Computer Science • 18(2) 2017 http://dx.doi.org/10.7494/csci.2017.18.2.117

Marcin Łoś
Maciej Paszyński

APPLICATIONS OF ALTERNATING DIRECTION SOLVER FOR SIMULATIONS OF TIME-DEPENDENT PROBLEMS

Abstract *This paper deals with the application of an Alternating Direction Solver (ADS) to a non-stationary linear elasticity problem solved with the isogeometric finite element method (IGA-FEM). Employing a tensor product B-spline basis in isogeometric analysis under some restrictions leads to a system of linear equations with a matrix possessing a tensor product structure. The ADI algorithm is a direct method that exploits this Kronecker product structure to solve the system in $\mathcal{O}(N)$, where N is the number of degrees of freedom (basis functions). This is asymptotically faster than state-of-the-art, general-purpose, multi-frontal direct solvers when applied to explicit dynamics. In this paper, we also present a complexity analysis of the ADS incorporating dependence on the B-spline basis of order p.*

Keywords finite element method, isogeometric analysis, Alternating Direction Solver

Citation Computer Science 18(2) 2017: 117–128

1. Introduction

Isogeometric analysis is a variant of the Finite Element Method utilizing B-spline basis functions and a geometry description to improve compatibility with existing Computer-Aided Design (CAD) software [4, 13, 15, 17]. For higher-dimensional problems, the basis is customarily constructed as a tensor product of one-dimensional bases. Under some additional conditions, this implies that the matrix of a system of linear equations resulting from the employment of FEM can be decomposed as a tensor product of matrices of one-dimensional problems that are banded and, thus, easily inverted. This structure is exploited by the Alternating Direction Solver (ADS) to solve a system with linear computational cost with respect to the number of degrees of freedom.

The Alternating Direction Implicit (ADI) algorithm has been originally introduced in [2, 6, 16, 19] to solve parabolic problems. This method has been applied in the context of isogeometric analysis to two-dimensional non-stationary problems [8], three-dimensional flow problems [21], and as a preconditioner for iterative solvers in the case of complex geometries [9]. Its main idea is to reduce the problem of solving a linear system with a matrix $\mathbf{M}$ possesing tensor product structure $\mathbf{A}_1 \otimes \mathbf{A}_2 \cdots \otimes \mathbf{A}_d$ to solve multiple linear systems with matrices $\mathbf{A}_i$. If these systems can be solved efficiently, so can the full system with matrix $\mathbf{M}$.

In this paper, we discuss the idea and implementation of the ADS algorithm, establish its complexity taking the order of B-spline basis p into account (thus, extending the results of [8]), and present an application of isogeometric analysis and ADS to a non-stationary three-dimensional linear elasticity problem.

The computational cost of the ADS solver is linear; thus, the ADI solver applied for the solution of a time-dependent problem is able to solve each time step with linear computational cost. In particular, the Newmark time-discretization method with IGA-FEM discretization in space requires an L2 projection problem to be solved at every time step. This projection problem can be solved using the ADS solver in linear $\mathcal{O}(N)$ computational cost. Alternative approaches includes multi-frontal direct solvers [3, 7], which have $\mathcal{O}\left(N^2\right)$ computational cost for 3D problems [3]. Another possibility is to apply iterative solvers; however, iterative methods require $\mathcal{O}(Nk)$ computational cost, where k is the number of iterations, k in a range of $(c, N^{0.5}]$ depending on the conditioning of the system, and the iterative algorithm, where $c \gg 1$ [18]. Thus, the computational cost of our method is one order of magnitude faster than multi-frontal solvers and k times faster than iterative solvers.

One can claim that the implicit method with direct solvers can perform faster; however, these kinds of simulations (fast elastodynamics) require the utilization of explicit methods that can be computed using linear computational cost ADS algorithms. Thus, a comparison with the implicit method makes no sense at all, since direct solvers have a computational cost of $\mathcal{O}\left(N^2\right)$ for 3D IGA-FEM, and they are one order of magnitude slower than our ADS algorithm.

