Optimizations in an numerical method code for two-phase fluids flow in porous media using the SDumont supercomputer

Stiw Herrera[1], Thiago Teixeira[1], Weber Ribeiro[1], André Carneiro[1], Frederico L. Cabral[1], Matheus Serpa[2], Márcio Borges[1], Carla Osthoff[1], Sanderson L. Gonzaga de Oliveira[3], Philippe Navaux[2]

[1]National Scientific Computing Laboratory
Avenue Getulio Vargas 333, 25651-076, Petrópolis-RJ, Brazil
stiw@lncc.br, tteixeira@lncc.br, webergdr@lncc.br, andrerc@lncc.br, fcabral@lncc.br, mrborges@lncc.br, osthoff@lncc.br
[2]Institute of Informatics of the Federal University of Rio Grande do Sul
Avenue Paulo Gama, 110, 90040-060, Porto Alegre-RS, Brazil matheusserpa@gmail.com, navaux@inf.ufrgs.br
[3]Universidade Federal de Lavras, Campus Universitário, 37200-000, Lavras/MG, Brazil
sanderson@ufla.br

Abstract.

In petroleum reservoir simulations, the level of detail incorporated into the geologic model typically exceeds the capabilities of traditional flow simulators. In this sense, such simulations demand new high-performance computing techniques to deal with a large amount of data allocation and the high computational cost to compute the behavior of the fluids in the porous media. This paper presents optimizations performed on a code that implements an explicit numerical scheme to approximate the solution of the governing differential equation for water saturation in a two-phase flow problem with heterogeneous permeability and porosity fields. The experiments were performed on the SDumont Supercomputer using 2nd Generation Intel® Xeon® Scalable Processors (formerly Cascade Lake architecture). We employed the domain decomposition method to split the 3-D reservoir onto n sub-domains that are solved using each Message Passing Interface (MPI) process. We analyzed the performance of the domain decomposition strategies and identified communication bottlenecks as the mesh size and the number of computational nodes increase. The results show that the optimizations implemented in the numerical code remarkably reduce the execution time of the simulations.

Keywords: Porous media, Numerical methods, Message Passing Interface (MPI), High-performance computing, Decomposition domain strategy, Supercomputing for enabling large-scale advanced simulations.

1 Introduction

Numerical simulations of gas and oil reservoirs consist of developing mathematical models to describe the physical processes of fluid flows in a porous medium. These mathematical problems often have no analytical solution, and numerical methods can provide approximated solutions for them.

Natural porous media, such as aquifers and reservoirs, present a high degree of heterogeneities in their petrophysical properties (porosity, permeability, etc.). These heterogeneities extend from the pore scale to the field scale, and governing equations incorporate them for flow problems based on geostatistical models [1–4].

The variability and complex connectivity of the petrophysical properties play a remarkable impact on subsurface flow and transport behavior, and simulations of realistic problems must include them, as mentioned in Chen et al. [5]. For these reasons, in typical simulations, we need to discretize huge domains in fine meshes. Consequently, one faces a very demanding computational problem that needs high-performance computing to be approximately solved.

The geometric representation of the reservoirs, which can present complex shapes, stimulates the

adoption of irregular meshes in their discretization, as discussed by Gonzaga de Oliveira [6]. On the other hand, despite their less fidelity to geometric representation, the application of regular grids to discretize the domains allows us to use more simplified and faster calculations. Additionally, as the resolution of the problem investigated in this study is highly scalable, it is possible to compensate for this loss of precision by increasing the number of volume elements. The scalability of the code also allows us to refine the mesh until reaching reasonable accuracy of the solution, thus compensating for the considerable numerical diffusion present in first-order schemes (upwind). Furthermore, according to Tuane [7] one can incorporate a hyperbolic conservation nature law to design better approximate code solutions.

Domain decomposition techniques divide a domain into subdomains. Thus, they can be calculated in parallel using Message Passing Interface (MPI), as recommended in Forum [8]. The number of domain divisions depends on computational and communication costs. The higher the computational cost of the method, the greater the number of possible domain partitions. The higher the communication cost generated by the method stencil, the lower the number of subdomain partitions.

