An algorithm for tensor product approximation of three-dimensional material data for implicit dynamics simulations

Krzysztof Podsiadło, Marcin Łoś, Leszek Siwik, and Maciej Woźniak

AGH University of Science and Technology, Krakow, Poland,
{podsiadlo,los,siwik,wozniak}@agh.edu.pl

Abstract. In the paper, a heuristic algorithm for tensor product approximation with B-spline basis functions of three-dimensional material data is presented. The algorithm has an application as a preconditioner for implicit dynamics simulations of a non-linear flow in heterogeneous media using alternating directions method. As the simulation use-case, a non-stationary problem of liquid fossil fuels exploration with hydraulic fracturing is considered. Presented algorithm allows to approximate the permeability coefficient function as a tensor product what in turn allows for implicit simulations of the Laplacian term in the partial differential equation. In the consequence the number of time steps of the non-stationary problem can be reduced, while the numerical accuracy is preserved.

1 Introduction

The alternating direction solver [1, 2] has been recently applied for numerical simulations of non-linear flow in heterogeneous media using the explicit dynamics [3, 4].

The problem of extraction of liquid fossil fuels with hydraulic fracturing technique has been considered there. During the simulation two (contradictory) goals i.e., the maximization of the fuel extraction and the minimization of the ground water contamination have been considered [4, 14]. The numerical simulations considered there are performed using the explicit dynamics with B-spline basis functions from isogeometric analysis [5] for approximation of the solution [6, 7]. The resulting computational cost of a single time step is linear, however the number of time steps is large due to the Courant-Fredrichs-Lewy (CFL) condition [8]. In other words, the number of time steps grows along with the mesh dimensions.

Our ultimate goal is to extend our simulator for implicit dynamics case, following the idea of the implicit dynamics isogeometric solver proposed in [9]. The problem is that the extension is possible only if the permeability coefficients of the elliptic operator are expressed as the tensor product structure. Thus, we focus on the algorithm approximating the permeability coefficients with tensor products iteratively.

The algorithm is designed to be a preconditioner for the implicit dynamics solver. With such the preconditioner the number of time steps of the non-stationary problem can be reduced, while the numerical accuracy preserved.

Our method presented in this paper is an alternative for other methods available for approximating coefficients of the model, e.g., adaptive cross approximation [15].

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

2 Explicit and implicit dynamics simulations

Following the model of the non-linear flow in heterogeneous media presented in [1] we start with our explicit dynamics formulation of the problem of non-linear flow in heterogeneous media where we seek for the pressure scalar field u:

$$\left(\frac{\partial u(x,y,z)}{\partial t}, v(x,y,z)\right) = \left(\left(K(x,y,z)e^{\mu u(x,y,z)}\right)\nabla u(x,y,z), \nabla v(x,y,z)\right) + \left(f(x,y,z), v(x,y,z)\right) \quad \forall v \in V \tag{1}$$

Here μ stands for the dynamic permeability constant, $K(x,y,z)$ is a given permeability map, and $f(x,y,z)$ represents sinks and sources of the pressure, modeling pumps and sinks during the exploration process.

The model of non-linear flow in heterogeneous media is called exponential model [12] and is taken from [10] and [11].

In the model, the permeability consists of two parts, i.e., the static one depending on the terrain properties, and the dynamic one reflecting the influence of the actual pressure.

The broad range of the variable known as the saturated hydraulic conductivity along with the functional forms presented above, confirm the nonlinear behavior of the process.

The number of time steps of the resulting explicit dynamics simulations are bounded by the CFL condition [8], requesting to reduce the time step size when increasing the mesh size. This is important limitation of the method, and can be overcome by deriving the implicit dynamics solver.

Following the idea of the implicit dynamics solvers presented in [9], we move the operator to the left-hand side:

$$\left(\frac{\partial u}{\partial t}, v\right) - \left(\left(K(x,y,z)e^{\mu u(x,y,z)}\right)\nabla u, \nabla v\right) = (f, v) \quad \forall v \in V, \tag{2}$$

where we skip all arguments but the permeability operator.

In order to proceed with the alternating directions solver, the operator on the left-hand-side needs to be expressed as a tensor product:

$$\left(\frac{\partial u}{\partial t}, v\right) - \left(\left(K(x)e^{\mu \bar{u}(x)}K(y)e^{\mu \bar{u}(y)}K(z)e^{\mu \bar{u}(z)}\right)\nabla u, \nabla v\right) = (f, v) + \left(K(x)K(y)K(z)e^{\mu \bar{u}(x)}e^{\mu \bar{u}(y)}e^{\mu \bar{u}(z)} - K(x,y,z)e^{\mu u(x,y,z)}\nabla u, \nabla v\right) \quad \forall v \in V \tag{3}$$

It is possible if we express the static permeability in a tensor product form:

$$K(x,y,z) = K(x)K(y)K(z) \tag{4}$$

using our tensor product approximation algorithm described in section 3.

