Computing and Informatics, Vol. 36, 2017, 423–448, doi: 10.4149/cai_2017_2_423

PARALLEL FAST ISOGEOMETRIC SOLVERS FOR EXPLICIT DYNAMICS

Maciej Woźniak, Marcin Łoś, Maciej Paszyński

AGH University of Science and Technoloy, Krakow, Poland
e-mail: {macwozni, los, paszynsk}@agh.edu.pl

Lisandro Dalcin

King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
e-mail: dalcinl@gmail.com

Victor Manuel Calo

Applied Geology, Western Australian School of Mines
Faculty of Science and Engineering, Curtin University
Perth, WA, Australia 6845
&
Mineral Resources
Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Kensington, WA, Australia 6152
e-mail: vmcalo@gmail.com

Abstract. This paper presents a parallel implementation of the fast isogeometric solvers for explicit dynamics for solving non-stationary time-dependent problems. The algorithm is described in pseudo-code. We present theoretical estimates of the computational and communication complexities for a single time step of the parallel algorithm. The computational complexity is $\mathcal{O}\left(p^6 \frac{N}{c} t_{comp}\right)$ and communication complexity is $\mathcal{O}\left(\frac{N}{c^{2/3}} t_{comm}\right)$ where p denotes the polynomial order of B-spline basis with C^{p-1} global continuity, N denotes the number of elements and c is number of processors forming a cube, t_{comp} refers to the execution time of a single operation,

and t_{comm} refers to the time of sending a single datum. We compare theoretical estimates with numerical experiments performed on the LONESTAR Linux cluster from Texas Advanced Computing Center, using 1 000 processors. We apply the method to solve nonlinear flows in highly heterogeneous porous media.

Keywords: Isogeometric finite element method, alternating direction solver, fast parallel solver, non-stationary problems, nonlinear flows in highly-heterogeneous porous media

Mathematics Subject Classification 2010: 65F05, 68W10, 65M60

1 INTRODUCTION

We describe a parallel solution algorithm to solve the isogeometric finite elements L^2 projection problem. The algorithm parallelizes the sequential method originally proposed by [25]. This algorithm results in a direct solver for separable geometries while it is a fast iterative solver for other configuration.

This class of solvers has been used to derive preconditioners for Helmholtz equations [26]. The goal of the methodology is to develop fast solvers to address the extra cost incurred per degree of freedom for higher-continuity discretizations [13, 14, 15]. We estimate the computational and communication costs of this parallel implementation. We estimate the costs of all stages of the algorithm, including the gather and scatter of data into faces of a $3D$ cube of processors, the local solution of several $1D$ problems with multiple right-hand sides over a face of the cube of processors, as well as the reordering of data for processing in other directions. We compare the theoretical estimates of the computational and communication complexities with numerical experiments performed at the LONESTAR Linux cluster from the Texas Advanced Computing Center using 1 000 processors, for the problem size of 512^3 elements as well as for $1\,024^3$ elements. We obtained good agreement between the theoretical estimates and experimental results. We show that using the solver we can solve $512^3 = 134\,217\,728$ unknowns within 20 seconds and $1\,024^3 = 1\,073\,741\,824 = \mathcal{O}(10^9)$ unknowns within 3 minutes by using 1 000 processors from the LONESTAR Linux cluster. Next, we apply our alternating direction solver to solve a challenging non-stationary example problem of nonlinear flows in highly heterogeneous porous media [2]. The non-stationary problem is solved with a forward Euler scheme as a sequence of isogeometric L^2 projection problems. Thus, the alternating direction solver is applied at every time step of the time-dependent problem simulation. Finally, we verify the numerical results obtained for the non-stationary problem by analyzing the relative error between simulations performed with different time steps, as well as by monitoring the energy of the solution.

2 STATE OF THE ART

Classical higher order finite element methods (FEM) using hierarchical basis functions [19, 20] maintain only C^0-continuity between elements. In isogeometric analysis (IGA) B-splines are used as basis functions, and C^k global continuity can be built [16]. The higher continuity obtained at element interfaces allows IGA to attain optimal convergence rates for high polynomial orders of approximation, while using fewer degrees of freedom [3]. Higher-continuous IGA methods have allowed to obtain the solution of higher-order partial differential equations (PDE) with elegance. Example applications are shear deformable shell theory [7], phase field models for topology optimization [17, 18], phase separation simulations with possible application to cancer growth simulations, by using either Cahn-Hilliard [28] or Navier-Stokes-Korteweg [29], Swift-Hohenberg [42] and Navier-Stokes-Cahn-Hilliard [28] higher order models. The IGA methods have also many applications in simulations of non-linear problems of engineering interest, such as wind turbine aerodynamics [31], incompressible hyper-elasticity [22], turbulent flow simulations [11], transport of drugs in cardiovascular applications [30] or to blood flow simulations itself [5, 9].

Nevertheless, the price to pay for the usage of higher order IGA methods is the higher computational cost of IGA solvers, since solution time per degree of freedom augments as the continuity is increased [10, 15]. This is true for all implementation of multi-frontal solver algorithms [24, 23], including MUMPS solver [35], modern implementations for adaptive and higher order methods [27] or graph-grammar based approach [39, 40, 41, 38].

The computational cost of the sequential IGA direct solver with C^{p-1} global continuity of the solution is $\mathcal{O}(N^{1.5}p^3)$ for the $2D$ case, and $\mathcal{O}(N^2p^3)$ for the $3D$ case [10, 15]. Here N refers to global number of degrees of freedom and p refers to the order of B-spline C^{p-1} global continuity basis functions, t_{comp} refers to the execution time of a single operation, and t_{comm} refers to the time of sending a single data. In case of distributed memory Linux cluster parallel machines, this computational cost can be reduced down to $\mathcal{O}(Np^2t_{comp})$ for the $2D$ case, and $\mathcal{O}(N^{4/3}p^2t_{comm})$ for the $3D$ case [47]. Similarly, in the case of shared memory GPU machines this computational cost can be reduced down to $\mathcal{O}(Np^2)$ for the $2D$ case, and for the $3D$ case GPU machines often run out of memory [46]. In this paper we focus on $3D$ isogeometric finite element L^2 projection problems solved exactly by means of direct solver algorithms. Thus, the computational cost of the state-of-the-art direct solver applied to $3D$ IGA L^2 projection problem when using Linux cluster parallel machine is $\mathcal{O}(N^{4/3}p^2t_{comp})$ and the communication cost is of the same order $\mathcal{O}\left(N^{4/3}p^2t_{comm}\right)$. Moreover, in those estimates as presented in [46, 47] it is assumed that we have infinite number of available processors. In this paper we propose the new parallel alternating direction solver algorithm for the isogeometric finite element L^2 projection problem designed for Linux cluster parallel machine, delivering $\mathcal{O}\left(p^6\frac{N}{c}t_{comp}\right)$ computational complexity and $\mathcal{O}\left(\frac{N}{c^{2/3}}t_{comm}\right)$ communication complexity, assuming we have c available processors, forming a hypercube. In other words our solver al-

lows for significant reduction of the computational cost of the direct solver solution of isogeometric L^2 projection problem.

The L^2 projection problem can be utilized for solution of the non-stationary problems in the following way. For a general time parabolic problem, where we seek a solution u of the time-dependent $\frac{\partial u}{\partial t}$ second-order partial differential equation with differential operator A, under the forcing f over the domain Ω within time interval $[0,T]$:

$$\begin{aligned} \frac{\partial u}{\partial t} - Au &= f \text{ over } \Omega \times [0,T], \\ u(x,0) &= u_0(x) \text{ over } \Omega \end{aligned} \tag{1}$$

we apply the forward Euler scheme $\frac{\partial u}{\partial t} \approx \frac{u_{t+1}-u_t}{\Delta t}$ in time and a variational formulation in space to obtain:

$$(u_{t+1}, v)_{L^2} = (u_t + \Delta t (Au_t + f), v)_{L^2} \quad \forall v \in V \tag{2}$$

where V is a suitable Sobolev space. The solution for each new time step is obtained by isogeometric L^2 projection, using the discretization with C^{p-1} global continuity B-splines basis functions, delivered by the isogeometric solver. Thus, the alternating direction solver is used as an explicit solver for the solution of non-stationary problems. In particular we focus on the numerical solution of nonlinear flows in highly-heterogeneous porous media [2].

The structure of the paper is the following. We first formulate the parallel alternating direction algorithm on the $3D$ cube of processors in Section 2. Next, in Section 3 we present the detailed analysis of the computational and communication complexity of the parallel algorithm. Section 4 is devoted to the experimental verification of the derived computational and communication complexities. Section 5 discusses application of the alternating directions solver for the solution of non-stationary problem of nonlinear flows in highly-heterogeneous porous media [2]. Section 6 presents the analysis on the correctness of the numerical results, and finally Section 7 presents conclusions and future work.

