GMRES based numerical simulation and parallel implementation of multicomponent multiphase flow in porous media

Saltanbek T. Mukhammadzhanov¹, Danil V. Lebedev², Nurislam M. Kassymbek¹*, Timur S. Imankulov³, Bazargul Matkerim¹ and Darkhan Zh. Akhmed-Zaki¹,²

Abstract: This article considered the numerical simulation of multicomponent multiphase flow in porous media. The resulting system of nonlinear equations linearized by the Newton-Raphson method and solved with the iterative Generalized minimal residual method (GMRES) algorithm. To achieve better convergence, we used the ILU(0) preconditioner to the GMRES algorithm. As a result, we used a completely implicit scheme called the Newton-ILU0-GMRES algorithm to solve the problem of interest. Based on the obtained sequential algorithm, we implemented a parallel algorithm using Message Passing Interface (MPI) technology. Additionally, we made comparisons between the parallel program of the presented algorithm and the parallel program using the ready-made Portable Extensible Toolkit for Scientific Computation (PETSc) library. We developed an MPI parallel algorithm and tested it on the MVS-10P supercomputer of the Interdepartmental Supercomputer Center of the Russian Academy of Sciences.

Subjects: Computer science; Numerical simulation; Linear algebra; parallel programming

Keywords: multicomponent multiphase flow; Newton-Raphson method; GMRES; MPI technology

1. Introduction
The development of high-performance computing (HPCwire: Global News on High Performance Computing, 2020) enables the solving of major scientific and applied problems in various fields. For example, supercomputers can be loaded with weather and climate forecasting tasks, acoustic
tasks, hydrodynamics, medical preparations manufacturing tasks, and biological research (Akhmed-Zaki et al., 2017; Farrukh, 2018; Yeshmukhametov et al., 2017).

Using supercomputers can significantly accelerate the solution of problems when using numerical methods. One of these tasks is forecasting oil and gas production in specific oil and gas fields. Modelling multicomponent multiphase fluid (oil and gas) flow in porous media (in oil reservoirs) is relevant and, at the same time, a complex problem of hydrodynamic simulation. To solve such problems, various methods and schemes are used (Aceto et al., 2006; Ahmed, 2006; Borisov et al., 2013; Chen, 2006; Chen et al., 2006; Edwards et al., 2018; Javidi & Ahmad, 2013; Imankulov et al. 2018; Zheng & Yin, 2014), some of which are iterative methods for solving linear systems (Lacroix et al., 2003; Mittal & Al-Kurdi, 2002; Vabishchevich & Vasilyeva, 2011; Wang et al., 2013).

To speed up scientific calculations, various technologies are used, such as Open Multi-Processing (OpenMP) (Iryanto & Gunawan, 2017; Werneck et al., 2016), Message Passing Interface (MPI) (Akhmed-Zaki et al., 2019; Pannaia et al., 2003), Open Computing Language (OpenCL) (Khramchenkov & Khramchenkov, 2018), Compute Unified Device Architecture (CUDA) (McClure et al., 2014; Zaza et al., 2016) and fragmented programming (Akhmed-Zaki et al., 2019; Malyskhin & Perepelkin, 2011).

In this paper, the equation system of multicomponent multiphase flow in porous media, which is linearized by the Newton-Raphson method is solved, and at each iteration for solving the linear equation system, the generalized minimal residual method (GMRES) (Saad & Schultz, 1986) is used with the ILU(0) preconditioner. The method, called Newton-ILU(0)-GMRES, is the combination of the Newton-Raphson method and GMRES algorithm with ILU0 preprocessing to develop a reliable and efficient method. A parallel implementation of the considered problem is realized by the MPI programming interface, which is the main tool in the development of parallel programs for supercomputers. In the future, a project to develop an active knowledge management system to solve the multicomponent fluid flow problem is planned (Akhmed-Zaki et al., 2019). The result of this project will be a system that will automatically choose the a solution method, one of which is the method described in this article, for practical problems.