Additionally, we need to replace the dynamic permeability with an arbitrary selected tensor product representation:

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

$$u(x,y,z) = \overline{u}(x)\overline{u}(y)\overline{u}(z) \tag{5}$$

It can be done by adding and subtracting from the left and the right hand sides the selected tensor product representation.

One simple way to do that is to compute the average values of u along particular cross-sections, namely using:

$$u(x,y,z) = \sum_{i=1}^{N_x}\Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(d_{ijk}B_{i,p}(x)B_{j,p}(y)B_{k,p}(z)\Big)\Big)\Big) \tag{6}$$

so we define:

$$\overline{u}(x) = \sum_{i=1}^{N_x}\overline{u}_i B_{i,p}(x) \tag{7}$$

$$\overline{u}(y) = \sum_{j=1}^{N_y}\overline{u}_j B_{j,p}(y) \tag{8}$$

$$\overline{u}(z) = \sum_{k=1}^{N_z}\overline{u}_k B_{k,p}(z) \tag{9}$$

and

$$\overline{u}_i = \frac{\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}(d_{ijk})\Big)}{N_y N_z};\quad \overline{u}_j = \frac{\sum_{i=1}^{N_x}\Big(\sum_{k=1}^{N_x}(d_{ijk})\Big)}{N_x N_z};\quad \overline{u}_k = \frac{\sum_{i=1}^{N_x}\Big(\sum_{j=1}^{N_y}(d_{ijk})\Big)}{N_x N_y} \tag{10}$$

In other words, we approximate the static permeability and we replace the dynamic permeability.

Finally we introduce the time steps, so we deal with the dynamic permeability explicitly, and with the static permeability implicitly:

$$\Big(u_{t+1},\upsilon\Big) - \Big(\Big(K(x)e^{\mu\overline{u}_t(x)}K(y)e^{\mu\overline{u}_t(y)}K(z)e^{\mu\overline{u}_t(z)}\Big)\nabla u_{t+1},\nabla\upsilon\Big) =$$
$$(f,\upsilon) + \Big(K(x)K(y)K(z)e^{\mu\overline{u}(x)}e^{\mu\overline{u}(y)}e^{\mu\overline{u}(z)} - K(x,y,z)e^{\mu u_t(xyz)}\nabla u_t,\nabla\upsilon\Big)\quad \forall\upsilon\in V \tag{11}$$

In the following part of the paper the algorithm for expression of an arbitrary material data function as the tensor product of one dimensional functions that can be utilized in the implicit dynamics simulator is presented.

3 Kronecker product approximation

As an input of our algorithm we take a scalar function defined over the cube shape three-dimensional domain. We call this function a bitmap, since often the material data is given in a form of a discrete 3D bitmap.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

First, we approximate this bitmap with B-spline basis functions using fast, linear computational cost isogeometric L2 projections algorithm.

$$Bitmap(x,y,z) \approx \sum_{i=1}^{N_x}\Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(d_{ijk}B_{i,p}(x)B_{j,p}(y)B_{k,p}(z)\Big)\Big)\Big) \tag{12}$$

Now, our computational problem can be stated as follows:

Problem 1. We seek coefficients $a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z$ to get the minimum of

$$\begin{aligned} &F(a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z)\\ &=\int_\Omega\Big[\Big(\sum_{i=1}^{N_x}a_iB_{i,p}^x\Big)\Big(\sum_{j=1}^{N_y}b_jB_{j,p}^y\Big)\Big(\sum_{k=1}^{N_z}c_kB_{k,p}^z-\sum_{i=1}^{N_x}\Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\big(d_{ijk}B_{i,p}(x)B_{j,p}(y)B_{k,p}(z)\big)\Big)\Big)\Big]^2\\ &=\int_\Omega\big[\sum_{i=1}^{N_x}\big(\sum_{j=1}^{N_y}\big(\sum_{k=1}^{N_z}\big(a_ib_jc_k-d_{ijk}B_{i,p}(x)B_{j,p}(y)B_{k,p}(z)\big)\big)\big)\big]^2 \end{aligned} \tag{13}$$

The minimum is realized when the partial derivatives are equal to zero:

$$\frac{\partial F}{\partial a_l^x}(a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z)=0 \tag{14}$$

$$\frac{\partial F}{\partial b_l^y}(a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z)=0 \tag{15}$$

$$\frac{\partial F}{\partial c_l^z}(a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z)=0 \tag{16}$$