This paper studies the domain decomposition technique called Yotov block decomposition, as seen in Parashar and Yotov [9]. Specifically, we use the scheme to test different vertical and horizontal cross-sections to evaluate the best block decomposition given a certain number of processes, which presents better performance. This work is part of a research effort that aims to develop a computational environment for academic studies to give scalability to the new numerical methods developed in the oil and gas field. We are developing a scalable code written in Fortran language to approach the solution of the advection problem using the well-known first-order finite volume method upwind, presented in LeVeque [10]. We refer to this code as the simulator. Next, we will implement numerical methods of higher-order in the code, as recommended in Correa and Borges [11], Carneiro et al. [12].

We organized this paper as follows. In Section 2, we present related work. Afterward, in Section 3, we describe the mathematical model. In Section 4, we describe the approaches of Domain Decomposition strategy with MPI. In Section 5, we describe the tests performed. Finally, in Section 6, the conclusion and future work are presented.

2 Related Work

Similar to the present study, Wang et al. [13] developed a parallel fluid dynamics simulation using finite volume discretizations. In general, numerical methods with an implicit approach Leveque [14] generate large matrices, making it difficult to calculate in a single machine. Thus, they used specific data structures to allocate matrices efficiently.

Palin [15] presented domain decomposition techniques and parallel processing with message exchange using MPI. Herrera et al. [16] presented a performance and scalability study of a restructured code in a real-world application of a fluid flow simulator. The authors ran the code on the Sdumont supercomputer. The present study presents a performance and scalability study of a strategy to reduce MPI messages according to the location of the mesh points. Lima [17] described several methods and examples of domain division and studied how to manage fluid meshes and reservoir properties datasets to reduce communications between MPI processes. The paper presented a domain split best suited for modeling reservoirs with complex geometries. The same study showed the performance in parallel for both 2D and 3D meshes. The present paper studies the performance and scalability of a strategy to reduce MPI messages according to the location of the mesh points.

3 Mathematical Model-Fluid Flow Equation

The mathematical model that describes the flow of fluids in porous media is composed of partial differential equations Chen [18], Tuane [7]. The transport equation is solved using the finite volume method. We describe the equation below:

For $\Omega \subset R^3$ a connected domain, open and limited and I a break time. We consider the scalar conservation law.

$$\phi \frac{\partial s}{\partial t} + \nabla \cdot \mathrm{f} = 0 \text{ in } \Omega \times \mathrm{I}, \tag{1}$$

where $\phi : \Omega \Rightarrow (0, \phi^{max}]$ is the storage coefficient, $s : \Omega \times \mathrm{I} \rightarrow Im\{s\} = [s^{min}, s^{max}]$ is the scalar function, and the vector function $\mathrm{f} : Im\{s\} \rightarrow R^3$ is the flow of the conserved quantity s. Although the code was

developed for the two-phase flow problem, for the computational tests (presented in this work) we have considered the passive tracer problem. In this case the velocity field $\vec{v}$ remains constant throughout the simulation and the flux function is given by:

$$\mathrm{f} = s\vec{v} = \left\{ \begin{array}{c} s \\ 0 \\ 0 \end{array} \right\}. \tag{2}$$

The partial differential equation (1) with the flux function given by (2) describes a passive tracer flow problem. This problem associated with appropriate boundary conditions is solved using the upwind finite volume method LeVeque [10]. For more information on initial and boundary conditions see Correa and Borges [11].

4 MPI and domain decomposition strategy

The domain decomposition division initially implemented in the simulator was carried out to balance the computational cost between the MPI processes. For this, the MPI processes were divided into a three-dimensional topology, seeking to generate subdomains with the same amount of volume elements for each dimension (Nx, Ny, and Nz). Herrera et al. [16], presented the execution time and speed-up from this domain decomposition. The authors identified that as the number of processors and mesh size increase, the number of MPI messages increases so that communication execution time dominates the total execution time of the simulation. 1 shows the results of this paper.

To decrease the number of MPI messages, we adopted a new domain decomposition strategy. The domain decomposition strategy adopted in this work is called block decomposition. Lima [17] verified its applicability and reliability. Furthermore, the block decomposition technique allows the implementation of several vertical and horizontal cross-sections. Each implemented cross-section presents a different relationship between the subdomain computational and communication cost.