2. Mathematical model and numerical method
The one-dimensional problem of compositional fluid flow in a porous medium is considered. Compositional flow involves multiple components, and the one-dimensional problem of compositional fluid flow in a porous medium is considered. Compositional flow involves multiple components and three phases, and there is a mass transfer between the vapour and liquid phases. The model has the following assumptions:

- the flow process is isothermal,
- the components form at most three phases (e.g., vapour, liquid, and water),
- there is no mass interchange between the water and the hydrocarbon (vapour, liquid) phases.
- the diffusion/dispersion effect is negligible.

We describe the compositional model that has been widely used in the petroleum industry because the general compositional model is very difficult to solve.

At point A, water is pumped in, which, due to the difference in pressure through porous media, moves to point B. Liquid/vapor is pumped from point B as shown in Figure 1. The distribution of water, vapour, and liquid in the porous medium should be calculated.

Let us consider a mathematical model of the multicomponent flow of three-phase fluid in a porous medium. We write the law of conservation of mass for each component (Chen, 2006; Chen et al., 2006):
\[
\frac{\partial (\phi \xi_m \xi_w)}{\partial t} + \nabla \cdot (\xi_w u_w) = q_w, \quad \frac{\partial (\phi [x_m \xi_m \xi_o + x_m \xi_m \xi_g])}{\partial t} + \nabla \cdot (x_m \xi_m \xi_o + x_m \xi_m \xi_g \lambda) = q_m
\]  
(1)

\[u_w = \frac{-k_m}{\mu} (\nabla p_o - \alpha \nabla z), \quad \alpha = \omega, \alpha, g.
\]  
(2)

\[\sum_{m=1}^{N_c} x_m = 1, \quad \sum_{m=1}^{N_c} x_{mg} = 1.
\]  
(3)

\[S_o + S_a + S_g = 1.
\]  
(4)

\[p_{cow} = p_o - p_{cw}, \quad p_{cgo} = p_g - p_o.
\]  
(5)

\[f_{m_1}(p_o, x_{1o}, x_{2o}, \ldots, x_{N_c}) = f_{m_2}(p_g, x_{1g}, x_{2g}, \ldots, x_{N_c}),
\]
\[f_{m_3} = \rho \xi_m \phi_{m_2}, \quad m = 1, 2, \ldots, N_c, \quad \alpha = \omega, \alpha, g.
\]  
(6)

where \(S_o, S_a, \) and \(S_g\) are the saturations of the water, oil and gas phase, respectively; \(x_m\) is the mole fraction of component \(m\) in phase \(\alpha\); \(\xi_\alpha\) is the molar density of phase \(\alpha\); \(\phi\) and \(k\) are the porosity and permeability, respectively, of the rock; \(\mu\) is the viscosity; \(p_o\) is the phase pressure; \(p_{cow}\) and \(p_{cgo}\) are the capillary pressures; \(k_m\) is the relative phase permeability; \(u_w\) is the velocity of phase \(\alpha\); \(q_w\) and \(q_m\) are the molar velocities of the water flow and \(m\)-th component, respectively; \(f_{m_1}\) and \(f_{m_2}\) are the fugacity functions; and \(\phi_{m_2}\) is the fugacity coefficient of component \(m\) in phase \(\alpha\).

Distribution of chemical components in the hydrocarbon phase described by the K-value method (Pederson & Christensen, 2008). The thermodynamic behaviour of fluids under reservoir conditions are described by the Peng-Robinson equations of state (Peng & Robinson, 1976).

The system of nonlinear Equations (1)—(6) is linearized by the Newton-Raphson method.

In the \((l+1)\)-th layer of the Newton iterative process, the value of the unknowns is updated by the following law (Chen et al., 2006):