We compute these partial derivatives:

$$\begin{aligned} &\frac{\partial F}{\partial a_l^x}(a_1^x,\ldots,a_{N_x}^x,b_1^y,\ldots,b_{N_y}^y,c_1^z,\ldots,c_{N_z}^z)=0\\ &=\int_\Omega\big[\sum_{j=1}^{N_y}\big(\sum_{k=1}^{N_z}\big(2(a_lb_jc_k-d_{ljk})\Big(\frac{\partial(a_ib_jc_k)}{\partial a_l^x}-\frac{\partial(d_{ijk})}{\partial a_l^x}\Big)B_{l,p}^xB_{j,p}^yB_{k,p}^z)\big)\big]=0, \end{aligned} \tag{17}$$

where the internal term:

$$\frac{\partial(a_ib_jc_k)}{\partial a_l^x}=\frac{\partial(a_i)b_jc_k}{\partial a_l^x}+a_i\frac{\partial(b_jc_k)}{\partial a_l^x}=b_jc_k\delta_{il}+0, \tag{18}$$

thus

$$=\int_\Omega\big[\sum_{j=1}^{N_y}\big(\sum_{k=1}^{N_z}\big(2(a_lb_jc_k-d_{ljk})b_jc_kB_{l,p}^xB_{j,p}^yB_{k,p}^z\big)\big]=0,\quad l=1,\ldots,N_x \tag{19}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

Similarly we proceed with the rest of partial derivatives to obtain:

$$= \int_{\Omega} \Big[\sum_{i=1}^{N_x} \Big(\sum_{k=1}^{N_z} \big(2(a_i b_l c_k - d_{ilk}) a_i c_k B^x_{i,p} B^y_{l,p} B^z_{k,p} \big) \Big] = 0, \quad l = 1, \ldots, N_y \tag{20}$$

$$= \int_{\Omega} \Big[\sum_{i=1}^{N_x} \Big(\sum_{j=1}^{N_y} \big(2(a_i b_j c_l - d_{ijl}) a_i b_j B^x_{i,p} B^y_{j,p} B^z_{l,p} \big) \Big] = 0, \quad l = 1, \ldots, N_z \tag{21}$$

This is equivalent to the following system of equations:

$$\sum_{j=1}^{N_y} \Big(\sum_{k=1}^{N_z} 2\big(a_l b_j c_k - d_{ljk}\big) b_j c_k \Big) = 0 \tag{22}$$

$$\sum_{i=1}^{N_x} \Big(\sum_{k=1}^{N_z} 2\big(a_i b_l c_k - d_{ilk}\big) a_i c_k \Big) = 0 \tag{23}$$

$$\sum_{i=1}^{N_x} \Big(\sum_{j=1}^{N_y} 2\big(a_i b_j c_c - d_{ijl}\big) a_i b_j \Big) = 0 \tag{24}$$

We have just got a non-linear system of $N_x + N_y + N_z$ equations with $N_x + N_y + N_z$ unknowns:

$$a_l \Big(\sum_{j=1}^{N_y} \Big(\sum_{k=1}^{N_z} \big(b_j c_k\big) b_j c_k \big) \Big) = \sum_{j=1}^{N_y} \Big(\sum_{k=1}^{N_z} \big(d_{ljk} b_j c_k\big) \Big) \tag{25}$$

$$b_l \Big(\sum_{i=1}^{N_x} \Big(\sum_{k=1}^{N_z} \big(a_i c_k\big) a_i c_k \big) \Big) = \sum_{i=1}^{N_x} \Big(\sum_{k=1}^{N_z} \big(d_{ilk} a_i c_k\big) \Big) \tag{26}$$

$$c_l \Big(\sum_{i=1}^{N_x} \Big(\sum_{j=1}^{N_y} \big(a_i b_j\big) a_i b_j \big) \Big) = \sum_{i=1}^{N_x} \Big(\sum_{j=1}^{N_y} \big(d_{ijl} a_i b_j\big) \Big), \tag{27}$$

what implies:

$$a_l = \frac{\sum_{j=1}^{N_y} \big(\sum_{k=1}^{N_z} d_{ljk} b_j c_k \big)}{\sum_{j=1}^{N_y} \big(\sum_{k=1}^{N_z} \big(b_j c_k\big)^2 \big)} \tag{28}$$

$$b_l = \frac{\sum_{i=1}^{N_x} \big(\sum_{k=1}^{N_z} d_{ilk} a_i c_k \big)}{\sum_{i=1}^{N_x} \big(\sum_{k=1}^{N_z} \big(a_i c_k\big)^2 \big)} \tag{29}$$

