

Parallel Reservoir Simulation of Oil Flow Using PETSc and Implicit Schemes

Fábio Bertolino Vasconcellos¹, Mayksoel Medeiros de Freitas¹, Grazione de Souza¹, Helio Pedro Amaral Souto¹

¹*Polytechnic Institute, State University of Rio de Janeiro
Nova Friburgo, 25625-570, RJ, Brazil*

fabio.bertolino@iprj.uerj.br, mayksoel@iprj.uerj.br, gsouza@iprj.uerj.br, helio@iprj.uerj.br

Abstract. We employ the Portable, Extensible Toolkit for Scientific Computation (PETSc), a numerical library designed for high-performance parallel computing, to solve the algebraic systems arising from the discretization of the governing equations for slightly compressible two-dimensional oil flow in porous media. This computational step, being the most resource-intensive, is optimized through parallelization to reduce execution time while preserving solution accuracy. Our methodology integrates an implicit time-stepping scheme and Picard linearization to model reservoir pressure evolution and well production, including well-reservoir coupling. We perform a mesh refinement study and sensitivity analyses by varying key reservoir parameters and fluid properties. PETSc's parallel solvers yield substantial performance gains, validating the efficiency of our approach. This work highlights the applicability of PETSc in reservoir simulation and emphasizes the role of parallel computing in accelerating large-scale subsurface flow simulations.

Keywords: Implicit Time Integration, Parallel Computing, PETSc, Reservoir Simulation, Slightly Compressible Flow

1 Introduction

The numerical solution of engineering problems has been enabled by the advent of computers since World War II. Progress in computational capabilities has allowed for the study of increasingly complex scenarios, including subsurface flow simulation, where petroleum reservoir simulation is crucial for optimizing hydrocarbon recovery. This work is dedicated to reservoir simulation utilizing the Portable Extensible Toolkit for Scientific Computation (PETSc) [1], a library designed for high-performance parallel computing, to address the challenge of solving the large-scale algebraic systems inherent in these simulations.

Field-scale petroleum reservoir simulation [2] often requires solving nonlinear algebraic systems with millions of unknowns in three dimensions. This computational step consumes the majority of the simulation effort. Given that real-world reservoir simulations can take weeks even on high-performance machines [3], employing efficient numerical methods and parallelization techniques for solving these algebraic systems is essential for the oil and gas industry to rapidly evaluate different production scenarios and determine optimal operational plans.

To tackle these computationally intensive problems, libraries like PETSc [1] have been developed, providing efficient and flexible tools for solving systems of the form $\mathbf{Ax} = \mathbf{b}$. PETSc offers data structures and routines for large-scale applications, with built-in support for parallel execution using the *Message Passing Interface* (MPI). This facilitates the development of scientific computing codes in languages such as C, C++, Fortran, or Python, and allows for integration with existing software.

The PETSc library has been successfully applied in various engineering and scientific domains requiring high computational performance. Examples include geothermal system simulation [4], thermal energy storage [5], geodynamics [6], optical flow computation on GPU clusters [7], and as a parallel linear solver within reservoir simulators like TOUGH2 [8]. It has also been used in challenging problems such as cardiac electrophysiology [9], fluid dynamics [10], and fluid-structure interaction [11]. The coupling of PETSc with other libraries, such as ADOL-C for derivative evaluation [12] and HPDDM for advanced Krylov subspace and domain decomposition methods [13], further extends its capabilities. Its application in diverse areas like free surface flow simulation with over 100 million particles [14] and image processing [15], as well as in aerospace applications using Newton-Krylov-Schwarz methods [16], underscores its versatility and efficiency in parallel computing. The use of PETSc's parallel solvers for the pressure correction in multiphase flow simulations also highlights its importance in reducing

computational costs [17]. Several other works, including [18, 19], further demonstrate the broad applicability and impact of the PETSc library in computationally intensive simulations.

Given the scale and complexity of algebraic systems arising from the discretization of reservoir flow equations, particularly for field-scale simulations and those involving intricate well configurations, the computational demands can be substantial. Parallel computing offers a pathway to overcome these limitations by distributing the computational workload across multiple processors, significantly reducing execution time and enabling the simulation of more realistic and larger-scale scenarios. The PETSc is specifically designed for high-performance parallel computing, providing a robust suite of data structures and scalable numerical solvers optimized for distributed memory architectures. Its inherent parallel capabilities make PETSc an ideal choice for tackling the resource-intensive algebraic systems encountered in reservoir simulation, allowing for efficient exploration of different production strategies and detailed analysis of subsurface flow behavior.

