this problem, we actually used an Improved IMPES method [6]. This method uses the fact that pressure changes less rapidly in time than saturation. Knowing this, it is appropriate to take a much larger time step for the pressure than for the saturation. Using the Improved IMPES method we have two different time steps: Δt^n for pressure and $\Delta t^{n,l}$ for saturation. Pressure p^n corresponds to instant $t^n = \sum_{1<=i<=n} \Delta t^i$ and saturation $s^{n,l}$ corresponds to instant $t^{n,l} = t^n + \sum_{1<=j<=l} \Delta t^{n,j}$. We deduced the CFL conditions given by the next two equations. One for cells that have injector wells $\Delta t^{n,l} \leq \frac{\phi(1-s_{o,res}-s_{i,j}^{n,l})}{\beta_1 q_w^{n,l}(1-f(s_{i,j}^{n,l}))}$, where

$\beta_1 > 1$ and another to the other cells, given by $max f'(s_m) \sum_m \frac{\Delta t^{n,l}}{\phi \Delta_m} |v_m^n| \leq \beta_2$, where $0 < \beta_2 < 1$ and m corresponds to interfaces where the flow enters the block. To control the pressure time step Δt^n we calculate the pressure variation percentage $VP^n = \frac{||p^{n+1} - p^n||}{||p^n||}$. If VP^n is greater than a given VP_{max} pressure time is reduced and if it is less then a given VP_{min}, it is increased.

3 Inverse Problem

3.1 Theory

In this work, the inverse problem proposed aims to estimate the absolute permeability field of a reservoir by history-matching its production data. We denote by K the vector of permeability to be determined and by O the vector of production observations and define as $u = (s, p)$ the vector of the forward problem unknowns (saturation and pressure). We have that u depends on the permeability $u = u(K)$ and O depends on both, permeability and u, $O(K) = O(K, u(K))$. If $\bar{O}$ is the vector with the real observations we can search K that minimizes the least square formulation

$$f(K) = \|O(K) - \overline{O}\|^2. \tag{6}$$

Note however that this is a contrained minimization problem since permeability is a strictly positive property. In this work, we transformed this problem in an unconstrained minimization problem via the change of variable $m_i = ln(K_i)$. From now on, the parameters to be estimated are those of the vector m. There are several ways for solving this optimization problem. In this work we compare two different approaches to optimize the proposed problem: Newton-like methods and Genetic Algorithm.

3.2 Newton-Like Methods

Newton's iterative method for minimizing $f(x)$ is $x_{k+1} = x_k + \alpha_k H_k^{-1} \nabla f^T$, where H_k is the Hessian matrix of f in iteration k. However the computation of H is almost allways unaffordable. The idea underlying Newton-like methods is to use an approximation to the inverse of the Hessian in place of the true inverse that is required on Newton's method [7]. In this work we use the *steepest descent*, the *conjugate gradient* and a *Quasi-Newton* methods. In steepest descent method, the inverse of H is taken as the identity matrix I. The Quasi-Newton Method iteratively builds up an approximation to the Hessian by keeping track of gradient differences along each step taken by the algorithm. The nonlinear conjugate gradient method is mainly an extension of the linear conjugate gradient method but it may be also consider a specialization of limited-memory Quasi-Newton methods.

