Reutilization of Partial LU Factorizations for Self-adaptive *hp* Finite Element Method Solver

Maciej Paszynski and Robert Schaefer

Department of Computer Science
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
`paszynsk,schaefer@agh.edu.pl`
`http://home.agh.edu.pl/~paszynsk`

Abstract. The paper presents theoretical analysis of the extension of the new direct solver dedicated for the fully automatic *hp* adaptive Finite Element Method. The self-adaptive *hp*-FEM generates in a fully automatic mode (without any user interaction) a sequence of meshes delivering exponential convergence of the numerical error with respect to the mesh size. The consecutive meshes are obtained by performing h, p or *hp* refinements. The proposed solver constructs an initial elimination tree based on the nested dissection algorithm executed over the initial mesh. The constructed elimination tree is updated each time the mesh is refined, by adding the elimination sub-tree related to the executed refinement. We propose a new strategy for reutilization of partial LU factorizations computed by the direct solver on the previous mesh, when solving a consecutive mesh from the sequence. We show that the number of LU factorizations that must be recomputed is linearly proportional to the number of singularities in the problem.

1 Motivation and the Basic Idea of Solution

The paper presents theoretical analysis of the extension of the sequential and parallel solvers [1], [2] dedicated for the self-adaptive *hp* Finite Element Method [3], [4], [5]. The self-adaptive *hp*-FEM generates a sequence of approximation spaces delivering exponential convergence of the numerical error of the resulting approximation of the variational problem under consideration. The exponential convergence of the error is obtained with respect to the dimension of the approximation space. The self-adaptive *hp*-FEM starts from an initial approximation space, constructed by utilizing a given uniform initial finite element mesh. The first order polynomial basis function ("pyramids") are related to vertices of the mesh, and the higher order polynomial basis functions are related to finite element edges and interiors [3]. The consecutive spaces from the produced sequence are obtained by performing so-called h or p refinements. The h refinement consists in breaking selected finite element into new son-elements, and adding new basis functions related to just created elements. The p refinement consists in adding higher order basis function associated with selected element edges or interiors. The refinements performed to improve the quality of the approximation

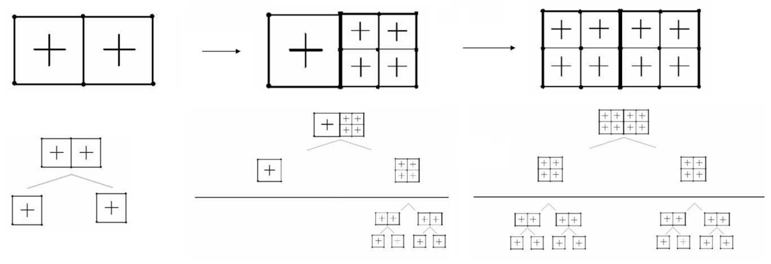

Fig. 1. Updating of the elimination tree when the mesh is h refined

space are selected by utilizing knowledge driven algorithm [6] based on the graph grammar formalism.

An efficient solver must be utilized to compute coefficients of the projection of the considered weak (variational) problem solution onto the current approximation space. The coefficients are called *degrees of freedom* (d.o.f.). These coefficients, denoted by u_{hp}^{i}, are computed by solving the system of equations

$$\sum_{i=1}^{dim} u_{hp}^{i} b\left(e_i, e_j\right) = l\left(e_j\right) \quad \forall j = 1, ..., dim \ , \tag{1}$$

where *dim* denotes the dimension of the approximation space (number of the basis functions), $\{e_k\}_{k=1}^{dim}$ denote the basis functions and $b\left(e_i, e_j\right)$ and $l\left(e_j\right)$ are matrix and right-hand-side vector entries obtained by computing some integrals resulting from the considered problem.

