Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 1485–1495

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Goal-Oriented Self-Adaptive *hp* Finite Element Simulation of 3D DC Borehole Resistivity Simulations

Victor M. Calo[a], David Pardo[b], Maciej R. Paszyński[c]

[a]*Department of Applied Mathematics, Computational Science, Earth Engineering, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudii Arabia*
[b]*Department of Applied Mathematics, Statistics, and Operational Research, Univsersity of the Basque Country and Ikerbasque, Bilbao, Spain*
[c]*Department of Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Kraków, Poland*

Abstract

In this paper we present a goal-oriented self-adaptive *hp* Finite Element Method (*hp*-FEM*)* with shared data structures and a parallel multi-frontal direct solver. The algorithm automatically generates (without any user interaction) a sequence of meshes delivering exponential convergence of a prescribed quantity of interest with respect to the number of degrees of freedom. The sequence of meshes is generated from a given initial mesh, by performing *h* (breaking elements into smaller elements), *p* (adjusting polynomial orders of approximation) or *hp* (both) refinements on the finite elements. The new parallel implementation utilizes a computational mesh shared between multiple processors. All computational algorithms, including automatic *hp* goal-oriented adaptivity and the solver work fully in parallel. We describe the parallel self-adaptive *hp*-FEM algorithm with shared computational domain, as well as its efficiency measurements. We apply the methodology described to the three-dimensional simulation of the borehole resistivity measurement of direct current through casing in the presence of invasion.

Keywords: hp-FEM, goal-oriented adaptivity, parallel data structures

1. Introduction

The self-adaptive *hp* Finite Element Method (*hp*-FEM) for two- and three-dimensional elliptic and Maxwell problems were designed and implemented by the group of Leszek Demkowicz [1, 2] at the University of Texas at Austin. The codes generate a sequence of *hp* meshes providing exponential convergence of the numerical solution with respect to the mesh size. The parallel version of the two- and three-dimensional algorithms have been designed and implemented based on the *distributed domain decomposition* paradigm, illustrated on the left panel of Fig. 1. [3, 4]. The main disadvantage of the distributed-domain-decomposition-based parallel code is the huge complexity of the mesh transformation algorithms executed over the computational mesh stored in distributed manner. There exist the following mesh regularity rules: 1) *the one irregularity rule*, preventing a finite element from being broken two consecutive times without first breaking larger adjacent elements, and 2) *the minimum rule,* which states that the order of approximation over a face must be the minimum of the corresponding orders of approximation from adjacent element interiors, and the order of approximation over an edge must be the minimum of the corresponding

doi:10.1016/j.procs.2011.04.161

orders of approximation from adjacent faces. The main technical difficulty in previous implementations was to maintain these mesh regularity rules over the computational mesh partitioned into sub-domains, e.g.,m a refinement performed over one sub-domain may require a sequence of additional refinements over adjacent elements, possibly located at adjacent sub-domains. A partial solution to the problem was the introduction of ghost elements in order to simplify mesh reconciliation algorithms [2, 4]. However, ghost elements increased the communication cost, especially after many refinements, since a layer of initial mesh elements, possibly broken into many smaller elements, had to be exchanged between adjacent sub-domains.

In this paper we propose an alternative parallelization technique, based on the *shared domain decomposition* paradigm, illustrated on the right panel in Fig. 1. The entire data structure with the computational mesh is stored on *every* processor. However, the computations performed over the mesh are shared between processors. It is done by assigning the so-called processor owners to particular mesh elements, and executing computations over these elements by assigned processors. This is usually performed by sharing the algorithm's loops by many processors, followed by `mpi_allreduce` call merging results.