3.3 Genetic Algorithms

Genetic Algorithm (GA) is a numerical optimization algorithm inspired in natural selection and natural genetics created by Holland in the 60th decade [8]. It is an alternative optimization technique since it is stochastic and does not need derivatives. In each generation, which corresponds to an iteration of the algorithm, GA has not only one possible solution. Instead, it keeps a population of individuals each representing a potential solution to the problem to be solved. This population evolves through generations using genetic mechanisms in search for an optimal individual. In order to use GA it is important to define some characteristics such as the representation of an individual, a fitness function and the genetic operators to be used. In this work we used the real-code representation which is commonly used in problems with continuous variables. In this representation the individual is a set of real values that represents a feasible solution. Every individual in the population is assigned, by means of a fitness function, a measure of its goodness with respect to the problem under consideration [8]. The fitness function used in this work is given by (6). The most important genetic operators applied in GA are selection, crossover, mutation and elitism. Selection is used to pick individuals to generate offsprings. The approach used in this work is the rank-based selection. Crossover is the operator that combines two individuals to generate a third one. In this work we used the *Blended-crossover* approach. Mutation is an operator applied in one individual. It introduces diversity to the population and prevents the algorithm of being stuck in local minimums. The GA begins with an initial population that may or not be generated randomly. To evolve to a next generation all individuals of the population must be evaluated using the fitness function. After this, selection is applied to choose which individuals will generate offsprings through crossover or/and mutation. This process is repeated until a good individual is found.

4 Methods

4.1 Implementation Details and Computer Platform

The numerical solution of the forward problem was implemented in C++. To solve the linear systems associated to the discretization of the Partial Differential Equations the PETSc library was used [9]. The GSL library [10] was used for the optimization with the Newton-like methods. In this work we used the *steepest descent*, *nonlinear conjugate gradient* and *Quasi-Newton* methods. These methods need the objective function (6) and its gradient ∇f to be evaluated for each solution candidate m. The objective function is calculated via the solution of the forward problem. The calculation of the gradient ∇f is obtained via finite difference and it involves performing $n+1$ solutions of the forward problem, where n is the dimension of the permeability vector to be estimated. The calculation of the gradient ∇f was implemented in parallel using the MPI library[11] with each component of the gradient vector calculated by a different process. The GA was implemented in C++. The implementation also exploits parallelism via

the MPI library and a master-slave decomposition strategy. The master process implements all the GA operations and requests the slave processes to perform the fitness evaluation of the solution candidates, or individuals, of the current population. Again, each fitness calculation given by (6) involves the solution of the forward problem. The algorithms were executed in a small cluster composed of 8 Intel Xeon (2GHz) processing cores connected by a 1000MBits Ethernet switch.

4.2 Numerical Experiments

The reservoir simulation we consider in this work is the classical five-spot configuration with 4 injection wells in the corners of the reservoir and one production well in its center (see Figure 1). The reservoir is a square of sizes equal to $200m$. Each injection well injects a total of $100m^3$ per day. The reservoir's history is given by the oil production of the center well during 350 days of simulation. The other parameters of the model are: porosity (0.2), relative permeability, given by the Corey curve, irreducible water saturation ($s_{iw} = 0.2$) and residual oil saturation ($s_{ro} = 0.2$). The spatial discretization used was.$\Delta x = \Delta y = 7.4m$. In this work, three different synthetic histories were generated from different reservoirs that differ only on the number of rectangular regions with different permeability values: a reservoir model with two different permeability blocks mapped as a 1x2 mesh (half by half), $K = (47.82, 142.5)$ (natural ordering); a model with an uniform 2x2 permeability mesh, $K = (39.12, 67.99, 52.85, 267.68)$; and an uniform 3x3 permeability mesh with $K = (43.22, 38.21, 55.76, 56.39, 95.61, 148.45, 36.84, 135.84, 261.57)$ These three synthetic histories are the targets of three different history-matching problems. Each one of these inverse problem was solved by the Newton-like methods and the GA, and each of the optimization methods were executed 10 times with different initial guesses. The population of the GA