Here we present a short description of direct solvers utilized by FEM. The *frontal solver* browses finite elements in the order prescribed by the user, aggregates d.o.f. to the so-called frontal matrix. Based on the elements connectivity information it recognizes fully assembled degrees of freedom and eliminates them from the frontal matrix [7]. This is done to keep the size of the frontal matrix as small as possible. The key for efficient work of the frontal solver is the optimal ordering of finite elements. The *multi-frontal solver* constructs the d.o.f. connectivity tree based on analysis of the geometry of computational domain [7]. The frontal elimination pattern is utilized on every tree branch. Finite elements are joined into pairs and d.o.f. are assembled into frontal matrix associated with the branch. The process is repeated until the root of the assembly tree is reached. Finally, the common dense problem is solved and partial backward substitutions are recursively executed on the assembly tree. The *sub-structuring method solver* is a parallel solver working over a computational domain partitioned into multiple sub-domains [8]. First, the sub-domains internal d.o.f. are eliminated with respect to the interface d.o.f. Second, the interface problem is solved. Finally, the internal problems are solved by executing backward substitution on each sub-domain. This can be done by performing frontal decomposition on each sub-domain, and then solving the interface problem by a sequential frontal solver (this method is called the *multiple fronts solver* [9]). The better method is to

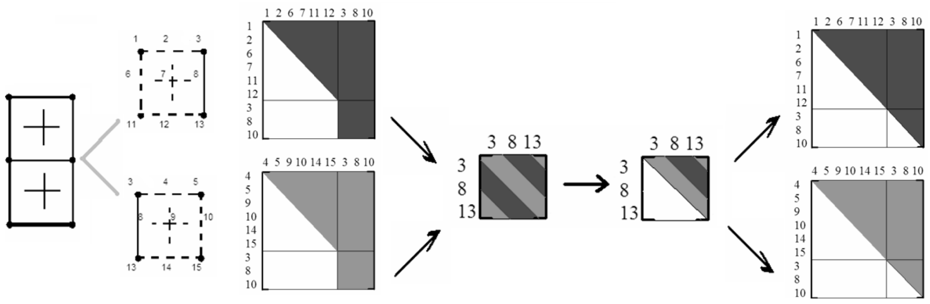

Fig. 2. Elimination tree for simple two finite elements mesh. Fully aggregated degrees of freedom for element interiors are eliminated in parallel, the resulting Schur complement contributions are added, and common interface problem is finally solved. The process is followed by performing recursive backward substitutions (not presented in the picture).

solve the interface problem also by a parallel solver (this is called the *direct sub-structuring method solver*). The parallel implementation of the multi-frontal solver is called the *sparse direct method solver*. The MUlti frontal Massively Parallel Solver (MUMPS) [10] is an example of such a solver.

A new efficient sequential and parallel solver for self-adaptive *hp*-FEM has been designed [1], [2], utilizing elimination tree constructed base on the history of mesh refinements. The elimination tree for the initial mesh is created by utilizing nested dissection algorithm. The exemplary two finite elements mesh with its elimination tree is presented on the first panel in Fig. 1. Each time decision about mesh refinement is made, the elimination tree is dynamically expanded by adding sub-tree related to the performed refinements. The example of two *h* refinements performed on the initial mesh with resulting expanding of the elimination tree is presented in Fig. 1. Thus, we can distinguish two levels on the elimination tree. The first level is related to the initial mesh elements, and the second level is related to refinements performed over the initial mesh.

The following observation is the key idea of the designed solver [1], [6]. The integral $b(e_i, e_j)$ is non-zero only if intersection of supports of e_i and e_j is not empty. The support of a vertex basis function spreads over finite elements having the vertex, the support of an element edge basis function spreads over two finite elements adjacent to the edge, and finally the support of an element interior basis function spreads only over the element. Thus, the integral $b(e_i, e_j)$ is zero if basis functions are related to distanced elements. The solver constructs first partially aggregated sub-matrices related to single finite elements, then it eliminates these entries that have already been fully assembled, and then it recursively merges resulting sub-matrices and eliminates fully assembled entries until it reaches the top of the elimination tree. Finally, it executes recursive backward substitutions, from the root of the tree down to the leaves. The exemplary execution of the solver on the two elements initial mesh from Fig. 1 is presented in Fig. 2.