\[
\begin{align*}
\Delta x_{m_1}^{l+1} & = x_{m_1}^{l+1} - x_{m_1}^l, \quad \Delta x_{m_1}^{l+1} = x_{m_1}^{l+1} - x_{m_1}^l, \quad m = 1, 2, \ldots, N_c - 1, \\
\Delta Z_{m}^{l+1} & = Z_{m}^{l+1} - Z_{m}^l, \quad \Delta Z_{m}^{l+1} = Z_{m}^{l+1} - Z_{m}^l, \quad m = 1, 2, \ldots, N_c - 1, \\
\Delta L_{m}^{l+1} & = L_{m}^{l+1} - L_{m}^l, \quad \Delta L_{m}^{l+1} = L_{m}^{l+1} - L_{m}^l, \\
\Delta F_{m}^{l+1} & = F_{m}^{l+1} - F_{m}^l, \quad \Delta F_{m}^{l+1} = F_{m}^{l+1} - F_{m}^l, \\
\Delta S_{m}^{l+1} & = S_{m}^{l+1} - S_{m}^l, \quad \Delta S_{m}^{l+1} = S_{m}^{l+1} - S_{m}^l, \\
\Delta p_{m}^{l+1} & = p_{m}^{l+1} - p_{m}^l, \quad \Delta p_{m}^{l+1} = p_{m}^{l+1} - p_{m}^l.
\end{align*}
\]  
(7)
The resulting system of linear Equations (7) is reduced to the following form, as shown by Chen et al. (2006):

\[ Ax = b \] (8)

There are several well-known methods for solving systems of algebraic equations of this type; direct methods, such as the Gauss method or LU decomposition, are not suitable for systems with sparse matrices since there is the possibility of overflow (Higham, 2011). The Conjugate gradient method (CG) is designed for systems with symmetric matrices, and the Biconjugate Gradient Method (BiCG) has slow convergence (Van der Vorst, 2003). In this work, system (8) was solved by the iterative method GMRES, which is a widely used Krylov subspace method (Saad, 2003).

To reduce the number of iterations in the computation, preconditioning was used as an explicit or implicit modification of the system of linear equations, making it possible to simplify the solution. One of the widely used methods for finding the preconditioner is incomplete LU (ILU) decomposition of the original matrix \( A \) (Mittal & Al-Kurdi, 2003). We choose ILU(0) (Chow & Saad, 1997) as a preconditioner for the GMRES algorithm, which was found by decomposing matrix \( A \) by a fine-grained ILU factorization algorithm. As a result, for the numerical solution of the given problem for solving a system of linear equations, (8) uses the Newton-ILU0-GMRES algorithm as a fully implicit scheme.

3. Parallel algorithm

To implement the parallel algorithm of the GMRES, we need to determine the parts of the calculations that can be the basis of parallelization.

For instance, the GMRES algorithm consists of the following steps:

(a) Initialization. Choose \( x_0 \), compute \( r = b - Ax_0 \), solve \( Pw = r \) and compute \( v_1 = \frac{w}{\|w\|_2} \)

(b) Arnoldi iteration. Iterate \( m \) times: \( j = 1, 2, ..., m \). The specific Arnoldi algorithm is as follows:

\[ \text{for } i = 1, ..., m \]
\[ Pw = Ax_i \]
\[ \text{for } k = 1, ..., i \]
\[ h_{kj} = \langle w, v_k \rangle \]
\[ w = w - h_{kj}v_k \]
\[ \text{end} \]

\[ h_{i+1;j} = \|w\|_2 \]
\[ v_{j+1} = \frac{w}{h_{i+1;j}} \]
\[ \text{end } i \]

(a) Compute the approximate solution \( x^m = x^0 + Q_{n,m}y_m \). Here, minimizes \( \|r - H_{m,m}y\|_2 \); \( Q_{n,k} \) is an \( n \times k \) matrix, the column vector of the matrix is composed of \( v_1, v_2, ..., v_k \) orthogonal vectors, and \( H_{mm} \) is the Hessenberg matrix.

(b) Determination. Compute \( r^m = b - x^m \), stop if the condition is met; otherwise let \( x^0 = x^m \) and go back to a) and recalculate. The convergence condition by which the method stops is given by an arbitrary number \( \varepsilon \).

In this GMRES algorithm, there are common operations such as matrix-matrix multiplication, matrix-vector multiplication, vector-vector multiplication and vector norm calculation in steps of
a), b) and c). These operations could be implemented in parallel with data decomposition. In a) and b) steps of the GMRES algorithm, P is used as the ILU(0) preconditioner.