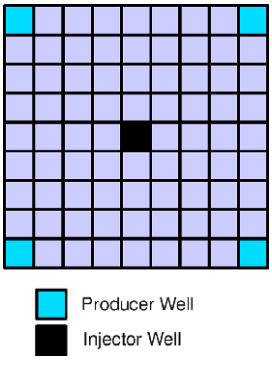

Fig. 1. 5-spot

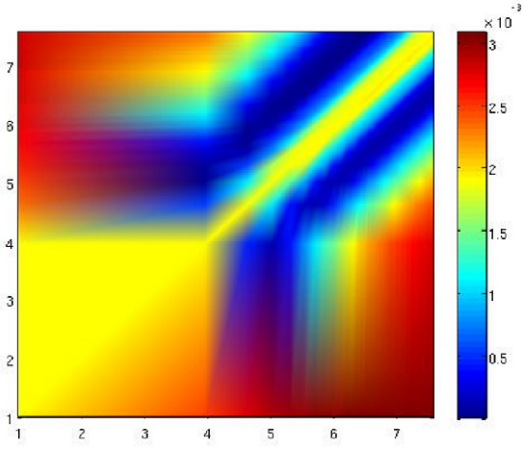

Fig. 2. Objective Function

has a size of 100 and the mutation rate was 0.025. Convergence is achieved when a solution candidate m satisfies $f(m) < 10^{-6}$. All initial guesses were randomly generated but satisfy $f(m) > 10^{-3}$. Therefore, after convergence the objective function is decreased by at least 3 orders of magnitude. In the case of the Newton-like methods there is also a stop criterion to handle local minima: $\nabla f(m) < 10^{-8}$. In addition, for all methods we implemented a stagnation criterion: the method stops if after 3 consecutive iterations $f(m)$ is decreased by less than 10^{-6}.

5 Results

Figure 2 shows the objective function for the problem that has two parameters or permeability values to be estimated. We note that there are infinitely many global minima and that the shape of the function is symmetric with respect to the line $m = \alpha(1, 1)$. The reason we have many global minima lies in the following observation. Imagine two simulations where in the second one the permeability is given by $K_2 = \beta K_1$, with K_1 the permeability of the first simulation. Let us further identify this simulations with s_1, p_1, K_1 and s_2, p_2, K_2. It follows that from (4) we have $\nabla(f(s_2)T_{t2}(s_2)\nabla p_2) = \nabla(f(s_2)K_2\lambda_t(s_2)\nabla p_2) = \beta\nabla(f(s_2)K_1\lambda_t(s_2)\nabla p_2) = \nabla(f(s_1)K_1\lambda_t(s_1)\nabla p_1)$. Using this in (5) we observe that for both simulations the saturation equation becomes exactly the same. Thus, the production history will be the same for these simulation with different permeability maps. Now, the fact that the shape of the objective function is symmetric is a particular feature of the simulation we perform, the classical five-spot. Since water injection is the same in all injection wells, 180^o rotations of the permeability distribution map do not alter the production of the centered well.

For the simulations with the two parameters to be estimated we have performed 4 different set of tests. Set 1, 2, 3 and 4 have initial guesses (initial population for the GA) randomly chosen but that satisfy $f(m) \geq 2.5\,10^{-3}$, $f(m) \geq 2.0\,10^{-3}$, $f(m) \geq 1.5\,10^{-3}$ and $f(m) \geq 1.0\,10^{-3}$, respectively. For the Newton-like methods, the initial guess was taken as the permeability pair with the smallest objective function value in the corresponding GA initial population. In each test the algorithms were executed 10 times. Table 1 presents the results in terms of successfull convergence of the methods. The GA is the most robust method. It has always converged to a global minimum. The performance of the Newton-like methods clearly depended on the proximity to a global minima. The performance improves if the initial guess is close to the minima. The Steepest-Descent (SD) achieved the worst result.

