PETROLEUM RESERVOIR SIMULATION USING FINITE VOLUME METHOD WITH NON-STRUCTURED GRIDS AND PARALLEL DISTRIBUTED COMPUTING

Neyval C. Reis Jr.[1]
João P. De Angeli
Alberto F. de Souza
Raul H. C. Lopes
Departamento de Informática
Centro Tecnológico – UFES
Av. Fernando Ferrari, s/n
29.060-970 – Vitória – ES

Abstract. *This paper discusses the implementation of a numerical method for simulating petroleum reservoirs, based on finite volume technique using non-structured grids, designed for parallel computing platforms with distributed-memory, particularly for clusters of workstations. Non-structured meshes are used in order to enhance the capability of conforming the irregular boundary of the reservoir allowing better application of boundary conditions and enabling an easier specification of geological faults. The parallel implementation is based on the domain decomposition, where the original reservoir is decomposed into several domains, each of which given to a separate processing node. All nodes then execute computations in parallel, each node on its associated sub-domain. The parallel computations include initialisation, coefficient generation, linear solution on the sub-domain, and inter-node communication. The exchange of information across the sub-domains, or processors, is achieved using the message passing interface standard, MPI. The use of MPI ensures portability across different computing platforms ranging from massively parallel machines to clusters of workstations. In order to validate the solution procedure, the well-known five-spot problem is simulated. The execution time and speed-up are evaluated through comparing the performance of different numbers of processors. The results indicate that the parallel code can significantly improve prediction capability and efficiency for large-scale simulations.*

Keywords: *Parallel Processing, Finite Volume Method, Reservoir Simulation, Non-structured Grids.*

[1] Corresponding author - e-mail: neyval@inf.ufes.br

1. INTRODUCTION

One of the main goals of petroleum industry is to maximize oil production from each reservoir. It is well known that only a fraction of the oil in a reservoir is produced due to its own internal pressure. As soon as the natural pressure of the reservoir decreases significantly and oil production starts to decline, it is necessary to utilize methods for additional recovery. One of the most used methods to increase oil recovery is based on the injection-production mechanism: apart from the production wells, which produce oil, there are injection wells, which inject water (or chemicals) into the reservoir. As water is injected, it occupies some space in the reservoir displacing the oil towards the production well, consequently increasing recuperation.

The success of this technique is very dependent on the fluid flow pattern. Therefore, the numerical simulation of petroleum reservoir is of great importance in petroleum engineering, since full experimental simulation in laboratory is impossible. With the advances in computing capabilities in terms of speed of computation and increase in computer memory, as well as, the advances in numerical techniques, reservoir simulators have become an established technique in the petroleum industry to assess the quality of reservoirs and to plan production strategies. Reservoir simulators are capable of giving valuable insights on the pattern of the fluid flow inside the porous rock, enabling the optimum positioning of wells (production/injection) to maximize recuperation efficiency.

Nevertheless, the complex nature of the problem requires an enormous computational effort to perform full-scale simulations. The need for high performance computing for reservoir simulations is due to for the physics of the process being investigated. In addition to the transient and tri-dimensional form of the governing equations, there are several length scales of significant importance upon the process. Based on core samples analysis of porous media one can obtain rock porosity maps with resolutions as high as 100 mm (Killough, 1995), while a characteristic reservoir size can exceed 1 km.

In fact, the question of how to scale up from laboratory scale to field scale is still open. However, a numerical scheme that can incorporate a description of local heterogeneities as fine as possible, given the state of art computing, represents a valuable tool in the development of full-scale reservoir simulations (Chang & Mahanty, 1997; Christie, 2001). This motivates the development of this work, which presents the implementation of a finite volume technique using non-structured grids, designed for parallel computing platforms. Non-structured meshes are used in order to enhance the capability of conforming the irregular boundary of the reservoir enabling easier specification of geological faults.

The algorithm is designed for parallel computing platforms with distributed-memory. Access to supercomputers can be limited or very expensive. Thus, the code is mainly developed for clusters of workstations, which is one of the modern solutions to avoid the cost limitations imposed by supercomputers, since the rapid increase of microprocessor and network performance has enabled high levels of computing power for a small fraction of the price of supercomputers.

2. PETROLEUM RESERVOIR MODELLING

When several fluid phases flow in a porous medium, each phase affects the flow of the other, depending on the concentration of the various phases. This section describes the governing equations of the problem. In this paper the well-known black oil is used to describe the flow of 2 components (water + oil). The set of equation used by the black-oil model are obtained by performing a mass conservation balance for each component, Allen & Furtado

(1997) present an in depth description of the subject. Neglecting body forces and the capillary pressure effects, the conservation equations for water and oil are written as:

$$\frac{\partial}{\partial t}\left[e\frac{C_w}{B_w}\right] - \nabla.\left[l_w \nabla P\right] = q_w \tag{1}$$

$$\frac{\partial}{\partial t}\left[e\frac{C_o}{B_o}\right] - \nabla.\left[l_o \nabla P\right] = q_o \tag{2}$$

where the subscripts *w* and *o* indicate water and oil phases, *P* represents pressure, *C* represents concentration (or saturations). ε denotes the medium porosity, *B* represents the formation factor, and *q* is the mass flow rate per unit volume. l represents the mobility of each phase, and is given as:

$$l_a = \frac{K.K_{ra}}{B_a m_a} \tag{3}$$

where l_a denotes the mobility for the α phase, *K* and K_{ra} are the medium permeability and relative permeability of the α phase, and μ_α is the absolute viscosity of the α phase. The value of K_{ra} is a function of the concentrations of oil and water, in this work, it is given as (Marcondes et al., 1994):

$$K_{rw} = C_w^2 \tag{4}$$
$$K_{ro} = C_o^2$$

Equations 1 and 2 form a set of two differential equations with three unknown variables (C_w, C_o and *P*). The closing equations is the conservation of the global mass:

$$C_w + C_o = 1 \tag{5}$$

Eqs. 1, 2 and 5 need to be solved in a coupled manner. There are two main methods used for solving this problem. The first one is the IMPES method (Implicit Pressure Explicit Saturation). In this method, eqs. 1 and 2 are combined by using eq. 5, yielding to an equation for pressure to be solved implicitly. Once the pressure values are determined, they are used in eq. 1 to determine (explicitly) water concentration and oil concentration is determined by using eq.4. The second method is to replace C_o (eq. 2) by 1 - C_w and solve eqs. 1 and 2 for pressure and water concentration in a simultaneous fashion using Newton's iteration. In this work, the IMPES method was used and it is described here in more details.

The next section describes the numerical method used for discretize the equations described above, and presents the sequence of operations required for the solution.

3. NUMERICAL METHOD

3.1 Finite Volume Method with Unstructured Meshes

In the last decade, the finite volume method in unstructured meshes has presented significant development, mainly in the discretization methodologies. The unstructured meshes are important to solve problems with complex geometry, such as moving boundary problems.

In spite of the large number of publications using finite volume method in unstructured meshes in the literature, the use of unstructured meshes with finite volume formulation can still be considered a recent technique, specially when compared to Finite Element methods. and continues to be a research area with a lot of activity.

The use of unstructured meshed has proven to be a valuable tool for reservoir simulations due to the geometry complexity of the problems. Accordingly, several works have been published in the specialized literature recently, among others Zang et al (2001) and Prévost et al. (2001).

