INTEGRATION OF WELL TEST DATA INTO STOCHASTIC MODELING

A REPORT
SUBMITTED TO THE DEPARTMENT OF PETROLEUM ENGINEERING
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Eric Tauzin
April 1995

I certify that I have read this report and that in my opinion it is fully adequate, in scope and in quality, as partial fulfillment of the degree of Master of Science in Petroleum Engineering.

Dr. Roland Horne
(Principal advisor)

Acknowledgements

The author wishes to express his gratitude to his research advisor Roland N. Horne for his constant support, encouragement and guidance during the course of this study. Sincere thanks are also due to Jean-Luc Boutaud de la Combe, from Elf Aquitaine Production, and Reidar B. Bratvold, from Odin Reservoir Software services AS.

Financial support from Elf Aquitaine Production, Odin and Statoil, and from the Stanford university Research Consortium for Innovation in Well Test Analysis (Supri-D) are gratefully acknowledged.

Abstract

This study developed an algorithm based on Simulated Annealing to constrain the permeability and porosity distributions of a given reservoir model to the well test data collected at several wells. The technique can be used for single or multiple well tests.

In order to keep the execution time of this algorithm within an acceptable range, the perturbation on the pressure transient due to a local heterogeneity is approximated by an analytic influence function. The results given by this approximation were compared to the results given by a simulator, and found to be reliable.

The algorithm was tested on several examples, showing that the use of the analytic influence function allows considerable reduction in the computing time. Moreover, it was shown that such a constraining leads to a better prediction of a waterflood performed during a longer period of time.

Contents

Acknowledgements iii

Abstract iv

Table of Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Simulated Annealing 3

2.1 Concept and Algorithm 3

2.2 Post-Processing with Simulated Annealing 4

2.3 The Execution Time Problem 5

3 The Influence Function 6

3.1 Definition 6

3.2 Characteristics 10

3.3 Validity of the Influence Function 13

3.3.1 Approximation of Block Influence 13

3.3.2 Approximation of Influence in Heterogeneous Reservoirs . . . 13

4 The Influence Function Algorithm **20**
4.1 Modifying Simulated Annealing 20
4.1.1 The Perturbation Procedure 21
4.1.2 The Decision Making 21
4.1.3 The Updating Procedure 22
4.2 A Well Test Post-Processor 22
4.2.1 Convergence Criterion and Objective Function 22
4.2.2 Algorithm Flow Chart 23

5 Example of Application **25**
5.1 Case Studied 25
5.1.1 The Reference Field 25
5.1.2 Stochastic Simulations 26
5.1.3 Sensitivity Study 27
5.2 Postprocessing Stochastic Realizations 32
5.2.1 Description of the Runs 32
5.2.2 Post-Processed Grids 33
5.2.3 Improving a Waterflood Forecast 33

6 Concluding Remarks **42**

7 Nomenclature and Bibliography **44**

A Computer Programs **49**

Appendix **49**
A.1 *influ.help* - Help File 50
A.2 *influ.par* - Parameter File 55
A.3 *influ.f* - Main Fortran Program 59
A.4 *influ.inc* - Definition of Variables 127

List of Tables

5.1 Petrophysical Parameters 26
5.2 Well Parameters 26
5.3 Sensitivity Study 28
5.4 Run Parameters 32
5.5 Waterflood Forecast 34

List of Figures

3.1 Geometry of the Analytical Solution. 7
3.2 Shematic Influence of a Circular Discontinuity. 10
3.3 Amplitude of the Influence of a Disc. 12
3.4 The Analytic Influence Function as an Approximation of the Influence of Grid Block Heterogeneities in Infinite Homogeneous Reservoirs. . . 16
3.5 Examples of Heterogeneous Reservoirs; Permeability Distributions and Well Locations. 17
3.6 The Analytic Influence Function as an Approximation of the Influence of Heterogeneities in Heterogeneous Reservoirs; Map of Shifts for Grid (A) and Influences at Block (1,5). 18
3.7 The Analytic Influence Function as an Approximation of the Influence of Heterogeneities in Heterogeneous Reservoirs; Map of Shifts for Grid (B) and Influences at Block (3,3). 19

4.1 Algorithm Flow Chart. 24

5.1 Reference Field. 29
5.2 Reference Pressure Transients. 30
5.3 Example of Stochastic Simulation. 31
5.4 Constraining to the Pressure Data - Best Match 35
5.5 Constraining to the Pressure Data - Worst Match 36
5.6 Example of Post-Processed grid 37
5.7 Connectivity at 10 md 38
5.8 Waterflood Forecast - Worst Case 39

5.9 Waterflood Forecast - Best Case 40
5.10 Waterflood Forecast - Breakthrough Times 41

Section 1

Introduction

The main goal of stochastic modeling is to produce numerical representations of a reservoir close enough to reality to be able to forecast the future performance of the field. From that point of view, the candidate realization should have the same flow characteristics as the true reservoir and should respect the past production of the field.

The flow characteristics of the reservoir can be seen as a transfer function between the flow rates and the pressure responses recorded at the different wells. This function is extremely complex, and depends on the distributions of permeability, thickness, porosity, and also a function of time. However, we can reasonably expect to estimate a good part of the characteristics by performing a well test analysis - a precise history matching over a short amount of time.

Therefore it appears useful to construct numerical representations of the reservoir that respect the well test data. This study investigates a way of constraining the estimates of the permeability/porosity distributions of a given reservoir model to well test data collected at several wells.

The goal is not only to generate numerical representations of the reservoir leading to the desired well test. The resulting image must also satisfy spatial constraints (histograms, variograms, local conditioning, etc...) to be representative of the true

distribution. We can expect the flow production predictions to be more precise and closer to reality if the numerical model is constrained to more data.

The approach we have investigated involves post-processing the stochastic realizations already constrained to certain spatial characteristics (variogram and histogram) in order to constrain them to an additional feature; namely the data from a single or multiple well test.

The original method used was the simulated annealing technique, but the algorithm was modified to reduce the execution time and to keep the original spatial characteristics of the numerical models. The execution time is kept within an acceptable range by the introduction of an analytic influence function which approximates the perturbation on the pressure transient due to a local heterogeneity.

It is shown through an example that this approximation is precise enough to allow a considerable reduction of the execution time. The post-processed grids satisfy the desired flow constraints and their previously constructed spatial features are not significantly altered. Moreover, it is shown that such a treatment leads to a better prediction of a waterflood performed during a longer period of time.

Section 2

Simulated Annealing

2.1 Concept and Algorithm

The simulated annealing technique is a solution method based on an analogy with the physical process of annealing, a process by which a material undergoes extended heating and is slowly cooled. Thermal vibrations permit a reordering of the atoms to a highly structured lattice, i.e. a low energy state. In the context of reservoir characterization, the annealing process can be simulated through the following steps:

1. An initial image of the reservoir is created by assigning random parameter values at each grid block.

2. An objective function (O) is defined as the measure of difference between desired features and those of the realization.

3. The image is perturbed according to a procedure replicating the thermal vibrations in true annealing and a new value of the objective function O_{new} is generated. This procedure can be for example:

(a) Swapping the values of two randomly chosen nodal locations.

(b) Randomly selecting a node and replacing its value with a random selection from the global histogram.

4. The perturbation is accepted if the energy is decreased; it is accepted with a

certain probability if the energy is increased. In practice, if $P[accept]$ is the probability of acceptance of the perturbation:

$$P[accept] = \begin{cases} 1 & \text{if } O_{new} \leq O_{old}. \\ \exp[(O_{old} - O_{new})/T] & \text{otherwise.} \end{cases} \tag{2.1}$$

The parameter T is called the "temperature" of the annealing.

5. The perturbation procedure is continued while reducing the probability with which unfavorable swaps are accepted until a low energy state is achieved. This reduction of probability is realized with the reduction of the parameter T. The temperature must not be lowered too quickly or else the image may be trapped in a suboptimal situation and never converge. The specification of how to lower T is described by a few parameters.

2.2 Post-Processing with Simulated Annealing

A useful application of simulated annealing is to process images that already have the desired spatial features, e.g., the result of stochastic simulations. The initial random image is replaced by a realization that already possesses some of the desired features and an objective function is constructed that imposes additional characteristics.

In such an application the initial schedule cannot be started at too high a temperature, since we want to keep the characteristics of the initial image. To avoid this problem the maximum a posteriori (MAP) algorithm is used. In this variant, the temperature T is always kept at zero. The acceptance probability distribution is given by:

$$P[accept] = \begin{cases} 1 & \text{if } O_{new} \leq O_{old}. \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

In our work, MAP was used to constrain stochastic realizations to well test data

collected at several wells. The definition of the objective function leads to a problem of execution time, which is discussed in the next paragraph.

2.3 The Execution Time Problem

The simulated annealing technique has been very successful when applied to stochastic simulation in geostatistics [1,2]. In this application only a few arithmetic operations are required to update the objective function after a perturbation, since all conventional spatial statistics are updated locally. The slow convergence of the algorithm is then compensated by the fast updating, i.e. the small amount of time required for each step balances the large number of steps necessary to reach convergence.

However this fast updating is not possible when the reservoir characterization is constrained by the pressure transients at a certain number of wells. The objective function is constructed from both the actual pressure data (coming from the true reservoir configuration) and the simulated pressure data (coming from the tested reservoir configuration), and there is no easy algebraic way by which we can obtain these simulated pressures. Numerical simulation is today the only way to compute the "exact" pressure distribution in a heterogeneous reservoir, but it would be impractically slow to systematically use a flow simulator in the simulated annealing algorithm we would need to construct. This work investigated a procedure to replace the simulator run taking account of the full heterogeneous grid by the modification on the pressure due to the local mobility/diffusivity changes taking place in the annealing algorithm. For that purpose we developed an approximation to the influence on the pressure transient of a local heterogeneity in a heterogeneous reservoir.

Section 3

The Influence Function

3.1 Definition

We define the influence function as the influence on the pressure transient of one single heterogeneity (H). This influence is a function of the characteristics of the heterogeneity (location, dimensionless mobility $\lambda_D = \lambda_0/\lambda_H$ with λ_0 mean mobility, dimensionless diffusivity $\eta_D = \eta_0/\eta_H$ with η_0 mean diffusivity), of the characteristics of its neighborhood, of space and of time.

If $\tilde{p}(x, y, z, t)$ is the pressure that the reservoir would experience without the heterogeneity (H) and $f_H(x, y, z, t)$ is the influence function due to (H), the pressure in the complete field can be written as:

$$p(x, y, z, t) = \tilde{p}(x, y, z, t) + f_H(x, y, z, t) \tag{3.1}$$

Since there is no analytic way to define f_H, we are reduced to make assumptions in order to derive an approximation. Oliver[3] found a first order approximation that has been used to characterize the permeability distribution[4,5] , but this approximation is only a function of the distance from the producing wellbore and the recovered reservoir permeability distribution is radial. The influence function needs to deal with the complete plane coordinates ((x, y) or (r, θ) in two dimensions) to constitute an acceptable replacement of the simulator.

Using Green's functions, Rosa and Horne[6] computed the exact dimensionless pressure $\bar{p_D}(r_D, \theta, s)$ in the case of an infinite homogeneous reservoir containing a single circular discontinuity, the producing well being either inside or outside the heterogeneity, see Figure (3.1).

The reservoir is in that case a composite system with two regions:

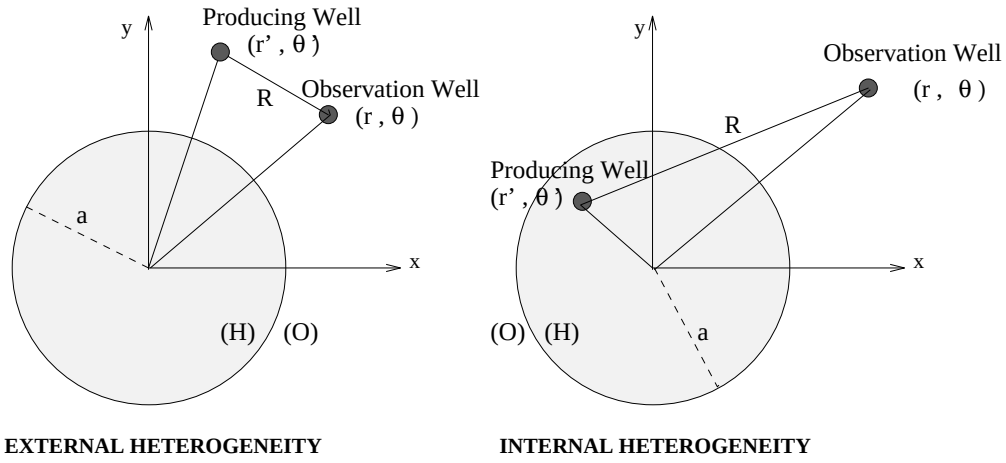


Figure 3.1: Geometry of the Analytical Solution.

$$\begin{cases} -\text{Region (H)} : 0 \leq r < a; \text{ heterogeneity (H) with } \eta_H \text{ and } \lambda_H. \\ -\text{Region (O)} : r \geq a; \text{ homogeneous medium with } \eta_0 \text{ and } \lambda_0. \end{cases}$$

The dimensionless pressure, expressed in the Laplace space, is $\bar{p_{DH}}(r_D, \theta, s)$ in the region (H), and $\bar{p_{D0}}(r_D, \theta, s)$ in the region (0).

If the producing well is external to the circular discontinuity, the pressure response can be written as:

$$\bar{p_{DH}}(r_D,\theta,s) = \frac{\eta_D}{\lambda_D}\frac{1}{s}\sum_{n=0}^{\infty}\epsilon_n \frac{K_n(r'_D\sqrt{s})}{I_n(a_D\sqrt{s/\eta_D})}\Big[I_n(a_D\sqrt{s}) + \frac{\Phi_n}{\Psi_n}K_n(a_D\sqrt{s})\Big] \times \mathrm{I}_n(r_D\sqrt{s/\eta_D})cos[n(\theta-\theta')] \tag{3.2}$$

$$\bar{p_{D0}}(r_D,\theta,s) = \frac{1}{s}\Big[K_0(R_D\sqrt{s}) + \sum_{n=0}^{\infty}\epsilon_n\frac{\Phi_n}{\Psi_n}K_n(r'_D\sqrt{s})K_n(r_D\sqrt{s}) \times \cos[\mathrm{n}(\theta-\theta')]\Big] \tag{3.3}$$

Where:

$$\frac{\Phi_n}{\Psi_n} = \Big[I_n(a_D\sqrt{s/\eta_D})I'_n(a_D\sqrt{s}) - \lambda_D\sqrt{\frac{1}{\eta_D}}I'_n(a_D\sqrt{s/\eta_D})I_n(a_D\sqrt{s})\Big] \Big/ \Big[\lambda_D\sqrt{\tfrac{1}{\eta_D}}I'_n(a_D\sqrt{s/\eta_D})K_n(a_D\sqrt{s}) - I_n(a_D\sqrt{s/\eta_D})K'_n(a_D\sqrt{s})\Big] \tag{3.4}$$

$$\lambda_D = \frac{k_H}{k_0} \tag{3.5}$$

$$\eta_D = \frac{k_0 k_H}{\phi_0\phi_H} \tag{3.6}$$

$$R_D = R/r_w \tag{3.7}$$

$$r_D = r/r_w \tag{3.8}$$

$$r'_D = r'/r_w \tag{3.9}$$

$$a_D = a/r_w \tag{3.10}$$

$$t_D = \frac{k_0 t}{\phi\mu c_t r_w^2} \tag{3.11}$$

$$p_D = \frac{2\pi k_0 h}{qB\mu}(p_i - p) \tag{3.12}$$

$$\epsilon_n = \begin{cases} 1 \text{ for } n = 0. \\ 2 \text{ for } n \geq 1. \end{cases} \tag{3.13}$$

K_n and I_n are the modified Bessel functions of order n, s is the variable for the Laplace transformation with respect to the dimensionless time t_D.

If the producing well is internal to the circular discontinuity, the pressure response can be written as:

$$\begin{aligned} \bar{p_{DH}}(r_D, \theta, s) = \frac{1}{s}\sum_{n=0}^{\infty} \epsilon_n I_n(r_D\sqrt{s})\Big[K_n(r'_D\sqrt{s}) - \frac{\Omega_n}{\Psi_n} I_n(r'_D\sqrt{s})\Big] \\ \times\cos[\mathrm{n}(\theta - \theta')] \qquad \text{if} \quad 0 \leq r_D < r'_D. \end{aligned} \tag{3.14}$$

r_D and r'_D are inversed if $0 \leq r'_D < r_D$.

$$\begin{aligned} \bar{p_{D0}}(r_D, \theta, s) \quad = \quad \frac{\lambda_D}{a_D}\frac{1}{s}\sum_{n=0}^{\infty} \epsilon_n \frac{1}{\Psi_{nD}} I_n(r'_D\sqrt{s}) K_n(r_D\sqrt{s\eta_D}) \\ \times\cos[\mathrm{n}(\theta - \theta')] \end{aligned} \tag{3.15}$$

Where:

$$\begin{aligned} \frac{\Omega_n}{\Psi_n} = &\Big[\lambda_D K'_n(a_D\sqrt{s}) K_n(a_D\sqrt{\eta_D s}) - \sqrt{\eta_D} K_n(a_D\sqrt{s}) K'_n(a_D\sqrt{\eta_D s})\Big] \\ \Big/ &\Big[\lambda_D I'_n(a_D\sqrt{s}) K_n(a_D\sqrt{\eta_D s}) - \sqrt{\eta_D} I_n(a_D\sqrt{s}) K'_n(a_D\sqrt{\eta_D s})\Big] \end{aligned} \tag{3.16}$$

$$\Psi_{nD} = \lambda_D\sqrt{s} I'_n(a_D\sqrt{s}) K_n(a_D\sqrt{s\eta_D}) - \sqrt{s\eta_D} I_n(a_D\sqrt{s}) K'_n(a_D\sqrt{s\eta_D}) \tag{3.17}$$

Since the pressure in the same reservoir without the heterogeneity (H) is just the line source solution, we can obtain an exact expression of the influence function due to (H):

$$\bar{f_H}(r_D, \theta, s) = \bar{p_D}(r_D, \theta, s) - \frac{K_0(r_D\sqrt{s})}{s} \tag{3.18}$$

3.2 Characteristics

A circular permeability heterogeneity acts on the pressure transient in two different ways depending on the zone, as shown on Figure (3.2):

Zone (a): $\bar{p_D}$ and its derivative increase if $k_H < k_0$ and/or $\phi_H < \phi_0$ or decrease if $k_H > k_0$ and/or $\phi_H > \phi_0$.

Zone (b): The amplitude of the pressure transient remains approximately the same but (H) is felt in the form of a time shift (delay if $k_H < k_0$ and/or $\phi_H < \phi_0$, time advance otherwise).

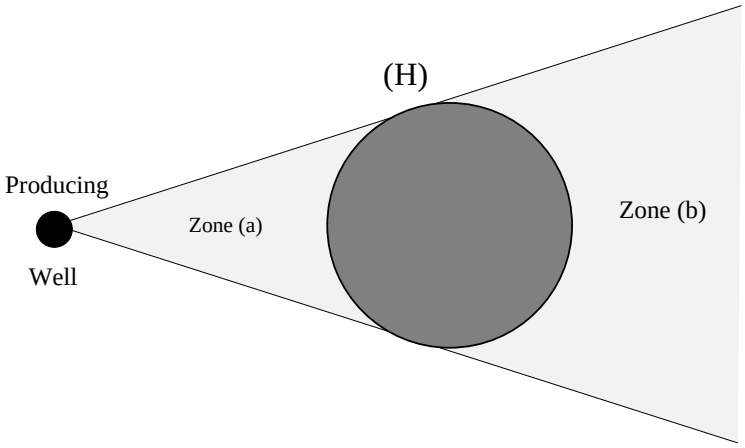


Figure 3.2: Shematic Influence of a Circular Discontinuity.

The influence of the heterogeneity is negligible outside these two zones. Figure (3.3) shows how the amplitude of the influence of such a heterogeneity behaves in space.

The circular discontinuity is located at (x, y)=(150,150) and has a dimensionless radius $r_D = 50$, with $\eta_D = \lambda_D = k_H/k_0$ ($\phi_0 = \phi_H$). The producing well is located at (x, y)=(0,0). The amplitude is computed from the integral over the time of $\mid f_H \mid$. Another interesting feature of $f_H(r_D, \theta, s)$ defined in Eqn.(3.18) is that this solution is still valid in well patterns where more than one well is flowing, according to the principle of reciprocity[7]. However the application of this influence function as an approximation in the simulated annealing algorithm must address two underlying requirements:

(1) The discontinuity must be circular.
(2) The reservoir must be homogeneous and infinite.

Nevertheless we will assume the following hypothesis:
Any real influence on the pressure due to a block (H) of different diffusivity and mobility can be approximated by the influence function computed in Eqn.(3.18) from the corresponding circular discontinuity of the same area.
The validity of this hypothesis will be studied in next section.

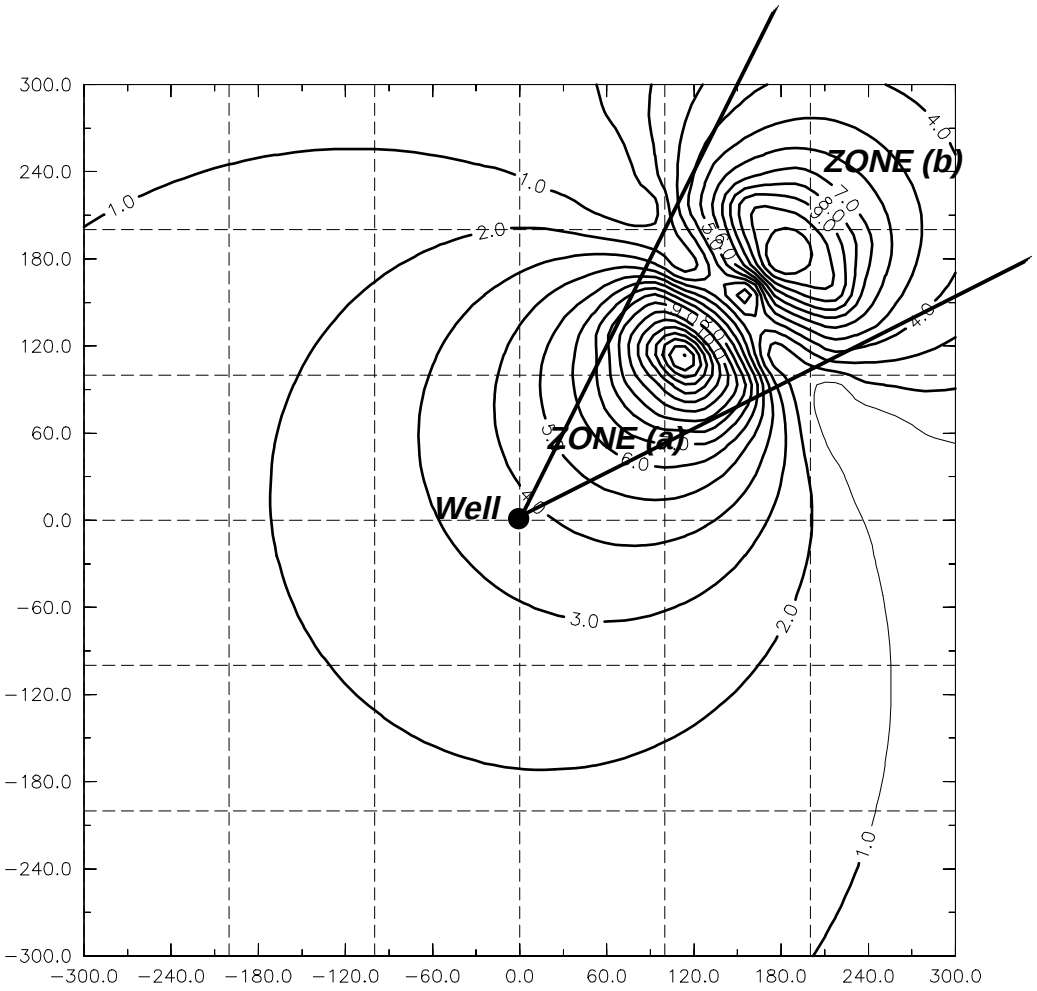


Figure 3.3: Amplitude of the Influence of a Disc.

3.3 Validity of the Influence Function

3.3.1 Approximation of Block Influence

Even if the influence function defined in Eqn.(3.18) applies in theory only for circular discontinuities, it is still a good approximation when the discontinuity is defined in the form of a grid block as in simulator grids. Figure (3.4) illustrates that we only need to multiply the analytic result obtained for a disc by the ratio $(\frac{4}{\pi})$ [area of the square/area of the disc], the grid block influence obtained by a simulation is then very similar to the analytic disc influence. The simulation was performed using a well-known commercial black oil simulator. The results show the influence (pressure - line source) of the heterogeneity (H) if seen as a disc (analytic approximation) or as a grid block (simulator). (H) is located at (x, y)=(150,150) and is such that $\eta_D = \lambda_D = 0.1$, $a_D = 50$. The results are drawn in the Laplace space - the influence function is computed in this space and it has been shown[8] that $s\bar{p}(1/s)$ behaves like $p(t)$. The figures show the pressures at the producing well (P) and an observation well (C) at (250,250).

3.3.2 Approximation of Influence in Heterogeneous Reservoirs

In heterogeneous patterns, the true influence of a given heterogeneity on the pressure transient is highly correlated with the values of diffusivity and mobility in its neighborhood. These correlations are ignored when the influence is approximated by the function defined in the last section. We need to have an understanding of the limitations of such a strong assumption before going further.

