REVISTA INTERDISCIPLINAR DE PESQUISA EM ENGENHARIA

CILAMCE 2016

XXXVII IBERIAN LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING
BRASÍLIA - DF - BRAZIL

An OpenMP Parallel Implementation for Numerical Simulation of Gas Reservoirs Using Intel Xeon Phi Coprocessor

Leonardo Figueira Werneck

Mayksoel Medeiros de Freitas

Hilton Guaraldi da Silva Júnior

Grazione de Souza

Helio Pedro Amaral Souto

lwerneck@iprj.uerj.br

mayksoel@iprj.uerj.br

guaraldi@iprj.uerj.br

gsouza@iprj.uerj.br

helio@iprj.uerj.br

Instituto Politécnico, Universidade do Estado do Rio de Janeiro

Rua Bonfim 25, Vila Amélia, 28625-570, Nova Friburgo, Rio de Janeiro, Brazil

Abstract. *The objective of this work is to parallelize, using the Application Programming Interface (API) OpenMP (Open Multi-Processing) and Intel Xeon Phi coprocessor based on Intel Many Integrated Core (MIC) architecture, the numerical method used to solve the algebraic system resulting from the discretization of the differential partial equation that describes the single-phase flow in a gas reservoir. The set of governing equations are the continuity equation, the Darcy's law and an equation of state. The Hydraulic Diffusivity Equation (HDE), for the unknown pressure, is obtained from this set of fundamental equations and it is discretized by means of the Finite Difference Method (FDM) along with a time implicit formulation. Different numerical tests are performed in order to study the computational efficiency of the parallelized versions of Conjugate Gradient (CG), BiConjugate Gradient (BiCG) and BiConjugate Gradient Stabilized (BiCGStab) methods, and different production scenarios were considered for*

horizontal wells and single-phase gas flow. The influence of different physical parameters as, for example, permeability on the wellbore pressure is also considered. Speed-up results are considered in order to evaluate the performance of the parallel algorithms.

Keywords: *Numerical reservoir simulation, Finite difference method, Sparse algebraic system, Iterative methods, OpenMP, Parallelization.*

1 INTRODUCTION

Without a doubt, numerical reservoir simulation has became very important for petroleum industry and has been used intensely by the scientific and engineering communities. Reservoir simulation combines physics, mathematics, reservoir engineering and computer programming to develop a tool for predicting hydrocarbon-reservoir performance under various operating conditions (Ertekin et al., 2001). Dumkwu et al. (2012) pointed out that a numerical reservoir simulator has long become a primary reservoir tool used by reservoir engineers to conduct reservoir simulations which are useful in order to take major decisions, estimate reserves and to diagnose and improve on the performance of producing reservoirs in the oil and gas industry. The main purpose of the reservoir simulators is to provide better reservoir management, leading to an increase in gas/oil recovery.

Aiming to enable or increase the production of unconventional reservoirs, horizontal wells (Fig. 1) have been used to increase the contact area between reservoir and well (Akgun, 2004). They are indicated for low reservoir thickness, low-permeability formations and naturally fractured reservoirs. Productivity forecasts for horizontal wells have a higher level of uncertainty compared to vertical wells. This is due to the fact that reservoir flow regimes in a horizontal well are not so clearly identified as those for a vertical well (Chaudhry, 2003). This partially results from the inherently three-dimensional nature of the flow (the radial symmetry observed in vertical wells does not occur in the same way for horizontal wells). Therefore, due to its importance for reservoir engineering applications, much research has been done considering horizontal wells (Al-Mohannadi, 2004; Al-Mohannadi et al., 2007; de Souza, 2013; de Souza et al., 2014; Shahbazi et al., 2015). It is important to note that in the last decades many works have been produced aimed at optimizing recovery in unconventional hydrocarbons reserves, as for example, shale gas. Indeed, shale gas reservoirs are increasing in importance as a source of natural gas.

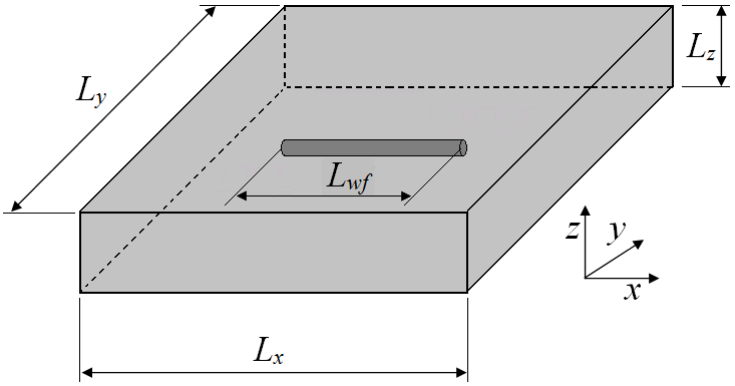

Figure 1: Representation of a reservoir and a horizontal well, of length L_{wf}, parallel to x- direction.

For a certain class of problems, as for example, slightly compressible one-dimensional single phase flow, there are some analytical solutions available in the literature. However, for complex systems and more realistic flows, numerical simulation is still a very important tool (Ertekin et al., 2001). For instance, numerical solutions are used for modeling gas reservoirs. In this case, governing partial differential equations for porous media flow are nonlinear. This follow from the fluid properties dependency on pressure (Jmili et al., 2011; He & Durlofsky, 2013).

The motivation to develop this work is due to the importance of gas recovery in reservoirs in the last decades. In order to study this problem, a three-dimensional reservoir simulator has been developed using Cartesian coordinates and a classical finite-difference approach. The physical-mathematical modeling considers single-phase and isothermal gas flow, without adsorption effects, which can be present in certain applications involving gas reservoirs. Hydraulic and natural fractures are not also considered in this study. The production through horizontal wells is considered in this work. A nonlinear partial differential equation for unknown pressure is numerically solved using parallelized versions of the Conjugate Gradient, BiConjugate Gradient and BiConjugate Gradient Stabilized methods. It is worth noting that the presence of rock heterogeneities and/or nonlinear equations, as in gas flow, generally will increase the computational costs. Moreover, for simulations involving horizontal wells typically it is necessary a more refined mesh and this contributes to increase computational efforts. Hence, high performance computation for numerical reservoir simulation is a subject of great interest.

Nowadays, the technologies used to characterize petroleum reservoirs can produce a large amount of data, so that an accurate simulation requires the use of highly refined computational grids. The direct consequence is that the calculations become significantly slower and a very high amount of computational resources (CPU and memory) is necessary. Currently, fast simulations using business software are based on parallel computing on CPU cores using MPI and OpenMP, which also motivated this work.

2 PHYSICAL-MATHEMATICAL MODELING

A governing equation for porous media flow will be derived using the balance equations (mass and momentum) and an equation of state, considering the following assumptions: heterogeneous rock formation; constant intrinsic permeabilities; low and constant rock compressibility; Newtonian fluid; dry gas reservoir; fluid of constant composition and isothermal single-phase flow. Typically, porous media flow is governed by the well-known Darcy's law, that establishes a linear relationship between the superficial velocity and the pressure gradient.

2.1 Governing Equations for Porous Media Flow

Mass conservation for porous media flow can be expressed by

$$\frac{\partial}{\partial t}\left(\phi\rho\right) + \nabla\cdot\left(\rho\mathbf{v}\right) - \frac{q_m}{V_b} = 0, \tag{1}$$

where $\mathbf{v}$ is the superficial fluid velocity, ρ represents density, q_m is a source term and V_b indicates the bulk volume.

