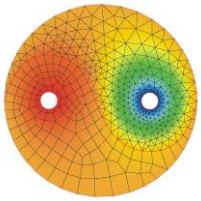
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE
Informatyka w Technologii Materiałów



Vol. 18, 2018, No. 1

USING THE HYPERGRAPH GRAMMAR FOR GENERATION OF QUASI OPTIMAL ELEMENT PARTITION TREES IN TWO DIMENSIONS

JAKUB RYZNER[1], MACIEJ PASZYŃSKI[2], ANNA PASZYŃSKA*[3]

[1]*Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland*

[2]*Department of Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland*

[3]*Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, ul. St. Łojasiewicza 11, 30-348 Krakow, Poland*

**Corresponding author: e-mail: anna.paszynska@uj.edu.pl*

Abstract

The paper presents the graph grammar model of Finite Element Method allowing for speeding up performed numerical simulations. In the presented approach, the finite element mesh operations are performed together with operations generating so-called element partition tree. The element partition tree sets the ordering of matrix operations performed by solver in order to solve the computational problem. The quality of element partition tree influences the computational time of the solver. Our method allows for generation good quality element partition trees for h-adaptive Finite Element Method. The paper is concluded with numerical results confirming the quality of generated element partition trees.

Key words: finite element method, multi-frontal solver, hypergraph grammar

1. INTRODUCTION

One of the method of performing numerical simulations is Finite Element Method (FEM) (Demkowicz 2006, Hughes 2000). This approach allows for finding the approximation of the solution as the linear combination of basis functions spread over finite element mesh.

The adaptive version of FEM allows for automatic control of the accuracy of the numerical solution over a sequence of computational grids by increasing the number of the basis functions used. There are several versions of adaptive-FEM algorithms (Babuška & Rheinboldt 1978; Babuška et al., 1981; Babuška, 1986; Demkowicz et al., 2002) with different areas of applications (Banaś et al., 2014; Demkowicz et al., 2006; Niemi et al., 2012).

The coefficients of the linear combination can be found by solving the corresponding system of linear equations. To solve the system of linear equations the multi-frontal solver can be used (Duff & Reid, 1983; Duff & Reid, 1984). The computational cost of solver application solving the system of equations depends hardly on the so-called ordering - the way in which elimination of rows of a matrix should be performed. The ordering for Finite Element computations can be prescribed by means of so-called element partition tree. The problem of finding optimal element partition tree as well as optimal ordering for an arbitrary mesh is NP-complete (Yanakakis, 1981). In the pa-

ISSN 1641-8581

per, the hypergraph grammar for simultaneous performing of mesh operations and generation of good quality corresponding element partition tree for 2-dimmensional meshes with triangular elements is presented.

We utilize the hypergraph grammar model originally proposed in (Ślusarczyk & Paszyńska, 2012) for rectangular grids. In this paper, we extend the model for more practical triangular elements.

Our hypergraph grammar model is used for simultaneous mesh generation and adaptation, and the generation of the element partition trees. The element partition trees were originally proposed in (Paszyński, 2016). In this paper, we present the hypergraph grammar for automatic generation of the trees. We verify our approach by numerical experiments performed on representative grids, including the triangular elements mesh refined to a point and triangular elements mesh refined to an edge.

The paper is organized as follows. In the recent section, the short introduction into Finite Element Method computation is presented. The next two chapters introduce the hypergraph grammar, Finite Element Method and the element partition trees. In the following chapter the hypergraph productions for mesh generation and element partition tree generation are presented. The paper ends with sections including numerical simulation and summary section.

2. HYPERGRAPH GRAMMAR

Hypergraphs are graphs which consist of nodes and so-called hyperedges. A hyperedge is an edge with sequences of nodes assigned to it. The nodes and hyperedges are labeled with the use of a fixed alphabet. Additionally, to nodes and hyperedges the sets of attributes are assigned. Figure 1a presents an exemplary hypergraph with three nodes labeled by *v*, three hyperedges labeled by *B* and one hyperedge labeled by *I*.

Hypegraphs can be created from simpler hypergraphs by replacing their subgraphs by new graphs. This operation is possible if for each hypergraph a sequence of its external nodes is specified. These nodes correspond to target nodes of a replaced subgraphs.

Hypergraphs are generated by applying productions, which specify the way of subhypergraph replacement. A hypergraph production is a pair $p = (L, R)$, where both L and R are hypergraphs with the same number of external nodes. In Figure 1b an exemplary production is presented, where hypergraphs L and R are hypergraphs with 2 external nodes. The application of a production $p = (L, R)$ to a hypergraph H consist of replacing a subhypergraph of H isomorphic with L by a hypergraph R and replacing nodes of the removed subhypergraph isomorphic with external nodes of L by the corresponding external nodes of R. Figure 1c presents the result of the application of the productions from figure 1b to the hypergraph from figure 1a.

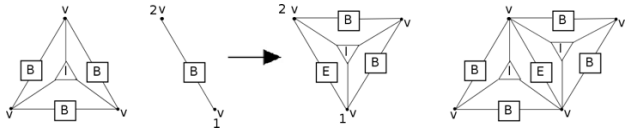


***Fig. 1**. (a) An exemplary hypergraph; (b) An exemplary hypergraph production; (c) Graph from figure 1a after applying production from figure 1b.*

3. FINITE ELEMENT METHOD FOR TRIANGULAR MESH AND ELEMENT PARTITION TREE

In the paper, the method of simultaneous performing of mesh operations and generation of element partition tree is presented. The Finite Element mesh consists of triangular elements. Each triangular element consists of three vertex nodes, one interior node and three edge nodes (see figure 2). The solution is approximated by means of the linear combination of basis functions spread over finite element nodes: edges, interior nodes and vertex nodes. In order to obtain more accurate solution, the number of used basis functions corresponding to finite element nodes can be increased. In the paper the automatic h-adaptations is performed - breaking elements into smaller elements in the areas where more accurate solution is needed. Figure 3 presents the triangular element from figure 2 after performing h-adaptation.

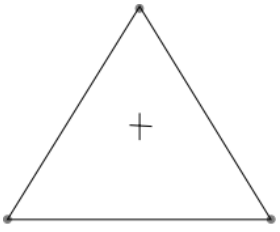


***Fig. 2.** Triangular element*

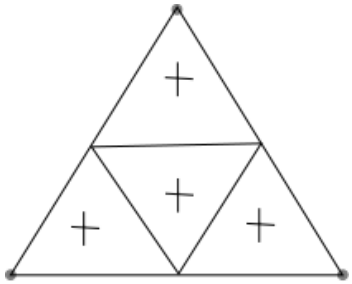


Fig. 3. *Triangular element after h-adaptation*

In the presented approach, the so called 1-irregularity rule is used to ensure the continuity of the finite element solution: a finite element can be broken only once without breaking the large adjacent element. Figure 4 explains the 1-irregularity rule.

