EURASIAN JOURNAL OF MATHEMATICAL AND COMPUTER APPLICATIONS
ISSN 2306–6172
Volume 4, Issue 1 (2016) 12–31

LARGE-SCALE SIMULATION OF OIL RECOVERY BY SURFACTANT-POLYMER FLOODING

D.Zh. Akhmed-Zaki, T.S. Imankulov, B. Matkerim, B.S. Daribayev, K.A. Aidarov, O.N. Turar

Abstract This article describes a hydrodynamic model of collaborative filtering of oil, water, surfactant and polymer in porous media for enhanced oil recovery, which takes into account the influence of temperature, polymer and surfactant concentration changes on water and oil viscosity. For the mathematical description of oil displacement process by polymer and surfactant injection in a porous medium, we used the balance equations for the oil and water phase, the transport equation of the polymer / surfactant / salt and heat transfer equation. Also, consider the change of permeability for an aqueous phase, depending on the polymer adsorption and residual resistance factor. Presented the numerical implementation for solving this problem and realized a high-performance hybrid parallelization using MPI and CUDA technology. The results of the numerical investigation on three-dimensional domain are presented and distributions of pressure, saturation, concentrations of polymer / surfactant / salt and temperature are determined. Visualization of simulation results are performed using the ray casting algorithm implemented using CUDA technology. With this method, it is real to draw large amount of data described by three-dimensional models containing up to several million polygons.

Key words: EOR, HPC, CUDA, 3D Visualization

AMS Mathematics Subject Classification: 65D25, 65D18, 65M06, 68W10

1 Introduction

Current positive trend of computer hardware environment level as well as increasing performance of high-performance systems across the world led to improving of problem solving efficiency in the area of fundamental and engineering studies. In this context, mathematical and computer modeling of subterranean oil reservoirs, development and analysis of fields became applicable areas of high-performance computing technologies. Majority of global and Kazakhstani oil companies use hydrodynamic simulators during oil field studies. From year to year hydrodynamic simulators becoming more accurate and complex allowing more detailed studies of physical and chemical properties of subterranean fluids. At the same time, modern numerical methods for solving oil related problems in complex domains being developed and overall scale of computations enlarge. Thereby, relevance of the use of the high-performance computing and computing technologies in oil industry growing every year.

Surfactant and polymer flooding methods are widely used in petroleum industry to enhance oil recovery. Polymer increases viscosity of a water thereby improving the

mobility ratio and increasing the recovery efficiency. Primary benefit of polymer flooding is to improve sweep efficiency and acceleration of oil production [1, 2]. Surfactant flooding method involves addition of surface-active agents or surfactants to the injected water. Surfactants reduces the interfacial tension between oil and water within reservoir, reduce the residual oil saturation and improve displacement efficiency [3].

Purpose of given work is to develop computational algorithms for solving three-dimensional surfactant/polymer flooding (SPF) problem and their implementation in the form of sequential programs with further development of corresponding parallel algorithms. Parallel algorithms can be implemented using MPI and CUDA technologies.

To show expedience of parallel implementation of computational algorithm by comparing execution times of sequential and parallel version through calculation of speedup and efficiency.

In order to analyze results obtained during simulation of computational algorithm necessary to develop visualization software with ability to render large-scale data arrays representing complex domains. For this purpose necessary to present visualization software allowing to render sample data based on real oil fields.

The paper consists of 7 sections including introduction and conclusion. Second section discusses mathematical model of the SPF problem. Section 3 briefly explains numerical method using explicit scheme. Sections 4 and 5 describe parallel algorithms using MPI and CUDA technologies respectively along with analysis of comparison of sequential and parallel implementations of algorithm. Sixth section introduces high-performance visualization module for large-scale data using various development approaches.

2 Mathematical Model of Surfactant/Polymer Flooding (SPF)

The mathematical model of two-phase flow in porous media has following assumptions:

- incompressible flow;
- gravitational forces and capillary effects is neglected;
- two-phase flow (water, oil) obeys Darcy's law;
- oil phase permeability does not depend on polymer adsorption.

Complexity of mathematical model of proposed problem justified by following considerations:

- effect of the concentration on viscosity of water phase;
- effect of the temperature effects on viscosity of water and oil phases;
- taking into account salinity of oil reservoir;
- taking into account effect of adsorption of polymer on permeability of water phase;

Mass conservation equations for each phase can be written as follows:

$$m\frac{\partial S_w}{\partial t} + div(v_w) = q_1 \tag{1}$$

$$m\frac{\partial S_o}{\partial t} + div(v_o) = q_2 \tag{2}$$

$$S_w + S_o = 1$$

where m - porosity, S_w,S_o-water and oil saturations, q_1, q_2 - source or sink, $\vec{v_w}, \vec{v_o}$ - velocity of the water and oil phases. The phase flux from Darcy's law is:

$$v_i = -K_0\frac{f_i(s)}{\mu_i}\nabla P, \qquad i = w, o \tag{3}$$

$f_i(s), \mu_i$ - relative permeability and viscosity for phase i, K_0 -permeability tensor.

Polymer, surfactant, salt and heat transport equations are given by [3, 4, 5]:

$$m\frac{\partial}{\partial t}(c_p s_w) + \frac{\partial a_p}{\partial t} + div(v_w c_p) = div(mD_{pw}s_w\nabla c_p) \tag{4}$$

$$m\frac{\partial}{\partial t}(c_{sw}s_w + c_{so}s_o) + \frac{\partial a_{surf}}{\partial t} + div(v_w c_{sw} + v_o c_{so}) = div(mD_{sw}s_w\nabla c_{sw} + mD_{so}s_o\nabla c_{so}) \tag{5}$$

$$m\frac{\partial}{\partial t}(c_s s_w) + div(v_w c_s) = 0 \tag{6}$$

$$\frac{\partial}{\partial t}[(1-m)C_r\rho_r + m(C_w s_w\rho_w + C_o s_o\rho_o)T] + div(\rho_w C_w v_w) + div(\rho_o C_o v_o) = = div[((1-m)\lambda_0 + m(\lambda_1 s_w + \lambda_2 s_o))\nabla T] \tag{7}$$

where c_p, c_s - polymer and salt concentrations in aqueous phase, c_{sw}, c_{so} - surfactant concentration in water and oleic phases, a_p, a_{surf} - polymer and surfactant adsorption functions, D_pw, D_sw, D_so - polymer and surfactant diffusion coefficients, C_w, C_o, C_r - specific heat of water, oil and rock, ρ_w, ρ_o, ρ_r - density of water, oil and rock, $\lambda_0, \lambda_1, \lambda_2$ - coefficients of thermal conductivity.

The modified Flory-Huggins equation [6] is used to calculate phase viscosities:

$$\mu_a = \mu_w[1 + (\gamma_1 c_p + \gamma_2 c_p^2 + \gamma_3 c_{sw} + \gamma_4 c_{sw}^2)c_s^{\gamma_5} - \gamma_6(T - T_p)] \tag{8}$$

$$\mu_o = \mu o_o[1 - \gamma_7(T - T_p)] \tag{9}$$

where $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7$ - constants.μo_o - initial viscosity of oelic phase, T_p - reservoir temperature. The imbibition relative permeability curve for water/oil flow is given by

$$f_w(S_w) = S_w^{3.5}; \qquad f_o(S_w) = (1 - S_w)^{3.5} \tag{10}$$

The adsorbed concentration of polymer is a function of polymer concentration, given by:

$$a = \frac{bc_p}{1 + bc_p} \tag{11}$$

where b - Langmuir constant.