We insert these coefficients into the third equation:

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

$$c_l \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\frac{\sum_{m=1}^{N_y} \left(\sum_{n=1}^{N_z} d_{imn} b_m c_n \right)}{\sum_{m=1}^{N_y} \left(\sum_{n=1}^{N_z} (b_m c_n)^2 \right)} \right)^2 \left(\frac{\sum_{m=1}^{N_x} \left(\sum_{n=1}^{N_z} d_{mjn} a_m c_n \right)}{\sum_{m=1}^{N_x} \left(\sum_{n=1}^{N_z} (a_m c_n)^2 \right)} \right)^2 \right) = \\ = \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} d_{ijl} \frac{\sum_{m=1}^{N_y} \left(\sum_{n=1}^{N_z} d_{imn} b_m c_n \right)}{\sum_{m=1}^{N_y} \left(\sum_{n=1}^{N_z} (b_m c_n)^2 \right)} \frac{\sum_{m=1}^{N_x} \left(\sum_{n=1}^{N_z} d_{mjn} a_m c_n \right)}{\sum_{m=1}^{N_x} \left(\sum_{n=1}^{N_z} (a_m c_n)^2 \right)} \right) \tag{30}$$

$$c_l \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{m=1}^{N_y} \left(\sum_{n=1}^{N_z} d_{imn} b_m c_n \right) \right) \left(\sum_{m=1}^{N_x} \left(\sum_{n=1}^{N_z} d_{mjn} a_m c_n \right) \right) \right) = \\ = \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} d_{ijl} \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} (b_m c_n)^2 \right) \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_x} (a_m c_n)^2 \right) \right) \tag{31}$$

$$c_l \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} d_{imn} b_m c_n \right) \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_x} d_{mjn} a_m c_n \right) \right) \right) = \\ = \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} d_{ijl} \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} (b_m c_n)^2 \right) \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_x} (a_m c_n)^2 \right) \right) \tag{32}$$

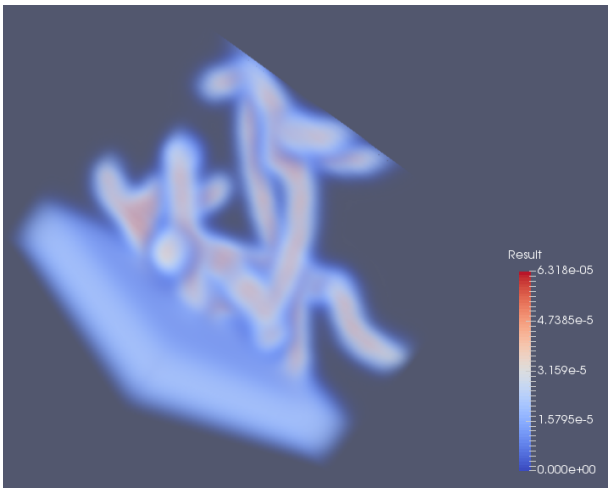


Fig. 1. The original configuration of static permeability

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

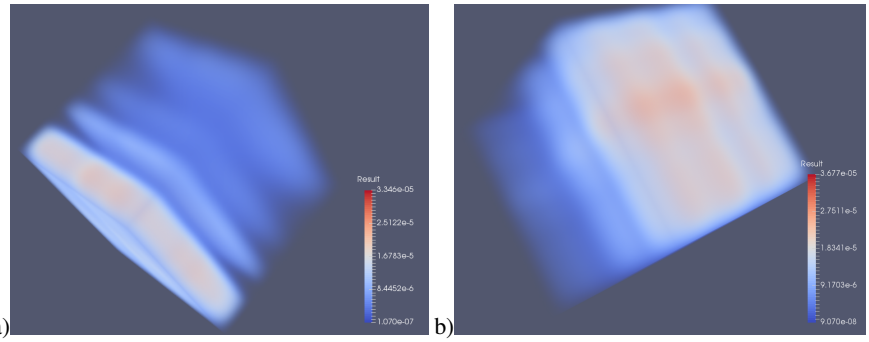


Fig. 2. The result obtained from the heuristic algorithm (a) and from the heuristic plus genetic algorithms (b).