For that purpose the validity of the influence function was studied on two example grids shown on Figure (3.5). Both grids (A) and (B) constitute a heterogeneous zone in an infinite homogeneous reservoir. This zone is composed of 49 blocks having a different value of dimensionless permeability $k_D = k_0/k_H$. We simplify the problem by assuming that the porosity field is homogeneous; with this assumption, we can

define $k_D = 1/\lambda_D = 1/\eta_D$. The histogram of the k_D's is the same for the two grids, i.e. the only difference between the two grids is the location of the heterogeneities. Five wells [(P), (A), (B), (C), and (D)] have been drilled in that zone, forming a five spot pattern. A well test is performed at the center well (P).

The influence of each heterogeneity on the five wells during the well test is computed in two different ways:

(1) Analytic approximation: the influence of each grid block is assumed to be close to that of a corresponding disc in an infinite homogeneous medium (i.e. without the 48 other blocks).
(2) Simulator: the influence of the block (i) is equal to the difference between the pressure obtained in the current grid and the pressure obtained in the same grid without the change in properties of block (i).

These two approaches are compared on Grid (A) [Figure (3.6)] and Grid (B) [Figure (3.7)]. The results are drawn in the Laplace space. Each figure shows the value of the "shift" (squared difference between analytic and numerical influences) computed for each grid block heterogeneity and an example of influences found at one block: block (1,5) for Figure (3.6) (low shift of 0.07), block (3,3) for Figure (3.7) (high shift of 1.55). These two examples show that the analytic influence function cannot completely replace the simulator in the simulated annealing algorithm we want to construct. Nonetheless, the influence function is usually sufficiently accurate to predict the direction and the order of magnitude of the perturbation (at 20/24 blocks for Grid (A); 17/24 blocks for Grid (B)), however it is not reliable enough to give a precise value of the perturbation. In other words, the analytic influence function is accurate enough to be used to decide whether the new objective function is lower or higher than the previous one in the algorithm (decision making), however it is not able to give an exact value of this objective function (updating).

The validity of the influence function approximation is governed strongly by the

shape of the grid. In Grid (A) the local heterogeneities do not form any particular pattern and the analytic approximation is usually close to the results given by the simulator. Larger objects are present in Grid (B) and the approximation is incorrect for these patterns since the correlation between the influences has increased.

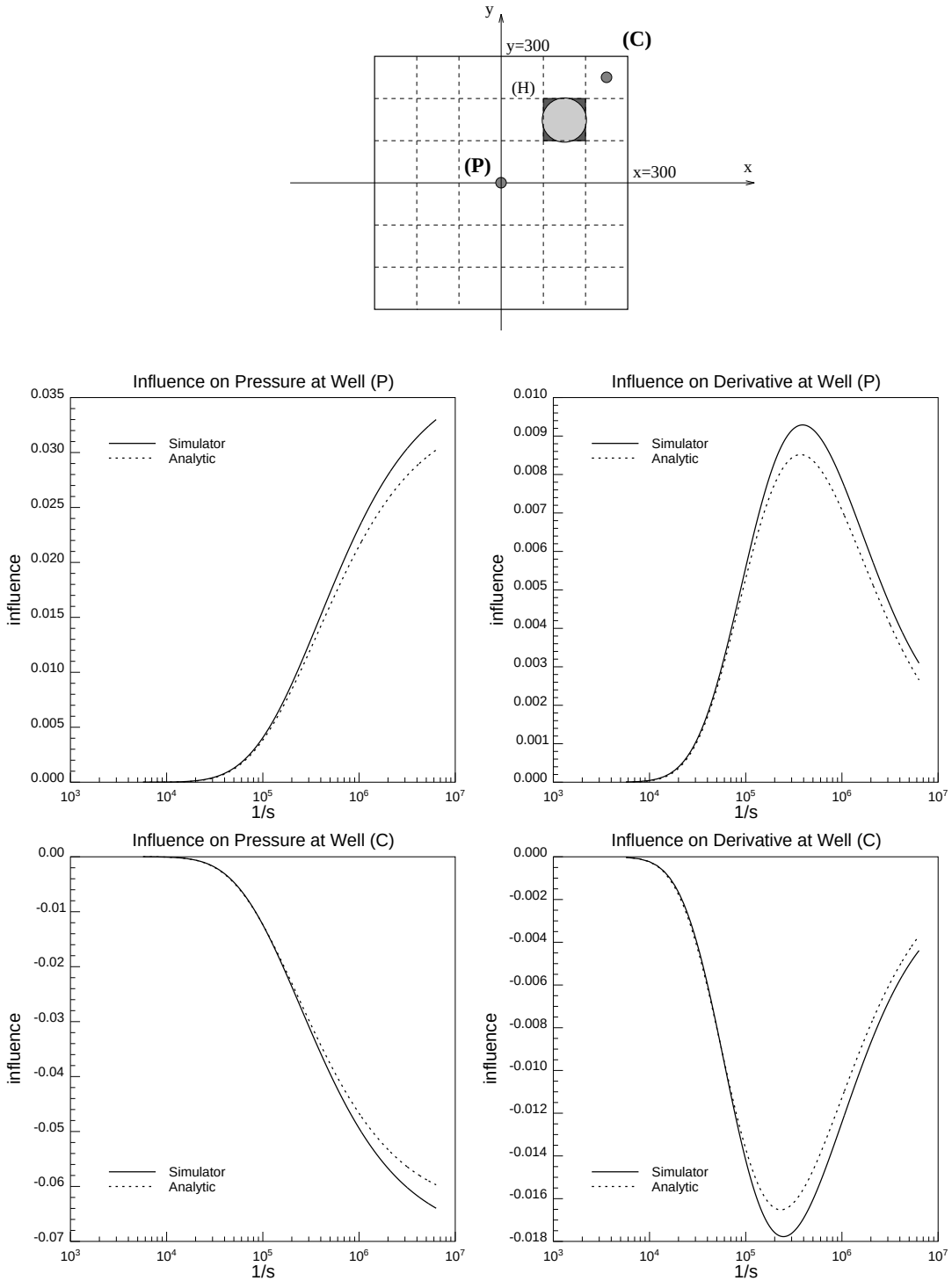


Figure 3.4: The Analytic Influence Function as an Approximation of the Influence of Grid Block Heterogeneities in Infinite Homogeneous Reservoirs.

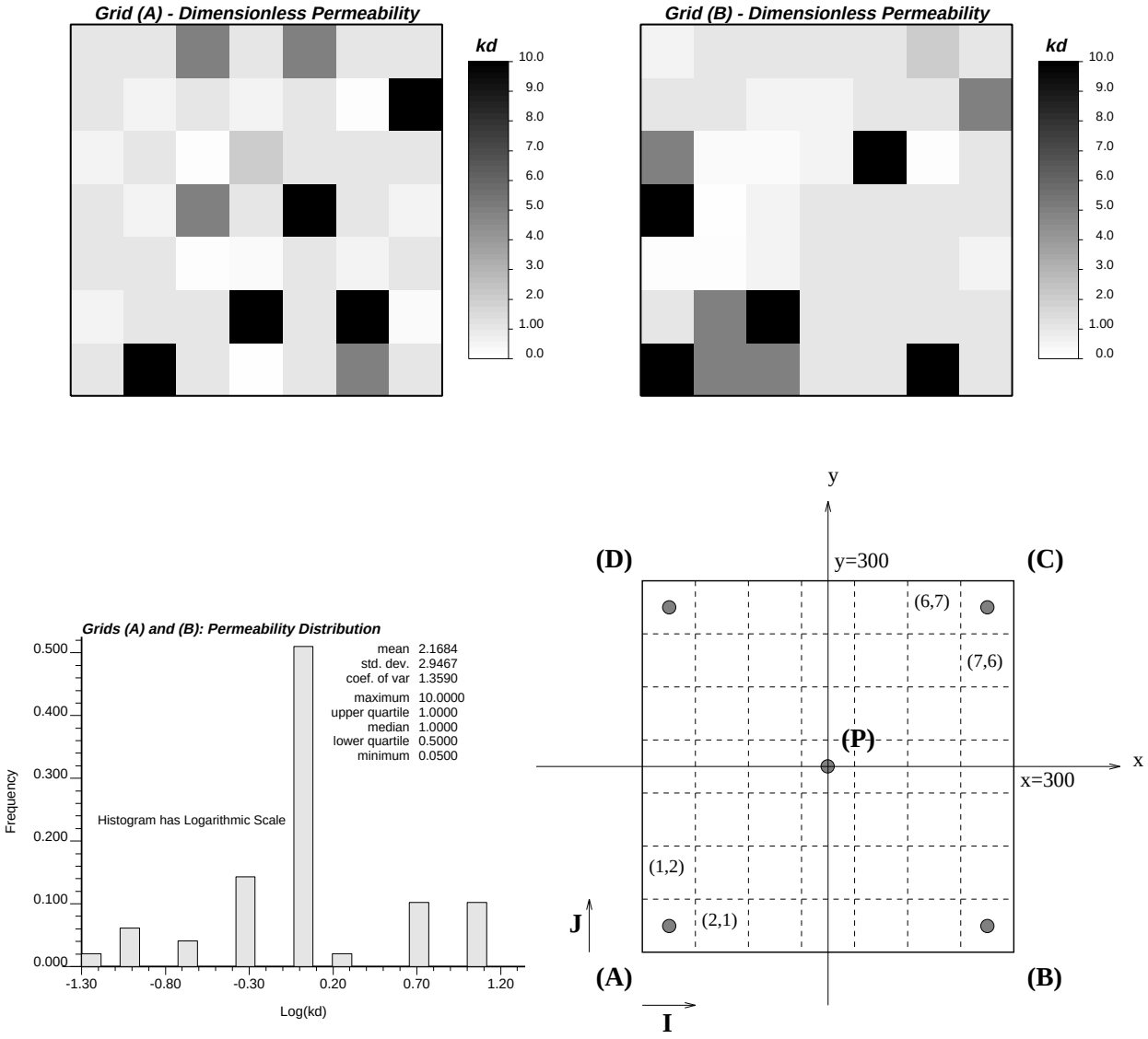


Figure 3.5: Examples of Heterogeneous Reservoirs; Permeability Distributions and Well Locations.

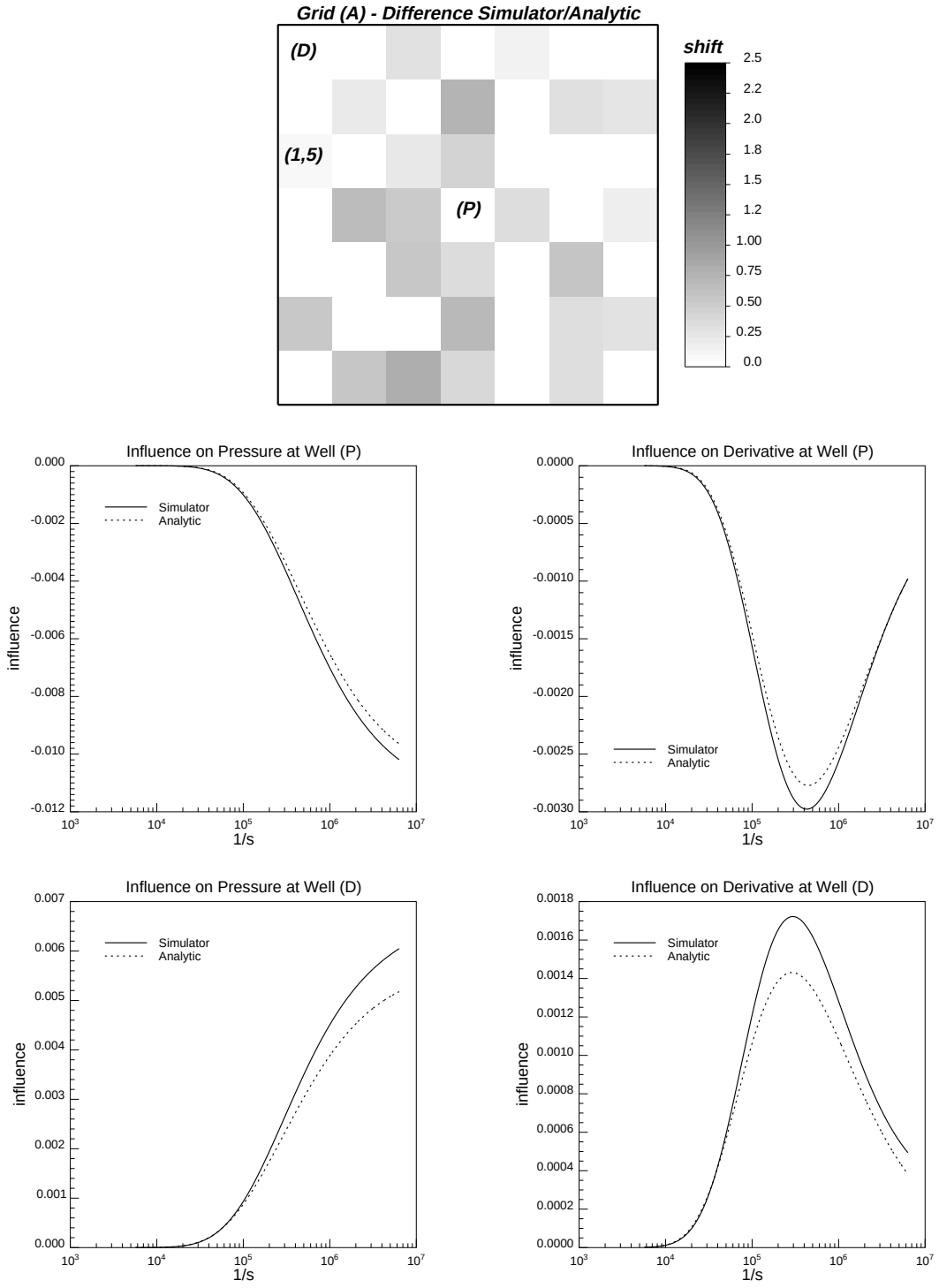


Figure 3.6: The Analytic Influence Function as an Approximation of the Influence of Heterogeneities in Heterogeneous Reservoirs; Map of Shifts for Grid (A) and Influences at Block (1,5).

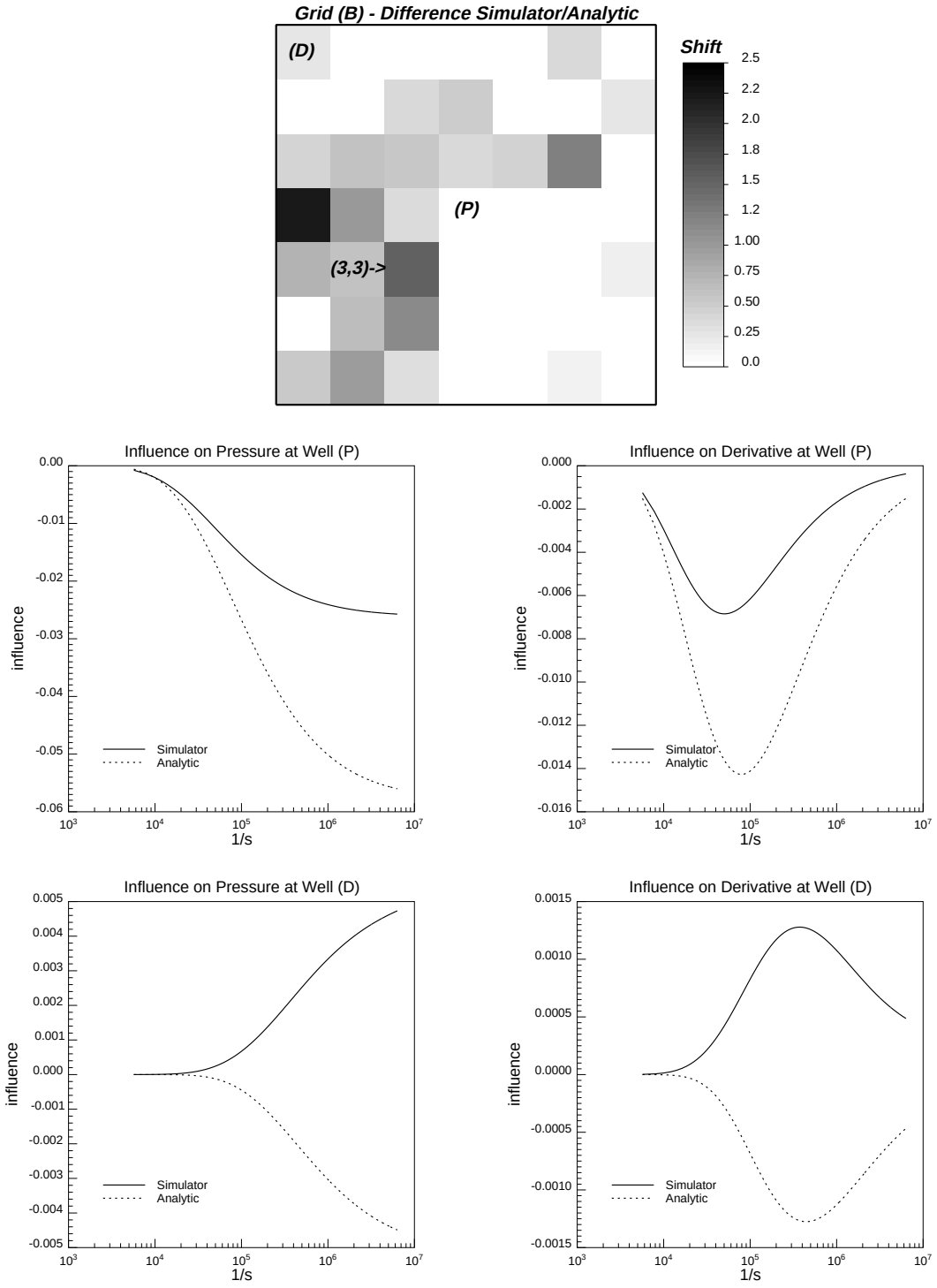


Figure 3.7: The Analytic Influence Function as an Approximation of the Influence of Heterogeneities in Heterogeneous Reservoirs; Map of Shifts for Grid (B) and Influences at Block (3,3).

Section 4

The Influence Function Algorithm

4.1 Modifying Simulated Annealing

The purpose of both flow simulator and analytic approximation is to compute the value of the objective function O related to a given tested configuration:

$$O = \sum_{i=1}^{n_w} \sum_{j=1}^{n_t} \Big[N(\mathrm{p}^{\mathrm{i,j}}_{\mathrm{data}}, \mathrm{p}^{\mathrm{i,j}}_{\mathrm{sim.}}) \Big] \tag{4.1}$$

n_w is the number of wells, n_t the number of time steps; the pressure data corresponding to the true configuration are p_{data} and the simulated pressures corresponding to the tested configuration are $p_{sim.}$. N is a function giving a measure of difference between the two sets of data.

Since the influences of the heterogeneities present in the reservoir are highly correlated, the influence function is not reliable enough to totally replace the flow simulator in the algorithm. The solution is to use sequences of both simulator and analytic approximation, which still leads to an important reduction of execution time. In practice the perturbation procedure, decision making and updating procedure are the three main steps modified by the introduction of the influence function.

4.1.1 The Perturbation Procedure

Swapping mobility and diffusivity values between two blocks leads to the computation of two influences on the pressure (one for each gridblock), whereas replacing the values at one block by variables drawn from a global histogram requires the computation of only one influence. Knowing that the consequence of every single use of the influence function is the introduction of a small error, it appears preferable to work with the second procedure.

In practice, the influence due to the replacement of (η_0, λ_0) in a gridblock by (η_H, λ_H) drawn from the histogram is computed using Equations (3.2) to (3.17), with $\eta_D = \eta_0/\eta_H$ and $\lambda_D = \lambda_0/\lambda_H$.

4.1.2 The Decision Making

The analytic influence function is usually sufficient to determine the sign of $(O_{new} - O_{old})$ and its order of magnitude, and thus allows the algorithm to decide whether to keep or reject the perturbation. Reliability in that decision process becomes more and more important with the number of observation wells from which the objective function is computed. The possible errors occurring at one or two wells are attenuated by the sum taking place in (O). One of the main characteristics of the algorithms derived from simulated annealing is their ability to reach convergence in spite of the acceptance of a few bad perturbations. In simulated annealing this acceptance is decided according to a probability $P[accept]$, this probability being high or low depending on the value of the temperature T. With its *internal* probability to make wrong modifications accepted because it is not fully reliable, the influence function would lead to an artificial increase of this parameter T. For the same reasons, the introduction of the influence function in a MAP algorithm will have for consequence an implicit temperature T slightly higher than zero. The point is that this internal increase will not be high enough to endanger the convergence of the algorithm.

4.1.3 The Updating Procedure

If the perturbation is accepted, O_{old} is replaced by O_{new}, the value of the objective function related to the new configuration. An error in the computation of O_{new} affects the reliability of the following decision making. In the case of N_S consecutive utilizations of the influence function, N_S errors are added and the decision making for the perturbation $(N_S + 1)$ becomes completely random. This updating problem can be solved easily by calling a simulator every N_S accepted perturbations in order to compute a new correct value of the objective function.

4.2 A Well Test Post-Processor

4.2.1 Convergence Criterion and Objective Function

An algorithm constraining stochastic realizations to well test data must be able to reach convergence with only a limited number of accepted perturbations, otherwise the initial image might be randomized and the spatial characteristics might be lost before the end of the run. To avoid this drawback, the algorithm is stopped when the number of accepted perturbations has reached a given criterion. On the other hand, every accepted modification is chosen as the best one among a population of several tried swaps, in order to lower the objective function to a value good enough before the end of the run.

The objective function (O) is computed using the following equation:

$$O = \frac{1}{O_i} \sum_{i=1}^{n_w} \omega_w^i \Big[\sum_{j=1}^{n_t} \Big(\omega_p^i \frac{|\Delta p_{data}^{i,j} - \Delta p_{sim}^{i,j}|}{|\Delta p_{data}^{i,\infty}|} + (1 - \omega_p^i) \frac{|\Delta p'^{i,j}_{data} - \Delta p'^{i,j}_{sim}|}{|\Delta {p'}_{data}^{\,i,\infty}|} \Big) \Big] \qquad (4.2)$$

Where:

ω_w^i is the weight given to well (i).

ω_p^i is the weight given to the pressure data at well (i) (i.e. $(1 - \omega_p^i)$ is the weight given to the derivative).

O_i is the initial value of the objective function.

$\Delta p_{sim} = p_i - p_{sim}$ is the simulated pressure drop (tested configuration).

$\Delta p_{data} = p_i - p_{data}$ is the constraining pressure drop (true reservoir).

$\Delta p'_{data}$ and $\Delta p'_{sim}$ are the corresponding logarithmic derivatives.

$\Delta p^{i,\infty}_{data}$ is the last pressure recorded at the well (i) during the well test, ${\Delta p'_{data}}^{i,\infty}$ its corresponding derivative.

Absolute values are used in Eqn.(4.2) in order to give a comparable weight to the pressure differences at the observation and production wells.

4.2.2 Algorithm Flow Chart

The flow chart shown in Figure (4.1) illustrates the procedure of the algorithm. The convergence of the algorithm is controlled by three main parameters:

N_{max}: Number of perturbations tried before to accept the one leading to the biggest decrease of the objective function.

N_{smax}: Number of perturbations accepted before to update the objective function using a simulator.

N_{tot}: Total maximum number of accepted perturbations.

The notion of temperature is not present in this flow chart, since the algorithm initially chosen was a MAP algorithm.

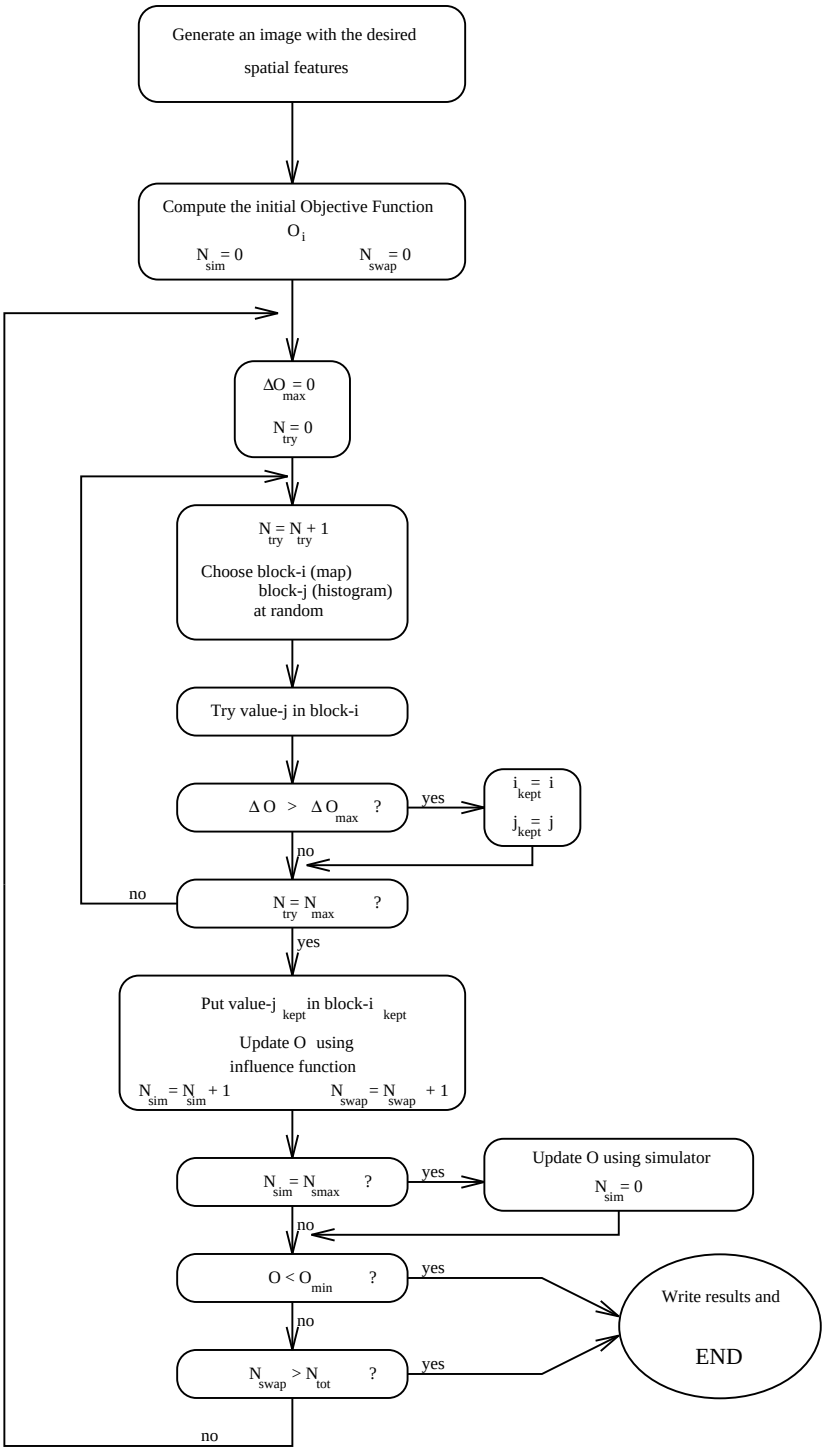


Figure 4.1: Algorithm Flow Chart.