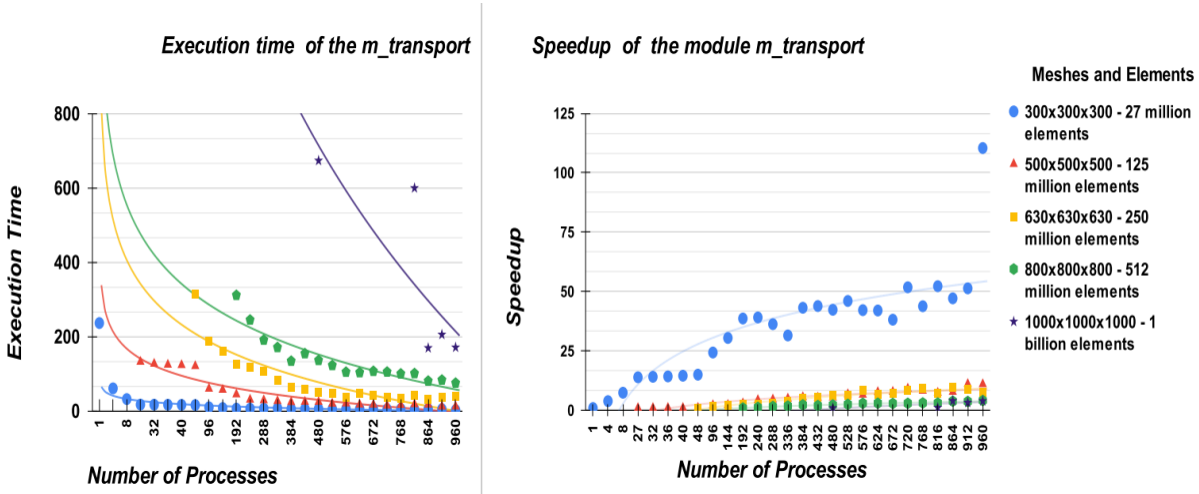

Figure 1. Execution Time and and Speedup for the fluid flow simulator,Scalability studies were presented in the article by Herrera et al. [16]

This new strategy uses numerical method stencil properties to identify the amount of data to be exchanged between processes and to group them in volume elements that have similar boundary conditions, in order to reduce the number of MPI messages. The Simulator stencil numerical method used in the present study, upwind Leveque [14], has six neighbors and is a first-order method with low complexity. The domain is divided into subdomains with the same amount of volume elements. Volume element data structure method classifies groups. One of them is called Central, which contains the inner volume elements of the simulation. The remaining groups are responsible for the method's fluid flow simulation boundary conditions. According to group volume element structure boundary condition, it is possible to assign MPI_PROC_NULL value for MPI messages for no neighboring subdomain, avoiding unnecessary messages Winkelmann et al. [19]. Also, it is possible to group face volume element information in order

to reduce the number MPI messages.

