Automatic Algorithms for the Construction of Element Partition Trees for Isogeometric Finite Element Method

Bartosz Janota[1] and Anna Paszyńska[2(✉)]

[1] Department of Computer Science, AGH University of Science and Technology, Kraków, Poland
[2] The Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Kraków, Poland
anna.paszynska@uj.edu.pl

Abstract. The IGA-FEM is a modern method for simulation of different engineering and biomedical problems by using B-spline basis functions. To allow for real time interaction between human and computer while performing numerical simulations, there is a need for extremely fast solvers solving systems of linear equations generated while performing simulations. In this paper, an algorithm for construction of the ordering that controls the execution of the multi-frontal direct solver algorithm is presented. The ordering prescribes the permutation of the computational matrix in order to minimize the computational cost of the direct solver. We show that execution of our algorithm generating the recursive partitions of the h-adaptive grids with B-spline basis functions allows reducing the computational cost of the IGA-FEM simulations.

Keywords: Simulations on biomedicine · Graph algorithms · Topology of computational meshes · Isogeometric finite element method

1 Introduction

The Computer Aided Design systems (CAD systems) are often used to design machine parts, civil engineering structures, cars, airplanes and many other engineering artifacts. These systems use the B-splines and Non-Uniform Rational B-splines (NURBS) as the basis functions modelling the geometry. The objects created within CAD system are later subject to finite element method simulations, computing stress, deformation, heat transfer, accoustic and other physical phenomena. However, simulations performed by means of Computer Aided Engineering systems (CAE) use Lagrange polynomials to approximate the solid object geometry. The differences in the geometrical representation require expensive transformation from the CAD model (with B-splines and NURBS) into a finite element method model (with Lagrange polynomials), which is computationally expensive and time-consuming.A newly developed computational approach integrating the finite element analysis into conventional NURBS-based

A. Gruca et al. (eds.), *Man-Machine Interactions 5*,
Advances in Intelligent Systems and Computing 659, DOI 10.1007/978-3-319-67792-7_28

CAD design tools is isogeometric finite element method (IGA-FEM), introduced by Tom Hughes and his group [10]. The IGA-FEM uses the same B-spline and NURBS basis functions for both design and simulations.

The IGA-FEM has several important applications in engineering simulations, starting from biomedical applications, including blood flow and drug delivery [5,6,8,19], phase field and phase separation simulations including the cancer simulations [11,12,17,18], through shell theory [7], wind turbine aerodynamics [20], incompressible hyper-elasticity [15], to turbulent flow simulations [9]. The Computer Aided Design and Engineering Systems by their basics principles involve real-time interaction between human and computer. So far it is possible during the designing process. Thus, there is a need for extremely fast solvers that will allow for real time interaction between human and computer also while performing numerical simulations. The goal of this paper is to propose a methodology to speed up this solvers. In this paper, we focus on the construction of element partition trees and the corresponding orderings that controls the execution of the multi-frontal direct solver algorithm for the case of solving systems of linear equations generated while performing simulations using isogeometric finite element method. The main goal of this paper is the construction of new orderings in order to optimize the performance of the multi-frontal direct solver algorithms performing IGA-FEM computations. The IGA-FEM with B-splines and NURBS works on regular patches of elements (all element have the same size), but recently it has been shown [22] that computations on h-adaptive grids are also possible with B-splines and NURBS. This is done by introducing of the so-called C^0 separators between layers of h-adaptive elements. Additionally, mixing of B-splines with C^0 separators allows to reduce the computational cost of direct solver on uniform grids [16]. In this paper we investigate further this concept on h-adaptive grids.

The element partition trees were proposed in [1] to speed up the performance of multi-frontal solvers for h-adaptive finite element method with hierarchical Lagrange polynomials [30]. Two algorithms for the construction of the element partition trees for h-adaptive grids with Lagrange polynomials were proposed. The first one was a greedy algorithm based on bottom-up approach [28,29]. The second one was based on the top-down approach, called *element size weighted bisection algorithm*, following the idea of graph partition. It is described in [1] for two and three-dimensional h-adaptive grids with Lagrange polynomials. It utilizes the graph partition algorithm on weighted graphs of elements, with weights related to elements, corresponding to the depth of the element within the h-refinement tree.

