VERSITA

Int. J. Appl. Math. Comput. Sci., 2013, Vol. 23, No. 4, 839–853
DOI: 10.2478/amcs-2013-0063

amcs

USING A GRAPH GRAMMAR SYSTEM IN THE FINITE ELEMENT METHOD

BARBARA STRUG *, ANNA PASZYŃSKA *, MACIEJ PASZYŃSKI **, EWA GRABSKA *

* Department of Physics, Astronomy and Applied Computer Science
Jagiellonian University, ul. Reymonta 4, 30-059 Kraków, Poland
e-mail: {barbara.strug,anna.paszynska,ewa.grabska}@uj.edu.pl

**Department of Computer Science
AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Kraków, Poland
e-mail: maciej.paszynski@agh.edu.pl

The paper presents a system of Composite Graph Grammars (CGGs) modelling adaptive two dimensional hp Finite Element Method (hp-FEM) algorithms with rectangular finite elements. A computational mesh is represented by a composite graph. The operations performed over the mesh are defined by the graph grammar rules. The CGG system contains different graph grammars defining different kinds of rules of mesh transformations. These grammars allow one to generate the initial mesh, assign values to element nodes and perform h- and p-adaptations. The CGG system is illustrated with an example from the domain of geophysics.

Keywords: graph grammar system, automatic hp adaptivity, finite element method.

1. Introduction

The ability to represent the structure of an object and relations of different types between its components makes graphs useful in many domains of computer science. Designing new artifacts requires a method for generation of graphs representing them as well as an easy method for visualization of objects represented by such structures. One of the methods of graph generation is graph grammars, which provide a rewriting mechanism on the level of graphs, consisting in the replacement of subgraphs of graphs by applying grammar rules called productions. The main problem of graph generation with graph grammars lies in the complexity and size (understood as the number of rules) of grammars needed in real world problems. Another difficulty appears in controlling the order in which productions are to be applied for large grammars. To tackle these difficulties, the replacement of graph grammars with a high number of productions by a system of simpler graph grammars was proposed by Grabska and Strug (2005). In the approach applied to CAD, cooperation and distribution formed a base for generating new graphs with the use of graph grammars. The paper is an attempt to use graph grammar system for modelling two dimensional hp Finite Element Method (hp-FEM) adaptive algorithms (Demkowicz, 2006). The finite element method is commonly used in engineering applications (Barboteu *et al*., 2013; Albers *et al*., 2012; Hild, 2011). Adaptive methods consist in constructing a sequence of approximation spaces, which generate with every step of the algorithm a more accurate solution to engineering problem. The finite element mesh contains finite elements and shape functions, corresponding to finite element edges, nodes and interiors which are glued together into a global basis function. Selected finite elements may be broken into smaller elements in order to increase the accuracy of the approximation in these areas of the mesh. The first attempt to model mesh transformations by applying the graph grammar concept was proposed by Flasiński and Schaefer (1996) for regular triangular two-dimensional meshes with h adaptation. This was done using a quasi-context sensitive graph grammar. Spicher *et al*. (2010) modeled mesh refinements by a simple rewriting framework, based on topological chain rewriting. Both approaches allow us only to model uniform refinements, because non-uniform mesh transformations are context dependent and cannot be modelled by the quasi-context sensitive graph grammar.

In the paper the composite graph grammar proposed by Grabska (1993) is used to model the hp-FEM. This approach allows modeling the uniform as well as

nonuniform refinements (Szymczak *et al*., 2011).

The paper is a generalization of the results presented by Paszynska *et al.* (2009; 2012a; 2012b; 2008), who used a composite graph grammar to model the mesh generation process. In some other works (Paszynski, 2009a; 2009b; 2011; 2013; Paszynski and Schaefer 2010) a graph grammar was also employed to describe the parallel algorithm of a multi-frontal direct solver utilized to compute the approximated solution of the boundary-value problem over two dimensional hp finite element meshes.

The paper will focus on modelling the mesh generation process by means of a graph grammar system. The system consists of different graph grammars responsible for different kinds of operations performed over finite elements. These grammars generate structures of the initial mesh and allow performing h- and p-adaptations. The graph grammar model makes it possible to exploit the concurrency of the hp adaptive algorithm. After expressing the mesh transformation and the solver algorithm by means of graph grammar productions, the model of concurrency can be constructed, where basic undividable tasks are represented by graph grammar productions. The approach allows one to find a dependency relation between the tasks, and to group tasks into sets of independent tasks which can be executed in concurrent. This has already been implemented and tested in a two dimensional case over a regular rectangular mesh for a distributed memory architecture in the LONESTAR Linux cluster environment (Paszynski, 2009a; 2009b).

The graph grammar system can also be used as a parser algorithm that, having the finite element mesh, constructs a sequence of graph grammar productions deriving the mesh. In this way the consistency of the finite element mesh obtained from some external mesh generator algorithm can be checked.

Modelling the hp-FEM by means of a graph grammar system instead of a graph grammar allows automation of the deriviation process of the graphs representing the finite element mesh.

The system can also be extended by using inferring graph grammars (Kukluk *et al*., 2008) to model the hp-FEM with other kinds of elements (for example, triangular ones). In this approach the graph grammar for generation of the initial triangular mesh can be found based on sample graphs corresponding to simple triangular meshes.

The paper is organized as follows. First, grammar systems are introduced. Next, formal aspects of composite graphs, composite graph grammars and grammar systems are described. Further, the graph grammar system for generation of the structure of a two dimensional mesh with rectangular elements as well as hp-adaptation of elements is presented. Finally, a numerical example from the domain of geophysics is presented.

2. Grammar systems

Grammar systems were introduced as a way of increasing the generative power of string grammars. A grammar system consists of a finite number of grammars (called the system components) which together change an environment by applying some operations to it. At a given moment, the system state is described by the current environment (sub-environments). The system works by changing its state. Two main types of grammar systems which were investigated for string grammars were Parallel Communicating (PC), grammar systems (Csuhaj-Varjú, 2004) and Cooperating Distributed (CD) grammar systems (Csuhaj-Varjú *et al.*, 1993, 1994; Kelemen, 19991; Martín-Vide and Mitrana 1998). The main difference between these types of grammar systems consists in the way an environment they work on is defined. In PC grammar systems each component grammar has its own copy of the form being operated on/generated and they only exchange information between each other on request. In the case of CD grammar systems, all grammars operate on one common object, but one grammar at a time. As we aim at generating a single mesh, CD grammar systems seem more appropriate here. They allow a number of grammars to work together on one object. At any given time step there is only one graph being generated. Each component grammar operates on this graph according to a communication protocol called the derivation mode. The protocol may allow a single component to perform a predefined number of steps or to work until no production of this component can be applied. The method of selecting which grammar should be used as the next "active" component is also important. Such cooperating grammars can be seen as independent cooperating agents solving the same problem by modifying a common environment (often compared with the blackboard model in artificial intelligence) (Kelemen, 1991)).

In the paper we deal with a modification of CD grammar systems. Yet, in the case of graph grammars, the requirement that only one grammar can operate on a given temporary form (called the sentential form) seems too strong as it would be useful to allow more then one grammar to operate on the same, common, sentential form. It can be seen as an equivalent of many simple tasks being performed on different parts of the same mesh. The only requirement here seems to be ensuring that no two grammars work at the same time on the same part of the mesh, i.e., no two grammars (components) operate on the same nodes of the sentential form. Moreover, some way of activating particular components must be defined. Such a method, called a cooperation strategy, may either be based on some predefined order, thus leading the system to operate under the control of an "external supervisor", which in this case would have the form of a control

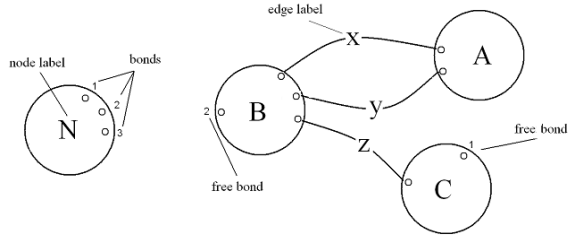

Fig. 1. Node of a cp-graph and an example of a cp-graph.

diagram. Or, it can be based on dynamic activation of components related to the current state of sentential form. The way a given grammar works (called the derivation mode), i.e., how long it performs its operations, must also be defined. In this paper, the so-called terminal derivation mode is used for most grammars, i.e., the grammar works as long as it contains a production that can be applied to a current graph.