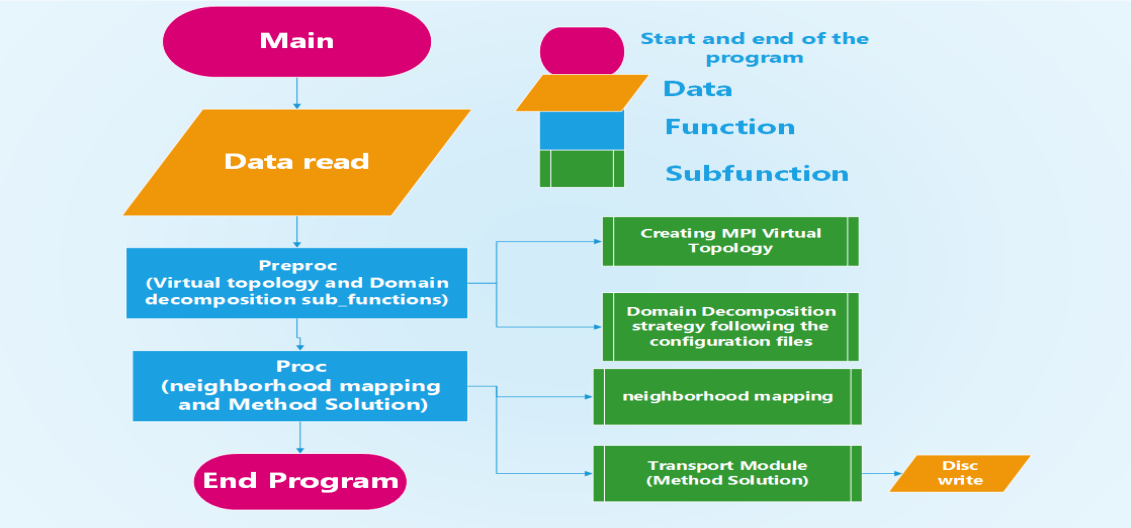

Figure 2. Code Fluxogram

Figure 2 presents current strategy flowchart. The algorithm starts with the main function, the data is read and the Preproc function creates the MPI virtual topology and decomposes the domain. Next, the Proc function maps the neighborhood volume elements and solves the method in the Transport module. This strategy divides the size of the mesh by the number of processes to generate all possible volume element mesh divisions to be executed. In an example of 48 processes for a 300x300x300 mesh, we have 4 volume element mesh division options (4x4x3, 6x4x2, 12x2x2, 12x4x1). In the example, the new strategy selects the division that presents the major amount of volume elements in one dimension in order to decrease communication cost.

As we increase the number of MPI processes, increasing the number of divisions in a single dimension the problem is that ultimately the code can compromise the computational cost of the numerical method per process by having only message exchange and almost no calculation. In order to keep the same computational cost of the subdomain we have to increase the number of divisions in others dimensions. The challenge is to find the best possible partitioning configuration, keeping an even balance between the computational cost and the communication cost. We implemented the following experiments in order to evaluate the best partitioning configuration for 300x300x300 mesh size for different number of MPI processes.

5 Results and analysis

The tests presented in this study were performed on the SDumont Supercomputer using 1 to 20 computational nodes, each composed of two Processors Intel(R) Xeon(R) Gold 6252 CPU running at 2.10GHz. Each processor has 24 physical cores per socket. Processor communication is through the INTEL UPI channel. The hyper-threading function was disabled. Since the present study evaluates MPI messages performance, we turned off the output write operations. We performed the tests in a typical size oil reservoir mesh composed of 300x300x300 volumes.

5.1 EXPERIMENT : single node performance evaluation

The purpose of this experiment is to verify whether we can improve the runtime of the simulator by decreasing the communication of MPI processes using the optimized domain decomposition technique presented in Section 4. We used one computational node (48 cores) of the SDumont supercomputer. The time measured for each test corresponds to an average of five runs.

The first column in Table 1 shows the current strategy which uses the number of processes to adopt all possible volume element mesh divisions to be executed. For 10 processes and for a 300x300x300 mesh, we have 9 volume element mesh division options (10x1x1, 5x2x1, 2x5x1, 1x10x1, 5x1x2, 1x5x2, 2x1x5,

Partition	nx	ny	nz	Runtime
10x1x1	30	300	300	86.900s
5x2x1	60	150	300	91.891s
2x5x1	150	60	300	88.676s
1x10x1	300	30	300	87.091s
5x1x2	60	200	1500	91.901s
1x5x2	300	60	150	88.166s
2x1x5	150	300	60	87.250s
1x2x5	200	150	60	88.321s
1x1x10	300	300	30	86.152s

Table 1. Domain decomposition strategy for a 300x300x300 mesh with 10 MPI processes.

1x2x5, 1x1x10). The second, third and fourth columns presents the number of volume elements subdomain for each dimension and last column presents total execution time for each test and the decomposition strategy used a single node.

Partition	nx	ny	nz	Runtime
6x4x1	50	75	300	10.58
6x2x2	50	150	150	10.82
12x2x1	25	150	300	1.01

(a) Results with 24 MPI processes

Partition	nx	ny	nz	Runtime
4x4x3	75	75	100	66.79
6x4x2	50	75	150	21.62
12x2x2	25	150	150	3.32
12x4x1	25	75	300	1.91

(b) Results with 48 MPI processes

Table 2. Domain decomposition strategy for a 300x300x300 mesh with 24 and 48 MPI processes

Table 2 shows the single node simulator total execution time for 24 MPI processes and for 48 MPI processes domain decomposition sections for the 300x300x300 mesh (27 million volume elements). We can see that the largest number of sections for one dimension presents a significantly shorter simulation runtime. Tables 2 and 3 omitted sections that presents same dimensional domain decomposition. For example, in table 2(a), decompositions 12x2x1, 12x1x2 and 2x1x12 presents the same amount of volume elements in each dimension, and have the same execution time. Therefore 12x2x1 and 12x1x2 have been omited from the table. In this case the decomposition strategy used two nodes (96 MPI processes) and 4 nodes (150 MPI processes);

5.2 EXPERIMENT: Multiple nodes performance evaluation

In this experiment, we ran the simulator with a mesh composed by 300x300x300 volume elements. We measured the runtime for all possible domain decomposition divisions from 2 nodes (96 MPI processes) to 4 nodes (150 MPI processes). The runtime for each test corresponds to an average of 5 runs. Results shows that the domain decomposition strategy based on a one-dimensional section presented the lowest execution time.