2 Governing equation for single-phase oil flow in a reservoir

The mathematical modeling of isothermal single-phase oil flow in a petroleum reservoir, under the assumptions of a heterogeneous and isotropic permeability field, slightly compressible rock and Newtonian fluid, negligible chemical reactions and gravitational effects, and a two-dimensional laminar flow regime in the xy -plane, leads to a governing partial differential equation for the oil pressure. Starting from the mass conservation equation and incorporating Darcy's law (neglecting gravity), we arrive at [20]:

$$\frac{\partial}{\partial t} \left(\frac{\phi}{B} \right) - \nabla \cdot \left(\frac{\mathbf{k}}{B\mu_o} \nabla p \right) - \frac{q_{sc}}{V_b} = 0. \quad (1)$$

Here, \mathbf{k} is the absolute permeability tensor, p is the pressure, q_{sc} is a source term at standard conditions, and V_b is the total volume. The oil formation volume factor (B) and porosity (ϕ) are considered slightly pressure-dependent:

$$B = \frac{B^0}{1 + c_o(p - p^0)}, \quad \phi = \phi^0[1 + c_\phi(p - p^0)], \quad (2)$$

where B^0 and ϕ^0 are reference values, and c_o and c_ϕ are the compressibilities of oil and rock, respectively.

By applying the chain rule, the accumulation term can be rewritten, leading to the final governing equation for pressure:

$$\Gamma_p \frac{\partial p}{\partial t} - V_b \nabla \cdot \left(\frac{\mathbf{k}}{B\mu_o} \nabla p \right) - q_{sc} = 0, \quad (3)$$

where the pressure-dependent coefficient Γ_p is given by:

$$\Gamma_p = V_b \left(\frac{\phi c_o}{B^0} + \frac{\phi^0 c_\phi}{B} \right). \quad (4)$$

As a simplifying assumption, the viscosity was considered constant.

This non-linear partial differential equation for oil pressure requires appropriate initial and boundary conditions for its solution. The initial condition is a known initial pressure distribution $p(x, y, t = 0) = p_{ini}(x, y) = p_{ini}$. For the reservoir boundaries, length L_x and width L_y , a no-flow condition is applied:

$$\left(\frac{\partial p}{\partial x} \right)_{x=0, L_x} = \left(\frac{\partial p}{\partial y} \right)_{y=0, L_y} = 0. \quad (5)$$

2.1 Well-reservoir coupling

If the source term q_{sc} is used to represent the production rate via a well-reservoir coupling technique, it can be expressed as [20]:

$$q_{sc} = -J_w (p - p_{wf}),$$

where J_w is the productivity index and p_{wf} is the wellbore pressure. The numerical determination of the productivity index in reservoir simulation will be discussed in the numerical solution section. This coupling allows for the calculation of wellbore pressure if the production rate is prescribed, and vice versa. In this work, a prescribed production rate condition is employed.

3 Numerical solution

The governing partial differential equation for oil pressure is solved numerically using the Finite Volume Method and Picard iteration [20], employing a fully implicit time discretization and three-point centered differences for spatial derivatives [21]. This results in a system of non-linear algebraic equations.

For a two-dimensional problem, the discretized form of the governing equation is [21]:

$$\begin{aligned} & \mathbb{T}_{x,i+1/2,j}^{n+1} (p_{i+1,j}^{n+1} - p_{i,j}^{n+1}) - \mathbb{T}_{x,i-1/2,j}^{n+1} (p_{i,j}^{n+1} - p_{i-1,j}^{n+1}) + \mathbb{T}_{y,i,j+1/2}^{n+1} (p_{i,j+1}^{n+1} - p_{i,j}^{n+1}) \\ & - \mathbb{T}_{y,i,j-1/2}^{n+1} (p_{i,j}^{n+1} - p_{i,j-1}^{n+1}) = \Gamma_{i,j}^{n+1} (p_{i,j}^{n+1} - p_{i,j}^n) + (q_{sc})_{i,j}^{n+1}, \end{aligned} \quad (6)$$