Permeability reduction factor R_k is modeled as [7]

$$R_k = 1 + (R_{RF} - 1)a \tag{12}$$

where R_{RF}- residual reduction factor. Initial conditions:

$$\begin{gathered} s_w|_{t=0} = s_{w0}, c_{pw}|_{t=0} = c_{p0}, a_p|_{t=0} = a_{p0}, c_s|_{t=0} = c_{s0}, T|_{t=0} = T_p \\ c_{sw}|_{t=0} = c_{sw0}, c_{so}|_{t=0} = c_{so0}, a_{surf0}|_{t=0} = a_{surf0} \end{gathered} \tag{13}$$

Boundary conditions:

$$\begin{gathered} \frac{\partial s_w}{\partial n}|_{\partial\Omega} = 0; \frac{\partial P}{\partial n}|_{\partial\Omega} = 0; \frac{\partial T}{\partial n}|_{\partial\Omega} = 0; \\ \frac{\partial c_{pw}}{\partial n}|_{\partial\Omega} = 0; \frac{\partial c_{sw}}{\partial n}|_{\partial\Omega} = 0; \frac{\partial c_s}{\partial n}|_{\partial\Omega} = 0; \end{gathered} \tag{14}$$

Pressure equation obtained by adding (1) and (2):

$$div(\vec{v_w}) + div(\vec{v_o}) = 0 \tag{15}$$

3 Numerical Method

Considered mathematical model describes displacement of oil by polymer and surfactant driven by water. Injected surfactant solution pushed to reservoir by the conventional water injection. Polymer solution usually injected after surfactant solution to control the slug and improve volumetric sweep efficiency. Polymer solution in turn followed by normal water injection. It is necessary to calculate distributions of pressure, phases saturation, chemical concentrations and temperature.

A finite difference scheme employed to solve the SPF problem numerically on a 3D domain (Figure 1). Initial data for the SPF problem is given by (13). For the fully explicit problem, a time stepping algorithm and the Jacobi method used. The elements of each three-dimensional array represent pressure, saturation, polymer/surfactant/salt concentrations and temperature at points of the cube. In the Jacobi method calculation of each element depends on neighbor element values (Figure 2) [8].

$$U_{ijk}^{n+1} = G(U_{(i+1jk)}^n, U_{i-1jk}^n, U_{ij+1k}^n, U_{ij-1k}^n, U_{ijk+1}^n, U_{ijk-1}^n) \tag{16}$$

For instance, after converting the pressure equation (15) to the finite difference

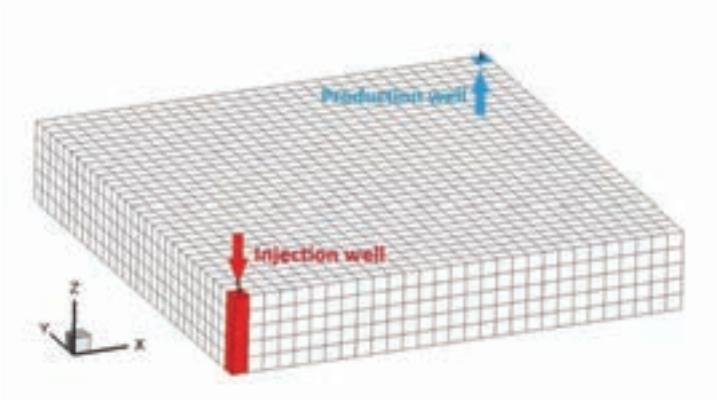


Figure 1: Calculation domain and well locations

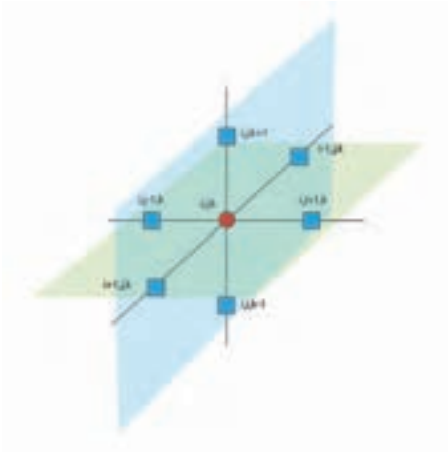


Figure 2: Numerical scheme of Jacobi method

form, the following expression will be obtained:

$$(M_x)_{i+\frac{1}{2}jk}\frac{P^n_{i+1jk}-P^{n+1}_{ijk}}{h_1^2}-(M_x)_{i-\frac{1}{2}jk}\frac{P^{n+1}_{ijk}-P^n_{i-1jk}}{h_1^2}+$$
$$(M_y)_{ij+\frac{1}{2}k}\frac{P^n_{ij+1k}-P^{n+1}_{ijk}}{h_2^2}-(M_y)_{ij-\frac{1}{2}k}\frac{P^{n+1}_{ijk}-P^n_{ij-1k}}{h_2^2}+$$
$$(M_z)_{ijk+\frac{1}{2}}\frac{P^n_{ijk+1}-P^{n+1}_{ijk}}{h_2^2}-(M_z)_{ijk-\frac{1}{2}}\frac{P^{n+1}_{ijk}-P^n_{ijk-1}}{h_2^2}=0 \quad (17)$$

With a few arithmetical operations it can be easily obtained:

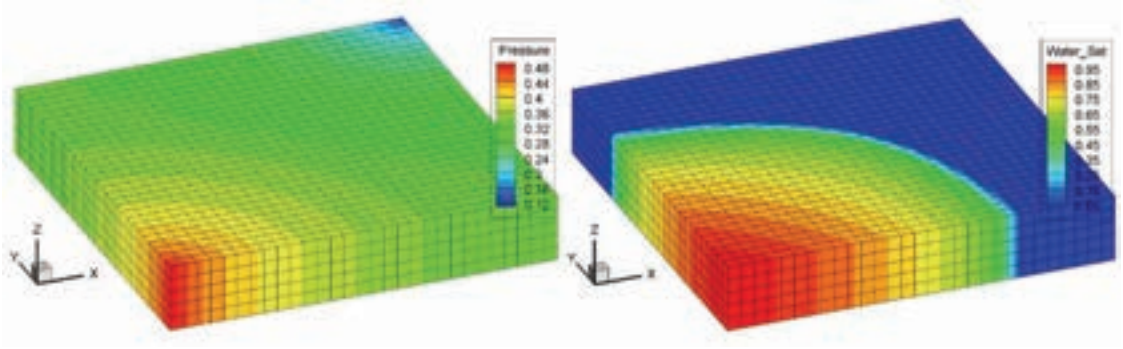
$$P^{n+1}_{ijk}=\frac{MPs}{Ms} \quad (18)$$

$$MPs=(M_x)_{i+\frac{1}{2}jk}P^n_{i+1jk}+(M_x)_{i-\frac{1}{2}jk}P^n_{i-1jk}+(M_y)_{ij+\frac{1}{2}k}P^n_{ij+1k}+$$
$$(M_y)_{ij-\frac{1}{2}k}P^n_{ij-1k}+(M_z)_{ijk+\frac{1}{2}}P^n_{ijk+1}+(M_z)_{ijk-\frac{1}{2}}P^n_{ijk-1} \quad (19)$$

$$Ms=(M_x)_{i+\frac{1}{2}jk}+(M_x)_{i-\frac{1}{2}jk}+(M_y)_{ij+\frac{1}{2}k}+(M_y)_{ij-\frac{1}{2}k}+(M_z)_{ijk+\frac{1}{2}}+(M_z)_{ijk-\frac{1}{2}} \quad (20)$$

