Maciej Paszyński*, Tomasz Jurczyk**, David Pardo***

MULTI-FRONTAL SOLVER FOR SIMULATIONS OF LINEAR ELASTICITY COUPLED WITH ACOUSTICS

This paper describes the concurrent multi-frontal direct solver algorithm for a multi-physics Finite Element Method (FEM). The multi-physics FEM utilizes different element sizes as well as polynomial orders of approximation over element edges, faces, and interiors (element nodes). The solver is based on the concept of a node, and management of unknowns is realized at the level of nodes. The solver is tested on a challenging multi-physis problem: acoustics coupled with linear elasticity over a 3D ball shape domain.

Keywords: *parallel simulations, multi-frontal solvers, finite element method, linear elasticity coupled with accoustics, 3D mesh generation*

SOLVER WIELOFRONTALNY DO SYMULACJI LINIOWEJ SPRĘŻYSTOŚCI SPRZĘŻONEJ Z AKUSTYKĄ

Artykuł opisuje współbieżny algorytm solwera wielofrontalnego przeznaczonego do rozwiązywania za pomocą metody elementów skończonych (MES) problemów liniowej sprężystości sprzężonych z akustyką. Natura problemów sprzężonych, takich jak rozważany problem akustyki sprzężonej ze sprężystością, wymaga zastosowania różnej ilości niewiadomych w różnych węzłach siatki obliczeniowej stosowanej w MES. Dlatego też algorytm solwera opiera się na koncepcji węzła obliczeniowego. Algorytm solwera testowany jest na trudnym problemie obliczeniowym – propagacji fal akustycznych na trójwymiarowej kuli reprezentującej uproszczony model głowy ludzkiej.

Słowa kluczowe: *symulacje równoległe, solvery wielofrontalne, metoda elementów skończonych, liniowa spreżystość sprzężona z akustyką, generacja siatek trójwymiarowych*

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, IT and Electronics, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland, paszynsk@agh.edu.pl

** AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, IT and Electronics, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland, jurczyk@agh.edu.pl

*** Department of Applied Mathematics, Statistics, and Operational Research, University of the Basque Country (UPV/EHU), Leioa, Spain and IKERBASQUE (Basque Foundation of Sciences), Bilbao, Spain, dzubiaur@gmail.com

1. Introduction

In this paper we focus on the development of an efficient concurrent multi-frontal direct solver. The multi-frontal direct solver is the current state-of-the art technique [1, 2, 3, 9] for solving sparse systems of linear equations resulting from Finite Element Method (FEM) [6].

We focus on multiphysics simulations, in particular on the linear elasticity coupled with accoustics. In the multi-scale problem, there are three components of the unknown elastic velocity vector field over the elastic domain, one component of the unknown pressure scalar field over the acoustic domain, and four unknowns over the interface. We also employ third order polynomials over the entire mesh, except for the two external layers, where four order polynomials are used for the Perfectly Matching Layer technique [8]. Thus, with the exception of the interface, the number of unknowns at each vertex node is equal to one or three (depending on the acoustic / elasticity domain type), the number of unknowns at each edge is $p-1$ or $3*(p-1)$ (acoustic / elasticity), the number of unknowns at each face is of the order of $(p-1)^2$ or $3*(p-1)^2$ (acoustic / elasticity) and the number of unknowns at each interior is of the order of $(p-1)^3$ or $3*(p-1)^3$ (acoustic / elasticity). Here p=3 or 4, respectively.

The main disadvantage of having a different number of unknowns on each node is the problem of ordering unknowns for the solver, and deciding which unknowns can be eliminated first. This is related to the location of the node only, so it is better to design the solver performing ordering at the level of nodes, not at the level of unknowns.

In order to overcome the above problem, we introduce the concept of a node, understood as a group of unknowns associated with an element edge, face, or interior, that share the same connectivities. The management of unknowns is performed at the level of nodes.

The main contribution of this paper is to introduce the concept of a node for management of the ordering for multi-level recursive solution performed by the multi-frontal parallel solver.

The structure of the paper is the following. In section *Main idea of the solver algorithm*, we introduce the concept of the node-based solver and explain the algorithm on a simple two dimensional example. The following section *Three dimensional solver algorithm* discusses the generalization of the nodes-based solver discussed in the second section into the three dimensional case. In the next section, entitled *The model problem for acoustics coupled with linear elasticity* we introduce the details of the multi-physics prolblem, the linear elasticity coupled with acoustics. In particular the section contains the variational formulation for the coupled problem and the definition of the PML technique. In particular the construction of the computational mesh is presented here. The section *3D multi-physics simulations* descibes the numerical results obtained by execution of the node-based solver onto the multi-physics problem. Finally, section *Measurements of the solver efficiency* discusses the efficiency

of the parallel algorithm and compares it with state of the art MUMPS solver. The paper is closed with the *Conclusions* section.