where the transmissibilities in the x and y directions, $\mathbb{T}_{x,i\pm\frac{1}{2},j}^{n+1}$ and $\mathbb{T}_{y,i,j\pm\frac{1}{2}}^{n+1}$, incorporate reservoir geometry, permeability, fluid viscosity, and formation volume factor. Rock and geometry properties in transmissibilities are typically handled with harmonic averages, while fluid properties (for uniform grids) use arithmetic averages [21]. The transient term coefficient, $\Gamma_{i,j}^{n+1}$, is derived using a conservative expansion [20]:

$$\Gamma_{i,j}^{n+1} = \left[\frac{\phi^n c_o}{B^0} + \frac{\phi^0 c_\phi}{B^{n+1}} \right] \frac{V_{bi,j}}{\Delta t}. \quad (7)$$

To solve the resulting system of non-linear algebraic equations, Picard iteration is employed. This involves linearizing the transmissibility and source terms by using values from the previous iteration level (v) to solve for the current iteration level ($v+1$) [21]:

$$\begin{aligned} & \mathbb{T}_{x,i+\frac{1}{2},j}^{n+1,v} (p_{i+1,j}^{n+1,v+1} - p_{i,j}^{n+1,v+1}) - \mathbb{T}_{x,i-\frac{1}{2},j}^{n+1,v} (p_{i,j}^{n+1,v+1} - p_{i-1,j}^{n+1,v+1}) + \mathbb{T}_{y,i,j+\frac{1}{2}}^{n+1,v} (p_{i,j+1}^{n+1,v+1} - p_{i,j}^{n+1,v+1}) \\ & - \mathbb{T}_{y,i,j-\frac{1}{2}}^{n+1,v} (p_{i,j}^{n+1,v+1} - p_{i,j-1}^{n+1,v+1}) = \Gamma_{i,j}^{n+1,v} (p_{i,j}^{n+1,v+1} - p_{i,j}^n) + (q_{sc})_{i,j}^{n+1,v}, \end{aligned} \quad (8)$$

This linearized system is then solved iteratively utilizing the capabilities of the PETSc library to obtain the pressure distribution at each time step.

4 Determination of well productivity index

Well-reservoir hydrodynamic coupling is crucial for this study. For a given production rate, the wellbore pressure can be determined once the productivity index J_w and the reservoir pressure p are known. This section focuses on the transient hydrodynamic coupling, where the productivity index is time-dependent, Transient Well Index (TWI), due to the transient flow behavior near the wellbore.

The wellbore pressure p_{wf} is related to the production rate q_{sc} through the discretized coupling equation:

$$(q_{sc})_{i,j}^{n+1} = - (J_w)_{i,j}^{n+1} [p_{i,j}^{n+1} - (p_{wf})_{i,j}^{n+1}], \quad (9)$$

where the transient productivity index $(J_w)_{i,j}^{n+1}$ is given by [22]:

$$(J_w)_{i,j}^{n+1} = \left\{ 4\pi\kappa L_z \left[Ei \left(\frac{r_w^2}{4\eta_t t} \right) - Ei \left(\frac{r_{eq}^2}{4\eta_t t} \right) \right]^{-1} \right\}_{i,j}^{n+1} \quad (10)$$

Here, r_w is the wellbore radius, r_{eq} is the time-dependent equivalent radius, $\eta_t = k/(\phi c_t \mu)$, $k = \sqrt{k_x k_y}$, $c_t = c_o + c_\phi$, and Ei is the exponential integral function. The equivalent radius r_{eq} is determined using the Newton-Raphson method:

$$(r_{eq})_{i+1} = (r_{eq})_i - \frac{f(r_{eq}_i)}{f'(r_{eq}_i)}, \quad (11)$$

with for the hydrodynamic case being:

$$f(r_{eq}_i) = \frac{q_{sc}}{4\pi\kappa L_z} Ei \left(\frac{r_{eq}_i^2}{4\eta_t t} \right) - \Delta p, \quad f'(r_{eq}_i) = \frac{q_{sc}}{2\pi\kappa L_z r_{eq}} \exp \left(\frac{r_{eq}_i^2}{4\eta_t t} \right). \quad (12)$$