For a real gas, the following equation of state can be considered

$$\rho = \frac{pM}{ZRT}, \tag{2}$$

where M is the molecular gas weight, Z represents real gas deviation factor, R is the universal gas constant and T is the temperature. Using the formation-value-factor, B, Eq. (2) can be rewritten as

$$B = \frac{V}{V_{sc}} = \frac{\rho_{sc}}{\rho} = \frac{p_{sc}M}{Z_{sc}RT_{sc}}\frac{ZRT}{pM} = \frac{p_{sc}TZ}{T_{sc}p}, \tag{3}$$

where V is fluid volume at reservoir conditions, V_{sc} is fluid volume at standard conditions and p_{sc} and T_{sc} are pressure and temperature at standard conditions, respectively.

The superficial velocity is determined by the traditional Darcy's law

$$\mathbf{v} = -\frac{\mathbf{k}}{\mu}[\nabla p - \rho g \nabla D], \tag{4}$$

where $\mathbf{k}$ is the absolute permeability tensor, μ the fluid viscosity, p the pressure, ∇D is the depth gradient and g is the gravity acceleration. For gas flow, viscosity must be modeled as a function of pressure, temperature and molecular weight. Equation (4) can be applied to model laminar flow through a porous medium at low Reynolds number. When the inertial effects are no longer negligible, classical Darcy's law can lead us to results that do not correspond to the physical reality and the law has to be modified in order to include the inertial effects (Barree & Conway, 2004).

2.2 Non-Linear Hydraulic Diffusivity Equation

In order to obtain a differential partial equation for the unknown pressure, we begin by replacing Eq. (4) in Eq. (1),

$$\frac{\partial}{\partial t}(\phi\rho) - \nabla \cdot \left[\frac{\rho\mathbf{k}}{\mu}(\nabla p - \rho g \nabla D)\right] - \frac{q_m}{V_b} = 0. \tag{5}$$

Now, applying Eq. (3) in Eq. (5), multiplying by V_b and using $q_m = q_{sc}\rho_{sc}$, it is possible to write

$$V_b\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right) - V_b\nabla \cdot \left[\frac{\mathbf{k}}{B\mu}(\nabla p - \rho g \nabla D)\right] - q_{sc} = 0. \tag{6}$$

For porosity variation we consider that (Ertekin et al., 2001)

$$\phi = \phi^0\left[1 + c_\phi\left(p - p^0\right)\right], \tag{7}$$

where ϕ^0 and p^0 are porosity and pressure in a reference condition, respectively, and c_ϕ is the rock compressibility, yield (Ertekin et al., 2001)

$$V_b\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right) = V_b\left[\frac{\phi_0 c_\phi}{B} + \phi\frac{\partial}{\partial p}\left(\frac{1}{B}\right)\right]\frac{\partial p}{\partial t} = \Gamma\frac{\partial p}{\partial t}. \tag{8}$$

From Eq. (8), Eq. (6) is rewritten as

$$\Gamma\frac{\partial p}{\partial t} - V_b\nabla\cdot\left[\frac{\mathbf{k}}{B\mu}\nabla p\right] + \Gamma_G - q_{sc} = 0, \tag{9}$$

where $\Gamma_G = V_b G$ and G includes gravitational effects.

The source term q_{sc} in Eq. (9) is written in terms of the wellbore pressure

$$q_{sc} = -J_w(p - p_{wf}) \tag{10}$$

where J_w is the Productivity Index (PI). J_w is a function of a set of parameters as, for example, fluid, rock and geometric properties. There are many different expressions that can be applied to compute PI, depending on reservoir, fluid, wellbore and flow characteristics (Dumkwu et al., 2012).

Replacing Eq. (10) in Eq. (9) we get

$$\Gamma\frac{\partial p}{\partial t} - V_b\nabla\cdot\left[\frac{\mathbf{k}}{B\mu}\nabla p\right] + J_w(p - p_{wf}) + \Gamma_G = 0. \tag{11}$$

Equation (11) has the form of hydraulic diffusivity equation, a non-linear partial differential equation, where the dependent variable is the gas pressure.

For the numerical resolution we consider an initial condition given by

$$p(x, y, z, t = 0) = p_{init}(z), \tag{12}$$

where p_{init} is a function of z, and no-flow for external boundary conditions

$$\left(\frac{\partial p}{\partial x}\right)_{x=0,L_x} = \left(\frac{\partial p}{\partial y}\right)_{y=0,L_y} = \left(\frac{\partial p}{\partial z}\right)_{z=0,L_z} = 0, \tag{13}$$

where L_x, L_y and L_z are reservoir lengths in x-, y- and z- directions, respectively. Finally, internal boundary condition is applied using the well rate production prescribed, Q_{sc}, which is equal to the summation of q_{sc} for the cells where there is a part of the horizontal well. If the flow rate is fixed, wellbore pressure is variable. Instead, if the wellbore pressure is specified, the well flow rate is an unknown.

3 NUMERICAL SOLUTION

In this section we give detailed information about the approach applied to solve the Hydraulic Diffusivity Equation through the use of the finite difference method and techniques for solving linear algebraic systems.

3.1 Discretization of the Hydraulic Diffusivity Equation

Finite Difference Method (FDM) (Ertekin et al., 2001) is applied on the discretization of the Eqs. (11)–(13). Figure 2 shows a representation of a discretized reservoir using three-dimensional Cartesian coordinates. In the following, integer indexes i, j and k indicate cell

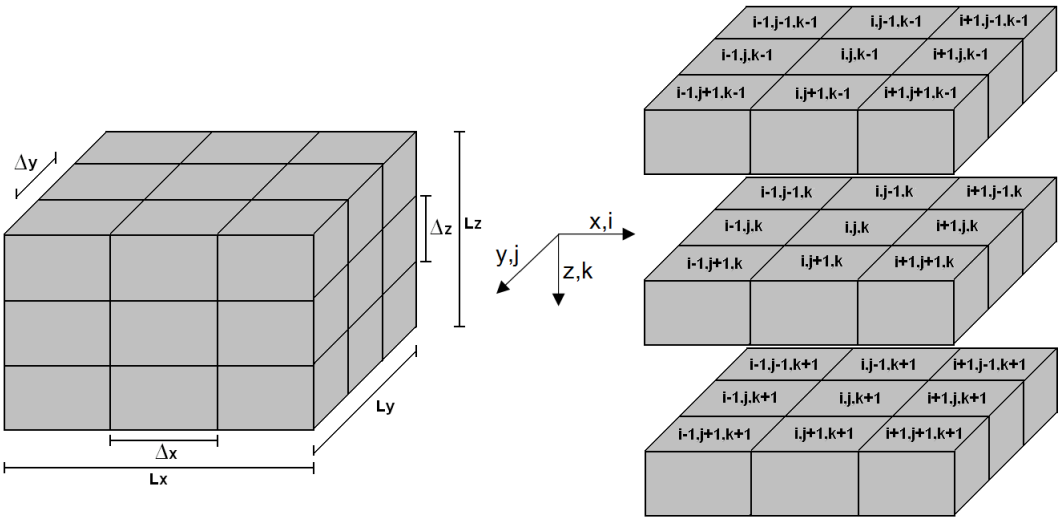

Figure 2: Finite difference discretization using Cartesian coordinates and centered blocks. Integer indexes indicate cell centers.

centers along x-, y- and z- directions, respectively. A fractional index, as $i \pm 1/2, j, k$, indicates a cell boundary.