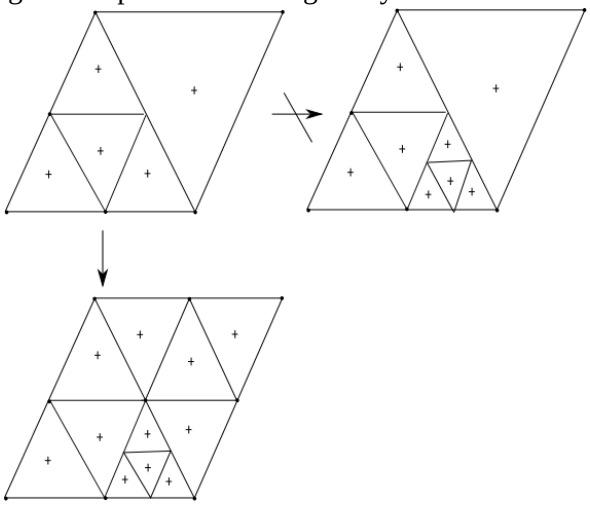


Fig. 4. *1-irregularity rule: enforcing of breaking big adjacent element before breaking the smaller one.*

To find the approximation of the solution as the linear combination of basis functions spread over finite element nodes, the system of linear equations should be solved. In the finite element method computations, the ordering is used to permute the system of linear equations into the equivalent form, for which the cost of a solution is lower than for the original system. In the paper, the algorithm of generation of element partition tree, which can be transformed into ordering, is presented. The element partition tree is defined as a binary tree with leaves associated with finite elements. The internal nodes of element partition tree denote sets of elements (sum of elements associated with their son nodes), and the root contains the list of all mesh elements. An exemplary four element triangular mesh and the corresponding element partition tree is presented in figure 5.

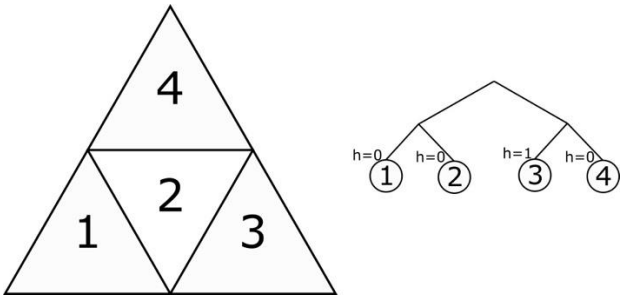


Fig. 5. *(a) An exemplary four element mesh; (b) an element partition tree of the mesh from figure 5a.*

4. HYPERGRAPH PRODUCTIONS FOR THE MESH GENERATION

In the paper, a mesh is represented by means of hypergraph and mesh transformations are represented as graph grammar productions. Additionally, productions allowing for element partition tree generation are defined.

The hypergraph represents the finite element mesh in the following way: graph nodes with label *v* represent vertices of the mesh, hyperedges labeled by *I* represent the element interiors, hyperedges labeled by *B* represent the boundary edges, hyperedges labeled by *F* represent the edges of elements. Each hyperedge denoting interior of an element has attribute *nr* which corresponds to a unique number of the corresponding finite element. Additionally the variable *a_nr* is defined as the counter of existing elements. It is increased each time when a new element is added to the mesh. After adding the first element, *a_nr* is set to 0.

An exemplary initial mesh consisting of one element, (numbered by 0), the corresponding hypergraph, and an element partition tree consisting of one node, are presented in figure 6.

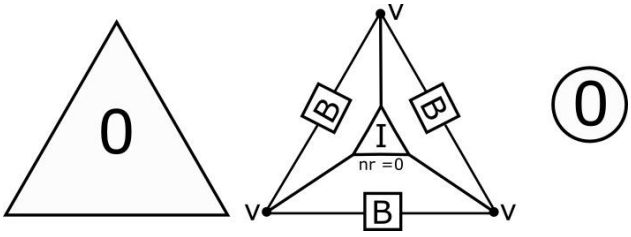


Fig. 6*. (a) An initial mesh; (b) A hypergraph representing an initial one element mesh; (c) an element partition tree of an initial mesh.*

Finite element method computations can be described as:

1. mesh generation,

2. calculation of solution and error of each element (for example defined as determinant of the jacobian of the solution over the element),
3. performing virtual h-adaptation:
   a. making a decision about the h-adaptation: each element, for which error is bigger than 33% of maximal error should be adapted, we will set attribute *h* to 1 for elements which should be adapted, 0 in the other case,
   b. performing propagation of the decision about refinements: enforcing of refinement of big elements adjacent to small elements which should be h-refined. Each element with *h=0*, adjacent to the smaller element which has attribute *h=1* get attribute *h* equal to *1*.
4. performing h-adaptation for elements with attribute *h* equal to 1
5. go to point 2 if the accuracy is not good enough.

In the paper, the productions for mesh generation and mesh adaptation together with the corresponding productions for generation of element partition tree are presented.

Remaining productions, corresponding to calculation of solution and error of each element as well as virtual h-adaptation and enforcing 1-irregularuty rule, are similar to productions defined in (Ślusarczyk & Paszyńska, 2012) for the case of the mesh consisting of rectangular elements.

The hypergraph production for adding a new element to the mesh is illustrated in figure 7. It generates one new element interior (labeled *I*), two new boundary edges (labeled *B*), one new edge (labeled *F*), and three new nodes (labeled *v*).

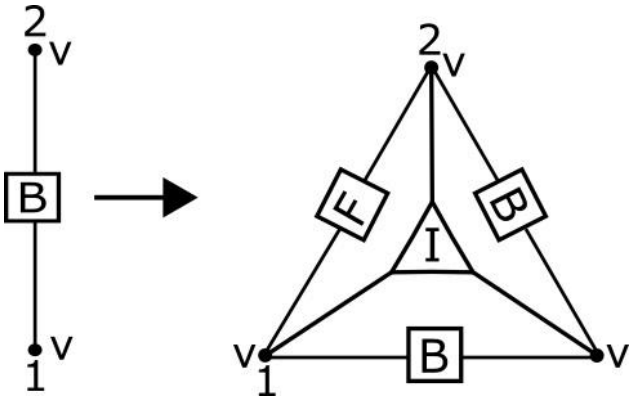


Fig. 7. *A hypergraph production for adding a new element to the mesh*

The production adding a new element adjacent to the two existing edges is shown in figure 8. It generates one new element interior (labeled *I*), one new boundary edge (labeled *B*), two new edges (labeled *F*), and three new nodes (labeled *v*).

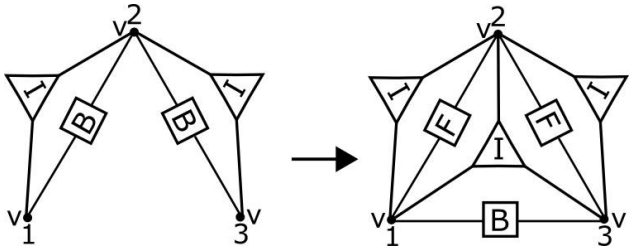


Fig. 8. *A hypergraph production for adding a new element adjacent to the two existing edges.*

The hypergraph production for breaking the interior of an element is illustrated in figure 9. It generates four new element interiors (labeled by *I*) and three new nodes (labeled by *w*).