2. Fast isogeometric L2 projections by Alternating Direction Implicit algorithm

The Alternating Direction Implicit (ADI) algorithm is a method of solving systems of linear equations $\mathbf{Mx} = \mathbf{b}$, where $\mathbf{M}$ has a Kronecker product structure, that is

$$\mathbf{M} = \mathbf{M}_1 \otimes \mathbf{M}_2 \otimes \cdots \otimes \mathbf{M}_n \tag{1}$$

and matrices $\mathbf{M}_i$ are square and invertible.[1] This works by reducing the problem of solving multiple systems of equations with $\mathbf{M}_i$ instead of the full $\mathbf{M}$. Under some assumptions, matrices arising from multidimensional L^2-projection problems are separable with respect to dimensions, and the matrices that appear in the decomposition have simple structures, allowing for the efficient application of their inverses.

2.1. Application of ADI solver to L2-projection problem

Let us now see how the ADI solver can be applied to solving L2-projection problems with the tensor product B-spline basis:

$$N^p_{i_1 i_2 i_3}(\mathbf{x}) = N^p_{i_1}(x_1) N^p_{i_2}(x_2) N^p_{i_3}(x_3) \tag{2}$$

The L2-projection problem is equivalent to solving the system of linear equations $\mathbf{Mx} = \mathbf{b}$, where $\mathbf{M}$ is the Gram matrix of vectors comprising the basis. As we shall see, the structure of matrix $\mathbf{M}$ facilitates the use of the ADI solver – $\mathbf{M}$ can be decomposed as a Kronecker product of easily invertible matrices.

Using tensor product basis functions yields a natural Kronecker product structure of matrix $\mathbf{M}$. We have

$$\begin{aligned}
\left(N^p_{i_1 i_2 i_3}, N^p_{j_1 j_2 j_3}\right)_{L^2(\Omega)} &= \int_\Omega N^p_{i_1 i_2 i_3}(\mathbf{x})\, N^p_{j_1 j_2 j_3}(\mathbf{x})\, \mathrm{d}\Omega = \\
&= \int_\Omega N^p_{i_1}(x_1) N^p_{i_2}(x_2) N^p_{i_3}(x_3)\, N^p_{j_1}(x_1) N^p_{j_3}(x_3) N^p_{j_3}(x_3)\, \mathrm{d}\Omega \\
&= \int_\Omega N^p_{i_1}(x_1) N^p_{j_1}(x_1) N^p_{i_2}(x_2) N^p_{j_2}(x_2) N^p_{i_3}(x_3) N^p_{j_3}(x_3)\, \mathrm{d}\Omega
\end{aligned} \tag{3}$$

Our domain Ω is a product of intervals,

$$\Omega = \left[\xi^{(1)}_0, \xi^{(1)}_{n_1}\right] \times \left[\xi^{(2)}_0, \xi^{(2)}_{n_2}\right] \times \left[\xi^{(3)}_0, \xi^{(3)}_{n_3}\right] \tag{4}$$

Therefore, the above integral can be factored into the product of integrals as

$$\begin{aligned}
\left(N^p_{i_1 i_2 i_3}, N^p_{j_1 j_2 j_3}\right)_{L^2(\Omega)} &= \left(\int_{\xi^{(1)}_0}^{\xi^{(1)}_{n_1}} N^p_{i_1} N^p_{j_1}\, \mathrm{d}x_1\right)\left(\int_{\xi^{(2)}_0}^{\xi^{(2)}_{n_2}} N^p_{i_2} N^p_{j_2}\, \mathrm{d}x_2\right)\left(\int_{\xi^{(3)}_0}^{\xi^{(3)}_{n_3}} N^p_{i_3} N^p_{j_3}\, \mathrm{d}x_3\right) \\
&= \left(N^p_{i_1}, N^p_{j_1}\right)_{L^2(\Omega_1)} \left(N^p_{i_2}, N^p_{j_2}\right)_{L^2(\Omega_2)} \left(N^p_{i_3}, N^p_{j_3}\right)_{L^2(\Omega_3)}
\end{aligned} \tag{5}$$

[1] It can be shown that $\operatorname{rank}(\mathbf{A} \otimes \mathbf{B}) = \operatorname{rank}\mathbf{A} \operatorname{rank}\mathbf{B}$, so the Kronecker product is invertible if all of the factors are invertible.