Section 5

Example of Application

5.1 Case Studied

5.1.1 The Reference Field

The example studied in this chapter is a closed reservoir (1000ft×1000ft) modeled as a two-dimensional 40×40×1 grid. Its permeability, porosity and thickness distributions are shown in Figure (5.1). Porosity and permeability are correlated by a simple linear relationship ($\phi = a \times log(k) + b$), but it is important to notice that the method described below does not require any kind of correlation between these parameters - it just makes the stochastic realizations easier to perform. The other reservoir parameters are listed in Table (5.1).

Five wells [(P), (A), (B), (C) and (D)] have been drilled in the reservoir, forming a five spot pattern. Using a well-known commercial simulator, a well test is simulated at the center well (P). The resulting pressure drops are shown on Figure (5.2), and the well parameters are listed in Table (5.2).

An analysis of the pressure drops recorded at well (P) gave $\bar{kh} = 400.0$ md×ft. Knowing that $\bar{h} \approx 40$ ft, we end up with $\bar{k_{W.T.}} \approx 8$ md.

Variable	Value
Initial Pressure, P_i	3500. psia
Formation Volume Factor, B	1.1
Oil Viscosity, μ	2. cp
Total Compressibility, c_t	3×10^{-6} psia^{-1}

Table 5.1: Petrophysical Parameters

Well	(I,J)	r_w (ft)	q (STB/Day)
A	(6,6)	0.5	0.0
B	(35,6)	0.5	0.0
C	(35,35)	0.5	0.0
D	(6,35)	0.5	0.0
P	(19,19)	0.5	400.0

Table 5.2: Well Parameters

5.1.2 Stochastic Simulations

The program SASIM (GSLIB[9]) was used to generate 12 stochastic simulations of the reference grid, using simulated annealing. These realizations are constrained to the permeability variogram and histogram, since permeability and porosity are correlated by a single linear relationship. The thickness distribution is assumed to be known - Section (5.1.3) will show that this assumption is not critical for the rest of the study. There is no local (well) conditioning.

Figure (5.3) shows one of these stochastic simulations, compared with the reference grid. None of the realizations are constrained to the reference well test data, so we can expect a poor replication of this well test. The objective of our approach is to use the influence function algorithm to post-process the stochastic realizations in order to constrain them to the reference well test data.

5.1.3 Sensitivity Study

Before constraining the stochastic realizations to the well test data, it is important to determine, among all the parameters describing the modeled reservoir (porosity, permeability, thickness), the ones playing an important role in the well test. These key parameters will be the variables of the influence function algorithm. A flow simulator was used to compare the effects on the pressure of perturbations on the porosity, permeability and thickness fields The perturbation mechanism was the same as in the influence function algorithm described in Section (4), and this sensitivity analysis was realized on the reference grid, shown on Figure (5.1). The effect E of each perturbation was computed using:

$$E = \sum_{i=1}^{n_w} \sum_{j=1}^{n_t} \frac{|\Delta p_{before}^{i,j} - \Delta p_{after}^{i,j}|}{|\Delta p_{before}^{i,\infty}|} \tag{5.1}$$

Where:

$\Delta p_{before} = p_i - p_{before}$ corresponds to the pressure field before perturbation.
$\Delta p_{after} = p_i - p_{after}$ corresponds to the pressure field after perturbation.

Using this equation, the effects of 100 perturbations (randomly chosen) on the permeability field (only k is changing, ϕ and h remain unchanged) were computed. This was compared to 100 porosity-perturbations, and 100 thickness-perturbations. The main results are displayed in Table (5.3). We concluded that:

1- Perturbations on the porosity and the permeability fields have a comparable influence, even if the permeability perturbations seem to be slightly more important.

2- Perturbations on the thickness field have a negligible effect compared to the other variables.

3- In the three cases, the probability distributions of the influences are lognormal. Figure (5.1) shows that the thickness distribution is much more correlated than the permeability and porosity distributions (smaller nugget effect). This might be the reason why the thickness perturbations do not play an important role. However,

this explains but does not limit the results to a particular case, since we can reasonably expect any thickness distribution to be more homogeneous than any permeability/porosity one. As a result, porosity and permeability were taken as the main variables in the influence function algorithm.

Variable	Number of Perturbations	$\bar{E}$	E_{max}	E_{min}	$var(E)$
Permeability	100	0.020	0.45	0.00	8.78
Porosity	100	0.011	0.11	0.00	2.81
Thickness	100	0.001	0.03	0.00	9.06

Table 5.3: Sensitivity Study

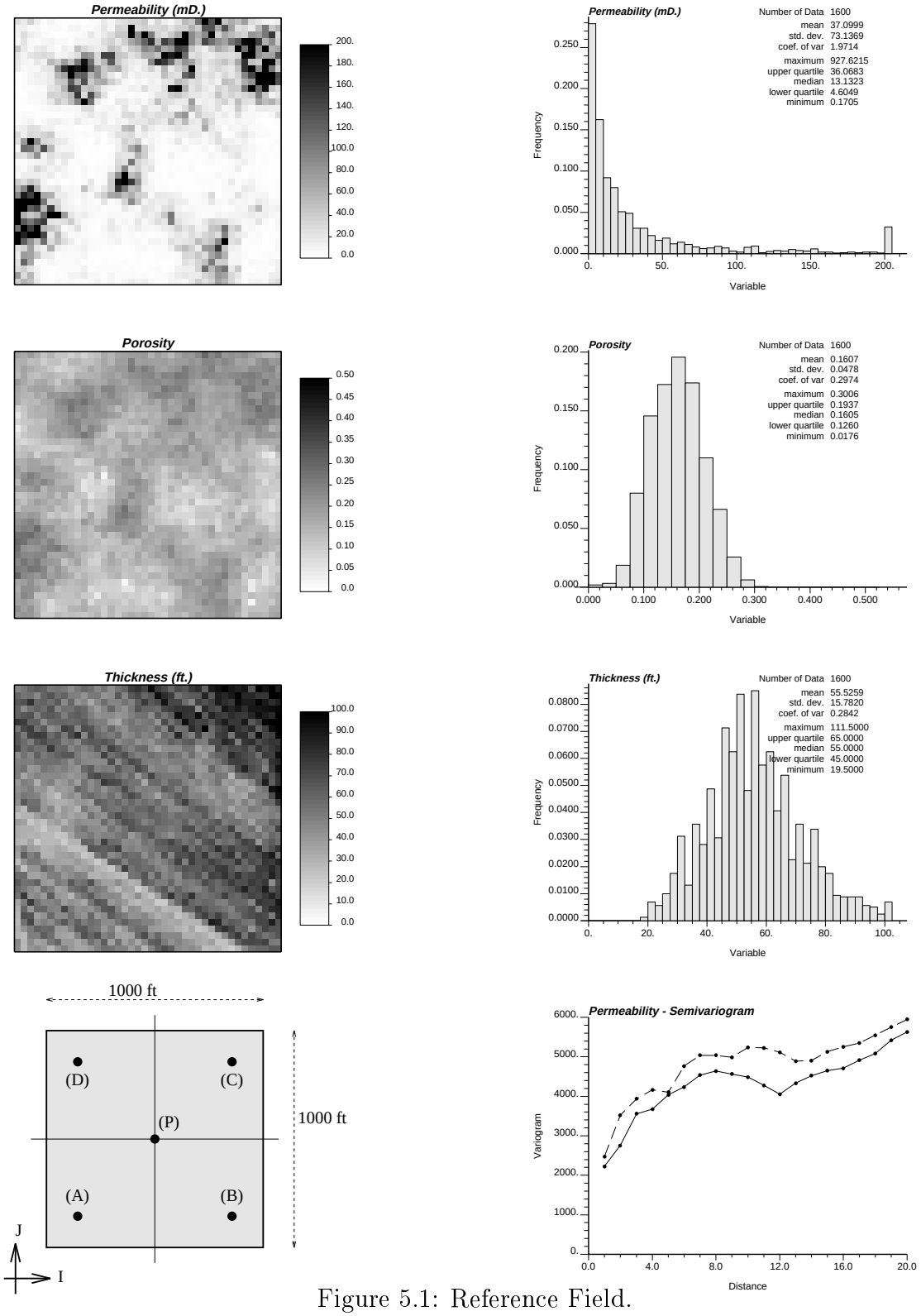


Figure 5.1: Reference Field.

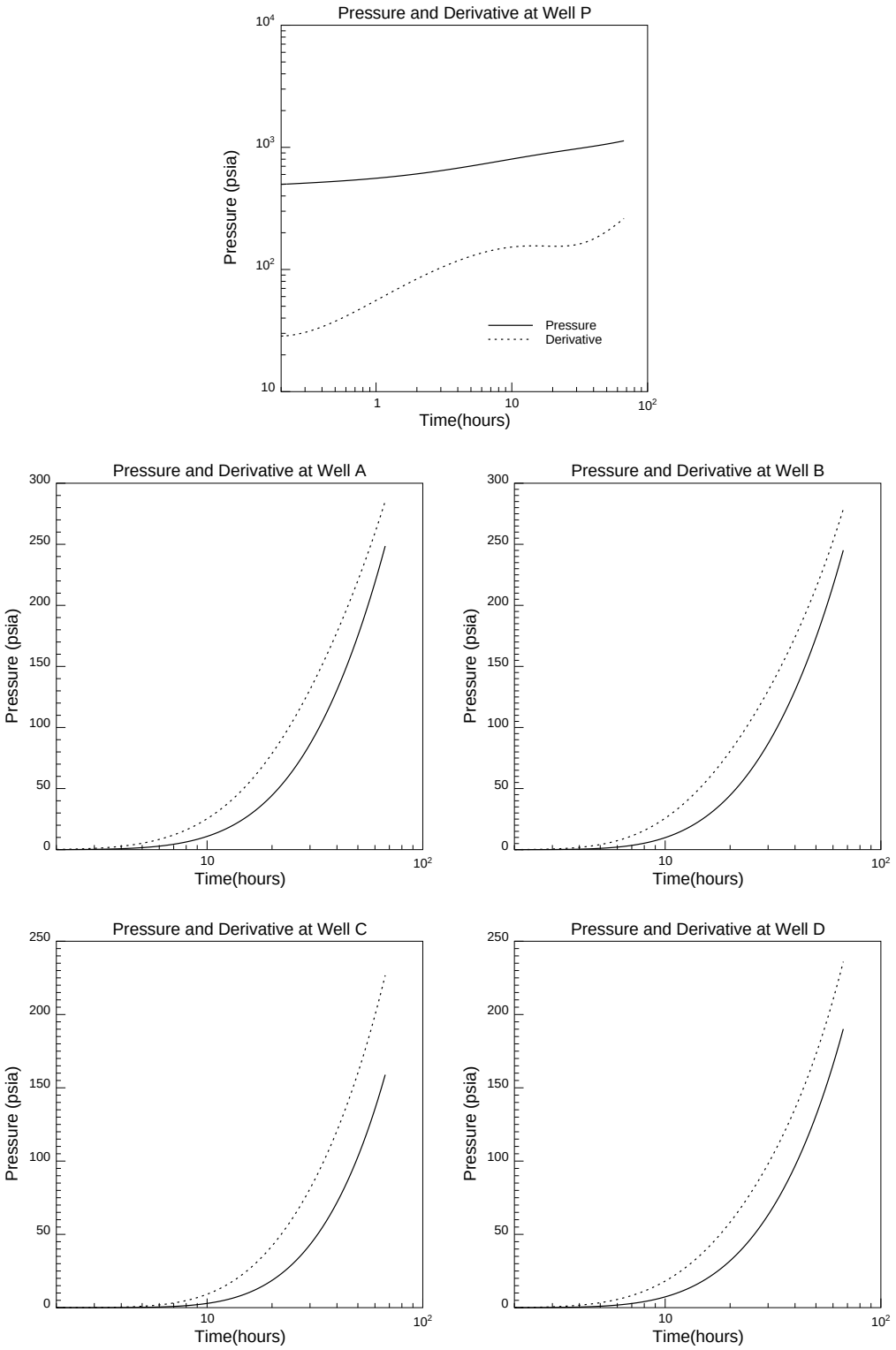


Figure 5.2: Reference Pressure Transients.

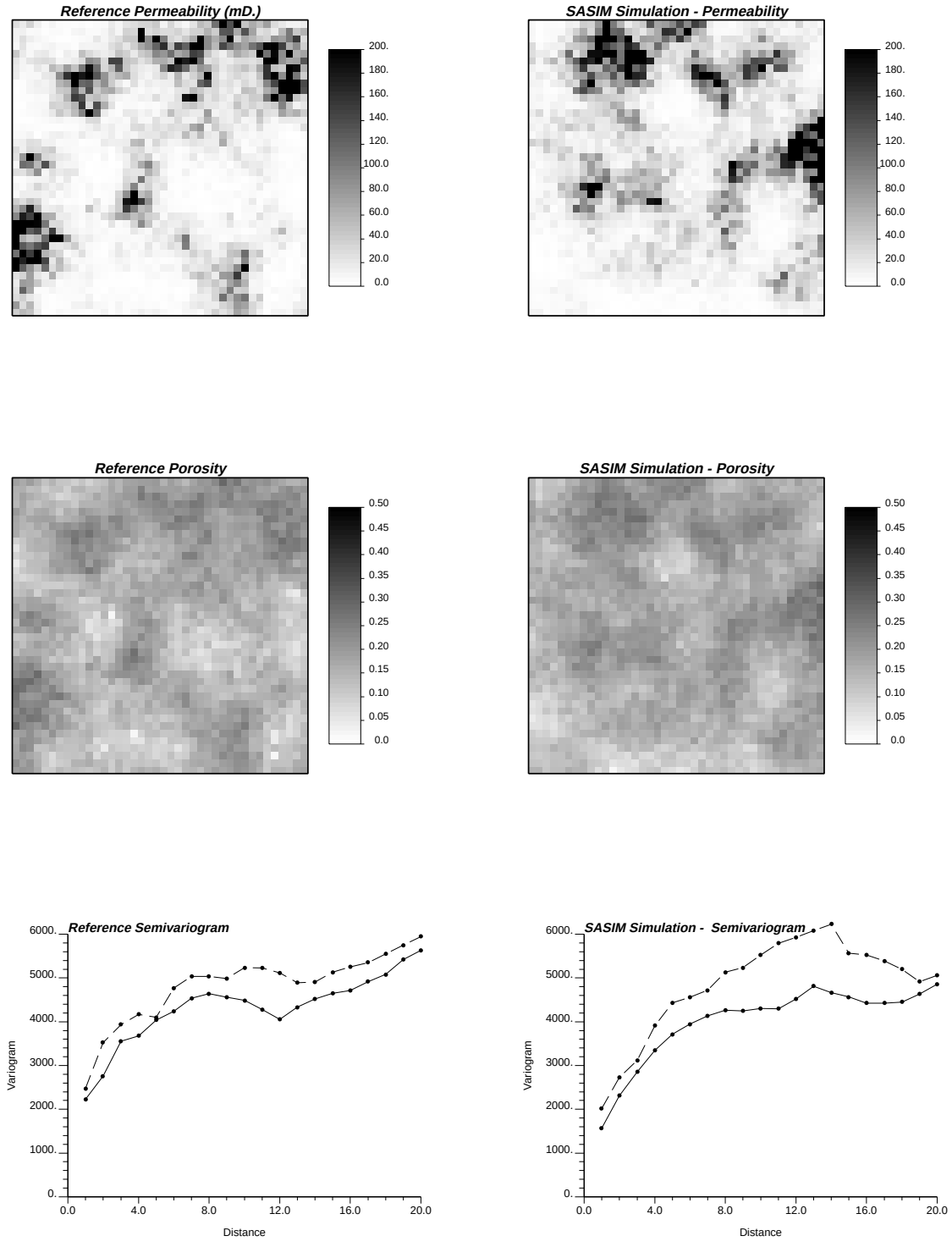


Figure 5.3: Example of Stochastic Simulation.

5.2 Postprocessing Stochastic Realizations

5.2.1 Description of the Runs

The 12 stochastic realizations generated using SASIM were constrained to the reference pressure data (Figure (5.2)). Table (5.4) shows the main parameters of the "worst" and the "best" runs, as well as the average characteristics of the runs. It was noticed that:

1. Using the influence function allows a considerable reduction of the execution time. An average run required the trial of $375 \times 35 = 13125$ perturbations. Since a simulator run required about 1 minute in this example, 13125 simulator runs would have lasted for more than 9 days. The use of the influence function permitted the same work in about 10 minutes.

2. In the objective function, the weight given to the derivative was found to be an important convergence parameter; this weight was often as big as 0.6 at well (P), and about 0.1 at the observation wells.

3. This important weight given to the derivative in the objective function has the consequence that the final value of O_f is rather high - always higher than 0.1. However the match obtained on the pressure transients was good in all cases, as shown on the Figures 5.4 (corresponding to the "best" match) and 5.5 ("worst" match).

	O_f	Execution Time	Simulator Runs	N_{max}	N_{smax}	N_{tot}
Worst Run	0.18	15min.45sec.	8	60	50	440
Best Run	0.10	6min.55sec.	7	20	50	360
Average	0.14	11min.12sec.	7	35	50	375

Table 5.4: Run Parameters

5.2.2 Post-Processed Grids

Figure (5.6) shows an example of the post-processed grid, as obtained after constraining to the well test data. A few conclusions can be drawn by looking at these grids:

1. The variograms are affected by the post-processing, but their main features remain the same. As a matter of fact, the most important modification is an increase of the nugget effect.

2. Even if they lead to a well test response much closer to the reference, the post-processed grids do not tend to "look" closer to the reference image than the initial stochastic realizations.

In order to have a better understanding of the modifications due to the Influence Function Algorithm, the connectivity[10] of the grids was studied, using a simple connectivity algorithm. Given a certain threshold value of permeability, a block in the grid is connected to the producing well (Well (P)) if and only if a path composed of values of permeability greater than the threshold can be found between the block and the well. Figure (5.7) shows the results obtained for a threshold of 10 md. The results show that the post-processing tends to recover a left-right asymmetry in the connectivity. Moreover the number of connected blocks changes from an average of 664 before post-processing to 457 after. This has to be compared with the number of connected blocks at 10 md in the reference image (459). However the results obtained depend strongly on the value of the threshold, and this value is taken arbitrarily.

5.2.3 Improving a Waterflood Forecast

A waterflood was simulated on the reference image, the 12 stochastic realizations and the 12 post-processed grids in order to determine whether the post-processing is able to improve a long-term production forecast. The waterflood was performed for a simulated time of 7000 days, Well (P) being the producer (controlled by pressure: 1000 psia), and the four wells (A), (B), (C), (D) being injectors (controlled by flow rate: 200 STB/day).

Figure (5.8) shows the results obtained in the "worst" case (small or no improvement before and after the post-processing), Figure (5.9) corresponds to the "best" case. Both figures show that the water cut is always close to the reference after post-processing. Table (5.5) shows how the errors on the pressure, liquid flow rate and water cut curves are modified. We define the error $E_{...}$ as: $E_{...} = \sum_{times} (data_{ref} - data_{...})^2$, the *data* being the pressure, liquid flow rate or water cut values. The most important results are shown by Figure (5.10) - on each histogram the values obtained for the reference grid are represented by a black dot. This figure demonstrates that post-processing leads to a prediction of the breakthrough times at 0.05 and 0.5 which is more accurate (smaller spread of the histograms), even if no improvement can be seen for 0.95. Even if this study was performed on only 12 cases, it shows that constraining on pressure data during a small period of time has the consequence of a better behavior of the stochastic realizations during long-term simulations - this result was also shown earlier by Deutsch[1].

	Pressures (A to D) E_{post}/E_{sasim}	Liq. Flow Rate (P) E_{post}/E_{sasim}	Water cut (P) E_{post}/E_{sasim}
Worst Forecast	2.36	0.96	0.09
Best Forecast	0.40	0.33	0.02
Average	0.47	0.45	0.05