The element partition trees have been also used in [22] to process two-dimensional grids with with B-splines and NURBS for the case of point and edge singularities. These trees were constructed "by hand" for the grids with point and edge singularities. In [22] there is no algorithm constructing these trees automatically. Our paper, differs from [22] by the following two facts. First, we propose an algorithm for automatic construction of the element partition trees

for IGA-FEM. Second, we extend the algorithm and the methodology for three-dimensional grids.

2 Ordering Based on Sparsity Graph of the Matrix

The first step of FEM computation is a representation of the geometry of the computational domain by means of a mesh of so-called finite elements. In the next step, the partial differential equations defined over the computational domain, together with the prescribed boundary conditions, are transformed into so-called weak form and discretized using a basis of polynomial functions spread over defined finite elements nodes. The basis functions have different supports, which spread over the adjacent finite elements. When we solve an engineering problem using the finite element method, the global system of linear equations is formulated. The rows and columns in the system correspond to different basis functions spread over the computational mesh. The non-zero entries in the matrix happen only when the corresponding basis functions from rows and columns overlap. The particular formulas for the basis functions depend on the kind of finite element method used. For example we can focus on Lagrange polynomials [13,14] presented in top left panel in Fig. 1, or we can focus on B-splines basis functions [10] or their modifications called NURBS [21], presented in right top panel in Fig. 1. For the case of Lagrange polynomials, like in the left panel in Fig. 1, we define first order basis functions at vertices, and higher order basis functions at edges and interiors. In the case of IGA-FEM the basis functions are obtained through tensor products of one-dimensional B-splines, defined through the so-called knot vectors. Let us consider a two-dimensional patch with 2×2 elements, with quadratic B-splines, like the one presented on the right panel in Fig. 1. Such the patch can be obtained by introducing a knot-vector [0 0 0 1 2 2 2 2] to any software compatible with IGA technology. We introduce such knot-vector for directions x, and y. The repetition of the first and the last knot-points three times implies that quadratic B-splines are uniformly distributed over the mesh in directions x and y. The introduced B-splines have C^1 continuity along the entire mesh. For the case of Lagrange polynomials, like in the left panel in Fig. 1, we define first order basis functions at vertices, and higher order basis functions at edges and interiors. The non-zero entries in the matrix correspond to integrals of multiplications of the basis functions and their derivatives. The sparsity patterns for the global matrices for the Lagrange and B-spline polynomials case are presented in top panels in Fig. 1. The IGA-FEM matrices are smaller but denser than FEM matrices.

The particular formula for the integral depends on the engineering problem being solved, but it does not depend on the sparsity pattern of the matrix. Thus, to speed up the factorization of the matrix by means of the direct solver algorithm, we only need to focus on finding a proper permutation of the global matrix, which results in lower number of non-zero entries generated during the factorization (the so-called fill-in). The quality of such the ordering influences hardly the computational cost of the solver algorithm. There are not known algorithms, constructing optimal ordering for arbitrary finite element mesh.

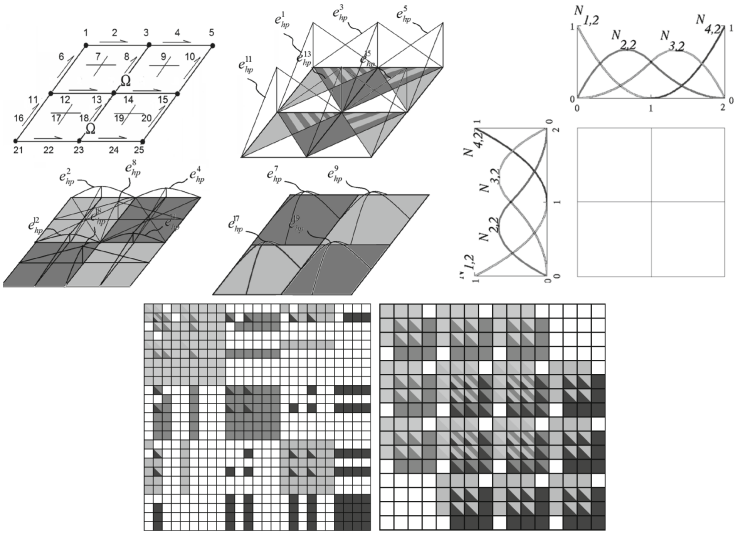

Fig. 1. Four finite element mesh. Top left panels: lagrange polynomials. Top right panel: B-splines polynomials. Bottom left panel: global matrix for the four finite element mesh with Lagrange polynomials. Bottom right panel: global matrix for the four finite element mesh with B-splines polynomials.