Considering an anisotropic medium and a diagonal permeability tensor we get from Eq. (11)

$$\Gamma \frac{\partial p}{\partial t} - \frac{\partial}{\partial x}\left(\frac{k_x A_x}{B\mu}\frac{\partial p}{\partial x}\right)\Delta x - \frac{\partial}{\partial y}\left(\frac{k_y A_y}{B\mu}\frac{\partial p}{\partial y}\right)\Delta y - \frac{\partial}{\partial z}\left(\frac{k_z A_z}{B\mu}\frac{\partial p}{\partial z}\right)\Delta z + J_w(p - p_{wf}) = -\Gamma_G, \tag{14}$$

where $V_b = \Delta x \Delta y \Delta z$, $A_x = \Delta y \Delta z$, $A_y = \Delta x \Delta y$ and $A_z = \Delta x \Delta y$.

Using a central difference approximation for centered blocks (Ertekin et al., 2001), we have

$$\frac{\partial}{\partial x}\left(\frac{k_x A_x}{B\mu}\frac{\partial p}{\partial x}\right)^{n+1}_{i,j,k} \approx \frac{1}{\Delta x_{i,j,k}}\left[\left(\frac{k_x A_x}{B\mu}\frac{\partial p}{\partial x}\right)_{i+\frac{1}{2},j,k} - \left(\frac{k_x A_x}{B\mu}\frac{\partial p}{\partial x}\right)_{i-\frac{1}{2},j,k}\right]^{n+1}, \tag{15}$$

where $\Delta x_{i,j,k}$ is the grid spacing in x- direction, $n+1$ represents the temporal level in which pressures are computed and n indicates a time level where pressure is known, and

$$\left(\frac{\partial p}{\partial x}\right)^{n+1}_{i+\frac{1}{2},j,k} \approx \frac{p^{n+1}_{i+1,j,k} - p^{n+1}_{i,j,k}}{\Delta x_{i+\frac{1}{2},j,k}} \qquad \text{and} \qquad \left(\frac{\partial p}{\partial x}\right)^{n+1}_{i-\frac{1}{2},j,k} \approx \frac{p^{n+1}_{i,j,k} - p^{n+1}_{i-1,j,k}}{\Delta x_{i-\frac{1}{2},j,k}}, \tag{16}$$

where $\Delta x_{i\pm 1/2,j,k}$ is the distance between cell centers numbered by i and $i \pm 1$. Similar forms can be obtained for y- and z- coordinates.

The transmissibility in x- direction is computed as

$$T^{n+1}_{x,i\pm\frac{1}{2},j,k} = \left(\frac{k_x A_x}{B\mu\Delta x}\right)^{n+1}_{i\pm\frac{1}{2},j,k} \tag{17}$$

where an harmonic average is applied to determine transmissibility at $i \pm 1/2, j, k$ from the known values in i, j, k and $i+1, j, k$. Analogous expressions can also be determined for y- and z- directions.

Finally, it is possible to get the final discretized form for Eq. (14) using an implicit formulation

$$\begin{aligned}
&\frac{\Gamma^{n+1}_{i,j,k}}{\Delta t}\left(p^{n+1}_{i,j,k}-p^{n}_{i,j,k}\right)-T^{n+1}_{x,i+\frac{1}{2},j,k}\left(p^{n+1}_{i+1,j,k}-p^{n+1}_{i,j,k}\right)-T^{n+1}_{x,i-\frac{1}{2},j,k}\left(p^{n+1}_{i-1,j,k}-p^{n+1}_{i,j,k}\right)\\
&-T^{n+1}_{y,i,j+\frac{1}{2},k}\left(p^{n+1}_{i,j+1,k}-p^{n+1}_{i,j,k}\right)-T^{n+1}_{y,i,j-\frac{1}{2},k}\left(p^{n+1}_{i,j-1,k}-p^{n+1}_{i,j,k}\right)\\
&-T^{n+1}_{z,i,j,k+\frac{1}{2}}\left(p^{n+1}_{i,j,k+1}-p^{n+1}_{i,j,k}\right)-T^{n+1}_{z,i,j,k-\frac{1}{2}}\left(p^{n+1}_{i,j,k-1}-p^{n+1}_{i,j,k}\right)\\
&-J^{n+1}_{w,i,j,k}\left(p^{n+1}_{wf,i,j,k}-p^{n+1}_{i,j,k}\right)=-\left(\Gamma_G\right)^{n+1}_{i,j,k}
\end{aligned} \tag{18}$$

where a backward Euler approximation was used,

$$\left(\frac{\partial p}{\partial t}\right)^{n+1}_{i,j,k}\approx\frac{p^{n+1}_{i,j,k}-p^{n}_{i,j,k}}{\Delta t}, \tag{19}$$

and (Ertekin et al., 2001)

$$\Gamma^{n+1}_{i,j,k}=V_{b,i,j,k}\left\{\frac{\phi^0 c_\phi}{B^{n+1}}+\frac{\phi^n}{B^n}\left[\frac{(B^n/B^{n+1})-1}{p^{n+1}-p^n}\right]\right\}_{i,j,k}. \tag{20}$$

3.2 Numerical Solution for Unknown Pressure

Written for each cell, Eq. (18) leads to a set of non-linear algebraic equations in terms of the unknown pressure. Aiming the use of techniques to solve systems of linear equations, this equation has to be linearized. For transmissibility terms we use (Ertekin et al., 2001),

$$T^{n+1}_{x,i\pm\frac{1}{2},j,k}\cong T^{n+1,v}_{x,i\pm\frac{1}{2},j,k}=\left(\frac{k_x A_x}{B\mu\Delta x}\right)^{n+1,v}_{i\pm\frac{1}{2},j,k}, \tag{21}$$

and Eq. (18) can be written as

$$\begin{aligned}
&\left(\Gamma^{n+1,v}_{i,j,k}/\Delta t\right)\left(p^{n+1,v+1}_{i,j,k}-p^{n}_{i,j,k}\right)-J^{n+1,v}_{w,i,j,k}\left(p^{n+1,v+1}_{wf,i,j,k}-p^{n+1,v+1}_{i,j,k}\right)\\
&-T^{n+1,v}_{x,i+\frac{1}{2},j,k}\left(p^{n+1,v+1}_{i+1,j,k}-p^{n+1,v+1}_{i,j,k}\right)-T^{n+1,v}_{x,i-\frac{1}{2},j,k}\left(p^{n+1,v+1}_{i-1,j,k}-p^{n+1,v+1}_{i,j,k}\right)\\
&-T^{n+1,v}_{y,i,j+\frac{1}{2},k}\left(p^{n+1,v+1}_{i,j+1,k}-p^{n+1,v+1}_{i,j,k}\right)-T^{n+1,v}_{y,i,j-\frac{1}{2},k}\left(p^{n+1,v+1}_{i,j-1,k}-p^{n+1,v+1}_{i,j,k}\right)\\
&-T^{n+1,v}_{z,i,j,k+\frac{1}{2}}\left(p^{n+1,v+1}_{i,j,k+1}-p^{n+1,v+1}_{i,j,k}\right)-T^{n+1,v}_{z,i,j,k-\frac{1}{2}}\left(p^{n+1,v+1}_{i,j,k-1}-p^{n+1,v+1}_{i,j,k}\right)=-\left(\Gamma_G\right)^{n+1,v}_{i,j,k},
\end{aligned} \tag{22}$$

where iterative levels are indicated by v (known values) and $v+1$ (unknown values). The same procedure is used for $\Gamma_{i,j,k}$ and $J_{w,i,j,k}$ terms.