There are two possibilities for the discrete location of the variables on unstructured meshes. These locations originate the finite volume discretization schemes called Cell-Centered (CC) and Vertex-Centered (VC). In the both schemes, the solution domain is divided into polygonal elements or cells (usually triangles for a 2D case and tetrahedra for a 3D case). In the VC scheme, the computational nodes are located at the vertices of the elements. The control volumes are formed around each node by joining the centroids of the elements and/or midpoints on element edges, as shown in figure 1a. By contrast, in the CC scheme, the variables are stored in the centroid of the control volumes.

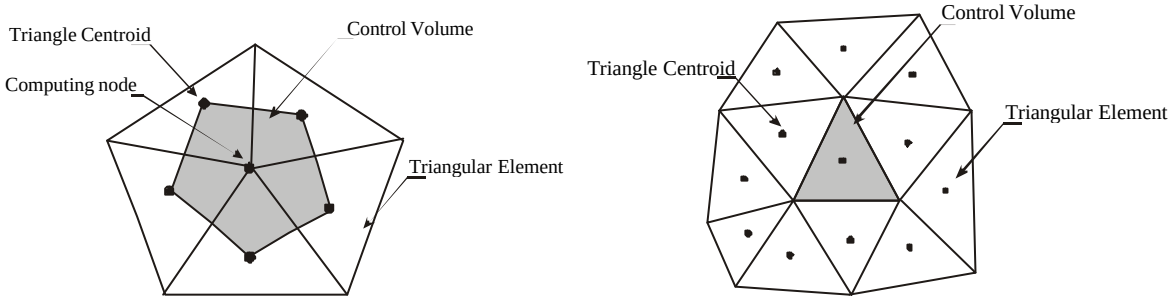


Figure 1 - Schematic representaion of the the finite volume method for unstructured grid using the (a) Vertex-Centered and (b) Cell-Centered schemes.

The VC and CC schemes are presented in equally large number of publications in the literature. The discussion regarding robustness, efficiency and performance of these schemes are still open. According to Martins et al. (2000) the simplicity in the discretization of conservation equations is the main feature that kept the researchers interest in the CC scheme. Another factor favoring this scheme is the direct use of the triangles as elementary control volumes, since in VC schemes the triangles are used indirectly; the possibility to use hybrid meshes, which are usually comprised of triangles and quadrilaterals, is the third factor that motivated the use of CC schemes.

Although the discussion regarding using either CC and VC schemes is still open, the CC scheme was selected for the use in the present work. The main reason for this choice is the ratio between the number of triangles and the number of vertices. In two-dimensional triangular meshes the ratio between the number of triangles and the number of vertices is two. However, for three-dimensional meshes composed by tetrahedra, this ratio can vary between 5 and 6. In CC schemes, the flow calculations are made by a sum over all tetrahedra faces, while in VC schemes, these same calculations require the sum over all surrounding vertex edges, which demands a considerably large computational effort. Although the effort for calculating the discretized equation for each control volume is smaller, the number of algebraic equations in the CC scheme is 2 times higher for 2D calculations (or 5 to 6 for 3D calculations), demanding larger amount of memory. Since the objective of this work is related

to distributed parallel computing, memory requirements are not a very serious issue in the computation, since the computational domain will be spread over several computing nodes (this procedure is described in detail in section 4). Furthermore, as will be discussed in the section of results (section 5), the speed-up provided by the use of additional computing nodes is superior for configurations with larger number of control volumes.

3.2 Finite Volume Discretization

As described in section 2, the IMPES technique is based on combining eqs. 1 and 2 by using eq. 5. In this procedure, the eqs. 1 and 2 are discretized and then combined, yielding to an equation for pressure to be solved implicitly. In this section, the dicretization procedure for eqs. 1 and 2 is presented, subsequently the final form of the pressure equation is derived, and the sequence of steps of the solution algorithm is presented. The equations are presented in a two-dimensional form, however, the formulations presented here can be easily extended to a three-dimensional form. As a first step, it is interesting to re-write eqs. 1 and 2 as:

$$\frac{\partial}{\partial t}\left[e\frac{C_a}{B_a} \right] - \nabla.\left[l_a \nabla P \right] = q_a \tag{6}$$

where the subscript α denotes the α-phase (water or oil). The main concept of a finite volume discretization consists of dividing the computational domain into control volumes or cells, and integrating the governing equations of a problem in each one of the control volumes. This integration over each control volume implies the conservation principle in the finite control volume, just as the differential equation expresses it for an infinitesimal control volume. Figure 2 presents a schematic representation of a generic control volume inside the calculation domain, where P denotes the nodal point in the center of the triangular control volume, whose vertices are A, B and C. The points 1,2 and 3 represent the centers of the neighboring control volumes.

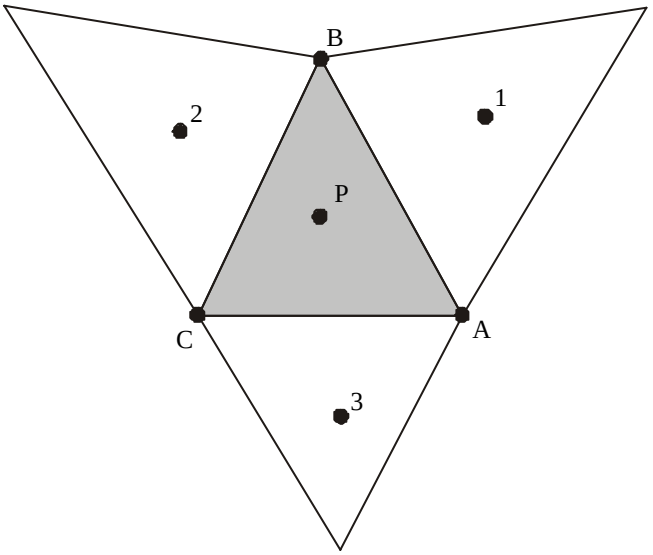


Figure 3 - Schematic representation of a generic control volume inside the calculation domain.

The discretized form of eq. 6 is obtained by integrating the general transport equation over the control volume (figure 2) and over the time interval from t to t + Δt:

$$\int_{t}^{t+\Delta t}\int_{\Omega}\frac{\partial}{\partial t}\left[e\frac{C_a}{B_a}\right]d\Omega dt-\int_{t}^{t+\Delta t}\int_{\Omega}\nabla.\left[l_a\nabla P\right]d\Omega dt=\int_{t}^{t+\Delta t}\int_{\Omega}q_a\,d\Omega dt \tag{7}$$

by using the Gauss Theorem, the second term in the left hand side can be replaced by a surface integral:

$$\int_{t}^{t+\Delta t}\int_{\Omega}\frac{\partial}{\partial t}\left[e\frac{C_a}{B_a}\right]d\Omega dt+\int_{t}^{t+\Delta t}\int_{S}\mathbf{J}.\mathbf{n}dSdt=\int_{t}^{t+\Delta t}\int_{\Omega}q_a\,d\Omega dt \tag{8}$$

where **n** is the outward normal vector to the surface of the control volume *S*, **J** is the flux vector given by:

$$\mathbf{J}=-l_a\frac{\partial P}{\partial x}\mathbf{i}-l_a\frac{\partial P}{\partial y}\mathbf{j} \tag{9}$$

where **i** and **j** are the components of the outward normal vector **n** in the *x* and *y* directions, respectively. By integrating eq. 8 explicitly over the time interval *t* to *t* + *Dt*, and considering the source term q_a constant over the control volume, it is possible to re-write eq.8 as:

$$\frac{e}{B_a}\left(C_a-C_a^0\right)+\Delta t\int_{S}\mathbf{J}.\mathbf{n}dS=q_a\Delta V\Delta t \tag{10}$$

or

$$\frac{e}{B_a}\frac{C_a-C_a^0}{\Delta t}+\int_{S}\mathbf{J}.\mathbf{n}dS=q_a\Delta V \tag{11}$$

where *DV* denotes the volume of the control volume (i.e. the area of the triangle in a two-dimensional coordinate system), and the superscripts 0 denotes a variable evaluated in the previous time interval.