The first step of the traditional way of finding the proper permutation of the global matrix is building a sparsity graph of the global matrix. In the graph, the vertices correspond to rows and columns in the matrix and edges corresponds to non-zero entries in the matrix. Let us focus now on Lagrange polynomials. The exemplary mesh and the corresponding sparsity graph are presented in Fig. 2. In order to find the elimination tree and the corresponding ordering, some algorithms, for example, the nested-dissections [23,26] algorithm, are executed

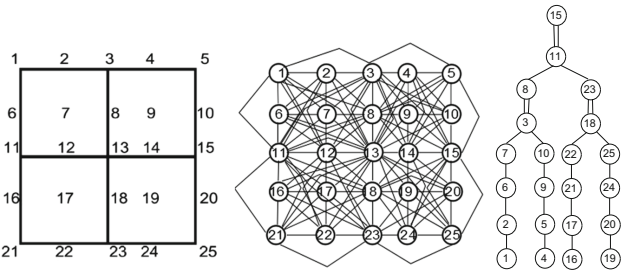

Fig. 2. Left panel: four finite elements mesh with first order Lagrange polynomials at vertices, and second order Lagrange polynomials at edges and interiors. Middle panel: sparsity pattern that corresponds to the mesh from left panel. Right panel: elimination tree for the mesh from left panel.

on the sparsity graph. Right panel from Fig. 2 presents a so-called elimination tree, found by nested dissection algorithm for the mesh from Fig. 2

The ordering for the permutation of the global matrix is obtained by browsing the elimination tree in BFS order. Alternatively, the elimination tree can be processed level by level, from leaves up to the root, and the rows of the matrix can be eliminated in the obtained order. The algorithm is the core of most multi-frontal direct solvers, such as MUMPS [2–4], PaSTiX [27] or SuperLU [24,25].

2.1 Ordering Based on Element Partition Tree

An alternative way of constructing the ordering for the permutation of the global matrix is the following. We construct the element partition tree in a similar way like the elimination tree, but this time working on the graph of the computational mesh. The vertices in our graph correspond now to the finite element, and edges correspond to the adjacency relation between the elements. We have moved the analysis from the level of rows and columns in the sparse matrix to the level of elements in the mesh and adjacency relation between the elements. The exemplary element partition tree obtained for our four finite element mesh is presented in Fig. 3. The leaves in the tree correspond to finite elements, and the upper nodes inside the tree correspond to parts of the mesh (lists of finite elements). The root represents the entire mesh merged together. The element partition tree can be transformed into the elimination tree in the following way. We can browse the element partition tree and identify the fully assembled nodes of elements assigned to given node of the tree. These mesh nodes can be eliminated at given level of the tree. For example,

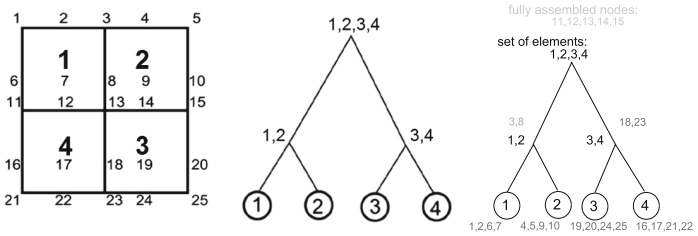

Fig. 3. Left panel: exemplary finite element mesh. Middle panel: element partition tree for the four finite element mesh. Right panel: transforming the element partition tree into the elimination tree.

- Nodes 1, 2, 6, 7 belong only to element 1, nodes 4, 5, 9, 10 belong only to element 2, nodes 19, 20, 24, 25 belong only to element 3, nodes 16, 17, 21, 22 belong only to element 4, so they can be eliminated at the leaf nodes.
- Nodes 3, 8 belong only to Element 1 and 2, so they can be eliminated at the node {1, 2}.

- Nodes 18, 23 belongs only to Element 3 and 4 so they can be eliminated at the node {3, 4}.
- Nodes 11, 12 belong to element 1 and 4, node 13 belongs to element 1, 2, 3, 4, nodes 14, 15 belong to element 2 and 3. At the root node we have all the elements. So {1, 4} is the subset of {1, 2, 3, 4}, and {2, 3} is the subset of {1, 2, 3, 4}. So they can be eliminated at the root node.