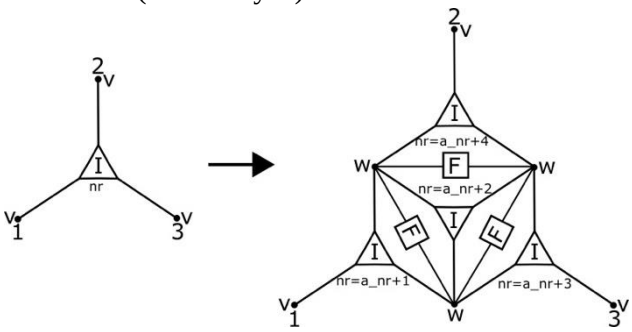


Fig. 9. *A hypergraph production for breaking the interior of an element.*

The corresponding production, which adds new nodes to the element partition tree, presented in figure 10, will be executed after adding new elements to the mesh.

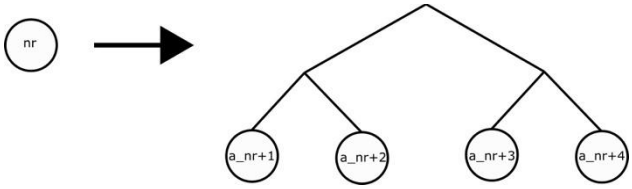


Fig. 10*. A production adding new nodes to the element partition tree.*

After breaking the interiors of the elements, boundary edges as well as shared edges surrounding by two broken interiors should be broken. To break a boundary edge means to generate two new boundary edges (labeled by *B*) and one new node (labeled by *w*). This is done by production presented in figure 11.

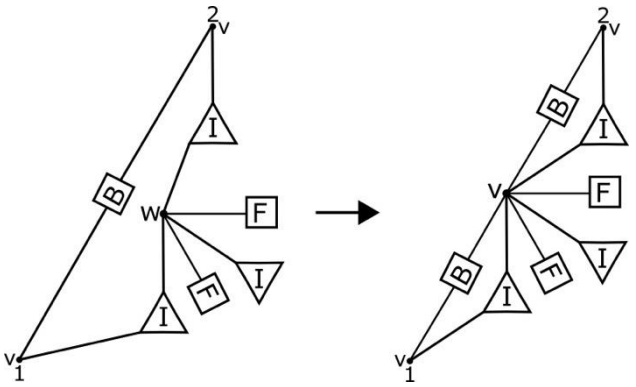


Fig. 11. *A hypergraph production for breaking the boundary edge*

To break a shared edge means to generate two new edges (labeled by *F*) and one new node (labeled by *w*), as it is illustrated in figure 12.

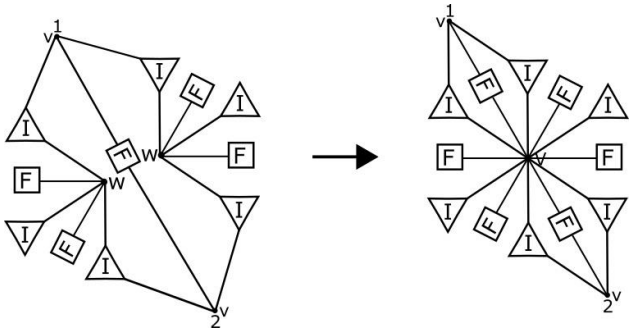


Fig. 12. *A hypergraph production for breaking the shared edge.*

After performing h-adaptation, the solution as well as error of each element are calculated. If the accuracy is insufficient, the virtual adaptation is performed – an element which should be adapted has attribute h set to 1. In order to obtain good element partition tree, some modification of element partition tree should be performed. The algorithm for element partition tree generation takes information about the level of refinement and the adjacency between elements. Two nodes corresponding to adjacent finite elements with the attribute h = 1 should have the same parent. This is done by executing production for element partition tree modification, presented in figure 13.

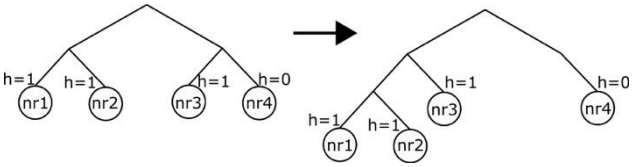


Fig. 13. *A production for element partition tree modification.*

To sum up, the exemplary derivations of the hypergraphs representing finite triangular element meshes together with the generation of element partition tree for the case of point singularity and edge singularity are presented in next two sections.

5. HYPERGRAPH GRAMMAR MODEL FOR FEM WITH THE MESH WITH SINGLE POINT SINGULARITY

The FEM computations are started with the one element mesh with point singularity in the right corner and the corresponding hypergraph and element partition tree presented in figure 6. In the case of singularities, we can refine the mesh in the area, where the numerical error is large, by applying the production breaking mesh interior, as it is illustrated in figures 14a, 14b, 14c.

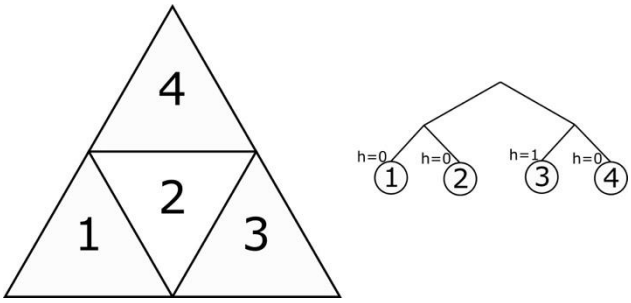


Fig. 14. *(a) An initial mesh after breaking into four elements; (b) an element partition tree of the mesh from figure 14a.*

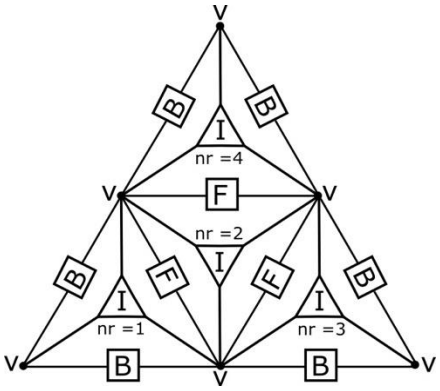


Fig. 14. *(c) A hypergraph representation of the mesh from figure 14a.*

If further adaptation is needed, the element number three will be broken because it is adjacent to the point of singularity, as it is illustrated in figures 15a, 15b, 15c.