In order to describe a well-reservoir coupling for a single phase flow, it is necessary an equation for the well flow rate and calculate J_w (Peaceman, 1983). We neglect frictional and convective effects inside the well. Total well flow rate, Q_{sc}, must be equal to gas flow summation from all cells containing the well, thus,

$$Q_{sc}=-\sum_{i=W1}^{i=W2}J^{n+1,v}_{w,i,j,k}\left[p^{n+1,v+1}_{i,j,k}-\left(p_{wf}\right)^{n+1,v+1}_{i,j,k}\right] \tag{23}$$

for a well that trespassing layers along with x- direction, starting from cell $W1, j, k$ to $W2, j, k$. A reference position is chosen in order to computing the wellbore pressure. Then, for the well-reservoir coupling we use

$$J_{w,i,j,k}^{n+1,v} = \left\{\frac{1}{\mu B}\right\}_{i,j,k}^{n+1,v} \left\{\frac{2\pi\sqrt{k_z k_y}\Delta x}{\left[1-(r_w/r_{eq})^2\right]\ln\left(r_{eq}/r_w\right)}\right\}_{i,j,k} \tag{24}$$

and

$$r_{eq,i,j,k} = \sqrt{\frac{(\Delta z \Delta y)_{i,j,k}}{\pi}}\ \exp\left(-0.5\right), \tag{25}$$

where r_{eq} is the equivalent radius (Peaceman, 1983, Al-Mohannadi et al., 2007) and r_w is the wellbore radius.

Equations (22) and Eq. (23) form a system of linearized equations for reservoir and wellbore pressures, and this set of equations is solved using an approximate factorization technique. This technique is appropriate to handle with practical simulation of heterogeneous reservoirs. The Conjugate Gradient (CG), Biconjugate Gradient (BiCG) and Biconjugate Gradient Stabilized (BiCGStab) methods were employed in this work. Considering the iterative procedure, the numerical solution is achieved when

$$max\left|\chi^{n+1,v+1} - \chi^{n+1,v}\right| < tol \tag{26}$$

where χ represents reservoir and wellbore pressures and tol is a numerical tolerance. A nested iteration procedure (Ertekin et al., 2001) is used: in the inner iteration, the iterative method is applied in order to solve pressures and, in the outer iteration, transmissibilities are updated. A general numerical solution algorithm can be found in de Souza & Souto (2016).

4 RESOLUTION OF ALGEBRAIC SYSTEM

In the previous section, we presented the discretized forms of the Hydraulic Diffusivity Equation, that governs the single-phase flow in a gas reservoir, and of the well-reservoir coupling equation. Next, to obtain a numerical solution for the reservoir and wellbore pressures, the set of non-linear algebraic equations was linearized (de Souza & Souto, 2016). In general, the procedure of finding a numerical solution consists of the resolution of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is known as the coefficient matrix and the vector $\mathbf{b}$ contains known terms. As is well known, there are two main techniques for solving linear algebraic systems which are *direct methods* and *indirect or iterative methods*. Here we are just concerned with the iterative methods.

The iterative methods are based on repetitive application of relatively simple algorithms. The convergence is not always guaranteed, but when it's possible, it is obtained after a set of iterations. In this work, the Conjugate Gradient (CG), Algorithm 1, BiConjugate Gradient (BiCG), Algorithm 2, and BiConjugate Gradient Stabilized (BiCGStab), Algorithm 3, were chosen (Barrett et al., 1994) to be applied in the resolution of algebraic systems. Seeking to improve performance of the numerical computation, our main goal is to develop parallelized versions of these methods using the API OpenMP to be running in a Intel Xeon Phi coprocessor with 57 cores (228 threads). In the algorithms we can see all the sequence of instructions/operations that were parallelized with the OpenMP.

Algorithm 1: Conjugate Gradient Method (CG).

Give the maximum number of iterations $nmax$ and the tolerance tol; Compute $r_0 = b - Ax_0$ for some initial guess x_0; Do $p_0 = r_0$.
Compute $||r_0||$
for $k = 0 : nmax$ **do**
    Compute Ap_k `// #pragma omp parallel for private (i,j,sum)`
    Compute $< p_k, Ap_k >$ `// #pragma omp parallel for reduction (+:pAp)`
    Compute $< r_k, r_k >$ `// #pragma omp parallel for reduction (+:r_old_norm)`
    $\alpha_k = \dfrac{< r_k, r_k >}{< p_k, Ap_k >}$
    $x_{k+1} = x_k + \alpha_k p_k$ `// #pragma omp parallel for`
    $r_{k+1} = r_k - \alpha_k Ap_k$ `// #pragma omp parallel for`
    Compute $< r_{k+1}, r_{k+1} >$ `// #pragma omp parallel for reduction (+:r_new_norm)`
    $\beta_k = \dfrac{< r_{k+1}, r_{k+1} >}{< r_k, r_k >}$
    $p_{k+1} = r_{k+1} + \beta_k p_k$ `// #pragma omp parallel for`
    Compute $||r_{k+1}||$
    if $||r_{k+1}||/||r_0|| < tol$ **then**
        $x = x_{k+1}$
    end
    else
        If $k = nmax$, there was no convergence
    end
end

Over the years, the scientific community have frequently required accurate numerical results that describe adequately real problems. In order to be up to date with all the latest innovations we must overcome the major barriers imposed by hardware constraints of some personal computers, such as processing speed, memory capacity and scientific visualization among others, searching for new technologies as graphics processing unit (GPU) and multicore coprocessor. With regard to storage capacity, some compression matrix methods have been developed and we highlighted here: Coordinate Format (COO); Compressed Sparse Row (CSR); Compressed Sparse Column (CSC) and Matrix Market Format. Aiming to optimize the use of available memory, these methods propose to storage only the non zero elements of the coefficient matrix that, in many practical problems, are sparse matrix. Proceeding in this way we can reduce considerably the need for storage and the Compressed Sparse Row (CSR) method is utilized in this work.

OpenMp is an Application Programming Interface (API) developed and maintained by the OpenMP group Architecture Review Board (ARB) and is based on parallel programming for shared-memory in multithreaded architectures (Chapman et al., 2008). This interface is composed by three basic components: Compiler Directives; Runtime Library Routines and Environment Variables. This API supports C, C++ and Fortran on a wide variety of architectures and platforms using Unix, Linux and Windows. The OpenMP is not a programming language, but rather a set of specifications that allows parallelization of numerical codes through the incorporation of directives that indicate how the work should be divided between the cores/threads.

Therefore, this interface allow us a easy way to take advantage of parallel processing without demanding great changes in the numerical codes.

Algorithm 2: BiConjugate Gradient Method (BiCG).