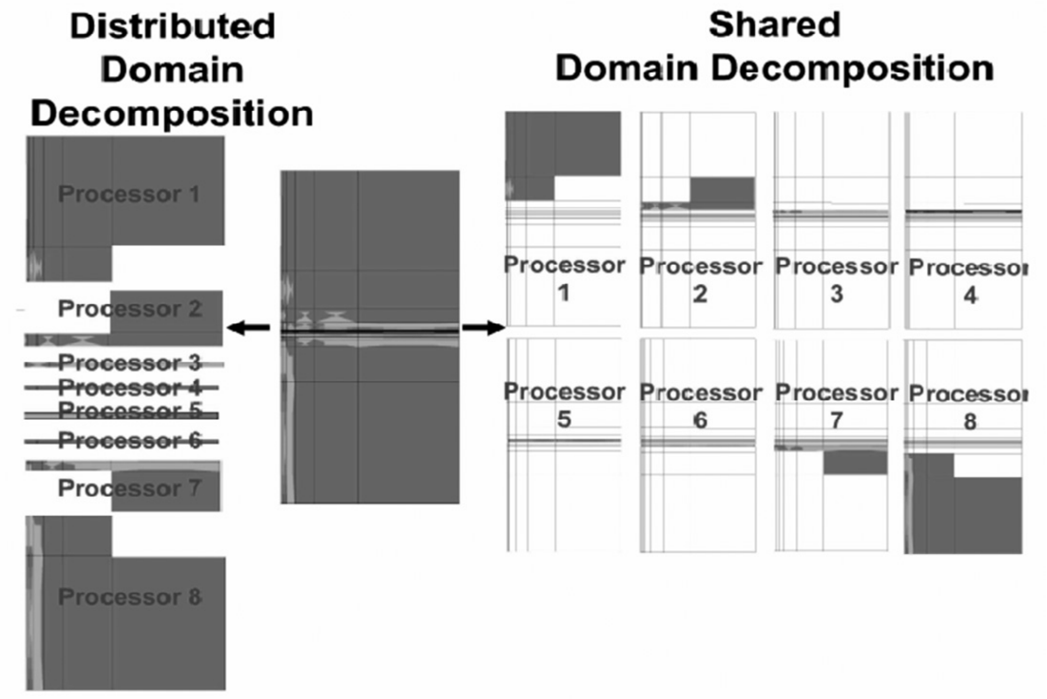

Figure 1. The shared-domain decomposition as opposed to the distributed-domain decomposition.

2. Automatic *hp*-Adaptivity

A general sequential algorithm for the fully automatic *hp* adaptation can be described is the following steps.

(1) Algorithm starts with the coarse initial mesh with uniform order of approximation.

(2) The computational problem is solved over the coarse mesh and the approximate solution u_{hp} is obtained.

(3) The coarse mesh is globally *hp*-refined in order to produce the fine mesh. It is done by breaking each finite element into four son elements and increasing the polynomial order of approximation by one. This will be the reference mesh used for calculation of the interpolation error over the coarse mesh.

(4) The computational problem is solved on the fine mesh and the approximate solution $u_{h/2,p+1}$ is obtained.

(5) As the relative error estimator for the coarse mesh, the difference (in H^1-seminorm) between the coarse and the fine mesh solutions is taken.
(6) The optimal refinements are selected based on the calculated error estimators for the subset of the coarse mesh elements with higher relative error estimators. For example, in 2D the selected elements are either broken into smaller son elements (this is so called *h*-refinement) isotropically (4 sons) or anisotropically (2 sons in the same direction) or the polynomial order of approximation is increased on element edges or interiors (this is so called *p*-refinement), or both. This is illustrated in Fig. 2.

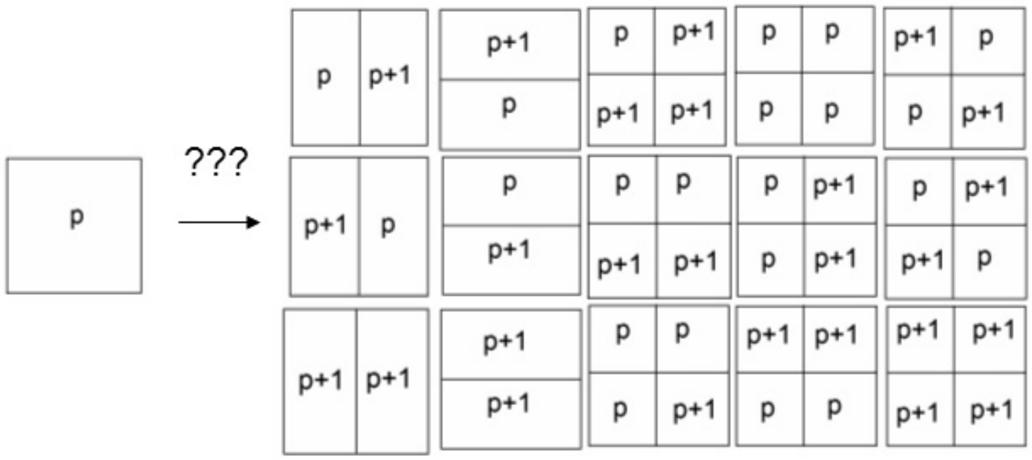

Fig. 2. Many possible refinements of a coarse mesh element.