3. Composite graphs

In this paper the structure of a mesh is represented by means of the so-called *composite graph (cp-graph)*. The graph nodes correspond to mesh components (edges,vertices and interiors of finite elements). Each node of the graph can contain the so-called *bonds* representing connections of the node with other nodes (Grabska, 1993). Graph nodes are labelled with names of mesh components associated with them and with numbers of their bonds. Graph edges can also be labelled—a label of an edge represents a relation between nodes (Fig. 1). A node of a graph can have the so-called *engaged* bonds, which constitute beginnings or ends of edges, and *free* bonds, which represent potential connections of a node with other nodes. The number of free bonds of a cp-graph is called the *type* of the graph. A formal definition of a labelled, attributed cp-graph is as follows. Let Σ be an alphabet of labels for object nodes, bond nodes and edges. Let for any sequence $(x_1, \ldots, x_n)$, $SET(x_1, \ldots, x_n)$ be equal to $\{x_1, \ldots, x_m\}$, where $m \leq n$, i.e., to the set of its elements. Let A be a set of attributes for nodes and edges, such that for each $a \in A$, D_a denotes its domain, i.e., the set of all possible values of this attribute. Let $P(A)$ denote the set of all subsets of A.

Definition 1. The *composition attributed graph* over Σ is the tuple $g = (V, E, B, bd, s, t, lb, att, val)$, where

- V, B, E are pairwise disjoint sets whose elements are called object nodes, bond nodes and edges, respectively,
- $bd : V \rightarrow B^*$ is the function which specifies a sequence of different bond nodes for each object node, such that $\forall b \in B\ \exists! v \in V : b \in SET(bd(v))$, i.e., each bond node is assigned to exactly one object node,
- $s, t : E \rightarrow B$ are mappings assigning to edges source and target bond nodes, respectively, in such a way that $\forall e \in E\ \exists v_1 \neq v_2, s(e) \in SET(bd(v_1)) \wedge t(e) \in SET(bd(v_2))$, i.e., each edge connects bond nodes of exactly two different object nodes and $\forall b \in B\ \exists$ at most one $e \in E$, such that $b = s(e) \vee b = t(e)$, i.e., there is at most one edge that starts or ends in a given bond node,
- $lb : V \cup B \cup E \rightarrow \Sigma$ is a node and edge labelling function,
- $att : V \cup E \rightarrow P(A)$ is a node and edge attributing function, and
- $val : (V \cup E) \times A \in D$ is a partial function assigning values to attributes, where $D = \bigcup_{a \in A} D_a$ such that $\forall (o, a) \in (V \cup E) \times A$ if $a \in attr(o)\ val(o, a) \in D_a$.

Let G_A be a family of attributed composition graphs. A composition graph $g \in G_A$ such that for each $e \in E$ $s(e) = t(e)$ is called *undirected*. In the following, only undirected composition graphs will be considered.

4. Composite graph grammar

The composite graph grammar is a rewriting mechanism, which consists in the replacement of a subgraph of composite graphs with another graph by applying grammar rules called *productions*. A production is a rule which allows a graph to be modified. Formally, a production is defined in the following way.

Definition 2. A *production* is the triple $p = (l, r, \pi)$, where l and r are composite graphs over Σ of the same type, and π is an applicability predicate $\pi : P(G_A) \rightarrow \{TRUE, FALSE\}$, which on the basis of the values of attributes of L (where L is a subgraph of the current graph c isomorphic with l), r and c determines whether a production can be applied or not.

Definition 3. A *composite graph grammar* G is defined as $G = (\Sigma, P, x)$, where

- Σ is an alphabet of labels,
- P is a finite set of productions $p = \{l, r, \pi\}$, where the set of free bonds of r and l are equipped with ordering relations, and
- x is a labelled node over Σ and is called the *axiom* of the grammar.

Let $p = (l, r, \pi) \in P$ be a production in G. The first element of the pair l and the second element r are called the *left-hand-side* and the *right-hand-side* of p, respectively. The *application of production p to a composite graph c* consists in finding a subgraph L of the current graph c isomorphic with l and substituting an composite graph r for a subgraph L of the current graph c isomorphic to the composite graph l and replacing the connections of l with the connections of r in such a way that each free bond of l is substituted by a free bond of r with the same order number.

Definition 4. Let $G = (\Sigma, P, x)$ be a composite graph grammar and c, c' be composite graphs over Σ. We say that c' *is directly derivable from* c and we write $c \Rightarrow c'$ if

- there exists a production $p = (l, r, \pi) \in P$ and a subgraph L of the composite graph c and c' is isomorphic with the result composite graph generated by an application of the production p to the composite graph c, and
- $\pi(l, r, c) = TRUE$.

For two composite graphs c and c', if c' is obtained as a result of the application of a sequence of productions, which starts from the composite graph c, we say that c' *is derivable from* c and we write $c \Rightarrow^* c'$, where $\Rightarrow^*$ is defined as the reflexive-transitive closure of the binary relation $\Rightarrow$.

Definition 5. Let $G = (\Sigma, P, x)$ be a composite graph grammar. The composite graph grammar in which the left hand-side of all production consists of one node is called the *context-free composite graph grammar*. Otherwise, the grammar is called the *context composite graph grammar*.

Having defined the composite graph grammar, we can introduce a *control diagram*, which determines the order in which the productions should be applied.

Definition 6. Let $G = (\Sigma, P, x)$ be a composite graph grammar. Let $\Omega = \{I, F\} \cup \{p_1, \ldots, p_m\}$, where $m = \sharp P$, be a set of node labels.

By a *control diagram CD over Ω* defined for G we understand a directed node labelled graph $CD = (V_{CD}, E_{CD}, s, t, lb_{CD})$, where

- $s, t : E_{CD} \mapsto V_{CD}$ are two mappings assigning to each edge $e \in E_{CD}$ elements $s(e)$ and $t(e)$ called the source and the target, respectively,
- $lb_{CD} : V_{CD} \mapsto \Omega$, is a labelling function,
- there exists exactly one node labelled I and exactly one node labelled F, and
- there is no edge in-coming to the *initial node* labelled I and no edge out-coming from the *final node* labelled F.

Definition 7. Let $G = (\Sigma, P, x)$ be a composite graph grammar and CD a control diagram over Ω defined for G. A pair (G, CD) is called a *Composite Programmable Graph Grammar (CPGG).*

Definition 8. Let (G, CD) be a composite programmable graph grammar. Let c and c' be two composite graphs and $p = (l, r, \pi) \in P$. Let $e \in E_{CD}$. The pair $(c', t(e))$ is *directly derived from the pair* $(c, s(e))$ iff

- $c \Rightarrow^p c'$ and $lb_{CD}(t(e)) = p$, or
- $x \Rightarrow^p c'$, where $x = c$ is an axiom of G, and $lb_{CD}(s(e)) = I$, $lb_{CD}(t(e)) = p$, or
- $c = c'$ and $lb_{CD}(t(e)) = F$.

Let (G, CD) be a composite programmable graph grammar. Let $e, \hat{e} \in E_{CD}$. For two composite graphs c and c' if the pair $(c', t(\hat{e}))$ is obtained as a result of the application of a sequence of directly derivations, which starts from the pair $(c, s(e))$, we write $(c, s(e)) \Rightarrow^* (c', t(\hat{e}))$, where $\Rightarrow^*$ is defined as the reflexive-transitive closure of the binary relation $\Rightarrow$.