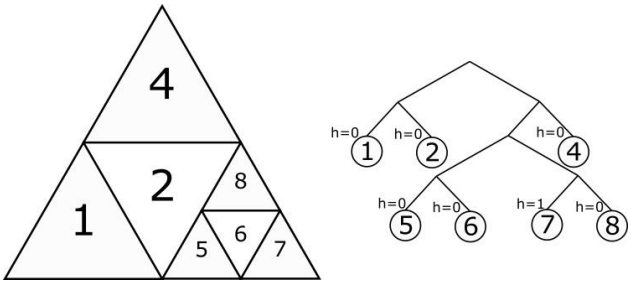


Fig. 15. *(a) Seven elements mesh after breaking element number three into four elements; (b) an element partition tree of the mesh from figure 15a.*

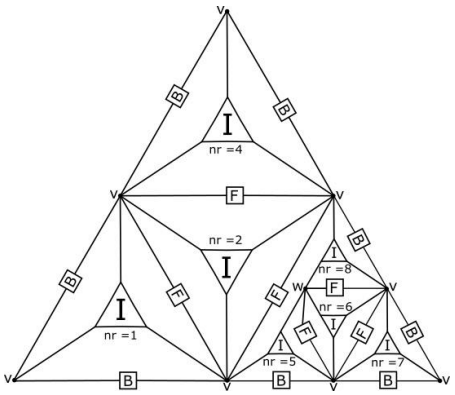


Fig. 15. *(c) A hypergraph representation of the mesh from figure 12a.*

6. HYPERGRAPH GRAMMAR MODEL FOR FEM WITH THE MESH WITH EDGE SINGULARITIES

The FEM computations are started with the one element mesh with edge singularity in the bottom edge and the corresponding hypergraph and element partition tree presented in figure 6. In the case of singularities, we perform refinement of the mesh in the area, where the numerical error is large, by applying the production breaking mesh interior, as it is illustrated in figures 16a, 16b, 16c.

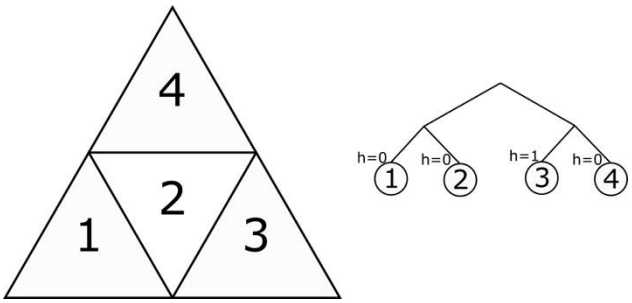


Fig. 16*. (a) An initial mesh after breaking into four elements; (b) an element partition tree of the mesh from figure 16a.*

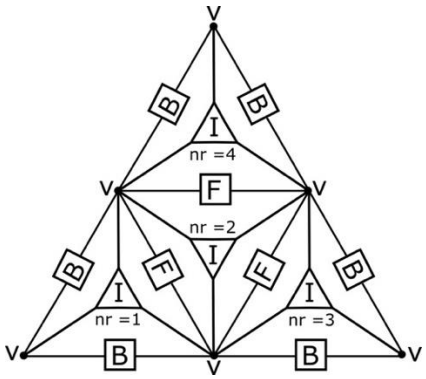


Fig. 16. *(c) A hypergraph representation of the mesh from figure 16a.*

Two adjacent nodes corresponding to finite elements with the attribute $h = 1$ should have the same parent. This is done by executing production for element partition tree modification, presented in figure 16, to the tree from figure 16b, as it is illustrated in figure. 17.

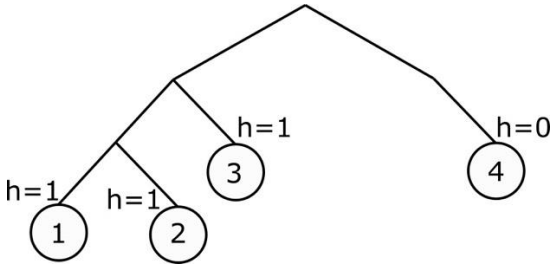


Fig. 17. *An element partition tree after executing production for element partition tree modification.*

In the case further adaptation is needed, the elements number 1, 2, 3 will be broken, each into four elements, because they are adjacent to edge singularity, as it is illustrated in figures 18a, 18b, 18c.

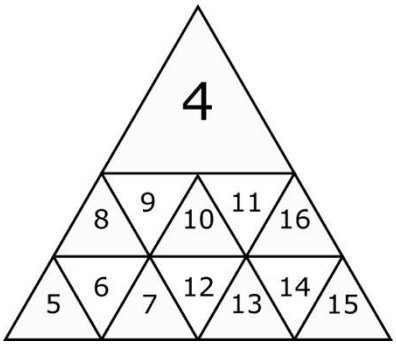


Fig. 18. *(a) Seven elements mesh after breaking elements number 1, 2, 3.*

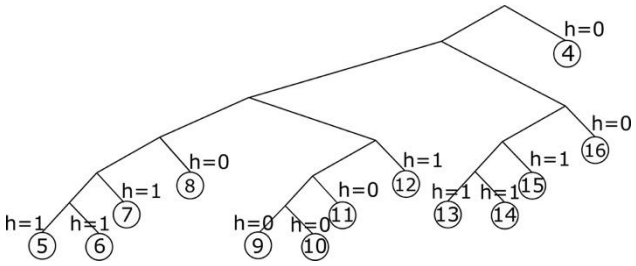


Fig. 18*. (b) An element partition tree of the mesh from figure 18a.*

7. TRANSFORMING THE ELEMENT PARTITION TREE INTO AN ORDERING FOR MULTI-FRONTAL DIRECT SOLVER ALGORITHM

The element partition tree can be transformed into an ordering, which is next sent to the multi-frontal direct solver. The ordering is obtained in the following way. For each mesh node, we list the finite elements where this node belongs.

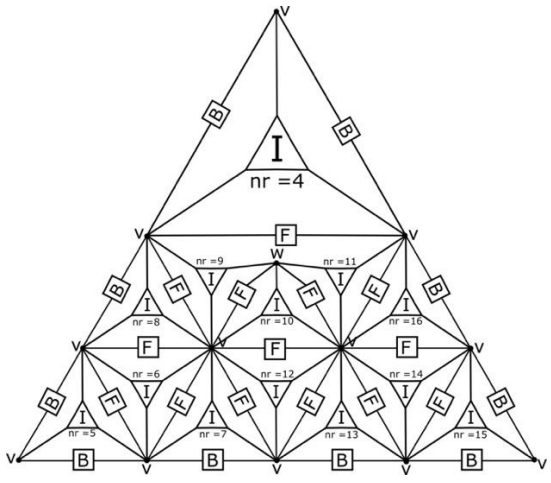


Fig. 18. *(c) A hypergraph representation of the mesh from figure 18a.*

Next, we browse the element partition tree in post-order. The leaves of the element partition tree contain single finite elements, while the internal nodes of the element partition tree inherit the lists of elements from their children. When visiting a given node of the element partition tree, we list in our ordering all the mesh nodes, which list of elements contains in the list of elements of the current node of the element partition tree.