The optimal refinements are selected independently over each coarse mesh element. It is done in a way to provide maximal error decrease rate given by:

$$rate = \max\left\{ \frac{\left\| u_{h/2,p+1} - u_{h,p} \right\|_1 - \left\| u_{h/2,p+1} - w \right\|_1}{\Delta nrdof} \right\} \tag{1}$$

where $\Delta nrdof$ is the number of added degrees of freedom during the considered refinement, w is the solution for proposed refinement strategy, obtained by utilizing the projection based interpolation [5] from the fine mesh solution $u_{h/2,p+1}$ into the considered refined element, $\left\| u_{h/2,p+1} - u_{h,p} \right\|$ is the relative error estimation over the current coarse mesh with respect the fine mesh and $\left\| u_{h/2,p+1} - w \right\|$ is the relative error estimation for the refinement strategy proposed for the coarse mesh element with respect to the fine mesh. Thus, we seek for a refinement that provides the best error decrease rate with a minimum increase in the number of degrees of freedom.
(7) The selected refinements are executed over the coarse mesh to obtain the new optimal mesh.
(8) The new optimal mesh becomes a coarse mesh for the next iteration, and the entire procedure is repeated as long as the global relative error estimation is larger than the required accuracy of the solution.

The algorithm is also illustrated in Fig. 3. The selection of the optimal refinements for the coarse mesh finite elements is actually performed in two steps, in order to limit the number of possibilities considered in point 6). Using 2d as an example, first, the optimal refinements are selected for finite element edges, and then the optimal refinements for element interiors are selected, with restriction to known optimal refinements for element edges. The relative error measurements over element edges are performed in the $H^{1/2}$ seminorm.

The above energy-norm based adaptive algorithm has been further generalized to the case of goal-oriented adaptivity. The necessary modifications included solving the so-called “dual” problem over the same coarse and fine grids, and estimate the goal-oriented errors as a combination of the solutions of both direct and dual problems. From the parallel data structures point-of-view, these modifications implied duplicating the number of degrees-of-freedom in order to accommodate solution of the dual problem.

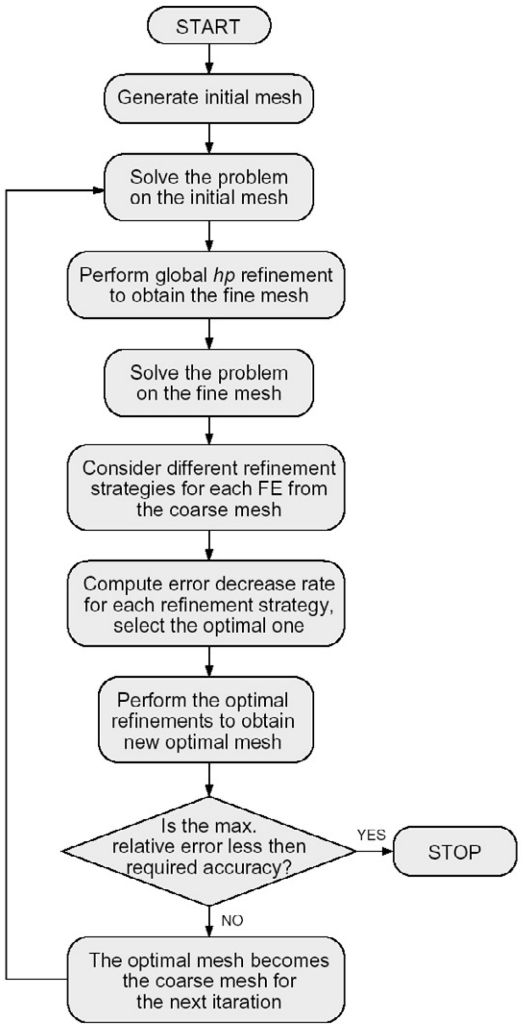
START
Generate initial mesh
Solve the problem on the initial mesh
Perform global hp refinement to obtain the fine mesh
Solve the problem on the fine mesh
Consider different refinement strategies for each FE from the coarse mesh
Compute error decrease rate for each refinement strategy, select the optimal one
Perform the optimal refinements to obtain new optimal mesh
Is the max. relative error less then required accuracy?
YES
STOP
NO
The optimal mesh becomes the coarse mesh for the next itaration

Fig. 3. The algorithm for the sequential self-adaptive *hp*-FEM.

3. Parallel fully automatic *hp*-Finite Element Method

In this section, we present the parallel version of the fully automatic goal-oriented *hp* adaptivity, implemented under the shared domain decomposition paradigm. The new parallel algorithm can be summarized in the following steps:

(1) The coarse initial mesh is generated on every processor. The initial mesh elements are assigned to different processors, by filling `processor_owner` attribute of the `element` object. It is performed either by interfacing with the ZOLTAN library [6], or by utilizing simple row-wise mesh partitioners for two dimensional meshes. The element's `processor_owner` attribute is filled on every processor, in other words, each processor knows processor owners of all elements. The element edges and vertices are assigned to processor owners. It is performed by browsing all elements and filling `processor_owners` lists located at element `node` or `vertex` objects.