Definition 9. Let (G, CD) be a CGG and let v_I and v_F denote the initial and final nodes of the CD, and let c be an composite graph. The *language generated by the CGG* is defined as $L(G, CD) = \{c : (x, v_I) \Rightarrow^* (c, v_F)\}$.

In some situations a need can arise to "propagate" some information throughout the generation process, i.e., a way of communicating between productions is needed. In this paper we propose an approach similar to the blackboard method in artificial intelligence, i.e., all the productions of a given grammar can access and/or modify a common set of variables. This set will be called *memory*.

Formally we can define a grammar with memory in the following way.

Definition 10. A *grammar with memory* is a pair $GM = (G, M)$, where

- M is a set of variables, $M = \{m_1, \ldots, m_n\}$, such that $\forall i = 1, \ldots, n, m_i \in D_i$, where D_i is the set of all possible values for m_i, and
- $G = (\Sigma, P, x)$ is a composite graph grammar, where P is the set of productions $p = (l, r, \pi, f)$ defined as follows:
  - l and r are left- and right- hand sides of p, respectively,

- π is an applicability predicate, $\pi : P(G_A) \times P(M) \rightarrow \{TRUE, FALSE\}$, where L is a subgraph of a current graph isomorphic with l and M is a memory associated with the grammar G,
- $f = (f_1, \ldots, f_n)$ is a set of functions responsible for changing values of memory variables, where $f_i : p(M) \times P(A_L) \rightarrow D_i$, $1 \leq i \leq n$.

It can be noticed here that the way a predicate π is defined had to be extended to take into account the possibility of including memory variables in deciding whether or not a production can be applied. It is also important that the above definition does not put any additional constraints on the type of grammar G, i.e., it can be a context-free or context grammar.

5. Composite graph grammar system

A composite graph grammar system is a system which consists of composite programmable graph grammars (composite graph grammars and their control diagrams), an initial graph and a control diagram for the whole system which controls the execution of graph grammars. The graph grammar system allows a number of grammars to work together on one object. Each component grammar operates on the graph according to a communication protocol called the derivation mode. The protocol may allow a single grammar to perform a predefined number of steps or to work until no production of this grammar can be applied. The method of selecting which grammar should be used as the next "active" component as well as the derivation mode of the component is defined by means of a control diagram.

An important factor here is that we do not put any constraints on types of grammar constituting the grammar system. That means that the same grammar system can contain context-free and context grammars, and each of them with or without memory. A real life problem that would normally require a context grammar with memory and thus result in a high computational cost associated with this type of grammars can be solved significantly faster by dividing it into a number of grammars. Usually only some of grammars would be context grammars and only some grammars with memory, while some grammars would be context-free. As a result, each part of a problem is solved by applying the simplest grammar possible.

Definition 11. A *composite graph grammar system* S is defined as

$$S_D = (g_0, G_1, \ldots, G_n, CD_1, \ldots, CD_n, D), \quad (1)$$

where g_0 is an initial composite graph, $G_1, \ldots, G_n$ are composite programmable graph grammars with control diagrams $CD_1, \ldots, CD_n$, respectively, with $G_i = (\Sigma_i, lb_i, V_i, P_i, x_i)$, and the control diagram $CD_i = ((V_{Di}, E_{Di}, s_{Di}, t_{Di}), \xi_{V_{Di}})$ being defined over $\Omega_i = \{I_i, F_i\} \cup \{p_1^i, \ldots, p_{m_i}^i\}$, $P_i = \{p_1^i, \ldots, p_{m_i}^i\}$.

Let $\Omega = \{I, F\} \cup \{G_1, \ldots, G_n\}$. $D = ((V_D, E_D, s_D, t_D), \xi_{V_D})$ is a control diagram for the grammar system, where the node labels, except the start node and the final node are names of grammars $G_1, \ldots, G_n$, the initial node has the label I and the final node has the label F.

Similarly as in a single grammar case, when applying a grammar system to real life problems a need to communicate between grammars may occur. To this end, we introduce a grammar system with memory.

Definition 12. A *grammar system with memory* is a triple $SM = (S, M, F)$, where S is a grammar system, M is a memory and F is a family of functions allowing each production of each grammar in S to change the value of elements of system memory M.

It has to be noted here that some or all grammar in the grammar system can actually be grammars with memory. Thus the system will contain two different levels of memory. One level memory is associated with a single grammar and is unknown to the rest of the members of the grammar system, while the global system memory can be accessed by any grammar.

Definition 13. Let

$$S_D = (g_0, G_1, \ldots, G_n, CD_1, \ldots, CD_n, D)$$

be a composite programmable graph grammar system. Let c and c' be two composite graphs. Let $\tilde{e} \in E_D$ of D and $e \in E_{Di}$ of D_i.

The triple$(c', t_D(\tilde{e}), t_{Di}(e))$ is *directly derived* from the triple $(c', s_D(\tilde{e}), s_{Di}(e))$ in the graph grammar system iff

- $s_D(\tilde{e}) = t_D(\tilde{e})$, $lb_{V_D}(s_D(\tilde{e})) = lb_{V_D}(t_D(\tilde{e})) = G_i$, $s_{Di}(e), t_{Di}(e) \in CD_i$ and the pair $(c', t_{Di}(e))$ is directly derived from the pair $(c, s_{Di}(e))$, or
- the graph c is isomorphic with the graph c', and one of the following conditions is fulfilled

$$\begin{aligned} lb_{V_D}(s_D(\tilde{e})) &= G_i, & lb_{V_D}(t_D(\tilde{e})) &= G_j, \\ lb_{V_{Dj}}(t_{Dj}(e)) &= I_j, & lb_{V_{Di}}(s_{Di}(e)) &= F_i, \\ lb_{V_D}(s_D(\tilde{e})) &= I, & lb_{V_D}(t_D(\tilde{e})) &= G_i, \\ lb_{V_{Di}}(t_{Di}(e)) &= I_i, & c &= g_0 \end{aligned}$$

is an initial graph,

$$lb_{V_D}(t_D(\tilde{e})) = F, \quad lb_{V_D}(s_D(\tilde{e})) = G_i, \quad lb_{V_{Di}}(s_{Di}(e)) = F_i.$$

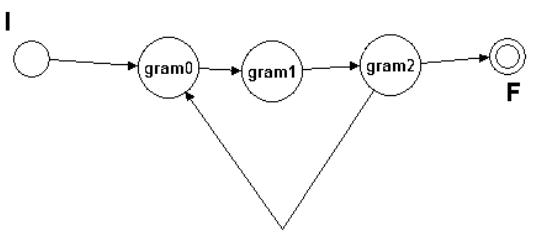

Fig. 2. Example control diagram for a graph grammar system with three grammars.

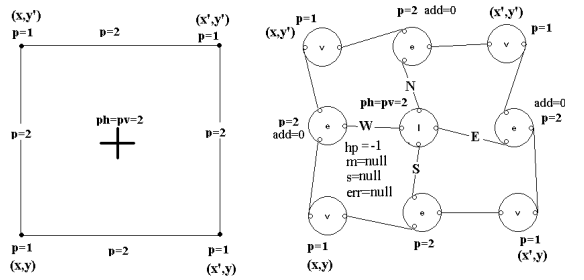

Fig. 3. Example rectangular elements and their cp-graph representation.

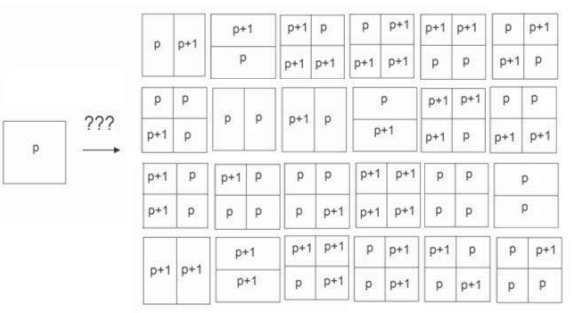

Fig. 4. Possible kinds of adaptation.

If the triple (c', V', v') is directly derived from the triple (c, V, v) in the graph grammar system, we write $(c, V, v) \Rightarrow (c', V', v')$.