Above experiments shows that communication among MPI messages significantly affects the running time of the simulator. When looking for the all balanced workload domain division possibilities, one must choose the ones that generate less MPI messages.

Figure 3 presents simulator execution time for 24, 48, 96 and 150 MPI processes. Blue line represents

Partition	nx	ny	nz	Runtime
6x4x4	50	75	75	21.87
12x4x2	25	75	150	9.24

(a) Results for 96 MPI processes

Partition	nx	ny	nz	Runtime
6x5x5	50	60	60	24.07
15x10x1	20	30	300	4.36
25x6x1	12	50	300	3.84
30x5x1	10	60	300	4.17
50x3x1	6	100	300	3.11

(b) Results for 150 MPI processes

Table 3. Domain decomposition strategy for a 300x300x300 mesh with 96 and 150 MPI processes

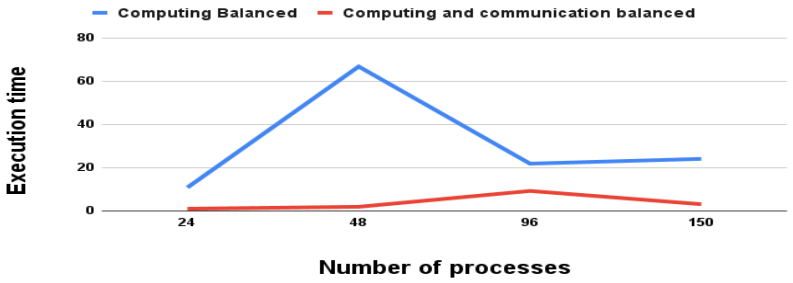

Figure 3. Computing balanced vs Computing and Communication balanced Execution time .

computing load balanced MPI processes domain decomposition (old strategy). Red line represents computing and communication load balanced MPI processes domain decomposition (new strategy). For a 48 cores computational node running one MPI processes per core, 48 MPI processes runs in one computing node, 96 runs in 2 computing nodes and 150 MPI processes runs in 4 computing nodes. We observe that for 48 MPI processes experiment, there is a huge execution time gap between the old strategy and the new strategy. The main reason is the number of MPI communication operation from old strategy domain decomposition. Finally, this experiment confirms that the new domain decomposition strategy decreases total simulator execution time.

6 Conclusions

The adoption of this domain decomposition strategy shows an improvement in simulator performance and scalability, this technique turned out to be very good for distributed memory architectures. However, domain decomposition with the MPI processes has limitations, especially in irregular domains due to the unbalanced workload causing the processes not to work together. Although our first tests are in domains of equal size, we intend to carry out studies with different reservoir domains. From the results presented, we already have an indication of the amount of information that must be exchanged between the MPI processes for each kind of partition adopted.

As future work, an algorithm will be implemented to find the best domain division for a given oil reservoir, this will include an analysis of the dimensions of the reservoirs, the kind of partitioning chosen and the amount of information to be exchanged between the MPI processes, such as way that we can find an efficient partition, yet preserving its scalability, balance for a given number of nodes. Furthermore, our research aims to develop a computational environment for academic studies in order to give scalability to new numerical methods developed in the area of reservoir engineering. Thus, we identified that this type of application is necessary to apply Parallel I/O searches to efficiently read the Computational mesh of the reservoir. We intend to better communicate the MPI processes using advanced techniques such as the adaptive MPI, based on the work presented in Huang et al. [20] in order to give greater scalability to the present work. Finally, the visualization of the large volume of data from the reservoirs is important for the calculation of the production process, so we will research large-scale visualization techniques such as the works presented in Lustosa et al. [21].

Acknowledgements. The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer, which have contributed to the research results reported within this paper. URL: http://sdumont.lncc.br . The authors also thank the Brazilian agency National Council for Scientific and Technological Development (CNPq, Portuguese) for the research support.