2. Main idea of the solver algorithm

In this section we explain the main idea of the node-based solver. For simplicity, we focus on a simple two dimensional rectangular mesh, and a scalar valued problem.

We consider a two dimensional rectangular finite element mesh, with polynomial orders of approximation equal to p_j on the four element edge nodes $j = 1, 2, 3, 4$ and (p_h, p_v) on the element interior node. The element local matrix consists of several submatrices. The element nodes are four vertices, four edges and one interior node. The number of unknowns over the vertex nodes is equal to one. The number of unknowns over the edge nodes is equal to $p_j - 1$, and the number of unknowns for the interior node is equal to $(p_h - 1)(p_v - 1)$. We utilize the hypermatrix concept, defined as a recursive matrix of matrices. The element local matrix is stored as a hypermatrix, and the ordering is performed at the level of nodes, not at the level of unknowns. See Fig. 1 for a two dimensional example.

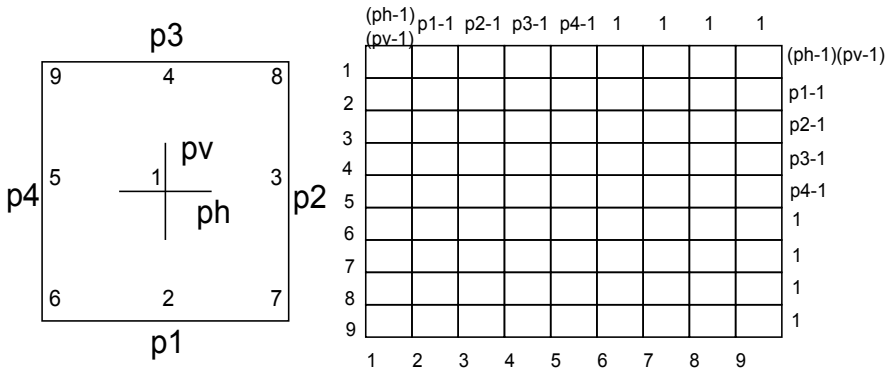

Fig. 1. (Left panel) A single two dimensional finite element with polynomial orders of approximation over edges equal to p1, p2, p3, p4 and polynomial order of approximation over the interior equal to ph in the horizontal direction and pv in the vertical direction. **(Right panel)** Sizes of blocks in element local matrix: 1 unknown per element vertex, pi-1 unknowns per edge, and (ph-1)(pv-1) unknowns for the element interior

The main idea of the solver is illustrated in Figures 2, 3, and 4 and Tables 1, 2, and 3. In this simple example we have a mesh with four finite elements. The mesh is partitioned into four processors. Thus, a subdomain matrix corresponds to a single element matrix.

In the first step of the algorithm, described in Figure 2 and in Table 1, the element local matrices are generated concurrently, on separate processors. This is realized by calling *get_subdomain_system* routine. At this point, the unknowns associated with element interior nodes can be eliminated.

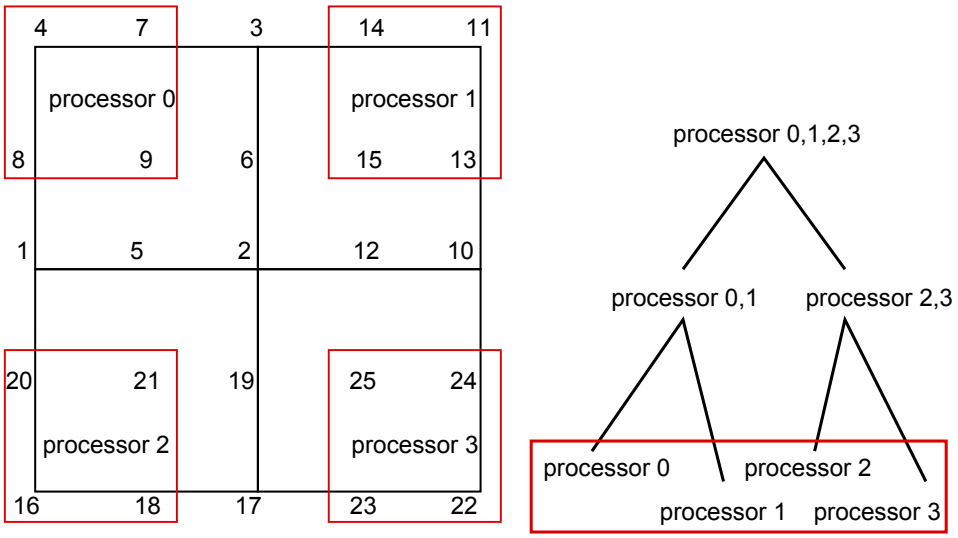

Fig. 2. Management of nodes associated with leaves of the elimination tree

These nodes are localized by *find_nodes_to_eliminate* routine. Finally, the elimination is performed by calling *get_schur_complement* routine.