where $\Omega_k = \left[\xi_0^{(k)}, \xi_{n_k}^{(k)}\right]$. Let us denote the Gram matrix of the k-th one-dimensional basis by $\mathbf{M}_k$; i.e.,

$$(M_k)_i^j = \left(N_i^p, N_j^p\right)_{L^2(\Omega_k)} \tag{6}$$

By eq. (5), we have

$$M_{i_1 i_2 i_3}^{j_1 j_2 j_3} = (M_1)_{i_1}^{j_1} (M_2)_{i_2}^{j_2} (M_3)_{i_3}^{j_3} \tag{7}$$

which shows that $\mathbf{M}$ is the Kronecker product of the matrices for one-dimensional bases:

$$\mathbf{M} = \mathbf{M}_1 \otimes \mathbf{M}_2 \otimes \mathbf{M}_3 \tag{8}$$

What remains to be shown is that the matrices for one-dimensional B-spline bases are easily invertible. This is thanks to the locality of support of the B-spline basis functions. Support of N_i^p is $[\xi_{i-p-1}, \xi_i]$, where we assume $\xi_k = \xi_0$ for $k < 0$ and $\xi_k = \xi_n$ for $k > n$; hence, the supports of N_i^p and N_j^p are disjoint for $|i - j| > p$, and so we have

$$\left(N_i^p, N_j^p\right)_{L^2(\Omega)} = 0 \tag{9}$$

Thus, matrix $\mathbf{M}$ is multi-diagonal with $2p + 1$ diagonals.

$$\begin{bmatrix}
M_1^1 & \cdots & M_1^{p+1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
M_2^1 & \cdots & M_2^{p+1} & M_2^{p+2} & 0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & & & & \vdots \\
M_{p+1}^1 & \ddots & \ddots & \ddots & \ddots & M_{p+1}^{2p+2} & 0 & \cdots & 0 \\
0 & M_{p+2}^2 & \ddots & \ddots & \ddots & \ddots & M_{p+2}^{2p+3} & \cdots & 0 \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & M_{n-p}^{n-2p} & \ddots & \ddots & \ddots & \ddots & M_{n-p}^n \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & 0 & M_n^{n-p} & M_n^{n-p+1} & \cdots & M_n^n
\end{bmatrix} \tag{10}$$

where $M_i^j = (B_i, B_j)_{L^2(\Omega)}$. Linear systems with such matrices can be efficiently solved in linear time with respect to the number of degrees of freedom using LU factorization, for example (which is $\mathcal{O}\left(p^2 N\right)$ for matrices with bandwidth p [10, Algorithm 4.3.1]).

3. Implementation

Let $\mathbf{M} = \mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \mathbf{A}_3$, where each $\mathbf{A}_i$ is $n_i \times n_i$ matrix and $\mathbf{b}$ is the right-hand-side vector. Vector $\mathbf{b}$ has $n_1 \times \cdots \times n_d$ components and can be stored in a multidimensional array, organized linearly in the memory so that the lower indices change faster. In this

way, the LAPACK solver can be utilized to solve a linear system with multiple right-hand sides stored in such an array. In the i-th step of the algorithm, linear systems are constructed so that the i-th index is variable in each single system while the rest are constant; hence, after each step, the array needs to be appropriately transposed so that the i-th index changes fastest at the i-th step.

The pseudocode below uses two auxiliary functions:

- SOLVE($\mathbf{A}$, $\mathbf{b}$) – LAPACK solver, takes matrix $\mathbf{A}$ and a sequence of right-hand sides (vectors), solves all of the systems, and overrides each right-hand-side vector with the solution of the corresponding system.
- REORDER($\mathbf{x}, i$) – reorders the multidimensional array representing the right-hand-side vector. This operation assumes $\mathbf{x}$ is ordered such that i-th index is changing fastest and changes this order so that afterwards $(i+1)$-th index changes fastest.

Algorithm 1: Alternating Direction Solver

Input: $\mathbf{A}_1, \mathbf{A}_2, , \mathbf{A}_3$ – matrices
$\mathbf{b}$ – right-hand-side vector
Output: solution $\mathbf{x}$

$\mathbf{x} \leftarrow \mathbf{b}$;
for $i \leftarrow 1$ **to** 3 **do**
  `Solve(`$\mathbf{A}_i$, $\mathbf{x}$`);`
  `Reorder(`$\mathbf{x}$, i`)`
end

The computational complexity of the algorithm is analyzed in Appendix A.