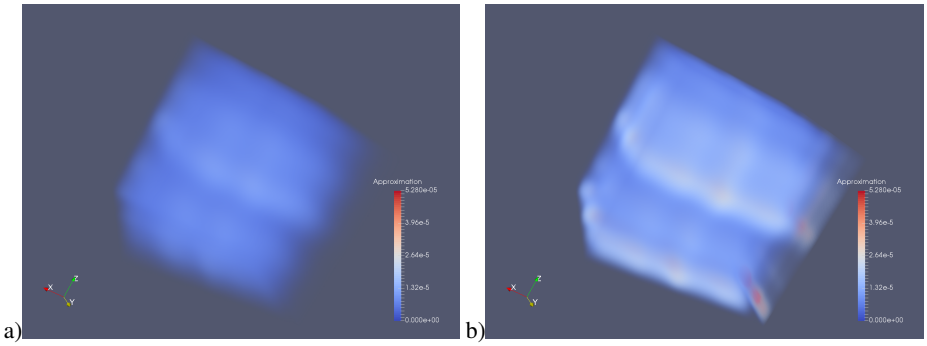


Fig. 3. The tensor product approximation after one (a) and five (b) iterations of Algorithm 1.

$$
\begin{aligned}
& c_l \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} d_{imn} b_m c_n \right) \right) \left(\sum_{m=1}^{N_x} d_{mjn} a_m c_n \right) \right) = \\
& = \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} d_{ijl} \right) \left(\sum_{m=1}^{N_y} \left(b_m c_n \right)^2 \right) \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_x} \left(a_m c_n \right)^2 \right) \right)
\end{aligned} \tag{33}
$$

$$
\begin{aligned}
& \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} \left(\sum_{o=1}^{N_x} d_{ojn} a_o c_n d_{imn} b_m c_n c_l \right) \right) \right) \right) = \\
& = \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{n=1}^{N_z} \left(\sum_{m=1}^{N_y} \left(\sum_{o=1}^{N_x} (a_o c_n b_m c_n)^2 d_{ijl} \right) \right) \right) \right)
\end{aligned} \tag{34}
$$

The above is true when

$$
d_{imn} b_m c_n c_l d_{ojn} a_o c_n = (a_o c_n b_m c_n)^2 d_{ijl}, \tag{35}
$$

so:

$$
d_{imn} c_l d_{ojn} = a_o c_n b_m c_n d_{ijl} \tag{36}
$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

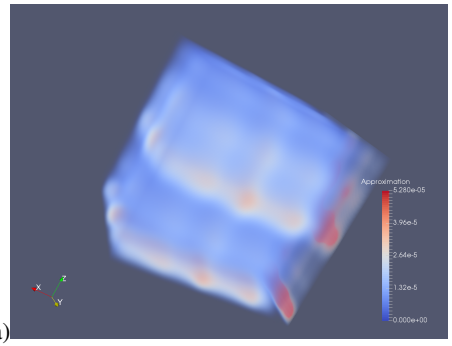
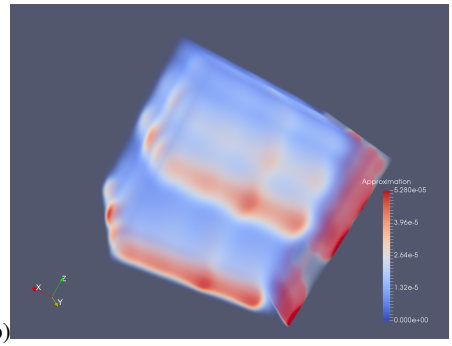
a)  b) 

Fig. 4. The tensor product approximation after ten (a) and fifty (b) iterations of Algorithm 1.

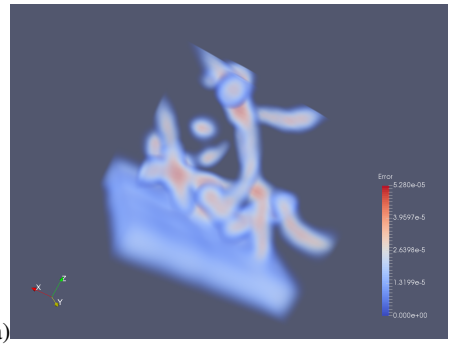
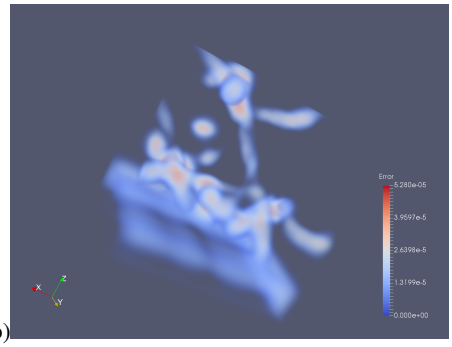
a)  b) 

Fig. 5. The error of the tensor product approximation after one (a), and five (b) iterations of Algorithm 1.

thus:

$$\frac{d_{ojn}d_{imn}}{d_{ijl}} = \frac{a_o c_n b_m c_n}{c_l} \tag{37}$$

We can setup now a_1, b_1, and c_1 arbitrary and compute c_l using the derived proportions.