The surface integral presented in eq. 11 describes the net flux over control volume (figure 2). This flux is calculated by the sum of the fluxes over each face of the control volume.

$$\frac{e}{B_a}\frac{C_a-C_a^0}{\Delta t}+\left(\sum_{k=1}^{3}\int_{S_k}\mathbf{J}.\mathbf{n}dS_k\right)=q_a\Delta V \tag{12}$$

where the index *k* represents the each face of the control volume P (*k* = 1, 2, 3), as depicted in figure 2, and S_k denotes the area of the face *k*.

The surface integral can be evaluated by using a mean flux $\bar{\mathbf{J}}$ over the control volume face:

$$\left(\sum_{k=1}^{3}\int_{S_k}\mathbf{J}.\mathbf{n}dS_k\right)=\sum_{k=1}^{3}\bar{\mathbf{J}}.\mathbf{n}S_k \tag{13}$$

In order to calculate this summation, it is necessary to determine the gradients in the *x* direction and *y* direction at each face of the control volume, since $\overline{\mathbf{J}}$ is given by:

$$\overline{\mathbf{J}} = -l_a \overline{\frac{\partial P}{\partial x}}\mathbf{i} - l_a \overline{\frac{\partial P}{\partial y}}\mathbf{j} \tag{14}$$

where the partial derivatives can be calculted in each face of the control volume by the Green theorem:

$$\overline{\frac{\partial P}{\partial x}} = \frac{1}{V}\int_A P\mathbf{n}_x dA \qquad \overline{\frac{\partial P}{\partial y}} = \frac{1}{V}\int_A P\mathbf{n}_y dA \tag{15}$$

where *V* is the area of the quadrilateral comprised of the two opposite centroids and the two vertices of each face, i.e. A1BP for face 1, B2CP for face 2 and C3AP for face 3 (Figure 2). In order to evaluate the integrals above it is necessary to know not only the value of *P* in the center the control volumes, but also in the vertices of the control volume. The vertex values are determined by interpolating the pressure at the neighboring control volumes using the inverse of the square of the distance between the center of the neighboring control volume and the vertex.

After the evaluation of $\overline{\mathbf{J}}$ at each face, based on the integrals above, eq. 13 can be re-written as:

$$\sum_{k=1}^{3} \overline{\mathbf{J}}.\mathbf{n}S_k = A_p P_p - A_1 P_1 - A_2 P_2 - A_3 P_3 - S^t \tag{16}$$

where

$$A_1 = l_a^1 \frac{\left|\dot{AB}\right|^2}{V_{A1BP}} \qquad A_2 = l_a^2 \frac{\left|\dot{BC}\right|^2}{V_{B2CP}} \qquad A_3 = l_a^3 \frac{\left|\dot{CA}\right|^2}{V_{C3AP}} \tag{17}$$

$$A_p = A_1 + A_2 + A_3$$

$$S^t = l_a^1 \frac{1\dot{P}.\dot{AB}}{V_{A1BP}}(P_A - P_B) + l_a^2 \frac{2\dot{P}.\dot{BC}}{V_{B2CP}}(P_B - P_C) + l_a^3 \frac{3\dot{P}.\dot{CA}}{V_{C3AP}}(P_C - P_A)$$

where l_a^1, l_a^2 and l_a^3 represent the mobility of the phase α, calculated at the faces 1, 2 and 3 of the control volume, respectively. The term S^t is due to the non-orthogonality of the mesh. If an orthogonal mesh is used, S^t vanishes. By using eq. 16, it is possible to re-write eq. 12 as:

$$\frac{e}{B_a}.\frac{C_a - C_a^0}{\Delta t} = -A_p P_p + A_1 P_1 + A_2 P_2 + A_3 P_3 + S^t + q_a.\Delta V \tag{18}$$

which represents the discretized equation for calculating the concentration of the component α. As discussed previous, the expression above can be used to write equations for

concentration of water and oil, which can be combined together via eq.5, yielding a discretized equation for pressure:

$$A_p P_p = A_1 P_1 + A_2 P_2 + A_3 P_3 + S^t + b \tag{19}$$

where

$$A_1 = \Gamma^1 \frac{\left|\overrightarrow{AB}\right|^2}{V_{A1BP}} \qquad A_2 = \Gamma^2 \frac{\left|\overrightarrow{BC}\right|^2}{V_{B2CP}} \qquad A_3 = \Gamma^3 \frac{\left|\overrightarrow{CA}\right|^2}{V_{C3AP}} \tag{20}$$

$$A_p = A_1 + A_2 + A_3$$

$$S^t = \Gamma^1 \frac{1\dot{P}.A\dot{B}}{V_{A1BP}}\left(P_A - P_B\right) + \Gamma^2 \frac{2\dot{P}.B\dot{C}}{V_{B2CP}}\left(P_B - P_C\right) + \Gamma^3 \frac{3\dot{P}.C\dot{A}}{V_{C3AP}}\left(P_C - P_A\right)$$

$$b = \Delta V\left[\frac{q_w}{e/B_w} + \frac{q_o}{e/B_w}\right]$$

where Γ^1, Γ^2 and Γ^3 represent the combined mobility of the phases, calculated at the faces 1, 2 and 3 of the control volume, respectively, which are given as:

$$\Gamma^1 = \frac{l_w^1}{e/B_w} + \frac{l_o^1}{e/B_o} \tag{21}$$

$$\Gamma^2 = \frac{l_w^2}{e/B_w} + \frac{l_o^2}{e/B_o}$$

$$\Gamma^3 = \frac{l_w^3}{e/B_w} + \frac{l_o^3}{e/B_o}$$

It is important to emphasize that the values of concentration of water and oil that are required for the calculations of Γ^1, Γ^2 and Γ^3, are stored in the center of the control volume, while the values of Γ are required calculated at the faces of the control volume. Thus, some kind of interpolation should be use to obtain Γ at the faces of the control volume. An obvious choice would be a piecewise linear interpolation from the control volume centers to the faces. In fact, this would be an excellent choice for a diffusion problem. However, although the governing equations resemble those of a pure diffusion problem (i.e. a Poison-like equation), the physical phenomenon is a convection dominated problem. Therefore, numerical methods for dealing with this class of equations may be subjected to wiggles or oscillations of the solutions, which are related to the use of higher order schemes in convection/diffusion problems. As such, numerical schemes usually require some form Upwinding or Flux Limiter technique to ensure an oscillation free solution. Thus, in this work, upwinding is applied to obtain Γ at the faces of the control volume for numerical stability (Lu et al., 2001).

The boundary conditions at the reservoir frontier are of flux equal to zero, since the reservoir boundaries are considered to be impermeable.

The non-orthogonality term (S^t) is treat explicitly, since it depends on the pressure values at the control volumes vertices, which are interpolated from values at the control volumes centers. Therefore, an iterative procedure is necessary to solve the pressure equation.