3 PARALLEL ISOGEOMETRIC L² PROJECTION ALGORITHM

The sequential version of the alternating direction solver is described in [25]. In this paper we propose a parallel version of the algorithm, targeting distributed memory Linux cluster parallel machines.

The parallel version of the isogeometric L^2 projection algorithm generates data distributed in an uniform way over a $3D$ cube of processors, as depicted in Figure 1. There are three steps of the algorithm where the data are gathered into OYZ, OXZ and OXY faces, respectively. The local $1D$ bended problems are solved there, using the LAPACK library. The data are scattered into a cube of processors, to proceed with the next step. The algorithm can be summarized as shown in Figure 1.

0. Integration
1a. Gather within every row of processors into OYZ face
1b. Solve N_yN_z $1D$ problems with multiple right hand sides
1c. Scatter results onto cube of processors
1d. Reorder right hand sides
2a. Gather within every row of processors into OXZ face
2b. Solve N_xN_z $1D$ problem with multiple right hand sides
2c. Scatter results onto cube of processors
2d. Reorder right hand sides
3a. Gather in every row of processors into OXY face
3b. Solve N_xN_y $1D$ problem with multiple right hand sides
3c. Scatter results onto cube of processors
3d. Reorder right hand sides

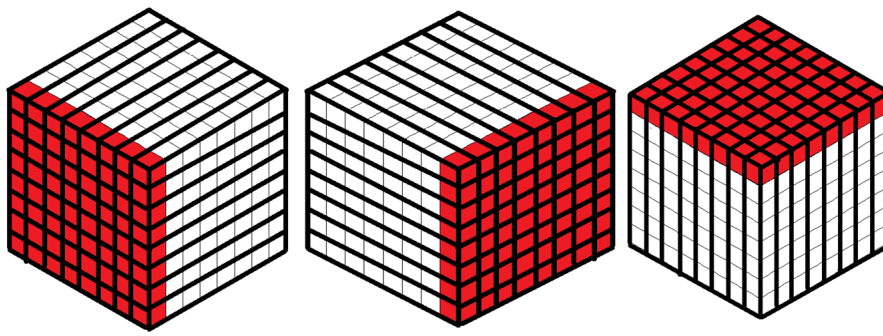

Figure 1. Gathering and scattering data into three faces of the $3D$ cube of processors

4 COMPLEXITY ANALYSIS

4.1 Integration over One Element

Every element is approximated by a set of polynomials in each direction where p is the order and there are $p+1$ B-splines over the element. We denote p_x as the degree in x direction and p_y and p_z as degrees in other directions.

The integration of the right hand side requires using Gauss quadrature with $(p_x+1)(p_y+1)(p_z+1)$ points. The integral over each element is:

$$\sum_{m=1}^{(p_x+1)(p_y+1)(p_z+1)} w_m B_x^i(x_m) B_y^j(y_m) B_z^k(z_m) f(x_m, y_m, z_m)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{3}$$

where w_m denotes the Gaussia quadrature weights, B_x^i, B_y^j, B_z^k denote the B-spline basis functions in x, y, and z directions, respectively, computed at x_m, y_m, z_m Gaus-

sian quadrature points, and we have $i = 1, \ldots, p_x + 1$, $j = 1, \ldots, p_y + 1$ and $k = 1, \ldots, p_z + 1$ entries to compute. Assume that for $d = 1, \ldots, (p_x+1)(p_y+1)(p_z+1)$ counting value at given point for given element and function f costs $\Phi^f((p_x+1)^2(p_y+1)^2(p_z+1)^2)$ arithmetic operations where Φ^f is the function depending on f.

The formula for Φ^f depends on the form of f. If f is given by a prescribed formula, then cost of computing a value of f is constant and Φ^f is constant. Otherwise when f is given by a combination of B-splines

$$f = \sum_{o=1}^{p_x+1} \sum_{q=1}^{p_y+1} \sum_{r=1}^{p_z+1} B_x^o B_y^q B_z^r f_{oqr} \tag{4}$$

then

$$\Phi^f(x_m) = (p_x + 1)(p_y + 1)(p_z + 1) \tag{5}$$

and total cost will be

$$(p_x + 1)^3(p_y + 1)^3(p_z + 1)^3. \tag{6}$$

In the following part of the paper we assume that f is prescribed by a given formula, and so the cost of computation a value of f at a given point is constant.

4.2 Integration over All Elements

We have a mesh of $N_x \times N_y \times N_z$ elements (where N_x, N_y, N_z denotes the number of elements in the x, y and z direction, respectively). The integration algorithm generates a local matrix over all $N_x N_y N_z$ element. It involves a loop with respect to local B-spline basis functios, and there are $(p_x + 1)(p_y + 1)(p_z + 1)$ local basis functions. It also involves a loop with respect to Gaussian integration points, and there are again $(p_x + 1)(p_y + 1)(p_z + 1)$ points.

The total cost of integration will be

$$(p_x + 1)^2(p_y + 1)^2(p_z + 1)^2 N_x N_y N_z \Phi^f. \tag{7}$$

We can do every integration with zero communication cost. When we have cuboid of $c_x c_y c_z$ cores it can be done in:

$$\frac{(p_x + 1)^2(p_y + 1)^2(p_z + 1)^2 N_x N_y N_z \Phi^f}{c_x c_y c_z} \tag{8}$$

with computational complexity of

$$O\left(\frac{p_x^2 p_y^2 p_z^2 N_x N_y N_z}{c_x c_y c_z}\right). \tag{9}$$

There are some sum factorization techniques for speeding up the integrations, like the one proposed in [8] for hierarchical basis functions. Another method applies again for 3D hierarchical basis functions and they may reduce the computational cost

from $\mathcal{O}(p^9)$ down to $\mathcal{O}(p^5)$ [20]. However, we are not aware of the sum-factorization technique for B-splines.

4.3 Solve

In each step of the algorithm we LU factorize a banded matrix resulting from one dimensional B-spline basis function of order p. This is done at a face of the $3D$ cuboid of processors. Let N be the number of elements in one direction. Then, the banded matrix M^N of size N with $2p+1$ diagonal blocks can be LU factorized in $O(p^2 N)$ steps.

When solving problem in the x direction we have to LU factorize matrix M^{N_x} of size N_x with $2p_x+1$ diagonal blocks and we have $\frac{N_y}{c_y} \times \frac{N_z}{c_z}$ right hand sides, each one of size N_x. The communication cost is zero, since we have $c_y \times c_z$ CPUs solving sequentially at the same time. Solving in x direction over each of these processors results in a computational complexity

$$O\left(N_x p_x^2 \frac{N_y}{c_y}\frac{N_z}{c_z}\right). \tag{10}$$

The solution complexity over y and z directions can be estimated in analogous way as

$$O\left(N_y p_y^2 \frac{N_x}{c_x}\frac{N_z}{c_z}\right) \tag{11}$$

and

$$O\left(N_z p_z^2 \frac{N_x}{c_x}\frac{N_y}{c_y}\right), \tag{12}$$

this results in computational complexity

$$O\left(\frac{\left(p_x^2 c_x + p_y^2 c_y + p_z^2 c_z\right)\left(N_x N_y N_z\right)}{c_x c_y c_z}\right). \tag{13}$$

4.4 Gathering Data

While collecting data in the x direction we need to collect information from $c_x c_y c_z - c_y c_z$ CPUs into $c_y c_z$ CPUs. Each processor has $\frac{N_x}{c_x}\frac{N_y}{c_y}\frac{N_z}{c_z}$ data. We apply a torus communication structure available on a linux cluster. This implies linear communication complexity in each row of processors in each direction and gives us communication complexity of:

$$O\left(N_x \frac{N_y}{c_y}\frac{N_z}{c_z}\right). \tag{14}$$

Additionally, the gathering data along y and z directions results in the communication complexity of:

$$O\left(N_y \frac{N_x}{c_x}\frac{N_z}{c_z}\right) \tag{15}$$

and

$$O\left(N_z \frac{N_x}{c_x}\frac{N_y}{c_y}\right). \tag{16}$$

Summing, the total communication complexity of gathering data is equal to

$$O\left(\frac{(c_x+c_y+c_z)N_xN_yN_z}{c_xc_yc_z}\right). \tag{17}$$

4.5 Reorder Data

After processing data in the x-direction we need to perform the reorder of data over each CPU before processing data along y direction. Similar reordering applies after processing data in the y-direction and before processing data in the z-direction. The computational complexity of each of the two reorders, executed over each processor is equal to

$$\mathcal{O}\left(\frac{N_xN_yN_z}{c_xc_yc_z}\right). \tag{18}$$

4.6 Scattering Data

Scatter is just an inverse of the gather, and its communication complexity is the same as the cost of the gather operation

$$\mathcal{O}\left(\frac{(c_x+c_y+c_z)(N_xN_yN_z)}{c_xc_yc_z}\right). \tag{19}$$

4.7 Total Complexity

From the discussion above, we conclude that we can construct isogeometric projection solver with the total cost

$$\begin{aligned}\left(\frac{p_x^2p_y^2p_z^2N_xN_yN_z}{c_xc_yc_z}\right)t_{comp} + \left(\frac{\left(p_x^2c_x+p_y^2c_y+p_z^2c_z\right)\left(N_xN_yN_z\right)}{c_xc_yc_z}\right)t_{comp} \\ + \left(\frac{N_xN_yN_zp_xp_yp_z}{c_xc_yc_z}\right)t_{comp} + \left(\frac{(c_x+c_y+c_z)N_xN_yN_z}{c_xc_yc_z}\right)t_{comm}\end{aligned} \tag{20}$$

for arbitrary polynomial orders p_x, p_y, p_z, dimension sizes N_x, N_y, N_z and processors numbers c_x, c_y, c_z, where t_{comp} is the cost of processing a single FLOAT, and t_{comm} is the cost of communicating a single byte.