4. Numerical example – linear elasticity

As an example problem, we chose a small strain linear elasticity problem describing the deformation of solid objects in the presence of an external force.

4.1. Formulation

Strong form of the equation is given by

$$\begin{cases} \rho \dfrac{\partial^2 \mathbf{u}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{F} & \text{on } \Omega \times [0,T] \\ \boldsymbol{u}(x,0) = u_0 & \text{for } x \in \Omega \\ \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} = 0 & \text{on } \partial\Omega \end{cases} \tag{11}$$

where $\Omega = [0,1]^3$ is an unit cube, $\mathbf{u}$ is a 3-dimensional displacement vector to be calculated, ρ is the material density, $\mathbf{F}$ is the applied external force density, and $\boldsymbol{\sigma}$ is the rank-2 stress tensor is given by

$$\sigma_{ij} = c_{ijkl}\epsilon_{lk} \tag{12}$$

where

$$\epsilon_{ij} = \frac{1}{2}\left(\partial_j u_i + \partial_i u_j\right) \tag{13}$$

and $\mathbf{c}$ is a rank-4 elasticity tensor (Einstein's summation convention applies in the preceding equation). For our simulations, we assumed an isotropic material with Lamé coefficients $\lambda = \mu = 1$.

4.2. Time discretization

In order to utilize the forward Euler integration scheme, the above system of three equations is converted to a system of six equations (disregarding the boundary conditions and initial state) by introducing additional variables corresponding to components of the displacement's velocity:

$$\mathbf{v_i} = \frac{\partial \mathbf{u_i}}{\partial t} \tag{14}$$

to obtain

$$\begin{cases} \partial_t \mathbf{u} = \mathbf{v} \\ \\ \partial_t \mathbf{v} = \rho^{-1}\left(\nabla \cdot \boldsymbol{\sigma} + \mathbf{F}\right) \end{cases} \tag{15}$$

The corresponding weak formulation discretized with the explicit Newmark's scheme (with $\beta = \gamma = 0$) is the following: find $u_i^{(t)}, v_i^{(t)} \in H^1(\Omega)$ for $t = 1, \ldots, T$, and $i = 1, 2, 3$ such that

$$\begin{cases} \left(u_i^{(t+1)}, w\right)_{L^2(\Omega)} = \left(u_i^{(t)} + \Delta t\, v_i^{(t)} + \dfrac{\Delta t^2}{2} a_i^{(t)}, w\right)_{L^2(\Omega)} \\ \\ \left(v_i^{(t+1)}, w\right)_{L^2(\Omega)} = \left(v_i^{(t)} + \Delta t\, a_i^{(t)}, w\right)_{L^2(\Omega)} \\ \\ a_i^{(t)} = \dfrac{1}{\rho}\left(\sigma_{ij,j} + F_i\right) \end{cases} \tag{16}$$

for all $w \in H^1(\Omega)$.

Using the Galerkin method, this problem can be restricted to a finite-dimensional subspace $V \subset H^1(\Omega)$ spanned by tensor product B-spline basis functions $\{\mathcal{B}_i\}$. Such a restricted problem is the same as above except that we require $u_i^{(t)}, v_i^{(t)} \in V$ and demand that (16) holds just for all $w \in V$.

Based on the structure of the problem, it can be seen that each computational step – finding $u_i^{(t+1)}$ and $v_i^{(t+1)}$ – for each equation in the above system can be seen as an instance of the following problem: given $f \in H^1(\Omega)$, find $a \in V$ such that $(a, w)_{L^2(\Omega)} = (f, w)_{L^2(\Omega)}$ for all $w \in V$.