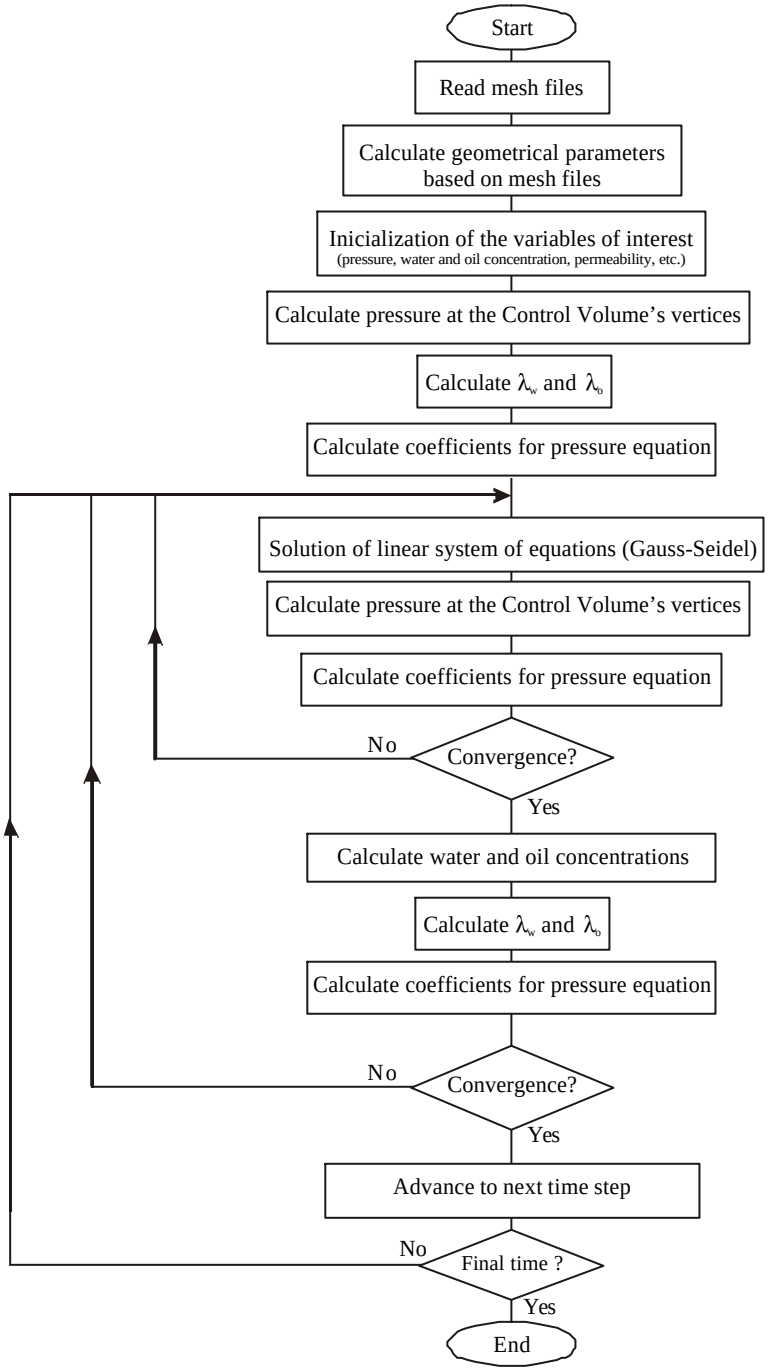


Figure 3 - Schematic representation of the sequence of operations involved solution algorithm.

According to the IMPES method, given the initial concentrations of water and oil and the initial pressure inside the calculation domain, eq. 19 is solved in order to determine the pressure field. The present work uses the Gauss-Seidel method to solve the linear system of

equations for the pressure field. Once pressure values are calculated eq. 18 is used to determine water concentration and eq.5 to determine oil concentration. Once concentrations of oil and water are calculated, new values of Γ^1, Γ^2 and Γ^3 must be re-calculated, and eq. 19 solved again, in order to determine a new pressure field, which will be used to calculate new concentrations of oil and water, and so on. This iterative process must continue until the values of water and oil concentrations, as well as, pressure no longer present a significant change from one iteration to the other.

Figure 3 presents a schematic representation of the sequence of operations involved in the procedure. There are four major loops in the algorithm. The first one, more internal, is the loop of the solution of the linear system of equations (Gauss-Seidel). The second loop performs the iterations due to the explicit treatment of the non-orthogonality terms of the pressure equation. The third is the IMPES procedure. And the outer loop (the fourth) is the time step advancing.

4. PARALLELIZATION STRATEGY

Before describing the parallelization strategy adopted for the solution algorithm, it is convenient to outline the main differences between the parallelization for shared-memory computers and for distributed parallel computers. Figure 4 presents as schematic representation of a shared memory architecture and a distributed memory architecture. Shared memory machines have several processors, which can access the memory through a high-speed connection bus. In this type of architecture any process can access any memory location with an equally fast access time. On the other hand, each processor of a distributed memory has its own memory, and can only access a memory position located through a connection bus. Therefore, the performance of distributed memory systems may be directly related to the data speed on the connection bus, which can be considerably large for massively parallel computers (such as Cray T3E and T3D) or relatively slow for clusters of workstations, where the connecting bus is the local network. Moreover, since the local memories of each processor are not directly linked, the programmer should orchestrate the message-passing between the processors. This requires considerable changes in the programming paradigm for distributed memory systems.

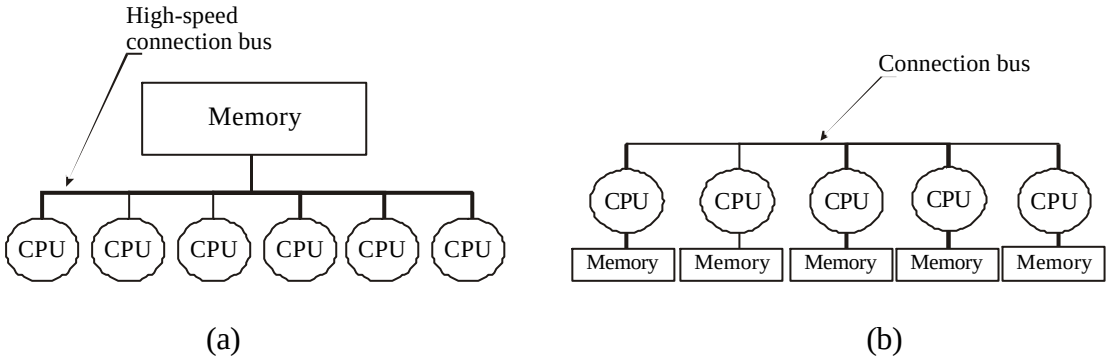
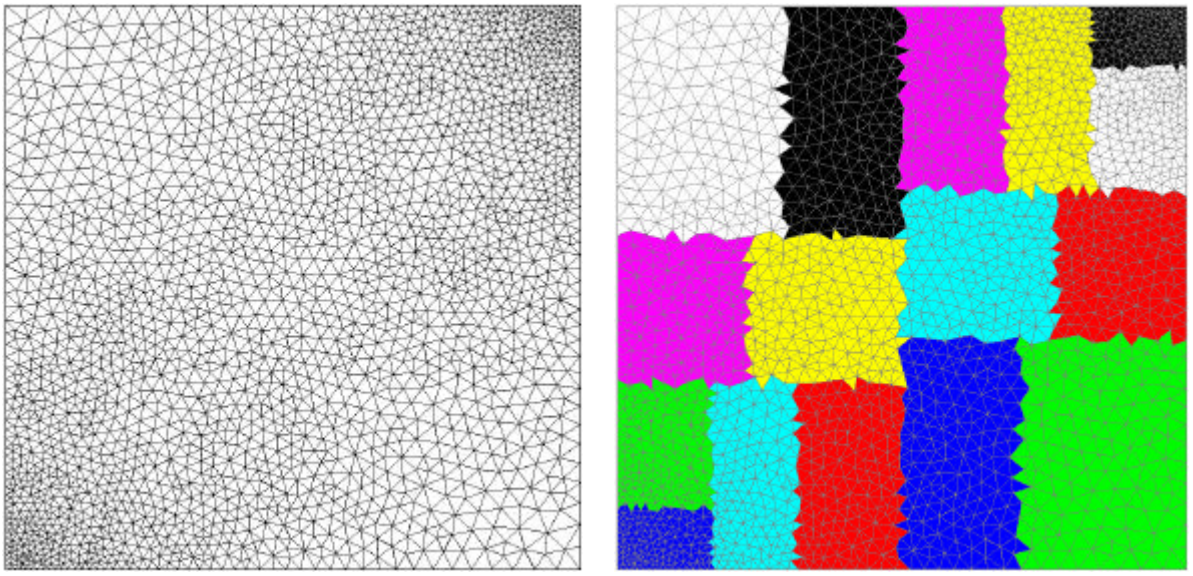