Definition 14. Let

$$S_D = (g_0, G_1, \ldots, G_n, CD_1, \ldots, CD_n, D)$$

be a composite programmable graph grammar system, and let V_I and V_F denote the initial and final nodes of D, respectively. Let v_I and v_F denote the initial and final nodes of the control diagrams defined for V_I and V_F, respectively. Let c be a composite graph. A *language generated by the composite programmable graph grammar system* is defined as $L(S) = \{c : (g_0, V_I, v_I) \Rightarrow^* (c, V_F, v_F)\}$.

Figure 2 presents an example control diagram for a graph grammar system, which consists of three grammars: $gram0, gram1, gram2$. The first node of the diagram is the initial node with label I. The last node of the diagram is the final node with label F. The labels of the remaining nodes are the names of the grammars. Each grammar has its own internal control diagram. The internal control diagram corresponding to the initial node consists only of one initial node. The internal control diagram corresponding to the final node consists only of one final node.

6. cp-graph grammar system for modelling the *hp*-adaptive finite element method

The section presents a cp-graph system modelling the hp-finite element method: the finite 2D element mesh with rectangular elements is represented by a cp-graph, the mesh transformations are modelled as cp-graph grammars rules.

Figure 3 presents an example rectangular element an its cp-graph representation. Each rectangular element consists of four vertices (nodes with label v), four edges (nodes with label e) and one interior (node with label I). The polynomial order of approximation is denoted by p, the coordinates of vertices are denoted by (x, y). Each node with label v of the cp-graph has attributes (x, y) and p. Each node with label e has attributes p and add. The attribute add can equal 0 if the edge belongs to only one rectangular element and 1 if it belongs to two rectangular elements. A node with label I has attributes ph and pv, which denote the polynomial order of approximation in the horizontal and vertical directions. Additionally, the node with label I has attributes m, s, err, which respectively denote the matrix, the vector of the solution for the rectangular element and the vector of error for the rectangular element. The attribute hp, which denotes the kind of hp-adaptation, has the value from the set $-1, 0, 1, 2, \ldots, 24$, where hp equals -1 if the decision about the adaptation has not been made yet, 0 in the case of no adaptation and one from the values $1, \ldots, 24$ in the case of adaptation. The meaning of the values of the attribute hp, which denotes several kinds of adaptation, is presented in Fig. 4. The kinds of adaptation are numbered from 1 to 24, row by row.

A cp-graph grammar system for modelling the hp-adaptive two dimensional finite element method consists of five grammars: grammar $G1$ responsible for generating the the initial mesh, grammar $G2$ for generating the matrix for each element, grammar $G3$ for solving the problem, grammar $G4$ for making virtual hp-adaptation and grammar $G5$ for performing hp-adaptation.

Figure 5 presents the control diagram for the cp-graph grammar system modeling the hp-FEM. The

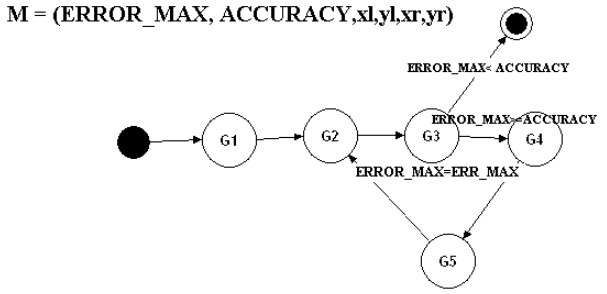

Fig. 5. Control diagram for a cp-graph grammar system modeling the hp-FEM.

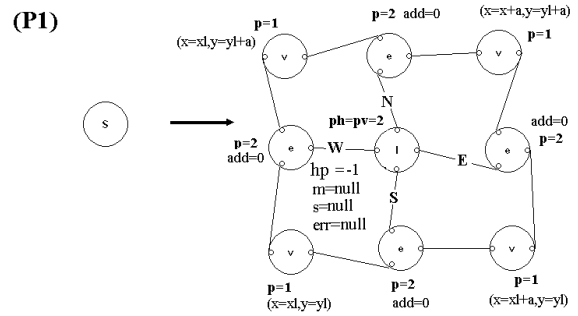

Fig. 6. One of the cp-graph grammar productions for generation of the initial mesh.

variables $ERROR\_MAX, ACCURACY, xl, yl, xr, yr$ from the system memory M are initialised by the user. The terminal derivation mode is used for each grammar from the system. After performing the graph grammar rules from the graph grammar $G1, G2$, the productions from graph grammar $G3$ are performed. The grammar $G3$, which solves the problem, has its own memory with the variable called ERR_MAX. The value of the variable ERR_MAX is equal to the maximal error calculated by productions from $G3$. After performing rules from the graph grammar $G3$, the value of the global variable $ERROR\_MAX$ is changed to ERR_MAX. If the maximal error ($ERROR\_MAX$) is bigger than the predefined accuracy of the solution ($ACCURACY$), the productions from the graph grammar $G4$ are preformed. If the maximal error is smaller than the predefined accuracy of the solution, the whole system is stopped.

Figures 6–11 present productions for generation of the initial mesh.

Figure 12 presents the control diagram for the cp-graph grammar $G1$ modeling the generation of the initial mesh. The graph grammar $G1$ is the grammar with the memory denoted by $M1$. Based on the actual values of the variables xl, yl, xr, yr from the memory $M1$, the decision about performing productions from a graph grammar $G1$ is made (see applicability predicates from Figs. 6–11).

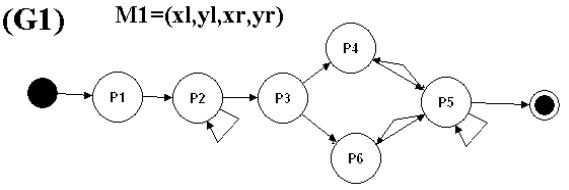

Fig. 12. Control diagram for the graph grammar $G1$ for generation of the initial mesh.

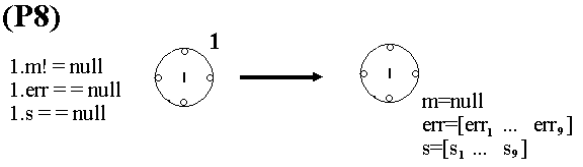

Fig. 14. cp-graph grammar production for calculating the vectors of the solution and error on the basis of a matrix.

Figure 13 presents the production of the graph grammar $G2$ which can be applied only if the matrix for an element was not calculated before (the attribute m equals $null$). The production calculates the matrix for a finite element.

Figure 14 presents the production of the graph grammar $G3$, which calculates the vector of the solution s and error err on the basis of the matrix m. The applicability predicate for the production is presented on the left-hand side of Fig. 14.

Figure 15 presents the control diagram for the cp-graph grammar $G3$, calculating error and solution vectors.

Figure 16 presents one of the productions of the graph grammar $G4$, which performs virtual hp-adaptation—it makes a decision about adaptation. The production $P9$ assigns a suitable value for the attribute hp. The decision about the kind of adaptation is made based on the error value calculated for the interior node I. If the value of the error for the interior I is bigger than 33 per cent of the maximal error, the external function z calculates the best kind of adaptation for this element interior (Figs. 16 and 17). If the value of the error for the interior I is smaller than 33 per cent of the maximal error, value 0 (no adaptation) is assigned to the attribute hp.

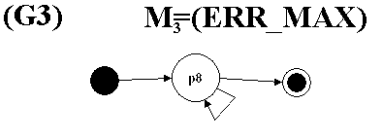

Fig. 15. Control diagram for the graph grammar $G3$ calculating error and solution vectors.