4. Existing libraries
The GMRES algorithm can be implemented on some well-known libraries, such as Intel MKL (Intel® Math Kernel Library Developer Reference, 2020), MATLAB (MathWorks, 2020), Portable, Extensible Toolkit for Scientific Computation (PETSc) (Abhyankar et al., 2018). In this work, the PETSc library was chosen for comparisons with our Newton-ILU0-GMRES MPI parallel implementation since this tool implements parallel versions of different methods for solving linear and nonlinear equations.

During the review, it was revealed that it was impossible to use the ILU(0) preconditioner in a parallel program with PETSc; by default, the Jacobi block preconditioner is used instead.

5. Testing results of the numerical experiment

5.1. Verification of the convergence of the algorithm with preconditioning
The testing of the program was carried out on matrices corresponding to the cases when the number of grid point schemes varied from 500 to 8000. The parameters of the matrix are shown in Table 1.

From this table, the size of the matrix being solved is 6 times larger than the number of grid points. For example, the size of matrix #5, which corresponds to the case when the number of points of the difference grid is 8000, is 48000 x 48000 because 6 Equations (7) are written for each point.

Problem (8) was solved by the Newton-GMRES algorithm without a preconditioner and with the ILU(0) preconditioner. The graph of the change in the solution residuals with the execution of a certain number of iterations for various cases is shown in Figure 2.

As seen in Figure 2, for all the matrices tested, the Newton-ILU(0)-GMRES algorithm converges in less than three times the number of iterations than the Newton-GMRES algorithm. Based on this result, further tests were conducted only on the Newton-1LU (0)-GMRES algorithm.

5.2. Comparison of the Newton-ILU(0)-GMRES algorithm with the PETSc library
To compare our parallel program of Newton-ILU(0)-GMRES algorithm and the program using the PETSc library, tests were carried out on a personal 8-core computer, since it was not possible to install third-party libraries on a supercomputer. The results are shown in the following Table 2.

The tests were carried out on a 3000×3000 matrix, which corresponds to the case when the number of nodes of the difference scheme of (1)—(6) was 500. As we can see from Table 2, in parallel calculations, the implementation of the Newton-ILU(0)-GMRES algorithm runs faster than the program using PETSc. The comparisons shown above are not under the same conditions, as different preconditioners were used in different parallel implementations. However, the

| Table 1. Parameters of the matrix |
|----------------------------------|
| **Matrix designation** | #1  | #2  | #3  | #4  | #5  |
| Number of grid points | 500 | 1000 | 2000 | 4000 | 8000 |
| Size of matrix A       | 3000×3000 | 6000×6000 | 12000×12000 | 24000×24000 | 48000×48000 |
implementation of the presented Newton-ILU(0)-GMRES algorithm is faster than the PETSc version on the same personal computer.

### 5.3. Test results of the Newton-ILU(0)-GMRES parallel algorithm

Test runs for the considered problem (8) were made on the MVS-10P supercomputer of the Interdepartmental Supercomputer Center of the Russian Academy of Sciences, which includes nodes with two Xeon X5450 processors and 8 GB of RAM for each node. The test results are shown in Table 3.

As we can see from the test results, the parallel program running on 8 processors achieved the shortest runtime. As the size of the task increases, the parallel application achieves the shortest calculation time running on 64 processors, but it is not much different from the runtime of a parallel program running on 8, 128 and 256 processors. The reason is because with an increase in the number of processors, the computation time on each processor decreases, and the time spent on communications increases.

It can be seen that the runtimes on 64 and 128 processors are about the same. This finding is explained by the fact that on 64 processors, the communication time is less than the computation time, and on 128 processors, the computation time is less than the communication time, but in total, the time consumed is almost the same.

For matrix #1 on 256 and 512 processors, the parallel program is slower than the sequential program. With an increase in the size of the matrix, for instance, for matrices #2—#5, the runtime
of the parallel program decreases compared to the sequential program. This regularity makes it possible to predict that with an increase in the size of the matrix, in large numbers of processors, acceleration also increases. Figure 3 shows the parallel program acceleration. It can be concluded that it makes sense to run the parallel algorithm of the method on a large number of processors but only with large of matrix that cannot fit in the memory of a single node.