The element partition tree has been proposed in [1] to speedup the computations over the h-adaptive grids with Lagrange polynomials. It has been shown that element partition trees result in elimination trees that provide better performance of the multi-frontal direct solvers when compared to standard nested-dissections algorithms executed on the sparsity graph of the global matrix. In this paper, we focus on h-adaptive grids utilized in isogeometric finite element method.

2.2 Automatic Algorithms for Construction of the Element Partition Tree

In this section we propose a new algorithm called *IGA-FEM weighted bisections* to process efficiently the h-adaptive grids with B-spline basis functions and C^0 separators. *IGA-FEM weighted bisection*

```
function IGA_FEM_Weighted_Bisection(G)
(1) If number of nodes in G equals to 1 (node v) then
(2)  create one element tree t with the node v; return t;
(3) Calculate balanced edge and node weighted partition
        of a graph G into G1 and G2;
(4) t1 = IGA_FEM_WeightedBisection(G1)
(5) t2 = IGA_FEM_WeightedBisection(G2)
(6) create new root node t with left child t1, right child t2
(7) return t
```

The algorithm uses a graph where vertices correspond to elements, and edges correspond to the adjacency between the element. The algorithm uses the balanced weighted partition of a graph into two subgraphs. The elements are weighted by the size of elements, defined in a similar way like in [1]. The edges are weighted in the following way: the edge between two nodes has weight 1 if the edge joints nodes corresponding two elements between which is a C^0 separator and p (the polynomial order od used B-splines) in the other case. The partition algorithm performs balanced partition of a graph in two subgraphs with the same sum of weights of nodes, in such a way, that minimizes the sum of weights of edges cut during partition. Figure 4 presents the initial mesh (left panel) and the partitions of the input graph for the case of $p = 2$ (right panel). The width of the partition line denotes the order of partitions. Figure 5 presents the element partition tree for the partitions of the input graph from the right panel of Fig. 4.

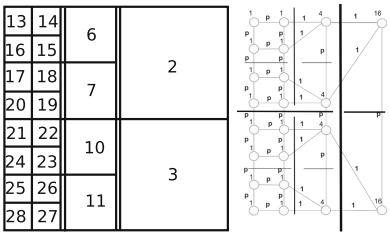

Fig. 4. The initial mesh (left panel) and the partitions of the input graph for the case of $p = 2$ (right panel).

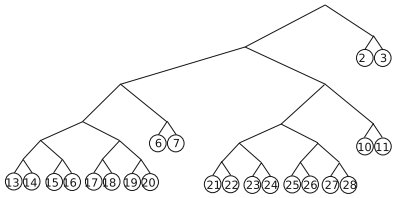

Fig. 5. The element partition tree for the partitions of the input graph for the case of $p = 2$ from the right panel of Fig. 4.

3 Numerical Results for *h*-Adaptive Grids for Isogeometric Finite Element Method

We test our algorithm on h-adaptive three-dimensional girds. As proposed in [22], the h-adaptive B-splines can be defined over h-adaptive grids by the introduction of the C^0 separators between layers of elements. The C^0 separators can be seen as repeating knots in knot vectors defining patches of elements on particular refinement level, and gluing the patches together, as it is illustrated in Fig. 6. We measured the computational cost of the multi-frontal direct solvers executed over three-dimensional grid refined towards edge and face singularity. For the edge and face singularity, the utilization of B-splines with C^0 separators wins with classical Lagrange polynomials. Both solvers deliver linear cost, but for the B-splines with C^0 separators the constant is lower. In our experiments, we use the ordering obtained from our element partition trees for the case of the B-splines with C^0 separators, and the ordering obtained from nested-dissections implemented in METIS library. We submit our ordering to MUMPS solver and report the resulting FLOPS number.