This leads to the iterative formula for the transient equivalent radius for the hydrodynamic coupling:

$$r_{eq_{i+1}} = r_{eq_i} \left[1 - \frac{q_{sc} E_i(\sigma) - 4\pi\kappa L_z \Delta p}{2q_{sc} \exp(\sigma)} \right], \quad (13)$$

where $\Delta p = p_{ini} - p_{i,j}$ ($p_{i,j}$ is the pressure in the cell containing the well), and $\sigma = (r_{eq_i}^2 / 4\eta t)$. The exponential integral $E_1(u) = -Ei(-u)$ is approximated using a polynomial fit for accurate evaluation over a wide range of arguments [22].

The formula for calculating the equivalent radius is given by the time required to reach the pseudo-steady-state regime. Based on the concept of dimensionless time and considering the normal area A , which represents the drainage area of the well in a developed reservoir [22],

$$t_{DA} = \frac{kt}{\phi\mu c_t A}. \quad (14)$$

Considering the well located at the center of the reservoir and that its drainage area has a square shape, for values of t_{DA} below 0.09, the reservoir's behavior is similar to that of an infinite system, indicating a transient regime in the porous medium. On the other hand, starting from $t_{DA} = 0.1$, the pseudo-steady-state regime is reached [22] and it is assumed that the equivalent radius no longer varies with time. The value of the product $\phi\mu$ was computed based on the initial pressure, assuming that this product remains approximately constant.

5 Numerical results

The numerical simulations were performed using PETSc 3.21 to solve the algebraic systems generated by the discretization of the governing equations. The computational work was conducted on a single node from a cluster, utilizing the following hardware and software configuration: Operating System OpenSUSE 15.5; Processor Intel(R) Xeon(R) Silver 4316 CPU @ 2.30GHz; Model R750; CPUs 40 (80 threads); and RAM 126 GB.

We began our analysis with a mesh refinement study to ensure the spatial discretization was sufficient. Subsequently, we conducted several sensitivity analyses to investigate the impact of key reservoir parameters and fluid properties on the simulation results. A summary of the parameters and properties utilized for the simulations can be found in Table 1, in which, Δ_{ini} the initial time step, which is successively multiplied by $F_{\Delta t}$ throughout the time steps until the final time step, Δ_{final} , is reached. The simulation ends when the final time, t_{final} , is reached.

Table 1. Rock, fluid, and geometry general parameters

Parameter	Value	Unit	Parameter	Value	Unit
B_0	1.3	m ³ /std m ³	q_{sc}	-1,000	m ³ /day
c_o	8.0×10^{-7}	kPa ⁻¹	t_{final}	100	day
c_r	5.0×10^{-7}	kPa ⁻¹	tol	1×10^{-6}	kPa
$F_{\Delta t}$	1.1	-	Δt_{ini}	0.0001	day
$k_x = k_y = k$	0.01	μm^2	Δt_{final}	5	day
$L_x = L_y$	1,000	m	μ^0	0.001	Pa · s
L_z	50	m	ρ_{ref}	840	kg/m ³
$p_{ini} = p_0$	69,000	kPa	$\phi_{ini} = \phi_0$	0.2	-

In the mesh refinement study, we not only verified numerical convergence but also highlighted the appearance of a numerical artifact resulting from the adopted well-reservoir coupling model. Five mesh configurations were employed: 41×41 (M1), 81×81 (M2), 161×161 (M3), 321×321 (M4), and 641×641 (M5). As a result, the 641×641 mesh was deemed sufficiently refined to ensure both numerical convergence and the acquisition of accurate results. Consequently, all subsequent sensitivity analyses were performed using this mesh.

5.1 Computational performance

In terms of physical CPU count, we started with one and progressively doubled this value to investigate if the performance improvement would tend toward a limiting value, as is typically the case in many parallel computing scenarios. An analysis of computational performance was conducted to evaluate the scalability of the proposed methodology using PETSc. Table 2 summarizes the execution times obtained for different physical processor counts using the most refined mesh.