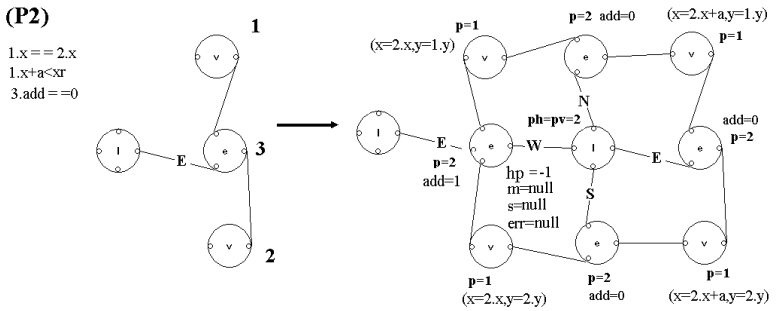

Fig. 7. One of the cp-graph grammar productions for generation of the initial mesh.

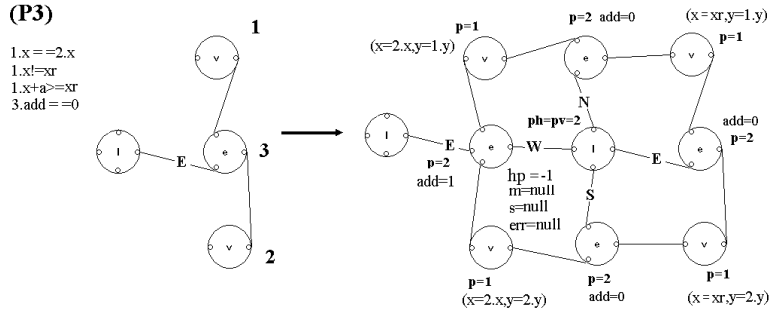

Fig. 8. One of the cp-graph grammar productions for generation of the initial mesh.

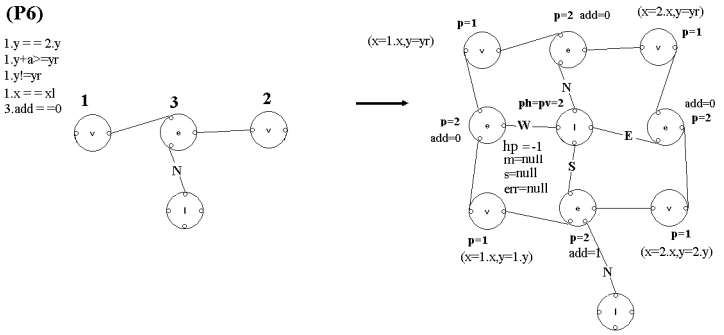

Fig. 9. One of the cp-graph grammar productions for generation of the initial mesh.

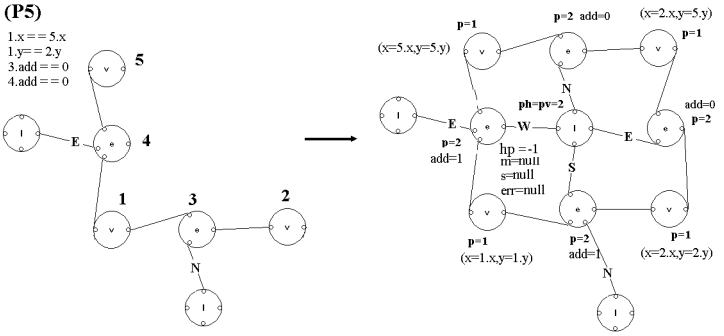

Fig. 10. One of the cp-graph grammar productions for generation of the initial mesh.

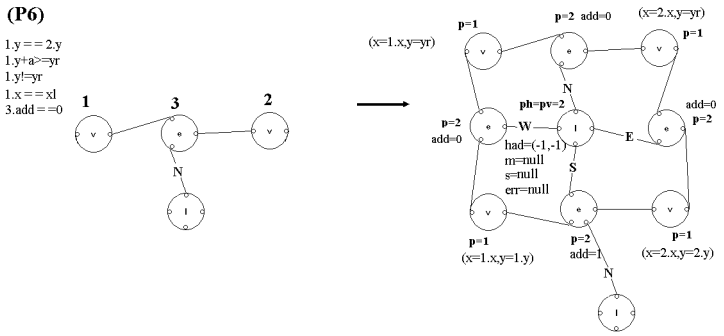

Fig. 11. One of the cp-graph grammar productions for generation of the initial mesh.

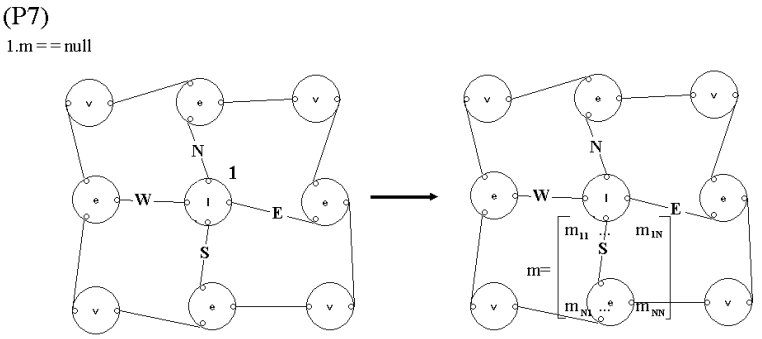

Fig. 13. cp-graph grammar productions for calculating the matrix.

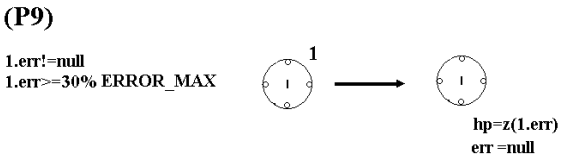

Fig. 16. cp-graph grammar production for virtual hp.

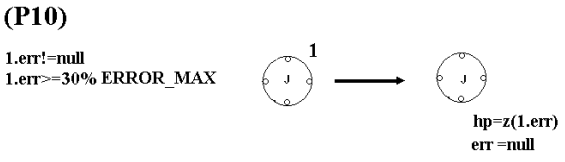

Fig. 17. cp-graph grammar production for virtual hp.

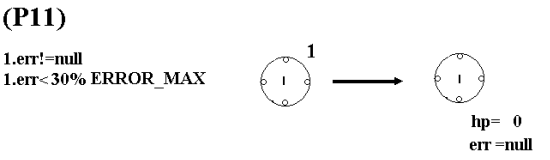

Fig. 18. cp-graph grammar production for virtual hp.

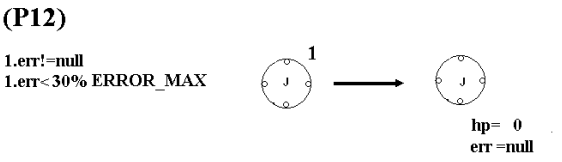

Fig. 19. cp-graph grammar production for virtual hp.

(Figs. 18 and 19). In order to follow the one irregularity rule (a finite element cannot be broken for the second time without first breaking larger adjacent elements), after performing virtual hp-adaptation for each element, production propagating the adaptation has to be applied (Fig. 20).

Figure 21 presents the control diagram for the cp-graph grammar $G4$.

The graph grammar $G5$ is responsible for performing

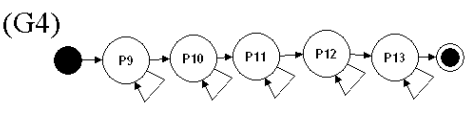

Fig. 21. Control diagram for the graph grammar $G4$ for virtual hp-adaptation.

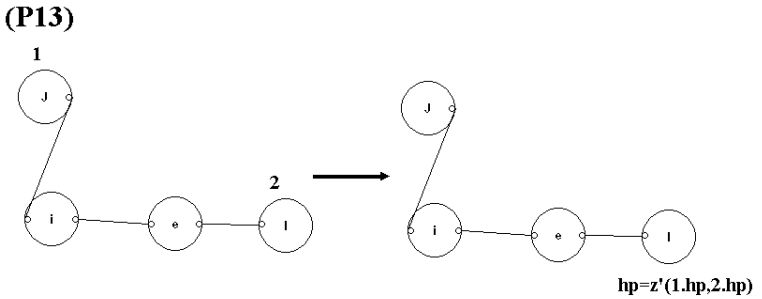

Fig. 20. cp-graph grammar production propagating the hp-adaptation.