Assuming

$$N_x = N_y = N_z = N^{1/3}, \quad p_x = p_y = p_z = p, \quad c_x = c_y = c_z = c^{1/3} \tag{21}$$

we have the following cost

$$\left(\frac{p^6 N}{c} + \frac{p^2 N}{c^{2/3}} + \frac{p^3 N}{c}\right) t_{comp} + \left(\frac{N}{c^{2/3}}\right) t_{comm} \tag{22}$$

which implies the computational complexity

$$O\left(p^6 \frac{N}{c}\right) \tag{23}$$

and communication complexity

$$O\left(\frac{N}{c^{2/3}}\right). \tag{24}$$

5 EXPERIMENTAL VERIFICATION

The model has been verified by comparing with numerical experiments performed on the LONESTAR Linux cluster, with $N = 512$ or $N = 1\,024$ degrees of freedom in each direction, with $p = 3$. The comparison of the total execution time is given in Figures 2 and 3. We can draw the following conclusions from these figures. There is good agreement between the theoretical estimates and numerical experiments for both 512^3 and $1\,024^3$ cases with cubic polynomials. Good scalability of the solver is maintained up to 1 000 of processors. We can solve 134 217 728 unknowns resulting from $3D$ cube of 512^3 elements with cubic B-splines within 20 seconds using 1 000 processors. We can also solve 1 073 741 824 unknowns resulting from $3D$ cube of $1\,024^3$ elements with cubic B-splines within 3 minutes by using 1 000 processors.

Figures 4 and 5 present the comparison of the experimental and theoretical integration times. We can draw the following conclusions from these figures. Again, there is a good match between the theoretical estimates and experimental results for the integration execution time. The integration time dominates the solution time significantly. In other words, the generation of the projection data takes much more time than actually the isogeometric L^2 projections using alternating direction solver itself, and it means that our solver algorithm performs very well (usually the solution takes much more time than integration). In order to speedup the solver, we may need to look for some new fast integration algorithms designed for B-spline basis functions.

Figures 6 and 7 present the comparison of the experimental and theoretical solution times. We measure three solution phases, corresponding to steps 1b, 2b and 3b of the general algorithm. We can draw the following conclusions from these figures. Again, there is a good match between the theoretical estimates and experimental results for the solution times. The solution time takes around 1 percent of the total solver time. In our solver we utilize multiple $1D$ sequential block diagonal multi-frontal solvers with many right hand sides working on the faces of the cube of

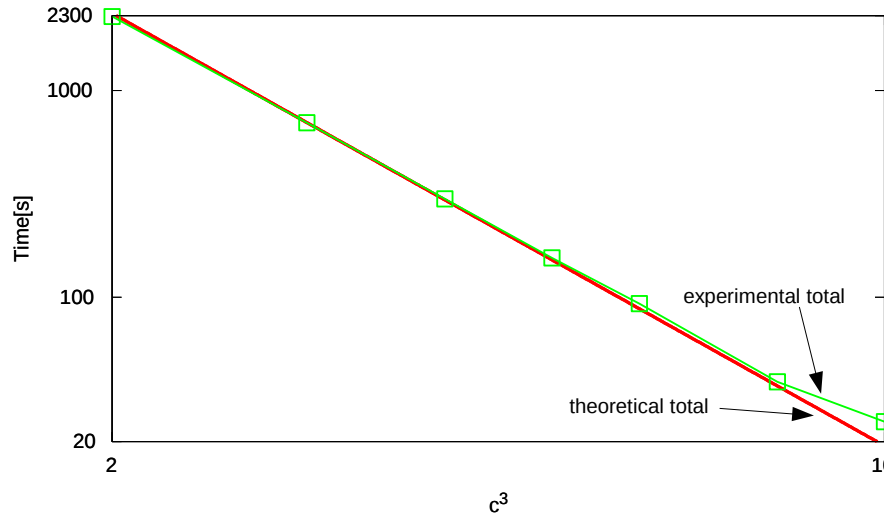

Figure 2. Comparison of total experimental and theoretical execution time for $N = 512$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

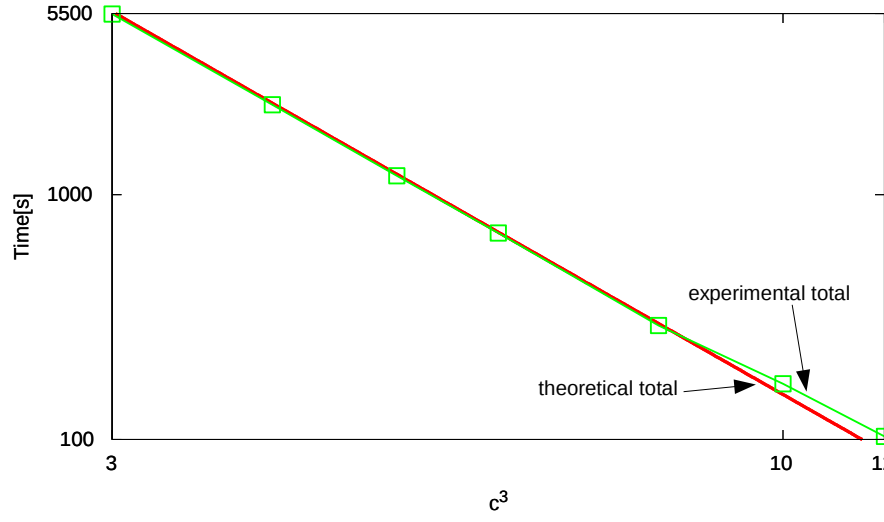

Figure 3. Comparison of total experimental and theoretical execution time for $N = 1\,024$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

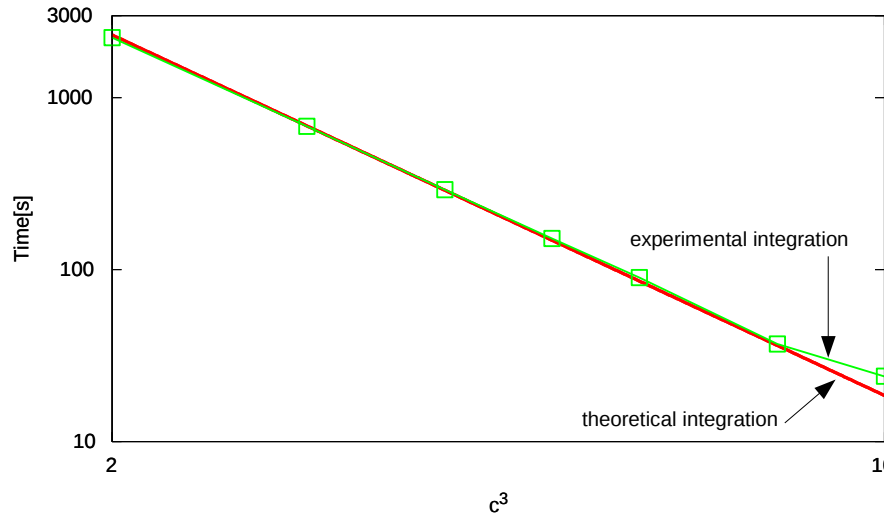

Figure 4. Comparison of experimental and theoretical integration time for $N = 512$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

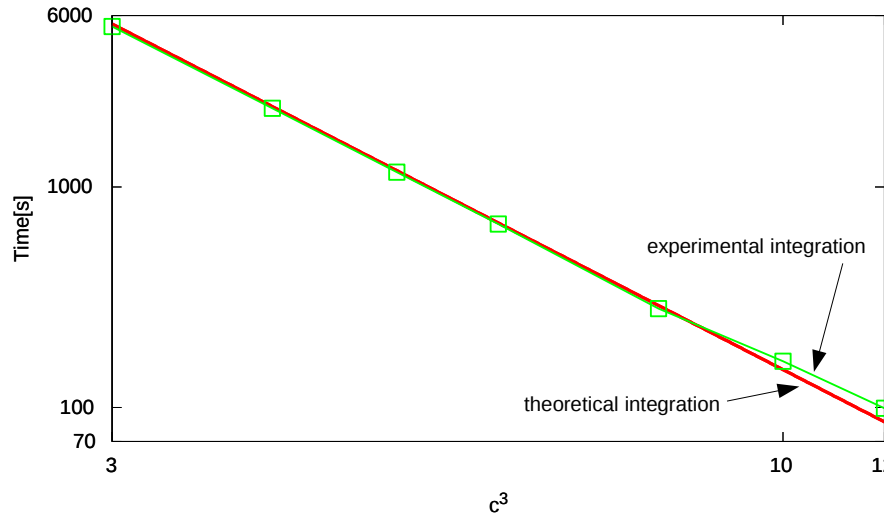

Figure 5. Comparison of total experimental and estimated integration time for $N = 1\,024$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

$3D$ processors. Possible improvement of the algorithm would be to utilize parallel block-diagonal solvers working within rows of processors (10 processors per solver in 1 000 processors case).