Figure 4 - Schematic representation of (a) a shared memory architecture and (b) a distributed memory architecture

While for shared-memory, parallelism is mainly directed to executing identical set of operations in parallel on the same data structure (Do-loop parallelization), parallelism in distributed memory systems is mainly directed to sub-dividing the data structures into sub-domains and assign each sub-domain to one processor. In this case, the same code runs on all processors, on its own set of data. Figure 5 shows an unstructured grid containing 4000

control volumes, by dividing the computational domain into sixteen sub-domains it is possible to spread the work load between sixteen different processor. However, it is important to note that in order compute the variables for each control volume, the variables at its neighboring points are required. Thus, in order to calculate the variables at the control volumes close to the interface between sub-domains, one processor will require information stored in the memory of another processor. This requires some amount of communication at regular intervals, which may slow down the computation.

In general, the computation procedure involves three steps (1) partitioning of the solution domain; (2) performing computations on each processor to update its own data set; (3) communicating data between processors. This technique is called domain decomposition. The key for an efficient computation is to maintain the communications between processors to a minimum level, as well as, to divide the workload equally between processors.



(a) (b)

Figure 5 - (a) Unstructured mesh containing 4000 triangular control volumes and (b) its decomposition in 16 sub-domains.

In this work a domain decomposition coordinate bisection is used (Streng, 1996). This method divides the number of points equally between processors, but makes no attempt to obtain a domain division that minimize communications between processors, i.e., a division with the smallest number of control volumes in boundaries between sub-domains. In general, coordinate bisection produces sub-domains with long interfaces, so that they lead to large communication volumes. This can be partly overcome by recursive application of alternatively *x*, *y* (and in 3D, *z*) bisection. The grid is first divided into 2 grids using bisection of the *x*-length of the calculation domain. Then to each of the resulting domains, *y*-bisection is applied, resulting in four blocks (or sub-domains). The procedure can be continued to obtain eight, sixteen, thirty two, … blocks.

Once a multi-block domain has been established, calculations on each block can begin in parallel if the boundary conditions of the block are known. This may be either a physical boundary condition (reservoir boundary) or an internal boundary as a consequence of the domain decomposition. The physical boundary conditions are managed by the source code; while the internal boundary requires boundary data from its neighbor, which may reside on a different processor. These data are provided by allowing a buffer on the boundary of each

block, which will store a copy of the corresponding overlap data. Figure 6 illustrates a calculation sub-domain and the buffer cells used to store the overlap data.

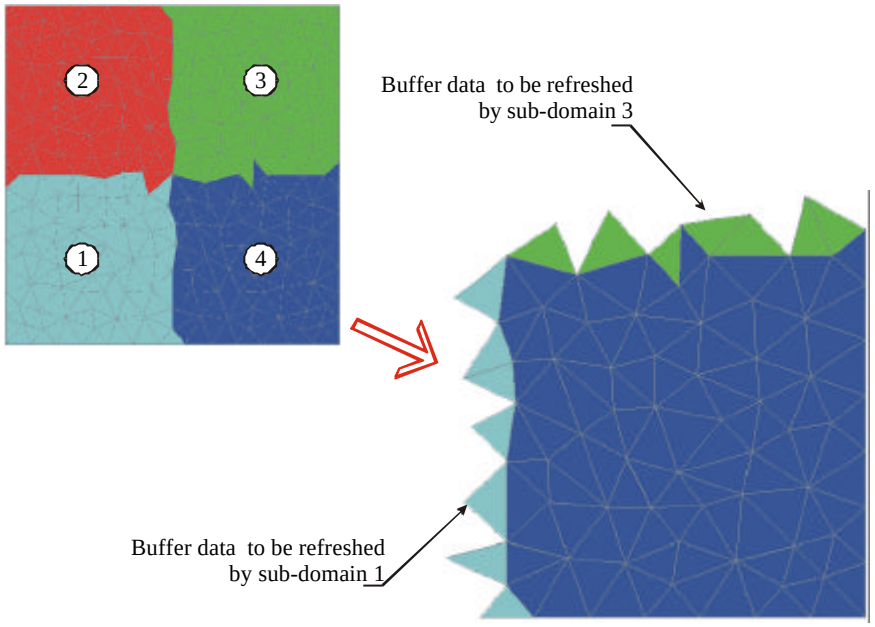


Figure 6 - Schematic representation of a calculation sub-domain divided into four sub-domains. Indicating the buffer cells used to store the overlap data

Once the buffer data has been received from all sides of the block, the computation of the block can commence, using the sequential algorithm. On completion of the solution for the block, the data at the boundaries of the current block is sent to the neighboring blocks. The current block then awaits for its buffer to be refreshed by its neighboring blocks, so that the next computation can commence.

In the present work, each block deals individually with the solution of the pressure equation, solving the linear system of equations and resolving the iterations required for the explicit treatment of the non-orthogonal terms on each block. Once pressure values are calculated for each block, the pressure values at each block boundary is communicated to its neighbors. Water and oil concentrations are then calculated on each sub-domain, and their values at each block boundary communicated to its neighbors. This iterative process is continued until the process reaches convergence. The exchange of information across the sub-domains, or processors, is achieved using the message passing interface standard, MPI. The use of MPI ensures portability across different computing platforms ranging from massively parallel machines to clusters of workstations. Figure 7 illustrates the sequence of operations involved in the computation.