Table 5.5: Waterflood Forecast

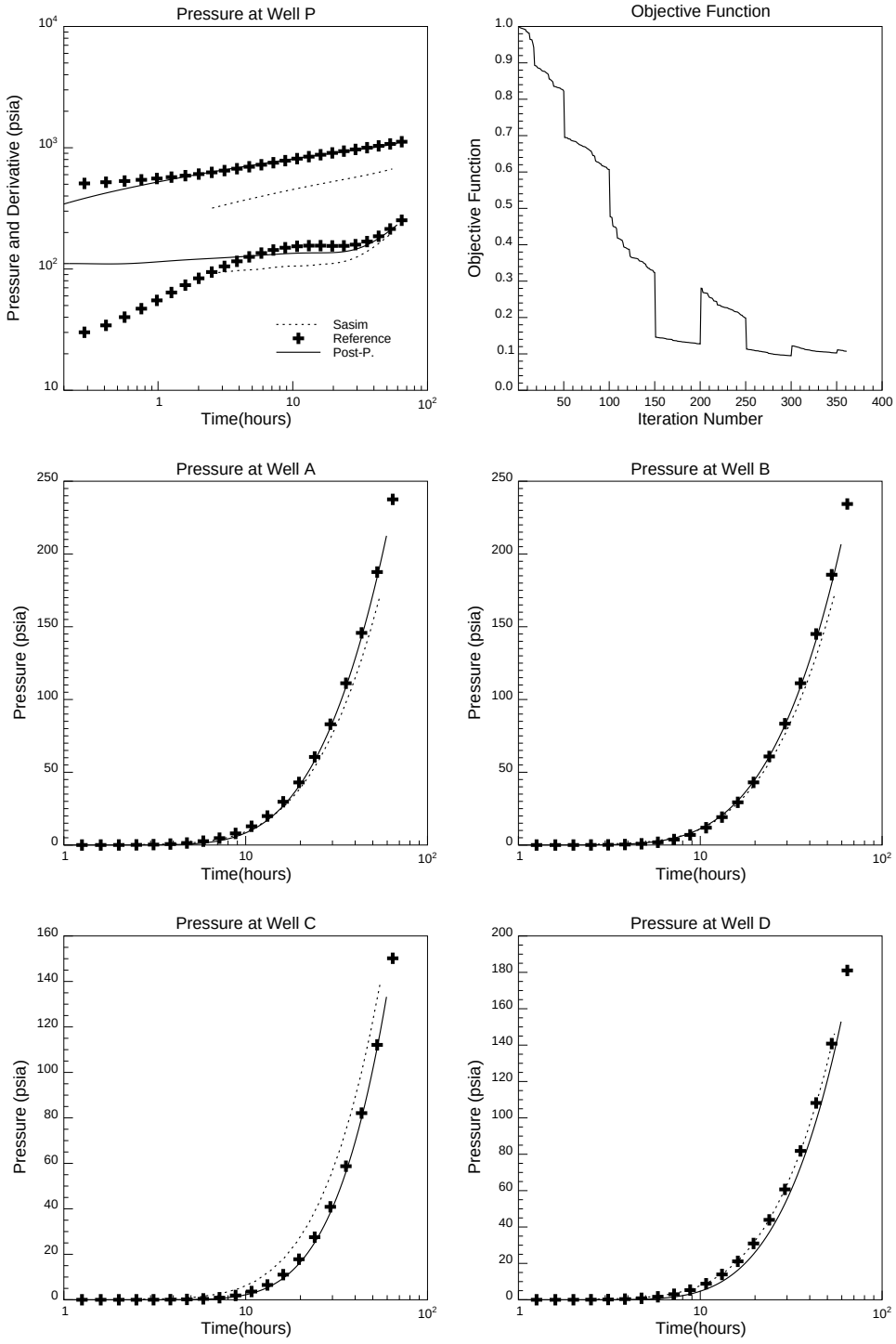


Figure 5.4: Constraining to the Pressure Data - Best Match

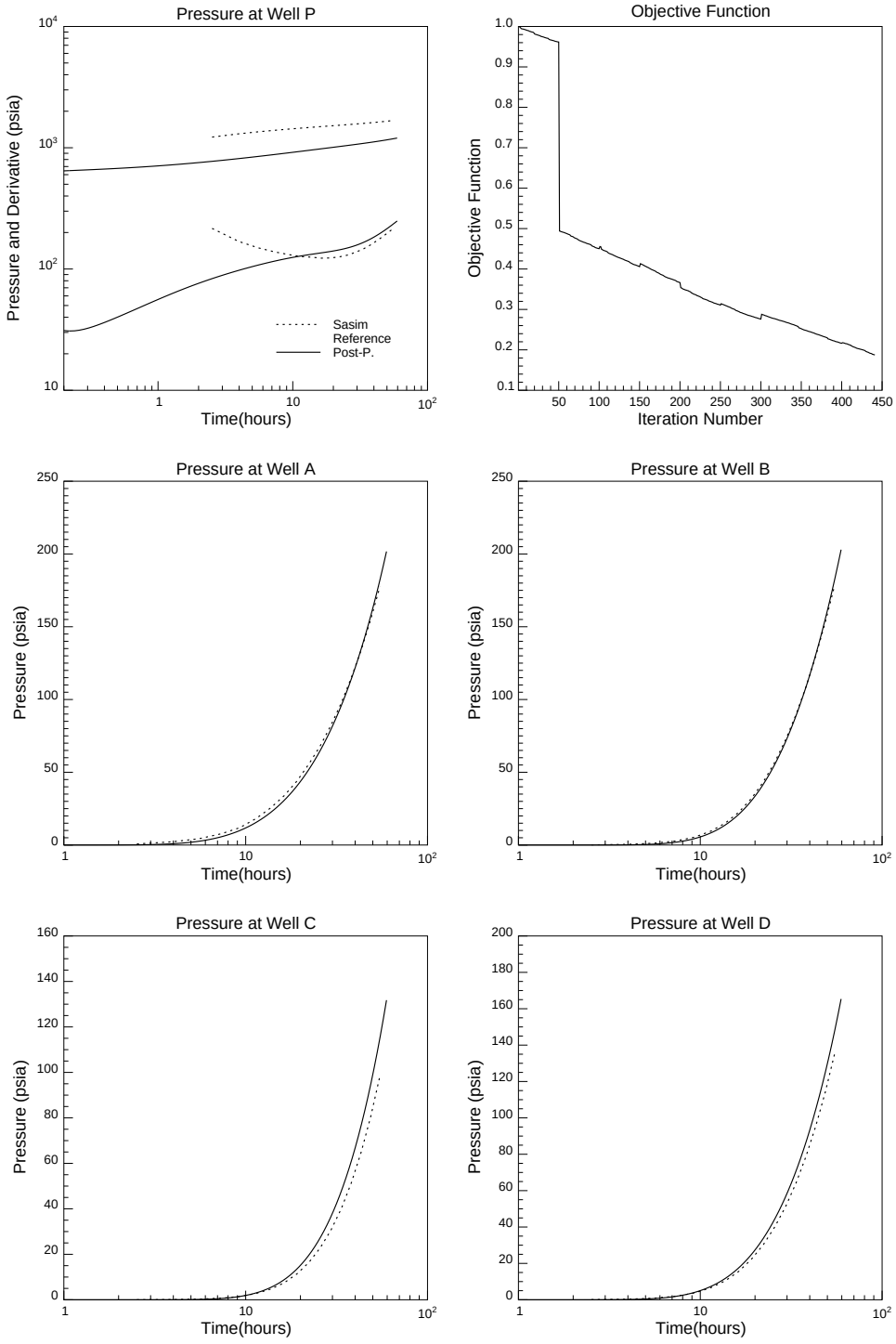


Figure 5.5: Constraining to the Pressure Data - Worst Match

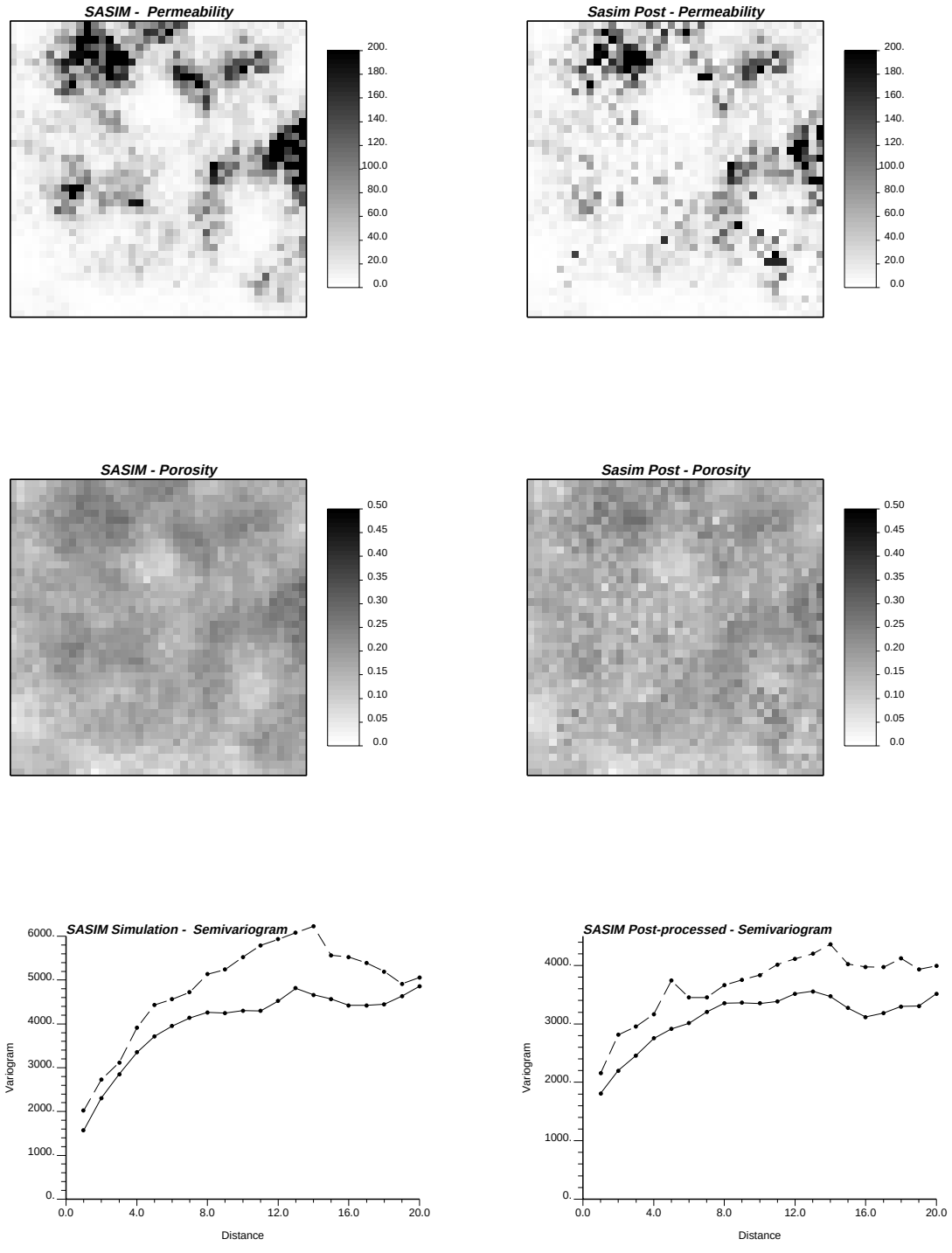


Figure 5.6: Example of Post-Processed grid

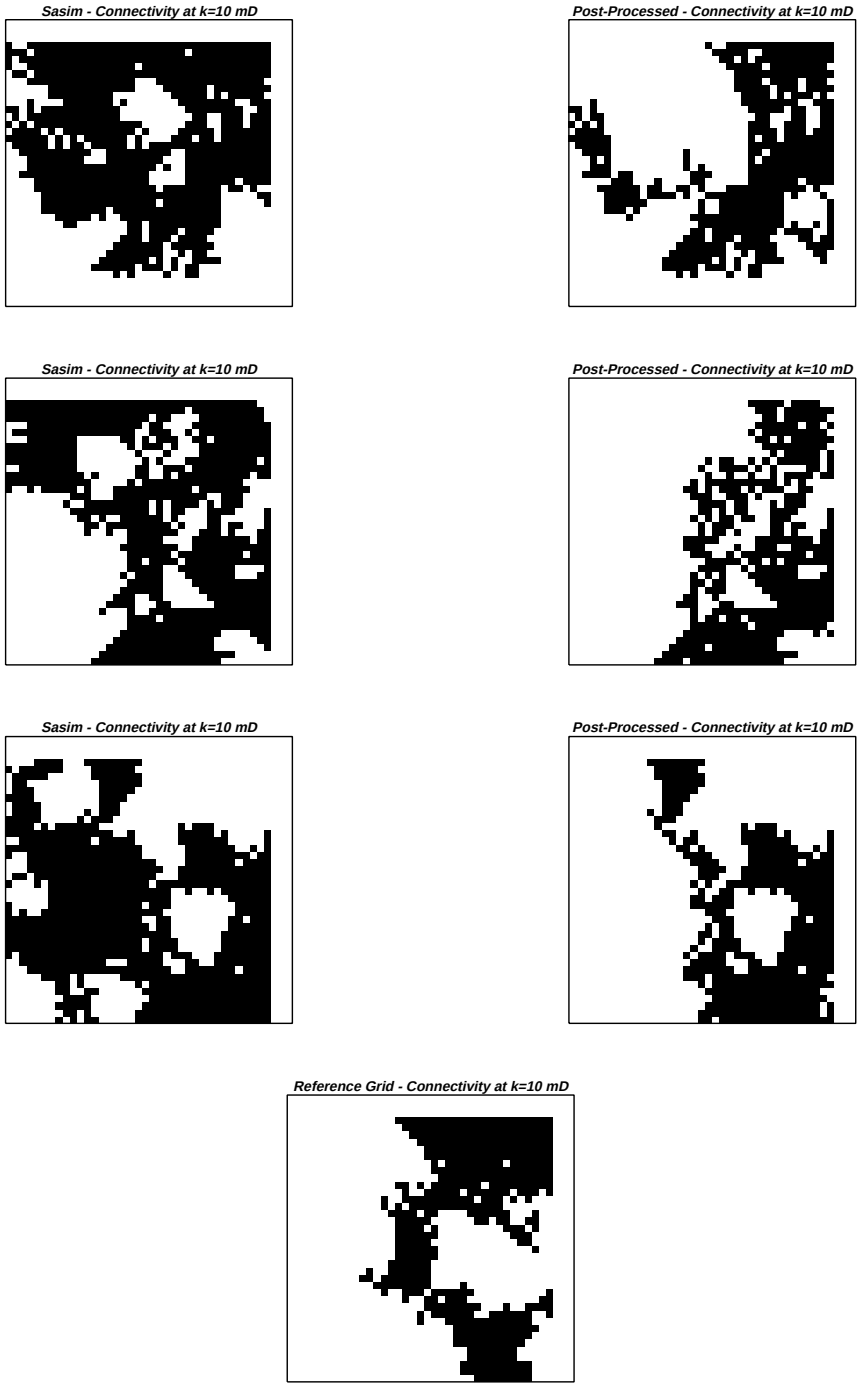


Figure 5.7: Connectivity at 10 md

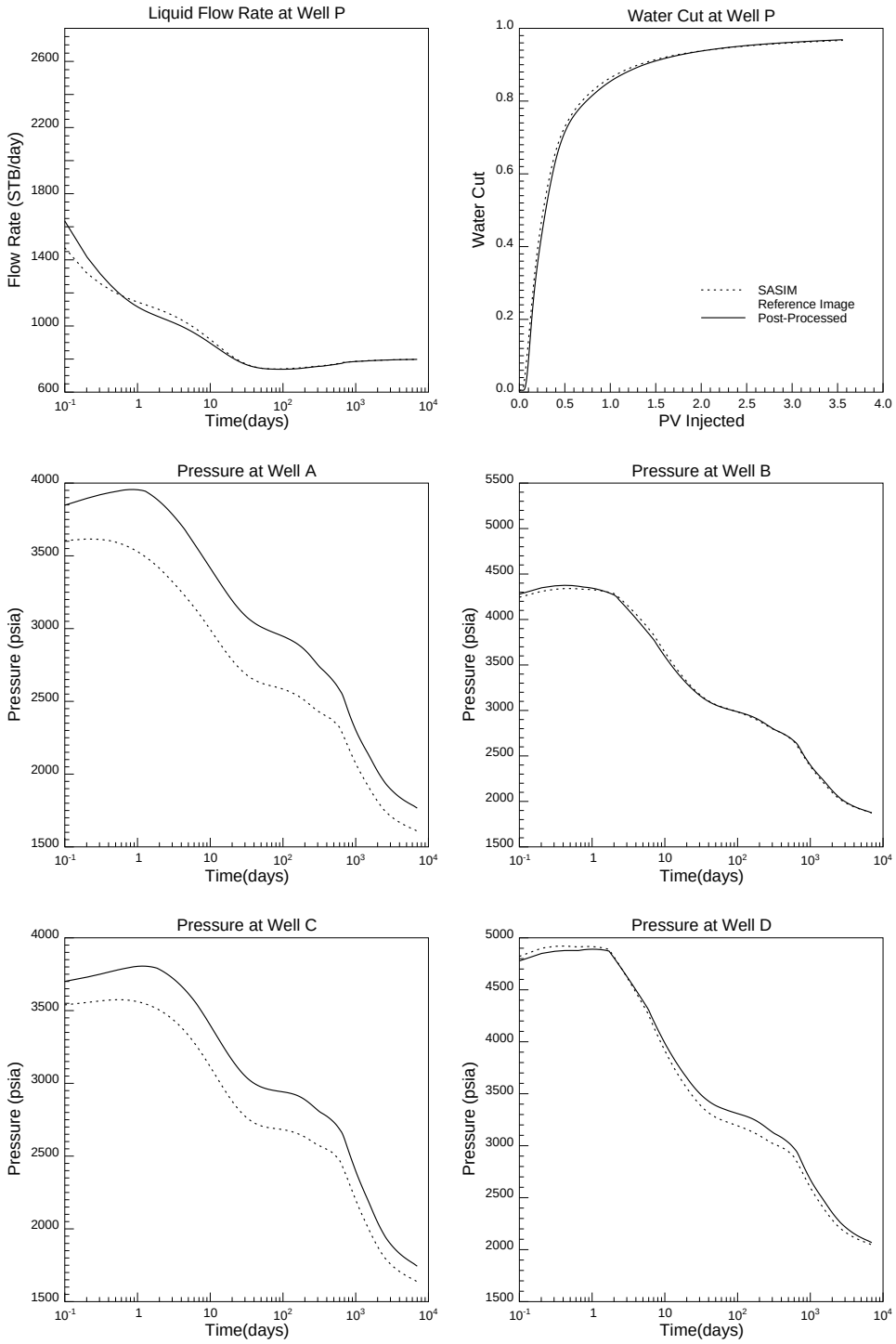


Figure 5.8: Waterflood Forecast - Worst Case

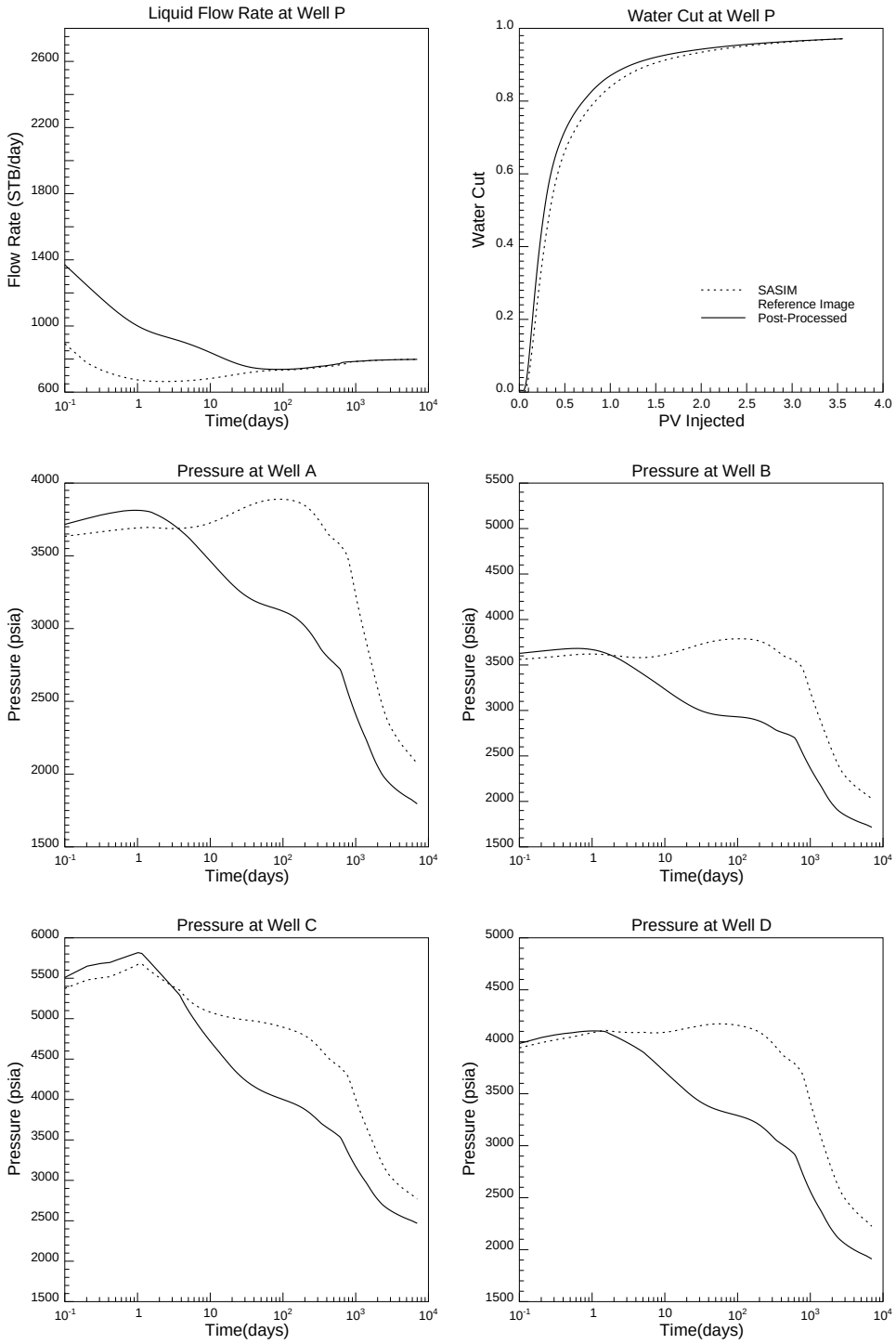


Figure 5.9: Waterflood Forecast - Best Case

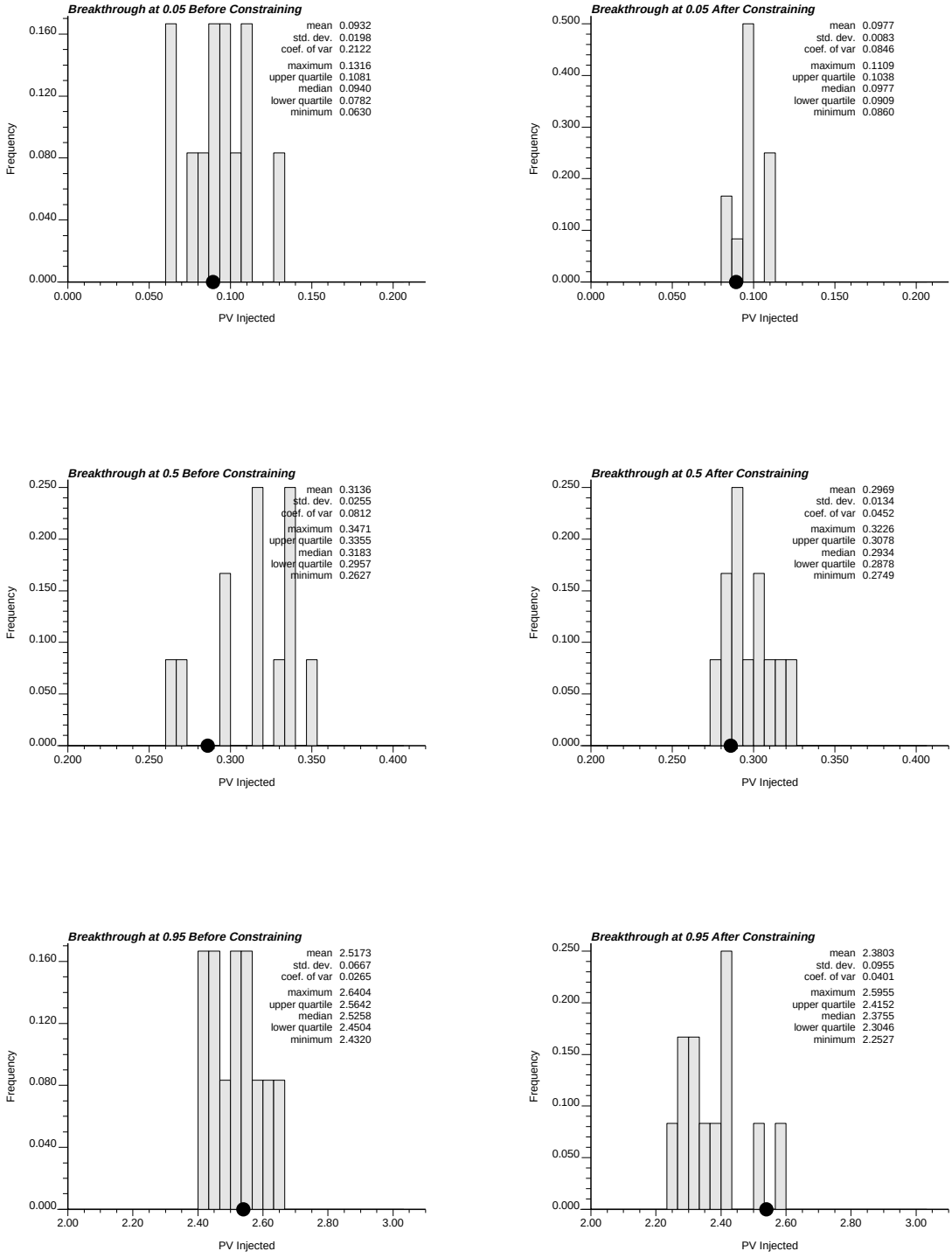


Figure 5.10: Waterflood Forecast - Breakthrough Times

Section 6

Concluding Remarks

We have constructed an algorithm which processes stochastic realizations in order to constrain them to the data from a single or multiple well test. This algorithm uses an analytic approximation to estimate the influence on the pressure transient of a given heterogeneity as a function of time and space. This approximation ignores the correlations between the influences of different heterogeneities, but permits a major improvement in the computational efficiency of the algorithm.

This algorithm has been applied to a simple example, from which three main remarks can be made:

1. The algorithm is able to make stochastic realizations replicate well test data recorded at several wells within a reasonable amount of computing time, and without significantly altering the prior spatial characteristics. It is important to notice that the initial stochastic realizations were not previously constrained to local well information. Such a preceding constraint should improve the results given by the post-processing.

2. Constraining to well test data over a short period of time tends to improve the forecast for a long-time production period such as a waterflood - this result was shown also by Deutsch[1].

3. The example showed that constraining to the pressure data does not lead to an improvement of the appearance of the image itself; the post-processed grids do not "look" closer to the reference image than the initial stochastic simulations. Indeed, this exercise shows that there is whole range of modeled reservoirs leading to a similar well test response at the five wells drilled in this example. The number of unknowns (1600 porosity + 1600 permeability values) is so much bigger than the number of constraining data (pressure data at five wells) that a different result would have been surprising. In fact, closing the problem would require many additional wells. However the point is that post-processing those grids to make them have the same well test signature tends to make them more accurate at forecasting production histories (at the same wells) as well.

Section 7

Nomenclature and Bibliography

Nomenclature

a	=	discontinuity radius, [L]
B	=	formation volume factor
c_t	=	compressibility, $[M^{-1}][L][T^2]$
f_H	=	influence function, $[M][L^{-1}][T^{-2}]$
h	=	reservoir thickness, [L]
I, J	=	grid block counters
I_n	=	modified Bessel function of the
		first kind and order n
k	=	permeability, $[L^2]$
K_n	=	modified Bessel function of the
	=	second kind and order n
N_{max}	=	maximum number of tried
		perturbations for each iteration
N_{smax}	=	number of accepted perturbations
		between two simulator runs
N_{tot}	=	total number of
		accepted perturbations

n	=	integer
O	=	objective function
p	=	pressure, [M][L^{-1}][T^{-2}]
$P[accept]$	=	probability
E	=	error
q	=	flow rate, [L^3][[T^{-1}]
r	=	radial distance, [L]
R	=	distance between two points, [L]
s	=	Laplace variable
t	=	time, [T]
T	=	temperature of the annealing schedule
x, y, z	=	space coordinates, [L]

Greek Symbols

Δ	=	(pressure) difference
ϵ	=	small number
ϕ	=	porosity
η	=	hydraulic diffusivity constant, [L^2][T^{-1}]
λ	=	mobility, [M^{-1}][L^3][T]
μ	=	viscosity, [M][L^{-1}][T^{-1}]
θ	=	cylindrical coordinate
ω	=	weight (in the Objective function)

Superscripts

$-$	=	Laplace transform
$\sim$	=	homogeneous (pressure)
$'$	=	derivation or position of a line source
i, j	=	counters

Subscripts

0	=	homogeneity
$data$	=	experimental data
D	=	dimensionless
H	=	heterogeneity
i	=	initial
f	=	final
n	=	counter or order of a Bessel function
new	=	tested configuration value
old	=	previous configuration value
sim	=	simulated data
ref	=	reference data
$post$	=	post-processed
$sasim$	=	sasim
t	=	time
w	=	well
∞	=	final value

Physical Units

$[L]$	=	length
$[M]$	=	mass
$[T]$	=	time

Bibliography

1. Deutsch, C.V.: "Annealing Techniques Applied to Reservoir Modeling and the Integration of Geological and Engineering (Well Test) Data", Ph.D. thesis, Stanford University, Stanford, California (1992).