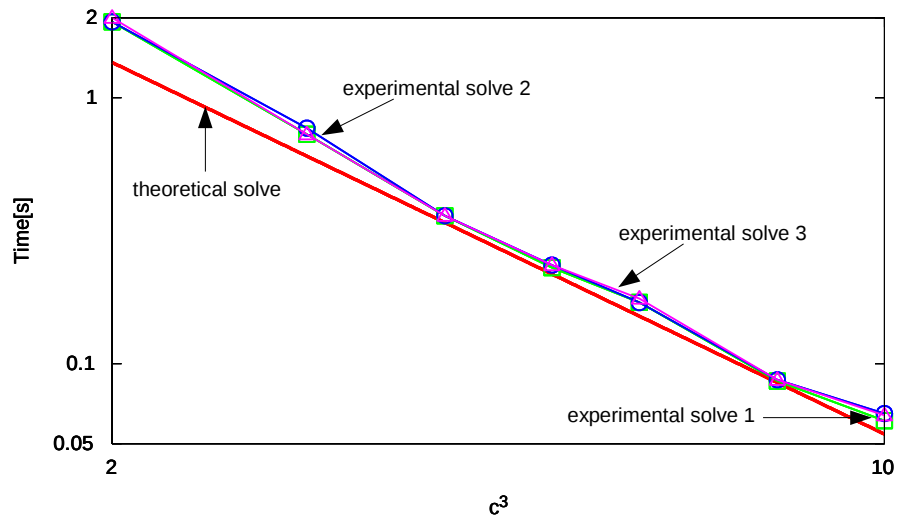

Figure 6. Comparison of experimental and theoretical solution times for $N = 512$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

Figures 8 and 9 present the comparison of the experimental and theoretical gather times. We measure three gathering phases, corresponding to steps 1a, 2a and 3a of the general algorithm. We can draw the following conclusions from these figures. There is a good match between the theoretical and experimental gather times for second and third gather. However, the first gather takes actually less time then predicted by the model. Our second and third gather times include the data reorder phase, while the first gather is just the communication itself.

Figures 10 and 11 present the comparison of the experimental and theoretical scatter times. We measure three scattering phases, corresponding to steps 1c, 2c and 3c of the general algorithm. We can draw the following conclusions from these figures. There is a good match between the theoretical and experimental scatter times for all the phases. The experimental scatters are becoming little slower when we increase the number of processors, however the difference is very small, less than 0.1 second.

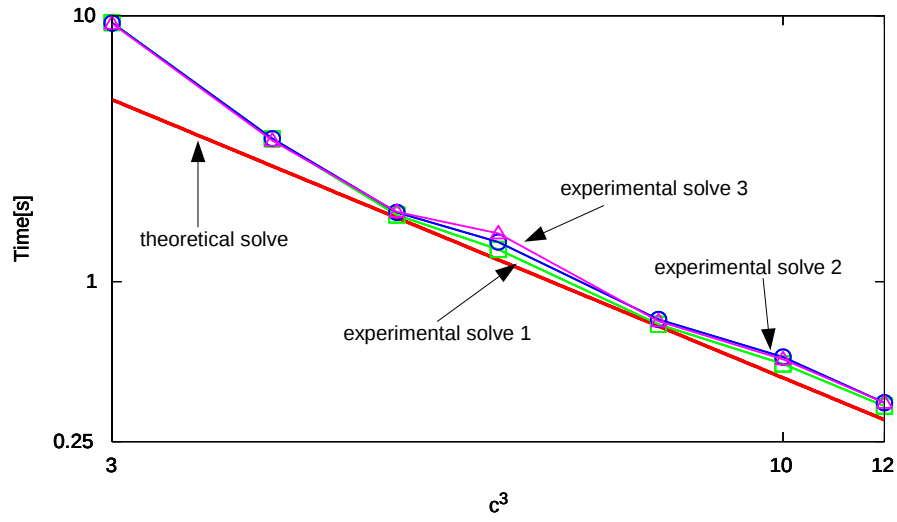

Figure 7. Comparison of total experimental and estimated solution times for $N = 1\,024$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

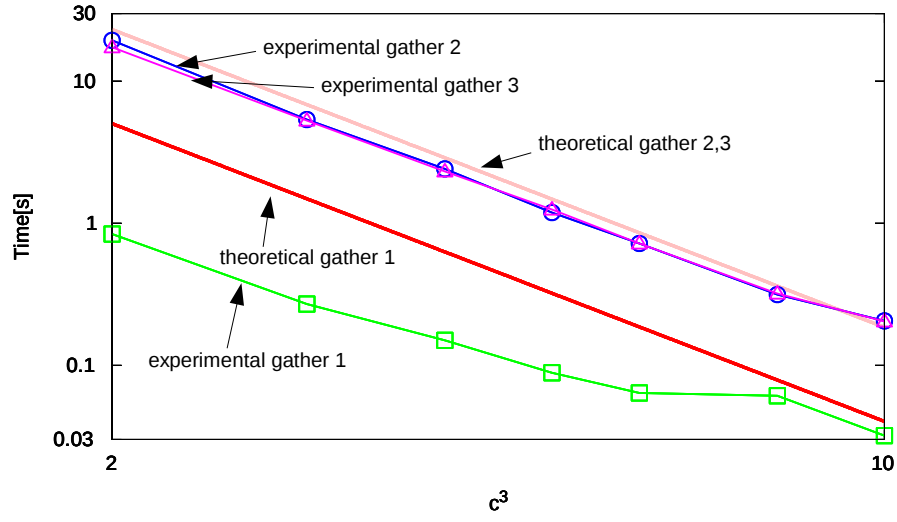

Figure 8. Comparison of experimental and theoretical gather times for $N = 512$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

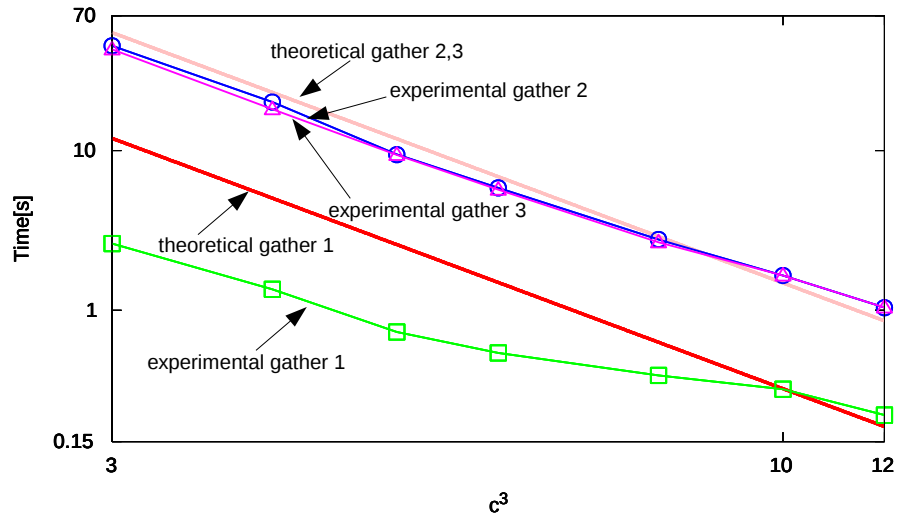

Figure 9. Comparison of total experimental and estimated gather times for $N = 1\,024$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