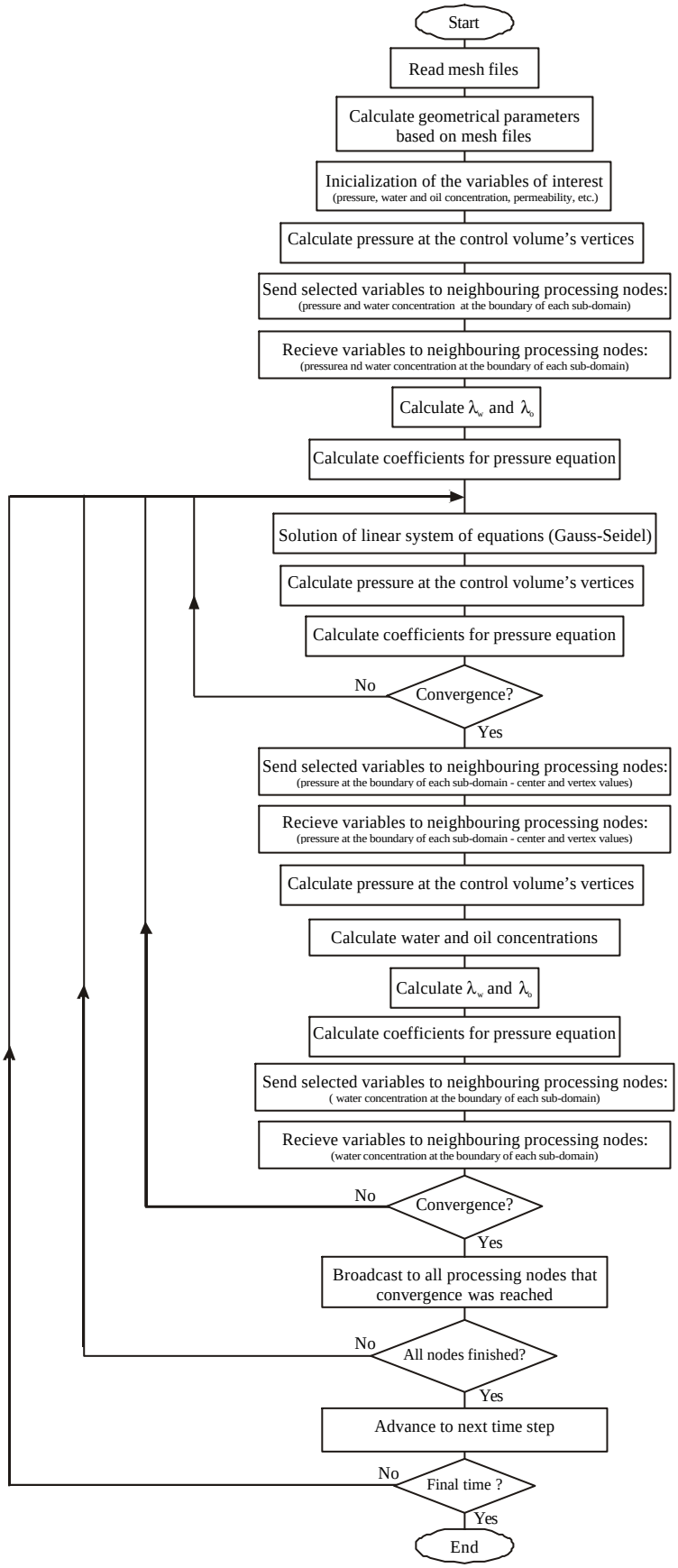


Figure 7 - Sequence of operations executed by the solution algorithm on each processor.

5. RESULTS AND DISCUSSION

This section is divided into two main parts. In the first part, the physical output of the solution algorithm is evaluated and its accuracy is assessed, by comparing the results obtained with previous work available in the literature. In the second part, the aspects of parallelization are explored, and the speed-up obtained by the use of additional processors is analyzed.

5.1 Validation and Verification

To verify the computational accuracy of the proposed algorithm the well-known five-spot problem is simulated. The problem consists of four injection wells symmetrically disposed around one production well (figure 8). Due to the symmetry of the configuration only 1/4 of the problem is simulated. The geometrical configuration and the physical data were taken from Mota & Maliska (1994). The water is injected at a rate of $2{,}648 \times 10^{-3}$ m^3/s in an injection well located at the lower corner of a square reservoir, whose sides measure 402,33 m. The depth of the reservoir is 6,09 m. The porous medium is considered to be homogeneous and isotropic, with porosity equal to 0,2 and permeability equal to 0,012337 μm^2. Water viscosity is equal 0,1 mPa.s and oil viscosity equal to 1,0 mPa.s.

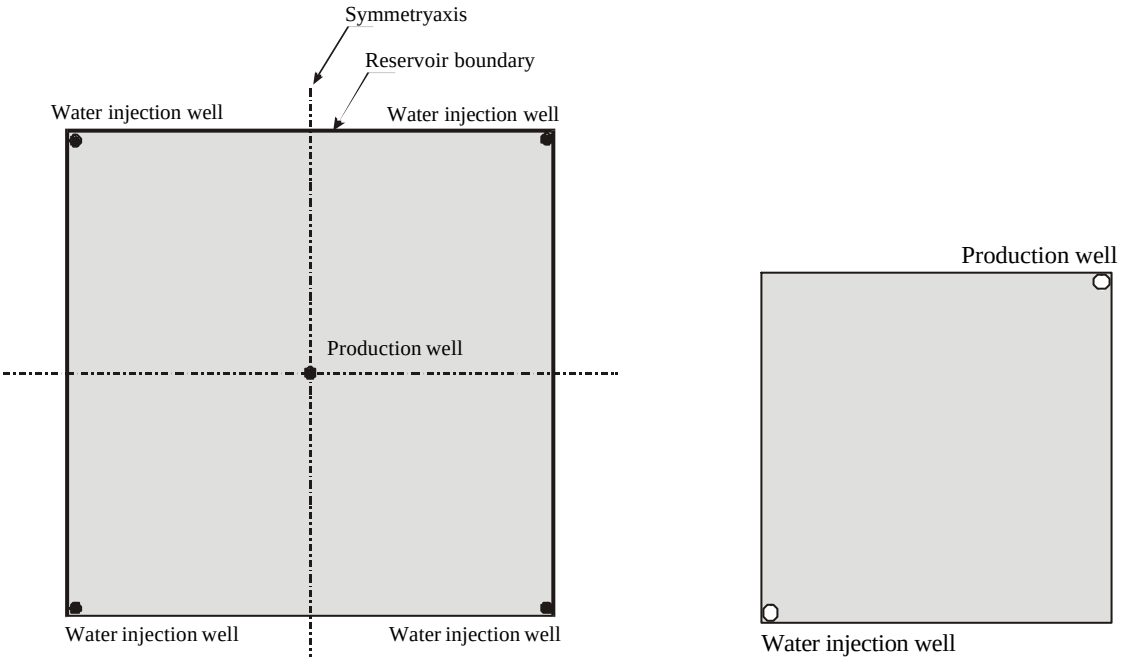


Figure 8 - Schematic representation of the five-spot problem (a), and the configuration simulated five-spot problem.

Although the IMPES method treats pressure equation implicitly, concentrations are treated explicitly. Thus, CFL stability criterion must be satisfied, as such, the time step sizes for each simulations are set based on the condition of CFL number < 1. The results are compared with the data published by Mota & Maliska (1994), which presents numerical results of simulations using a TVD scheme (mesh 20 x 20 control volumes) and upwind scheme (mesh 30 x 40 control volumes) with structured grid and body-fitted coordinates. To

avoid repetition from this point onwards the simulation using TVD scheme with a mesh of 20 x 20 is referred to as RUN A and the simulation using upwind scheme with a mesh of 30 x 40 is referred to as RUN A.

The grid size used is 1020 control volumes, with a control volume size comparable to the mesh 30 x 40 used with the upwind scheme presented by Mota & Maliska (1994). Figure 9 presents the evolution of the water concentration inside the reservoir. The time evolution is presented as a function of the porous volume of the reservoir injected by water (PVI). A significantly sharp front is observed while the water displaces the oil, until reaching the production well.