We have

$$a = a_1\mathcal{B}_1 + \ldots a_k\mathcal{B}_N \tag{17}$$

where N is the number of basis functions (dimension of V), so by the linearity of the scalar product, the above problem is equivalent to the system of equations

$$\begin{cases} a_1 \left(\mathcal{B}_1, \mathcal{B}_1\right)_{L^2(\Omega)} + \cdots & + a_N \left(\mathcal{B}_N, \mathcal{B}_1\right)_{L^2(\Omega)} & = \left(f, \mathcal{B}_1\right)_{L^2(\Omega)} \\ a_1 \left(\mathcal{B}_1, \mathcal{B}_2\right)_{L^2(\Omega)} + \cdots & + a_N \left(\mathcal{B}_N, \mathcal{B}_2\right)_{L^2(\Omega)} & = \left(f, \mathcal{B}_2\right)_{L^2(\Omega)} \\ \vdots & \vdots & \vdots \\ a_1 \left(\mathcal{B}_1, \mathcal{B}_N\right)_{L^2(\Omega)} + \cdots & + a_N \left(\mathcal{B}_N, \mathcal{B}_N\right)_{L^2(\Omega)} & = \left(f, \mathcal{B}_N\right)_{L^2(\Omega)} \end{cases} \tag{18}$$

The matrix of this system is the Gram matrix of the B-spline basis. In other words, the computation of a single Newmark step is equivalent to solving the isogeometric L^2 projection problem. This can be done with linear computational cost using a sequential ADS algorithm.

4.3. Numerical results

In our test case, a force is briefly applied to an initially undeformed cube. The force is exerted by

$$\mathbf{F}(\mathbf{x}, t) = -\phi(t/t_0) r(\mathbf{x})\, \mathbf{p} \tag{19}$$

$$\mathbf{p} = (1, 1, 1) \tag{20}$$

$$t_0 = 0.02 \tag{21}$$

$$\phi(t) = \begin{cases} t^2(1-t)^2 & \text{if } t \in (0,1) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

$$r(\mathbf{x}) = 10 \exp\left(-10 \left\|\mathbf{x} - \mathbf{p}\right\|^2\right) \tag{23}$$

i.e., it is a short impulse directed towards the origin applied at the opposite corner of the cube.

Figure 1 displays results of the simulation with the time step $\Delta t = 10^{-4}$, mesh size $12 \times 12 \times 12$, and B-spline basis of order $p = 2$ (thus, $(12+2)^3 = 2,744$ degrees of freedom in total). Displacement is magnified 20 times to make it visible. As expected, the deformation originating at the corner where the force was applied propagates through the whole cube. During the simulation, 30,000 iterations were performed in less than 11 hours (1.46 seconds per iteration, on average) on a single machine with Intel Core i5-2410M CPU (4 cores, 2.3 GHz) and 4 GB of RAM.