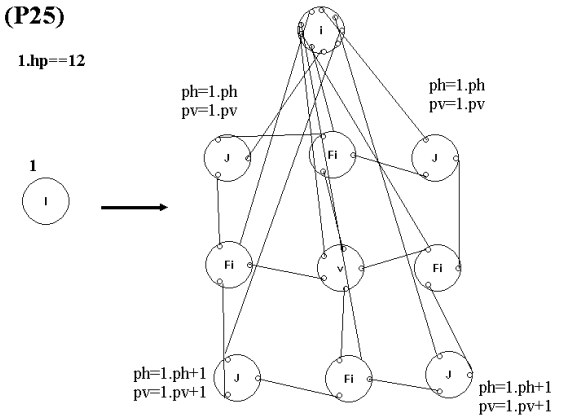

Fig. 22. Graph grammar production for breaking the interior of a rectangular element.

hp-adaptation. Possible kinds of adaptation are presented in Fig. 4. The process of breaking an element consists of

- breaking elements interiors,
- allowing for breaking an edge, if two adjacent elements are broken,
- breaking an edge, and
- allowing breaking the interior in the case of broken adjacent edges.

To break a rectangular element interior into four new elements means generating new element interiors, four new edges and one new vertex. Figure 22 presents one of the productions for hp-adaptation, where hp equals 12. Similar productions ($P14-P37$ from Fig. 28) are defined for other values of the attribute hp.

After breaking element interiors, the edges have to be broken. An edge can be broken only if two adjacency interiors are broken. Figure 23 presents the production which allows breaking an edge if two adjacency interiors are broken.

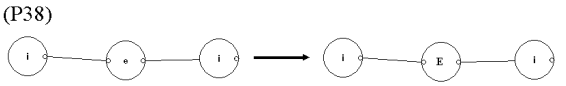

Fig. 23. Graph grammar production which allows breaking an edge.

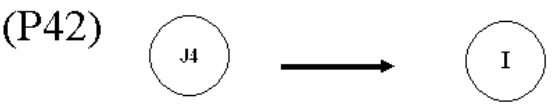

Fig. 26. Production which allows breaking an interior.

To break an element edge means generating two new edge nodes and one new vertex. Figure 24 presents the production for breaking an edge.

An element interior can be broken if it is not adjacent to a large unbroken edge. Figure 25 presents one of productions checking if an element interior is adjacent to a proper size edge. The number of adjacent proper size edges is denoted by changing the graph node symbol label from J into $J3$ and finally into $J4$.

Production in Fig. 26 allows breaking an interior if it is adjacent to proper size edges—changes the label of the interior node from $J4$ to I.

The last step of the hp-adaptation process is performing the minimum rule in order to calculate the polynomial order of approximation for edges (the value of attribute p). The production, which assigns values for attribute p for edges is presented in Fig. 27

Figure 28 presents the control diagram for breaking 2D finite elements.

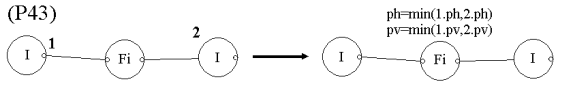

Fig. 27. Production for performing the minimum rule.

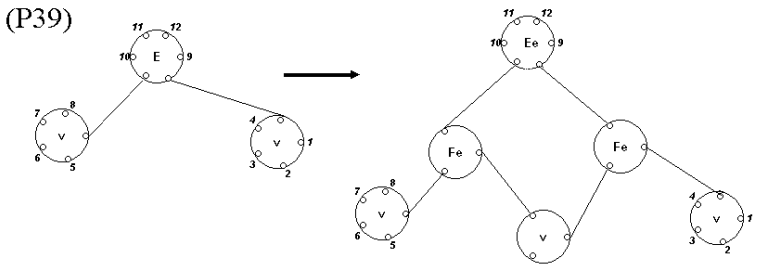

Fig. 24. Graph grammar production for breaking an edge.

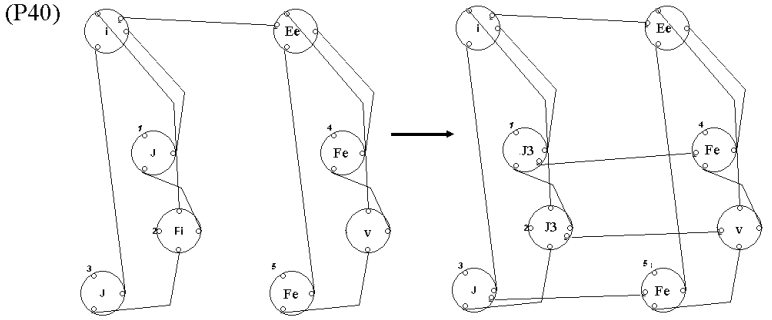

Fig. 25. Production checking if an element interior is adjacent to a proper size edge.

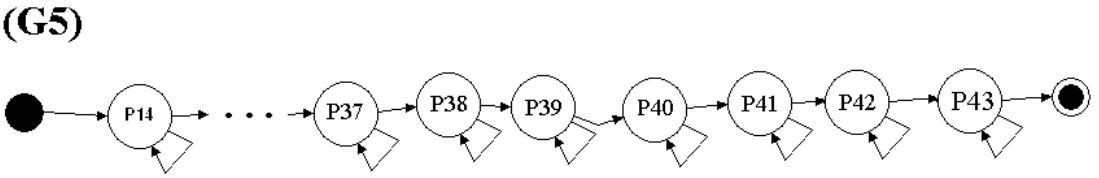

Fig. 28. Control diagram for breaking 2D finite elements.

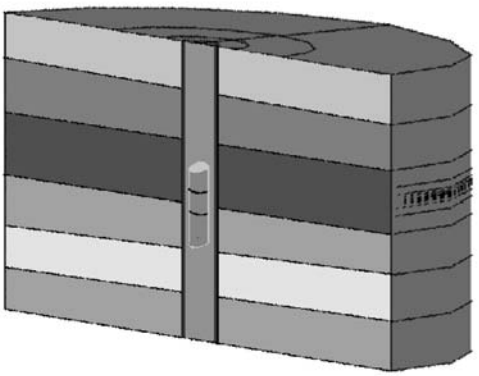

Fig. 29. Borehole with the tool and formation layers.

7. Numerical results

The paper is illustrated with numerical results presenting the 3D DC resistivity measurement simulations, which are of great interest to oil industry. In the last years, several numerical methods have been utilized to simulate three-dimensional resistivity logging measurements for the assessment of rock formation properties. The most widely used methods are finite-difference schemes using Krylov-based subspace solvers (see, e.g., Zhang *et al.*, 1995; Druskin *et al.*, 1999; Newman and Alumbaugh, 2002; Wang and Fang, 2001) or finite elements. The term "simulations of measurement" is widely used in the geophysical community. The goal of the measurements is to detect the properties of the formation layers, with the ultimate goal of locating oil or gas formations in the ground. The detection of the bearing formations is performed with the use of the resistivity logging tools, which are located in the borehole environment, see Fig. 29. The borehole is denoted by the light gray color, and there are several formation layers in the ground, which are assumed to be perpendicular to the borehole.

The formation layers may correspond to rock, sand, mud and, finally, oil. Each of these formation layers may have different resistivity, as is illustrated in Fig. 30.