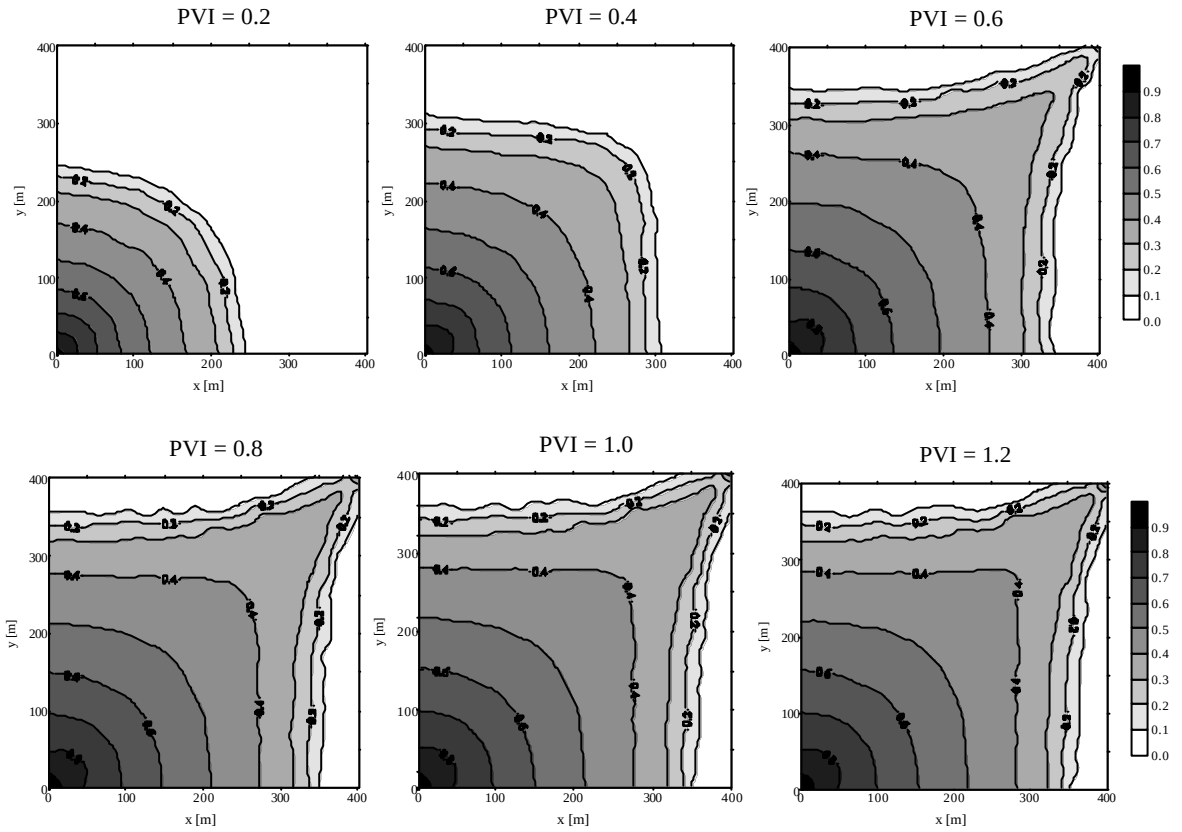


Figure 9 - Time evolution of the water concentration inside the reservoir.

Figure 10 compares the obtained results for water concentration at the production well for the simulation carried out in the present work and the results obtained by Mota & Maliska (1994). The discrepancies between curves are very small, thus, the solutions obtained are very similar. It is interesting to note that although the present simulation uses the same discretization scheme and a roughly equivalent mesh size as RUN B, the results seems to match more closely results obtained by RUN A, which uses the TVD and, a priori, produces better results than upwind schemes. This is probably related to mesh refinement used in the present work close to the production and injection wells.

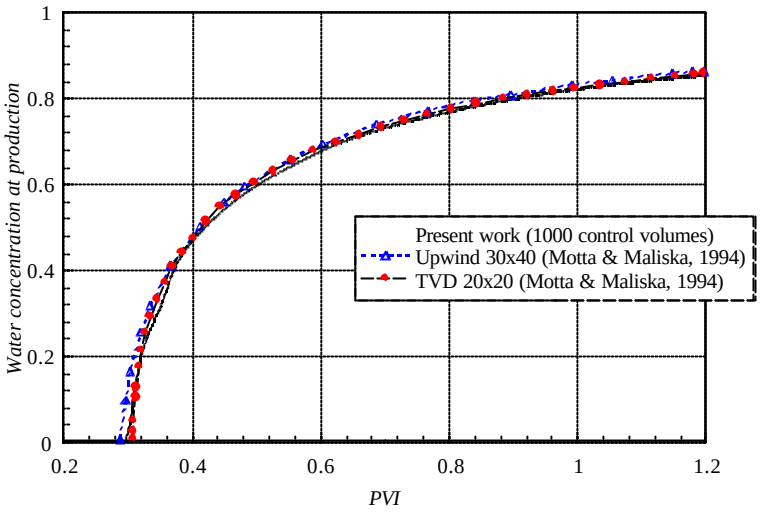


Figure 10 - Comparison of the results obtained for water concentration at the production well for the simulation carried out in the present work and the results obtained by Mota & Maliska (1994).

5.1 Speed-up results and discussion

The experiments were run on a cluster of 16 Pentium III (650 MHz, 256k cache L2, 512 Mb RAM), using Giganet 32-bits 33 MHz cLAN NIC and a 30 port Giganet switch, which presents a latency for 4 bytes equal to 8.2 μs and a bandwidth for 32 Kbytes of 101 Mbytes/s. The five-spot problem described in the previous sections was simulated using three different mesh sizes, containing 1000, 4000 and 16.000 control volumes. These meshes will be referred to as 1KCV, 4KCV and 16KCV respectively, from this point onwards. The simulations were run using 1, 2, 4, 8 and 16 processors. It is important to emphasize that the simulation for 1 processor is, in fact, slightly different from those executed on various processors, since a truly sequential code was utilized. So that, it was possible to evaluate the real performance gain in using parallel instead of sequential computing. Figure 11 presents the speed-up obtained during the experiments.

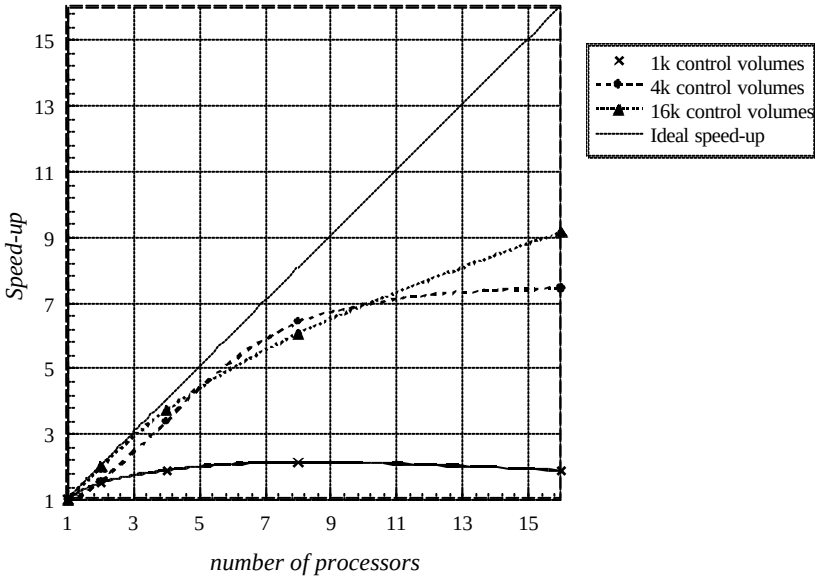


Figure 11 - Speed-up obtained for mesh sizes, containing 1000, 4000 and 16.000 control volumes, with simulations running on 1, 2, 4, 8 and 16 processors.