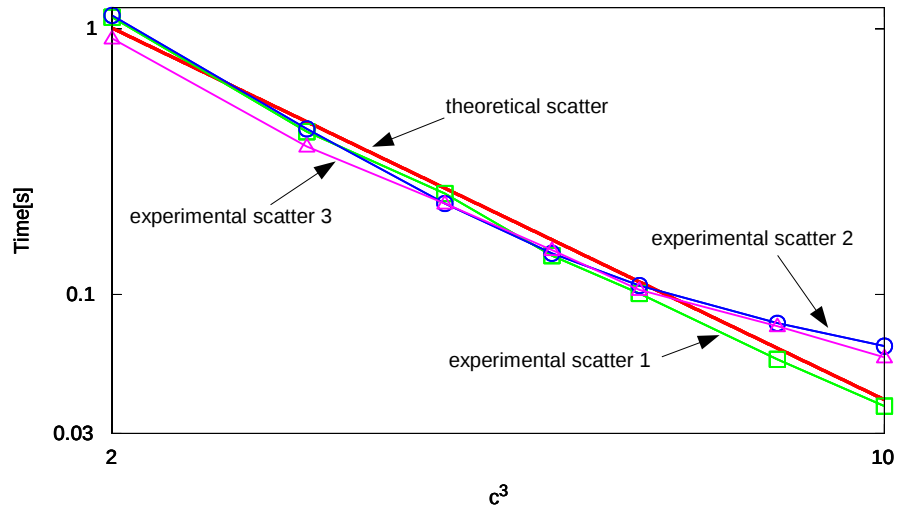

Figure 10. Comparison of experimental and theoretical scatter times for $N = 512$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

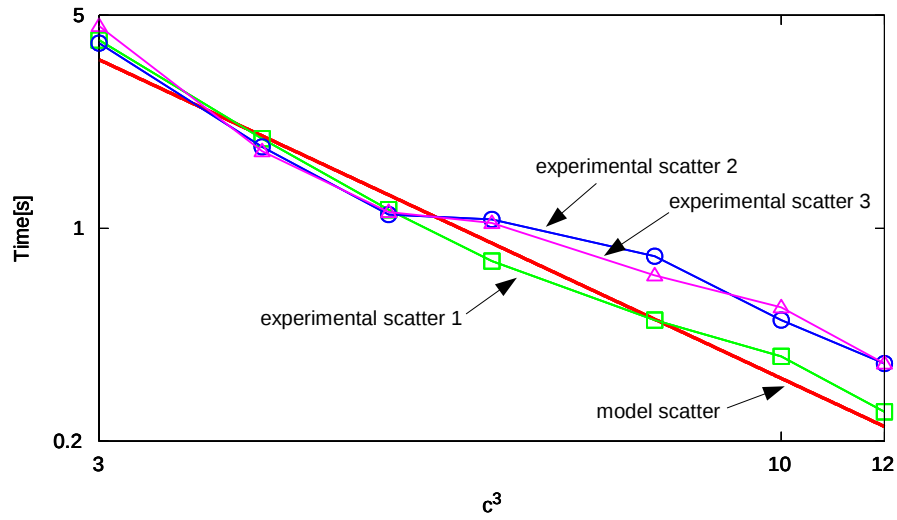

Figure 11. Comparison of total experimental and estimated scatter times for $N = 1\,024$ for $p = 3$ for different number of processors $c^3 = 2^3, \ldots, 10^3 = 8, \ldots, 1\,000$

6 NUMERICAL RESULTS

6.1 Problem Formulation

In this section we present the application of the parallel isogeometric L^2 projection solver for the simulation of the problem of nonlinear flows in highly-heterogeneous porous media [2]. The problem is formulated in dimensionless units. The governing equation in the strong form is given by

$$\frac{\partial u}{\partial t} - \nabla \cdot \left(K_q\left(x\right) e^{10u(x)} \nabla u \right) = h\left(x\right) \tag{25}$$

where $K_q\left(x\right)$ is a material data function, as depicted in Figure 12, and $h\left(x\right) = 1 + \sin\left(2\Pi x_1\right) \sin\left(2\Pi x_2\right) \sin\left(2\Pi x_3\right)$.

The strong form is transformed into a weak one by taking the L^2 scalar product with test functions $v \in H^1\left(\Omega\right)$, and the Euler integration scheme is utilized with respect to time

$$\left(u_{t+1}, v\right)_{L^2} = \left(u_t + Dt \nabla \cdot \left(K_q e^{10u_t} \nabla u_t\right) + h, v\right)_{L^2}. \tag{26}$$

We solve the problem over the cube $\Omega = [0,1]^3$ domain. We utilize the isogeometric L^2 projection solver to execute the Euler scheme for the above problem. The time step size has been selected as 10^{-5}. The initial value is a ball with radius 0.05 and

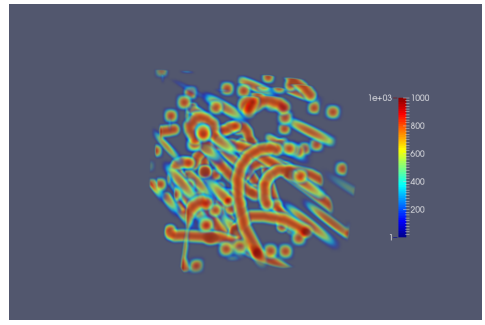

Figure 12. Initial distribution of the material data function $K_q(x)$

maximum value 0.02 is presented in Figure 13. The snapshots from the numerical simulation from time steps 20, 100, 200, 300, 500 and 1 000 are presented in Figures 14–18, respectively.

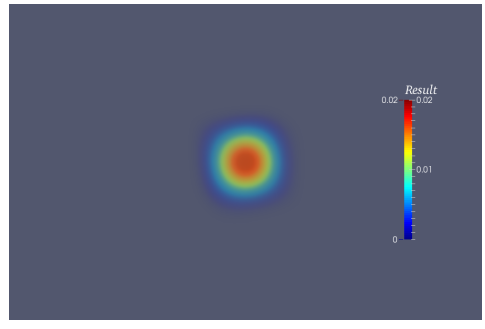

Figure 13. Initial state

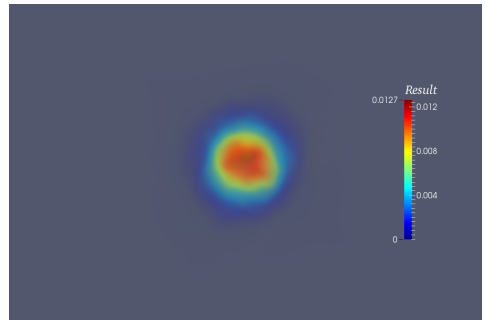

Figure 14. Solution at time step 20

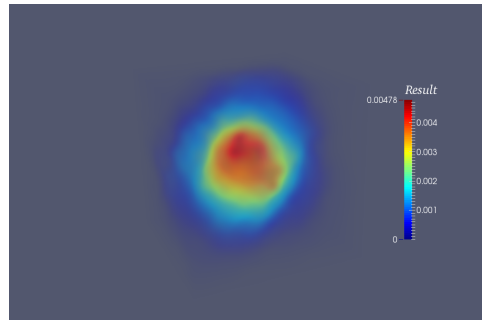

Figure 15. Solution at time step 100

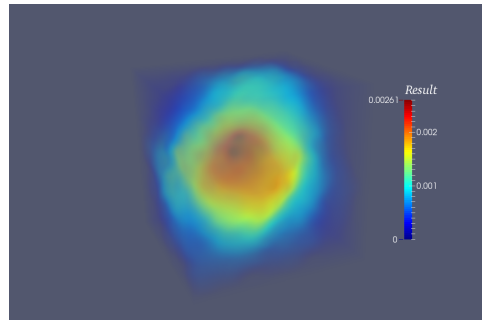

Figure 16. Solution at time step 200

6.2 Verification

In order to verify the correctness of the numerical results we have performed two tests. First, while looking for the correct time step size, we have checked the relative error for time step $Dt = 10^{-4}$ which happened to be too large, and then for time step