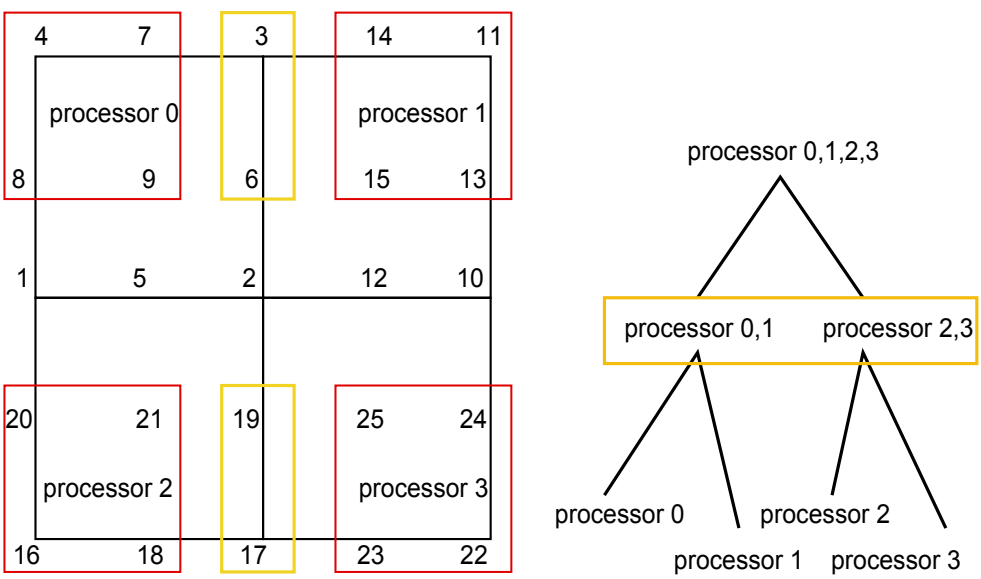

Fig. 3. Management of nodes from second level of the elimination tree

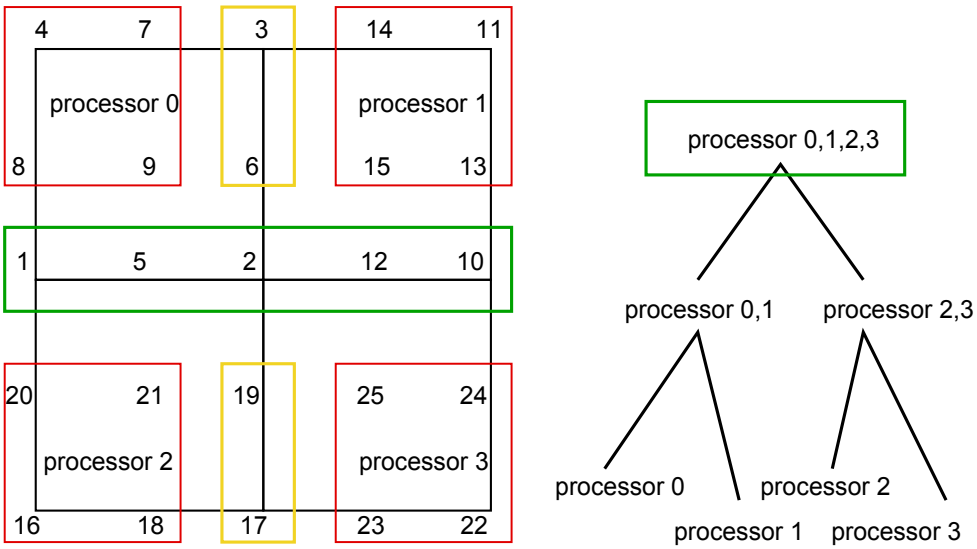

Fig. 4. Management of nodes associated with top of the elimination tree

Table 1
First step of the solver algorithm

Processor 0
call get_subdomain_system 1,2,3,4,5,6,7,8,9 call find_nodes_to_eliminate 4,7,8,9 —— 1,2,3,5,6 call get_schur_complement receive nodes 2,10,3,12,6 from Processor 1
Processor 1
call get_subdomain_system 2,10,11,3,12,13,14,6,15 call find_nodes_to_eliminate 11,13,14,15 —— 2,10,3,12,6 call get_schur_complement send nodes 2,10,3,12,6 to Processor 0
Processor 2
call get_subdomain_system 16,17,2,1,18,19,5,20,21 call find_nodes_to_eliminate 16,18,20,21 —— 17,2,1,19,5 call get_schur_complement receive nodes 17,10,2,12,19 from Processor 3
Processor 3
call get_subdomain_system 17,22,10,2,23,24,12,19,25 call find_nodes_to_eliminate 22,23,24,25 —— 17,10,2,12,19 call get_schur_complement send nodes 17,10,2,12,19 to Processor 2

Table 2
Second step of the solver algorithm

Processors 0,1
call create_system 1,2,3,5,6 + 2,10,3,12,6 = 1,2,3,5,6,10,12
call find_nodes_to_eliminate 3,6 —— 1,2,5,10,12
call get_schur_complement
receive nodes 1,2,5,10,12 from Processor 2
Processors 2,3
call create_system 17,2,1,19,5 + 17,10,2,12,19 = 1,2,5,10,12,17,19
call find_nodes_to_eliminate 19,17 —— 1,2,5,10,12
call get_schur_complement
send nodes 1,2,5,10,12 to Processor 0

Table 3
Third step of the solver algorithm

Processors 0,1,2,3
call create_system 1,2,5,10,12 + 1,2,5,10,12 = 1,2,5,10,12
call full_forward_elimination

The first step ends after sending the resulting Schur complement matrices from processor 1 to processor 0 and from processor 3 to processor 2.

In the second step of the algorithm, summarized in Figure 3 and in Table 2, the Schur complements matrices are merged into a single matrix. This is performed in *create_system* routine. At this point, only unknowns associated with common edges (shared between two subdomains) can be eliminated. These nodes are localized by *find_nodes_to_eliminate* routine, and the elimination is performed by *get_schur_complement* routine. Notice that we have two groups of processors: 0,1 and 2,3, working together over two matrices. The second step of the algorithm ends after sending the Schur complement contribution from processors 2,3 to processors 0,1.

In the last step of the algorithm, described in Figure 4 and Table 3, all processors are utilized to solve the resulting top interface problem. This problem is always associated with the cross-section of the domain, denoted in Figure 5 by the green color. This is done by merging two Schur complement contributions in routine *create_system*, as well as by performing full elimination in *create_system* routine, followed by recursive backward substitutions.

3. Three dimensional solver algortihm

In this section we present how the node-based algorithm introduced in section *Main idea of the solver algorithm* is generalized to solve the three dimensional ball-shape domain.