Table 2. Computational Performance

Processors	1	2	4	8	16	32
Execution Time (s)	1291.53	813.08	462.21	266.50	178.66	175.05

This performance gain demonstrates the high efficiency of our parallel implementation. However, increasing the processor count from 16 to 32 resulted in only a marginal reduction in execution time, from 178.66 s to 175.05 s. This behavior indicates that the performance gain is approaching a saturation limit, a common phenomenon in parallel systems explained by Amdahl's Law [23]. Beyond 16 processors, the serial portion of the code and communication overhead begin to dominate the total execution time, thus limiting further scalability.

We also conducted a comparative analysis of computational performance by varying the number of processors (1, 16, and 32) and employing the five meshes used in the numerical convergence study. The results can be seen in Tables 3 to 5.

Table 3. Computational Performance for 1 Processor and 5 Meshes

Meshes	41×41	81×81	161×161	321×321	641×641
Execution Time (s)	1.203	5.259	28.58	180.22	1291.53

Table 4. Computational Performance for 16 Processors and 5 Meshes

Meshes	41×41	81×81	161×161	321×321	641×641
Execution Time (s)	5.046	7.107	13.09	36.72	178.66

Table 5. Computational Performance for 32 Processors and 5 Meshes

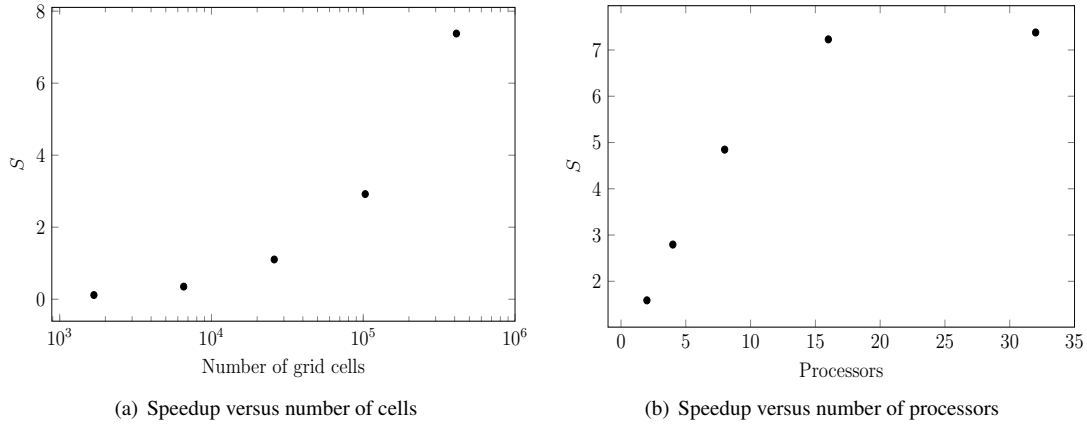
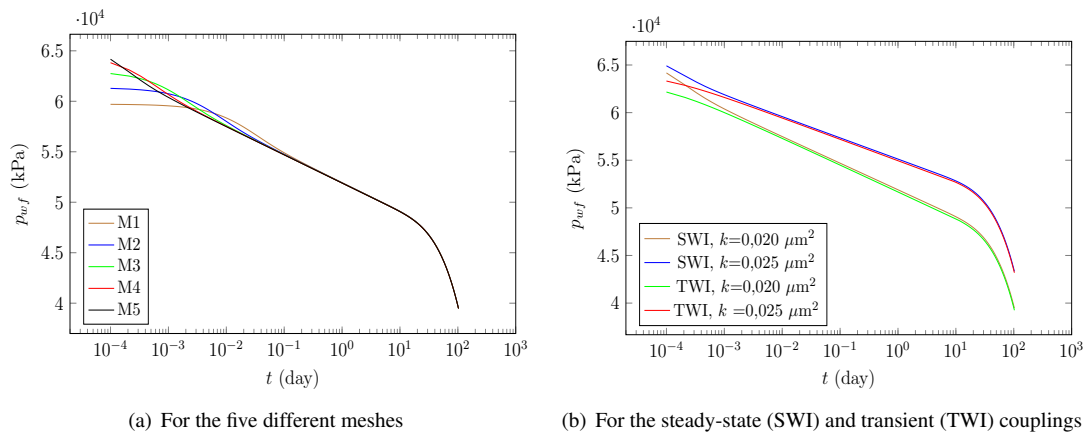
Meshes	41×41	81×81	161×161	321×321	641×641
Execution Time (s)	10.302	14.99	25.89	61.71	175.05