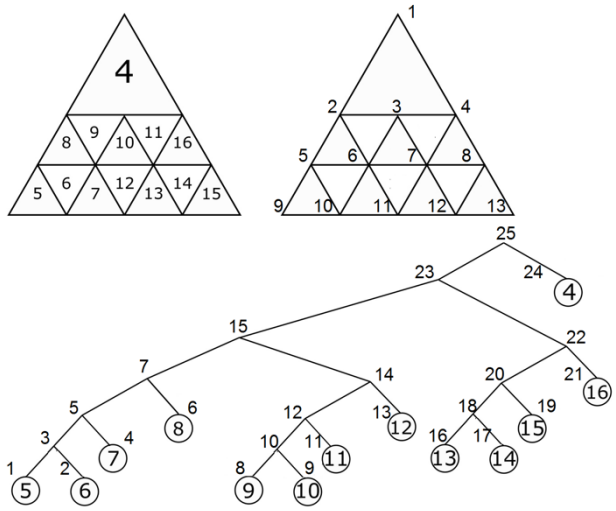


Fig. 19. *Left-top panel: Numbering of elements. Right-panel: Numbering of nodes. Bottom panel: Post-order transition of element partition tree.*

Let us focus on the example presented in figure 19. When we browse the element partition tree in post-order, we start from leaf assigned to element 5. Element 5 has mesh nodes 5, 9, 10, and mesh node number 9 is assigned to element 5 only. Thus our first entry in the ordering is mesh node 9. Next, we go to element 6 (right neighbor in the tree) and this element has mesh nodes 5, 6, 10, but all of them belongs to two or three elements, not only to element 6. Then we go to the parent node in the tree, so our list of elements is 5 and 6. On the common edge between elements 5 and 6 there are mesh nodes 5 and 10. Unfortunately, mesh node 5 belongs to elements 5, 6 and 8, and mesh node 10 belongs to elements 5, 6 and 7. So we do not update the ordering, since now we have only elements 5 and 6 processed. Next, we move to element 7 (no new mesh nodes in the ordering), to parent tree node with elements 5, 6 and 7, and now we put to the ordering mesh node 10. Next, we process element 8, elements 5, 6, 7, 8 at parent tree node, and we update the ordering with mesh node 5. We continue like that with the traveling through the element partition tree presented in figure 19, and we end up with the ordering 9, 10, 5, 6, 11, 12, 13, 7, 8, 1, 2, 3, 4.

8. COMPARISON OF THE NUMBER OF FLOATING POINT OPERATIONS

In this section, we present the experimental comparison on the number of floating-point operations (FLOPs) of our ordering generated from the element partition tree obtained from hypergraph grammar algorithm, with the number of FLOPs resulting from the usage of the nested-dissections algorithm (Liu, 1990) implemented in METIS (METIS) library. Both orderings are sent to multi-frontal solver MUMPS solver (MUMPS) executed in sequential mode, and we record the number of FLOPs reported by MUMPS. The elimination process presented in this chapter concerns any elliptic problem, e.g. Hemholtz equations used for modeling of the wave propagation process, heat transfer or reaction-diffusion models, where the sparsity pattern of the matrix follows the interactions of the mesh nodes where the hierarchical basis functions are used (Demkowicz, 2006). The comparison concerns the triangular mesh refined towards one corner and the triangular mesh refined towards one edge.

The experiments concerning the triangular mesh refined towards one point are presented in figures 20-21. The experiments concerning the triangular mesh refined towards one edge are presented in figure 22-23.

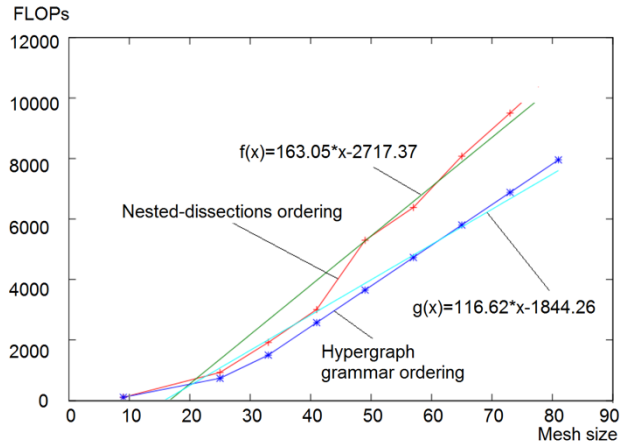


Fig. 20. *Comparison of the flops resulting from the execution of the multi-frontal MUMPS solver with nested-dissections algorithm from METIS library (experimental flops) to flops resulting from execution of our hypergraph grammar solver on triangular mesh h refined towards one point, for linear basis functions.*

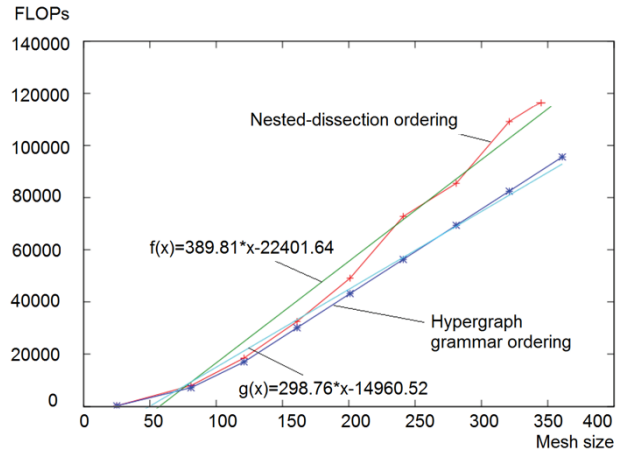


Fig. 21*. Comparison of the flops resulting from the execution of the multi-frontal MUMPS solver with nested-dissections algorithm from METIS library (experimental flops) to flops resulting from execution of our hypergraph grammar solver on triangular mesh h refined towards one point, for quadratic basis functions.*

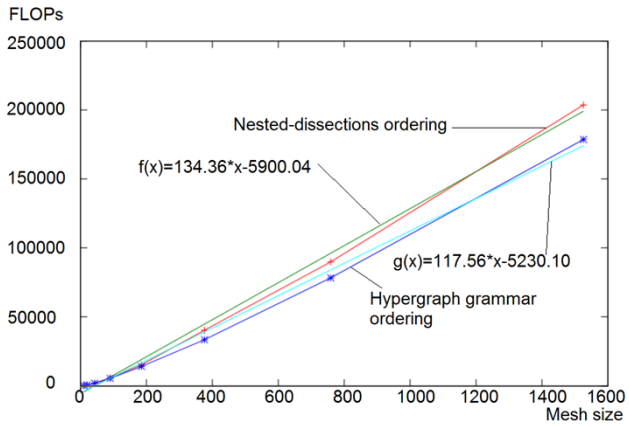


Fig. 22. *Comparison of the flops resulting from the execution of the multi-frontal MUMPS solver with nested-dissections algorithm from METIS library (experimental flops) to flops resulting from execution of our hypergraph grammar solver on triangular mesh h refined towards one edge, for linear basis functions.*

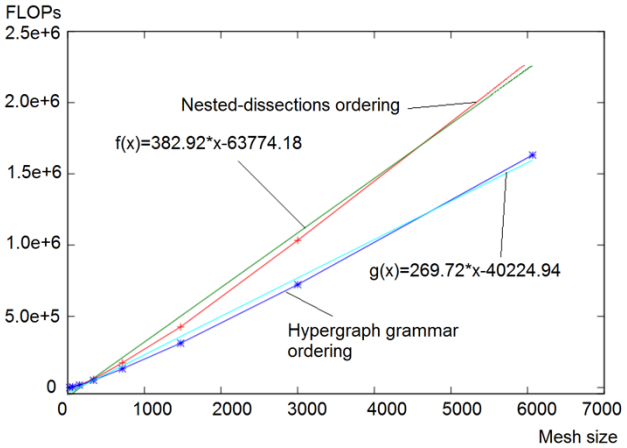


Fig. 23. *Comparison of the flops resulting from the execution of the multi-frontal MUMPS solver with nested-dissections algorithm from METIS library (experimental flops) to flops resulting from execution of our hypergraph grammar solver on triangular mesh h refined towards one edge, for quadratic basis functions.*

We can plot the following conclusions from the experiments:

- Multi-frontal solver executed over triangular mesh refined towards point singularity delivers linear computational cost.
- The hypergraph grammar based algorithm outperforms the nested-dissections algorithm over the mesh with point singularity. In other words, the constant (the slope) in front of the linear computational cost is lower there.
- The multi-frontal solver executed over triangular mesh refined towards edge singularity delivers linear computational cost.
- The hypergraph grammar based algorithm outperform the nested-dissections algorithm over the mesh with edge singularities. In other words the constant (the slope) in front of the linear computational cost is lower there.