On matrix #1 with 256 and 512 processes, the parallel program considers it slower than the sequential one. With an increase in the size of the matrix for instance, on matrix #2—matrix #5, the runtime of the parallel program decreases compared to the sequential one. This regularity makes it possible to predict that with an increase in the size of the matrix, in large numbers of processes, acceleration also increases. Figure 3 shows the parallel program acceleration. It can be concluded that the parallel algorithm of the method makes sense to run on a large number of processors, only on large sizes that cannot fit in the memory of one node.

6. Conclusion
This research examines the parallel implementation of the Newton-ILU0-GMRES algorithm for a multicomponent three-phase flow problem. The considered problem is compared to the parallel implementation of the presented method and parallel program with the PETSc library on an 8-core personal computer. The results have shown that the presented Newton-ILU(0)-GMRES algorithm achieves faster convergence than the Newton-GMRES algorithm and has a shorter runtime than the parallel program with PETSc. Tests were also carried out with different node sizes and numbers of processes on supercomputers. The study showed that on a large number of processes, the MPI program runs better in large-scale tasks. In future research, we plan to consider other variants of

---

**Table 3. The runtimes of the parallel program in MPI**

| Matrix Number designation of processors | #1    | #2    | #3    | #4    | #5    |
|-----------------------------------------|-------|-------|-------|-------|-------|
| 1                                       | 3.74  | 48.38 | 334.24| 1688.74| 20219.34|
| 8                                       | 1.19  | 7.57  | 53.82 | 394.74| 2942.02 |
| 64                                      | 2.59  | 12.03 | 64.16 | 389.68| 2565.59 |
| 128                                     | 2.95  | 12.66 | 65.25 | 395.58| 2569.01 |
| 256                                     | 4.39  | 17.29 | 82.75 | 450.28| 2798.32 |
| 512                                     | 11.53 | 42.56 | 202.01| 866.98| 4185.07 |

---

**Figure 3. Acceleration of the parallel algorithm.**

---

**Speed-up**

---

**Figure 3. Acceleration of the parallel algorithm.**

---

**Page 7 of 10**
the method to improve the efficiency of the parallel algorithm. In particular, we plan to consider other options instead of the Arnoldi iteration and develop a fragmented algorithm (Akhmed-Zaki et al., 2019) that will allow users to not to spend time on the implementation of communications.

**Funding**

This work was supported by the Science Committee of the Ministry of Education and Science of the Republic of Kazakhstan [Grant No. AP05130366].

**Author details**

Saltanbek T. Mukhambetzhanov¹
E-mail: mukhambetzhanov_v@mail.ru
Danil V. Lebedev²
E-mail: danil.lebedev.0881@gmail.com
Nurislam M. Kassymbek⁴
E-mail: nuryslam.qassymbek@gmail.com
Timur S. Imankulov¹
E-mail: imankulov_ts@mail.ru
Bazargul Matkerim³
E-mail: bazargul.m@gmail.com
Darkhan Zh. Akhmed-Zaki¹
E-mail: darhan_zl@mail.ru

¹ Computer science, Al-Farabi Kazakh National University, Almaty, 050040, Kazakhstan.
³ Computer science, University of International Business, Almaty, 050010, Kazakhstan.

**Citation information**

Cite this article as: GMRES based numerical simulation and parallel implementation of multicomponent multiphase flow in porous media, Saltanbek T. Mukhambetzhanov, Danil V. Lebedev, Nurislam M. Kassymbek, Timur S. Imankulov, Bazargul Matkerim & Darkhan Zh. Akhmed-Zaki, Cogent Engineering (2020), 7: 1785189.

**References**

Abhyankar, S., Brown, J., Constantinescu, E., Ghosh, D., Smith, B., & Zhang, H. (2018). PETScTS: A modern scalable ODE/DAE solver library. Preprint ANL/MCS-P5061-0114.