The computational performance results in Tables 3-5 show that the optimal processor configuration varies with mesh size. For coarser meshes (41×41 and 81×81), single processor performance is superior due to parallelization overhead. As mesh density increases (161×161 and 321×321), the 16-processor configuration becomes most efficient. Only for the finest mesh (641×641) does the 32-processor setup (175.05 s) slightly outperform 16 processors (178.66 s). Overall, except for the finest mesh case, 16 processors consistently provide the best performance, indicating an optimal balance between computational load distribution and communication overhead.

In Figure 1 we present the variation of Speedup values (S) as a function of the number of mesh cells (Figure 1(a)) obtained with 16 processors, as well as the respective values as a function of the number of processors (Figure 1(b)). As we can verify, there was a significant gain as the number of cells increased, reaching a maximum value close to eight. On the other hand, we observe that the speedup reaches its maximum value for a number of processors between 16 and 32. For this reason, we chose to perform the subsequent simulations of the sensitivity analysis using 16 processors.

5.2 Sensitivity analysis

Beyond computational performance, the focus of this work was also directed toward the implementation of the well-reservoir coupling model considering the transient regime. Therefore, Figure 2 presents the results for the variation of pressure in the producer well (p_{wf}) as a function of time. On the left, in Figure 2(a), we observe that as the meshes are refined, we successfully eliminated the undesirable effect of numerical storage [22], using the conventional well index [20]. The importance of calculating the Transient Well Index (TWI) can be verified when compared with the results obtained with the Steady Well Index (SWI), the conventional well index, Figure 2(b), for two different values of reservoir permeability and mesh M5. The coupling model considering steady-state conditions provides values that are overestimated for the initial time steps of production. However, as time progresses, we observe that the predictions provided by both models tend to yield practically identical values.


 Figure 1. Speedup (S) as a function of the number of cells and processors

 Figure 2. Pressure at the producer well (p_{wf}) as a function of time

6 Conclusions

The present study successfully integrated a Transient Well Index (TWI) formulation with the PETSc library to mitigate the numerical artifacts arising from our well-reservoir coupling model. By capitalizing on PETSc's parallel solvers, we achieved substantial reductions in computational time. Our scalability analysis confirmed that the speedup is directly proportional to the computational load and the number of processors utilized, though it inevitably encounters a plateau, a behavior consistent with theoretical parallel computing models. This research, therefore, validates the efficacy of our simulation approach and emphasizes the importance of parallel computing in tackling the challenges of large-scale subsurface flow simulation. Future work will focus on exploring alternative well-reservoir coupling systems and further refining the implementation to take advantage of more extensive parallel resources, such as a greater number of computational nodes.

Acknowledgements. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

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] Satish Balay et al. PETSc Web page, 2024.