The resulting LU factorizations computed at every node of the elimination tree can be stored at tree nodes for further reutilization. Each time the mesh

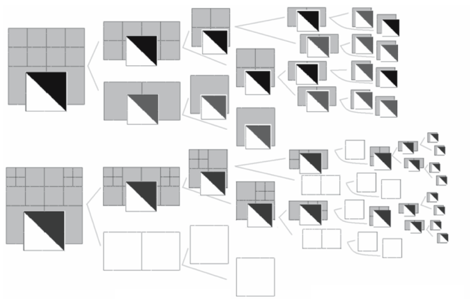

Fig. 3. The problem is solved over the first mesh. All LU factorizations (black and grey) are computed. Then, the mesh is refined, and the problem is solved again. Grey LU factorizations are reutilized from the previous mesh, but all brown LU factorizations must be recomputed. Black LU factorizations from previous mesh are deleted.

is refined, the LU factorizations from the unrefined parts of the mesh can be reutilized. There is a need to recompute LU factorization over the refined elements, as well as on the whole path from any refined leaf up to the root of the elimination tree. The example of the reutilization of partial LU factorizations after performing two local refinements is presented in Fig. 3.

2 Theoretical Analysis of the Solver Efficiency

We start this section with the sketch of the recursive solver algorithm, with reutilizations of LU factorizations.

```
matrix function recursive_solver(tree_node)
if (tree_node has no son nodes) then
  eliminate leaf element stiffness matrix internal nodes;
  store Schur complement sub-matrix at tree_node;
  return (Schur complement sub-matrix);
else if (tree_node has son nodes) then
  do (for each tree_node_son)
    if (sub-tree has been refined) then
      son_matrix = recursive_solver(tree_node_son);
    else
      get the Schur complement sub-matrix from tree_node_son;
    endif
    merge son_matrix into new_matrix;
  enddo
  decide which unknowns of new_matrix can be eliminated;
  perform partial forward elimination on new_matrix;
  store Schur complement sub-matrix at tree_node;
  return (Schur complement sub-matrix);
endif
```

Computational Complexity of the Sequential, Recursive Solver Without Reutilization of LU Factorizations. Let us estimate first the number of operations performed by a sequential recursive solver during forward elimination over a square shape 2D finite element mesh with $N = 2^n \times 2^n$ finite elements.

The order of approximation in the interior of the element is assumed to be equal to (p_1, p_2). The orders of approximation on element edges are assumed to be equal to the corresponding orders in the interior. From this assumption it follows that there are 2 faces with orders p_1 and 2 faces with orders p_2. The total number of d.o.f. in such an element is $nrdof = (p_1+1)(p_2+1) = O(p_1 p_2)$. To estimate the efficiency of the sequential solver, we assume that $p_1 = p_2 = p$, e.g. by taking $p = max\{p_1, p_2\}$. Thus, the total number of d.o.f. satisfies $nrdof = (p+1)^2 = O(p^2)$, while the number of interior d.o.f. can be evaluated as $interior\ nrdof = (p-1)^2 = O(p^2)$, and the number of interface d.o.f. satisfies $interface\ nrdof = 4p^2 = O(p^2)$.

The recursive solver eliminates d.o.f. related to elements interiors. The computational complexity of this step is $2^{2n} \times O(p^6)$ since there are 2^{2n} such finite elements and the internal d.o.f. elimination cost is $O(p^6)$ on every element.

Then, the solver joints elements into pairs, and eliminates d.o.f. related to common edges. The computational complexity of this operation is $2^{2n-1} \times ((2+4+1) \times p)^2 \times (2+4) \times p$ since there are 2^{2n-1} such pairs of elements, and there are 7 total edges within a pair, and only one edge is eliminated.

In the next step elements are joint into sets of four, and d.o.f. related to two common edges are eliminated. The computational complexity of this step is $2^{2n-2} \times ((4 \times 2 + 2) \times p)^2 \times (4 \times 2) \times p$ since there are 2^{2n-2} such sets of elements, and there are 10 edges in every set, and only 2 edges are eliminated.