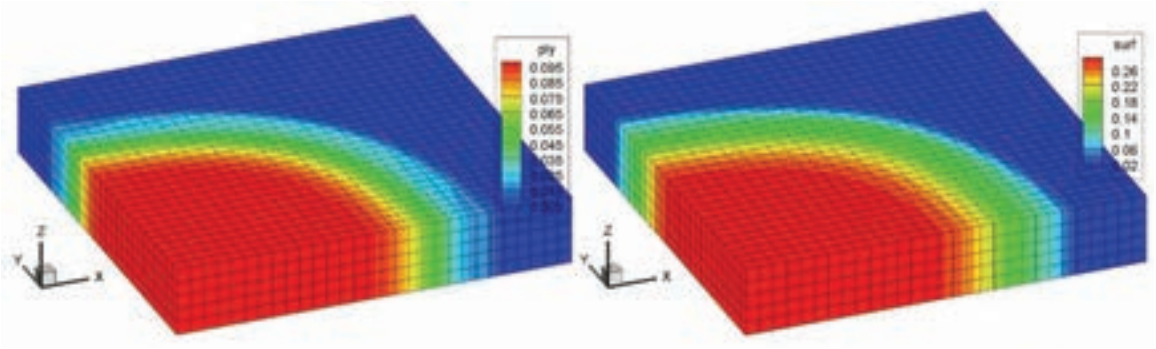
The pressure calculated until the condition below fulfilled

$$\sqrt{\sum_{i=1}^{Nx-1}\sum_{j=1}^{Ny-1}\sum_{k=1}^{Nz-1}\left(P_{ijk}^{n+1}-P_{ijk}^{n}\right)^2}<\varepsilon \tag{21}$$



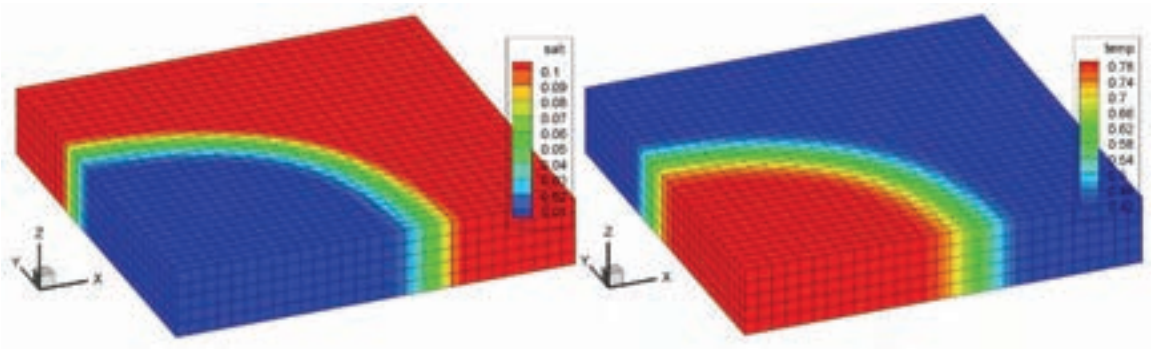
(a) (b)

Figure 3: Distribution of (a) pressure and (b) water saturation



(a) (b)

Figure 4: Distribution of (a) polymer concentration and (b) surfactant concentration



(a) (b)

Figure 5: Distribution of (a) salt concentration and (b) temperature

The other parameters of SPF problem employ the Jacobi method as pressure. The calculation of SPF problem organized in the following order:

- distribution of pressure;
- saturation (by known distribution of pressure);
- distribution of salt and polymer and surfactant concentrations;
- distribution of temperature in reservoir;
- water phase viscosity, depending on salt and polymer concentrations;
- water phase permeability considering the polymer adsorption.

Results of numerical calculations for non-isothermal oil displacement shown in the Figures 3-5.

4 Parallel Algorithm Using MPI

The following parallel algorithm used to solve the SPF problem defined by (1) - (15) equation systems with Jacobi method: first, divide original domain into subdomains through 3D decomposition shown in Figure 6. Each subdomain consists of three parts: ghost slab, boundary slab and interior slab. Interior elements belonging to a task are independent from other tasks, when boundary elements are dependent upon neighbor task's data and require communication. Further computation requires boundary slab values that stored in ghost slabs of neighbor subdomains, which are located in neighbor processors (i.e. ghost slabs store copies of their neighbor's boundary slab values). As a result, at each iteration processors exchange their boundary slab values with their left, right, south, north, top, or bottom neighbors (Figure 7) and recalculate their interior values. Termination conditions set by user and iterations carried out until they satisfied.

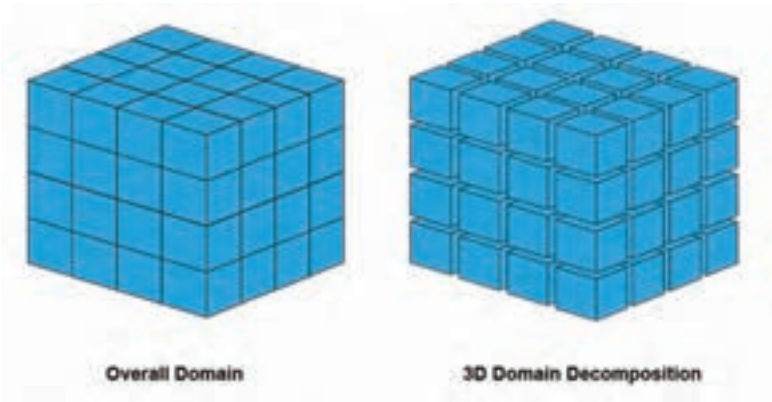


Figure 6: The 3D domain decomposition of original computation domain

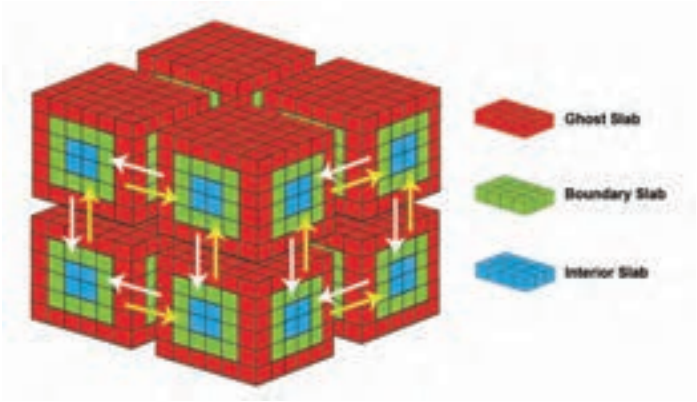


Figure 7: The scheme of data communication of subdomain

Figure 9 represents activity diagram of MPI algorithm in UML where each processing element compute SPF problem independently. Figure 10 shows activity diagram of SPF problem in each processing element according to parallel computation algorithm.