2. Deutsch, C.V., and Cockerham, P.W.: "Practical Considerations in the Application of Simulated Annealing to Stochastic Simulation", Mathematical Geology, Vol.26, No.1 (1994).

3. Oliver, D.S.: "The Averaging Process in Permeability Estimation from Well Test Data", Paper 19845 presented at the 64th Annual Technical Conference of the SPE, San Antonio (Oct. 1989).

4. Oliver, D.S.: "Estimation of Radial Permeability Distribution from Well Test Data", Paper 20555 presented at the 65th Annual Technical Conference of the SPE, New Orleans (Sept. 1990).

5. Feitosa, G.S., Chu, L.F., Thompson, L.G., and Reynolds, A.C.: "Determination of Reservoir Permeability Distribution from Pressure Buildup Data", Paper 26457 presented at the 68th Annual Technical Conference of the SPE, Houston (Oct. 1993).

6. Rosa, A.J., and Horne, R.N.: "Pressure Transient Behavior in Reservoirs with an Internal Circular Discontinuity", Paper 26455 presented at the 68th Annual Technical Conference of the SPE, Houston (Oct. 1993).

7. Deng, X.F., and Horne, R.N.: "Well Test Analysis of Heterogeneous Reservoirs", Paper 26458 presented at the 68th Annual Technical Conference of the SPE, Houston (Oct. 1993).

8. Bourgeois, M.J., and Horne, R.N.: "Well Test Model Recognition Using Laplace Space Type Curves", Paper 22682 presented at the 66th Annual Technical Conference of the SPE, Dallas (Oct. 1991); *SPEFE* (March 1993) p. 17-25.

9. Deutsch, C.V., and Journel, A.G.: "GSLIB: Geostatistical Software Library and User's Guide", Oxford University Press, New York (1992).

10. Alabert, F.G., and Modot, V.: "Stochastic Models of Reservoir Heterogeneity: Impact on Connectivity and Average Permeabilities", Paper 24893 presented at the 67th Annual Technical Conference of the SPE, Washington, DC. (Oct. 1992)

Appendix A

Computer Programs

This appendix presents the code used to constrain the stochastic realizations to the well test data. This program uses the black oil simulator Eclipse (ECL) as a subroutine. It is composed by the following files:

1. *influ.help* : Help file explaining how the code works.
2. *influ.par* : Example of parameter file.
3. *influ.f* : Main fortran code.
4. *influ.inc* : Definition of variables.

We did not include the small script file *runeclipse* (file calling the black oil simulator Eclipse as a subroutine), since it is really short (4 lines) and easy to write.

A.1 *influ.help* - Help File

```
influ.f:
--------

This code post-processes stochastic realizations to constrain
them to well test
 data (pressure transients+derivative) recorded at several wells.
 The initial simulations must be in 2D, they can be heterogeneous
in permeability
 and/or porosity and/or thickness.
The constraining is performed on permeability and/or porosity.
The input grid files have to be in a GSLIB format.
 The output grid files are in a GSLIB format as well.
The input/output pressure data are in a free format.

The program uses the black oil simulator Eclipse as a subroutine.

Programs:        influ.f = main code
                 bessel.f = bessel functions
                 influ.inc = variable definitions
                 influ.par = parameter file
                 runeclipse = script file, transition with Eclipse.

to compile influ.f: f77 influ.f bessel.f -o influ
                       chmod 755 runeclipse

influ reads the parameter file influ.par.

influ.par:
----------

Parameter file for the code influ.

* PETROPHYSICS section:
```

This section defines the variables to be constant throughout the field.
These values will be used in case of dimensionless (&Laplace)
output pressures/derivatives.
(Note that the dimensionless derivative will NOT stabilize at 1/2
if random (and wrong) values are entered in this section.)

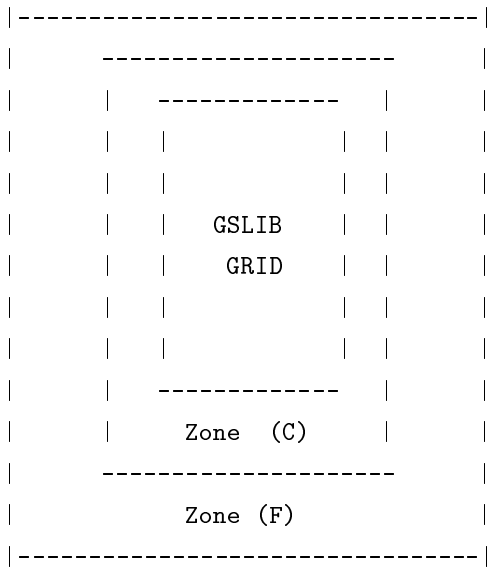
* HOMOGENEOUS VARIABLES Section:

This section defines the variables not to be kept constant in the
eclipse input grid.
the choice is: Porosity and/or permeability and/or thickness.
If a '1' is entered for any of these variables, the program will
ask for the corresponding GSLIB-format file.

* GRID SPECIFICATION Section:

The GSLIB grid is a square (2D).

It can be completed by an outer zone, in order to get rid of the boundaries
effects:



```
-Zone (F) can have a different (larger) grid blocks size.
-Zone (C) is a transition between the GSLIB-format grid and Zone (F).
(to avoid grid-size effects at the wells)

Both zones (C) and (F) are filled with the values of
porosity/permeability/t
hickness defined in the PETROPHYSICS section.

The GSLIB-GRID can be refined: The blocks can be cut into
smaller pieces. (Enter 3 will cut each block in this zone into
 9 smaller blocks.)

This section offers to upscale the input grid before to constrain
it to the pressure data.
The upscaling factor is the number of fine grid blocks to be
upscaled in a coarse grid block.
 Entering 1 for the factor means no upscaling.
 The geometric mean is taken for thickness/permeability upscaling,
the
 arithmetic mean is taken for the porosity.
 The program offers also not to upscale around the wells, but to keep
 the fine grid values.
The results obtained by post-processing an upscaled grid
were found to be rather poor, but this option is still in the program.

* WELL SPECIFICATION Section:

Enter I,J (GSLIB convention) for the location of each well in
the fine scale grid.
The programs asks also for rw (wellbore radius), q (flow rate), the weight
 w given to each well in the objective function and the weight wderiv given
 to the derivative at each well in the objective function.
 All these weights have to sum up to 1.
See Master's thesis (chapter 4) for more informations.

* TIME STEPS Section:
```

```
First time step  = tstart
t(1)=tstart
Second sime step = tstart * (1.+ tstep)
t(2)=t(1)+tstart * (1.+ tstep)
...
Ith time step    = tstat * (1 + tstep + ... + tstep**I)
t(i)=t(i-1)+tstat * (1 + tstep + ... + tstep**I)

Stops when t(i)>tend

* INPUT & OUTPUT FILES Section:

GSLIB format input - output files.

* ANNEALING SCHEDULE Section:

influ is some kind of mix between conventionnal simulated annealing
and the algorithm described in the thesis.
 As a results... I don't think the role of each parameter is really
easy to explain.
 BUT it works so... Good luck.

- Start of Algorithm: If set to 0, the initial grid is chosen to
 be homogeneous (with the values given in the petrophysics section).
 Otherwise, the starting grid is the input GSLIB grids read in the
previous section.

- Lambda(Ecl): Decrease rate of the frequency of Eclipse runs
(to update the objective function).
 If this frequency is x initially, an eclipse run will be performed
after x accepted perturbations, then x-Lambda, then x-2*Lambda...

- errmin: Value of the objective function at which the convergence
is accepted.
- itmax: maximum number of iterations before to stop the code.
 (each iteration consists in trying nloo*naccept_max swaps, in
```

```
order to keep naccept_max.)

- ntrymax: maximum number of tried swaps before to decrease the
annealing temperature.

- Lambda (t): maximum number of times when ntrymax has been reached.
 If Lambda(t) is reached, the code is stopped.

- Naccept_max: Maximum number of perturbations accepted before to
decrease the annealing temperature t.

- To: Initial annealing temperature.

- Eclo: Initial simulator run frequency.

- updating/no updating: If 0 is entered, the influence function is
not used to update the objective function.
 This objective function value changes only when a simulator run is
performed...

- Random seed:...

- nloo: Number of tried swaps before to accept the best of them.

- Number of summations in the series: cf Green functions.

- Value of epsilon: A swap is tried if and only if its
mobility/diffusivity ratio is higher than epsilon.
 This has been installed in order to avoid numerical problems.
```

A.2 *influ.par* - Parameter File

```
INFLU INPUT PARAMETERS
PETROPHYSICS:
#initial reservoir pressure(psia):
3500.
#mean reservoir permeability(mD) (homogeneous):
100
#reservoir thickness(ft) (homogeneous):
50.
#formation volume factor(reservoir volume/standard volume):
1.1
#viscosity(cP):
2.
#total compressibility(1/psia):
.000003
#porosity (homogeneous):
.2
HOMOGENEOUS VARIABLES:
#Homogeneous thickness -> 0 otherwise -> 1
1
#Homogeneous porosity -> 0 otherwise -> 1
1
#Homogeneous permeability -> 0 otherwise -> 1
1
CONSTRAINED VARIABLE:
#Porosity -> 0. Permeability -> 1. Both -> 2.
2.
GRID SPECIFICATION:
#number of nodes along x and y axis (fine grid):
40
#upscaling factor:
1
#conservation of wellbore skin (0/1=no/yes)
1
#length of a block(ft) (fine grid):
25.
#number of nodes along x and y axis of the zone (C) (coarse):
```

```
0
#number of nodes along x and y axis of the zone (F) (coarse):
0
#length of a block(ft) in the zone (F):
500.
#grid refinement in Eclipse runs (coarse):
1
WELL SPECIFICATION:
#number of wells:
5
#block locations of wells & Wellbore radius(ft) & flowrates(STB/D) & Weights:
#I J   rw  q      w  wderiv(fine grid coordinates)
6  6   .5  0.   .2    .1
35 6   .5  0.   .2    .1
35 35  .5  0.   .1    .1
6  35  .5  0.   .1    .1
18 18  .5  400. .4    .6
TIME STEPS
#First time step(days):
.05
#Time of simulation(days):
3.
#Increase rate between two time steps:
1.1
INPUT & OUTPUT FILES
#Input permeability grid (heterogeneous):
../results/influ/exem2/exem219e.out
#column number
1
#Reservoir thickness(ft) input file (heterogeneous):
../results/sto_simu/berea/bereaI.DAT
#column number
1
#Reservoir porosity input file (heterogeneous):
../results/influ/exem2/exem219o.out
#Column number
1
```

```
#Input pressure file:
../results/influ/exem2/exem2.ECL
#Number of time steps
139
# output files prefix (6 characters)
exem11
# output 0/1=Give/Don't
# True / Laplace pressures (TL)
0
# True / Real pressures (TR)
1
# Start / Laplace pressures (SL)
0
# Start / Real pressures (SR)
1
# Final / Laplace pressures (FL)
0
# Final / Real pressures (FR)
1
#ANNEALING SCHEDULE
#Start of algorithm (0=homo./1=heterogeneous)
1.
#Lambda(Ecl)
0
#errmin
.01
#itmax
20
#ntrymax
20
#Lambda(t)
.1
#Naccept_max
50
#To
0.
#Eclo
```

```
50
#1.=Updating/0.=No Updating
1.
#Random Seed
115335
#Sequence of Swaps tested nloo
30
#Number of summations in the series
1
#Value of Epsilon
.1
```