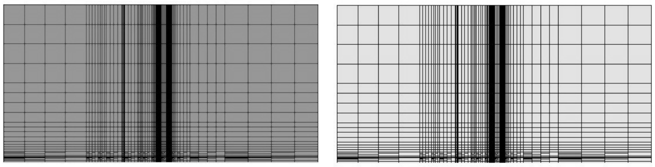

Fig. 4. Two tested meshes with uniform $p = 4$ and $p = 5$

The process is repeated until we reach the root of the elimination tree. The total computational complexity of this process is

$$2^{2n} \times p^6 + 2^{2n-1} \times (2+4+1)^2 p^2 \times (2+4) \times p +$$
$$\sum_{k=1,\ldots,n} \Big[2^{2n-2k-1} \left(2 \times 2^{k+1} + 2 \times 2^k + 2^k\right)^2 p^2 \left(2 \times 2^{k+1} + 2 \times 2^k\right) p +$$
$$\left(2^{2n-2k} \left(2 \times 2^k + 2 \times 2^k + 2^k\right)^2 p^2 \left(2 \times 2^k + 2 \times 2^k\right) p\right)\Big] \ .$$

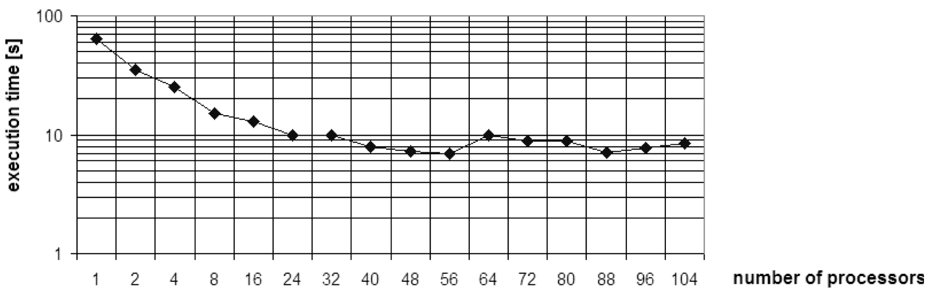

Fig. 5. The execution time of the parallel solver over the second tested mesh

This can be estimated by utilizing the sum of the geometrical series as

$$T_1 = O\left(2^{2n}p^6\right) + O\left(2^{2n-1}p^3\right) + O\left(\sum_{k=1,\ldots,n} 2^{2n+k+5}p^3\right)$$
$$= O\left(2^{2n}p^6 + \left(2^{2n-1} + 2^{3n+6} - 2^{2n+4}\right)p^3\right) = O(2^{2n}p^6 + 2^{3n}p^3 + 2^{2n}p^3) \ . \quad (2)$$

Computational Complexity of the Sequential Solver With Reutilization of LU Factorizations. In this section we perform the same analysis of the computational complexity like in the previous section, but this time we assume that the problem over the computational mesh has been already solved, and only one element has been h refined in the direction of a mesh corner singularity. In this case, there is a need to compute all LU factorizations related to the elimination sub-tree associated with broken corner element. It is also necessary to recompute all LU factorizations on the single path from the refined element (represented by a leaf in the original elimination tree) up to the root of the tree. The computational complexity over the broken element is

$$4 \times p^6 + 2 \times (2+4+1)^2 p^2 \times (2+4) \times p + (4*2+2)^2 p^2 \times (4*2) \times p \ , \quad (3)$$

since there are 4 element interiors, two single common edges and 1 twofold edge. The computational complexity of the recomputation of the whole path from the refined leaf up to the elimination tree root can be estimated by utilizing equation (2) with the correction that there is only one set of elements on every level of the tree, and without the leaf element computations, already estimated in (3).