The logging tool is usually equipped with one transmitter and several receiver electrodes. The transmitter electrode generates electromagnetic waves which propagate through the formation layers, and are reversed and recorded by the receiver electrodes. The logging tool is shifted along the borehole and the values recorded by the receiver form the so-called logging curve, which is used by the geologists to locate the bearing formations in the ground. Numerical simulation of the measurement process in the case of direct current antennas reduces to the solution of the conductive media equation

$$\nabla \cdot (\sigma \nabla u) = f, \tag{2}$$

where u is the potential of the electric field $E = -\nabla u$ and

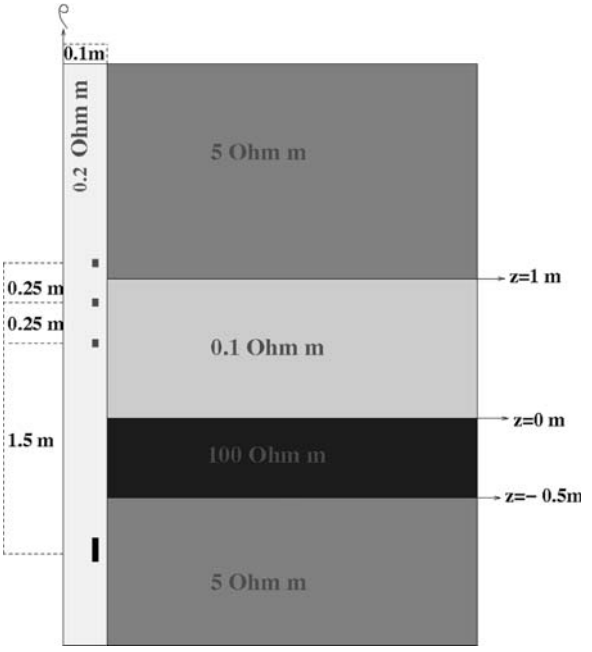

Fig. 30. Cross-section of the computational domain: the borehole with resistivity 0.2 ohm $\times$ m, and the four formation layers with resistivities 5, 0.1, 100 and 5 ohm $\times$ m, corresponding to the rock, mud and oil. The tool is modeled with one transmitter and three receiver electrodes.

σ corresponds to the resistivities of the formation layers. Equation (2) results from zero frequency Direct Current (DC) formulation of Maxwells equations (Pardo *et al.*, 2006; 2007; 2008).

In the case of formation layers perpendicular to the borehole, the problem becomes axially symmetric, and we can limit ourselves to the 2D dimensional problem formulated on the cross-section of the mesh. We start our computation with the two-dimensional mesh presented in Fig. 31. The mesh was obtained by using the graph grammar $G1$ responsible for the generation of the initial mesh. The mesh was intentionally adjusted to the geometry of the antennas and the assumed formation layers. The domain under consideration presented in Fig. 30 consists of four formation layers with different resistivities varying from 0.1 ohm $\times$ m up to 100 ohm$\times$m. In the borehole with resistivity 0.2 ohm$\times$m there is one transmitter and three receiver electrodes. The three electrodes are modeled by assuming non-zero right-hand-side representing the imposed electric current. The height of the domain is 200 meters, and the radius is 100 meters. The zero Dirichlet boundary conditions are assumed on the exterior of the domain.

The simulation requires a sequence of solutions of Eqn. (2) for 80 different positions of transmitter and receiver electrodes. The electrodes are shifted along the borehole, and the values at the receiver electrodes are

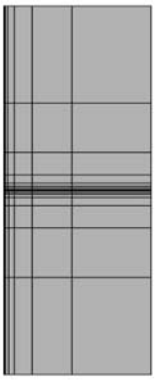

Fig. 31. Initial computational mesh obtained by using the $G1$ graph grammar.

obtained from the finite element method for each new position. The estimation of the second vertical difference of potential at receiver electrodes is finally computed based on the values at the three receiver electrodes. This quantity is an approximation of

$$\frac{\partial^2 u}{\partial z^2} = \frac{\partial E}{\partial z}, \tag{3}$$

which is physically expected to be sensitive to the resistivity of the formation.

The goal of this simulation is to obtain this quantity for a sequence of different vertical positions of the tool.

The problem was solved by using the graph grammar based adaptivity, namely, we executed in a loop the graph grammar $G2$ for generation of the system of linear equations, the graph grammar $G3$ for solution of the problem, followed by graph grammars $G4$ and $G5$ for mesh refinements.

The problem was solved for a sequence of 80 different positions of the tool. Each problem requires a full sequence of graph grammars, $G1$ for generation of the mesh with a new tool position, and $G2$–$G5$ for a loop of mesh refinements. A sample mesh obtained for a single position of the tool is presented in Fig. 32. The resulting logging curve is presented in Fig. 33.

The graph grammar based software for resistivity logging simulations can be used by geologists for experimenting on the influence of different configurations of the formation layers on the shape of the resulting logging curve.

8. Conclusions and future work

The system of graph grammars can be utilized for modeling engineering computations by means of the adaptive finite element method. The expression of the computational process by a graph grammar allows partitioning the process into basic undivisible tasks,

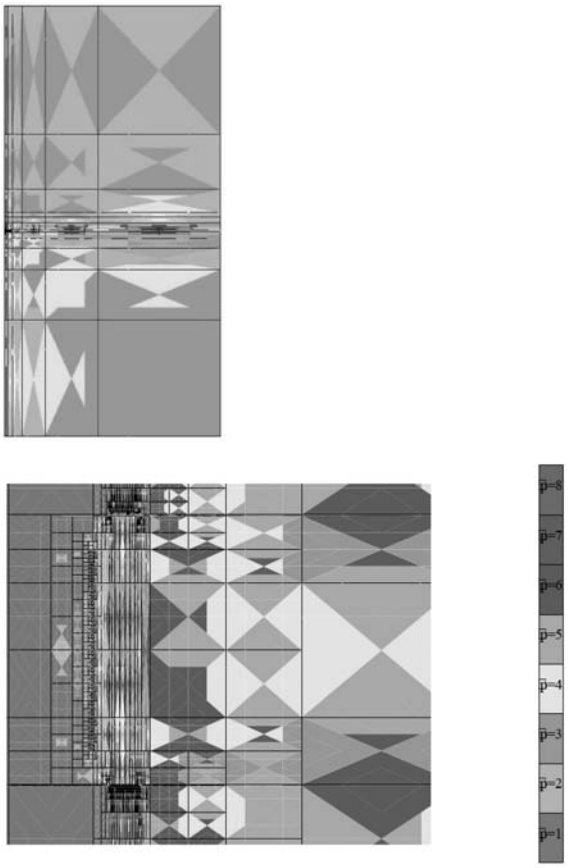

Fig. 32. Top panel: the computational mesh generated by the adaptive procedure based on the graph grammars $G2$–$G5$. Middle panel: zoom at the adaptivities on the receiver electrodes. Different shades correspond to different polynomial orders of approximation on finite element edges and interiors. Right panel: legend for the correspondence between the polynomial order of approximation and colors.

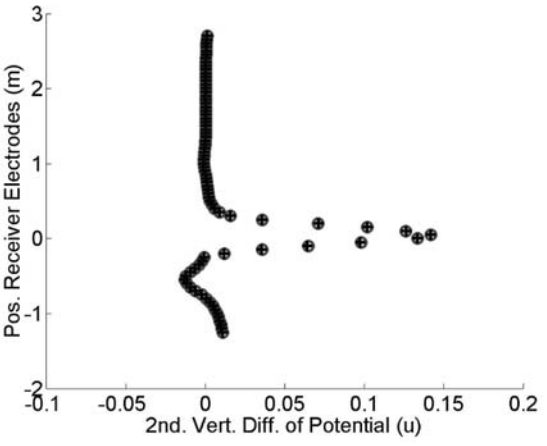

Fig. 33. Logging curve obtained from simulation.

called the graph grammar productions, and localizing the dependency relationship between them. This, in turn, enables us to develop a parallel version of the computational algorithm where the tasks are executed step by step, concurrently, with sets of independent tasks. The presentation is concluded with an example from the domain of geophysics. In the future the graph grammar system for two dimensional model with the rectangular elements will be extended to the case of mixed triangular and rectangular elements as well as to the 3D case.

Acknowledgment

The work presented in this paper was partially supported by the Polish National Science Center, Grant No. NN 519 447739.

References

Albers, B., Savidis, S.A., Taşan, E., von Estorff, O. and Gehlken, M. (2012). BEM and FEM results of displacements in a poroelastic column, *International Journal of Applied Mathematics and Computer Science* **22**(4): 883–896, DOI: 10.2478/v10006-012-0065-y.