(2) The additional data structure presented in Figure 4 is initialized. We utilize here the Unified Modelling Language notation [7]. The `element_refined` objects are created for each active finite element. The `middle_node` links are related with interior nodes of active finite elements, represented by `node` objects with `type='mdlq'`. The `edge_refined` objects are created for all active finite element edges. The `edge_node` links are related with element edge nodes, represented by `node` objects with `type='medg'`. Notice, that `element_refined` objects (related to active finite elements) do not correspond to `element` objects (related to the initial mesh elements only).

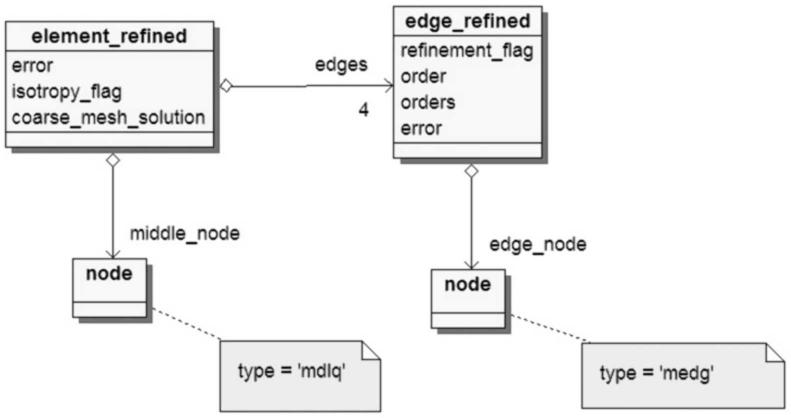

Figure 4. The UML diagram presenting the data structure managing mesh refinements.

(3) The computational problem is solved over the current coarse mesh, by utilizing multi-frontal parallel direct solver [8, 9]. Each processor stores the local solution vector at its active finite element `node` and `vertex` objects, in the `solution_d.o.f.` attribute. The coarse mesh solution d.o.f. are also recorded at `coarse_mesh_solution` arrays of `elements_refined` objects.

(4) The global *hp* refinement is executed over the coarse mesh in order to construct the reference fine mesh. This is performed by every processor over the entire data structure. Each finite element from the coarse mesh is partitioned into four new finite elements, and the polynomial order of approximation is uniformly raised by one. This is done by executing isotropic *h* refinement over each element interior `node` object, as well as refinement over each element edge `node` object. Also, the `order_of_approximation` attribute is increased for each active node (for each leaf `node` object).

(5) The `processor_owners` of newly created `node` and `vertex` objects are filled based on the information inherited from father `node` objects.

(6) The computational problem is solved again over the fine mesh by utilizing the multi-frontal parallel direct solver [8, 9]. Each processor stores the local solution vector at its active finite element `node` and `vertex` objects, in the `solution_d.o.f.` attribute. Notice that the coarse mesh solution is still stored at parent nodes as well as at `elements_refined` objects. For the case of goal-oriented adaptivity, we also solve for the "dual" problem.

(7) Each processor loops through its active elements and computes the relative error estimation over the element

$$\frac{\left\| u_{hp} - u_{h/2,p+1} \right\|_{H^1}}{\left\| u_{h/2,p+1} \right\|_{H^1}}, \tag{2}$$

with u_{hp} being the coarse mesh solution restored from the `element_refined` objects, and $u_{h/2,p+1}$ being the fine mesh solution restored from the `solution_d.o.f.` attribute of active finite element `node` and `vertex` objects. The relative error is stored in the `error` attribute of the `element_refined` objects. For the case of goal-oriented adaptivity, the relative error estimation over the element also incorporates terms corresponding to the solution of the dual problem.

(8) The maximum element relative error is computed, and elements with the relative error estimation larger than 33% of the maximum error are to be refined.

(9) For elements with strong gradient of the error in one direction, the `isotropy_flag` attribute of the `element_refined` object is set to enforce the element refinement in one direction.