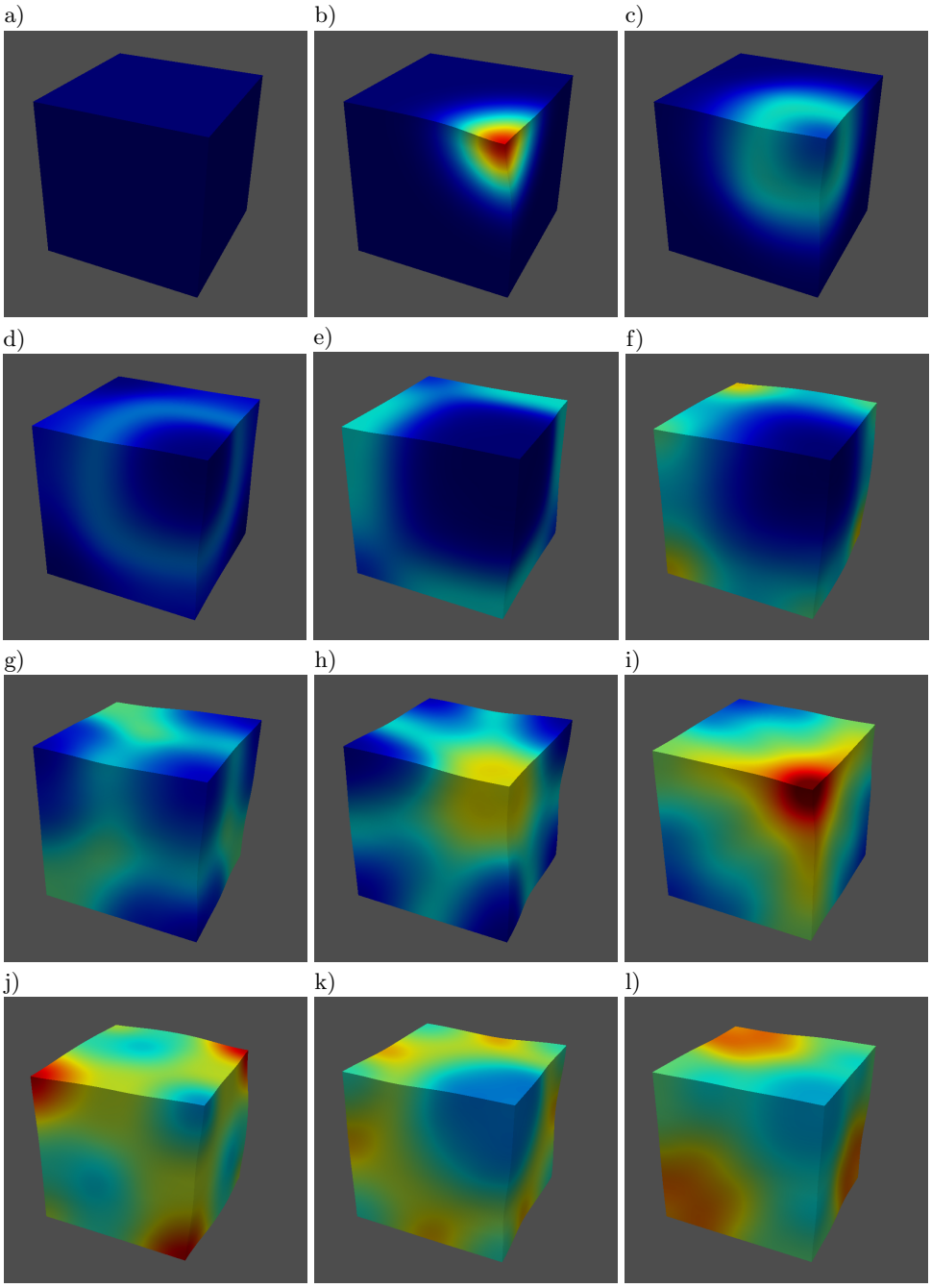

Figure 1. Deformation of cube during the simulation (color indicates the magnitude of deformation): a) initial state; b) step 1000; c) step 4000; d) step 6000; e) step 8000; f) step 10 000; g) step 12 000; h) step 14 000; i) step 16 000; j) step 18 000; k) step 20 000; l) step 22 000

5. Conclusion and future work

In this paper, we described an Alternating Direction algorithm, explored its complexity, and presented its application to a three-dimensional, non-stationary, linear elasticity problem using the isogeometric finite element method. We have shown its linear computational cost, which is superior with respect to the direct and iterative solvers applied to the explicit dynamics problems. The three main directions of development are parallelization of the method, extending the method for more-complex geometries, and implementing some adaptive algorithms. We are also currently investigating the feasibility of applying a variant of ADS to diffusion interface models with implicit time-stepping schemes (used in tumor modeling, for example) [11, 12].

Acknowledgements

This work has been supported by Polish National Science Center grant no. DEC-2014/15/N/ST6/04662.

Appendix A. Complexity of the ADI algorithm

Definition 1. Let $f, g\colon \mathbb{N} \to \mathbb{R}^+$ be functions. We write

- $f \in \mathcal{O}(g(n))$ if there exist $N \in \mathbb{N}$ and $M > 0$ such that for all $n \geq N$ we have $f(n) \leq g(n)$,
- $f \in \Omega(g(n))$ if $g \in \mathcal{O}(f(n))$,
- $f \in \Theta(g(n))$ if $f \in \mathcal{O}(g(n))$ and $f \in \Omega(g(n))$.

Let $\mathbf{M} = \mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_d$ where $\mathbf{A}_i$ are $n_i \times n_i$ matrices and $\mathbf{b}$ be $n_1 \cdots n_d$ vector, indexed with d indices. Thus, the total number of degrees of freedom is $N = n_1 n_2 \cdots n_d$. Let us denote by C_i the cost of solving a single linear system with matrix $\mathbf{A}_i$.