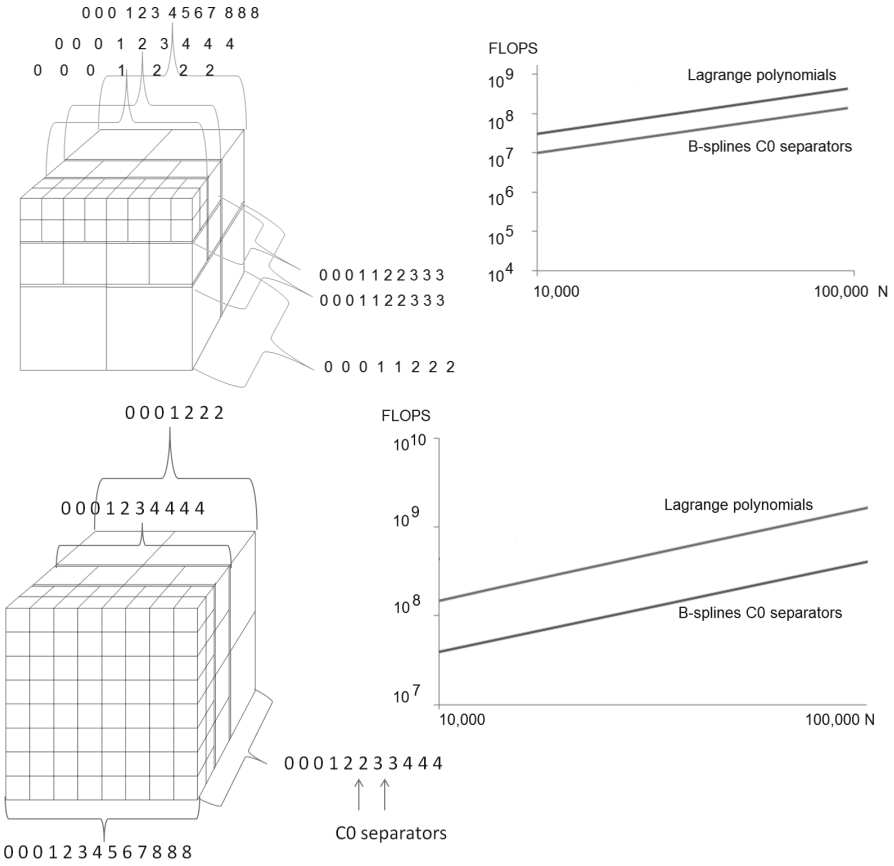

Fig. 6. Scalability of the multi-frontal solver with B-splines and C^0 separators versus the solver with Lagrange polynomials over three-dimensional mesh with edge and face singularities.

4 Conclusions

In this paper we proposed an algorithm for generation of the ordering over h-refined grids with B-splines and C^0 separators. The ordering enables for the efficient solution of systems of linear equations obtained from isogeometric finite element method computations. The resulting performance of the multi-frontal direct solver outperforms the one obtained with hierarchical Lagrange polynomials and the state-of-the-art ordering obtained from METIS.

References

1. Aboueisha, H., Calo, V.M., Jopek, K., Moshkov, M., Paszyńka, A., Paszyński, M., Skotniczny, M.: Element partition trees for h-refined meshes to optimize direct solver performance. Part I. Dynamic Programming. Int. J. Appl. Math. Comput. Sci. **27**(2), 351–365 (2017)
2. Amestoy, P., Guermouche, A., L'Excellent, J.Y., Pralet, S.: Hybrid scheduling for the parallel solution of linear systems. Parallel Comput. **32**(2), 136–156 (2006)
3. Amestoy, P., Duff, I., L'Excellent, J.Y.: Multifrontal parallel distributed symmetric and unsymmetric solvers. Comput. Methods Appl. Mech. Eng. **184**, 501–520 (1998)
4. Amestoy, P., Duff, I., L'Excellent, J.Y., Koster, J.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM J. Matrix Anal. Appl. **23**(1), 15–41 (2001)
5. Bazilevs, Y., Calo, V., Cottrell, J., Hughes, T., Reali, A., Scovazzi, G.: Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Comput. Methods Appl. Mech. Eng. **197**(1), 173–201 (2007)
6. Bazilevs, Y., Calo, V.M., Zhang, Y., Hughes, T.J.R.: Isogeometric fluid–structure interaction analysis with applications to arterial blood flow. Comput. Mech. **38**(4–5), 310–322 (2006)
7. Benson, D., Bazilevs, Y., Hsu, M.C., Hughes, T.: A large deformation, rotation-free, isogeometric shell. Comput. Methods Appl. Mech. Eng. **200**(13), 1367–1378 (2011)
8. Calo, V.M., Brasher, N.F., Bazilevs, Y., Hughes, T.J.R.: Multiphysics model for blood flow and drug transport with application to patient-specific coronary artery flow. Comput. Mech. **43**(1), 161–177 (2008)
9. Chang, K., Hughes, T., Calo, V.: Isogeometric variational multiscale large-eddy simulation of fully-developed turbulent flow over a wavy wall. Comput. Fluids **68**, 94–104 (2012)
10. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric Analysis: Toward Unification of CAD and FEA. Wiley, Chichester (2009)
11. Dedè, L., Borden, M.J., Hughes, T.: Isogeometric analysis for topology optimization with a phase field model. Technical report, The Institute for Computational Engineering and Sciences (2011)
12. Dedè, L., Hughes, T., Lipton, S., Calo, V.: Structural topology optimization with isogeometric analysis in a phase field approach. In: USNCTAM 2010, State College, PA, USA (2010)
13. Demkowicz, L.: Computing with *hp*-Adaptive Finite Elements, vol. I. One and Two Dimensional Elliptic and Maxwell Problems. Chapman and Hall/CRC, Boca Raton (2006)
14. Demkowicz, L., Kurtz, J., Pardo, D., Paszyński, M., Rachowicz, W., Zdunek, A.: Computing with hp-Adaptive Finite Elements, Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications. Chapman & Hall/CRC, Boca Raton (2007)
15. Duddu, R., Lavier, L., Hughes, T., Calo, V.: A finite strain Eulerian formulation for compressible and nearly incompressible hyper-elasticity using high-order NURBS elements. Int. J. Numer. Methods Eng. **89**(6), 762–785 (2011)
16. Garcia, D., Pardo, D., Dalcin, L., Paszyński, M., Collier, N., Calo, V.: The value of continuity: refined isogeometric analysis and fast direct solvers. Comput. Methods Appl. Mech. Eng. **316**, 586–605 (2017)