The computational ball-shape domain described in Figure 5 has been obtained by generating the core of the mesh, representing the tissue part of the domain (Figure 6). The tissue part of the mesh has been generated with tetrahedral elements. In the following steps, the tissue ball has been extended to the skull layer, by generating an additional layer of tetrahedral elements. Finally, the air and PML layers have been generated, by adding four layers of prism elements over the boundary based on triangles. We employ third order polynomials over the entire mesh, except within the PML, where four order polynomials are used.

The mesh is partitioned into uniformly loaded sub-domains by cutting the 3D ball shape into slices, as presented on bottom panel in Figure 6. This simple load balancing procedure is satisfactory for large number of elements, and when the polynomial order of approximation is almost uniformly distributed over the entire mesh. This is the case in our simulations.

The multi-level approach described in details in section *Main idea of the solver algorithm* is employed here on the multi-level three dimensional elimination tree, presented on the bottom panel in Figure 6. The three dimensional algorithm is summarized below:

1. First, nodes from the interior of each sub-domain are identified. This is done by browsing all elements from a subdomain, generating element local matrices and transforming them into a submatrices of a hypermatrix. The ordering of unknowns is performed at the level of nodes. This step is executed in parallel over each subdomain. The subdomain internal nodes are identified, and marked to be eliminated.
2. In this step, the Schur complements over each subdomain is computed. Subdomain internal nodes are eliminated with respect to the interface nodes. This is performed by submitting a subdomain element local hypermatrices into a sequential instance of the MUMPS solver [1, 2, 3, 9]. MUMPS performs partial forward elimination over each sub-domain to calculate the local contribution to the global interface problem. In this step, several sequential instances of the MUMPS solver are executed in parallel. The output from MUMPS is again transformed into a hypermatrix form.
3. Sub-domains are joined into pairs, and local contributions to the global interface problem are assembled within these pairs. This is realized by sending one Schur complement hypermatrix from an odd processor to an even one, and by browsing the two Schur complement hypermatrices at the level of nodes, identifying common nodes and merging them together into a new (third) hypermatrix. Fully assembled nodes are identified and marked for the elimination.
4. In the next step, the Schur complement over each pair of processors is computed. This is realized by submitting a subdomain hypermatrix into a parallel instance of the MUMPS solver, utilizing the two processors. Parallel MUMPS performs partial forward elimination over each sub-domain to calculate the local contribution to the global interface problem. In this step, several parallel instances of the