A.3 *influ.f* - Main Fortran Program

```
      intrinsic char
      include 'influ.inc'
      character *80 str
      character *6 ofile
      logical accept
      real lambde,affi(NBM)
      dimension ptest(NTM,NWM),dptest(NTM,NWM)

      rset=1.

* ask name of parameter file
      write(*,*) 'Enter name of parameter file'
      read(*,'(a20)') str(1:20)
      if(str(1:1).eq.' ') str(1:20) = 'influ.par        '

* load data
      call readata(str,ofile,Nc,tc,
     &pc,varperm,affi)

* initialize random numbers generators
      do 31 i=1,500
      u=acorn(xdum,rset)
 31   continue

* transforms input pressure data
      call treatin(N,Nc,tid,ti,tc,pc,pcl,dpcl)

* compute initial objective function
      call initob(N,ti,pcl,dpcl,pe,pel,dpel
     &,psim,dpsim,oinit)

* initialize Annealing parameters
      rset=1.
      obj=1.
      nswap=0
      nrunecl=0
```

```
      nperturb=0
      iend=0
      necl=0
      temp=sas(7)
      ntecl=sas(8)
      accept=.false.

* ANNEALING

      open(999,file=ofile//".OBJ",status="unknown")
      open(626,file=ofile//".LOC",status="unknown")

* loop until convergence or the stopping number
 1    naccept=0
      ntry=0

* keep trying perturbations until some limit is exceded

 2    ntry=ntry+1
      nswap=nswap+1

* pick a block to perturb

 950  objnew=obj
      ilmin=0

      do 900 il=1,nloo

 850  ibe=1+int(nxy*acorn(xdum,rset))
      jbe=1+int(nxy*acorn(xdum,rset))

      write(*,*) '-----------------------------------'
      write(*,*) 'block ',ibe,jbe

      if ((ibe.eq.19).and.(jbe.eq.19)) goto 850
* pick a value of perm from histogram
```

```
 800  ihe=1+int(nxy*acorn(xdum,rset))
      jhe=1+int(nxy*acorn(xdum,rset))

      if (varperm.eq.0.) then
      if (xporo(ibe,jbe).eq.xporoh(ihe,jhe)) goto 800
      lambde=1.
      etade=xporo(ibe,jbe)/xporoh(ihe,jhe)
      endif

      if (varperm.eq.1.) then
      if (xk(ibe,jbe).eq.xkh(ihe,jhe)) goto 800
      etade=xkh(ihe,jhe)/xk(ibe,jbe)
      lambde=etade
      endif

      if (varperm.eq.2.) then
      if ((xkh(ihe,jhe).eq.xk(ibe,jbe)).and.
     &(xporo(ibe,jbe).eq.xporoh(ihe,jhe))) goto 800
      lambde=xkh(ihe,jhe)/xk(ibe,jbe)
      etade=xkh(ihe,jhe)*xporo(ibe,jbe)/
     &(xk(ibe,jbe)*xporoh(ihe,jhe))
      endif

      if ((etade.lt.epstad).or.(lambde.lt.epstad)) goto 800

* compute new objective function

      call object(ibe,jbe,etade,lambde,objne,
     &ptest,dptest,N,oinit,
     &ti,psim,dpsim,pcl,dpcl,varperm)

      write(*,*)
      write(*,*) 'Try ',il,' -> obj=',obj,' ... obj-tried=',objne
      write(*,*)

      if (objne.lt.objnew) then
      ilmin=il
```

```
      objnew=objne
      ib=ibe
      jb=jbe
      ih=ihe
      jh=jhe
      etad=etade
      lambd=lambde
      endif

 900  continue
      if (il.eq.0) goto 950

      write(626,*) ib,jb,' 1'

      call object(ib,jb,etad,lambd,objnew,
     &ptest,dptest,N,oinit,
     &ti,psim,dpsim,pcl,dpcl,varperm)

* test to keep or reject perturbation

      accept=.false.
      if (objnew.gt.obj) then
      if (objnew.lt.(obj-temp*log(acorn(xdum,rset))))
     + accept=.true.
      else
      accept=.true.
      endif

* if the perturbation is kept then update grid

      if (accept) then

* update grid
      if (varperm.eq.0.) then
      xporo(ib,jb)=xporoh(ih,jh)
      endif
      if (varperm.eq.1.) then
```

```
      xk(ib,jb)=xkh(ih,jh)
      endif
      if (varperm.eq.2.) then
      xporo(ib,jb)=xporoh(ih,jh)
      xk(ib,jb)=xkh(ih,jh)
      endif

* update counters
      nperturb=nperturb+1
      naccept=naccept+1
      necl=necl+1
* update objective func.+ pressures if required
      if (Hupdate) then
      obj=objnew
      do 5 j=1,nw
      do 5 i=1,N
      psim(i,j)=ptest(i,j)
      dpsim(i,j)=dptest(i,j)
 5    continue
      endif
      write(999,*) nswap,nperturb,nrunecl,obj
      endif

* time to perform an Eclipse run??

      if (necl.ge.ntecl) then
      necl=0
      nrunecl=nrunecl+1
      call update(N,psim,dpsim,pcl,dpcl,obj,oinit)
      endif

* converge to a solution?

      if (obj.lt.sas(2).or.iend.ge.sas(3)) then
      goto 100
      endif
```

```
* too many tried at this temperature?

      if (ntry.gt.sas(4)) then
      iend=iend+1
      temp=sas(5)*temp
      ntecl=ntecl-sas(1)
      if (ntecl.lt.1) ntecl=1
      goto 1
      endif

* accepted enough at this temperature?

      if (naccept.ge.sas(6)) then
      iend=0
      temp=sas(5)*temp
      ntecl=ntecl-sas(1)
      if (ntecl.lt.1) ntecl=1
      goto 1
      endif

* go back for another perturbation

      goto 2

* CONVERGENCE ACHIEVED
* Write ouput files and exit

 100  close(999)
      close(626)
      call outprog(N,Nc,ofile,nswap,obj,
     &nperturb,tid,ti,pcl,tc,pc,pel
     &,nrunecl,pe,affi)

      stop
      end

*-------------------------------------------------
```

```
************************************************
*-----------------------------------------------

* INFLU.F SUBROUTINES:

* readata = input data lecture
* treatin = input pressures handling
* writecl = Eclipse input file construction
* outpres = Eclipse output pressures handling
* expin   = exponential function
* flapl6  = takes Laplace transform
* deriv   = takes derivative
* adim    = put into dimensionless form
* acorn   = random number generator
* object  = objective function updating
* place   = compute parameters for influence function
* so      = external influence function
* si      = internal influence function
* so0     = line source
* outprog = write output files
* update  = runs Eclipse to update solution
* BBSLK   = Bessel function of order n
* BBSLI   = Bessel function of order n

*-----------------------------------------------
************************************************
*-----------------------------------------------


      subroutine readata(str,ofile,Nc,tc,
     &pc,varperm,affi)

* read data main data from parameter file
* read perm. data from GSLIB format file
* read pressure data from following format file:
* t(i), pc(i,1), ..., pc(i,nw) for i time steps
```

```
      include 'influ.inc'
      character *80 str,permI,pressI,poroI,thickI
      character *6 ofile
      logical Skin,Hderiv,Hstart
      real affi(NBM)
      real tmp(NBM)

      open(1,file=str(1:20),status="unknown")
      read(1,'()')
* ---Petrophysics
      read(1,'()')
      read(1,'()')
      read(1,*) presi
      read(1,'()')
      read(1,*) bark
      read(1,'()')
      read(1,*) barh
      read(1,'()')
      read(1,*) B
      read(1,'()')
      read(1,*) vis
      read(1,'()')
      read(1,*) ctot
      read(1,'()')
      read(1,*) barporo
      read(1,'()')
* ---Homogeneous variables
      read(1,'()')
      read(1,*) xhomo
      if (xhomo.eq.0) Hthick=.true.
      read(1,'()')
      read(1,*) xhomo
      if (xhomo.eq.0) Hporo=.true.
      read(1,'()')
      read(1,*) xhomo
      if (xhomo.eq.0) Hperm=.true.
```

```
* ---Chosen Variable
      read(1,'()')
      read(1,'()')
      read(1,*) varperm
* ---Grid specification
      read(1,'()')
      read(1,'()')
      read(1,*) nxyf
      read(1,'()')
      read(1,*) nups
      read(1,'()')
      Skin=.false.
      read(1,*) alors
      if (alors.eq.1.) Skin=.true.
      read(1,'()')
      read(1,*) a
      read(1,'()')
      read(1,*) nclo
      read(1,'()')
      read(1,*) nfar
      read(1,'()')
      read(1,*) afar
      read(1,'()')
      read(1,*) nsub
* ---Well specification
      read(1,'()')
      read(1,'()')
      read(1,*) nw
      read(1,'()')
      read(1,'()')
      qmax=0.
      do 2 i=1,nw
      read(1,*) nxw(i),nyw(i),rw(i),q(i),wel(i)
     &,wderiv(i)
      nxw(i)=nxw(i)/nups+1
      nyw(i)=nyw(i)/nups+1
      if (q(i).ge.qmax) then
```

```
      qmax=q(i)
      rwmax=rw(i)
      nxwmax=nxw(i)
      nywmax=nyw(i)
      endif
 2    continue
* ---Simulation Time Steps
      read(1,'()')
      read(1,'()')
      read(1,*) tstart
      read(1,'()')
      read(1,*) tend
      read(1,'()')
      read(1,*) tstep
* ---Input Files Names
      read(1,'()')
      read(1,'()')
      read(1,'(a)') permI
      read(1,'()')
      read(1,*) iperm
      read(1,'()')
      read(1,'(a)') thickI
      read(1,'()')
      read(1,*) ithick
      read(1,'()')
      read(1,'(a)') poroI
      read(1,'()')
      read(1,*) iporo
      read(1,'()')
      read(1,'(a)') pressI
      read(1,'()')
      read(1,*) Nc
* ---Output files prefix
      read(1,'()')
      read(1,'(a)') ofile
      read(1,'()')
      do 80 i=1,6
```

```
      read(1,'()')
      read(1,*) affi(i)
 80   continue
* ---Annealing Schedule
* ---Start of algorithm
      read(1,'()')
      read(1,'()')
      Hstart=.true.
      read(1,*) alors
      if (alors.eq.1.) Hstart=.false.
      do 3 i=1,8
      read(1,'()')
      read(1,*) sas(i)
 3    continue
      read(1,'()')
      read(1,*) alors
      Hupdate=.false.
      if (alors.eq.1.) Hupdate=.true.
      read(1,'()')
      read(1,*) xdum
      read(1,'()')
      read(1,*) nloo
      read(1,'()')
      read(1,*) m
      read(1,'()')
      read(1,*) epstad
      close(1)

      do 290 i=1,nxyf
      do 290 j=1,nxyf
      xporof(i,j)=barporo
      xkf(i,j)=bark
      xthickf(i,j)=barh
 290  continue

      if (Hperm) goto 20
* input permeability file (GSLIB format)
```

```
      open(4,file=permI,status="unknown")
      read(4,'()')
      read(4,*) ivar
      do 5 i=1,ivar
      read(4,'()')
 5    continue
      do 6 j=1,nxyf
      do 6 i=1,nxyf
      read(4,*) (tmp(k),k=1,ivar)
      xkf(i,j)=tmp(iperm)
 6    continue
      close(4)

 20   if (Hthick) goto 30
* input thickness grid
      open(8,file=thickI,status="unknown")
      read(8,'()')
      read(8,*) ivar
      do 9 i=1,ivar
      read(8,'()')
 9    continue
      do 10 j=1,nxyf
      do 10 i=1,nxyf
      read(8,*) (tmp(k),k=1,ivar)
      xthickf(i,j)=tmp(ithick)
 10   continue
      close(8)

 30   if (Hporo) goto 40
* input porosity file (GSLIB format)
      open(11,file=poroI,status="unknown")
      read(11,'()')
      read(11,*) ivar
      do 51 i=1,ivar
      read(11,'()')
 51   continue
      do 61 j=1,nxyf
```

```
      do 61 i=1,nxyf
      read(11,*) (tmp(k),k=1,ivar)
      xporof(i,j)=tmp(iporo)
 61   continue
      close(11)

* input pressure grid (Eclipse format)
 40   Hderiv=.true.
      treatpres=0.
      call outpres(pressI,treatpres,Hderiv,tc,pc,dpc,Nc)

* upscaling
      a=a*float(nups)
      nxy=nxyf/nups

      do 50 j=1,nxy
      do 50 i=1,nxy

      sumk=1.
      sumporo=0.
      sumthick=1.
      do 60 ki=1,nups
      do 60 kj=1,nups
      sumporo=sumporo+xporof((i-1)*nups+ki,(j-1)*nups+kj)
      sumthick=sumthick*xthickf((i-1)*nups+ki,(j-1)*nups+kj)
     &**(1./float(nups**2))
      sumk=sumk*xkf((i-1)*nups+ki,(j-1)*nups+kj)
     &**(1./float(nups**2))
 60   continue

      xporoh(i,j)=sumporo/(float(nups**2))
      xthick(i,j)=sumthick
      xkh(i,j)=sumk

 50   continue

      if (Skin) then
```

```
      xporoh(nxwmax,nywmax)=xporof(nxwmax*nups,nywmax*nups)
      xkh(nxwmax,nywmax)=xkf(nxwmax*nups,nywmax*nups)
      xthick(nxwmax,nywmax)=xthickf(nxwmax*nups,nywmax*nups)
      endif

* initialize upscaled grid

      do 70 j=1,nxy
      do 70 i=1,nxy

      if (varperm.eq.0.) then
      if (Hstart) then
      xporo(i,j)=barporo
      else
      xporo(i,j)=xporoh(i,j)
      endif
      xk(i,j)=xkh(i,j)
      endif

      if (varperm.eq.1.) then
      if (Hstart) then
      xk(i,j)=bark
      else
      xk(i,j)=xkh(i,j)
      endif
      xporo(i,j)=xporoh(i,j)
      endif

      if (varperm.eq.2.) then
      if (Hstart) then
      xporo(i,j)=barporo
      xk(i,j)=bark
      else
      xporo(i,j)=xporoh(i,j)
      xk(i,j)=xkh(i,j)
      endif
      endif
```

```
 70   continue

* dimensionless radius of disc (influence function)
      ad=a/(2.*rwmax)
* well coordinates [Producer at (0,0)]
      do 7 i=1,nw
      xw(i)=(nxw(i)-nxwmax)*2.*ad
      yw(i)=(nyw(i)-nywmax)*2.*ad
 7    continue


      return
      end

*--------------------------------------------------
**************************************************
*--------------------------------------------------

      subroutine treatin(N,Nc,tid,ti,tc,pc,pcl,dpcl)

* computes ti(i,j),pci(i,j), for i=1,N;j=1,nw
* takes Laplace transform of pci(i,j)

      include 'influ.inc'
      dimension tloc(NTM),pci(NTM,NWM),ploc(NTM)

* computes the N simulated times ti(i)
      do 5 j=1,nw
      numeun=1
      ti(numeun,j)=tstart
      deltat=tstart
      i=1
 1    i=i+1
      deltat=deltat*tstep
      ti(i,j)=ti(i-1,j)+deltat
      if (ti(i,j).lt.tend) then
```

```
      goto 1
      endif
 5    continue

      N=i-2

* transforms times into hours
      do 6 i=1,N
      do 6 j=1,nw
      ti(i,j)=ti(i,j)*24.
      tid(i,j)=ti(i,j)
 6    continue

* interpolate well test pressures
      do 2 j=1,nw
      do 2 i=1,N
      do 2 k=1,Nc
      if ((ti(i,j).ge.tc(k,j)).and.(ti(i,j).
     &le.tc(k+1,j))) then
      pci(i,j)=pc(k,j)+(pc(k+1,j)-pc(k,j))*
     &(ti(i,j)-tc(k,j))/(tc(k+1,j)-tc(k,j))
      endif
 2    continue

* dimensionless variables

      call adim(N,ti,pci)

* takes Laplace transform pcl of pci
      pi=0.
      do 3 j=1,nw
      do 4 i=1,N
      tloc(i)=ti(i,j)
      ploc(i)=pci(i,j)
 4    continue
      do 3 i=1,N
      z=1./tloc(i)
```

```
      pcl(i,j)=flapl6(z,ploc,tloc,N,pi)
      pcl(i,j)=z*pcl(i,j)
 3    continue

* takes derivative of Laplace transform
      call deriv(N,ti,pcl,dpcl)

      return
      end

*--------------------------------------------------
**************************************************
*--------------------------------------------------

      subroutine initob(N,ti,pcl,dpcl,pe,pel,dpel
     &,psim,dpsim,oinit)

      include 'influ.inc'
      logical Hderiv
      character *40 Eclfile
      dimension loca(NTM,NWM)

      Hderiv=.false.
      nante=6

* write Eclipse input file
      call writecl
* run Eclipse
      call calsimu
* read output pressures & Laplace t.
      Eclfile="ECLI.A0001                              "
      treatpres=0.
      call outpres(Eclfile,treatpres,Hderiv,
     &ti,pe,loca,N)
      Eclfile="ECLI.A0001                              "
      treatpres=2.
      call outpres(Eclfile,treatpres,Hderiv,
```

```
     &ti,pel,loca,N)
      call deriv(N,ti,pel,dpel)

* initialize simulated pressures psim
      do 1 j=1,nw
      do 1 i=1,N
      psim(i,j)=pel(i,j)
      dpsim(i,j)=dpel(i,j)
 1    continue

* compute initial objective function
      oinit=0.
      do 2 j=1,nw
      x=0.
      y=0.
      do 3 i=1,N
      x=x+abs(pcl(i,j)-psim(i,j))/pcl(N,j)
      y=y+abs(dpcl(i,j)-dpsim(i,j))/dpcl(N-nante,j)
 3    continue
      oinit=oinit+(1.-wderiv(j))*wel(j)*x
      oinit=oinit+wderiv(j)*wel(j)*y
 2    continue

      return
      end

*--------------------------------------------------
**************************************************
*--------------------------------------------------

      subroutine writecl

* WRITES ECLIPSE INPUT FILE (.DATA)

      include 'influ.inc'
      dimension nxwe(NWM),nywe(NWM)
```

```
* LOCAL DATA:
* dwater = Water density (lb/ft3)
* doil = Oil density (lb/ft3)
* dgas = Gas density (lb/ft3)
* datum=datum depth
* woc=water aquifer depth
* goc=gas cap depth
* pmax=maximum admissible pressure

      parameter(dwater=63.02,doil=58.6928,dgas=5.34e-2,datum=
     &8000.,woc=10000,goc=6000,pmax=7000)

* total number of nodes
      NtotL=nxy*nsub+(nfar+nclo)*2
      asub=a/float(nsub)

* coordinates of wells in new grid
      do 2 i=1,nw
      nxwe(i)=nsub*(nxw(i)-1)+int(nsub/2.)+1+nfar+nclo
      nywe(i)=nsub*(nyw(i)-1)+int(nsub/2.)+1+nfar+nclo
 2    continue

      open(1,file="ECLI.DATA",status="unknown")
      write(1,100) 'RUNSPEC','FIELD   00:00 06 APR 93'
      write(1,101) '3D2P  CELLS'
      write(1,102) 'NDIVIX','NDIVIY','NDIVIZ','QRDIAL','NUMRES',
     &'QNNCON','MXNAQN','MXNAQC','QDPORO','QDPERM'
      write(1,*) NtotL,NtotL,1,'  /'
      write(1,103) 'OIL','GAS','DISGAS','VAPOIL','QAPITR','QWATTR'
     &,'QGASTR','NOTRAC','NWTRAC','NGTRAC'
      write(1,*) '  T   F   F     F      F   /'
      write(1,104) 'UNIT CONVENTION'
      write(1,*) '   FIELD /'
      write(1,105) 'NRPVT','NPPVT','NTPVT','NTROCC','QROCKC'
     &,'QRCREV'
      write(1,*) '   /'
      write(1,106) 'NSSFUN','NTSFUN','QDIRKR','QREVKR','QVEOPT'
```

```
&,'QHYST','QSCAL','QSDIR','QSCREV'
 write(1,*) '   /'
 write(1,107) 'NDRXVD','NTEQUL','NDPRVD','QUIESC','QTHPRS'
&,'QREVTH','QMOBIL'
 write(1,*) '   /'
 write(1,104) 'NTFIP QGRAID  QPAIR'
 write(1,*) '   /'
 write(1,104) 'NWMAXZ NCWMAX NGMAXZ NWGMAX'
 write(1,*) nw,' 0 1 ',nw,'   /'
 write(1,104) 'QIMCOL NWFRIC NUPCOL'
 write(1,*) '   /'
 write(1,108) 'MXMFLO','MXMTHP','MXMWFR','MXMGFR','MXMALQ'
&,'NMMVFT'
 write(1,*) '   /'
 write(1,107) 'MXSFLO','MXSTHP','NMSVFT','MXCFLO','MXCWOC'
&,'MXCGOC','NCRTAB'
 write(1,*) '   /'
 write(1,104) 'NAQFET NCAMAX'
 write(1,*) '   /'
 write(1,104) '  DAY   MONTH  YEAR'
 write(1,*)     '1    ''JAN''  0000 /'
 write(1,108) 'QSOLVE','NSTACK','QFMTOU','QFMTIN','QUNOUT'
&,'QUNINP'
 write(1,*) ' T 10 T F F F /'
 write(1,*)
 write(1,109) 'GRID    ==============================='
 write(1,*)
 write(1,109) 'PSEUDOS'
 write(1,109) 'INIT'
 write(1,*)
 write(1,109) 'BOX'
 write(1,109) '--     ------- BOX ------'
 write(1,*) '1 ',NtotL,' 1 ',NtotL,' 1 1 /'
 write(1,*)
 write(1,109) 'TOPS'
 write(1,110) NtotL*NtotL,'*',datum,'  /'
 write(1,*)
```

```
      write(1,109) 'ENDBOX'
      write(1,*)

      write(1,109) 'DZ'
      if (Hthick) then

          if (barh.ge.100.) then
          write(1,111) NtotL*NtotL,'*',barh,'  /'
          else
          write(1,112) NtotL*NtotL,'*',barh,'  /'
          endif

      else

        if ((nfar+nclo).gt.0) then
          if (barh.ge.100.) then
          write(1,111) NtotL*(nfar+nclo),'*',barh,' '
          else
          write(1,112) NtotL*(nfar+nclo),'*',barh,' '
          endif
        endif

        do 3 j=1,nxy
        do 3 kj=1,nsub
        if ((nfar+nclo).gt.0) then
          if (barh.ge.100.) then
          write(1,111) nfar+nclo,'*',barh,' '
          else
          write(1,112) nfar+nclo,'*',barh,' '
          endif
        endif
        do 4 i=1,nxy
        do 4 ki=1,nsub
        write(1,*) xthick(i,j)
4       continue
        if ((nfar+nclo).gt.0) then
          if (barh.ge.100.) then
```

```
          write(1,111) nfar+nclo,'*',barh,' '
          else
          write(1,112) nfar+nclo,'*',barh,' '
          endif
        endif
3       continue

        if ((nfar+nclo).gt.0) then
          if (barh.ge.100.) then
          write(1,111) NtotL*(nfar+nclo),'*',barh,' '
          else
          write(1,112) NtotL*(nfar+nclo),'*',barh,' '
          endif
        endif

        write(1,*)
        write(1,109) '/'

      endif

      write(1,109) 'DXV'
      if (nfar.eq.0) then
        if (asub.ge.100.) then
        write(1,111) nxy*nsub+nclo*2,'*',asub,' /'
        else
        write(1,112) nxy*nsub+nclo*2,'*',asub,' /'
        endif
      else
        if (asub.ge.100.) then
        write(1,113) nfar,afar,nxy*nsub+nclo*2,asub,nfar,afar
        else
        write(1,114) nfar,afar,nxy*nsub+nclo*2,asub,nfar,afar
        endif
      endif

      write(1,109) 'DYV'
      if (nfar.eq.0) then
```

```
  if (asub.ge.100.) then
  write(1,111) nxy*nsub+nclo*2,'*',asub,' /'
  else
  write(1,112) nxy*nsub+nclo*2,'*',asub,' /'
  endif
else
  if (asub.ge.100.) then
  write(1,113) nfar,afar,nxy*nsub+nclo*2,asub,nfar,afar
  else
  write(1,114) nfar,afar,nxy*nsub+nclo*2,asub,nfar,afar
  endif
endif
write(1,*)

if (Hperm) goto 11
write(1,109) 'PERMX'
write(1,*)

if ((nfar+nclo).gt.0) then
  if (bark.ge.100.) then
  write(1,111) NtotL*(nfar+nclo),'*',bark,' '
  else
  write(1,112) NtotL*(nfar+nclo),'*',bark,' '
  endif
endif

do 5 j=1,nxy
do 5 kj=1,nsub
if ((nfar+nclo).gt.0) then
  if (bark.ge.100.) then
  write(1,111) nfar+nclo,'*',bark,' '
  else
  write(1,112) nfar+nclo,'*',bark,' '
  endif
endif
do 6 i=1,nxy
do 6 ki=1,nsub
```

```
      write(1,*) xk(i,j)
 6    continue
      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
        write(1,111) nfar+nclo,'*',bark,' '
        else
        write(1,112) nfar+nclo,'*',bark,' '
        endif
      endif
 5    continue

      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
        write(1,111) NtotL*(nfar+nclo),'*',bark,' '
        else
        write(1,112) NtotL*(nfar+nclo),'*',bark,' '
        endif
      endif

      write(1,*)
      write(1,109) '/'

      write(1,109) 'PERMY'
      write(1,*)

      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
        write(1,111) NtotL*(nfar+nclo),'*',bark,' '
        else
        write(1,112) NtotL*(nfar+nclo),'*',bark,' '
        endif
      endif

      do 7 j=1,nxy
      do 7 kj=1,nsub
      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
```

```
        write(1,111) nfar+nclo,'*',bark,' '
        else
        write(1,112) nfar+nclo,'*',bark,' '
        endif
      endif
      do 8 i=1,nxy
      do 8 k=1,nsub
      write(1,*) xk(i,j)
8     continue
      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
        write(1,111) nfar+nclo,'*',bark,' '
        else
        write(1,112) nfar+nclo,'*',bark,' '
        endif
      endif
7     continue

      if ((nfar+nclo).gt.0) then
        if (bark.ge.100.) then
        write(1,111) NtotL*(nfar+nclo),'*',bark,' '
        else
        write(1,112) NtotL*(nfar+nclo),'*',bark,' '
        endif
      endif

      write(1,*)
      write(1,109) '/'
      write(1,*)
11    if (Hporo) goto 12
      write(1,109) 'PORO'
      write(1,*)

      if ((nfar+nclo).gt.0) then
      write(1,121) NtotL*(nfar+nclo),barporo
      endif
```

```
      do 9 j=1,nxy
      do 9 kj=1,nsub
      if ((nfar+nclo).gt.0) then
      write(1,121) nfar+nclo,barporo
      endif
      do 10 i=1,nxy
      do 10 k=1,nsub
      write(1,*) xporo(i,j)
10    continue
      if ((nfar+nclo).gt.0) then
      write(1,121) nfar+nclo,barporo
      endif
9     continue

      if ((nfar+nclo).gt.0) then
      write(1,121) NtotL*(nfar+nclo),barporo
      endif
      write(1,*)
      write(1,109) '/'
      write(1,*)

12    write(1,109) 'EQUALS'
      if (Hporo) write(1,*) ''''PORO'' ',barporo,' /'
      if (Hperm) write(1,*) ''''PERMX'' ',bark,' /'
      if (Hperm) write(1,*) ''''PERMY'' ',bark,' /'
      write(1,*) ''''PERMZ'' ',bark,' /'
      write(1,109) '/'
      write(1,*)
      write(1,109) 'PROPS     ============================='
      write(1,*)
      write(1,109) 'ROCK'
      write(1,109) '--  p(psia) cf'
      write(1,*) '14.7 ',ctot,' /'
      write(1,*)
      write(1,109) 'DENSITY'
      write(1,*) '--  OIL       WATER     GAS'
      write(1,*) doil,dwater,dgas,' /'
```

```
      write(1,*)
      write(1,109) 'RPTPROPS'
      write(1,*) '13*0 /'
      write(1,*)
      write(1,109) 'PVDO'
      write(1,109) '--Pb(psia)     B_o        vis_o(cp)'
      write(1,*) '14.7 ',B,vis
      write(1,*) pmax,B-.000001,vis,' /'
      write(1,*)
      write(1,109) 'PMAX'
      write(1,*) pmax,' /'
      write(1,*)
      write(1,109) 'SOLUTION    ==============================='
      write(1,*)
      write(1,109) 'EQUIL'
      write(1,115) datum,presi,woc,goc
      write(1,*)
      write(1,109) 'RPTSOL'
      write(1,109) ' /'
      write(1,*)
      write(1,109) 'SUMMARY    ==============================='
      write(1,109) 'WBHP'
      write(1,109) ' /'
      write(1,109) 'RPTSMRY'
      write(1,109) '1 /'
      write(1,*)
      write(1,109) 'RUNSUM'
      write(1,*)
      write(1,109) 'SCHEDULE    ==============================='
      write(1,*)
      write(1,109) 'RPTSCHED'
      write(1,*) '1 0 0 0 0 0 0 0 1 0 1 1 0 0 1 /'
      write(1,*)
      write(1,109) 'WELSPECS'
      do 13 i=1,nw
      write(1,116) i,nxwe(i),nywe(i),datum
 13   continue
```

```
      write(1,109) '/'
      write(1,109) 'COMPDAT'
      do 14 i=1,nw
      write(1,117) i,nxwe(i),nywe(i),rw(i)*2.
14    continue
      write(1,109) '/'
      write(1,109) 'WCONPROD'
      do 15 i=1,nw
      if (q(i).gt.0.) then
      write(1,118) i,q(i)
      else
      write(1,119) i,100.
      endif
15    continue
      write(1,109) '/'
      write(1,109) 'TUNING'
      write(1,120) tstart,tend,tstart,' 1.',tstep,tstep
      write(1,109) '/'
      write(1,109) '/'
      write(1,109) 'TSTEP'
      write(1,*) tend
      write(1,109) '/'
      write(1,109) 'END     ============================='
      close(1)

100   format(A7,42x,A23)
101   format(2x,A11)
102   format('=',1x,10(A6,1x))
103   format('=',1x,3(A3,1x),8(A6,1x))
104   format('=',1x,A)
105   format('=',1x,3(A5,1x),3(A6,1x))
106   format('=',1x,5(A6,1x),3(A5,1x),A6)
107   format('=',1x,7(A6,1x))
108   format('=',1x,6(A6,1x))
109   format(A)
110   format(2x,I8,A,F6.1,A)
111   format(2x,I8,A,F6.2,A)
```

```
 112  format(2x,I8,A,F6.3,A)
 113  format(2x,I8,'*',F6.2,1x,I8,'*',F6.2,1x,I8,'*',F6.2,'  /')
 114  format(2x,I8,'*',F6.2,1x,I8,'*',F6.3,1x,I8,'*',F6.2,'  /')
 115  format(1x,F7.1,1x,F7.1,1x,F7.1,' 0 ',F7.1,' 0 0 0 0 /')
 116  format(''WELL',I2,''' ''GROUP1''',1x,I3,1x,I3,1x,
     &F7.1,1x,'''OIL''',' /')
 117  format(''WELL',I2,''' ',I3,1x,I3,' 1 1 ''OPEN'' 0 1* '
     &,F4.1,' 3* ''z'' /')
 118  format(''WELL',I2,''' ',''OPEN'' ''ORAT'' ',F7.1,' /')
 119  format(''WELL',I2,''' ',''STOP'' ''ORAT'' ',F7.1,' /')
 120  format(1x,F7.6,1x,F7.2,1x,F7.6,1x,A,1x,F5.3,1x,F5.3,' /')
 121  format(1x,I8,'*',F5.4)

      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine calsimu

* CALL FLOW SIMULATOR ECLIPSE

      integer system,estat
      character *20 simul

      external system

      simul="runeclipse"
      estat=system(simul)

      return
      end

*-------------------------------------------------
*************************************************
```

```
*-------------------------------------------------

      subroutine outpres(Eclfile,treatpres,Hderiv,
     &to,po,dpo,No)

* TREATS ECLIPSE OUTPUT PRESSURES

      include 'influ.inc'
      logical Hderiv
      character *40 Eclfile
      real tloc(NTM),ploc(NTM),dploc(NTM)
      real to(NTM,NWM),po(NTM,NWM),pdo(NTM,NWM)

* reads ECLIPSE output file

      open(1,file=Eclfile,status="unknown")
      do 2 i=1,7
      read(1,'()')
 2    continue
      do 3 i=1,No
      read(1,'()')
      read(1,'()')
      read(1,'()')
      read(1,*) tloc(i),qloc,(po(i,j),j=1,nw)
 3    continue
      close(1)

* convert times in hours and p in Dp
      do 10 i=1,No
      do 10 j=1,nw
      to(i,j)=tloc(i)*24.
      po(i,j)=presi-po(i,j)
 10   continue

* dimensionless real pressures
      if (treatpres.eq.1.) call adim(No,to,po)
```

```
* dimensionless Laplace pressures
      if (treatpres.eq.2.) then
      call adim(No,to,po)

      pi=0.
      do 4 j=1,nw
      do 5 i=1,No
      tloc(i)=to(i,j)
      ploc(i)=po(i,j)
 5    continue
      do 4 i=1,No
      z=1./tloc(i)
      po(i,j)=flapl6(z,ploc,tloc,No,pi)
      po(i,j)=z*po(i,j)
 4    continue

      endif

* derivative
      if (Hderiv) then
      call deriv(No,to,po,dpo)
      endif

      return
      end

*****************************************************************

      subroutine adim(No,to,po)
* dimensionless variables
      include 'influ.inc'
      dimension to(NTM,NWM),po(NTM,NWM)

      do 1 j=1,nw
      if (q(j).eq.0.) then
      qad=qmax
      rwad=rwmax
```

```
      else
      qad=q(j)
      rwad=rw(j)
      endif
      alph1=bark*barh/(141.2*qad*B*vis)
      alph2=.000264*bark/(barporo*vis*ctot*rwad**2.)
      do 1 i=1,No
      po(i,j)=po(i,j)*alph1
      to(i,j)=to(i,j)*alph2
 1    continue

      return
      end

*****************************************************************

      subroutine deriv(No,tloc,ploc,dploc)
* computes log derivative
      include 'influ.inc'
      real tloc(NTM,NWM),ploc(NTM,NWM),dploc(NTM,NWM)

      do 1 iw=1,nw
      do 1 j=2,No-1
      dploc(j,iw)=(ploc(j+1,iw)-ploc(j-1,iw))
     &/(tloc(j+1,iw)-tloc(j-1,iw))
      dploc(j,iw)=dploc(j,iw)*tloc(j,iw)
 1    continue

      return
      end

*****************************************************************

*     PUTS SET OF DATA INTO LAPLACE SPACE
*     Marcel Bourgeois, 1992.

      function flapl6(z,pdm,tdm,ndp,pinitl)
```

```
      include 'influ.inc'
      dimension pdm(NTM),tdm(NTM)
* -- computation of last semilog slope
      nstep = 2
      if (nstep.eq.1) width=0.2
      if (nstep.eq.2) width=0.1
      if (nstep.eq.3) width=0.05
      if (nstep.eq.4) width=0.025
      do 60 ii=ndp-1, 2, -1
            if (log(tdm(ndp)/tdm(ii)).ge.width) then
*            if (dlog(tdm(ndp)/tdm(ii)).ge.width) then
            i=ii
            goto 62
            endif
 60   continue
 62   do 65 kk=i-1, 1, -1
            if (log(tdm(i)/tdm(kk)).ge.width) then
*            if (dlog(tdm(i)/tdm(kk)).ge.width) then
            k=kk
            goto 67
            endif
 65   continue
 67    blo = log(tdm(i)/tdm(k)) * (pdm(ndp)-pinitl)
     & / ( log(tdm(ndp)/tdm(i)) * log(tdm(ndp)/tdm(k)) )
     & + log( tdm(ndp)*tdm(k)/(tdm(i)*tdm(i)) ) * (pdm(i)-pinitl)
     & / ( log(tdm(ndp)/tdm(i)) * log(tdm(i)/tdm(k)) )
     & - log(tdm(ndp)/tdm(i)) * (pdm(k)-pinitl)
     & / ( log(tdm(i)/tdm(k)) * log(tdm(ndp)/tdm(k)) )
* 67    blo = log(tdm(i)/tdm(k)) * (pdm(ndp)-pinitl)
*     & / ( log(tdm(ndp)/tdm(i)) * log(tdm(ndp)/tdm(k)) )
*     & + log( tdm(ndp)*tdm(k)/(tdm(i)*tdm(i)) ) * (pdm(i)-pinitl)
*     & / ( log(tdm(ndp)/tdm(i)) * log(tdm(i)/tdm(k)) )
*     & - log(tdm(ndp)/tdm(i)) * (pdm(k)-pinitl)
*     & / ( dlog(tdm(i)/tdm(k)) * dlog(tdm(ndp)/tdm(k)) )
* --
*      blo = (pdm(ndp)-pdm(ndp-1))/dlog(tdm(ndp)/tdm(ndp-1))
```

```
*
* -- computation of Laplace transform with linear interpolation between
* -- data points, and semilog interpolation afterwards
*
      f0 = (pdm(1)-pinitl) / tdm(1)
      sum= f0 * (1. - exp(-z*tdm(1)) ) / (z*z)
      do 1 i=1, ndp-1
         fi = (pdm(i+1)-pdm(i)) / (tdm(i+1)-tdm(i))
         sum = sum + fi * ( exp(-z*tdm(i)) - exp(-z*tdm(i+1)) ) / (z*z)
 1    continue
* ---- a semi-log extrapolation for large t -------------
      sum= sum+ blo * expin(z*tdm(ndp)) / z
      flapl6 = sum
      return
      end

*****************************************************************

* COMPUTES EXPONENTIAL INTEGRAL
* From Abrahmovitz&Stegun

      real*8 function expin(x)

      a1=8.5733287401
      a2=18.0590169730
      a3=8.6347608925
      a4=.2677737343
      b1=9.5733223454
      b2=25.6329561486
      b3=21.0996530827
      b4=3.9584969228
      w=x**4.+a1*x**3.+a2*x**2.+a3*x+a4
      y=x**4.+b1*x**3.+b2*x**2.+b3*x+b4
      w=w/y
      expin=w*exp(-x)/(x)
      return
      end
```

```
*-------------------------------------------------
*************************************************
*-------------------------------------------------

      function acorn(seed,rset)

* RANDOM NUMBER GENERATOR

      parameter (KORDEI=50,MAXOP1=KORDEI+1,MAXINT=2**30)
      common/iaco/ ixv(MAXOP1)

* check to see if this is the first call
      if (rset.eq.1.)then
      do 10 i=1,KORDEI
      ixv(i)=i*seed
 10   continue
      rset=0.
      endif

* generates a random number
      do 20 i=1,KORDEI
            ixv(i+1)=(ixv(i+1)+ixv(i))
            if(ixv(i+1).ge.MAXINT) ixv(i+1)=ixv(i+1)-MAXINT
 20   CONTINUE
      acorn=dble(ixv(KORDEI+1))/MAXINT
      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine object(ib,jb,etad,lambd,objnew,ptest
     &,dptest,N,
     &oinit,ti,psim,dpsim,pcl,dpcl,varperm)
```

```
      include 'influ.inc'
      logical varperm
      dimension r(NWM),teta(NWM),xt1(NTM,NWM)
     &,xt2(NTM,NWM),ptest(NTM,NWM),dptest(NTM,NWM)

      xc=(ib-nxwmax)*2.*ad
      yc=(jb-nywmax)*2.*ad
      call place(xc,yc,r,rp,teta,tetap)


      nante=6
      write(*,*) 'etad=',etad,' lambd=',lambd

* computes influence function
      if (rp.ge.ad) then
      call so(r,rp,etad,lambd,teta,tetap
     &,N,ti,xt1)
      else
      call si(r,rp,etad,lambd,teta,tetap,