- [2] M. Zhang, H. Yang, S. Wu, and S. Sun. Parallel multilevel domain decomposition preconditioners for monolithic solution of non-isothermal flow in reservoir simulation. *Computers & Fluids*, vol. 232, pp. 105183, 2022.
- [3] L. F. Werneck, M. M. Freitas, G. Souza, L. F. C. Jatobá, and H. P. A. Souto. An OpenMP parallel implementation using a coprocessor for numerical simulation of oil reservoirs. *Computational and Applied Mathematics*, vol. 38, pp. 1–30, 2019.
- [4] A. Croucher, M. O’Sullivan, J. O’Sullivan, A. Yeh, J. Burnnell, and W. Kissiling. Waiwera: A parallel open-source geothermal flow simulator. *Computers & Geosciences*, vol. 141, pp. 104529, 2020.
- [5] W. Wang, O. Kolditz, and T. Nagel. A parallel FEM scheme for the simulation of large scale thermochemical energy storage with complex geometries using PETSc routines. *Energy Procedia*, vol. 75, pp. 2080–2086, 2015.
- [6] R. Katz, M. Knepley, B. Smith, M. Spiegelman, and E. Coon. Numerical simulation of geodynamic processes with the portable extensible toolkit for scientific computation. *Physics of the Earth and Planetary Interiors*, vol. 163, n. 1, pp. 52–68, 2007.
- [7] S. Cuomo, A. Galletti, G. Giunta, and L. Marcellino. Toward a multi-level parallel framework on GPU cluster with PETSc-CUDA for PDE-based optical flow computation. *Procedia Computer Science*, vol. 51, pp. 170–179, 2015.
- [8] Y. Jung, G. S. H. Pau, S. Finsterle, and R. M. Pollyea. TOUGH3: A new efficient version of the TOUGH suite of multiphase flow and transport simulators. *Computers & Geosciences*, vol. 108, pp. 2–7, 2017.
- [9] E. Centofanti and S. Scacchi. A comparison of algebraic multigrid bidomain solvers on hybrid CPU–GPU architectures. *Computer Methods in Applied Mechanics and Engineering*, vol. 423, pp. 116875, 2024.
- [10] S. Kang, A. Dener, A. Hamilton, H. Zhang, E. M. Constantinescu, and R. L. Jacob. Multirate partitioned Runge–Kutta methods for coupled Navier–Stokes equations. *Computers & Fluids*, vol. 264, pp. 105964, 2023.
- [11] D. Boffi, F. Credali, L. Gastaldi, and S. Scacchi. A parallel solver for fluid–structure interaction problems with Lagrange multiplier. *Mathematics and Computers in Simulation*, vol. 220, pp. 406–424, 2024.
- [12] H. Bonart, S. Fillinger, E. Esche, G. Wozny, and J.-U. Repke. Source code generation for parallelized simulations of large-scale nonlinear equation systems on a supercomputer using MOSAIC, PETSc, and ADOL-C. In A. Espuña, M. Graells, and L. Puigjaner, eds, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pp. 2083–2088. Elsevier, 2017.
- [13] P. Jolivet, J. E. Roman, and S. Zampini. KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners. *Computers & Mathematics with Applications*, vol. 84, pp. 277–295, 2021.
- [14] X. Guo, B. D. Rogers, S. Lind, and P. K. Stansby. New massively parallel scheme for incompressible smoothed particle hydrodynamics (ISPH) for highly nonlinear and distorted flow. *Computer Physics Communications*, vol. 233, pp. 16–28, 2018.
- [15] L. Carracciolo, L. D’Amore, and A. Murli. Towards a parallel component for imaging in PETSc programming environment: a case study in 3-D echocardiography. *Parallel Computing*, vol. 32, n. 1, pp. 67–83, 2006.
- [16] P. D. Hovland and L. C. McInnes. Parallel simulation of compressible flow using automatic differentiation and PETSc. *Parallel Computing*, vol. 27, n. 4, pp. 503–519, 2001.
- [17] L. E. Clarke and G. Krishnamoorthy. Pre-conditioning strategies to accelerate the convergence of iterative methods in multiphase flow simulations. *Mathematics and Computers in Simulation*, vol. 165, pp. 200–222, 2019.
- [18] B. Mantravadi, P. Jagad, and R. Samtaney. A hybrid discrete exterior calculus and finite difference method for Boussinesq convection in spherical shells. *Journal of Computational Physics*, vol. 491, pp. 112397, 2023.
- [19] H. Zhang and E. M. Constantinescu. Optimal checkpointing for adjoint multistage time-stepping schemes. *Journal of Computational Science*, vol. 66, pp. 101913, 2023.
- [20] T. Ertekin, J. H. Abou-Kassem, and G. R. King. *Basic Applied Reservoir Simulation*. Society of Petroleum Engineers, Richardson, USA, 2001.
- [21] F. B. Vasconcellos, P. de Tarço Honório Jr., G. de Souza, and H. P. Amaral Souto. Simulação do escoamento bidimensional de óleo em um reservatório empregando a biblioteca numérica PETSc. *CALIBRE - Revista Brasileira de Engenharia e Física Aplicada*, vol. 9, n. 3, pp. 1–14, 2024.
- [22] R. Rosário, G. de Souza, and H. Amaral Souto. A comparative study of some well-reservoir coupling models in the numerical simulation of oil reservoirs. *International Journal of Advanced Engineering Research and Science*, vol. 7, n. 9, 2020.
- [23] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2018.