Give the maximum number of iterations $nmax$ and the tolerance tol; Compute $r_0 = b - Ax_0$ for some initial guess x_0; Do $\widehat{r}_0 = r_0$; Do $p_0 = r_0$, $\widehat{p}_0 = \widehat{r}_0$.
Compute $||r_0||$
for $k = 0 : nmax$ **do**
    Compute Ap_k `// #pragma omp parallel for private (i,j,Ap)`
    Compute $< r_k, \widehat{r}_k >$ `// #pragma omp parallel for reduction (+:r_new_norm)`
    Compute $< Ap_k, \widehat{p}_k >$ `// #pragma omp parallel for reduction (+:pAp)`
    $\alpha_k = \dfrac{< r_k, \widehat{r}_k >}{< Ap_k, \widehat{p}_k >}$
    $x_{k+1} = x_k + \alpha_k p_k$ `// #pragma omp parallel for`
    $r_{k+1} = r_k - \alpha_k Ap_k$ `// #pragma omp parallel for`
    Compute $A^T\widehat{p}_k$ `// #pragma omp parallel for private (i,j,Ap_)`
    $\widehat{r}_{k+1} = \widehat{r}_k - \alpha_k A^T\widehat{p}_k$ `// #pragma omp parallel for`
    Compute $< r_{k+1}, \widehat{r}_{k+1} >$ `// #pragma omp parallel for reduction (+:r_new_norm)`
    $\beta_k = \dfrac{< r_{k+1}, \widehat{r}_{k+1} >}{< r_k, \widehat{r}_k >}$
    $p_{k+1} = r_{k+1} + \beta_k p_k$ `// #pragma omp parallel for`
    $\widehat{p}_{k+1} = \widehat{r}_{k+1} + \beta_k \widehat{p}_k$ `// #pragma omp parallel for`
    Compute $||r_{k+1}||$
    if $||r_{k+1}||/||r_0|| < tol$ **then**
        $x = x_{k+1}$
    end
    else
        If $k = nmax$, there was no convergence
    end
end

5 NUMERICAL RESULTS

In order to verify the computational efficiency of the parallelized versions (employing the API OpenMP) of the linear solvers, a problem of practical interest was considered: gas flow in a hydrocarbons reservoir producing through a horizontal well.

All the following simulations were performed using a set of parameters (default data) that remained unchanged unless otherwise specified, and they can be found in Table 1. In this table, Δt_i is the initial time step, Δt_f is the final time step and F_{Δ_t} is the increase rate for Δt (multiplying consecutive time steps, starting by the Δt_i). The term P_{init} indicates the pressure at the top of the reservoir. In all numerical experiments, correlations by Lee et al. (1966) for viscosity and by Dranchuk & Abou-Kassem (1975) for real gas deviation factor were applied. The horizontal well is also parallel to the $x-$direction.

Algorithm 3: BiConjugate Gradient Stabilized Method (BiCGStab).

Give the maximum number of iterations $nmax$ and the tolerance tol; Compute $r_0 = b - Ax_0$ for some initial guess x_0; Do $\widehat{r}_0 = r_0$; Do $p_0 = r_0$.
Compute $||r_0||$
for $k = 0 : nmax$ **do**
    Compute Ap_k `// #pragma omp parallel for private (i,j,Ap)`
    Compute $< r_k, \widehat{r}_0 >$ `// #pragma omp parallel for reduction (+:r_old_norm)`
    Compute $< Ap_k, \widehat{r}_0 >$ `// #pragma omp parallel for reduction (+:t1)`
    $\alpha_k = \dfrac{< r_k, \widehat{r}_0 >}{< Ap_k, \widehat{r}_0 >}$
    $s_k = r_k - \alpha_k Ap_k$ `// #pragma omp parallel for`
    Compute As_k `// #pragma omp parallel for private (i,j,As)`
    Compute $< As_k, As_k >$ `// #pragma omp parallel for reduction (+:t2)`
    Compute $< As_k, s_k >$ `// #pragma omp parallel for reduction (+:t)`
    $\omega_k = \dfrac{< As_k, s_k >}{< As_k, As_k >}$
    $x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k$ `// #pragma omp parallel for`
    $r_{k+1} = s_k - \omega_k As_k$ `// #pragma omp parallel for`
    Compute $< r_{k+1}, \widehat{r}_0 >$ `// #pragma omp parallel for reduction (+:r_new_norm)`
    $\beta_k = \dfrac{\alpha_k < r_{k+1}, \widehat{r}_0 >}{\omega_k < r_k, \widehat{r}_0 >}$
    $p_{k+1} = r_{k+1} + \beta_k (p_k - \omega_k Ap_k)$
    `// #pragma omp parallel for`
    Compute $||r_{k+1}||$
    if $||r_{k+1}||/||r_0|| < tol$ **then**
        $x = x_{k+1}$
    end
    else
        If $k = nmax$, there was no convergence
    end
end

5.1 Computational Efficiency

The appropriateness of using parallel computation in the reservoir simulation, with a horizontal well, was verified using three-dimensional numerical grids defined by $n_x \times n_y \times n_z$ where n_x, n_y and n_z stand for the number of cells in the $x-$, $y-$ and $z-$directions, as shown in Table 2. Also in this table n_w represents the number of cells occupied by the well. Initially, all the runs were performed considering the three linear solvers employed in this work: Conjugate Gradient, BiConjugate Gradient and BiConjugate Gradient Stabilized.

Wellbore pressure results are depicted in Figs. 3 and 4. Figure 3 shows serial and parallel results for CG method and the different grids. Experimental evidence of numerical convergence can be observed. Wellbore results for early times are impacted by a numerical artifact related to the well-reservoir technique applied (Peaceman, 1983). In Fig. 4 results are depicted for the three different methods using the grid 4. Concordant results were found in these tests.

Table 1: Default data used in all simulations.

Parameter	Value	Parameter	Value	Parameter	Value
Δt_i (days)	1.0×10^{-3}	Δt_f (days)	10	F_{Δ_t}	1.1
p_{sc} (Pa)	1.01353×10^5	M (kg/kg-mol)	17.4	n_x	128
p^0 (Pa)	3.0×10^7	$L_x = L_y$ (m)	5000	n_y	129
P_{init} (Pa)	3.0×10^7	L_z (m)	100	n_z	17
c_ϕ (Pa^{-1})	4.0×10^{-10}	L_{wf} (m)	1250	n_w	64
Q_{sc} (std m^3/day)	-5.0×10^5	t_{max} (days)	90	ϕ_{init}	0.2
$k_x = k_y$ (m^2)	5.0×10^{-15}	T_{sc} (K)	288.71	ϕ_0	0.2
k_z (m^2)	2.0×10^{-15}	T (K)	340	r_w (m)	0.05

Table 2: Numerical grids considered.

Grid	n_x	n_y	n_z	n_w
1	16	17	3	4
2	32	33	5	8
3	64	65	9	16
4	128	129	17	32
5	256	257	33	64

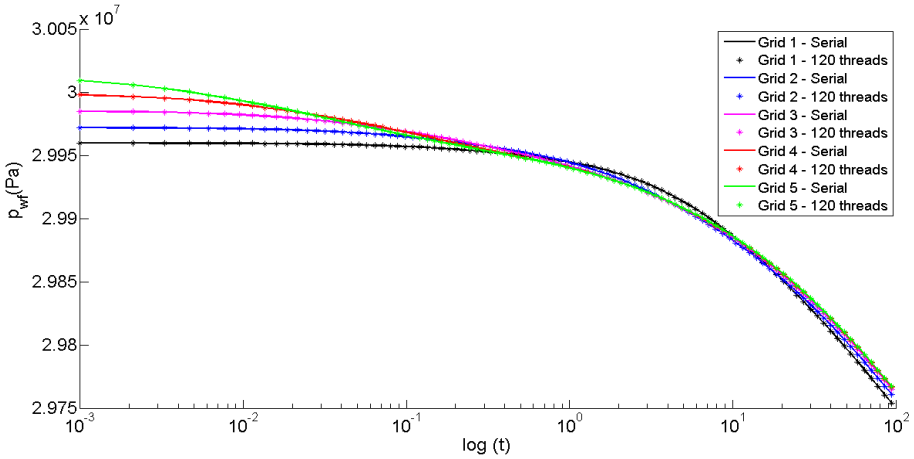

Figure 3: Results for CG method (different grids).