Figure 11 compares the obtained results with the ideal speed-up, which would represent a linear reduction of the computation time as the number of processors increases. It is possible to note that the speed-up obtained for the 1KCV mesh is far from the ideal speed-up. When two processors are used there is a considerable gain in performance, but not a linear gain. Although the workload is divided among processor, the control volumes of each computation block that require communication represent a large proportion of the total number of control volumes in each block. Since the speed of the data in the network connecting the processors is limited, each processor spends a large amount of time waiting for information form the other. This fact is even more noticeable when the number of processors is increased, and the proportion of control volumes of each computation block requiring communication becomes even larger. It can be seen that computing time for 16 processors is larger than that for 8 processors.

When the number of control volumes in the calculation domain increases, there is a considerable gain in performance for the parallel runs in relation to the run using only one processor. For the 4KCV mesh, the number of control volumes of each computation block that require communication represent a smaller proportion of the total number of control in each block, when compared to the 1KCV mesh. Although the speed-up obtained for the 4KCV mesh is considerably better, it is still far from the ideal speed-up, especially when the number of processors increases. On the other hand, for the 16KCV mesh the speed up for simulations using 2 and 4 processors is nearly linear, and very close to the ideal. However, as the number of processors increases the speed-up obtained starts to reduce, which illustrates that even for a mesh size of 16.000 the proportion of control volumes requiring communication is considerable for 16 sub-domains.

One interesting feature of the speed-up curves obtained is that the speed-up for the grid using 4000 control volumes was slightly superior to that obtained for the 16KCV mesh. This is probably related to a more efficient use of the processor cache. Although the proportion of control volumes requiring communication of the 4KCV mesh is larger than for that of 16KCV, the memory size required for the computation was considerably smaller. Thus, dividing the 4KCV mesh into 8 blocks, creates a computation with memory occupation small enough to nearly fit inside the processor L2 cache. Since the access time for cache memory is 5 to 10 times faster that for conventional memory, each processor performs the computation on its own sub-domain slightly faster, reducing the total computing time. However, as the domain is further divided the increase of performance provided by the more efficient use of the processor's cache is not sufficient avoid the considerable performance degradation.

It is important to remember that although the speed-up provided is not linear, the performance gains are considerable. The complete simulation of the problem using 16.000 control volumes took 9 hours and 47 minutes using a single processor, while the simulation using 16 processors took only 1 hour and 4 minutes. Furthermore, the mesh sizes used here represent only a small fraction of those used in a real scale field simulation, where computational meshes can exceed 1 million grid cells.

6. CONCLUSION

A numerical algorithm for simulating petroleum reservoirs, based on finite volume technique using non-structured grids, was presented. The algorithm was designed for parallel computing platforms with distributed-memory. The parallel implementation was based on the domain decomposition, with the original reservoir decomposed into several domains, each of which given to a different processing node. All nodes then execute computations in parallel,

each node on its associated sub-domain. The exchange of information across the sub-domains, or processors, is achieved using the message passing interface standard.

In order to validate the solution procedure, the well-known five-spot problem is simulated and the results compared with data from Mota & Maliska (1994) obtained from numerical simulations using TVD and Upwinding schemes. The results show a good agreement with previous numerical simulations.

The execution time and speed-up are evaluated through comparing the performance of different numbers of processors. The results indicate that the parallel code can significantly improve prediction capability and efficiency for large-scale simulations. However, there is a considerable degradation of the speed-ups obtained with the increase of the proportion between the control volumes that require communication and the total number of control volumes in each block. Thus, for simulation with a small number of grid points, the speed-up provided by the execution using parallel processors is far from linear. Nevertheless, mesh sizes used in real scale field simulation can exceed 1 million grid cells, so that they can really take advantage of large number of processors.

Although this work has presented good levels of performance for parallel computations, there is still some ground for improvement. Further work should be carried out to include higher order convection/diffusion schemes, such as higher order Godunov (Dicks, 1993) or TVD schemes. The use of more modern methods for the solution of the linear system of equations, such as GMRES, is also desirable. In addition, more attention should me devoted to utilize more efficient domain decomposition techniques, since the coordinate bisection only divides the number of points equally between processors, but makes no attempt to obtain a domain division that minimize communications between processor. This can be improved by the use of an inertial or spectral bisection methods (Streng, 1996).

Another possible ground for improvement is to attempt to move the communication between processors from inside the IMPES Do-loop to outside. In this manner instead of calculating pressure at each sub-domain and then updating the data between processors, the pressure field inside a sub-domain would be used for calculating water and oil concentrations, only then updating the data between processors. This would, theoretically, reduce the frequency of communication between processors. Moreover, processor cache influence should be further investigated by using more advanced metrics (Meira et al., 1996).

Acknowledgements

The authors would like to acknowledge Prof. C. L. Amorim, of the group of High-Performance Computing from COPPE/UFRJ, for granting access to the high-performance computing facilities used during the course of this work.

REFERENCES

Allen, M. B. & Furtado, F., 1997, Computational Methods for Porous Media Flows, in Fluid Transport in Porous Media, edited by J. P. du Plessis, Computational Mechanics Publ., Southampton.

Chang, Y. C. & Mohanty, K. K., 1997, Scale-up of two-phase flow in heterogeneous porous media, Journal of Petroleum Science and Engineering, 18, pp. 21-34.

Christie, M. A., 2001, Flow in porous media - scale up of multiphase flow, Current Opinion in Colloid & Interface Science, 6, pp. 236-241.

Dicks, E.M., 1993, Higher Order Godunov Black-oil Simulations for Compressible Flow in Porous Media, PhD Thesis submitted to University of Reading, Reading, UK

Prévost, M., edwards, M.G., Blunt, M.J., 2001, Streamline Tracing on Curvilinear Structured and Unstructures Grids, Paper SPE 66347, 2001 SPE Reservoir Simulation Symposium.

Lu, Q., Peszynska M. & Wheeler F., 2001, A Parallel Multi-Block Black-Oil Model in Multi-Model Implementation, Paper SPE 66359, 2001 SPE Reservoir Simulation Symposium.

Martins, M.A., Valle, R.M., Oliveira, L.S., 2000, Fundamental of Finite Volume Discretization of the Conservation Equations using Structuredand Unstructured Meshes, 21st. Iberian Latin-American Congress on Computational Methods in Engineering, Rio de Janeiro.

Meira, W.Jr., LeBlanc, T., Poulos, A., 1996, Waiting time analysis and performance visualization in carnival, In ACM SIGMETRICS Symposium on Parallel and Distributed Tools.

Mota, M.A.A. & Maliska, C.R., Simulação Numérica de reservatórios de Petróleo Utilizando Coordenadas Generalizadas e Interpolação TVD, Procedings of V Encontro Nacional de ciências Térmicas, pp. 325-328, São Paulo.

Streng, M., 1996, Load Balancing for Computational Fluid Dynamics Calculations, in High Performance Computing Fluid Dynamics, ed. P. Wesseling, Kluwer Academic Publishers.

Thiele, M.R., Batycky, R.P., Blunt, M. J., 1997, A Streamline-Based 3D Field-Scale Compositional Reservoir Simulator, SPE Journal, SPE 38889.

Zang, K., Wu, Y.S. Ding, C., Pruess, K. & Elmroth, E., 2001, Parallel Computing Techniques for Large-Scale Reservoir Simulation of Multi-Component and Multiphase Fluid Flow, Paper SPE 66343, 2001 SPE Reservoir Simulation Symposium.