9. LINEAR MEMORY USAGE

In this section, we present the experimental measurements on the number of non-zero entries generated by our ordering obtained from element partition tree generated by the hypergraph grammar algorithm. The memory usage has been measured using the GALOIS based (Pingali et al., 2011) implementation of the multi-frontal solver (Paszyńska et al., 2015). This is because the solver works directly on the element partition tree, and thus our measurements involve all the stages of the computational process, involving the browsing of the element partition tree and generation of the ordering. Measuring the memory usage directly from MUMPS solver (MUMPS) will skip the part related to the processing of the element partition tree,

since the MUMPS solver on the interface takes the ordering generated from the element partition tree.

We execute our experiments on the triangular mesh refined towards point singularity, and the triangular mesh refined towards edge singularity.

The experiments for the triangular mesh refined toward one point are presented in figures 24-25, and the experiments for the triangular mesh refined toward one edge are presented in figures 26-27.

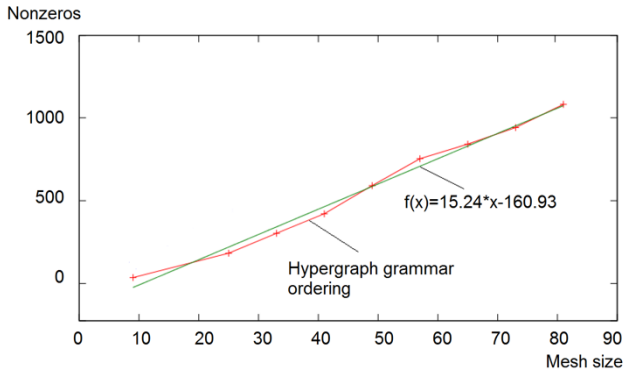


Fig. 24. *Linear O(N) number of non-zeros of the solver executed on 2D triangular mesh h refined towards one point, with hypergraph grammar ordering, for linear basis functions.*

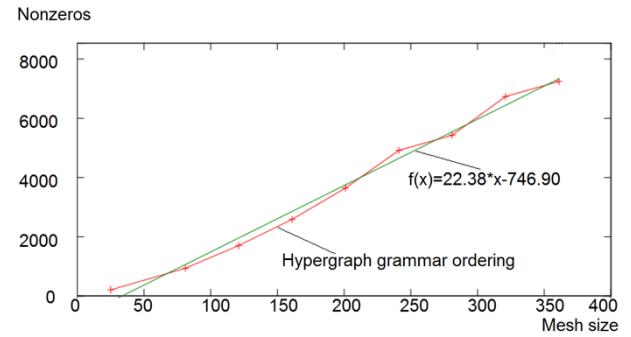


Fig. 25*. Linear O(N) number of non-zeros of the solver on 2D triangular mesh h refined towards one point, for quadratic basis functions.*

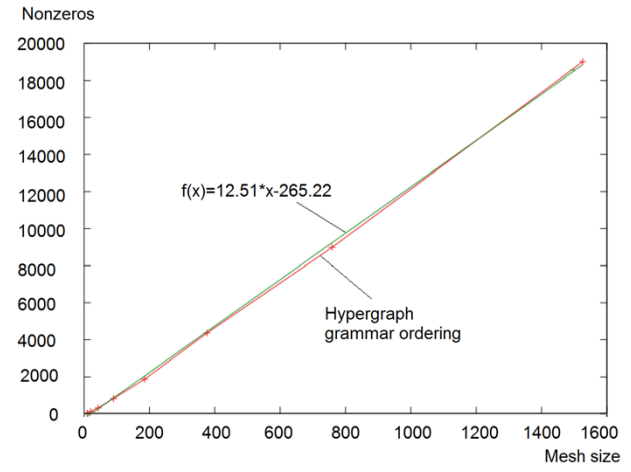


Fig. 26. *Linear O(N) number of non-zeros of the solver on 2D triangular mesh h refined towards one edge, with hypergraph grammar ordering, for linear basis functions.*

We can plot the following conclusions from the experiments:

- Multi-frontal solver executed over triangular mesh refined towards one point delivers linear memory usage,
- Memory usage of the multi-frontal solver with the ordering obtained from the element partition trees generated by the hypergraph grammar outperform the one obtained from the nested-dissections algorithm implemented in METIS library, for the triangular mesh refined towards one point
- Multi-frontal solver executed over triangular mesh refined towards one edge delivers linear memory usage,
- Memory usage of the multi-frontal solver with the ordering obtained from the element partition trees generated by the hypergraph grammar outperform the one obtained from the nested-dissections algorithm implemented in METIS library, for the triangular mesh refined towards one edge

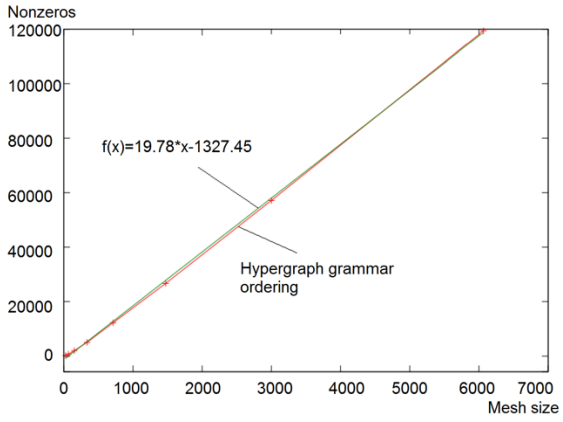


Fig. 27. *Linear O(N) number of non-zeros of the solver on 2D triangular mesh h refined towards one edge, with hypergraph grammar ordering, for quadratic basis functions.*

10. DC RESISTIVITY LOGGING MEASUREMENT SIMULATIONS

In this section we solve 3D direct current (DC) borehole resistivity measurement simulations in a deviated well. Logging instruments move along the borehole through the formation layers, They are equipped with several transmitter and receiver electrodes, and measure the voltage induced at the receiver electrodes at different positions. The measured voltage is expected estimate the electrical conductivity of the nearby formation, so the logging instruments are used to estimate the electrical conductivity of the sub-surface material, with the ultimate objective of describing oil and gas bearing formations. The behavior of an instrument is simulated by performing

computer-based simulations in a borehole environment. The tool with receiver and transmitter electrodes is moving along the trajectory of the well. The electromagnetic waves generated by the transmitter electrode are reflected from formation layers and recorded by the receiver electrodes. Of particular interest are 3D simulations of resistivity measurements in deviated wells, where the angle between the borehole and the formation layers is not equal to 90 degrees A forward DC problem is formulated in the following way:

Find $u: R^3 \supset \Omega \ni x \to u(x) \in R$ the electrostatic scalar potential such that

$$-\sum_{i=1}^{3} \sigma \frac{\partial^2 u}{\partial x_i^2} = \nabla \circ J \text{ in } \Omega \quad (1)$$

where $\nabla \circ J$ is the divergence of the impressed current, (Pardo et al., 2006) and σ represents the conductivity of the media. We introduce a quasi-cylindrical non-orthogonal system of coordinates shown. The full derivation is presented in Pardo et al. (2008a) and Pardo et al. (2008b). The variational formulation, with respect to the electric potential u in the new system of coordinates is the following:

Find $u \in V$ such that:

$$\int_{\Omega} \sum_{n=1}^{3} \frac{\partial u}{\partial \zeta_n} \hat{\sigma} \frac{\partial v}{\partial \zeta_n} d\zeta = \int_{\Omega} v \nabla \circ \hat{J} \, d\zeta \quad \forall v \in V \quad (2)$$

where

$$V = \left\{ v \in L^2(\Omega) : \int_{\Omega} \|v\|^2 + \|\nabla v\|^2 \, dx < \infty : \mathrm{tr}(v) = 0 \text{ on } \Gamma_D \right\} \quad (3)$$

The electric conductivity of media in the new system is equal to $\hat{\sigma} := Jac^{-1} \sigma \, Jac^{-1^T} \, |Jac|$, and $\nabla \circ \hat{J} := \nabla \circ J \, |Jac|$ with Jac being the Jacobian matrix of the change of coordinates from the Cartesian system of coordinates (x_1, x_2, x_3)

$$Jac = \frac{\partial(x_1, x_2, x_3)}{\partial(\zeta_1, \zeta_2, \zeta_3)} \quad (4)$$

We take a Fourier series expansion of the solution, material and the divergence of the impressed current (the source) in the ζ_2 direction

$$u(\zeta_1, \zeta_2, \zeta_3) = \sum_{l=-\infty}^{l=+\infty} u_l(\zeta_1, \zeta_3) e^{\,jl\zeta_2} \quad (5)$$

$$\sigma(\zeta_1, \zeta_2, \zeta_3) = \sum_{m=-\infty}^{m=+\infty} \sigma_m(\zeta_1, \zeta_3) e^{\,jm\zeta_2} \quad (6)$$

$$\nabla \circ J(\zeta_1, \zeta_2, \zeta_3) = \sum_{l=-\infty}^{l=+\infty} \nabla \circ J_l(\zeta_1, \zeta_3) e^{\,jl\zeta_2} \quad (7)$$

where

$$u_l = \frac{1}{2\Pi} \int_0^{2\Pi} u e^{-jl\zeta_2} \, d\zeta_2 ,$$

$$\sigma_m = \frac{1}{2\Pi} \int_0^{2\Pi} \sigma e^{-jm\zeta_2} \, d\zeta_2 ,$$

$$\nabla \circ J_l = \frac{1}{2\Pi} \int_0^{2\Pi} f e^{-jl\zeta_2} \, d\zeta_2$$ and j is the imaginary unit. We introduce symbol F_l such that applied to a scalar function u it produces the l^{th} Fourier modal coefficient u_l, and when applied to a vector or matrix, it produces a vector or matrix of the components being l^{th} Fourier modal coefficients of the original vector or matrix components.

We employ the Fourier series expansions to get the following formulation:

Find $F_l(u) \in V$ such that:

$$\int_{\Omega} \sum_{l,m=-\infty}^{+\infty} F_l\left(\frac{\partial u}{\partial \xi}\right) F_m(\hat{\sigma}) \frac{\partial v}{\partial \xi} e^{\,j(l+m)\zeta_2} \, d\zeta = \int_{\Omega} v F_l\left(\hat{f}\right) e^{\,jl\zeta_2} \, d\zeta \quad \forall v \in V \quad (8)$$

The summation in (8) is with respect to $-\infty \le l, m \le \infty$. We select a mono-modal test function $v = v_k \, e^{\,jk\zeta_2}$. We apply the orthogonality of the Fourier modes in $L^2(\Omega)$ to reduce the problem (8) into:

Find $F_l(u) \in V$ such that:

$$\int_{\Omega_{2D}} \sum_{n=k-2}^{n=k+2} F_l\left(\frac{\partial u}{\partial \xi}\right) F_{k-l}(\hat{\sigma}) F_l\left(\frac{\partial v}{\partial \xi}\right) d\zeta_1 \, d\zeta_3 = \int_{\Omega_{2D}} F_k(v) F_k\left(\hat{f}\right) d\zeta_1 \, d\zeta_3 \quad \forall F_k(v) \in V \quad (9)$$

This is because the five Fourier modes used in (9) are enough to represent the new material coefficients exactly. We refer to Pardo et al. (2008a) for more details.

The exemplary computational mesh generated by the code and processed by our solver is presented in figure 28. The mesh size is 148256 dofs. Different colors correspond to different polynomial orders of approximation from p=1 (blue) to p=8 (pink). Our

solver delivers solution with accuracy 0.001 of the relative error for a single position of the tool within 14.506 s, while the MUMPS solver requires 27.859 s for a solution on this single position antenna. The computations are repeated for consecutive 50 positions of the logging tool.

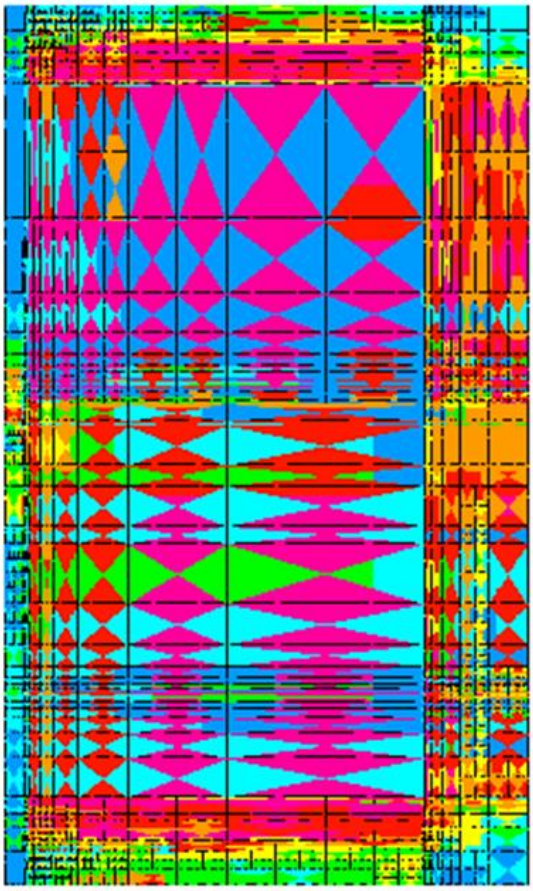


Fig. 28. *Exemplary mesh generated for the DC resistivity logging measurement simulations for a single position of antenna.*