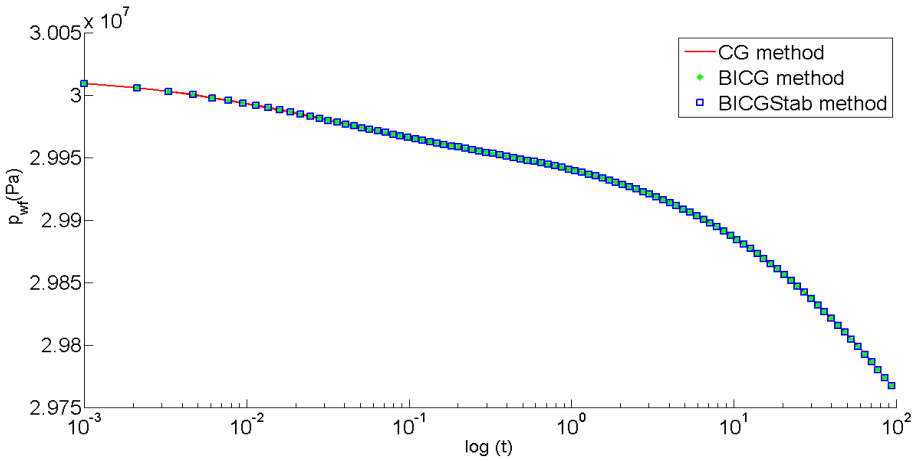

Figure 4: Results for CG, BiCG and BiCGStab methods (grid 4 and with parallezation).

Tables 3-6 show for the five different grids used here the execution times and the corresponding speedups obtained for each simulation while only the execution times are depicted in Figs. 5-8. As already expected, the speedup increases as the number of cells increases. For the less refined grid, computational costs using OpenMP are the same or larger than when we used serial implementations for all the three linear solvers. The maximum speedups of only 1.0 are reached for 40 threads and for CG and BiCGStab methods (Table 3). Figure 5 depicts the execution times, showing the same results to serial case for 40 threads and CG and BiCGStab methods for the less refined grid. It is possible to conclude that there is no gain to justify the parallelization of the algebraic linear system algorithms for n_x = 16, n_y = 17 and n_z = 3.

Table 3: Comparison between the methods of CG, BiCG and BiCGStab (n_x = 16, n_y = 17 and n_z = 3)

Nº of Threads	CG		BiCG		BiCGStab	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	3	-	3	-	3	-
40	3	1.0	4	0.8	3	1.0
80	4	0.8	4	0.8	5	0.6
120	4	0.8	4	0.8	5	0.6
160	5	0.6	5	0.6	5	0.6
200	5	0.6	6	0.5	5	0.6

For the grid in which n_x = 32, n_y = 33 and n_z = 5 the advantage of using OpenMP is almost negligible for all the three linear solvers for some runs, but a maximum speedup of 1.8 is reached for 40 threads and the BiCGStab method (Table 4). Execution times for this grid are shown in Fig. 6). In such a case we can conclude that depending on the number of threads can exist or can not exist enough gain to justify the parallelization of the algebraic linear system algorithms. For the grid under analysis, there are execution times which are quite the same for parallelized and serial algorithms or gains nearly 2.

Table 4: Comparison between the methods of CG, BiCG and BiCGStab (n_x = 32, n_y = 33 and n_z = 5)

Nº of Threads	CG		BiCG		BiCGStab	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	22	-	25	-	25	-
40	15	1.5	15	1.7	14	1.8
80	16	1.4	16	1.6	16	1.6
120	18	1.2	18	1.4	18	1.4
160	18	1.2	18	1.4	17	1.5
200	20	1.1	20	1.3	19	1.3

When we consider the next grid, n_x = 64, n_y = 65 and n_z = 9, the effort employed to parallelize the solvers begins to be more rewarded and we get a top speedup of 2.6 for the BiCG and BiCGStab methods with 40 threads although it does not exceed 2.1 for 200 threads, see Table 5 and Fig. 7. The shorter execution time for one thread is obtained for the CG method (212 s) and its corresponding for 40 threads is reached for the BiCGStab (92 s).

Table 5: Comparison between the methods of CG, BiCG and BiCGStab (n_x = 64, n_y = 65 and n_z = 9)

Nº of Threads	CG		BiCG		BiCGStab	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	212	-	242	-	237	-
40	93	2.3	93	2.6	92	2.6
80	97	2.2	98	2.5	105	2.3
120	106	2.0	105	2.3	104	2.3
160	105	2.0	105	2.3	104	2.3
200	114	1.9	113	2.1	113	2.1

The computational efficiency is still greater for a grid with a total number of cells equal to 280,704 (n_x = 128, n_y = 129 and n_z = 17). In this case, Table 6 and Fig. 8, the serial execution time can be reduced by a factor of five approximately in one of the simulations. The best results are obtained for the BiCG method for a speedup of 4.8 and 160 threads. Anyway, all the three parallelized methods attained speedups that exceed 2.5 and can already provide great savings with respect to the overall runtime. The CG method holds the shorter execution time (2321 s) for one thread and the BiCG for 160 threads (573 s).

Finally, for a grid with 256 $\times$ 257 $\times$ 33 cells, as we can see in Table 7 and Fig. 9, the best results for the speedup values are attained in this comparative study. The highest speedup is 6.1 (160 threads) for the BiCG method. Regarding execution times, 28818 s are need to reach

Table 6: Comparison between the methods of CG, BiCG and BiCGStab (n_x = 128, n_y = 129 and n_z = 17)

Nº of Threads	CG		BiCG		BiCGStab	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	2321	-	2739	-	2537	-
40	823	2.8	719	3.8	694	3.7
80	759	3.1	699	3.9	676	3.8
120	691	3.4	699	3.9	690	3.7
160	701	3.3	573	4.8	602	4.2
200	681	3.4	625	4.4	611	4.2

convergence with the CG method using one thread while only 5415 s are necessary with the BiCGStab for 160 threads.

Table 7: Comparison between the methods of CG, BiCG and BiCGStab (n_x = 256, n_y = 257 and n_z = 33)

Nº of Threads	CG		BiCG		BiCGStab	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	28818	-	34462	-	31463	-
40	5823	4.9	6043	5.7	5904	5.3
80	5569	5.2	5755	6.0	5603	5.6
120	5536	5.2	5753	6.0	5532	5.7
160	5479	5.3	5632	6.1	5415	5.8
200	5560	5.2	5707	6.0	5489	5.7

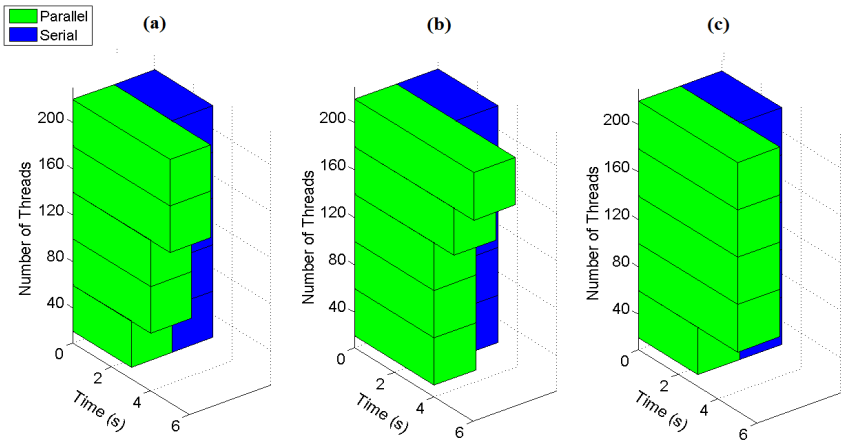

Figure 5: Execution time (n_x = 16, n_y = 17 and n_z = 3): (a) CG, (b) BiCG and (c) BiCGStab methods.