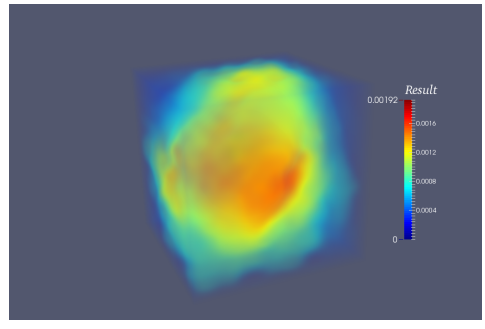

Figure 17. Solution at time step 300

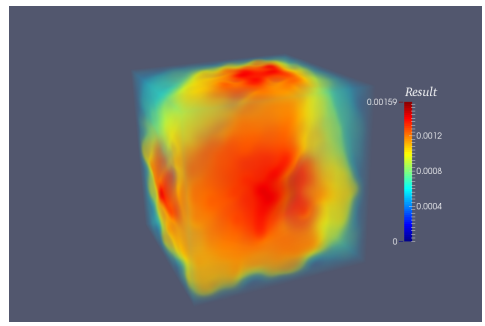

Figure 18. Solution at time step 500

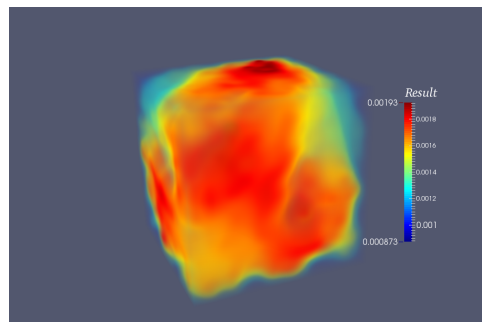

Figure 19. Solution at time step 1 000

$Dt = 10^{-5}$ which seems to be a correct one. The experiments for time step $Dt = 10^{-4}$ are presented in Figure 20. We have executed a sequence of experiments for time step $Dt_1 = 10^{-4}$, as well as with smaller time steps, $Dt_{1/2} = 10^{-4}/2$, $Dt_{1/3} = 10^{-4}/3$, $Dt_{1/4} = 10^{-4}/4$, $Dt_{1/5} = 10^{-4}/5$, $Dt_{1/4} = 10^{-6}/6$ and $Dt_{1/7} = 10^{-4}/7$. We have used the smallest time step $Dt_{1/7}$ as the reference time step. For each time step of Dt_1 we have performed k time steps of $Dt_{1/k}$. We have computed the relative error between the particular solutions from corresponding time steps. The relative errors are presented in Figure 20, where $e_{1/k} = ||u_t^{D_{1/k}} - u_t^{D_{1/7}}||_{L^2}$. We can see that the relative error is growing for all the cases, except when we measure the relative error of the reference solution itself, since it is zero by definition. We conclude that the time step $Dt = 10^{-4}$ is too large.

Thus, we repeat the experiment for the smaller time step $Dt = 10^{-5}$. The new tests have been performed in the analogously and their results are shown in Figure 21. We can see now that all the relative errors are small of the order of 10^{-5} or less and not growing.

Second, we measured the energy (defined as $\int_\Omega |\nabla u|^2$) and the L^2 norm (defined as $\left(\int_\Omega |u|^2\right)^{1/2}$ of the solution u in particular time steps. The L^2 norm and energy

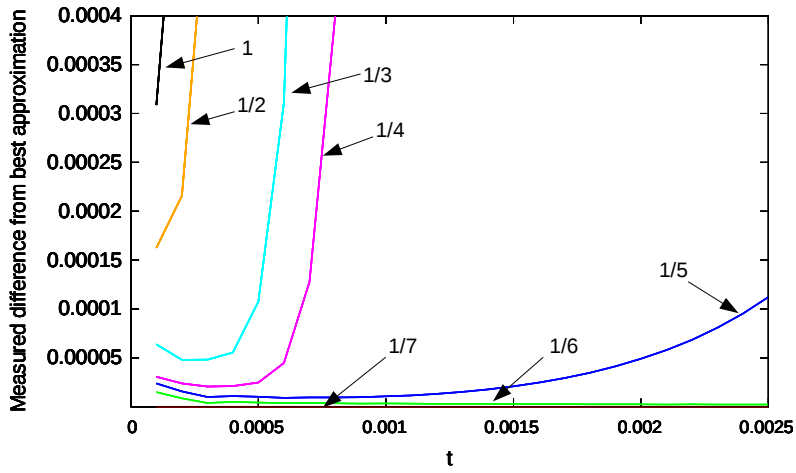

Figure 20. Relative errors for the time step $Dt = 10^{-4}$

are presented in Figure 22. The reason they are linearly and continuously growing is the fact that right hand side of our problem has form 1+term$\in [-1, 1]$, and we are constantly adding the energy to the system.

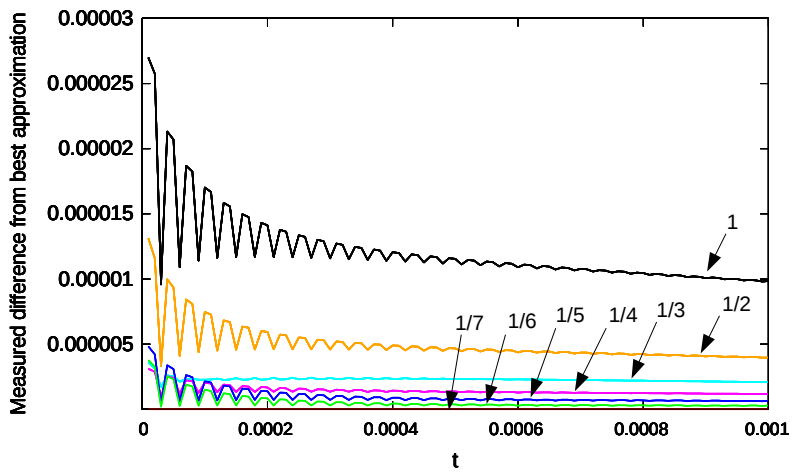

Figure 21. Relative errors for the time step $Dt = 10^{-5}$

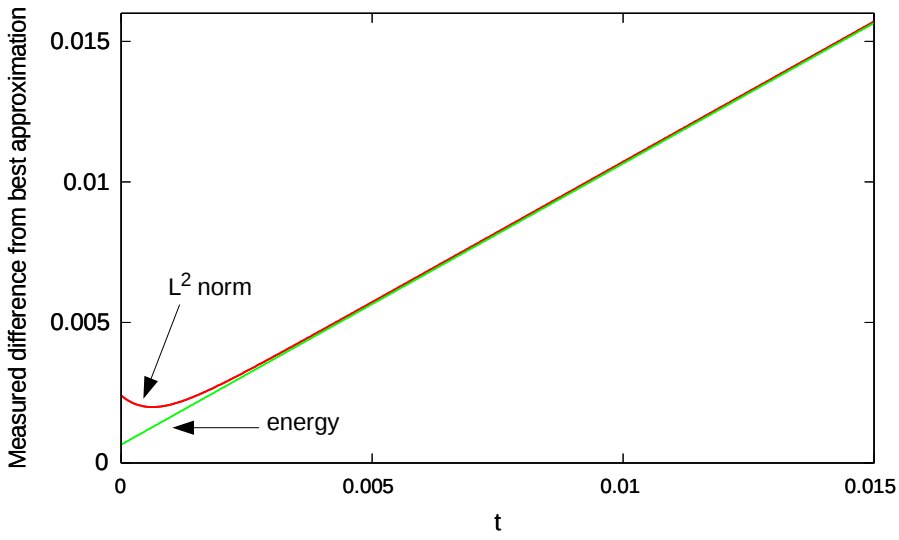

Figure 22. L^2 norm and energy of the solution during the simulation

7 CONCLUSIONS AND FUTURE WORK

In this paper, we present a fast parallel the alternating directions isogeometric L^2 projections solver. The parallel alternating direction solver is available through PetIGA library [12] and PetIGA-MF [42] libraries, and as a standing alone package [34] for shared-memory environment. The computational complexity of the parallel algorithm is of order $O\left(\frac{p^6 N}{c}\right)$ and the communication complexity is of order $O\left(\frac{N}{c^{2/3}}\right)$, where p denotes the order of the B-spline basis with C^{p-1} global continuity, N denotes the number of elements and c the number of processors over the $3D$ hypercube. We verify the theoretical estimates with numerical experiments performed at the LONESTAR linux cluster from the Texas Advanced Computing Center. In particular, we show that we can solve the $3D$ isogeometric L^2 projection problem with $512^3 = 134\,217\,728$ unknowns within 20 seconds and $1\,024^3 = 1\,073\,741\,824 = \mathcal{O}(10^9)$ unknowns within 3 minutes by using 1 000 processors. We are not aware of any other solver delivering such fast solution for 100–1 000 millions of unknowns of in highly-continuous discretizations.

We apply this parallel alternating direction isogeometric solver to an unsteadz nonlinear flow problem in highly-heterogeneous porous media. The solver performs 1 000 time steps in order to simulate the flow through the entire domain. We verify the correctness of the solution by checking the relative error for two simulations with different time steps as well as by controlling the energy of the solution in particular time steps.

The future work may involve replacement of c^2 sequential solves over a face of the $3D$ cube by c^2 parallel solves executed within rows of a cube of processors, utilizing the parallel multi-frontal one dimensional isogeometric solver [32]. This however will not affect the general scalability of the solver, since at this point the integration time is dominating the entire solution. An alternative way of improvement of the solver scalability would be to consider some fast integration schemes for B-spline basis functions. It may also include generalization of the method to non-uniform adapted grids with T-splines technique [21]. We also consider expression of the alternating direction algorithm by graph grammar productions and Petri nets, as it has been done for two and three dimensional finite element method [43, 36, 37, 44].