$$\begin{aligned} &(2+4+1)^2 p^2 \times (2+4) \times p + \\ \sum_{k=1,\ldots,n} &\left[\left(2 \times 2^{k+1} + 2 \times 2^k + 2^k\right)^2 p^2 \left(2 \times 2^{k+1} + 2 \times 2^k\right) p + \right. \\ &\left. \left(\left(2 \times 2^k + 2 \times 2^k + 2^k\right)^2 p^2 \left(2 \times 2^k + 2 \times 2^k\right) p\right)\right] \ . \quad (4) \end{aligned}$$

Table 1. Execution time at different elimination tree nodes on two tested meshes

		First mesh		Second mesh	
Tree level	Nodes number	min time [s]	max time [s]	min time [s]	max time [s]
1	1	-	0.115	-	0.212
2	2	0.854	0.883	1.631	1.674
3	4	0.864	2.406	1.617	4.625
4	8	0.828	2.542	1.675	4.535
5	16	0.904	2.750	1.621	4.686
6	32	0.049	0.230	1.606	4.763
7	64	$< 10^{-2}$	$< 10^{-2}$	$< 10^{-2}$	0.110
8-14	128-9216	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$

The total computational complexity of the solver reutilizing LU factorization is equal to the sum of (3) and (4), that is

$$\begin{aligned} T_1^1 &= O\left(p^6\right) + O\left(p^3\right) + O\left(\sum_{k=1,\dots,n} 2^{3k+6}p^3\right) \\ &= O\left(p^6 + \left(1 + 2^{3n+6} - 2^6\right)p^3\right) = O(p^6 + 2^{3n}p^3) \ . \end{aligned} \qquad (5)$$

In the case of multiple refined leaves, the pessimistic estimation is that each leaf will generate a separate path to be totally recomputed. Thus, the total computational complexity with r refined leafs (resulting from $\frac{r}{4}$ singularities) is

$$T_1^r = O\left(rp^6 + \left(r + r2^{3n+6} - r2^6\right)p^3\right) = O(rp^6 + r2^{3n}p^3) \ . \qquad (6)$$

We conclude this section with the comparison of the execution times of the sequential solver with reutilization of LU factorization with respect to the sequential solver without the reutilization

$$\frac{T_1}{T_1^r} = O\left(\frac{2^{2n}}{r}\right) = O\left(\frac{N}{r}\right) \ . \qquad (7)$$

The solver with reutilization of partial LU factorizations is $O\left(\frac{N}{r}\right)$ times faster.

Complexity of the Parallel Solver Without Reutilization of LU Factorizations. The parallel version of the solver exchanges the partially aggregated matrices between the same level nodes [1]. Leaves of the elimination tree are assigned to different processors. When traveling up the elimination tree, the local Schur complements are sent from the second children node to the first one (to the first processor in every set). To estimate the computational complexity of the parallel recursive solve, we assume that the number of processors is $P = 2^{2m}$.

Each processor is responsible for its part of the mesh, with 2^{2n-2m} finite elements. Thus, each processor performs

$$O(2^{2(n-m)}p^6 + 2^{3(n-m)}p^3) \qquad (8)$$

operations on its part of the mesh. After this step, all computations over the elimination tree are performed fully in parallel:

$$\sum_{k=m+1,\dots,n}\Big[\left(2\times 2^{k+1}+2\times 2^{k}+2^{k}\right)^{2}p^{2}\left(2\times 2^{k+1}+2\times 2^{k}\right)p+$$
$$\Big(\left(2\times 2^{k}+2\times 2^{k}+2^{k}\right)^{2}p^{2}\left(2\times 2^{k}+2\times 2^{k}\right)p\Big)\Big]$$
$$=O(p^{3}\sum_{k=m+1,n}2^{2k})=O(p^{3}\sum_{k=1,n-m}2^{2(m+k)})=O(2^{2(n-m)}p^{3})\ . \tag{9}$$