(10) Different refinement strategies are considered for element edges, by utilizing the formula

$$\sum_K \frac{\left\| u_{h/2,p+1} - u_{h,p} \right\|_{1/2,K} \left\| \tilde{u}_{h/2,p+1} - \tilde{u}_{h,p} \right\|_{1/2,K} - \left\| u_{h/2,p+1} - w \right\|_{1/2,K} \left\| \tilde{u}_{h/2,p+1} - \tilde{w} \right\|_{1/2,K}}{\Delta nrdof} \tag{3}$$

with K denoting an element, u_{hp} the coarse mesh solution restored from the `element_refined` objects, $u_{h/2,p+1}$ the fine mesh solution restored from active finite element `node` and `vertex` objects, and w being the projection based interpolant of the fine mesh solution $u_{h/2,p+1}$ into the considered edge refinement. Tilde symbol denotes solution of the "dual" problem needed for goal-oriented adaptivity. The $H^{1/2}$ seminorm is utilized to measure the relative error over an element edge. The selected refinement is stored in `edge_refined` object. If an element edge is going to be *p* refined, the `ref_flag` attribute for the edge is set to 1, and the proposed order of approximation is stored at `order` attribute. If an element edge is going to be *h* refined, the `ref_flag` attribute for the edge is set to -1, and the proposed orders of approximation for son edges are stored at `orders` attribute array. These estimations are performed by every processor over active finite elements assigned to the processor. Thus, the optimal refinement information is stored in distributed manner in `element_refined` and `edge_refined` objects. These estimations are performed only for edges of elements with relative error estimation larger than 33% of the maximum relative error.

(11) The proposed refinement data (`ref_flag`, `order` and `orders` attributes of `edge_refined` object as well as `isotropy_flag` of `element_refined` object) are broadcasted to all processors.

(12) The fine mesh is deallocated and the coarse mesh is restored.

(13) The selected optimal refinements are executed for element edges. This is done by all processors over the entire data structure. It can be done, since we broadcasted the proposed refinement data. Some edges are *h* refined: one new `vertex` object and two new edge `node` objects are created are connected to the original edge `node`. The order of approximation for new `node` objects is taken from `orders` attribute array of `edge_refined` object. Some edges are *p* refined, and the new order of approximation is taken from `order` attribute of `edge_refined` object. Some edge refinements are modified based on the `isotropy_flag` from `element_refined` objects.

(14) Different element interior node refinements are considered for elements. This is done by utilizing the formula

$$\sum_K \frac{\left\| u_{h/2,p+1} - u_{h,p} \right\|_{1,K} \left\| \tilde{u}_{h/2,p+1} - \tilde{u}_{h,p} \right\|_{1,K} - \left\| u_{h/2,p+1} - w \right\|_{1,K} \left\| \tilde{u}_{h/2,p+1} - \tilde{w} \right\|_{1,K}}{\Delta nrdof} \quad (4)$$

with K denoting an element u_{hp} the coarse mesh solution restored from the `element_refined` objects, $u_{h/2,p+1}$ the fine mesh solution restored from active finite element `node` and `vertex` objects, and w being the projection based interpolant of the fine mesh solution $u_{h/2,p+1}$ into the considered element refinement. Tilde symbol denotes solution of the "dual" problem needed for goal-oriented adaptivity. The H^1 seminorm is utilized to measure the relative error over an element interior. These estimations are performed by every processor over active finite elements assigned to the processor. The selected refinement is stored in `element_refined` object. The type of refinement is coded within `refinement_flag`, and new orders of approximation for son nodes are coded within `orders` array. Thus, the optimal refinements information is stored in distributed manner in `element_refined` and `edge_refined` objects. These estimations are performed only for edges of elements with relative estimated error above 33% of the maximum relative error.

(15) The proposed interiors refinement data (`refinement_flag` and `orders` attributes of `element_refined` objects) are broadcasted to all processors.

(16) The selected optimal refinements are executed for element interiors. This is done by all processors over the entire data structure. Some elements are *h* refined: new edge and interior `node` objects and vertex objects (for isotropic *h* refinement) are created and connected to the original interior `node`. The order of approximation for new `node` objects is taken from `orders` attribute array of `element_refined` object. Some elements are *p* refined, and the new orders of approximation are taken from `orders` attribute of `element_refined` object.

(17) The minimum rule is enforced over the entire data structure: the order of approximation over element edges is set to be equal to the minimum of orders for adjacent element interiors. This is done by all processors over the entire data structure. Thus, an identical copy of the new optimal mesh is stored on every processor.

(18) The `element_refined` and `edge_refined` objects are deallocated.

(19) If the maximum error is still greater than the required accuracy of the solution, the new optimal mesh becomes a coarse mesh and the next iteration is executed.