MUMPS solver are executed in parallel. The output from the MUMPS solver is again transformed into a hypermatrix form.

5. Processors are joint into sets of four processors (two previous step pairs in each set). The procedure is repeated recursively until there is only one common interface problem matrix with all remaining interface d.o.f. aggregated.
6. This common interface problem hypermatrix is formed in the root of the elimination tree and the single instance of parallel MUMPS solver is executed over all processors. In general, this interface is associated with a cross-section of the domain. The solution obtained from the MUMPS solver is transformed back into the hypermatrix structure.
7. The solution of the common interface is broadcast back into two group of processors. Local solutions are substituted into Schur complement matrices from the corresponding step. Backward substitution is executed to obtain further contributions to the global interface problem solution.
8. The procedure is repeated until we complete the solution of the global interface problem.
9. In the last step, the backward substitution is executed over sub-domains and the global problem is finally solved.

4. The model problem for acoustics coupled with linear elasticity

In this section the details of the computational problem are introduced. The problem falls into the category of general coupled elasticity/acoustics problems discussed in [5] with a few modifications. The domain Ω in which the problem is defined is the interior of a ball presented in Figure 5, and it is split into an acoustic part Ω_a, and an elastic part Ω_e. The problem is intended to model in a simplified way the acoustic response of the human head to an incoming acoustic wave. Depending upon a particular example, the acoustic part Ω_a includes:

- air surrounding the ball, bounded by the ball surface and a truncating sphere
- an additional layer of air bounded by the truncating sphere and the outer sphere terminating the computational domain, where the equations of acoustics are replaced with the corresponding *Perfectly Matched Layer* (PML) modification.

The elastic part of the domain includes:

- skull,
- tissue.

The term *tissue* is understood here as the part of the head not occupied by the skull (bone). In the current stage of the project it is assumed that the elastic constants for the entire tissue domain are the same.

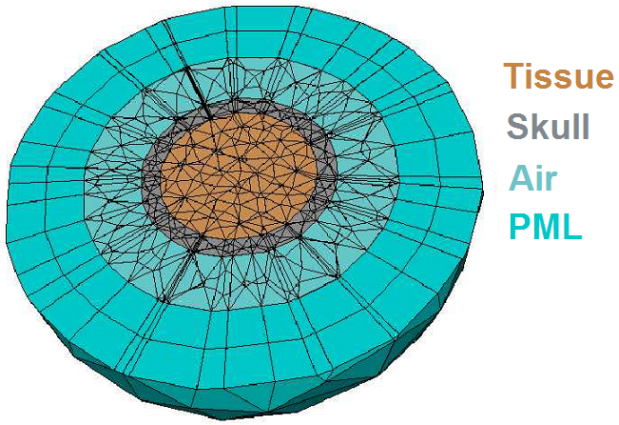

Fig. 5. Structure of the computational mesh

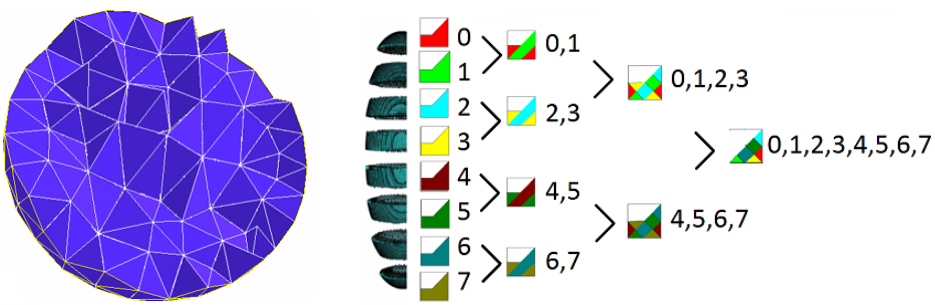

Fig. 6. (Left panel) Cross-section of the 3D ball-shape domain for linear elasticity computations. **(Right panel)** Elimination tree for 8 subdomains