Theorem 1. *Assuming $C_i \in \Omega(n_i)$, the total cost of solving system $\mathbf{Ax} = \mathbf{b}$ using the ADS algorithm is of order*

$$\sum_{i=1}^{d} \left(\prod_{j \neq i} n_i \right) C_i \tag{24}$$

In particular, if $C_i \in \Theta(n_i)$, the cost is $\Theta(N)$.

Proof. The ADS algorithm comprises d steps, each consisting of applying the inverse of $\mathbf{A}_i$ to a set of vectors and transposing the right-hand side. At step k, there are

$$n_1 \cdots n_{k-1} n_{k+1} \cdots n_d = \prod_{i \neq k} n_i \tag{25}$$

systems of linear equations to be solved (systems are indexed with $j_1, \ldots, j_{k-1}, j_{k+1}, \ldots, j_d$ with j_k ranging from 1 to n_k); thus, the total cost of solving them is

$$n_1 \cdots n_{k-1} n_{k+1} \cdots n_d \, C_k = \left(\prod_{i \neq k} n_i \right) C_k \tag{26}$$

Summing these costs over all of the steps yields an estimate from the statement of the theorem. Furthermore, if $C_k \in \Omega(n_k)$ then

$$\left(\prod_{i \neq k} n_i\right) C_k \in \Omega\left(n_k \prod_{i \neq k} n_i\right) = \Omega\left(\prod_i n_i\right) = \Omega(N) \tag{27}$$

Size of the right-hand side vector is N. Transposition involves copying each of its elements exactly once, so the cost of transposition is $\mathcal{O}(N)$. Thus, the dominating factor is the cost of solving linear systems. Finally, assuming $C_k \in \Theta(n_k)$, we can replace Ω with Θ in the above, and we get

$$\left(\prod_{i \neq k} n_i\right) C_k \in \Theta(N) \tag{28}$$

Summing the costs of all of the steps, the total cost is thus $\Theta(dN) = \Theta(N)$. □

Since in the case of the L^2-projection problem with B-spline basis matrices $\mathbf{M}_i$ are banded with bandwidth p, we indeed have $C_i \in \Theta(n_i)$, and the above theorem implies that the ADS algorithm is linear with respect to the number of degrees of freedom. For a more-precise estimate, we need to consider the order of continuity p.

Corollary 1. *Using the LAPACK solver, the cost of solving system* $\mathbf{Mx} = \mathbf{b}$ *resulting from the* L^2*-projection problem is* $\Theta(pN)$ *under the following assumptions:*

(1) $d > 2$
(2) *all* n_i *grow at the same rate; i.e.,* $n_i = \Theta(n_j)$ *for all* i, j

Proof. The cost of LU factorization of $n \times n$ banded matrix with bandwidth p is $\Theta\left(p^2 n\right)$ [10, Algorithm 4.3.1]. With the given LU factorization, solving the system using forward and backward substitution can be done in $\Theta(pn)$ [10, Section 4.2.2]. Since the ADS algorithm requires solving multiple systems with the same matrix, LU factorization can be computed only once and used to efficiently solve all of the systems. For each i, there are $\prod_{j \neq i} n_j$ systems with matrix $\mathbf{M}_i$; hence, we have

$$C_i = \Theta\left(p\, n_i + \frac{p_i^2 n_i}{\prod_{j \neq i} n_j}\right) = \Theta\left(p\, n_i + p_i^2 n_i^2 / N\right) \tag{29}$$

Given the assumptions, we have

$$n_i^2 / N = \mathcal{O}\left(\frac{1}{n_i}\right) \tag{30}$$

and so, the second summand in the above estimate is negligible. Therefore,

$$C_i = \Theta(p\, n_i) \tag{31}$$

and by Theorem 1, the total cost is $\Theta(pN)$. □

References

[1] Anderson E., Bai Z., Dongarra J., Greenbaum A., McKenney A., Croz J.D., Hammerling S., Demmel J., Bischof C., Sorensen D.: LAPACK: A portable linear algebra library for high-performance computers. In: *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, IEEE Computer Society Press, pp. 2–11, Los Alamitos, CA, USA, 1990. http://dl.acm.org/citation.cfm?id=110382.110385.