4. Computational problem formulation

In this section we present exemplary parallel simulations for the 3D DC resistivity logging measurement simulation problem. The problem consists in solving the conductive media equation

$$\nabla \circ (\sigma \nabla u) = -\nabla \circ J^{imp} \quad (5)$$

in the 3D domain with different formation layers presented in Fig. 5. There is a logging tool with one transmitter and two receiver electrodes in the borehole. The tool is shifted along the borehole. The reflected waves are recorded by the receiver electrodes in order to determine location of the oil formation in the ground. Of particular interest to the oil industry are 3D simulations with deviated wells, where the angle between the borehole and formation layers is sharp ($\theta_0 \neq 90$). This 3D problem can be decomposed as a sequence of coupled 2D problems by considering the non-orthogonal system of coordinates presented in Fig. 7. Following [10], the variational formulation in the new system of coordinates consists in finding $u \in u_D + H_D^1(\Omega)$ such that:

$$\left\langle \frac{\partial u}{\partial \xi}, \hat{\sigma}\frac{\partial v}{\partial \xi} \right\rangle_{L^2(\Omega)} = \left\langle v, \hat{f} \right\rangle_{L^2(\Omega)} \quad \forall v \in H_D^1(\Omega) \tag{6}$$

where new electrical conductivity of the media $\hat{\sigma} := J^{-1}\sigma J^{-1^T}|J|$ and $\hat{f} := f|J|$ with $f = \nabla J^{imp}$ being the gradient of the impressed current, and

$$J = \frac{\partial(x_1, x_2, x_3)}{\partial(\zeta_1, \zeta_2, \zeta_3)} \tag{7}$$

stands for the Jacobian matrix of the change of variables from the Cartesian reference to non-orthogonal systems of coordinates, and $|J| = \det(J)$ is its determinant. We take Fourier series expansions in the azimuthal ζ_2 direction

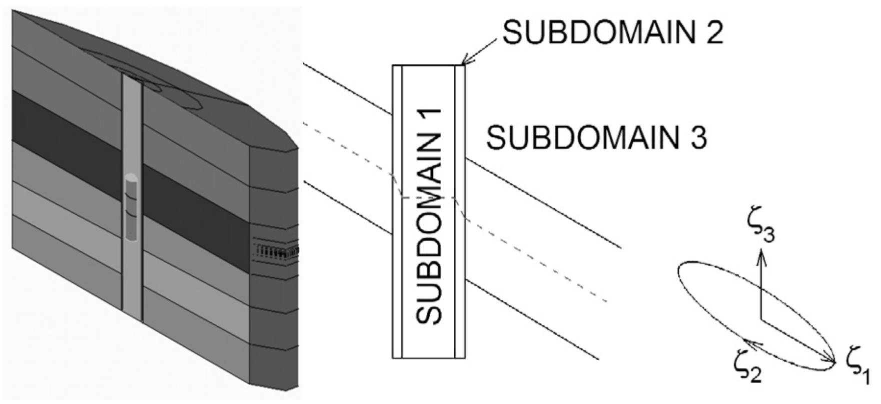

Fig. 5. The borehole, the tool with receiver and transmitter electrodes and the deviated formation layers. The utilized non-orthogonal system of coordinates.

$$u(\zeta_1,\zeta_2,\zeta_3) = \sum_{l=-\infty}^{l=+\infty} u_l(\zeta_1,\zeta_3)e^{jl\zeta_2}; \tag{8}$$

$$\sigma(\zeta_1,\zeta_2,\zeta_3) = \sum_{m=-\infty}^{m=+\infty} \sigma_m(\zeta_1,\zeta_3)e^{jm\zeta_2}; \tag{9}$$

$$f(\zeta_1,\zeta_2,\zeta_3) = \sum_{l=-\infty}^{l=+\infty} f_l(\zeta_1,\zeta_3)e^{jl\zeta_2}; \tag{10}$$

where $u_l = \frac{1}{2\Pi}\int_0^{2\Pi} ue^{-jl\zeta_2}\,d\zeta_2$, $\sigma_m = \frac{1}{2\Pi}\int_0^{2\Pi} \sigma e^{-jm\zeta_2}\,d\zeta_2$ and $f_l = \frac{1}{2\Pi}\int_0^{2\Pi} f e^{-jl\zeta_2}\,d\zeta_2$ 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 lth Fourier modal coefficients of the original vector or matrix components.

Using the Fourier series expansions we obtain the following variational formulation:

Find $F_l(u) \in F_l(u_D) + H_D^1(\Omega)$ such that:

$$\left\langle F_l\left(\frac{\partial u}{\partial \xi}\right), F_m(\hat{\sigma})\frac{\partial v}{\partial \xi} e^{j(l+m)\zeta_2} \right\rangle_{L^2(\Omega_{2D})} = \left\langle v, F_l\left(\hat{f}\right) e^{jl\zeta_2} \right\rangle_{L^2(\Omega_{2D})} \quad \forall v \in H_D^1(\Omega). \quad (11)$$

The Einstein's summation convention is applied with respect to $-\infty \le l, m \le \infty$. We select a mono-modal test function $v = v_k e^{jk\zeta_2}$. Thanks to the orthogonality of the Fourier modes in L^2, the variational problem defined in Eq. (11) reduces to