     &N,ti,xt1)
      endif
      call so0(r,rp,teta,tetap,N
     &,ti,xt2)

* computes new objective function

      objnew=0.
      do 1 j=1,nw
      do 2 i=1,N
      ptest(i,j)=psim(i,j)+xt1(i,j)-xt2(i,j)
 2    continue
 1    continue
      call deriv(N,ti,ptest,dptest)

      do 4 j=1,nw
      x=0.
      y=0.
```

```
      do 3 i=1,N
      x=x+abs(ptest(i,j)-pcl(i,j))/pcl(N,j)
      y=y+abs(dptest(i,j)-dpcl(i,j))/dpcl(N-nante,j)
 3    continue
      objnew=objnew+(1.-wderiv(j))*wel(j)*x
      objnew=objnew+wderiv(j)*wel(j)*y
 4    continue
      objnew=objnew/oinit

      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine place(xc,yc,r,rp,teta,tetap)

      include 'influ.inc'
      dimension r(NWM),teta(NWM)

* COMPUTES LOCATION PARAMETERS FOR SUBROUTINES SO,SOO,SI

      rp=sqrt(xc**2.+yc**2.)
      if (rp.eq.0) then
      rp=1.
      endif
      if (xc.gt.0.) then
      tetap=atan(yc/xc)*57.29577951+180.
      endif
      if (xc.lt.0.) then
      tetap=atan(yc/xc)*57.29577951
      endif
      if (xc.eq.0.) then
      if (yc.ge.0.) then
      tetap=-90.
      else
```

```
      tetap=90.
      endif
      endif

      do 1 i=1,nw-1
      r(i)=sqrt((xw(i)-xc)**2.+(yw(i)-yc)**2.)
      if (r(i).eq.0) then
      r(i)=1.
      endif

      if (xw(i).gt.xc) then
      teta(i)=atan((yw(i)-yc)/(xw(i)-xc))*57.29577951
      endif
      if (xw(i).lt.xc) then
      teta(i)=atan((yw(i)-yc)/(xw(i)-xc))*57.29577951+180.
      endif
      if (xw(i).eq.xc) then
      if (yw(i).ge.yc) then
      teta(i)=90.
      else
      teta(i)=-90.
      endif
      endif
 1    continue

      r(nw)=rp
      teta(nw)=tetap

      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine so(r,rp,etad,lambd,teta,tetap,N,rsio,pio)
```

```
* computes P(s) and its derivative for a
* EXTERNAL CIRCULAR DISCONTINUITY
* LAPLACE SPACE
* input : data about the geometry(a,r,rp,dk,teta,tetap)
* input : data about the number of terms in the series(m)
*---------------------------------------------------------------
      include 'influ.inc'
      real rsio(NTM,NWM),pio(NTM,NWM)
      real bsk0(NTM),bsk1(NTM),bsi0(NTM),bsi1(NTM)
      real bk0p(NTM),bir(NTM)
      dimension r(NWM),teta(NWM)
      complex z,com

      do 1 jd=1,nw

      Rg=r(jd)**2.+rp**2.-2*r(jd)*rp*cos(.017453292*(teta(jd)
     &-tetap))
      Rg=sqrt(Rg)
      if (Rg.lt.1.) then
      Rg=1.
      endif

      do 1 id=1,N
      s=1./rsio(id,jd)
      z=sqrt(s)
      call bsslk(0,z,1,com)
      Bk1=real(com)
      z=Rg*sqrt(s)
      call bsslk(0,z,0,com)
      Bk0=real(com)

      do 100 i=1,(m+1)
      z=rp*sqrt(s)
      call bsslk(0,z,i-1,com)
      bk0p(i)=real(com)
      z=ad*sqrt(s)
      call bsslk(0,z,i-1,com)
```

```
      bsk1(i)=real(com)
      call bssli(0,z,i-1,com)
      bsi0(i)=real(com)
      z=ad*sqrt(s/etad)
      call bssli(0,z,i-1,com)
      bsi1(i)=real(com)
      z=r(jd)*sqrt(s)
      call bsslk(0,z,i-1,com)
      bsk0(i)=real(com)
      z=r(jd)*sqrt(s/etad)
      call bssli(0,z,i-1,com)
      bir(i)=real(com)
  100 continue

      phio=bsi1(1)*bsi0(2)
      phio=phio-bsi1(2)*bsi0(1)*lambd/sqrt(etad)
      psio=bsi1(2)*bsk1(1)*lambd/sqrt(etad)
      psio=psio+bsi1(1)*bsk1(2)

      if (r(jd).ge.ad) then
      pw=Bk0/(sqrt(s)*Bk1)
      pw=pw+phio*bsk0(1)*bk0p(1)/psio
      else
      pw=bk0p(1)/bsi1(1)
      xmul=bsi0(1)+bsk1(1)*phio/psio
      pw=pw*xmul*bir(1)
      endif

      if (m.ge.1) then
      do 3 j=1,m
      phin=bsi1(j+1)*(bsi0(j)+bsi0(j+2))
      phin=phin-bsi0(j+1)*(bsi1(j)+bsi1(j+2))
     &*lambd/sqrt(etad)
      psin=bsk1(j+1)*(bsi1(j)+bsi1(j+2))
     &*lambd/sqrt(etad)
      psin=psin+bsi1(j+1)*(bsk1(j)+bsk1(j+2))
      xj=j
```

```
      ang=xj*.017453292*(teta(jd)-tetap)
      if (r(jd).ge.ad) then
      pw=pw+2*phin*bsk0(j+1)*bk0p(j+1)*cos(ang)/psin
      else
      xm=2.*bk0p(j+1)/bsi1(j+1)
      if (phin.eq.0.0) then
      xmul=bsi0(j+1)
      else
      xmul=bsi0(j+1)+bsk1(j+1)*phin/psin
      endif
      xmul=xmul*bir(j+1)
      xm=xm*xmul*cos(ang)
      pw=pw+xm
      endif
 3    continue
      endif

      pio(id,jd)=pw
 1    continue
      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine si(r,rp,etad,lambd,teta,tetap,N,rsio,pio)

* computes Pd(s) and its derivative for a
* INTERNAL CIRCULAR DISCONTINUITY
* LAPLACE SPACE
* need to use stehfest.f to invert in real space
* input : data about the geometry(a,r,rp,teta,tetap)
* input : data about the time scale(rsmin,drs,N)
* input : data about the number of terms in the series(m)
*-------------------------------------------------------------
      include 'influ.inc'
```

```
      dimension rsio(NTM,NWM),pio(NTM,NWM)
      dimension bkra(NTM),bka(NTM),bkr(NTM),bkrp(NTM)
      dimension bkaa(NTM),bir(NTM),birp(NTM),bia(NTM),biaa(NTM)
      dimension r(NWM),teta(NWM)
      complex zc,com

      do 1 jd=1,nw
      do 1 id=1,N
      s=1./rsio(id,jd)
      z=sqrt(s)
      za=sqrt(s*etad)

      do 100 i=1,m+1
      zc=z*ad
      call bsslk(0,zc,i-1,com)
      bka(i)=real(com)
      zc=rp*z
      call bsslk(0,zc,i-1,com)
      bkrp(i)=real(com)
      zc=ad*za
      call bsslk(0,zc,i-1,com)
      bkaa(i)=real(com)
      zc=rp*z
      call bssli(0,zc,i-1,com)
      birp(i)=real(com)
      zc=ad*z
      call bssli(0,zc,i-1,com)
      bia(i)=real(com)
      zc=ad*za
      call bssli(0,zc,i-1,com)
      biaa(i)=real(com)
  100 continue

      do 200 i=1,m+1
      zc=z*r(jd)
      call bsslk(0,zc,i-1,com)
      bkr(i)=real(com)
```

```
      zc=r(jd)*z
      call bssli(0,zc,i-1,com)
      bir(i)=real(com)
      zc=r(jd)*za
      call bsslk(0,zc,i-1,com)
      bkra(i)=real(com)
200   continue
      zc=r(jd)*za
      call bsslk(0,zc,0,com)
      bk0=real(com)

      if (r(jd).ge.ad) then
      psiod=z*bia(2)*bkaa(1)*lambd
      psiod=psiod+bia(1)*bkaa(2)*z*sqrt(etad)
      pw=birp(1)*bk0/psiod

      if (m.ge.1) then
      do 31 j=1,m
      xj=j
      ang=xj*.017453292*(teta(jd)-tetap)
      psiod=bkaa(j+1)*(bia(j)+bia(j+2))*lambd
      psiod=psiod+bia(j+1)*(bkaa(j)+bkaa(j+2))*sqrt(etad)
      psiod=psiod*z/2.
      pw=pw+2.*birp(j+1)*bkra(j+1)*cos(ang)/psiod
31    continue
      endif

      pw=pw*lambd/ad

      else
      omeo=bka(1)*bkaa(2)*sqrt(etad)
      omeo=omeo-bka(2)*bkaa(1)*lambd
      psio=bia(2)*bkaa(1)*lambd
      psio=psio+bia(1)*bkaa(2)*sqrt(etad)

      if (r(jd).le.rp) then
      pw=bir(1)*(bkrp(1)-birp(1)*omeo/psio)
```

```
      else
      pw=birp(1)*(bkr(1)-bir(1)*omeo/psio)
      endif

      if (m.ge.1) then
      do 32 j=1,m
      xj=j
      ang=xj*.017453292*(teta(jd)-tetap)
      omem=bka(j+1)*(bkaa(j)+bkaa(j+2))*sqrt(etad)
      omem=omem-bkaa(j+1)*(bka(j)+bka(j+2))*lambd
      psimi=bkaa(j+1)*(bia(j)+bia(j+2))*lambd
      psimi=psimi+bia(j+1)*(bkaa(j)+bkaa(j+2))*sqrt(etad)
      if (r(jd).le.rp) then
      pw=pw+2*bir(j+1)*(bkrp(j+1)-birp(j+1)*omem/psimi)
     &*cos(ang)
      else
      pw=pw+2*birp(j+1)*(bkr(j+1)-bir(j+1)*omem/psimi)
     &*cos(ang)
      endif
 32   continue
      endif

c      pw=pw*dk

      endif

      pio(id,jd)=pw
 1    continue
      return
      end

*-------------------------------------------------
*************************************************
*-------------------------------------------------

      subroutine so0(r,rp,teta,tetap,N,rsio,pio)
```

```
* LAPLACE SPACE
* r : distance puits producteur-centre du disque
* rp : distance puits obsevateur-centre du disque
* teta,tetap : angles
* dk=k1/k2    : rapport des permeabilites
* smin,smax,ds : bornes sur s

*-------------------------------------------------------------
      include 'influ.inc'
      dimension rsio(NTM,NWM),pio(NTM,NWM)
      dimension r(NWM),teta(NWM)
      complex z,com

      do 1 jd=1,nw
      do 1 id=1,N
      Rg=r(jd)**2.+rp**2.-2*r(jd)*rp*cos
     &(.017453292*(teta(jd)-tetap))
      Rg=sqrt(Rg)
      if (Rg.lt.1.) then
      Rg=1.
      endif
      s=1./rsio(id,jd)

      z=Rg*sqrt(s)
      call bsslk(0,z,0,com)
      Bk0=real(com)
      z=sqrt(s)
      call bsslk(0,z,1,com)
      Bk1=real(com)
      pw=Bk0/(sqrt(s)*Bk1)

      pio(id,jd)=pw

 1    continue

      return
      end
```

```
*-------------------------------------------------
************************************************
*-------------------------------------------------

      subroutine update(No,psim,dpsim,pcl,dpcl,obj,oinit)

* update simulated pressures using Eclipse run

      include 'influ.inc'

      logical Hderiv
      character *40 Eclfile

      Hderiv=.true.
      nante=6

* write Eclipse input file
      call writecl
* run Eclipse
      call calsimu
* read output pressures & Laplace t.
      Eclfile="ECLI.A0001                              "
      treatpres=2.
      call outpres(Eclfile,treatpres,Hderiv,
     &ti,psim,dpsim,No)

* update Objective function
      obj=0.
      do 1 j=1,nw
      x=0.
      y=0.
      do 2 i=1,No
      x=x+abs(psim(i,j)-pcl(i,j))/pcl(No,j)
      y=y+abs(dpsim(i,j)-dpcl(i,j))/dpcl(No-nante,j)
 2    continue
      obj=obj+(1.-wderiv(j))*wel(j)*x
```

```
      obj=obj+wderiv(j)*wel(j)*y
 1    continue
      obj=obj/oinit

      return
      end

*--------------------------------------------------
**************************************************
*--------------------------------------------------

      subroutine outprog(N,Nc,ofile,nswap,obj,
     &nperturb,tid,ti,pcl,tc,pc,pel,
     &nrunecl,pe,affi)


      include 'influ.inc'
      logical Hderiv
      real affi(NBM)
      character *40 Eclfile
      character *6 ofile
      character *80 filef
      real pfin(NTM,NWM),dp(NTM,NWM)
      real toutd(NTM,NWM),tout(NTM,NWM),pfinl(NTM,NWM)

* write output files

      Hderiv=.false.
      Nout=136
      tstart=.0005
      tstep=1.04
      tend=3.

* output file in real space
* write Eclipse input file
      call writecl
* run Eclipse
```

```
      call calsimu
      Eclfile="ECLI.A0001                                  "
* output file in real space
      treatpres=0.
      call outpres(Eclfile,treatpres,Hderiv,
     &tout,pfin,dp,Nout)
* ouput file in Laplace space
      treatpres=2.
      Eclfile="ECLI.A0001                                  "
      call outpres(Eclfile,treatpres,Hderiv,
     &toutd,pfinl,dp,Nout)

* write Permeability output file
      filef=ofile//'-perm.out'
      open(1,file=filef,status="unknown")
      write(1,*) 'permeability recovered'
      write(1,*) '1'
      write(1,*) 'Permeability'
      do 2 j=1,nxy
      do 2 i=1,nxy
      write(1,*) xk(i,j)
 2    continue
      close(1)

* write Porosity output file
      filef=ofile//'-poro.out'
      open(15,file=filef,status="unknown")
      write(15,*) 'porosity recovered'
      write(15,*) '1'
      write(15,*) 'Porosity'
      do 16 j=1,nxy
      do 16 i=1,nxy
      write(15,*) xporo(i,j)
 16   continue
      close(15)

* write Thickness output file
```

```
      filef=ofile//'-thick.out'
      open(17,file=filef,status="unknown")
      write(17,*) 'thickness upscaled'
      write(17,*) '1'
      write(17,*) 'Thickness'
      do 18 j=1,nxy
      do 18 i=1,nxy
      write(17,*) xthick(i,j)
 18   continue
      close(17)

      indi=1
      if (affi(indi).eq.1.) then
* write Laplace/true pressures
      call deriv(N,ti,pcl,dp)
      do 3 iw=1,nw
      j=iw+64
      filef=ofile//'TL-WEll'//char(j)//'.out'
      open(iw+20,file=filef,status="unknown")
      do 4 i=2,N-1
      write(iw+20,*) ti(i,iw),pcl(i,iw),dp(i,iw)
 4    continue
      close(iw+20)
 3    continue
      endif

      indi=indi+1
      if (affi(indi).eq.1.) then
* write Real/true pressures
      call deriv(Nc,tc,pc,dp)
      do 5 iw=1,nw
      j=iw+64
      filef=ofile//'TR-WEll'//char(j)//'.out'
      open(iw+30,file=filef,status="unknown")
      do 6 i=2,Nc-1
      write(iw+30,*) tc(i,iw),pc(i,iw),dp(i,iw)
 6    continue
```

```
      close(iw+30)
 5    continue
      endif

      indi=indi+1
      if (affi(indi).eq.1.) then
* write Laplace/starting pressures
      call deriv(N,ti,pel,dp)
      do 7 iw=1,nw
      j=iw+64
      filef=ofile//'SL-WEll'//char(j)//'.out'
      open(iw+40,file=filef,status="unknown")
      do 8 i=2,N-1
      write(iw+40,*) ti(i,iw),pel(i,iw),dp(i,iw)
 8    continue
      close(iw+40)
 7    continue
      endif

      indi=indi+1
      if (affi(indi).eq.1.) then
* write Real/starting pressures
      call deriv(N,tid,pe,dp)
      do 9 iw=1,nw
      j=iw+64
      filef=ofile//'SR-WEll'//char(j)//'.out'
      open(iw+50,file=filef,status="unknown")
      do 10 i=2,N-1
      write(iw+50,*) tid(i,iw),pe(i,iw),dp(i,iw)
 10   continue
      close(iw+50)
 9    continue
      endif

      indi=indi+1
      if (affi(indi).eq.1.) then
* write Laplace/Final pressures
```

```
      call deriv(Nout,toutd,pfinl,dp)
      do 11 iw=1,nw
      j=iw+64
      filef=ofile//'FL-WEll'//char(j)//'.out'
      open(iw+60,file=filef,status="unknown")
      do 12 i=2,Nout-1
      write(iw+60,*) toutd(i,iw),pfinl(i,iw),dp(i,iw)
 12   continue
      close(iw+60)
 11   continue
      endif

      indi=indi+1
      if (affi(indi).eq.1.) then
* write Real/Final pressures
      call deriv(Nout,tout,pfin,dp)
      do 13 iw=1,nw
      j=iw+64
      filef=ofile//'FR-WEll'//char(j)//'.out'
      open(iw+70,file=filef,status="unknown")
      do 14 i=2,Nout-1
      write(iw+70,*) tout(i,iw),pfin(i,iw),dp(i,iw)
 14   continue
      close(iw+70)
 13   continue
      endif

* write debug file
      filef=ofile//'DEBUG.out'
      open(100,file=filef,status="unknown")
      write(100,*) 'INFLU RUN - RUN PARAMETERS'
      write(100,*)
      write(100,200) 'Lecl',sas(1)
      write(100,300) 'Errmin',sas(2)
      write(100,200) 'Itmax',sas(3)
      write(100,200) 'Ntrymax',sas(4)
      write(100,300) 'Lt',sas(5)
```

```
      write(100,200) 'Nacceptmax',sas(6)
      write(100,300) 'To',sas(7)
      write(100,200) 'Eclo',Sas(8)
      write(100,400) 'nloo',nloo
      write(100,*)
      write(100,300) 'Final O',obj
      write(100,400) 'Number of tried swaps',nswap
      write(100,400) 'Number of accepted swap',nperturb
      write(100,400) 'Number of Eclipse runs',nrunecl
      close(100)


 200  format(A,'=',1x,F6.0)
 300  format(A,'=',1x,F6.4)
 400  format(A,'=',1x,I8)
      return
      end

************************************************************************
************************************************************************
************************************************************************

      SUBROUTINE BSSLK (MO, A, IN, W)
************************************************************************
*     FORTRAN SUBROUTINE FOR MODIFIED BESSEL FUNCTION OF INTEGRAL ORDER
*     MO = MODE OF OPERATION
*     'LIBNAVAL' SUBROUTINES LIBRARY.
************************************************************************
*     A  = ARGUMENT (COMPLEX NUMBER)
*     IN = ORDER (INTEGER)
*     W  = FUNCTION OF SECOND KIND (COMPLEX NUMBER)
*     -------------------
c     implicit double precision(a-h,o-z)
      COMPLEX A, W
      DIMENSION AZ(2)
      DIMENSION CD(30), CE(30)
      DIMENSION SZ(2), RZ(2), ZL(2)
```

```
      DIMENSION TS(2), TM(2), SM(2), SL(2), SQ(2), SR(2), AQ(2), QF(2)
      DATA CD(1) / 0.00000000000000E00/,  CD(2) /-1.64899505142212E-2/,
     1     CD(3) /-7.18621880068536E-2/,  CD(4) /-1.67086878124866E-1/,
     2     CD(5) /-3.02582250219469E-1/,  CD(6) /-4.80613945245927E-1/,
     3     CD(7) /-7.07075239357898E-1/,  CD(8) /-9.92995790539516E-1/,
     4     CD(9) /-1.35583925612592E00/,  CD(10)/-1.82105907899132E00/,
     5     CD(11)/-2.42482175310879E00/,  CD(12)/-3.21956655708750E00/,
     6     CD(13)/-4.28658077248384E00/,  CD(14)/-5.77022816798128E00/,
     7     CD(15)/-8.01371260952526E00/
      DATA CD(16)/ 0.00000000000000E00/,  CD(17)/-5.57742429879505E-3/,
     1     CD(18)/-4.99112944172476E-2/,  CD(19)/-1.37440911652397E-1/,
     2     CD(20)/-2.67233784710566E-1/,  CD(21)/-4.40380166808682E-1/,
     3     CD(22)/-6.61813614872541E-1/,  CD(23)/-9.41861077665017E-1/,
     4     CD(24)/-1.29754130468326E00/,  CD(25)/-1.75407696719816E00/,
     5     CD(26)/-2.34755299882276E00/,  CD(27)/-3.13041332689196E00/,
     6     CD(28)/-4.18397120563729E00/,  CD(29)/-5.65251799214994E00/,
     7     CD(30)/-7.87863959810677E00/
      DATA CE(1) / 0.00000000000000E00/,  CE(2) /-4.80942336387447E-3/,
     1     CE(3) /-1.31366200347759E-2/,  CE(4) /-1.94843834008458E-2/,
     2     CE(5) /-2.19948900032003E-2/,  CE(6) /-2.09396625676519E-2/,
     3     CE(7) /-1.74600268458650E-2/,  CE(8) /-1.27937813362085E-2/,
     4     CE(9) /-8.05234421796592E-3/,  CE(10)/-4.15817375002760E-3/,
     5     CE(11)/-1.64317738747922E-3/,  CE(12)/-4.49175585314709E-4/,
     6     CE(13)/-7.28594765574007E-5/,  CE(14)/-5.38265230658285E-6/,
     7     CE(15)/-9.93779048036289E-8/
      DATA CE(16)/ 0.00000000000000E00/,  CE(17)/ 7.53805779200591E-2/,
     1     CE(18)/ 7.12293537403464E-2/,  CE(19)/ 6.33116224228200E-2/,
     2     CE(20)/ 5.28240264523301E-2/,  CE(21)/ 4.13305359441492E-2/,
     3     CE(22)/ 3.01350573947510E-2/,  CE(23)/ 2.01043439592720E-2/,
     4     CE(24)/ 1.18552223068074E-2/,  CE(25)/ 5.86055510956010E-3/,
     5     CE(26)/ 2.25465148267325E-3/,  CE(27)/ 6.08173041536336E-4/,
     6     CE(28)/ 9.84215550625747E-5/,  CE(29)/ 7.32139093038089E-6/,
     7     CE(30)/ 1.37279667384666E-7/
*     -------------------
      AZ(1)=REAL(A)
      AZ(2)=0.
      ZS=AZ(1)*AZ(1)+AZ(2)*AZ(2)
```

```
      ZL(1)=0.5*LOG(ZS)
      ZL(2)=ATAN2(AZ(2),AZ(1))
      AN=IABS(IN)
      TM(1)=0.0
      TM(2)=0.0
      IF(MO.NE.0)GO TO 002
      TM(1)=AZ(1)
      TM(2)=AZ(2)
  002 IF(ZS-1.0)020,020,003
  003 IF(ZS-289.0)004,010,010
  004 IF(AZ(1)+0.096*AZ(2)*AZ(2))020,020,015
  010 QM=1.25331413731550*EXP(-0.5*ZL(1)-TM(1))
      QF(1)=QM*COS(-0.5*ZL(2)-TM(2))
      QF(2)=QM*SIN(-0.5*ZL(2)-TM(2))
      IF(AN.GT.1.0)GO TO 012
      PN=AN
      ASSIGN 011 TO LA
      GO TO 100
  011 TS(1)=QF(1)*SM(1)-QF(2)*SM(2)
      TS(2)=QF(1)*SM(2)+QF(2)*SM(1)
      SM(1)=TS(1)
      SM(2)=TS(2)
      GO TO 029
  012 PN=1.0
      ASSIGN 013 TO LA
      GO TO 100
  013 SQ(1)=QF(1)*SM(1)-QF(2)*SM(2)
      SQ(2)=QF(1)*SM(2)+QF(2)*SM(1)
      PN=0.0
      ASSIGN 014 TO LA
      GO TO 100
  014 SR(1)=QF(1)*SM(1)-QF(2)*SM(2)
      SR(2)=QF(1)*SM(2)+QF(2)*SM(1)
      GO TO 026
  015 QM=1.25331413731550*EXP(-0.5*ZL(1)-TM(1))
      QF(1)=QM*COS(-0.5*ZL(2)-TM(2))
      QF(2)=QM*SIN(-0.5*ZL(2)-TM(2))
```

```
      IF(AN.GT.1.0)GO TO 017
      PN=AN
      ASSIGN 016 TO LR
      GO TO 104
  016 TS(1)=QF(1)*SM(1)-QF(2)*SM(2)
      TS(2)=QF(1)*SM(2)+QF(2)*SM(1)
      SM(1)=TS(1)
      SM(2)=TS(2)
      GO TO 029
  017 PN=1.0
      ASSIGN 018 TO LR
      GO TO 104
  018 SQ(1)=QF(1)*SM(1)-QF(2)*SM(2)
      SQ(2)=QF(1)*SM(2)+QF(2)*SM(1)
      PN=0.0
      ASSIGN 019 TO LR
      GO TO 104
  019 SR(1)=QF(1)*SM(1)-QF(2)*SM(2)
      SR(2)=QF(1)*SM(2)+QF(2)*SM(1)
      GO TO 026
  020 QF(1)=1.0
      QF(2)=0.0
      IF(MO.EQ.0)GO TO 021
      QM=EXP(AZ(1))
      QF(1)=QM*COS(AZ(2))
      QF(2)=QM*SIN(AZ(2))
  021 IF(AN.GT.1.0)GO TO 023
      PN=AN
      ASSIGN 022 TO LK
      GO TO 106
  022 TS(1)=QF(1)*SM(1)-QF(2)*SM(2)
      TS(2)=QF(1)*SM(2)+QF(2)*SM(1)
      SM(1)=TS(1)
      SM(2)=TS(2)
      GO TO 029
  023 PN=1.0
      ASSIGN 024 TO LK
```

```
    GO TO 106
024 SQ(1)=QF(1)*SM(1)-QF(2)*SM(2)
    SQ(2)=QF(1)*SM(2)+QF(2)*SM(1)
    PN=0.0
    ASSIGN 025 TO LK
    GO TO 106
025 SR(1)=QF(1)*SM(1)-QF(2)*SM(2)
    SR(2)=QF(1)*SM(2)+QF(2)*SM(1)
026 RZ(1)=+AZ(1)/ZS
    RZ(2)=-AZ(2)/ZS
    PN=0.0
    GO TO 028
027 SQ(1)=SR(1)
    SQ(2)=SR(2)
    SR(1)=SM(1)
    SR(2)=SM(2)
028 SM(1)=2.0*PN*(RZ(1)*SR(1)-RZ(2)*SR(2))+SQ(1)
    SM(2)=2.0*PN*(RZ(1)*SR(2)+RZ(2)*SR(1))+SQ(2)
    PN=PN+1.0
    IF(PN.LT.AN)GO TO 027
029 W=CMPLX(SM(1),SM(2))
    RETURN
100 SM(1)=0.0
    SM(2)=0.0
    RZ(1)=+0.5*AZ(1)/ZS
    RZ(2)=-0.5*AZ(2)/ZS
    QN=(PN-0.5)*(PN+0.5)
    TM(1)=1.0
    TM(2)=0.0
    PM=0.0
    GO TO 102
101 QN=QN-2.0*PM
    PM=PM+1.0
    TS(1)=RZ(1)*TM(1)-RZ(2)*TM(2)
    TS(2)=RZ(1)*TM(2)+RZ(2)*TM(1)
    TM(1)=QN*TS(1)/PM
    TM(2)=QN*TS(2)/PM
```

```
      IF(ABS(SM(1))+ABS(TM(1)).NE.ABS(SM(1)))GO TO 102
      IF(ABS(SM(2))+ABS(TM(2)).EQ.ABS(SM(2)))GO TO 103
  102 SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
      IF(PM.LT.36.0)GO TO 101
  103 GO TO LA,(011,013,014)
  104 SM(1)=1.0
      SM(2)=0.0
      M=15.0*PN+2.0
      N=15.0*PN+15.0
      DO 105 I=M,N
      TS(1)=AZ(1)-CD(I)
      TS(2)=AZ(2)
      SS=TS(1)*TS(1)+TS(2)*TS(2)
      TM(1)=+CE(I)*TS(1)/SS
      TM(2)=-CE(I)*TS(2)/SS
      SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
  105 CONTINUE
      GO TO LR,(016,018,019)
  106 AQ(1)=1.0
      AQ(2)=0.0
      RN=0.0
      SN=-1.0
      PM=0.0
      GO TO 108
  107 PM=PM+1.0
      RN=RN+0.5/PM
      SN=-SN
      TS(1)=0.5*(AZ(1)*AQ(1)-AZ(2)*AQ(2))
      TS(2)=0.5*(AZ(1)*AQ(2)+AZ(2)*AQ(1))
      AQ(1)=TS(1)/PM
      AQ(2)=TS(2)/PM
  108 IF(PM.LT.PN)GO TO 107
      SZ(1)=0.25*(AZ(1)-AZ(2))*(AZ(1)+AZ(2))
      SZ(2)=0.5*AZ(1)*AZ(2)
      SR(1)=0.0
```

```
      SR(2)=0.0
      SS=AQ(1)*AQ(1)+AQ(2)*AQ(2)
      TM(1)=+AQ(1)/SS
      TM(2)=-AQ(2)/SS
      PM=0.0
      GO TO 110
  109 TM(1)=TM(1)/(PN-PM)
      TM(2)=TM(2)/(PN-PM)
      SR(1)=SR(1)+0.5*TM(1)
      SR(2)=SR(2)+0.5*TM(2)
      PM=PM+1.0
      TS(1)=SZ(1)*TM(1)-SZ(2)*TM(2)
      TS(2)=SZ(1)*TM(2)+SZ(2)*TM(1)
      TM(1)=-TS(1)/PM
      TM(2)=-TS(2)/PM
  110 IF(PM.LT.PN)GO TO 109
      SM(1)=0.0
      SM(2)=0.0
      RM=1.0
      QM=0.0
      AQ(1)=SN*AQ(1)
      AQ(2)=SN*AQ(2)
      SL(1)=-0.115931515658412+ZL(1)-RN
      SL(2)=+ZL(2)
      PM=0.0
      GO TO 112
  111 QM=QM+RM
      PM=PM+1.0
      RM=0.25*ZS*RM/(PM*(PN+PM))
      TS(1)=SZ(1)*AQ(1)-SZ(2)*AQ(2)
      TS(2)=SZ(1)*AQ(2)+SZ(2)*AQ(1)
      AQ(1)=TS(1)/(PM*(PN+PM))
      AQ(2)=TS(2)/(PM*(PN+PM))
      SL(1)=SL(1)-0.5/PM-0.5/(PN+PM)
  112 TM(1)=AQ(1)*SL(1)-AQ(2)*SL(2)
      TM(2)=AQ(1)*SL(2)+AQ(2)*SL(1)
      SM(1)=SM(1)+TM(1)
```

```
      SM(2)=SM(2)+TM(2)
      IF(QM+RM.GT.QM)GO TO 111
      SM(1)=SR(1)+SM(1)
      SM(2)=SR(2)+SM(2)
      GO TO LK,(022,024,025)
      END

*---------------------------------------------
*********************************************
*---------------------------------------------

      SUBROUTINE BSSLI (MO, A, IN, W)
************************************************************************
*     FORTRAN SUBROUTINE FOR MODIFIED BESSEL FUNCTION OF INTEGRAL ORDER
*     'LIBNAVAL' SUBROUTINES LIBRARY
************************************************************************
*     MO = MODE OF OPERATION
*     A  = ARGUMENT (COMPLEX NUMBER)
*     IN = ORDER (INTEGER)
*     W  = FUNCTION OF FIRST KIND (COMPLEX NUMBER)
*     -------------------
c     implicit double precision(a-h,o-z)
      COMPLEX A, W
      DIMENSION AZ(2), FI(2)
      DIMENSION CD(30), CE(30)
      DIMENSION QZ(2), RZ(2), SZ(2), ZR(2)
      DIMENSION TS(2), TM(2), RM(4), SM(4), AQ(2), QF(2)
      DATA CD(1) / 0.00000000000000E00/,  CD(2) /-1.64899505142212E-2/,
     1     CD(3) /-7.18621880068536E-2/,  CD(4) /-1.67086878124866E-1/,
     2     CD(5) /-3.02582250219469E-1/,  CD(6) /-4.80613945245927E-1/,
     3     CD(7) /-7.07075239357898E-1/,  CD(8) /-9.92995790539516E-1/,
     4     CD(9) /-1.35583925612592E00/,  CD(10)/-1.82105907899132E00/,
     5     CD(11)/-2.42482175310879E00/,  CD(12)/-3.21956655708750E00/,
     6     CD(13)/-4.28658077248384E00/,  CD(14)/-5.77022816798128E00/,
     7     CD(15)/-8.01371260952526E00/
      DATA CD(16)/ 0.00000000000000E00/,  CD(17)/-5.57742429879505E-3/,
     1     CD(18)/-4.99112944172476E-2/,  CD(19)/-1.37440911652397E-1/,
```

```
     2     CD(20)/-2.67233784710566E-1/,  CD(21)/-4.40380166808682E-1/,
     3     CD(22)/-6.61813614872541E-1/,  CD(23)/-9.41861077665017E-1/,
     4     CD(24)/-1.29754130468326E00/,  CD(25)/-1.75407696719816E00/,
     5     CD(26)/-2.34755299882276E00/,  CD(27)/-3.13041332689196E00/,
     6     CD(28)/-4.18397120563729E00/,  CD(29)/-5.65251799214994E00/,
     7     CD(30)/-7.87863959810677E00/
      DATA CE(1) / 0.00000000000000E00/,  CE(2) /-4.80942336387447E-3/,
     1     CE(3) /-1.31366200347759E-2/,  CE(4) /-1.94843834008458E-2/,
     2     CE(5) /-2.19948900032003E-2/,  CE(6) /-2.09396625676519E-2/,
     3     CE(7) /-1.74600268458650E-2/,  CE(8) /-1.27937813362085E-2/,
     4     CE(9) /-8.05234421796592E-3/,  CE(10)/-4.15817375002760E-3/,
     5     CE(11)/-1.64317738747922E-3/,  CE(12)/-4.49175585314709E-4/,
     6     CE(13)/-7.28594765574007E-5/,  CE(14)/-5.38265230658285E-6/,
     7     CE(15)/-9.93779048036289E-8/
      DATA CE(16)/ 0.00000000000000E00/,  CE(17)/ 7.53805779200591E-2/,
     1     CE(18)/ 7.12293537403464E-2/,  CE(19)/ 6.33116224228200E-2/,
     2     CE(20)/ 5.28240264523301E-2/,  CE(21)/ 4.13305359441492E-2/,
     3     CE(22)/ 3.01350573947510E-2/,  CE(23)/ 2.01043439592720E-2/,
     4     CE(24)/ 1.18552223068074E-2/,  CE(25)/ 5.86055510956010E-3/,
     5     CE(26)/ 2.25465148267325E-3/,  CE(27)/ 6.08173041536336E-4/,
     6     CE(28)/ 9.84215550625747E-5/,  CE(29)/ 7.32139093038089E-6/,
     7     CE(30)/ 1.37279667384666E-7/
*     -------------------
      AZ(1)=REAL(A)
      AZ(2)=0.
      ZS=AZ(1)*AZ(1)+AZ(2)*AZ(2)
      ZM=SQRT(ZS)
      PN=IABS(IN)
      SN=+1.0
      IF(AZ(1))002,003,003
  002 QZ(1)=-AZ(1)
      QZ(2)=-AZ(2)
      IF(IN.EQ.IN/2*2)GO TO 004
      SN=-1.0
      GO TO 004
  003 QZ(1)=AZ(1)
      QZ(2)=AZ(2)
```

```
004 IF(ZM.LE.17.5+0.5*PN*PN)GO TO 005
    QN=PN
    GO TO 011
005 QN=0.5*ZM-0.5*ABS(QZ(1))+0.5*ABS(0.5*ZM-ABS(QZ(1)))
    IF(PN.LE.QN)GO TO 006
    QN=+AINT(0.0625*ZS)
    IF(PN.LE.QN)GO TO 039
    QN=PN
    GO TO 039
006 IF(ZM.LE.17.5)GO TO 007
    QN=+AINT(SQRT(2.0*(ZM-17.5)))
    GO TO 011
007 IF(ZS-1.0)009,008,008
008 IF(-ABS(AZ(1))+0.096*AZ(2)*AZ(2))009,010,010
009 QN=AINT(0.0625*ZS)
    IF(PN.LE.QN)GO TO 039
    QN=PN
    GO TO 039
010 QN=0.0
011 SZ(1)=QZ(1)
    SZ(2)=QZ(2)
    QM=SN*0.398942280401433
    ZR(1)=SQRT(SZ(1)+ZM)
    ZR(2)=SZ(2)/ZR(1)
    ZR(1)=0.707106781186548*ZR(1)
    ZR(2)=0.707106781186548*ZR(2)
    QF(1)=+QM*ZR(1)/ZM
    QF(2)=-QM*ZR(2)/ZM
    IF(ZM.LE.17.5)GO TO 017
012 RZ(1)=+0.5*QZ(1)/ZS
    RZ(2)=-0.5*QZ(2)/ZS
    AN=QN*QN-0.25
    SM(1)=0.0
    SM(2)=0.0
    SM(3)=0.0
    SM(4)=0.0
    TM(1)=1.0
```

```
      TM(2)=0.0
      PM=0.0
      GO TO 014
  013 AN=AN-2.0*PM
      PM=PM+1.0
      TS(1)=TM(1)*RZ(1)-TM(2)*RZ(2)
      TS(2)=TM(1)*RZ(2)+TM(2)*RZ(1)
      TM(1)=AN*TS(1)/PM
      TM(2)=AN*TS(2)/PM
  014 SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
      AN=AN-2.0*PM
      PM=PM+1.0
      TS(1)=TM(1)*RZ(1)-TM(2)*RZ(2)
      TS(2)=TM(1)*RZ(2)+TM(2)*RZ(1)
      TM(1)=AN*TS(1)/PM
      TM(2)=AN*TS(2)/PM
      IF(ABS(SM(3))+ABS(TM(1)).NE.ABS(SM(3)))GO TO 015
      IF(ABS(SM(4))+ABS(TM(2)).EQ.ABS(SM(4)))GO TO 016
  015 SM(3)=SM(3)+TM(1)
      SM(4)=SM(4)+TM(2)
      IF(PM.LT.35.0)GO TO 013
  016 TS(1)=SM(1)+SM(3)
      TS(2)=SM(2)+SM(4)
      SM(1)=SM(1)-SM(3)
      SM(2)=SM(2)-SM(4)
      SM(3)=TS(1)
      SM(4)=TS(2)
      GO TO 019
  017 SM(1)=1.0
      SM(2)=0.0
      SM(3)=1.0
      SM(4)=0.0
      M=15.0*QN+2.0
      N=15.0*QN+15.0
      DO 018 I=M,N
      TS(1)=-QZ(1)-CD(I)
```

```
      TS(2)=-QZ(2)
      SS=TS(1)*TS(1)+TS(2)*TS(2)
      TM(1)=+CE(I)*TS(1)/SS
      TM(2)=-CE(I)*TS(2)/SS
      SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
      TS(1)=QZ(1)-CD(I)
      TS(2)=QZ(2)
      SS=TS(1)*TS(1)+TS(2)*TS(2)
      TM(1)=+CE(I)*TS(1)/SS
      TM(2)=-CE(I)*TS(2)/SS
      SM(3)=SM(3)+TM(1)
      SM(4)=SM(4)+TM(2)
  018 CONTINUE
  019 RM(1)=SM(1)
      RM(2)=SM(2)
      IF(QZ(1).GE.17.5)GO TO 023
      AQ(1)=-2.0*QZ(1)
      IF(QZ(2))020,021,021
  020 AQ(2)=-2.0*QZ(2)-3.14159265358979*(QN+0.5)
      GO TO 022
  021 AQ(2)=-2.0*QZ(2)+3.14159265358979*(QN+0.5)
  022 QM=EXP(AQ(1))
      TS(1)=QM*COS(AQ(2))
      TS(2)=QM*SIN(AQ(2))
      RM(1)=RM(1)+TS(1)*SM(3)-TS(2)*SM(4)
      RM(2)=RM(2)+TS(1)*SM(4)+TS(2)*SM(3)
  023 IF(QN.EQ.PN)GO TO 037
      RM(3)=RM(1)
      RM(4)=RM(2)
      QN=QN+1.0
      IF(ZM.LE.17.5)GO TO 029
  024 AN=QN*QN-0.25
      SM(1)=0.0
      SM(2)=0.0
      SM(3)=0.0
      SM(4)=0.0
```

```
      TM(1)=1.0
      TM(2)=0.0
      PM=0.0
      GO TO 026
  025 AN=AN-2.0*PM
      PM=PM+1.0
      TS(1)=TM(1)*RZ(1)-TM(2)*RZ(2)
      TS(2)=TM(1)*RZ(2)+TM(2)*RZ(1)
      TM(1)=AN*TS(1)/PM
      TM(2)=AN*TS(2)/PM
  026 SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
      AN=AN-2.0*PM
      PM=PM+1.0
      TS(1)=TM(1)*RZ(1)-TM(2)*RZ(2)
      TS(2)=TM(1)*RZ(2)+TM(2)*RZ(1)
      TM(1)=AN*TS(1)/PM
      TM(2)=AN*TS(2)/PM
      IF(ABS(SM(3))+ABS(TM(1)).NE.ABS(SM(3)))GO TO 027
      IF(ABS(SM(4))+ABS(TM(2)).EQ.ABS(SM(4)))GO TO 028
  027 SM(3)=SM(3)+TM(1)
      SM(4)=SM(4)+TM(2)
      IF(PM.LT.35.0)GO TO 025
  028 TS(1)=SM(1)+SM(3)
      TS(2)=SM(2)+SM(4)
      SM(1)=SM(1)-SM(3)
      SM(2)=SM(2)-SM(4)
      SM(3)=TS(1)
      SM(4)=TS(2)
      GO TO 031
  029 SM(1)=1.0
      SM(2)=0.0
      SM(3)=1.0
      SM(4)=0.0
      M=15.0*QN+2.0
      N=15.0*QN+15.0
      DO 030 I=M,N
```

```
      TS(1)=-QZ(1)-CD(I)
      TS(2)=-QZ(2)
      SS=TS(1)*TS(1)+TS(2)*TS(2)
      TM(1)=+CE(I)*TS(1)/SS
      TM(2)=-CE(I)*TS(2)/SS
      SM(1)=SM(1)+TM(1)
      SM(2)=SM(2)+TM(2)
      TS(1)=+QZ(1)-CD(I)
      TS(2)=+QZ(2)
      SS=TS(1)*TS(1)+TS(2)*TS(2)
      TM(1)=+CE(I)*TS(1)/SS
      TM(2)=-CE(I)*TS(2)/SS
      SM(3)=SM(3)+TM(1)
      SM(4)=SM(4)+TM(2)
  030 CONTINUE
  031 RM(1)=SM(1)
      RM(2)=SM(2)
      IF(QZ(1).GE.17.5)GO TO 036
      AQ(1)=-2.0*QZ(1)
      IF(QZ(2))032,033,033
  032 AQ(2)=-2.0*QZ(2)-3.14159265358979*(QN+0.5)
      GO TO 034
  033 AQ(2)=-2.0*QZ(2)+3.14159265358979*(QN+0.5)
  034 QM=EXP(AQ(1))
      TS(1)=QM*COS(AQ(2))
      TS(2)=QM*SIN(AQ(2))
      RM(1)=RM(1)+TS(1)*SM(3)-TS(2)*SM(4)
      RM(2)=RM(2)+TS(1)*SM(4)+TS(2)*SM(3)
      GO TO 036
  035 TM(1)=-2.0*QN*QZ(1)/ZS
      TM(2)=+2.0*QN*QZ(2)/ZS
      TS(1)=TM(1)*RM(1)-TM(2)*RM(2)+RM(3)
      TS(2)=TM(1)*RM(2)+TM(2)*RM(1)+RM(4)
      RM(3)=RM(1)
      RM(4)=RM(2)
      RM(1)=TS(1)
      RM(2)=TS(2)
```

```
    QN=QN+1.0
036 IF(QN.LT.PN)GO TO 035
037 IF(MO.NE.0)GO TO 038
    QM=EXP(QZ(1))
    TM(1)=QM*COS(QZ(2))
    TM(2)=QM*SIN(QZ(2))
    TS(1)=TM(1)*RM(1)-TM(2)*RM(2)
    TS(2)=TM(1)*RM(2)+TM(2)*RM(1)
    RM(1)=TS(1)
    RM(2)=TS(2)
038 FI(1)=QF(1)*RM(1)-QF(2)*RM(2)
    FI(2)=QF(1)*RM(2)+QF(2)*RM(1)
    W=CMPLX(FI(1),FI(2))
    RETURN
039 SZ(1)=0.25*(QZ(1)*QZ(1)-QZ(2)*QZ(2))
    SZ(2)=0.5*QZ(1)*QZ(2)
    AN=QN
    SM(1)=0.0
    SM(2)=0.0
    SM(3)=0.0
    SM(4)=0.0
    TM(1)=1.0
    TM(2)=0.0
    PM=0.0
040 AN=AN+1.0
    TS(1)=TM(1)/AN
    TS(2)=TM(2)/AN
    SM(3)=SM(3)+TS(1)
    SM(4)=SM(4)+TS(2)
    TM(1)=TS(1)*SZ(1)-TS(2)*SZ(2)
    TM(2)=TS(1)*SZ(2)+TS(2)*SZ(1)
    PM=PM+1.0
    TM(1)=TM(1)/PM
    TM(2)=TM(2)/PM
    IF(ABS(SM(1))+ABS(TM(1)).NE.ABS(SM(1)))GO TO 041
    IF(ABS(SM(2))+ABS(TM(2)).EQ.ABS(SM(2)))GO TO 042
041 SM(1)=SM(1)+TM(1)
```

```
      SM(2)=SM(2)+TM(2)
      GO TO 040
  042 SM(1)=SM(1)+1.0
      AN=QN+1.0
      SM(3)=AN*SM(3)
      SM(4)=AN*SM(4)
      GO TO 044
  043 AN=QN*(QN+1.0)
      TM(1)=SZ(1)/AN
      TM(2)=SZ(2)/AN
      TS(1)=+TM(1)*SM(3)-TM(2)*SM(4)
      TS(2)=+TM(1)*SM(4)+TM(2)*SM(3)
      SM(3)=SM(1)
      SM(4)=SM(2)
      SM(1)=SM(1)+TS(1)
      SM(2)=SM(2)+TS(2)
      QN=QN-1.0
  044 IF(QN.GT.PN)GO TO 043
      QF(1)=SN
      QF(2)=0.0
      QN=0.0
      GO TO 046
  045 QN=QN+1.0
      TM(1)=QF(1)*QZ(1)-QF(2)*QZ(2)
      TM(2)=QF(1)*QZ(2)+QF(2)*QZ(1)
      QF(1)=0.5*TM(1)/QN
      QF(2)=0.5*TM(2)/QN
  046 IF(QN.LT.PN)GO TO 045
      IF(MO.EQ.0)GO TO 047
      QM=EXP(-QZ(1))
      TM(1)=QM*COS(-QZ(2))
      TM(2)=QM*SIN(-QZ(2))
      TS(1)=TM(1)*QF(1)-TM(2)*QF(2)
      TS(2)=TM(1)*QF(2)+TM(2)*QF(1)
      QF(1)=TS(1)
      QF(2)=TS(2)
  047 FI(1)=QF(1)*SM(1)-QF(2)*SM(2)
```

```
      FI(2)=QF(1)*SM(2)+QF(2)*SM(1)
      W=CMPLX(FI(1),FI(2))
      RETURN
      END
```