Acknowledgment

The work was supported by the National Science Centre, Poland grant No. 2012/07/B/ST6/01229. The visits of Maciej Woźniak, Marcin Łoś and Maciej Paszyński to the King Abdullah University of Science and Technology (KAUST) were supported by the Center for Numerical Porous Media (NumPor). This publication was also made possible in part by the CSIRO Professorial Chair in Computational Geoscience at Curtin University, the National Priorities Research Program grant 7-1482-1-278 from the Qatar National Research Fund (a member of The Qatar Foundation), and by the European Union's Horizon 2020 Research and Innovation Program of the Marie Skodowska-Curie grant agreement No. 644202. The J. Tinsley Oden Faculty Fellowship Research Program at the Institute for Computational Engineering and Sciences (ICES) of the University of Texas at Austin has partially supported the visits of VMC to ICES.

REFERENCES

[1] Akkerman, I.—Bazilevs, Y.—Calo, V. M.—Hughes, T. J. R.—Hulshoff, S.: The Role of Continuity in Residual-Based Variational Multiscale Modeling of Turbulence. Computational Mechanics, Vol. 41, 2008, pp. 371–378, doi: 10.1007/s00466-007-0193-7.

[2] Alotaibi, M.—Calo, V. M.—Efendiev, Y.—Galvis, J.—Ghommem, M.: Global-Local Nonlinear Model Reduction for Flows in Heterogeneous Porous Media. arXiv:1407.0782 [math.NA].

[3] Bazilevs, Y.—Beirao da Veiga, L.—Cottrell, J. A.—Hughes, T. J. R.—Sangalli, G.: Isogeometric Analysis: Approximation, Stability and Error Estimates for h-Refined Meshes. Mathematical Methods and Models in Applied Sciences, Vol. 16, 2006, pp. 1031–1090, doi: 10.1142/s0218202506001455.

[4] Bazilevs, Y.—Calo, V. M.—Cottrell, J. A.—Hughes, T. J. R.—Reali, A.—Scovazzi, G.: Variational Multiscale Residual-Based Turbulence Modeling for Large Eddy Simulation of Incompressible Flows. Computer Meth-

ods in Applied Mechanics and Engineering, Vol. 197, 2007, pp. 173–201, doi: 10.1016/j.cma.2007.07.016.

[5] Bazilevs, Y.—Calo, V. M.—Zhang, Y.—Hughes, T. J. R.: Isogeometric Fluid-Structure Interaction Analysis with Applications to Arterial Blood Flow. Computational Mechanics, Vol. 38, 2006, No. 4, pp. 310–322, doi: 10.1007/s00466-006-0084-3.

[6] Benson, D. J.—Bazilevs, Y.—De Luycker, E.—Hsu, M. C.—Scott, M.—Hughes, T. J. R.—Belytschko, T.: A Generalized Element Formulation for Arbitrary Basis Functions: From Isogeometric Analysis to XFEM. International Journal for Numerical Methods in Engineering, Vol. 83, 2010, pp. 765–785, doi: 10.1002/nme.2864.

[7] Benson, D. J.—Bazilevs, Y.—Hsu, M.-C.—Hughes, T. J. R.: A Large-Deformation, Rotation-Free Isogeometric Shell. Computer Methods in Applied Mechanics and Engineering, Vol. 200, 2011, pp. 1367–1378, doi: 10.1016/j.cma.2010.12.003.

[8] Beuchler, S.—Pillwein, V.—Zaglmayr, S.: Fast Summation Techniques for Sparse Shape Functions in Tetrahedral hp-FEM. Domain Decomposition Methods in Science and Engineering. Lecture Notes in Computational Science and Engineering, Vol. 91, 2013, pp. 511–518, doi: 10.1007/978-3-642-35275-1_60.

[9] Calo, V. M.—Brasher, N.—Bazilevs, Y.—Hughes, T. J. R.: Multiphysics Model for Blood Flow and Drug Transport with Application to Patient-Specific Coronary Artery Flow. Computational Mechanics, Vol. 43, 2008, No. 1, pp. 161–177, doi: 10.1007/s00466-008-0321-z.

[10] Calo, V. M.—Collier, N. O.—Pardo, D.—Paszyński, M.: Computational Complexity and Memory Usage for Multi-Frontal Direct Solvers Used in p Finite Element Analysis. Procedia Computer Science, Vol. 4, 2011, pp. 1854–1861, doi: 10.1016/j.procs.2011.04.201.

[11] Chang, K.—Hughes, T. J. R.—Calo, V. M.: Isogeometric Variational Multiscale Large-Eddy Simulation of Fully-Developed Turbulent Flow over a Wavy Wall. Computers and Fluids, Vol. 68, 2012, pp. 94–104, doi: 10.1016/j.compfluid.2012.06.009.

[12] Dalcin, L.—Collier, N.—Vignal, P.—Côrtes, A. M. A.—Calo, V. M.: PetIGA: A Framework for High-Performance Isogeometric Analysis. Computer Methods in Applied Mechanics and Engineering, Vol. 308, 2016, pp. 151–181.

[13] Collier, N.—Dalcin, L.—Calo, V. M.: On the Computational Efficiency of Isogeometric Methods for Smooth Elliptic Problems Using Direct Solvers. International Journal for Numerical Methods in Engineering, Vol. 100, 2014, No. 8, pp. 620–632.

[14] Collier, N.—Dalcin, L.—Pardo, D.—Calo, V. M.: The Cost of Continuity: Performance of Iterative Solvers on Isogeometric Finite Elements. SIAM Journal on Scientific Computing, Vol. 35, 2013, No. 2, pp. A767–A784.

[15] Collier, N. O.—Pardo, D.—Paszyński, M.—Dalcín—Calo, V. M.: The Cost of Continuity: A Study of the Performance of Isogeometric Finite Elements Using Direct Solvers. Computer Methods in Applied Mechanics and Engineering, Vols. 213–216, 2012, pp. 353–361, doi: 10.1016/j.cma.2011.11.002.

[16] Cottrel, J. A.—Hughes, T. J. R.—Bazilevs, Y.: Isogeometric Analysis: Toward Integration of CAD and FEA. Wiley, 2009.

[17] Dedè, L.—Hughes, T. J. R.—Lipton, S.—Calo, V. M.: Structural Topology Optimization with Isogeometric Analysis in a Phase Field Approach. 16th US National Congress of Theoretical and Applied Mechanics (USNCTAM 2010), 2010.

[18] Dedè, L.—Borden, M. J.—Hughes, T. J. R.: Isogeometric Analysis for Topology Optimization with a Phase Field Model. ICES REPORT 11-29, The Institute for Computational Engineering and Sciences, The University of Texas at Austin, 2011.

[19] Demkowicz, L.: Computing with hp-Adaptive Finite Element Method. Vol. I. One and Two Dimensional Elliptic and Maxwell Problems. Chapmann & Hall/CRC Applied Mathematics & Nonlinear Science, 2006, doi: 10.1201/9781420011685.

[20] Demkowicz, L.—Kurtz, J.—Pardo, D.—Paszyński, M., Rachowicz, W.—Zdunek, A.: Computing with hp-Adaptive Finite Element Method. Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems. Chapmann & Hall/CRC Applied Mathematics & Nonlinear Science, 2007.

[21] Dorfel, M. R.—Juttler, B.—Simeon, B.: Adaptive Isogeometric Analysis by Local h-Refinement with T-Splines. Computer Methods in Applied Mechanics and Engineering, Vol. 199, 2010, No. 5–8, pp. 264–275.

[22] Duddu, R.—Lavier, L.—Hughes, T. J. R.—Calo, V. M.: A Finite Strain Eulerian Formulation for Compressible and Nearly Incompressible Hyper-Elasticity Using High-Order NURBS Elements. International Journal of Numerical Methods in Engineering, Vol. 89, 2012, No. 6, pp. 762–785.

[23] Duff, I. S.—Reid, J. K.: The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems. ACM Transactions on Mathematical Software, Vol. 9, 1983, pp. 302–325, doi: 10.1145/356044.356047.

[24] Duff, I. S.—Reid, J. K.: The Multifrontal Solution of Unsymmetric Sets of Linear Systems. SIAM Journal on Scientific and Statistical Computing, Vol. 5, 1984, pp. 633–641, doi: 10.1137/0905045.

[25] Gao, L.—Calo, V. M.: Fast Isogeometric Solvers for Explicit Dynamics. Computer Methods in Applied Mechanics and Engineering, Vol. 274, 2014, pp. 19–41, doi: 10.1016/j.cma.2014.01.023.

[26] Gao, L.—Calo, V. M.: Preconditioners Based on the Alternating-Direction-Implicit Algorithm for the 2D Steady-State Diffusion Equation with Orthotropic Heterogeneous Coefficients. Journal of Computational and Applied Mathematics, Vol. 273, 2015, pp. 274–295.