Find $F_l(u) \in F_l(u_D) + H_D^1(\Omega)$ such that:

$$\sum_{n=k-2}^{n=k+2} \left\langle F_l\left(\frac{\partial u}{\partial \xi}\right), F_{k-l}(\hat{\sigma}) F_l\left(\frac{\partial v}{\partial \xi}\right) \right\rangle_{L^2(\Omega_{2D})} = \left\langle F_k(v), F_k\left(\hat{f}\right) \right\rangle_{L^2(\Omega_{2D})} \quad \forall F_k(v) \in H_D^1(\Omega_{2D}) \quad (12)$$

since five Fourier modes are enough to represent exactly the new material coefficients. We refer to [10] for more details.

5. Numerical results

We conclude the presentation by describing a numerical example of parallel computations on the lonestar [11] linux cluster of the 3D DC borehole resistivity measurement simulations in deviated wells. Figure 6 presents the logging curves for the resistivity logging measurement simulations in zero, 30, 45 and 60 degrees deviated wells. The results are an extension of [12] and they include also the case in presence of 10 cm and 50 cm invasion.

The problem geometry can be described by using cylindrical coordinates (ρ, φ, z).

a) Four (one current and three voltage) 2 × 5-cm ring electrodes located 8 cm from the axis of symmetry and moving along the vertical direction (*z* axis). Voltage (collector) electrodes are located 100, 125, and 150 cm above the current (emitter) electrode, respectively.
b) Borehole: a cylinder Ω_A of radius 10 cm surrounding the axis of symmetry $\Omega_A = \{(x, \varphi, z): \rho \le 10\,\text{cm}\}$ with resistivity $R = 0.1\,\Omega \cdot m$.
c) Casing: a pipe (cylindrical shell) Ω_B of thickness 1.27 cm surrounding the axis of symmetry $\Omega_B = \{(x, \varphi, z): 10\,\text{cm} \le \rho \le 11.27\,\text{cm}\}$, with resistivity $R = 10^{-5}\,\Omega \cdot m$.
d) Formation material 1: a subdomain Ω_C defined by $\Omega_C = \{(x, \varphi, z): \rho > 11.27\,\text{cm}, 0\,\text{cm} \le z \le 100\,\text{cm}\}$ with resistivity $R = 10^4\,\Omega \cdot m$.
e) Formation material 2: a subdomain Ω_D defined by $\Omega_D = \{(x, \varphi, z): \rho > 11.27\,\text{cm}, -50\,\text{cm} \le z \le 0\,\text{cm}\}$ with resistivity $R = 0.01\,\Omega \cdot m$
f) Formation material 3: a subdomain Ω_E defined by $\Omega_E = \{(x, \varphi, z): \rho > 11.27\,\text{cm or } z > 100\,\text{cm}\}$ with resistivity $R = 5\,\Omega \cdot m$.

Notice that each point on the plot requires a solution of the 3D problem, and the point corresponds to the value of the solution at receiver electrode, computed with a very high accuracy thanks to the goal-oriented *hp* adaptive methodology. The logging tool has been shifted along the borehole, from the relative position of 2meters down to -2meters, and we perform a new simulation for each position of the logging tool. We refer to [12] for more details.

Figure 7 presentes the scalability tests of the parallel solver algorithm. The parallel version of the solver has been tested on the two dimensional mesh with 576 finite elements with uniform polynomial order of approximation *p=2* and 10 Fourier modes utilized to approximate the solution in its third direction (thus, total number of d.o.f. per node is equal to 20). We refer to [8, 9] for more details on the solver algorithm. The total number of d.o.f. over the entire mesh is 210,370. The parallel solver reduces the execution time from 582 seconds on a single processor to 6.5 seconds on 128 processors. Notice that the scalability test corresponds to a single position of a receiver antenna.