11. CONCLUSIOS

In this paper, we introduced the hypergraph grammar model for generation and adaptation of two-dimensional grids with triangular finite elements. Our hypergraph grammar productions generating the mesh are accompanied by the one generating the element partition tree. The element partition tree can be used for guiding the elimination process of the multi-frontal solver. This can be done either by transferring the element partition tree directly to the GALOIS based solver (Paszyńska et. al., 2015) or by post-processing the tree in post-order to obtain the ordering that can be passed to standard multi-frontal solver like MUMPS (MUMPS). The resulting orderings outperform the computations performed with state-of-the-art nested-dissections ordering, provided by the METIS (METIS) library incorporated into the MUMPS solver. The processing element partition tree and the factorization process for the triangular grids refined towards point or edge delivers linear computational cost and memory usage. We also tested our strategy on industrial process related to borehole resistivity measurements simulations in deviated wells.

ACKNOWLEDGEMENT

This work as supported by National Science Centre, Poland grant no. DEC-2015/17/B/ST6/01867.

REFERENCES

Babuška, I., Rheinboldt, W., 1978, Error Estimates for Adaptive Finite Element Computations, *SIAM Journal of Numerical Analysis*, 15(4), 736-754.

Babuška, I., Szabo, B. A., Katz, I. N., 1981, The p-Version of the Finite Element Method, *SIAM Journal on Numerical Analysis*, 18, 515-545.

Babuška, I., 1986, *Accuracy estimates and adaptive refinements in finite element computations*, John Wiley and Sons.

Banaś, K., Chłoń, K., Cybułka, P., Michalik, K., Płaszewski, P., Siwek, A, 2014, Adaptive finite element modelling of welding processes, *Lecture Notes in Computer Science*, 8500, 391-406.

Demkowicz, L., Pardo, D. , Rachowicz, W., 2002, *3D hp-Adaptive Finite Element Package (3Dhp90) Version 2.0. The Ultimate (?) Data Structure for Three-Dimensional Anisotropic hp-Renements, TICAM Report*, 2-4.

Demkowicz, L., 2006, *Computing with hp-Adaptive Finite Elements*, Vol. I. One and Two Dimensional Elliptic and Maxwell Problems, Chapman and Hall/Crc Applied Mathematics and Nonlinear Science.

Demkowicz, L., Kurtz, J., Paszyński, M, Rachowicz, W., Zdunek, A., 2006, *Computing with hp-Adaptive Finite Elements*, Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications, Chapman and Hall/Crc Applied Mathematics and Nonlinear Science.

Duff, I. S., Reid, J. K., 1983, The multifrontal solution of indefinite sparse symmetric linear, *ACM Trans. Math. Softw*, 9(3), 302-325.

Duff, I. S., Reid, J. K., 1984, The multifrontal solution of unsymmetric sets of linear equations, *Journal on Scientific and Statistical Computing*, 5, 633-641.

Hughues, T., 2000, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Civil and Mechanical Engineering.

Liu, J., 1990, The role of element partition trees in sparse factorization, *SIAM Journal of Matrix Analysis Applications*, 11(1), 134-172.

METIS (n.d.). Metis - graph partitioning and fill-reducing matrix ordering, available online at: http://gl ros.dtc.umn.edu/gkhome/views/metis, accessed: 1.02.2018

MUMPS (n.d.). Multi-frontal massively parallel sparse direct solver, available online at: http://mumps.enseeiht.fr/, accessed: 1.02.2018

Niemi, A., Babuška, I. , Pitkaranta, J., Demkowicz, L/, 2012, Finite element analysis of Girkmann problem using the modern hp-version and the classical h-version, *Engineering with computers*, 28(2), 123-134.

Pardo, D., Demkowicz, L., Torres-Verdin, C., Paszyński, M., 2006, Simulation of Resistivity Logging-While-Drilling (LWD) Measurements Using a Self-Adaptive Goal-Oriented hp-Finite Element Method, *SIAM Journal on Applied Mathematics*, 66, 2085-2106.

Pardo, D., Calo, V. M., Torres-Verdin, C., Nam, M. J., 2008a, Fourier Series Expansion in a Non-Orthogonal System of Coordinates for Simulation of 3D DC Borehole Resistivity Measurements, *Computer Methods in Applied Mechanics and Engineering*, 197(1-3), 1906-1925.

Pardo, D., Torres-Verdin, C., Nam, M. J., Paszyński, M., Calo, V., 2008b, Fourier Series Expansion in a Non-Orthogonal System of Coordinates for the Simulation of 3D Alternating Current Borehole Resistivity Measurements, *Computer Methods in Applied Mechanics and Engineering*, 197(45-48), 3836-3849.

Paszyńska, A., Paszyński, M., Jopek, K., Woźniak, M., Goik, D., Gurgul, P., AbouEisha, P., Moshkov, M., Calo, V. M., Lenharth, A., Nguyen, D., Pingali, K., 2015, Quasi-optimal elemination trees for 2d grids with singularities, *Scientific Programming*, 1-18.

Paszyński, M., 2016, *Fast Solvers for Mesh Based Computations*, Taylor & Francis, CRC Press.

Pingali, K., Nguyen, D., Kulkarni, K., Burtscher, M., Hassaan, M., Kaleem, R., Lee, T.-H., Lenharth, A., Manevich, R., Mendez-Lojo, M., Prountzos, D., Sui, X., 2011, The tao of parallelism in algorithms, *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*, 12-25.

Ślusarczyk, G., Paszyńska, A., 2012, Hypergraph grammars in hp-adaptive finite element method, *Procedia Computer Science*, 18, 1545-1554.

Yannakakis, M., 1981, Computing the minimum fill-in is np-complete, *SIAM Journal on Algebraic Discrete Methods*, 2, 77-79.

ZASTOSOWANIE GRAMATYK HIPERGRAFOWYCH DO GENERACJI QUASIOPTYMALNYCH DRZEW PODZIAŁÓW SIATKI W DWÓCH WYMIARACH

Streszczenie

W artykule przedstawiony został model gramatyk grafowych dla metody elementów skończonych umożliwiający przyspieszenie czasu działania symulacji numerycznych. W proponowanym podejściu operacje na siatce elementów skończonych są wykonywane równocześnie z operacjami generującymi tak zwane drzewo podziałów siatki. Drzewo podziałów siatki określa kolejność wykonywania operacji na macierzy przez solver rozwiązujący problem obliczeniowy. Jakość drzewa podziałów siatki wpływa na czas obliczeniowy solvera. Nasza metoda umożliwia generowanie quasioptymalnych drzew podziałów siatki dla metody elementów skończonych z h-adaptacją. Artykuł jest zakończony opisem wyników numerycznych potwierdzających jakość wygenerowanych drzew podziałów siatki.

Received: May 28, 2018
Received in a revised form: November 19, 2018
Accepted: November 22, 2018