Aceto, L., Pandolfi, R., & Trigiante, D. (2006). One parameter family of linear difference equations and the stability problem for the numerical solution of ODEs. Advances in Difference Equations, 019276(2006).

https://doi.org/10.1155/ADE/2006/19276

Akhmed-Zaki, D. Z. (2009). Reservoir engineering handbook (3rd ed.). Gulf Professional Publishing.

Akhmed-Zaki, D., Lebedev, D., Malyshkin, V., & Perelpetkin, V. (2019). Automated construction of high performance distributed programs in LuNA system. In V. Malyshkin (Ed.), Parallel Computing Technologies - 15th International Conference, PaCT 2019, Proceedings (Vol. 11657 LNCS, pp. 3–9). (Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer-Verlag GmbH and Co. KG. https://doi.org/10.1007/978-3-030-25636-4_1

Akhmed-Zaki, D. Z., Daribayev, B. S., Imankulov, T. S., & Turor, O. N. (2017). High-performance computing of oil recovery problem on a mobile platform using CUDA technology. Eurasian Journal of Mathematical and Computer Applications, 2(2), 4–13. https://doi.org/10.32523/2306-3172-2017-5-2-4-13

Borisy, V. E., Kritsky, B. V., Marchenko, N. A., Mitrushkin, D. A., & Savenchuk, E. B. (2013). Non-isothermal compositional flow model with chemical reactions and active solid phase for reservoir simulation. Preprint IPM im. M. V. Keldysha, 091, (pp. 32).

Chen, Z. (2006). Reservoir simulation: Mathematical techniques in oil recovery. Society for Industrial and Applied Mathematics.

Chen, Z., Huan, G., & Ma, Y. (2006). Computational methods for multiphase flows in porous media. Society for Industrial and Applied Mathematics.

Chow, E., & Saad, Y. (1997). Ilus: An incomplete LU preconditioner in sparse skyline format. International Journal for Numerical Methods in Fluids, 25(7), 739–748. https://doi.org/10.1002/sicn.1036-1997(19971015)25:7<739::AID-FLD581>3.0.CO;2-Y

Edwards, L., Dhanpat, D., Chakraborti, D. P., & Barigou, M. (2018). Hydrodynamics of three phase flow in upstream pipes. Cogent Engineering, 5(1), 1433983. https://doi.org/10.1080/23311916.2018.1433983

Farrukh, A. (2018). On the high performance computing for motif discovery in dna sequences. International Journal of Advanced Research, 6(14), 880–887. https://doi.org/http://dx.doi.10.21474/IJAR01/7437

Higham, N. (2011). Gaussian elimination. Wiley Interdisciplinary Reviews: Computational Statistics, 3(3), 230–238. https://doi.org/10.1002/wics.164

HPWire: Global News on High Performance Computing (HPC). (2020). Retrieved May 6, 2020, from https://www.hpcwire.com/

Imankulov, T. S., Akhmed-Zaki, D. Z., Daribayev, B. S., Lebedev, D. V., Aidarov, K. A., Narukhov, Y. S., & Turor, O. N. (2018). Intellectual system for analyzing thermal compositional modeling with chemical reactions. ECOR XVI - 16th European Conference on the Mathematics of Oil Recovery (Vol. 2018, pp. 1–9). European Association of Geoscientists & Engineers https://doi.org/10.3997/2214-4609.201802181

Intel® Math Kernel Library Developer Reference. (2020, March 19). Retrieved May 6, 2020, from https://software.intel.com/en-us/articles/mkl-reference-manual

Iryanto, I., & Gunawan, P. H. (2017). An OpenMP parallel godunov scheme for 1D two phase oil displacement problem. 5th International Conference on Information and Communication Technology (ICOiCT), IEEE. https://doi.org/10.1109/ICoICT.2017.8074664

Javid, M., & Ahmad, B. (2013). Numerical solution of fractional partial differential equations by numerical Laplace inversion technique. Advances in Difference Equations, 2013(1), 375. https://doi.org/10.1186/1687-1847-2013-375

Khramchenkov, E., & Khramchenkov, M. (2018). Numerical model of two-phase flow in dissolvable porous media and simulation of reservoir acidizing. Natural Resources Research, 27(4), 531–537. https://doi.org/10.1007/s11053-018-9371-x