The communication complexity involves $2(n-m+1)$ parallel point to point communications where sub-matrices related to local Schur complements are exchanged between pairs of tree nodes. The communication complexity is then

$$\sum_{k=m+1,n}2\times(2^{k}\times p)^{2}=O(p^{2}\sum_{k=1,n-m}2^{(m+k)})=O(2^{2(n-m)}p^{2}) \tag{10}$$

since the size of every sub-matrix is $2^k \times p$. The total complexity of the parallel solver without reutilization of the LU factorizations is then

$$T_P=(2^{2(n-m)}p^{6}+2^{3(n-m)}p^{3}+2^{2(n-m)}p^{3})\times t_{comp}+2^{2(n-m)}p^{2}\times t_{comm} \tag{11}$$

with $P = 2^{2m}$ the number of processor, and p the order of approximation.

Complexity of the Parallel Solver With Reutilization of LU Factorizations. In the case of the parallelization of the reutilization, the maximum number of processors that can be utilized is equal to r, the number of elements refined within the actual mesh. Each refinement requires the recomputation of the whole path from the refined leaf up to the tree root, which is a purely sequential. If the number of processors $P = 2^{2m}$is larger or equal to the number of executed refinements $2^{2m} \geq r$, then the total computational complexity can be roughly estimated as parallel execution of r paths from a leaf to the root of the tree, which is equal to (5). The communication complexity remains unchanged, since there is still a need to exchange the LU factorization, even if they are taken from local tree nodes. Thus the communication complexity is equal to (10). The total complexity of the parallel solver with reutilization of LU factorizations is

$$T_P^r=(p^{6}+2^{3n}p^{3})\times t_{comp}+2^{2(n-m)}p^{2}\times t_{comm}\ . \tag{12}$$

This is the "best parallel time" that can be obtain by the parallel solver with reutilization of partial LU factorizations, under the assumption that we have enough available processors ($P = 2^{2m} \geq r$). In other words, it is not possible to utilize more processors then number of refined elements r. We can compare the execution time of the parallel solver with reutilization to the parallel solver without the reutilization (as usually under the assumption that we have enough processors $P = 2^{2m} \geq r$).

$$\frac{T_P}{T_P^r}=O\left(2^{2(n-m)}\right)=O\left(\frac{N}{2^{2m}}\right)=O\left(\frac{N}{P}\right)\leq O\left(\frac{N}{r}\right)\ . \tag{13}$$

The parallel solver with reutilization is $O\left(\frac{N}{r}\right)$ times faster than the parallel solver without the reutilization.

3 Test Results

We conclude the presentation with two numerical experiments, presented in Fig. 4. The goal of these experiments is to illustrate the limitation of the scalability of the solver by the sequential part of the algorithm - the longest path from the root of the elimination tree down to the deepest leaf. For more numerical experiments executed for much larger problems, with more detailed discussion on the performance of the solver, as well as for the detailed comparison with the MUMPS solver, we refer to [1]. Both numerical experiments have been performed for the 3D Direct Current (DC) borehole resistivity measurement simulations [11]. The 3D problem has been reduced to 2D by utilizing the Fourier series expansions in the non-orthogonal system of coordinates. We refer to [11] for the detailed problem formulation. The first mesh contains 9216 finite elements with polynomial order of approximation $p = 4$, and $148,257$ d.o.f. The second mesh contains 9216 finite elements with polynomial order of approximation $p = 5$, and $231,401$ d.o.f. Both meshes have been obtained by performing two global *hp* refinements from the initial mesh with $32 \times 18 = 576$ finite elements with polynomial order of approximation $p = 2$ or $p = 3$, respectively. There are necessary 10 nested dissection cross-sections of the initial mesh, since $32 \times 18 \leq 2^5 \times 2^5$. Thus, the depth of the initial elimination tree is 10. Each global *hp* refinement consists in breaking each finite element into 4 son elements and increasing polynomial order of approximation by 1. Thus, each global *hp* refinement adds 2 levels to the elimination tree, so the total number of levels in the elimination tree is 14. Table 1 contains the total number of nodes at given elimination tree level, as well as the minimum and maximum Schur complement computation times for nodes located at given level of the elimination tree. The time of computing the entire path of partial LU factorization from a tree leaf up to the elimination tree root varies from 4 sec. to 9 sec. on the first mesh and from about 10 sec. up to 17 sec. on the second mesh. The execution time of the sequential solver with reutilization of LU factorizations over r times refined mesh will be within $(4 \times r, 9 \times r)$ sec. over the first and $(10 \times r, 17 \times r)$ sec. over the second mesh. The execution time of the parallel solver with reutilization of LU factorizations over r times refined mesh will be within $(4, 9)$ sec. over the first and $(10, 17)$ sec. over the second mesh, if there are more processors than refined elements. We present also in Fig. 5 the execution time of the parallel solver over the first mesh with $N = 231,401$ unknowns, for increasing number of processors. We observe that the parallel solver execution time is limited by the maximum time required to solve the entire path, which is about 9 second in this case.