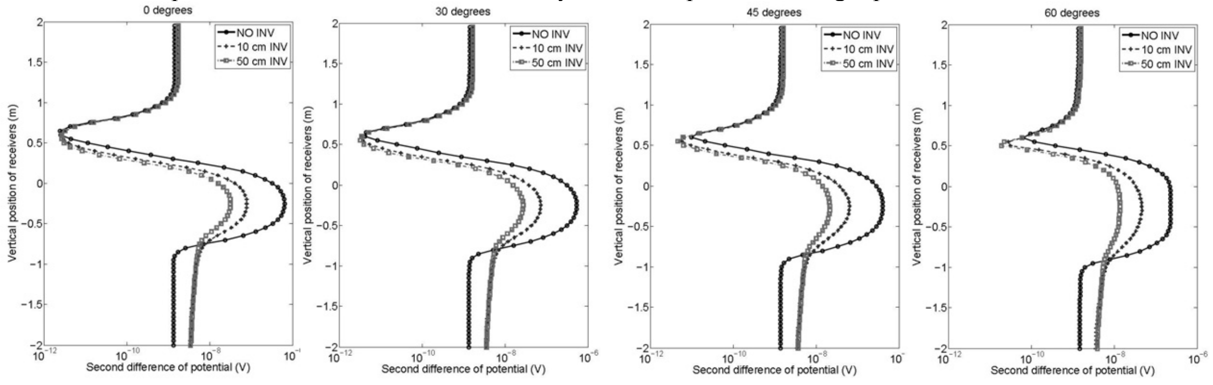

Fig. 6. Logging curves for through casing resistivity logging measurement simulations in deviated wells.

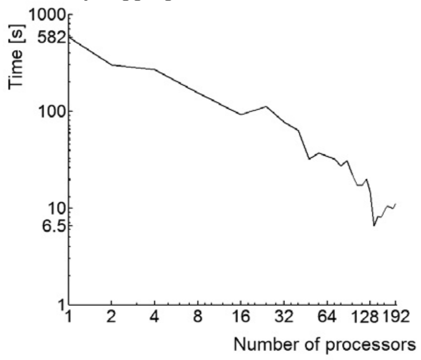

Fig. 7. Parallel solver execution time [s] up to 192 processors. Logarithmic scales are utilized on both axes.

6. Conclusions

In this paper we presented the parallel goal-oriented self-adaptive *hp* finite element method platform for resistivity logging simulations. We solved the through-casing resistivity logging simulation problem in presence of invasion. This is the first existing simulation of the through-casing resistivity logging simulation in the presence of invasion. This is due to large numerical constrast (6 orders of magnitude from 10^{-12} to 10^{-6}) that can be resolved only by utilizing the *hp* adaptive goal oriented metodology. The developed parallel version of the solver algorithm with shared data structure allows for a fast solution, and the solver algorithm scalles well up to 128 processors.

Acknowledgements

The work reported in this paper has been partially supported by the Polish MNiSW grant no. NN 519 447739, and the grant from the Spanish Ministry of Sciences MTM2010-16511.

References

1. Demkowicz L., 2006: *Computing with hp-Adaptive Finite Elements, Vol. I. One and Two Dimensional Elliptic and Maxwell Problems*, Chapmann & Hall / CRC Press.

2. Demkowicz L., Rachowicz W., Pardo D., Paszyński M., Kurtz J., Zdunek A., 2007: *Computing with hp-Finite Elements. Volume II*, Chapmann & Hall / CRC Press.

3. Paszyński M., Kurtz J., Demkowicz L., 2006: *Parallel Fully Automatic hp-Adaptive 2D Finite Element Package*, Computer Methods in Applied Mechanics and Engineering, 195, 7-8, 711-741.

4. Paszyński M., Demkowicz L., 2006: *Parallel Fully Automatic hp-Adaptive 3D Finite Element Package*, Computers and Mathematics with Applications, 22, 3-4, 255-276.

5. Demkowicz, L. 2004: *Projection-based interpolation*, ICES Report 04-03, The University of Texas in Austin.

6. ZOLTAN: *Data-Management Services for Parallel Applications*, http://www.cs.sandia.gov/Zoltan

7. Booch G., Rumbaugh J., Jacobson I., 1994: *The Unified Modeling Language User Guide*, Addison-Wesley, 1st edition.

8. Paszyński M., Pardo D., Torres-Verdin C., Demkowicz L., Calo V.M., 2010: *A Parallel Direct Solver for the Self-Adaptive hp Finite Element Method*, Journal of Parallel and Distributed Computing, 70, 270-281.

9. Paszyński M., Schaefer R., 2010: *Graph grammar-driven parallel partial differential equation solver*, Concurrency & Computations, Practise & Experience, 22, 9, 1063-1097

10. Pardo D., Calo V.M., Torres-Verdin C., Nam M.J., 2007: *Fourier Series Expansion in a Non-Orthogonal System of Coordinates for Simulation of 3D Borehole Resistivity Measurements. Part I: DC*, Computer Methods in Applied Mechanics and Engineering, 197, 1-3, 1906-1925.

11. Lonestar Cluster User's Manual *http://www.tacc.utexas.edu/services/userguides/lonestar/*

12. Pardo D., Torres-Verdin C., Paszyński M., 2008: *Simulations of 3D DC borehole resistivity measurements with a goal-oriented hp finite-element-method. Part II: thorugh-casing resistivity instruments,* Computational Geosciences 12, 83-89.