Lacroix, S., Vassilevski, Y. U., Wheeler, J., & Wheeler, M. (2003). Iterative solution methods for modeling multiphase flow in porous media fully implicitly. SIAM Journal on Scientific Computing, 25(3), 905–922. https://doi.org/10.1137/S10648275024443X

Malyshkin, V. E., & Perelpetkin, V. A. (2011). LuNA fragmented programming system, main functions and peculiarities of run-time subsystem. Proceedings of the 11th International Conference on Parallel Computing Technologies. https://doi.org/10.1007/978-3-642-23178-0_5

MathWorks. (2020). Bioinformatics toolbox: User's guide. Retrieved May 06, 2020, from www.mathworks.com/help/pdf_doc/bioinfo/bioinfo Ug.pdf
McClure, J. E., Prins, J. F., & Miller, C. T. (2014). A novel heterogeneous algorithm to simulate multiphase flow in porous media on multicore CPU-GPU systems. Computer Physics Communications, 185(7), 1865–1874. https://doi.org/10.1016/j.cpc.2014.03.012

Mittal, R. C., & Al-Kurdi, A. H. (2002). LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. Computers & Mathematics with Applications, 43(2), 391–403. https://doi.org/10.1016/S0898-1221(03)00154-8

Pannala, S., D’Azevedo, E., Syamlal, M., & O’Brien, T. (2003). Hybrid (OpenMP and MPI) parallelization of MFIX: a multiphase CFD code for modeling fluidized beds. SAC ’03: Proceedings of the 2003 ACM symposium on Applied computing, (pp. 199–206). Association for Computing Machinery. https://doi.org/10.1145/952532.952574

Pederson, K. S., & Christensen, P. L. (2008). Phase behavior of petroleum reservoir fluids. CRC Press.

Peng, D. Y., & Robinson, D. B. (1976). A new two-constant equation of state. Industrial and Engineering Chemistry Fundamentals, 15(1), 59–64. https://doi.org/10.1021/i160057a011

Saad, Y. (2003). Iterative methods for sparse linear systems (2nd ed.). SIAM.

Saad, Y., & Schults, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7(3), 856–869. https://doi.org/10.1137/0907058

Vabischevich, P. N., & Vasilyeva, M. V. (2011). Iterative methods for solving the pressure problem at multiphase filtration. ArXiv. Abs/1107.5479.

Van der Vorst, H. (2003). Iterative Krylov methods for large linear systems (Cambridge monographs on applied and computational mathematics). Cambridge University Press. https://doi.org/10.1017/CBO9780511611511

Wang, B., Wu, S., & Li, Q. (2013). Applications of BILU-GMRES in reservoir numerical simulation. Acta Petroleini Sinica, 34(5), 954–958. https://www.doi.org/10.7623/syxb201305017

Werneck, L. F., Medeiros de Freitas, M., Guaraldi da Silva Jr, H., de Souza, G., & Amaral Souto, H. P. (2016, November 69). An OpenMP parallel implementation for numerical simulation of gas reservoirs using Intel Xeon Phi coprocessor. In S. M. Avila (Ed.), Proceedings of the XXXVII Iberian Latin-American Congress on Computational Methods in Engineering. ABMEC.

Yeshmukhametov, A., Kalimoldayev, M., Mamyrbayev, O., & Amirgaliev, Y. (2017). Design and kinematics of serial/parallel hybrid robot. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR) (pp. 162–165). IEEE. https://doi.org/10.1109/ICCAR.2017.7942679

Zaza, A., Awotunde, A. A., Fairag, F. A., & Al-Mouhamed, M. A. (2016). A CUDA based parallel multi-phase oil reservoir simulator. Computer Physics Communications, 206(1), 2–16. https://doi.org/10.1016/j.cpc.2016.04.010

Zheng, G., & Yin, J. (2014). Numerical approximation for a time optimal control problems governed by semi-linear heat equations. Advances in Difference Equations, 2014(1), Article 94. https://doi.org/10.1186/1687-1847-2014-94