4 Conclusions

We proposed a new algorithm for the sequential and parallel solver, that allows for significant reduction of the solver execution time over a sequence of meshes generated by the self-adaptive *hp*-FEM. The solver reutilized partial LU factorizations computed in previous iterations over unrefined parts of the mesh.

Every local h refinements requires a sequential recomputation of all LU factorization on a path from the refined leaf up to the root of the elimination tree. The maximum number of processors that can be utilized by the parallel solver with reutilization is equal to the number of refined elements. Both, the sequential and parallel solver with reutilization is $O\left(\frac{N}{r}\right)$ faster than the solver without the reutilization, where N is number of elements and r is number of refinements.

Acknowledgments. We acknowledge the support of Polish MNiSW grant no. 3TO8B05529 and Foundation for Polish Science under Homming Programme.

References

1. Paszyński, M., Pardo, D., Torres-Verdin, C., Demkowicz, L., Calo, V.: Multi-Level Direct Sub-structuring Multi-frontal Parallel Direct Solver for hp Finite Element Method. ICES Report 07-33 (2007)
2. Paszyński, M., Pardo, D., Torres-Verdin, C., Matuszyk, P.: Efficient Sequential and Parallel Solvers for hp FEM. In: APCOM-EPMSC 2007, Kioto, Japan (2007)
3. Demkowicz, L.: Computing with hp-Adaptive Finite Elements, vol. I. Chapman & Hall/Crc Applied Mathematics & Nonlinear Science, New York (2006)
4. Demkowicz, L., Pardo, D., Paszyński, M., Rachowicz, W., Zduneka, A.: Computing with hp-Adaptive Finite Elements, vol. II. Chapman & Hall/Crc Applied Mathematics & Nonlinear Science, New York (2007)
5. Paszyński, M., Kurtz, J., Demkowicz, L.: Parallel Fully Automatic hp-Adaptive 2D Finite Element Package. Computer Methods in Applied Mechanics and Engineering 195(7-8), 711–741 (2007)
6. Paszyński, M.: Parallelization Strategy for Self-Adaptive PDE Solvers. Fundamenta Informaticae (submitted, 2007)
7. Duff, I.S., Reid, J.K.: The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems. ACM Trans. on Math. Soft. 9, 302–325 (1983)
8. Giraud, L., Marocco, A., Rioual, J.-C.: Iterative Versus Direct Parallel Substructuring Methods in Semiconductor Device Modelling. Numerical Linear Algebra with Applications 12(1), 33–55 (2005)
9. Scott, J.A.: Parallel Frontal Solvers for Large Sparse Linear Systems. ACM Trans. on Math. Soft. 29(4), 395–417 (2003)
10. Milti-frontal Massively Parallel Sparse Direct Solver (MUMPS), `http://graal.ens-lyon.fr/MUMPS/`
11. Pardo, D., Calo, V.M., Torres-Verdin, C., Nam, M.J.: Fourier Series Expansion in a Non-Orthogonal System of Coordinates for Simulation of 3D Borehole Resistivity Measurements; Part I: DC. ICES Report 07-20 (2007)