17. Gómez, H., Calo, V., Bazilevs, Y., Hughes, T.: Isogeometric analysis of the cahn-hilliard phase-field model. Comput. Methods Appl. Mech. Eng. **197**(49–50), 4333–4352 (2008)
18. Gomez, H., Hughes, T., Nogueira, X., Calo, V.: Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. Comput. Methods Appl. Mech. Eng. **199**(25–28), 1828–1840 (2010)
19. Hossain, S., Hossainy, S.A., Bazilevs, Y., Calo, V., Hughes, T.R.: Mathematical modeling of coupled drug and drug-encapsulated nanoparticle transport in patient-specific coronary artery walls. Comput. Mech. **49**(2), 213–242 (2012)
20. Hsu, M., Akkerman, I., Bazilevs, Y.: High-performance computing of wind turbine aerodynamics using isogeometric analysis. Comput. Fluids **49**(1), 93–100 (2011)
21. Hughes, T., Cottrell, J., Bazilevs, Y.: Isogeometric analysis: cad, finite elements, nurbs, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. **194**(39–41), 4135–4195 (2005)
22. Janota, B.: Algorithms for construction of elimination tree for multi-frontal solver of isogeometric finite element method. Master's thesis, AGH University, Poland (2016)
23. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1), 359–392 (1998)
24. Li, X., Demmel, J., Gilbert, J., Grigori, I., Shao, M., Yamazaki, I.: Superlu users' guide. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory (1999)
25. Li, X.S.: An overview of SuperLU: algorithms, implementation, and user interface. ACM Trans. Math. Softw. **31**(3), 302–325 (2005)
26. Liu, J.: The role of elimination trees in sparse factorization. SIAM J. Matrix Anal. Appl. **11**(1), 134–172 (1990)
27. Hénon, P., Ramet, P., Roman, J.: PaStiX: a high-performance parallel direct solver for sparse symmetric positive definite systems. Parallel Comput. **28**(2), 301–321 (2002)
28. Paszyńska, A.: Volume and neighbors algorithm for finding elimination trees for three dimensional-adaptive grids. Comput. Math. Appl. **68**(10), 1467–1478 (2014)
29. Paszyńska, A., Paszyński, M., Jopek, K., Woźniak, M., Goik, D., Gurgul, P., AbouEisha, H., Moshkov, M., Calo, H., Lenharth, A., Nguyen, D., Pingali, K.: Quasi-optimal elimination trees for 2D grids with singularities. Sci. Programm. **2015**, 1–18 (2015)
30. Paszyński, M.: Fast Solvers for Mesh-Based Computations. CRC Press, Boca Raton (2016)