In a similar way we compute a_l, namely we insert:

$$b_l = \frac{\sum_{i=1}^{N_x}\left(\sum_{k=1}^{N_z} d_{ilk}a_i c_k\right)}{\sum_{i=1}^{N_x}\left(\sum_{k=1}^{N_z}\left(a_i c_k\right)^2\right)} \tag{38}$$

$$c_l = \frac{\sum_{i=1}^{N_x}\left(\sum_{j=1}^{N_y} d_{ijl}a_i b_j\right)}{\sum_{i=1}^{N_x}\left(\sum_{j=1}^{N_y}\left(a_i b_j\right)^2\right)} \tag{39}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

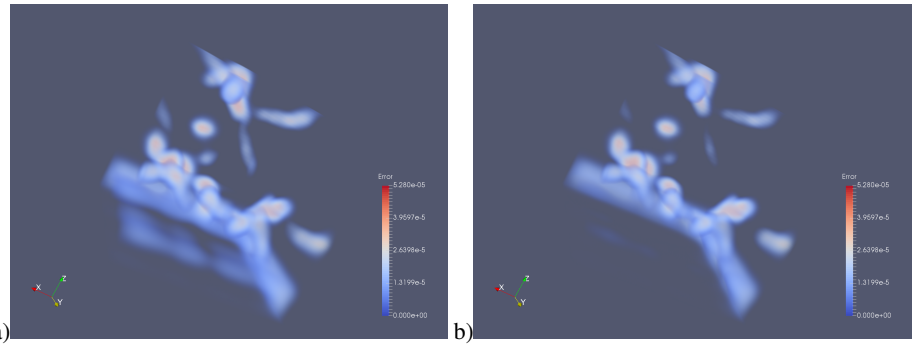


Fig. 6. The error of the tensor product approximation after ten (a), and fifty (b) iterations of Algorithm 1.

into

$$a_l\Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}(d_{mjn}a_mc_n)\Big)\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_y}(d_{mnk}a_mb_n)\Big)\Big)\Big)\Big)\Big) = \\ = \Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}d_{ljk}\Big)\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}(a_mc_n)^2\Big)\Big)\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_y}(a_mb_n)^2\Big)\Big)\Big), \tag{40}$$

then:

$$a_l\Big(\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}d_{mjn}a_mc_n\Big)\Big(\sum_{o=1}^{N_y}d_{mok}a_mb_o\Big)\Big)\Big)\Big)\Big) = \\ = \sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}d_{ljk}\Big)\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}(a_mc_n)^2\Big)\Big)\Big(\sum_{m=1}^{N_x}\Big(\sum_{o=1}^{N_y}(a_mb_o)^2\Big)\Big), \tag{41}$$

and finally:

$$\sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}\Big(\sum_{o=1}^{N_y}a_ld_{mok}a_mb_od_{mjn}a_mc_n\Big)\Big)\Big)\Big)\Big) = \\ = \sum_{j=1}^{N_y}\Big(\sum_{k=1}^{N_z}\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}\Big(\sum_{m=1}^{N_x}\Big(\sum_{n=1}^{N_z}(a_mb_oa_mc_n)^2d_{ljk}\Big)\Big)\Big)\Big)\Big), \tag{42}$$

what results in:

$$a_ld_{mok}a_mb_od_{mjn}a_mc_n = (a_mb_oa_mc_n)^2d_{ljk}, \tag{43}$$

so:

$$a_ld_{mok}d_{mjn} = a_mb_oa_mc_nd_{ljk}, \tag{44}$$

thus:

$$\frac{d_{mok}d_{mjn}}{d_{ljk}} = \frac{a_mb_oa_mc_n}{a_l} \tag{45}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

We compute b_l from (we already have a_i and c_k):

$$b_l = \frac{\sum_{i=1}^{N_x} \left(\sum_{k=1}^{N_z} d_{ilk} a_i c_k \right)}{\sum_{i=1}^{N_x} \left(\sum_{k=1}^{N_z} \left(a_i c_k \right)^2 \right)} \tag{46}$$

The just analyzed Problem 1 has multiple solutions, and the algorithm presented above finds one exemplary solution, for the assumed values of a_1, b_1, and c_1.

This however may not be the optimal solution, in the sense of equation (13), and thus we may improve the quality of the solution executing simple genetic algorithm, with the individuals representing the parameters $a_1^x, \ldots, a_{N_x}^x, b_1^y, \ldots, b_{N_y}^y, c_1^z, \ldots, c_{N_z}^z$, and with the fitness function defined as (13).