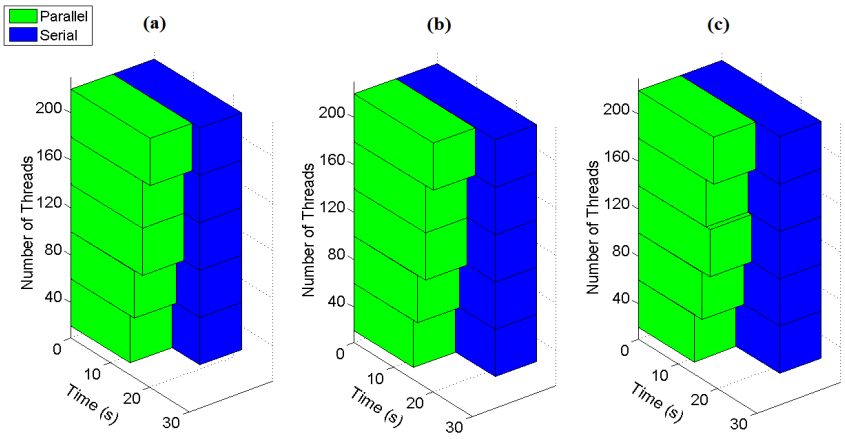

Figure 6: Execution time (n_x = 32, n_y = 33 and n_z = 5): (a) CG, (b) BiCG and (c) BiCGStab methods.

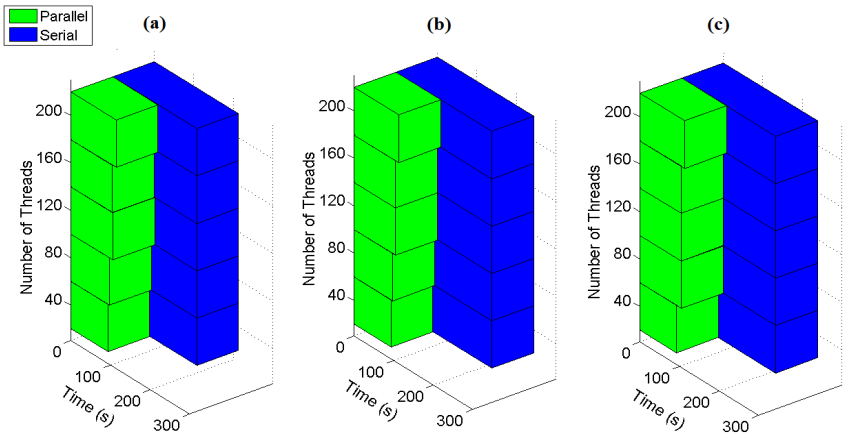

Figure 7: Execution time (n_x = 64, n_y = 65 and n_z = 9): (a) CG, (b) BiCG and (c) BiCGStab methods.

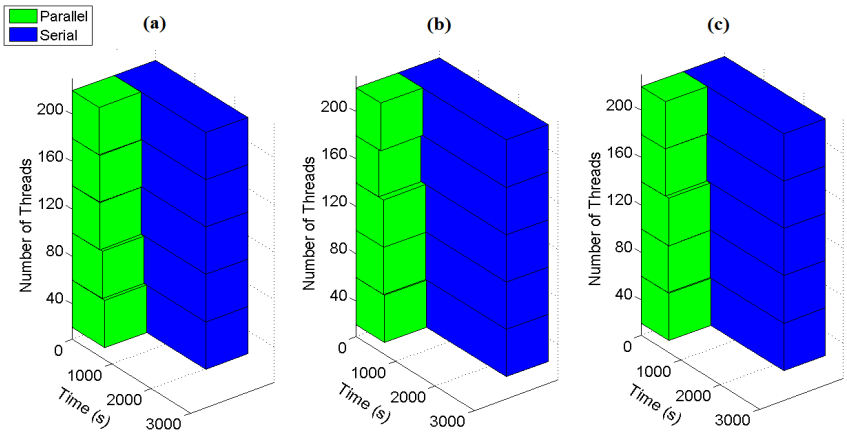

Figure 8: Execution time (n_x = 128, n_y = 129 and n_z = 17): (a) CG, (b) BiCG and (c) BiCGStab methods.

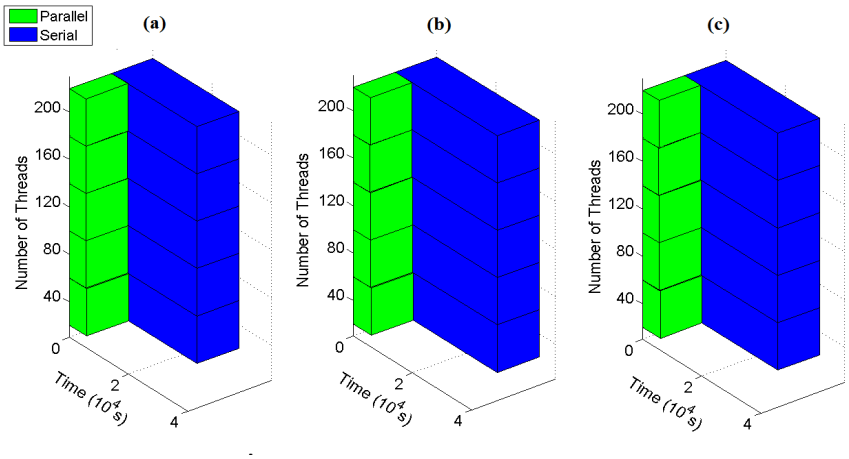

Figure 9: Execution time (n_x = 256, n_y = 257 and n_z = 33): (a) CG, (b) BiCG and (c) BiCGStab methods.

5.2 Sensibility Analysis

For the sensibility analysis where we determined the influence of some important parameters in the wellbore pressure only the Conjugate Gradient method was used in all simulations so as not to extend this work. In these tests a more refined grid was applied, in which n_x = 256, n_y = 257 and n_z = 33. The first analysis deal with molecular gas weight variations and we performed simulations for M equal to 17.40, 18.80 and 20.20 kg/kg-mol (Table 8). As we can see the use of the OpenMP and the coprocessor allowed us to obtain a considerable gain in execution time when compared to the serial algorithm. We can get up to a speedup of 5.3 ($M = 17.40$ kg/kg-mol) for 160 threads with a minimum of 4.4 ($M = 20.20$ kg/kg-mol) for 40 threads.

Table 8: Sensibility analysis for molecular weight.

Nº of Threads	Molecular Weight (kg/kg-mol)					
	17.40		18.80		20.20	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	28818	-	28459	-	29552	-
40	5823	4.9	5843	4.9	6696	4.4
80	5569	5.2	5630	5.1	6439	4.6
120	5536	5.2	5539	5.1	6358	4.6
160	5479	5.3	5468	5.2	6322	4.7
200	5560	5.2	5575	5.1	6338	4.7

According to the results show in Table 9, obtained for three set of different permeabilities, we realized that we can get better top speedups than such determined in the previous analysis. Now we attain a maximum speedup of 7.9 ($k_x = k_y$ = 20.0 $\times 10^{-15}$ m^2 and k_z = 8.0$\times 10^{-15}$ m^2)

for 160 threads and a minimum of 4.9 (k_x = k_y = 5.0 $\times 10^{-15}$ m^2 and k_z = 2.0 $\times 10^{-15}$ m^2) for 40 threads.