As initial data to estimate developed parallel algorithm used several parameters characterizing chosen problem, and hardware characteristics of HPC cluster. The HPC cluster named T-cluster [9] located within campus represents multiprocessor computational system with distributed memory consisting of 26 high-performance computational nodes connected with InfiniBand network reaching 20 Gb/sec throughput. As computational nodes of computing cluster 26 computing blades containing two-and six-core processors (Intel(R) Xeon (R) CPU E5645 2.40GHz) each are used. The total amount of RAM is 624 Gb, the total disk storage is 20 Tb. Calculations for parallel algorithm made using MPI technology on T-cluster and taking into account parameters obtained during testing of given system with Linpack and Intel MPI benchmark packages [10].

As values characterizing quality of the parallel algorithm used coefficients of speedup $S_p = T_1/T_p$ and efficiency $E_p = S_p/p$. Value T_1 characterizing execution time of the sequential algorithm, and T_p —execution time of the algorithm using p computational cores. When constructing model to estimate efficiency of constructed algorithm considered that execution time of sequential algorithm made up of time, required by processor

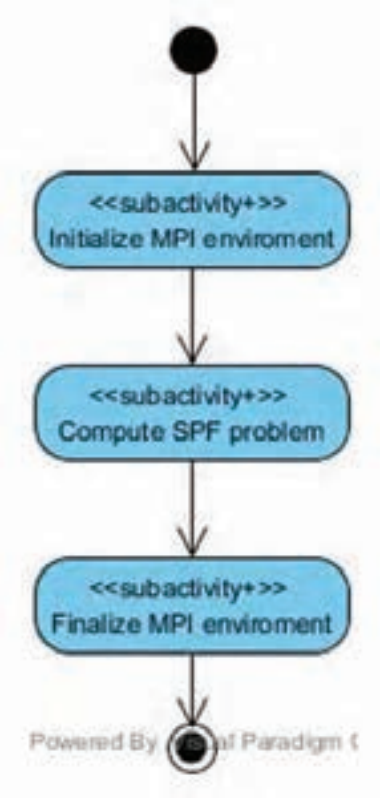


Figure 8: Activity diagram of MPI algorithm in UML

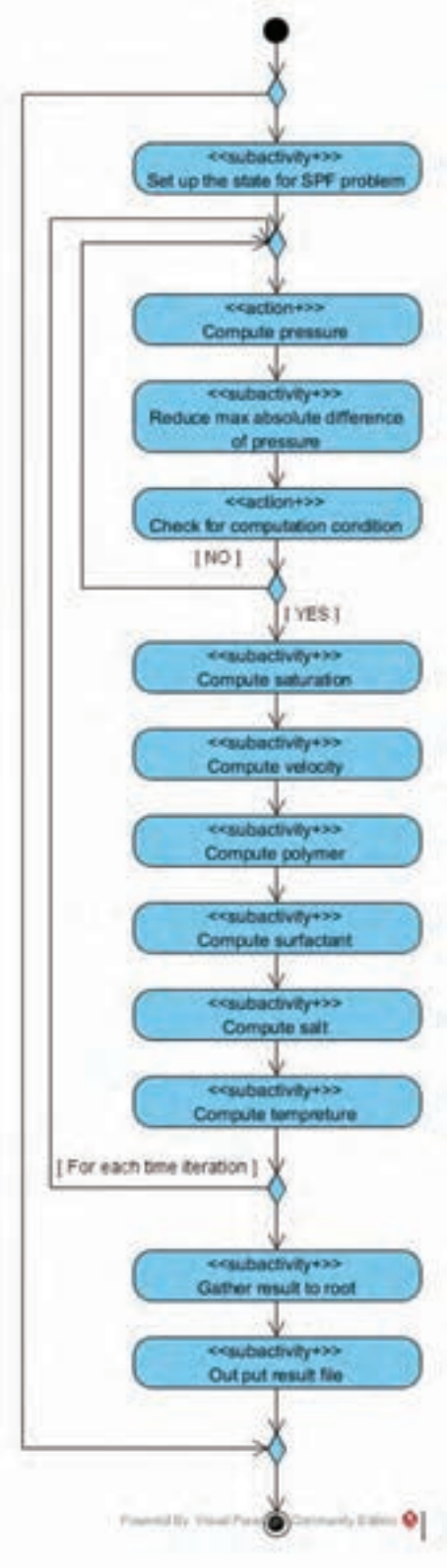


Figure 9: Activity diagram of parallel computation of SPF problem in each processing element

to perform actual computations, as well as time, which spent on reading data, necessary for computations from main memory to cache memory. For parallel algorithm additional expenses on organization of parallel computations (creation and closing parallel sections) and necessity to exchange data between cluster nodes also considered.

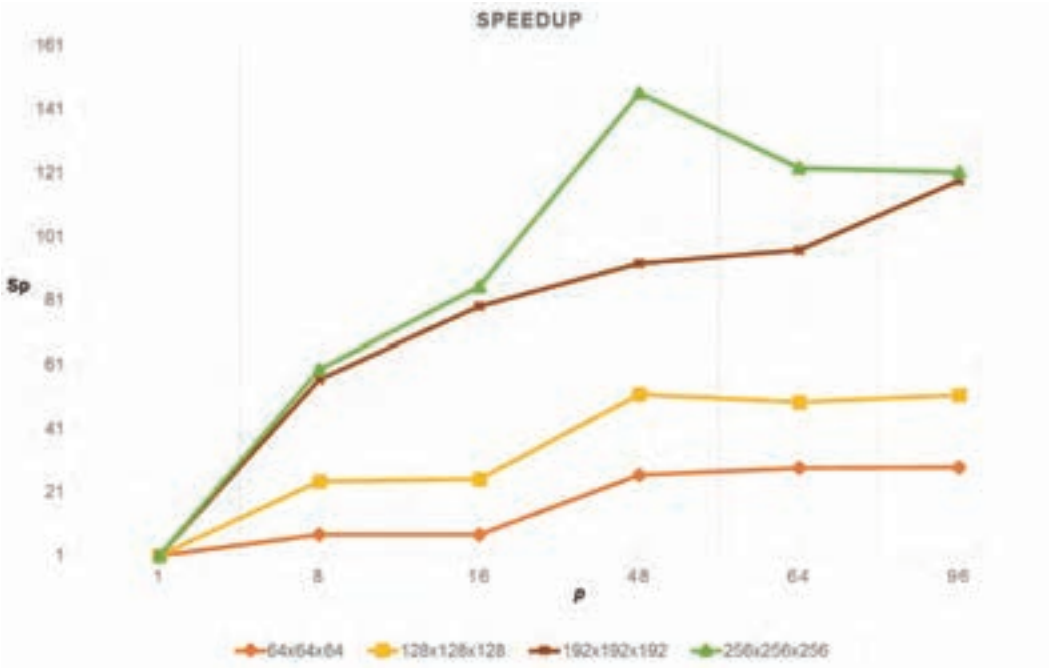


Figure 10: Change of speedup S_p values depending on $N_x \times N_y \times N_z$ and p

Characteristics entered previously used to estimate quality of parallel algorithm. Figure 10 shows change of value of speedup S_p depending on $N_x \times N_y \times N_z$ size of domain and p number of computational cores.