4 Iterative algorithm with evolutionary computations

The heuristic algorithm mixed with the genetic algorithm, as presented in section 3, is not able to find the solution with 0 error, for non-tensor product structures, since we approximate $N*N$ data with $2*N$ unknowns. Thus, the iterative algorithm presented in 1 is proposed, with the assumed accuracy ε.

Algorithm 1 Iterative algorithm with evolutionary computations

1: m=1
2: Bitmap[m](x,y,z)=K(x,y,z)
3: **repeat**
4: Find d_{ijk} for Bitmap[m](x,y,z) $\approx \sum_{i=1}^{N_x} \left(\sum_{j=1}^{N_y} \left(\sum_{k=1}^{N_z} \left(d_{ijk} B_{i,p}(x) B_{j,p}(y) B_{k,p}(z) \right) \right) \right)$ using the linear computational cost isogeometric L2 projection algorithm
5: Find $a_1^x, \ldots, a_{N_x}^x, b_1^y, \ldots, b_{N_y}^y, c_1^z, \ldots, c_{N_z}^z$ to minimize $F[m]\left(a_1^x, \ldots, a_{N_x}^x, b_1^y, \ldots, b_{N_y}^y, c_1^z, \ldots, c_{N_z}^z\right)$ given by (13) using the heuristic algorithm to generate initial population and the genetic algorithm to improve the tensor product approximations
6: $m = m+1$
7: Bitmap[m](x,y,z)=Bitmap[m-1](x,y,z)-$\left(\sum_{i=1}^{N_x} a_i B_{i,p}^x \right) \left(\sum_{j=1}^{N_y} b_j B_{j,p}^y \right) \left(\sum_{k=1}^{N_z} c_k B_{k,p}^z \right)$
8: **until** $F[m]\left(a_1^x, \ldots, a_{N_x}^x, b_1^y, \ldots, b_{N_y}^y, c_1^z, \ldots, c_{N_z}^z\right) \geq \varepsilon$

In the aforementioned algorithm we approximate the static permeability as a sequence of tensor product approximations:

$$K(x,y,z) = \sum_{m=1}^{M} K_m^x(x) K_m^y(y) K_m^z(z) \tag{47}$$

Practically, it is realized according to the following equations:

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

$$\left(u_{t+m},\upsilon\right)-\left(K_m^x(x)e^{\mu\overline{u}_{t+m-1}(x)}\right)\left(K_m^y(x)e^{\mu\overline{u}_{t+m-1}(y)}\right)\left(K_m^z(x)e^{\mu\overline{u}_{t+m-1}(z)}\right)\nabla u_{t+m},\nabla\upsilon\Big)$$
$$=-\sum_{n=1,m\neq n}\left(K_n^x(x)e^{\mu\overline{u}_{t+n}(x)}K_n^y(y)e^{\mu\overline{u}_{t+n}(y)}K_n^z(z)e^{\mu\overline{u}_{t+n}(z)}\nabla u_{t+n},\nabla\upsilon\right)$$
$$+\big(f,\upsilon\big)+K_m^x(x)K_m^y(y)K_m^z(z)$$
$$\left[\left(e^{\mu\overline{u}_{t+m}(x)}e^{\mu\overline{u}_{t+m-1}(y)}e^{\mu\overline{u}_{t+m-1}(z)}\right)-e^{\mu\overline{u}_{t+m-1}(x,y,z)}\right]\nabla u,\nabla\upsilon\Big)\quad\forall\upsilon\in V \tag{48}$$

5 Numerical results

We conclude the paper with the numerical results concerning the approximation of the static permeability map. The original static permeability map is presented in Figure 1. The first approximation has been obtained from the heuristic algorithm described in section 3. We used the formulas (25)-(27) with the suitable substitutions. In the first approach we first compute the values of a, next, the values of b and finally the values of c. As the initial values we picked $\sqrt[3]{d_{111}}$.

Deriving this method further we decided to compute particular points in the order of a_2, b_2, c_2, a_3, b_3 and so on. This gave us the final result presented in Figure 2a.

We have improved the approximation by post-processing with the generational genetic algorithm as implemented in jMetal package [13] with variables from [0,1] intervals. The fitness function was defined as:

$$f(a_1,\ldots,a_{N_x},b_1,\ldots,b_{N_y},c_1,\ldots,c_{N_z})=\sum_{i=1}^{N_x}\sum_{l=1}^{N_y}\sum_{k=1}^{N_z}\left(d_{ilk}-a_ib_lc_k\right)^2 \tag{49}$$

The results are summarized in Figure 2b.

To improve the numerical results we have employed the Algorithm 1. In Figure 3 and Figure 4 results obtained after 1, 5, 10 and 50 iterations of Algorithm 1 are presented.