The weak (variational) final formulation of the problem has the following form. We seek for elastic velocity $\boldsymbol{u}$ and pressure scalar field p such that

$$\begin{cases} \boldsymbol{u} \in \tilde{\boldsymbol{u}}_D + \boldsymbol{V},\ p \in \tilde{p}_D + V, \\ b_{ee}(\boldsymbol{u}, \boldsymbol{v}) + b_{ae}(p, \boldsymbol{v}) = l_e(\boldsymbol{v}), \quad \forall \boldsymbol{v} \in \boldsymbol{V} \\ b_{ea}(\boldsymbol{u}, q) + b_{aa}(p, q) = l_a(q), \quad \forall q \in V \end{cases} \tag{4.1}$$

with the bilinear and linear forms defined as follows.

$$\begin{aligned} b_{ee}(\boldsymbol{u},\boldsymbol{v}) &= \int_{\Omega_e} \left(E_{ijkl} u_{k,l} v_{i,j} - \rho_s \omega^2 u_i v_i \right) \, d\boldsymbol{x} \\ b_{ae}(p,\boldsymbol{v}) &= \int_{\Gamma_I} p v_n \, dS \\ b_{ea}(\boldsymbol{u},q) &= -\omega^2 \rho_f \int_{\Gamma_I} u_n q \, dS \\ b_{aa}(p,q) &= \int_{\Omega_a} \left(\boldsymbol{\nabla} p \boldsymbol{\nabla} q - k^2 p q \right) \, d\boldsymbol{x} \\ l_e(\boldsymbol{v}) &= -\int_{\Gamma_I} p^{inc} v_n \, dS \\ l_a(q) &= -\int_{\Gamma_I} \frac{\partial p^{inc}}{\partial n} q \, dS \end{aligned} \tag{4.2}$$

$\boldsymbol{V}$ and V are the spaces of the test functions,

$$\begin{aligned} \boldsymbol{V} &= \{ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega_e) \colon \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_{De} \} \\ V &= \{ q \in H^1(\Omega_a) \colon q = 0 \text{ on } \Gamma_{Da} \} \end{aligned} \tag{4.3}$$

Here ρ_f is the density of the fluid, ρ_s is the density of the solid, E_{ijkl} is the tensor of elasticities, ω is the circular frequency, c denotes the sound speed, k is the acoustic wave number and p^{inc} is the incident wave impinging from the top. For more details we refer to [5]. Coupled problem (4.1) is symmetric if and only if diagonal forms b_{ee} and b_{aa} are symmetric and,

$$b_{ae}(p,\boldsymbol{u}) = b_{ea}(\boldsymbol{u},p).$$

Thus, in order to recover the symmetry of the formulation, it is necessary to rescale the problem. For instance by dividing the second equation by factor $-\omega^2 \rho_f$.

$$\begin{aligned} b_{ee}(\boldsymbol{u},\boldsymbol{v}) &= \int_{\Omega_e} \left(E_{ijkl} u_{k,l} v_{i,j} - \rho_s \omega^2 u_i v_i \right) \, d\boldsymbol{x} \\ b_{ae}(p,\boldsymbol{v}) &= \int_{\Gamma_I} p v_n \, dS \\ b_{ea}(\boldsymbol{u},q) &= -\int_{\Gamma_I} u_n q \, dS \\ b_{aa}(p,q) &= \frac{1}{\omega^2 \rho_f} \int_{\Omega_a} \left(\boldsymbol{\nabla} p \boldsymbol{\nabla} q - k^2 p q \right) \, d\boldsymbol{x} \\ l_e(\boldsymbol{v}) &= -\int_{\Gamma_I} p^{inc} v_n \, dS \\ l_a(q) &= -\frac{1}{\omega^2 \rho_f} \int_{\Gamma_I} \frac{\partial p^{inc}}{\partial n} q \, dS \end{aligned} \tag{4.4}$$

The incident wave is assumed in the form of a plane wave impinging from the top, $p^{inc} = p_0 e^{ikex}$, $e = (0, 0, -1)$, $p_0 = 1[Pa]$. The test problem is being solved with frequency $f = 200\,\text{Hz}$. The precise geometry data are as follows: brain $r < 0.1\,\text{m}$, skull $0.1\,\text{m} < r < 0.125\,\text{m}$, air $0.125\,\text{m} < r < 0.2\,\text{m}$ and PML air $0.2\,\text{m} < r < 0.3\,\text{m}$.