[2] Birkhoff G., Varga R.S., Young D.: Alternating Direction Implicit Methods, *Advances in Computers*, vol. 3, pp. 189–273, 1962.

[3] Collier N., Pardo D., Dalcin L., Paszyński M., 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*, vol. 213–216, pp. 353–361, 2012.

[4] Cottrell J.A., Hughes T.J.R., Bazilevs Y.: *Isogeometric Analysis: Toward Integration of CAD and FEA*, Wiley Publishing, 1st ed., 2009.

[5] Dalcin L., Collier N., Vignal P., Cortes A.M.A., Calo V.M.: PetIGA: High-Performance Isogeometric Analysis. In: *arxiv*, (1305.4452), 2013. http://arxiv.org/abs/1305.4452.

[6] Douglas J., Rachford H.: On the numerical solution of heat conduction problems in two and three space variables, *Transactions of American Mathematical Society*, vol. 82, pp. 421–439, 1956.

[7] Duff I.S., Reid J.K.: The Multifrontal Solution of Indefinite Sparse Symmetric Linear, *ACM Transactions on Mathematical Software (TOMS)*, vol. 9(3), pp. 302–325, 1983.

[8] Gao L., Calo V.M.: Fast isogeometric solvers for explicit dynamics, *Computer Methods in Applied Mechanics and Engineering*, vol. 274, pp. 19–41, 2014.

[9] 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, pp. 274–295, 2015.

[10] Golub G.G., Loan C.F.V.: *Matrix Computations*, The Johns Hopkins University Press, 4th ed., 2013.

[11] Hawkins-Daarud A., Prudhomme S., Zee van der K.G., Oden J.T.: Bayesian calibration, validation, and uncertainty quantification of diffuse interface models of tumor growth, *Journal of Mathematical Biology*, vol. 67(6), pp. 1457–1485, 2013. http://dx.doi.org/10.1007/s00285-012-0595-9.

[12] Hawkins-Daarud A., Zee van der K.G., Tinsley Oden J.: Numerical simulation of a thermodynamically consistent four-species tumor growth model, *International Journal for Numerical Methods in Biomedical Engineering*, vol. 28(1), pp. 3–24, 2012, http://dx.doi.org/10.1002/cnm.1467.

[13] Hughes T.J.R., Cottrell J.A., Bazilevs Y.: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Computer Methods in Applied Mechanics and Engineering*, vol. 194(39–41), pp. 4135–4195, 2005.

[14] Łoś M., Woźniak M., Paszyński M., Dalcin L., Calo V.M.: *Parallel alternating direction preconditioner for isogeometric simulations of explicit dynamics*, 1st Pan-American Congress on Computational Mechanics, Buenos Aires, April 27–29.

[15] Marsh D.: *Applied Geometry for Computer Graphics and CAD*, Springer Undergraduate Mathematics Series, Sprlinger-Verlag London, 2005.

[16] Peaceman D., Rachford H.: The numerical solution of parabolic and elliptic differential equations, *Journal of Society of Industrial and Applied Mathematics*, vol. 3(1), pp. 28–41, 1955.

[17] Piegl L., Tiller W.: *The NURBS Book*, 2nd ed., Springer-Verlag, New York, 1997.

[18] Saad Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[19] Wachspress E., Habetler G.: An alternating-direction-implicit iteration technique, *Journal of Society of Industrial and Applied Mathematics*, vol. 8(2), pp. 403–424, 1960.

[20] Woźniak M., Łoś M., Paszyński M., Dalcin L., Calo V.M.: Dynamics with matrices possesing Kronecker product structure, *Procedia Computer Science*, vol. 51, pp. 286–295, 2015.

[21] Woźniak M., Łoś M., Paszyński M., Dalcin L., Calo V.M.: Parallel fast isogeometric solvers for explicit dynamic, *Computing and Informatics*, vol. 32, pp. 1001–1026, 2015.

Affiliations

Marcin Łoś
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, Krakow, Poland,
los@student.agh.edu.pl

Maciej Paszyński
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, Krakow, Poland, paszynsk@agh.edu.pl

Received: 31.01.2016
Revised: 13.10.2016
Accepted: 17.10.2016