Figure 11 shows change of value of efficiency E_p characterizing average portion of execution time of the algorithm, during which processors actually used for solving the problem, depending on size of the problem $N_x \times N_y \times N_z$ and number of computational cores p.

5 Parallel Algorithm Using CUDA

The CUDA [11] technology used on graphics processing device in order to develop and test three-dimensional problem. As a tested hardware platform used personal computer with NVIDIA GeForce GTX550Ti graphics card. Given graphics device consists of 192 CUDA cores, each of cores work on 900 Mhz clock rate and has 1 Gb memory [12] (Figure 12).

It is well known, that CUDA - architecture of parallel computations from NVIDIA - allowing essentially increase computational performance due to use of GPUs. In recent years, sales of CUDA processors reached millions, and software developers, scientists and engineers widely using CUDA in different fields, including processing video and images, computational biology and chemistry, modeling of fluid dynamics, recovering

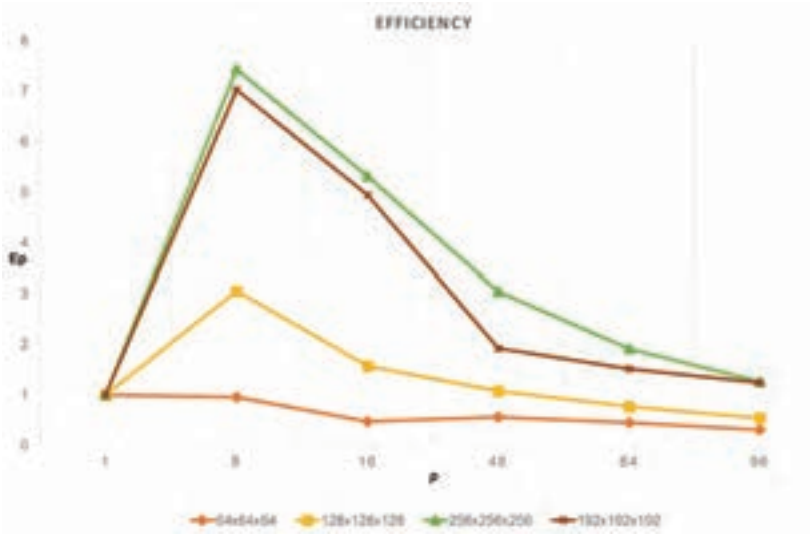


Figure 11: Change of efficiency E_p depending on $N_x \times N_y \times N_z$ and p

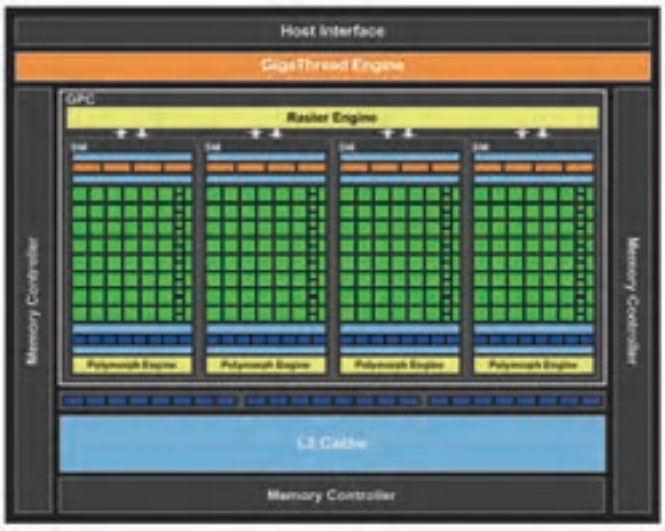


Figure 12: Fermi Architecture for NVIDIA GeForce GTX550Ti

Table 1: Computation times (seconds) of parallel and sequential algorithm implementations

Program / problem size	$32 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
Parallel	0.007	0.055	0.449
Sequential	0.547	26.938	205.443

images, acquired using computer tomography, seismic analysis, ray tracing and many more.

At parallel implementation of the SPF problem on graphics cards, used three-dimensional decomposition of a domain. When decomposing the domain it is necessary to split the grid onto blocks where each block can copy data into shared memory. Hence, each block node conducting calculation and saving calculated data in global memory. To calculate in each subdomain it is necessary to use data from neighbor subdomain, i.e. necessary to copy boundary data from global memory. Afterwards size of subdomain will increase (Figure 7).

Given problem optimized in two stages. First stage is when data threads of subdomain's internal cells copied into shared memory, then boundary node values copied from global memory. In given case size of subdomain does not change. This algorithm is effective due to low access delay of shared memory. Second stage is to get around of issue related to repeated copying of the data on the subdomain boundary from global memory which cannot be avoided. In the mentioned case columns and rows on subdomain boundaries are copied. Therefore it is necessary to change three-dimensional representation of an array to single dimensional. Thereby it is possible to exclude repeated copying of columns.

Results of implemented parallel algorithm for the SPF problem provided in this section. The size of the single computational block taken as 512 (8x8x8) threads considering usage of the shared memory. After comparing computation times taking into account grid size it becomes clear that on both devices parallel algorithm vastly superior to sequential algorithm. On the grid of size 128x128x128 computation times of parallel algorithm and sequential algorithm are equal to 0.449 seconds and 205.443 seconds respectively. Depending of the grid size it is possible to see that on tested hardware computational time of parallel algorithm 70-900 times less than sequential algorithm.

6 The Visualization Module for Large-scale Simulation

In recent years rise of modern computing technologies allowing to manipulate and simulate data of exaflops scale led to necessity to develop methods to display this data. This is the reason of implementation of a specialized visualization module appropriate for simulation of results of the SPF problem in complex domains (with faults and fractures). Initially the visualization module implemented as a desktop application, then reimplemented as Java applet for a web browser.

During implementation of the high performance desktop visualization module the Ray Casting algorithm - one of the variations of the Ray Tracing - used. The choice is justified by the fact that it gives the Asymptotic advantage [13]. The CUDA (Compute Unified Device Architecture) technology provided by NVIDIA as technological platform. For this reason it was decided to use NVIDIA Optix Ray Tracing Engine framework [14] which grants functionality for efficient implementation of different variations of Ray Tracing algorithm.