Table 2 presents some statistics of the results for set # 3. The general behavior of the methods were similar for the other test sets. The results obtained by the GA in terms of error (fitness) average and standard deviation (std) are at least an order of magnitude smaller than those obtained by the CG (Conjugate Gradient) and QN (Quasi-Newton) methods and two orders of magnitude better than those of the SD method (Steepest Descent). However, the computational cost of the

Table 1. Successfull convergence with 2 parameters

Initial guess (m)	GA	CG	QN	SD
$f(m) \geq 2.5\,10^{-3}$	10	3	3	2
$f(m) \geq 2.0\,10^{-3}$	10	4	4	4
$f(m) \geq 1.5\,10^{-3}$	10	6	6	3
$f(m) \geq 1.0\,10^{-3}$	10	7	7	4

Table 2. 2 parameters statistics- initial guess m - $f(m) \geq 1.5\,10^{-3}$

	Best fit	Mean	Std	# of eval. (fastest)	Total # eval.
GA	1.587007e-09	1.4882e-07	2.0064e-07	208	3236
CG	2.868986e-16	6.6401e-05	9.9601e-05	32	486
QN	3.400530e-17	6.6401e-05	9.9601e-05	32	462
SD	5.446458e-07	1.3550e-04	1.0114e-04	25	700

GA in terms of total number of objective function evaluations is around 8 (6) times higher than that of the CG and QN methods (SD).

For the tests with 4 and 9 parameters we chose to compare the GA against the CG method, as this method and the QN method achieved very similar performance in the tests with 2 parameters. The two algorithms were executed 10 times with different initial guesses for the inverse problem with 4 parameters. The GA successfully converged 6 times whereas the CG converged only once to the desired error tolerance of 10^{-6}. Table 3 presents some statistics of the results. The results obtained by the GA in terms of error average and standard deviation (std) are near two order of magnitudes smaller than those obtained by the CG. However, the computational cost of the GA is around 6 times higher than that of the CG. The fastest execution of the GA needed 10 times more evaluations of the objective function than the CG method. Figure 4 presents the evolution of the executions of the GA and CG methods. The GA is quite robust in terms of convergence and the executions perform similarly whereas the evolution of a CG execution highly depends on the initial guess.

Table 4 presents some statistics of the results obtained by the methods for the History-matching problem with 9 parameters. The two algorithms were executed 10 times with different initial guesses. The CG successfully converged twice whereas the GA converged only once. Nevertheless, the results obtained by the GA in terms of error average and standard deviation are smaller than those obtained by the CG. However, the computational cost of the GA is around

Table 3. Statistics of the tests with 4 parameters

	Best fit	Mean	Std	# of eval. (fastest)	Total # eval.
GA	1.2590e-07	4.7753e-06	5.4781e-06	624	18489
CG	3.000341e-07	1.7023e-04	2.7912e-04	62	2989

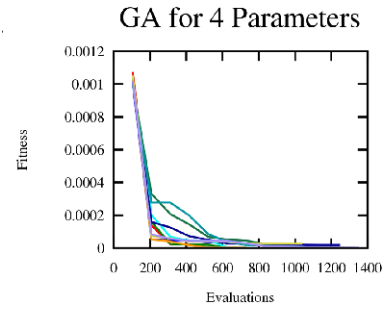

Fig. 3. GA performance for 4 parameters

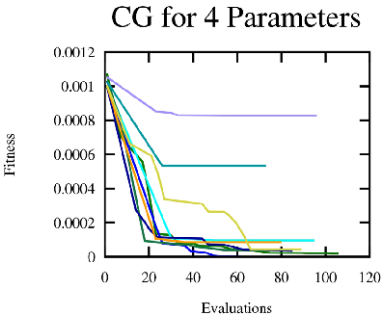

Fig. 4. CG performance for 4 parameters

Table 4. Statistics of the tests with 9 parameters

	Best fit	Mean	Std	# of eval. (fastest)	Total # eval.
GA	9.479320e-07	3.8718e-06	3.0535e-06	4160	50960
CG	9.197778e-07	6.1349e-05	8.2611e-05	252	3940