Barboteu, M., Bartosz, K. and Kalita, P. (2013). An analytical and numerical approach to a bilateral contact problem with nonmonotone friction, *International Journal of Applied Mathematics and Computer Science* **23**(2): 263–276, DOI: 10.2478/amcs-2013-0020.

Csuhaj-Varjú, E. (2004). Grammar systems: A short survey, *Proceedings of Grammar Systems Week 2004, Budapest, Hungary*, pp. 141–157.

Csuhaj-Varjú, E., Dassow, J., Kelemen, J. and Paun, G. (1994). *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Topics in Computer Mathematics 8, Gordon and Breach Science Publishers, Yverdon.

Csuhaj-Varjú, E., Dassow, J. and Paun, G. (1993). Dynamically controlled cooperating/distributed grammar systems, *Information Sciences* **69**(1–2): 1–25.

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, Taylor & Franics Group, Boca Raton, FL/London/New York, NY.

Druskin, V., Knizhnerman, A. and Lee, P. (1999). New spectral Lanczos decomposition method for induction modeling in arbitrary 3-d geometry, *Geophysics* **64**(3): 701–706.

Flasiński, M. and Schaefer, R. (1996). Quasi context sensitive graph grammars as a formal model of FE mesh generation, *Computer-Assisted Mechanics and Engineering Science* **3**: 191–203.

Grabska, E. (1993). Theoretical concepts of graphical modeling, Part two: cp-graph grammars and languages, *Machine Graphics and Vision* **2**(2): 149–178.

Grabska, E. and Strug, B. (2005). Applying cooperating distributed graph grammars in computer aided design, *in* R. Wyrzykowski, J. Dongarra, N. Meyer and J. Waśniewski (Eds.), *Parallel Processing and Applied Mathematics*, Lecture Notes in Computer Science, Vol. 3911, Springer, Berlin/Heidelberg, pp. 567–574.

Hild, P. (2011). A sign preserving mixed finite element approximation for contact problems, *International Journal of Applied Mathematics and Computer Science* **21**(3): 487–498, DOI: 10.2478/v10006-011-0037-7.

Kelemen, J. (1991). Syntactical models of cooperating/distributed problem solving, *Journal of Experimental and Theoretical AI* **3**(1): 1–10.

Kukluk, J., Holder, L. and Cook, D. (2008). Inferring graph grammars by detecting overlap in frequent subgraphs, *International Journal of Applied Mathematics and Computer Science* **18**(2): 241–250, DOI: 10.2478/v10006-008-0022-y.

Martín-Vide, C. and Mitrana, V. (1998). Cooperation in contextual grammars, *Proceedings of the MFCS'98 Satellite Workshop on Grammar Systems, Brno, Czech Republic*, pp. 289–302.

Newman, G. and Alumbaugh, D. (2002). Three-dimensional induction logging problems, Part 2: A finite-difference solution, *Geophysics* **67**(2): 484–491.

Pardo, D., Demkowicz, L., Torres-Verdín, C. and Paszynski, M. (2006). Two-dimensional high-accuracy simulation of resistivity logging-while-drilling (LWD) measurements using a self-adaptive goal-oriented hp finite element method, *SIAM Journal on Applied Mathematics* **66**(6): 2085–2106.

Pardo, D., Demkowicz, L., Torres-Verdín, C. and Paszynski, M. (2007). A self-adaptive goal-oriented hp-finite element method with electromagnetic applications, Part II: Electrodynamics, *Computer Methods in Applied Mechanics and Engineering* **196**(37): 3585–3597.

Pardo, D., Torres-Verdín, C. and Paszynski, M. (2008). Simulations of 3d DC borehole resistivity measurements with a goal-oriented hp finite-element method, Part II: Through-casing resistivity instruments, *Computational Geosciences* **12**(1): 83–89.

Paszynska, A., Grabska, E. and Paszynski, M. (2012a). A graph grammar model of the hp adaptive three dimensional finite element method, Part I, *Fundamenta Informaticae* **114**(2): 149–182.

Paszynska, A., Grabska, E. and Paszynski, M. (2012b). A graph grammar model of the hp adaptive three dimensional finite element method, Part II, *Fundamenta Informaticae* **114**(2): 183–201.

Paszynska, A., Paszynski, M. and Grabska, A. (2008). Graph transformations for modeling hp-adaptive finite element method with triangular elements, *in* M. Bubak, G.D. Albada, J. Dongarra and P.M.A. Sloot (Eds.), *ICCS 2008*, Lecture Notes in Computer Science, Vol. 5103, Springer, Berlin/Heidelberg, pp. 604–613.

Paszynska, A., Paszynski, M. and Grabska, E. (2009). Graph transformations for modeling *hp*-adaptive finite element method with mixed triangular and rectangular elements, *in* G. Allen, J. Nabrzyski, E. Seidel, G.D. Albada, J. Dongarra and P.M.A. Sloot (Eds.), *ICCS 2009*, Lecture Notes in Computer Science, Vol. 5545, Springer, Berlin/Heidelberg, pp. 875–884.

Paszynski, M. (2009a). On the parallelization of self-adaptive *hp*-finite element methods, Part I: Composite programmable graph grammar model, *Fundamenta Informaticae* **93**(4): 411–434.

Paszynski, M. (2009b). On the parallelization of self-adaptive *hp*-finite element methods, Part II: Partitioning communication agglomeration mapping (PCAM) analysis, *Fundamenta Informaticae* **93**(4): 435–457.

Paszynski, M., Pardo, D. and Calo, V. (2013). A direct solver with reutilization of LU factorizations for *h*-adaptive finite element grids with point singularities, *Computers & Mathematics with Applications* **65**(8): 1140–1151.

Paszynski, M., Pardo, D. and Paszynska, A. (2011). Out-of-core multi-frontal solver for multi-physics *hp* adaptive problems, *Procedia Computer Science* **4**: 1788–1797.

Paszynski, M. and Schaefer, R. (2010). Graph grammar-driven parallel partial differential equation solver, *Computer-Assisted Mechanics and Engineering Science* **22**(9): 1063–1097.

Spicher, A., Michel, O. and Giavitto, J. (2010). Declarative mesh subdivision using topological rewriting in MGS, *Graph Transformations: 5th International Conference, ICGT 2010, Enschede, The Netherlands*, pp. 298–313.

Szymczak, A., Paszynska, A. and Paszynski, M. (2011). Anisotropic 2d mesh adaptation in *hp*-adaptive FEM, *Procedia Computer Science* **4**: 1818–1827.

Wang, T. and Fang, S. (2001). 3-d electromagnetic anisotropy modeling using finite differences, *Geophysics* **66**(5): 13861398.

Zhang, R., Mackie, L. and Madden, T. (1995). 3-d resistivity forward modeling and inversion using conjugate gradients, *Geophysics* **60**: 1312–1325.

Barbara Strug received her Ph.D. (2002) in computer science from the Institute of Fundamental Technological Research of the Polish Academy of the Sciences in Warsaw, Poland. She is currently an assistant professor at the Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University in Kraków, Poland. Her research interests include computer aided design, standard and distributed graph transformations, graph-based knowledge representation, graph patterns and graph similarity analysis.

Anna Paszyńska received her Ph.D. (2007) in computer science from the Institute of Fundamental Technological Research of the Polish Academy of the Sciences in Warsaw, Poland. She currently works as an assistant professor at the Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University in Kraków, Poland. Her research interests include evolutionary algorithms, graph grammar and computer aided design.

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

Ewa Grabska received her Ph.D. (1982) in mathematics from Jagiellonian University, Kraków, Poland, and the D.Sc. (1997) in computer science from the Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland. Since 2008, she has been a professor at the Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, and the head of the Department of Design and Computer Graphics at this faculty. Her research areas include computer aided design, graph transformations, visual communication, ontology, engineering, and computer games.

Received: 20 February 2013
Revised: 10 July 2013