Visualization software with ray tracing algorithm often requires a certain indexing of rendering polygons. Therefore, various spatial indexing algorithms are implemented

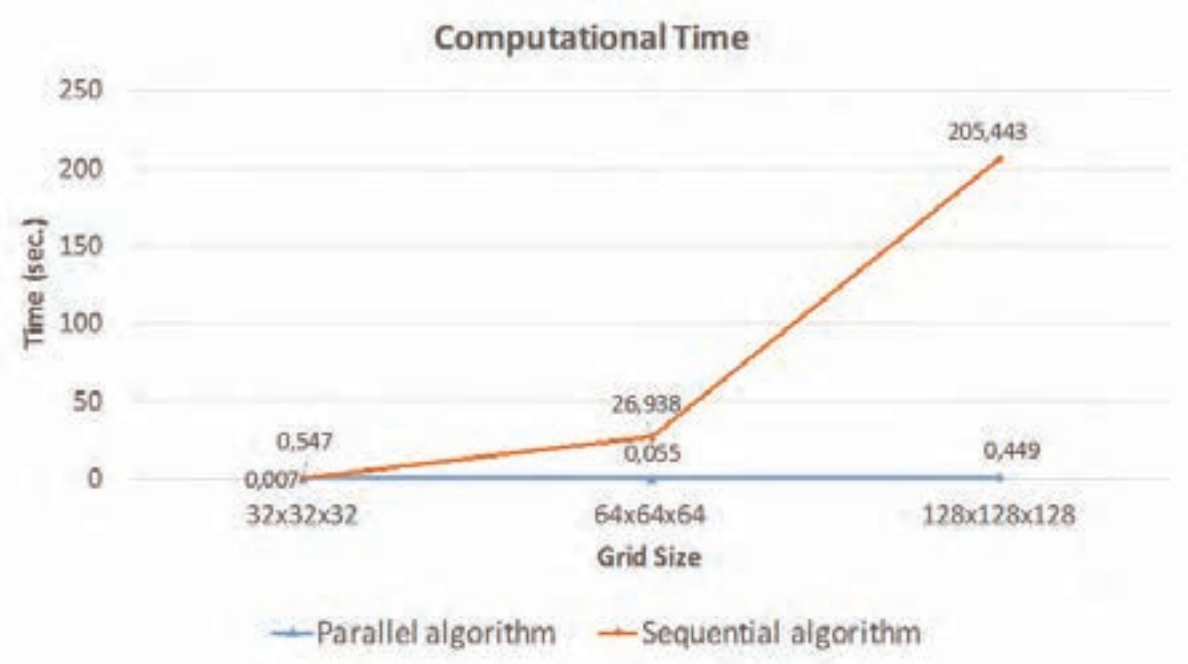
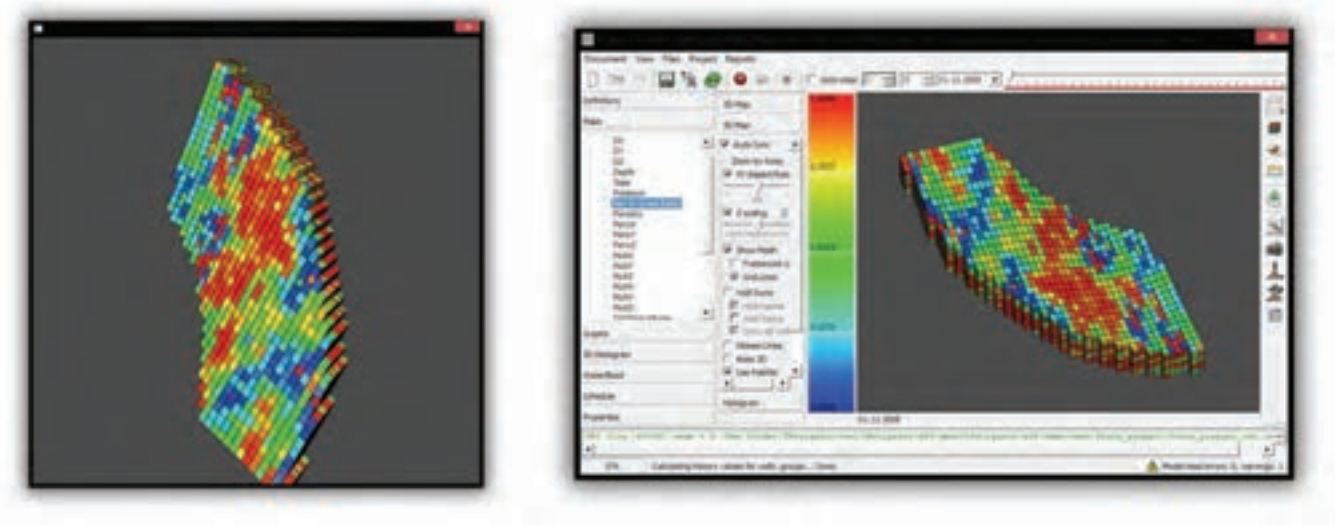


Figure 13: Comparison of parallel and sequential programs of SPF problem

along with the ray tracing algorithm. In the presented study, it was decided to use the BVH (Bounding Volume Hierarchy) - binary tree of bounding volumes built on certain sets of primitives [15] since it is most suitable for rendering static data.

Further the comparison of results of the presented visualization module and the work of similar visualizing programs Schlumberger Petrel [16] and Rock Flow Dynamics



(a) (b)

Figure 14: Comparison of visualization of the simple 33x33x11 field on presented visualization module (a) and Rock Flow Dynamics tNavigator (b)

tNavigator [17] (Figure 14) are shown:

1. Simple 33x33x11 model (Figure 14).

2. Model of East Moldabek site of the Kenbai oil field with full active cells visualization (Figures 15 (a - d)) and the cells only with proper values.

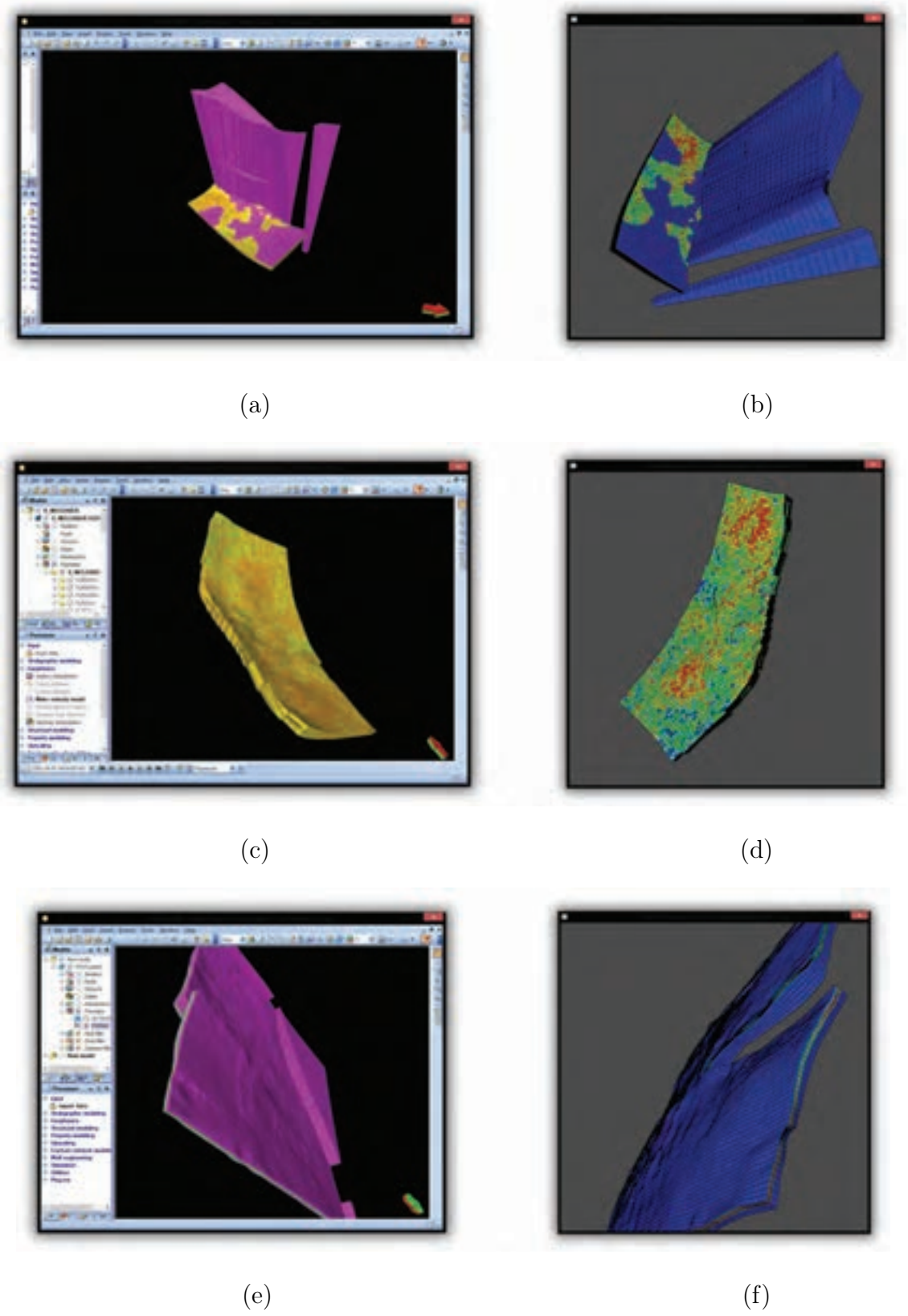
3. Public reservoir model of the project вЂњGeological Storage of CO2: Mathematical Modelling and Risk AnalysisвЂќ (MatMoRA) [18] from MATLAB Reservoir Simulation Toolbox [19] (100x100x21, Figures 15 (e,f)).