Authorship statement. Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] Dagan, G., 1989. Flow and transport in porous formations. Springer-Verlag.

[2] Durlofsky, L. J., 1991. Numerical calculation of equivalent grid block permeability tensors for heterogeneous porous media. vol. 27, n. 5, pp. 699–708.

[3] Gelhar, L. W. & Axness, C. L., 1983. Three-dimensional stochastic analysis of macrodispersion in aquifers. vol. 19, n. 1, pp. 161–180.

[4] Borges, M. R., Furtado, F., Pereira, F., & Souto, H. P. A., 2008. Scaling analysis for the tracer flow problem in self-similar permeability fields. Multiscale Modeling & Simulation, vol. 7, n. 3, pp. 1130–1147.

[5] Chen, Y., Durlofsky, L., Gerritsen, M., & Wen, X., 2003. A coupled local–global upscaling approach for simulating flow in highly heterogeneous formations. Advances in Water Resources, vol. 26, n. 10, pp. 1041–1060.

[6] Gonzaga de Oliveira, S. L., 2015. Introdução à geração de malhas triangulares. Sociedade Brasileira de Matemática Aplicada e Computacional, São Carlos.

[7] Tuane, V. L., 2012. Simulação numérica tridimensional de escoamento em reservátorios de petróleo heterogêneos. Master's thesis, LNCC/MCT, Petrópolis, RJ, Brasil.

[8] Forum, M., 1994. Mpi: A message-passing interface standard. Technical report, USA.

[9] Parashar, M. & Yotov, I., 1998. An environment for parallel multi-block, multi-resolution reservoir simulations. In Proceedings of the 11th International Conference on Parallel and Distributed Computing Systems (PDCS 98), Chicago, IL, International Society for Computers and their Applications (ISCA), pp. 230–235.

[10] LeVeque, R. J., 2007. Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems. SIAM.

[11] Correa, M. & Borges, M., 2013. A semi-discrete central scheme for scalar hyperbolic conservation laws with heterogeneous storage coefficient and its application to porous media flow. International Journal for Numerical Methods in Fluids, vol. 73, n. 3, pp. 205–224.

[12] Carneiro, I. B., Borges, M. R., & Malta, S. M. C., 2018. Aplicação de métodos de alta ordem na resolução de problemas bifásicos. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, vol. 6, n. 2.

[13] Wang, M., Xu, X., Ren, X., Li, C., Chen, J., & Yang, X., 2017. Mesh partitioning using matrix value approximations for parallel computational fluid dynamics simulations. Advances in Mechanical Engineering, vol. 9, n. 11, pp. 1687814017734109.

[14] Leveque, R. J., 1992. Numerical Methods for Conservation Laws. Birkhauser.

[15] Palin, M. F., 2007. Técnicas de decomposição de domínio em computação paralela para simulação de campos eletromagnéticos pelo método dos elementos finitos. PhD thesis, Universidade de São Paulo.

[16] Herrera, S., Ribeiro, W., Teixeira, T., Carneiro, A., Cabral, F., Borges, M., & Osthoff, C., 2020. Avaliação de desempenho no supercomputador sdumont de uma estratégia de decomposição de domínio usando as funcionalidades de mapeamento topológico do mpi para um método numérico de escoamento de fluidos. In Anais da VI Escola Regional de Alto Desempenho do Rio de Janeiro, pp. 31–35. SBC.

[17] Lima, I. d. C. M., 2017. Simulação de reservatórios de petróleo em paralelo utilizando malhas não-estruturadas 2D e 3D. PhD thesis, Universidade Federal do Ceará.

[18] Chen, Z., 2007. Reservoir simulation: mathematical techniques in oil recovery. SIAM.

[19] Winkelmann, R., Häuser, J., & Williams, R. D., 1999. Strategies for parallel and numerical scalability of cfd codes. Computer methods in applied mechanics and engineering, vol. 174, n. 3-4, pp. 433–456.

[20] Huang, C., Lawlor, O., & Kale, L. V., 2003. Adaptive mpi. In International workshop on languages and compilers for parallel computing, pp. 306–322. Springer.

[21] Lustosa, H., Porto, F., & Valduriez, P., 2019. Savime: A database management system for simulation data analysis and visualization. In SBBD 2019-Simpósio Brasileiro de Banco de Dados, pp. 1–12.