In order to analyze the accuracy of the tensor product approximation, we also present in Figures 5–6 the error after 1, 5, 10, 50 iterations. We can read from these Figures, how the error decreases when adding particular components.

6 Conclusions and the future work

In the paper the heuristic algorithm for tensor product approximation of material data for implicit dynamics simulations of non-linear flow in heterogeneous media is presented.

The algorithm can be used as a generator of initial configurations for a genetic algorithm, improving the quality of the approximation. The future work will involve the implementation of the implicit scheme and utilizing the proposed algorithms as a preconditioner for obtaining tensor product structure of the material data.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

We have analyzed the convergence of our tensor product approximation method but assessing how the convergence influences the reduction of the iteration number of the explicit method will be the matter of our future experiments.

Our intuition is that 100 iterations (100 components of the tensor product approximation) should give a well approximation, and thus we can use the implicit method not bounded by the CFL condition, which will require 100 sub-steps in every time step.

Acknowledgments

This work was supported by National Science Centre, Poland, grant no. 2014/15/N/ST6/04662. The authors would like to acknowledge prof. Maciej Paszyński for his help in this research topic and preparation of this paper.

References

1. Łoś M, Woźniak M, Paszyński M, Dalcin L, Calo VM (2015) Dynamics with Matrices Possessing Kronecker Product Structure, Procedia Comput Sci 51:286-295. doi:10.1016/j.procs.2015.05.243
2. Łoś M, Woźniak M, Paszyński M, Lenharth A, Amber-Hassan M, Pingali K (2017) IGA-ADS: Isogeometric analysis FEM using ADS solver, Comput Phys Commun 217:99-116. doi:10.1016/j.cpc.2017.02.023
3. Woźniak M, Łoś M, Paszyński M, Dalcin L, Calo VM (2017) Parallel fast isogeometric solvers for explicit dynamics, Comput Inform 36(2):423-448. doi:10.4149/cai.2017.2.423
4. Siwik L, Łoś, Kisiel-Dorohinicki M, Byrski A (2016) Hybridization of isogeometric finite element method and evolutionary mulit-agent system as a tool-set for multi-objective optimization of liquid fossil fuel exploitation with minimizing groundwater contamination, Procedia Comput Sci, 80:792-803. doi:10.1016/j.procs.2016.05.369
5. Łoś M (2015) Fast isogeometric L2 projection solver for non-linear flow in non-homogenous media, Master Thesis, AGH University, Krakow, Poland
6. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput Methods Appl Mech Eng 194(39):4135-4195. doi:10.1016/j.cma.2004.10.008
7. Cottrell JA, Hughes TJR, Bazilevs Y (2009) Isogeometric Analysis: Toward Unfication of CAD and FEA, John Wiley and Sons. The Attrium, Southern Gate, Chichester, West Sussex
8. Courant R, Friedrichs K, Lewy H (1956) On the partial difference equations of mathematical physics, AEC Research and Development Report, NYO-7689, New York: AEC Computing and Applied Mathematics Centre-Courant Institute of Mathematical Sciences
9. Paszyński M, Ło{s, Calo VM (2017) Fast isogeometric solvers for implicit dynamics, submitted to Comput Math Appl
10. Alotaibi M, Calo VM, Efendiev Y, Galvis J, Ghommem M (2015) Global-Local Nonlinear Model Reduction for Flows in Heterogeneous Porous Media, Comput Methods Appl Mech Eng 292:122-137. doi:10.1016/j.cma.2014.10.034
11. Efendiev Y, Ginting V, Hou T (2004) Multiscale finite element methods for nonlinear problems and their applications, Commun Math Sci, 2(4):553-589. doi: 10.4310/CMS.2004.v2.n4.a2
12. Warrick AW (1976) Time-dependent linearized in filtration: III. strip and disc sources, Soil Sci Soc Am J, 40:639-643. doi:

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12

13. Nebro AJ, Durillo JJ, Vergne M (2015) Redesigning the jMetal Multi-Objective Optimization Framework, Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15
14. Siwik L, Łoś M, Kisiel-Dorohinicki M, Byrski A (2016) Evolutionary Multiobjective Optimization of Liquid Fossil Fuel Reserves Exploitation with Minimizing Natural Environment Contamination, Artificial Intelligence and Soft Computing: 15th International Conference, Zakopane, Poland, June 12-16, Proceedings, Part II, 384-394.
15. Goreinov SA, Tyrtyshnikov EE, Zamarashkin NL (1997) A theory of pseudoskeleton approximations, Linear Algebra and its Applications 261(1-3):1-21. doi:10.1016/S0024-3795(96)00301-1

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_12