4. Public reservoir model of the project вЂњSensitivity Analysis of the Impact of Geological Uncertainties on ProductionвЂќ [20] (40x120x20,Figures 18 (a, b)).

5. The test models of the simulator ResInsight [21] (Figures 18 (c, d)).

After the implementation of the program on the NVIDIA OptiX series of tests was carried out for the large-scale generated data. Figure 17 demonstrates correct work of program on the test consisting more than 6.6 million polygons. Due to the fact that only the outer layer of polygons is shown that amount of polygons may correspond to several millions ($O(N^{3/2})$) of elementary volumes.

The next step is moving of the desktop application module to the web. For this purpose it is decided to use Java applet. Applet is a special class in Java that allows to launch the program from internet browser. To use Java applet at first it is required to move the application to Java since originally the algorithm implemented using CUDA - the modification of C programming language. For this purpose a special wrapper to main native functions was developed using the Java Native Interface [22] (Figure 18).

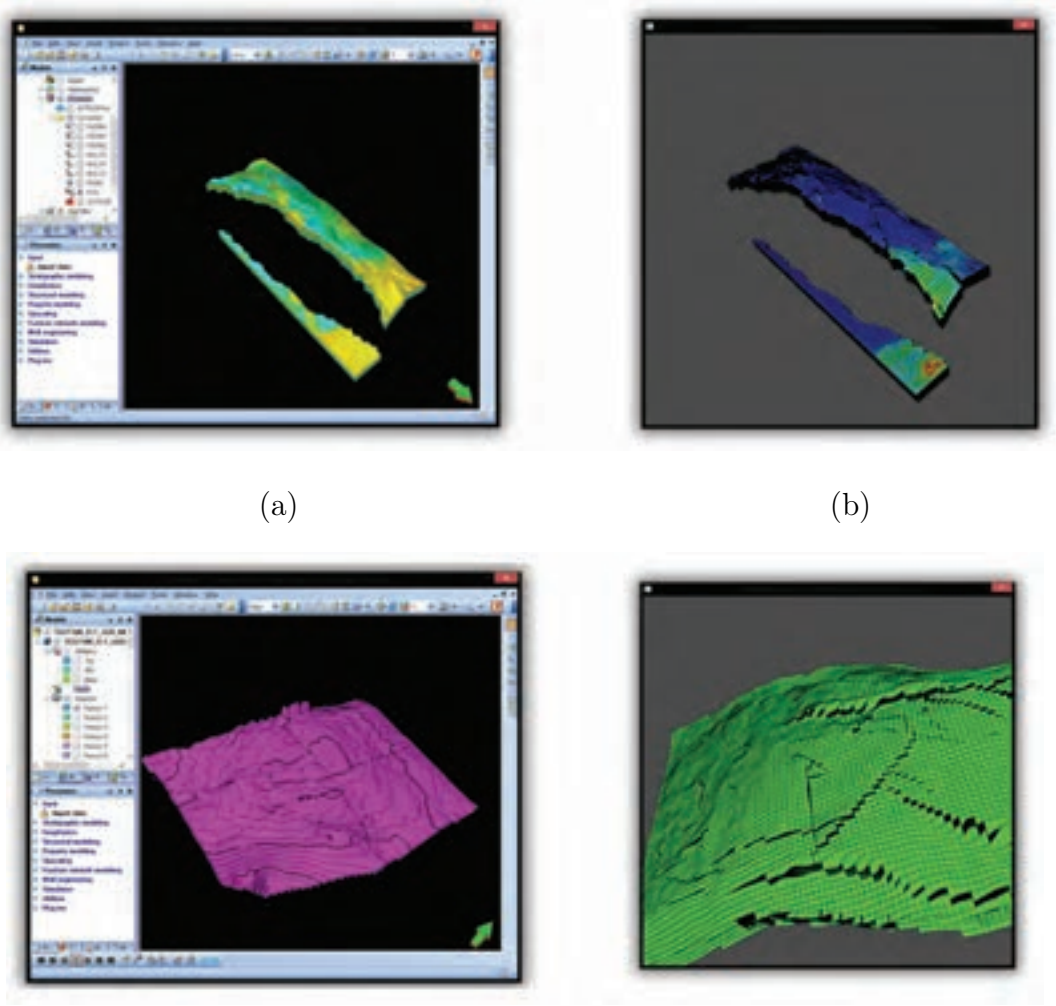


(a) (b)

(c) (d)

(e) (f)

Figure 15: Comparison of 3D models of some industrial fields visualized by Petrel program (a, c, e) and presented visualization module (b, d, f)



(a) (b)

(c) (d)

Figure 16: Comparison of 3D models of some industrial fields visualized by Petrel program (a, c) and presented visualization model (b, d)