Table 9: Sensibility analysis for permeability.

Nº of Threads	Permeability (m^2)					
	$k_x = k_y = 5.0 \times 10^{-15}$ $k_z = 2.0 \times 10^{-15}$		$k_x = k_y = 10.0 \times 10^{-15}$ $k_z = 4.0 \times 10^{-15}$		$k_x = k_y = 20.0 \times 10^{-15}$ $k_z = 8.0 \times 10^{-15}$	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	28818	-	36738	-	46545	-
40	5823	4.9	6162	6.0	6565	7.1
80	5569	5.2	5776	6.4	6137	7.6
120	5536	5.2	5712	6.4	5991	7.8
160	5479	5.3	5681	6.5	5907	7.9
200	5560	5.2	5822	6.3	6298	7.4

The next results (Table 10) concern the sensibility analysis for horizontal well lengths: 312.5, 625.0 and 1250.0 meters. Once again we achieve good performance with the parallel algorithm with speedups varying between 4.9 (L_{wf}=1250 m) with 40 threads and 5.5 (L_{wf}=312.5 m) with 160 threads. Despite the decreasing values regarding the results for the variation of permeabilities we still have a non negligible increase in the computational efficiency.

Table 10: Sensibility analysis for well length (m).

Nº of Threads	Well Length (m)					
	312.5		625.0		1250.0	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
1	30037	-	29593	-	28818	-
40	5843	5.1	5879	5.0	5823	4.9
80	5620	5.3	5581	5.3	5569	5.2
120	5517	5.4	5533	5.3	5536	5.2
160	5491	5.5	5463	5.4	5479	5.3
200	5515	5.4	5507	5.4	5560	5.2

6 CONCLUSIONS

In this paper, we used the Hydraulic Diffusivity Equation in order to simulate numerically gas flow in hydrocarbons reservoirs. Finite-difference method and an implicit formulation were

used in order to obtain numerical solutions for the wellbore pressure for a constant production rate, for field scale gas flow simulations and for production through horizontal wells. Several scenarios for different physical parameters were also considered in our study of computational costs.

Results showed that the parallelism using API OpenMP makes sense when there is an actual workload. For less refined numerical grids there were no expressive gains, with some performance loss when we use parallelism. In fact, it must be considered that the parallezation brings some computational cost, what is compensated when grids are more refined. It is worth to note that there is a optimum number of threads for each problem.

The API OpenMP and the Intel Xeon Phi coprocessor enabled a significant reduction in the CPU time for the numerical reservoir simulations addressed in this work, resulting in increased computational efficiency for the parallelized versions of the Conjugate Gradients, Bi-Conjugate Gradient and Bi-conjugate Gradient Stabilized methods. Therefore, this approach can be considered as a useful tool to simulate gas flow through porous media in situations where we have to use refined mesh grids as is the case of the simulations involving three-dimensional space, heterogeneous reservoirs, unconventional reservoirs, low-permeability reservoirs, presence of hydraulic fractures, naturally fractured reservoirs, etc.

ACKNOWLEDGMENTS

Authors gratefully thanks State University of Rio de Janeiro, CAPES and CNPq for their support.

REFERENCES

Akgun, F., 2004. A finite element model for analyzing horizontal well BHA behavior. *Journal of Petroleum Science and Engineering*, vol. 42, No. 1-2, pp. 121–132.

Al-Mohannadi, N. S., 2004. *Simulation of Horizontal Well Tests*. PhD Thesis, Colorado School of Mines/Golden.

Al-Mohannadi, N.S., Ozkan, E. & Kazemi, H., 2007. Pressure-transient responses of horizontal and curved wells in anticlines and domes. *Society of Petroleum Engineers Reservoir Evaluation & Engineering*, vol. 10, n. 1, pp. 66–76.

Barree, R. D. & Conway, M. W., 2004. Beyond Beta Factors: A Complete Model for Darcy, Forchheimer, and Trans-Forchheimer Flow in Porous Media. *Society of Petroleum Engineers Annual Technical Conference and Exhibition*, 26-29 September, Houston, Texas, USA.

Barrett R., Berry M. W., Chan T. F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C. & Van der Vorst H., 1994. *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM.

Chapman, B., Jost G. & Pas, R. V. D., 2008. *Using OpenMP: Portable Shared Memory Parallel Programming*, MIT Press.

Chaudhry, A. U., 2003. *Gas Well Testing Handbook*. Elsevier, Burlington, Massachusetts, USA.

De Souza, G., 2013. *Acoplamento Poço-reservatório na Simulação Numérica de Reservatórios de Gás*. PhD Thesis, Universidade Estadual do Norte Fluminense/Macaé.

De Souza, G., Pires, A.P. & Abreu, E., 2014. Well-reservoir coupling on the numerical simulation of horizontal wells in gas reservoirs. *In Society of Petroleum Engineers Latin America and Caribbean Petroleum Engineering Conference.*, 21-23 May, Maracaibo, Venezuela.

De Souza G. & Amaral Souto H. P., 2016. A comparative study of non-Darcy flows in gas naturally fractured reservoirs. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*. DOI: 10.1007/s40430-016-0486-x.

Dranchuk, P. M. & Abou-Kassem, J. H., 1975. Calculation of Z factors for Natural Gases using Equations of State. *Journal of Canadian Petroleum Technology*, vol. 14, n. 3, pp. 34–36.

Dumkwu, F. A., Islam, A.W., & Carlson, E. S., 2012. Review of Well Models and Assessment of their Impacts on Numerical Reservoir Simulation Performance. *Journal of Petroleum Science and Engineering*, vols. 82-83, pp. 174–186.

Ertekin, T., Abou-Kassem, J. H., & King, G. R., 2001. *Basic Applied Reservoir Simulation*. Society of Petroleum Engineers.

He, J. & Durlofsky, L. J., 2013. Reduced-order Modeling for Compositional Simulation Using Trajectory Piecewise Linearization. *Society of Petroleum Engineers Reservoir Simulation Symposium*, 18-20 February, The Woodlands, Texas, USA.

Jmili, A., Wilhite, G. P., & Green, D., 2011. Modeling Gas-phase Mass Transfer Between Fracture and Matrix in Naturally Fractured Reservoirs. *Society of Petroleum Engineers Journal*, vol. 16, n. 4, pp. 795–811.

Lee, A. L., Gonzalez, M. H., & Eakin, B. E., 1966. The Viscosity of Natural Gases. *Petroleum Technology, Transactions of AIME*, vol. 18, n. 8, pp. 997–1000.

Peaceman, D. W., 1983. Interpretation of Well-block Pressures in Numerical Reservoir Simulation with Nonsquare Grid Blocks and Anisotropic Permeability. *Society of Petroleum Engineers Journal*, vol. 23, n. 3, pp. 531–543.

Shahbazi, S., Maarefvand, P. and Gerami, S., 2015. Investigation on flow regimes and non-Darcy effect in pressure test analysis of horizontal gas wells. *Journal of Petroleum Science and Engineering*, vol. 129, pp. 121–129.