In the PML part of the acoustical domain, the bilinear form b_{aa} is modified as follows:

$$b_{aa}(p,q) = \int_{\Omega_{a,PML}} \left(\frac{z^2}{z' r^2} \frac{\partial p}{\partial r} \frac{\partial q}{\partial r} + \frac{z'}{r^2} \frac{\partial p}{\partial \psi} \frac{\partial q}{\partial \psi} + \frac{z'}{r^2 \sin^2 \psi} \frac{\partial p}{\partial \theta} \frac{\partial q}{\partial \theta} \right) r^2 \sin \psi dr d\psi d\theta$$

Here r, ψ, θ denote the conventional spherical coordinates and $z = z(r)$ is the PML stretching factor defined as

$$z(r) = \left(1 - \frac{i}{k} \left[\frac{r-a}{b-a} \right]^{\alpha} \right) r \tag{4.5}$$

Here a is the radius of the truncating sphere, b is the external radius of the computational domain ($b - a$ is thus the thickness of the PML layer), i denotes the imaginary unit, k is the acoustical wave number, and r is the radial coordinate. In computations, all derivatives with respect to spherical coordinates are expressed in terms of the standard derivatives with respect to Cartesian coordinates. In all reported computations, parameter $\alpha = 5$. For a detailed discussion on derivation of PML modifications and effects of higher order discretizations, see [8].

The material data corresponding the considered layers are presented in Table 4.

Table 4
Material constants

Material	E [MPa]	ν	ρ [kg/m^3]	c_p [m/s]	c_s [m/s]
tissue	0.67	0.48	1040	75	15
skull (bone)	6500	0.22	1412	2293	1374
air			1.2	344	

5. 3D multi-physics simulations

In this section, we describe numerical simulations of a three dimensional acoustics problem coupled with linear elasticity. We utilize the methodology described [6] to analyse the quality of the core tetrahedral elements ball shape mesh. The statistic for the core mesh is summarized in Table 5.

In this model, the domain consists of four concentric spheres, as presented in Figure 5. Inner sphere is filled with an elastic material with data corresponding to human brain. The first layer is also elastic with constants corresponding to human skull. The second layer corresponds to air, and the last one to the PML air.

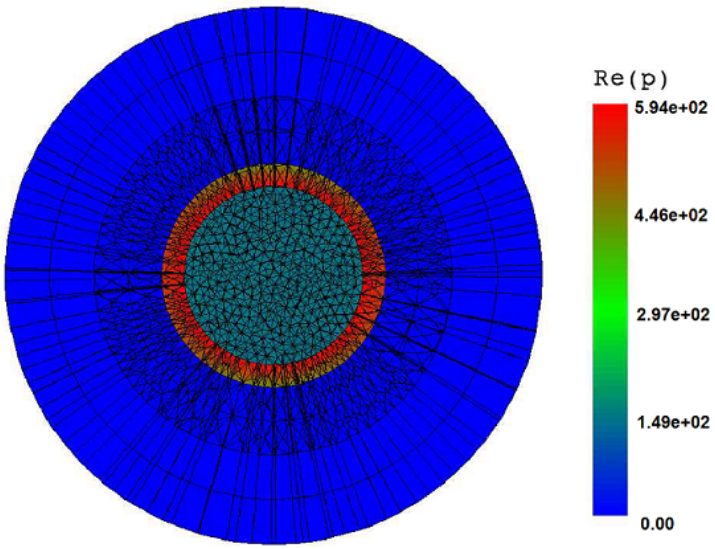

Fig. 7. Plot of the real part of pressure on plany $y = 0$ passing throught the center of the ball

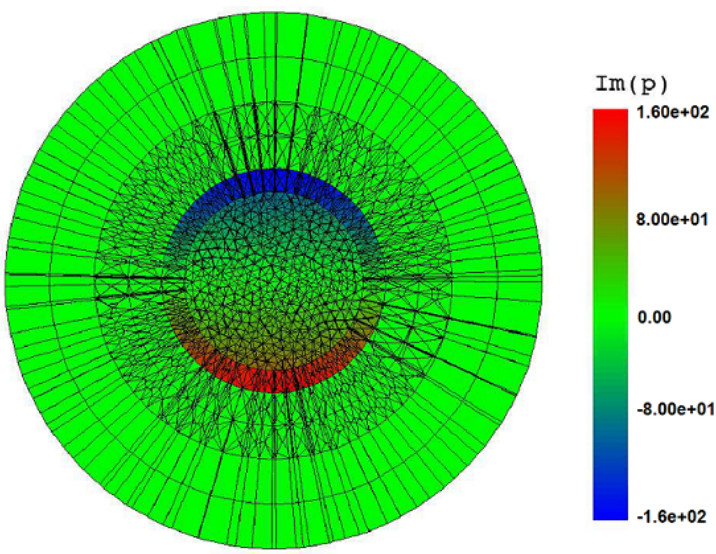

Fig. 8. Plot of the imaginary part of pressure on plany $y = 0$ passing throught the center of the ball