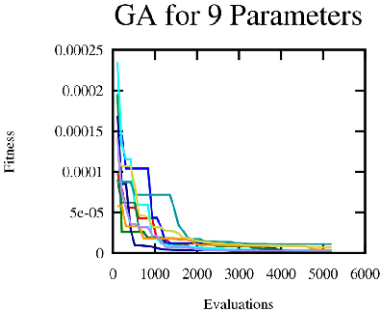

Fig. 5. GA performance for 9 parameters

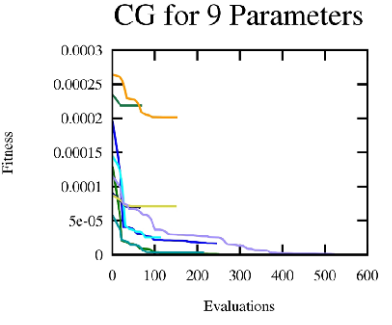

Fig. 6. CG performance for 9 parameters

13 times higher than that of the CG. Figure 6 presents the evolution of the executions of the GA and CG methods. Again, the evolution of the GA executions were very similar whereas the evolution of a CG execution highly depends on the initial guess.

6 Conclusion

This work presents a comparison of different optimization methods for the automatic history matching problem of reservoir simulation. The computational model implemented was based on the formulation of a two dimensional two-phase (water/oil) incompressible and immiscible flow in a gravity-free porous media. The reservoir simulation considered was the classical 5-spot configuration. The

inverse problem proposed aims to estimate the absolute permeability distribution of the reservoir by history-matching its production data. Derivative-based methods were compared to a free-derivative algorithm. In particular, we compared the Quasi-Newton method (QN), non-linear Conjugate-Gradient (CG), Steepest-Descent (SD) and a Genetic Algorithm (GA) implementation. The GA presented the most consistent results in terms of accuracy and convergence but it was computationally very expensive. On the other side, the performances of CG and QN methods were very dependent on the initial guesses. Their results were less consistent, but the methods demanded few evaluations of the objective function and therefore few executions of the reservoir simulator. In conclusion, the preliminary results suggests a tradeoff in terms of robustness, towards the Genetic Algorithm, and speed, which favors the Newton-like methods.

Acknowledgment. The authors thank PETROBRAS for the financial support.

References

1. Oliver, D.S., Albert, C.R., Liu, N.: Inverse theory for Petroleum Reservoir Characterization and History Matching. Cambridge University Press, Cambridge (2008)
2. Watson, A.T., Wade, J.G., Ewing, R.E.: Parameter and system identification for fluid flow in underground reservoirs. In: Conference on Inverse Problems and Optimal Design in Industry, Philadelphia (1994)
3. Brun, B., Gosselin, O., Barker, J.W.: Use of Prior Information in Gradient-Based History-Matching. In: SPE Reservoir Simulation Symposium, pp. 13–23 (2001)
4. Soleng, H.: Oil reservoir production forecasting with uncertainty estimation using genetic algorithms. In: Congress on Evolutionary Computation, pp. 1217–1223 (1999)
5. Versteeg, H., Malalasekra, W.: An Introduction to Computational Fluid Dynamics: The Finite Volume Method, 2nd edn. Prentice Hall, Harlow (2007)
6. Chen, Z., Huan, G., Li, B.: An improved IMPES method for two-phase flow in porous media. Transport in Porous Media 32, 261–276 (2004)
7. Luenberg, D.G.: Introduction to Linear and Nonlinear Programming. Addison-Wesley, Reading (1973)
8. Coley, D.A.: An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific Publishing Company, River Egde (1997)
9. Balay, S., Buschelman, K., Gropp, W.D., et al.: PETSc users manual. Technical Report ANL-95/11, Argonne National Laboratory (2002)
10. Galassi, M., Davies, J., Theiler, J., et al.: GNU Scientific Library Reference Manual. Network Theory, Bristol (2006)
11. Message Passing Interface Forum: MPI, a message-passing interface standard. Oregon Graduate Institute School of Science & Engineering (1994)
12. Chen, Z.: Reservoir Simulation: Mathematical Techniques in Oil Recovery. Society for Industrial and Applied Mathematics (2007)