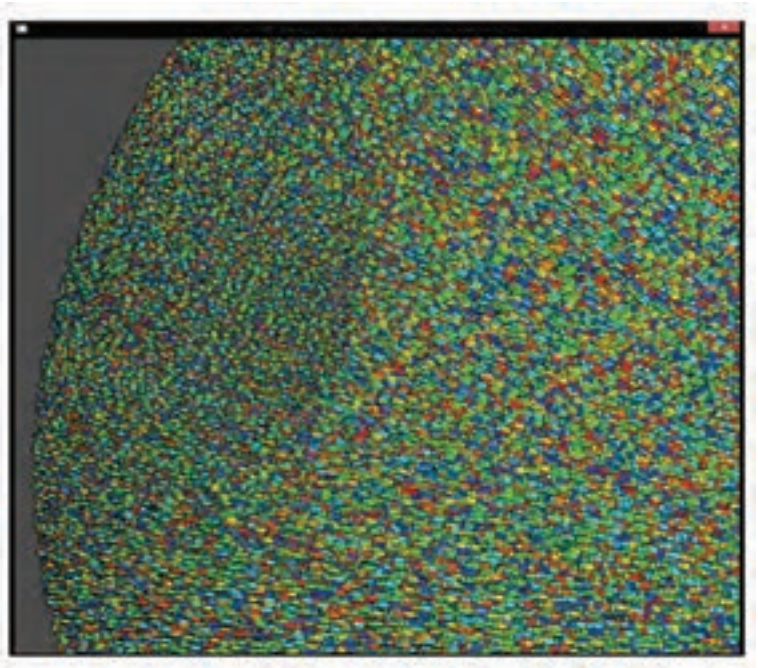


Figure 17: Test of large generated data. Values have been set randomly. Model has 6611916 triangle polygons

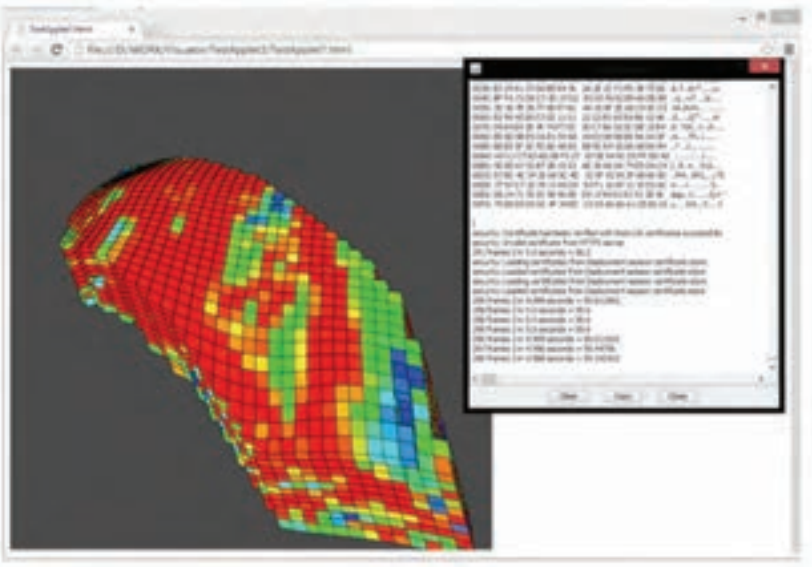


Figure 18: Web version (Java applet) of the visualization module with Java Console

7 Conclusion

Sections 2-5 of the paper considers the SPF problem, its algorithm as well as sequential and parallel implementations. A mathematical model of oil displacement process by polymer-surfactant injection is considered and solved, taking into account the dependence of viscosity of water phase on salt, surfactant and polymer concentrations and where viscosity of both phases depend on temperature.

System of equations solved using explicit method and following numerical results obtained: distribution of pressure, saturations of both phases, salt, surfactant and polymer concentrations and temperature distribution in oil reservoir. The polymer injection process into the oil reservoir for enhanced oil recovery can be modeled using proposed simulator.

Developed computational algorithms devoted to solve the 3D SPF problem and their implementation in the form of sequential programs with further development of corresponding parallel algorithms are provided. Parallel algorithms implemented using MPI and CUDA technologies. Parallel implementation of computational algorithm verified though comparison of execution times of sequential and parallel version calculating speedup and efficiency parameters.

Section 6 presents several implementations of visualization module for large-scale data rendering. Correctness of provided simulator verified with several data samples obtained from actual oil fields. Computational results of the SPF problem can be applied to the large-scale complex domains with faults and fractures, and furthermore visualized through the prototype of visualization simulator developed for this purpose.

References

[1] Lake L.W., *Enhanced oil recovery*, New Jersey: Prentice Hall Inc., 1989.

[2] Sorbie K.S., *Polymer improved oil recovery*, Boca Raton: CRC Press, 1991.

[3] Babalyan G.A., Levy B.I., Tumasyan A.B., Khalimov E.M., *Oilfield development using surfactants*, Moscow: Nedra, 1983.

[4] Benyamin Y. J., Riyaz K., Koorosh A., *The influence of salinity on the viscous instability in viscous-modifiedlow-interfacial tension flow during surfactant-polymer flooding in heavyoil reservoirs*, Fuel, 97 (2012), 174-185.

[5] *Eclipse Blackoil Simulator*, Technical Description. Schlumberger, 2013.

[6] Flory P.J., *Principles of polymer chemistry*, Cornell University Press, 1953.

[7] Wegner J., Ganzer L., *Numerical simulation of oil recovery by polymer injection using COMSOL*, Proceeding of the COMSOL conference, Milan, 2012.

[8] Samarskii A.A., Gulin A.V., *Numerical methods*, Moscow: Nauka, 1989.

[9] *T-cluster web site: http://cluster.kaznu.kz/ganglia/*

[10] J.J.Dongarra, J. Langou, *The Problem With the Linpack Benchmark 1.0 Matrix Generator*, International Journal of High Performance Computing Applications archive, 23. 1 (2009), 5-13.

[11] *http://www.nvidia.com/object/cuda_ home_ new.html*

[12] *http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-550ti*

[13] Roth, Scott D., *Ray Casting for Modeling Solids*, Computer Graphics and Image Processing, 18 (1982), 109-144.

[14] *NVIDIAB® OptiXѳ„ÿ Ray Tracing Engine Programming Guide, Version 3.0.*, NVIDIA Corporation, 2009-2012.

[15] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, *Efficient collision detection using bounding volume hierarchies of k-DOPs*, IEEE Trans. Visualization and Computer Graphics, 4. 1 (1998), 21-36.

[16] *Petrel E&P Software Platform - Schlumberger - www.slb.com/petrel*

[17] *Rock Flow Dynamics: tNavigator - www.rfdyn.com/technology*

[18] *Geological Storage of CO2: Mathematical Modelling and Risk Assessment - http:// www.sintef.no/MatMoRa*

[19] *MATLAB Reservoir Simulation Toolbox - http://www.sintef.no/Projectweb/MRST/Downloadable-Resources*

[20] *SAIGUP - http://www.nr.no/en/SAIGUP*

[21] *ResInsight - http://resinsight.org/*

[22] Liang S., *Java Native Interface*, Wesley, (1999), 318.

Darkhan Akhmed-Zaki,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: darhan_a@mail.ru,

Timur Imankulov,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: imankulov_ts@mail.ru,

Beimbet Daribayev,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: beimbet.daribaev@gmail.com,

Bazargul Matkerim,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: bazargulmm@gmail.com,

Kanat Aidarov,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: kanataydarov@yahoo.com,

Olzhas Turar,
al-Farabi Kazakh National University,
al-Farabi ave., 71,
Email: turar_olzhas@mail.ru,

Received 18.02.2016, Accepted 20.03.2016