[27] Geng, P.—Oden, T. J.—van de Geijn, R. A.: A Parallel Multifrontal Algorithm and Its Implementation. Computer Methods in Applied Mechanics and Engineering, Vol. 149, 2006, pp. 289–301, doi: 10.1016/S0045-7825(97)00052-2.

[28] Gómez, H.—Calo, V. M.—Bazilevs, Y.—Hughes, T. J. R.: Isogeometric Analysis of the Cahn-Hilliard Phase-Field Model. Computer Methods in Applied Mechanics and Engineering, Vol. 197, 2008, pp. 4333–4352, doi: 10.1016/j.cma.2008.05.003.

[29] Gómez, H.—Hughes, T. J. R.—Nogueira, X.—Calo, V. M.: Isogeometric Analysis of the Isothermal Navier-Stokes-Korteweg Equations. Computer Methods in Applied Mechanics and Engineering, Vol. 199, 2010, pp. 1828–1840, doi: 10.1016/j.cma.2010.02.010.

[30] Hossain, S.—Hossainy, S. F. A.—Bazilevs, Y.—Calo, V. M.—Hughes, T. J. R.: Mathematical Modeling of Coupled Drug and Drug-Encapsulated Nanoparticle Transport in Patient-Specific Coronary Artery Walls. Computational Mechanics, Vol. 49, 2012, No. 2, pp. 213–242, doi: 10.1007/s00466-011-0633-2.

[31] Hsu, M.-C.—Akkerman, I.—Bazilevs, Y.: High-Performance Computing of Wind Turbine Aerodynamics Using Isogeometric Analysis. Computers and Fluids, Vol. 49, 2011, No. 1, pp. 93–100.

[32] Kuźnik, K.—Paszyński, M.—Calo, V.: Grammar Based Multi-Frontal Solver for Isogeometric Analysis in 1D. Computer Science, Vol. 14, 2013, No. 4, pp. 589–613.

[33] LONESTAR Linux Cluster User Guide, `https://www.tacc.utexas.edu/user-services/user-guides/lonestar-user-guide`, Texas Advanced Computing Center, 2014.

[34] Łoś, M.—Woźniak, M.—Paszyński, M.—Lenharth, A.—Amber Hassaan, M.—Pingali, K.: IGA-ADS: Isogeometric Analysis FEM Using ADS Solver. Computer Physics Communications, Vol. 217, 2017, pp. 99–116.

[35] MUlti-Frontal Massivelly Parallel Sparse Direct Solver, `http://graal.ens-lyon.fr/MUMPS/`.

[36] Paszyńska, A.—Grabska, E.—Paszyński, M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part I. Fundamenta Informaticae, Vol. 114, 2012, pp. 149–182.

[37] Paszyńska, A.—Grabska, E.—Paszyński, M.: A Graph Grammar Model of the hp Adaptive Three Dimensional Finite Element Method. Part II. Fundamenta Informaticae, Vol. 114, 2012, pp. 183–201.

[38] Paszyński, M.—Jurczyk, T.—Pardo, D.: Multi-Frontal Solver for Simulations of Linear Elasticity Coupled with Acoustics. Computer Science, Vol. 12, 2013, pp. 85–102.

[39] Paszyński, M.—Pardo, D.—Torres-Verdin, C.—Demkowicz, L.—Calo, V. M.: A Parallel Direct Solver for Self-Adaptive hp Finite Element Method. Journal of Parallel and Distributed Computing, Vol. 70, 2010, pp. 270–281, doi: 10.1016/j.jpdc.2009.09.007.

[40] Paszyński, M.—Pardo, D.—Paszyńska, A.: Parallel Multi-Frontal Solver for p Adaptive Finite Element Modeling of Multi-Physics Computational Problems. Journal of Computational Science, Vol. 1, 2010, pp. 48–54, doi: 10.1016/j.jocs.2010.03.002.

[41] Paszyński, M.—Schaefer, R.: Graph Grammar Driven Partial Differential Equations Solver. Concurrency and Computations: Practice and Experience Vol. 22, 2010, No. 9, pp. 1063–1097.

[42] Sarmiento, A. F.—Côrtes, A. M. A.—Garcia, D. A.—Dalcin, L.—Collier, N.—Calo, V. M.: PetIGA-MF: A Multi-Field High-Performance Toolbox for Structure-Preserving B-Splines Spaces. Journal of Computational Science, Vol. 18, 2017, pp. 117–131.

[43] Strug, B.—Paszyńska, A.—Paszyński, M.—Grabska, E.: Using the System of Graph Grammar in Finite Element Method. International Journal of Applied Mathematics and Computer Science, Vol. 23, 2014, No. 4, pp. 1140–1151.

[44] Szymczak, A.—Paszyński, M.—Pardo, D.—Paszyńska, A.: Petri Nets Modeling Dead-End Refinement Problems in 3*D* Anisotropic hp-Adaptive Finite Element Method. Computing and Informatics, Vol. 34, 2015, No. 2, pp. 425–457.

[45] Verhoosel, C. V.—Scott, M. A.—Hughes, T. J. R.—de Borst, R.: An Isogeometric Analysis Approach to Gradient Damage Models. International Journal for Numerical Methods in Engineering, Vol. 86, 2011, pp. 115–134, doi: 10.1002/nme.3150.

[46] Woźniak, M.—Kuźnik, K.—Paszyński, M.—Calo, V. M.—Pardo, D.: Computational Cost Estimates for Parallel Shared Memory Isogeometric Multi-Frontal Solvers. Computers and Mathematics with Applications, Vol. 67, 2014, No. 10, pp. 1864–1883, doi: 10.1016/j.camwa.2014.03.017.

[47] Woźniak, M.—Kuźnik, K.—Paszynski, M.—Pardo, D.—Calo, V. M.: Computational Cost of Isogeometric Multi-Frontal Solvers on Distributed Memory Parallel Machines. Computer Methods in Applied Mechanics and Engineering, Vol. 284, 2015, pp. 971–987, doi: 10.1016/j.cma.2014.11.020.

Maciej Woźniak is a fourth year Ph.D. student of computer science in AGH University of Science and Technology, Kraków, Poland. He received his M.Sc. degree in computer science in 2013. Since 2012 he is a member of Prof. Maciej Paszyński research group, working primarily on fast parallel direct solvers for isogeometric finite element methods targeting different parallel architectures.

Marcin Łoś is a second year Ph.D. student of computer science in AGH University of Science and Technology, Kraków, Poland. He received his M.Sc. degree in computer science in 2015. Since 2015 he is a member of Prof. Maciej Paszyński research group, working primarily on different simulations using isogeometric finite element methods.

Maciej Paszyński received his Ph.D. (2003) in mathematics with applications to computer science from the Jagiellonian University, Kraków, Poland and habilitation (2010) in computer science from the AGH University of Science and Technology, Kraków, Poland. His research interests include parallel direct solvers for isogeometric finite element method and computational science. He was a frequent visiting professor at The University of Texas at Austin, the King Abdullah University of Science and Technology and the University of The Basque Country. He holds a position as an affiliated professor of the Department of Computer Science at AGH University of Science and Technology of Kraków, Poland.

Lisandro Dalcin received his Ph.D. in engineering from Universidad Nacional del Litoral (Santa Fe, Argentina) in 2008. His first degree is in electromechanical engineering from Universidad Tecnológica Nacional (Concepción del Uruguay, Argentina). In 2010, he joined the National Council for Scientific and Technological Research of Argentina (Consejo Nacional de Investigaciones Cientificas y Tecnológicas) as Assistant Researcher. He is the author of *mpi4py* and *petsc4py/slepc4py* – a set of bindings for the MPI standard and PETSc/SLEPc libraries targeting parallel distributed computing with the high-level scripting language Python. He is a semi-regular contributor to the Cython project and also contributes to the development of the PETSc library. He won the R & D 100 Award in 2009 as part of the PETSc team. Since 2013 he is a postdoctoral fellow at King Abdullah University of Science and Technology in Thuwal, Saudi Arabia. His research interests include scientific computing in distributed memory architectures, medium to large scale finite element simulation software development and programming tools mixing Python and C/C++ and Fortran.

Victor Manuel Calo is Professor in the Department of Applied Geology of the Western Australian School of Mines in the Faculty of Science and Engineering at Curtin University. He is a highly cited researcher who is actively involved in disseminating knowledge: he has authored over 150 peer-reviewed publications. Also, in the last two years, he has given more than 25 invited presentations and keynotes at conferences and seminars, and organized 15 mini-symposia at international conferences. He holds a professional engineering degree in civil engineering from the University of Buenos Aires. He received his M.Sc. in geomechanics and Ph.D. in civil and environmental engineering from Stanford University. His research interests include modeling and simulation of geomechanics, fluid dynamics, flow in porous media, phase separation, fluid-structure interaction, solid mechanics, and high-performance computing.