A.4 *influ.inc* - Definition of Variables

```
*     NTM= max number of time steps
*     NWM= max number of wells
*     NBM=max number of blocks along x (or y) axis

      parameter(NTM=200,NWM=20,NBM=100)

* declaration of variables and arrays

      real rw(NWM),q(NWM),wel(NWM),xw(NWM),yw(NWM)
      real sas(8)
     &,wderiv(NWM)
      real tc(NTM,NWM),ti(NTM,NWM),pc(NTM,NWM),
     &tid(NTM,NWM),pcl(NTM,NWM)
     &,pe(NTM,NWM),pel(NTM,NWM),
     &psim(NTM,NWM)
     &,dpcl(NTM,NWM),dpel(NTM,NWM),dpsim(NTM,NWM)
      real xkh(NBM,NBM),xk(NBM,NBM),xkf(NBM,NBM)
      real xporoh(NBM,NBM),xporo(NBM,NBM),xporof(NBM,NBM)
      real xthick(NBM,NBM),xthickf(NBM,NBM)
      integer nxw(NWM),nyw(NWM)
      real presi,bark,barh,B,vis,ctot,barporo,a,ad,afar,
     &epstad,lambd,xdum,tstart,tend,tstep,qmax,rwmax
      integer m,nloo,nxy,nclo,nfar,nsub,nw,nxwmax,nywmax
      logical Hthick,Hporo,Hperm,Hupdate

* common blocks

      common rw,q,wel,xw,yw,sas,xkh,xk,nxw,nyw
     &,epstad,xporo,xporoh,xthick
     &,wderiv
      common presi,bark,barh,B,vis,ctot,barporo,a,ad,afar,
     &xdum,tstart,tend,tstep,qmax,rwmax
      common m,nloo,nxy,nclo,nfar,nsub,nw,nxwmax,nywmax
      common Hthick,Hporo,Hperm,Hupdate
```