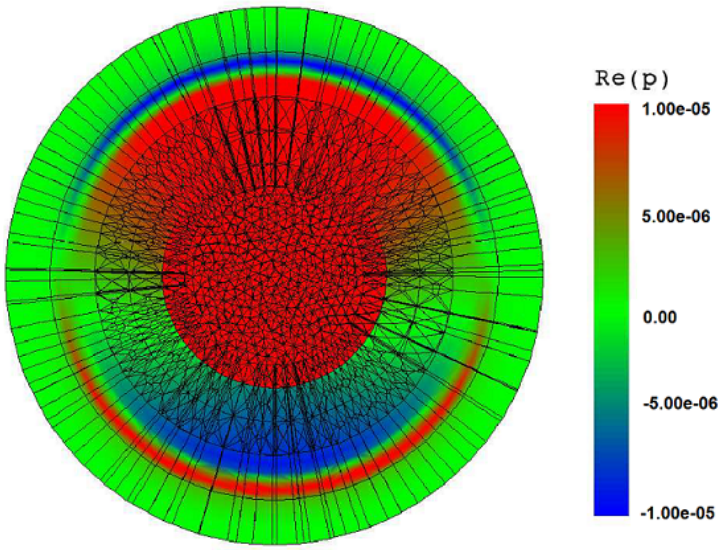

Fig. 9. Plot of the real part of pressure on plany $y = 0$ passing throught the center of the ball, in the range -0.00001 to 0.00001

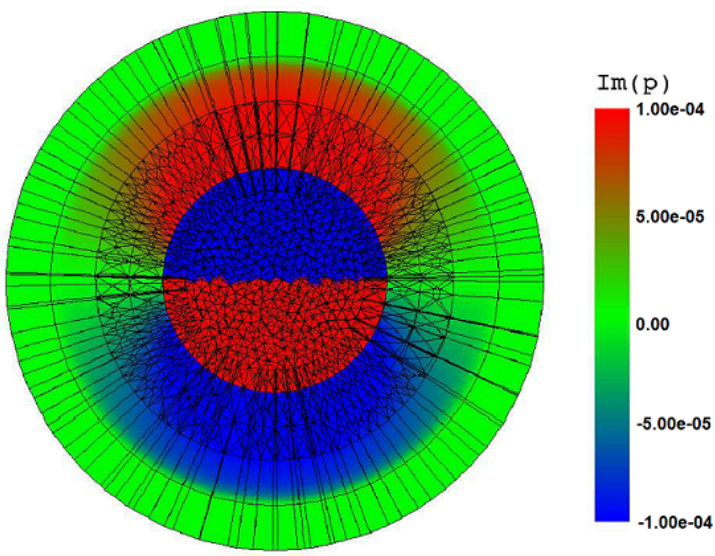

Fig. 10. Plot of the imaginary part of pressure on plany $y = 0$ passing throught the center of the ball, in the range -0.00001 to 0.00001

Table 5
Parameters of the core mesh

Parameter	Fine mesh
Number of points	1865
Number of edges	10875
Number of tetrahedrals	8085
Maximum number of tetrahedrals with common edge	9
Average number of tetrahedrals with common edge	4.5

The total number of unknowns is 288,899, however what really maters is not size of matrix but number of non-zero entries and sparsity pattern. In this 3D problem we have 40,215,303 non zero entries.

Figures 7 and 8 display the distribution of real and imaginary parts of the pressure over the $y = 0$ section of the mesh. Figures 9 and 10 display the distribution of real and imaginary parts of the pressure over the $y = 0$ section obtained on the big mesh rescaled to values from $-.00001$ to .00001 for the real part, and from $-.0001$ to .0001 for the imaginary part.

For the discussion on the validity of the numerical solution, based on the comparison to the analytical one, we refer to [4].

To visualize our meshes, we have implemented a simple interface to the Visualization Toolkit (VTK) [11], a collection of C++ classes that implement a wide range of visualization algorithms. The central data structure for this project is the vtkUnstructuredGrid, which represents volumetric data as a collection of points with corresponding scalar values (in this case, the real and imaginary part of the pressure), connected by cells of arbitrary type and dimension (i.e. lines, triangles, quads, tetrahedra, prisms, etc.).

6. Measurements of the solver efficiency

In this section, we compare the measurements of the execution times and memory usage of our multi-level solver, against parallel MUMPS solver [9]. In all tests, the METIS ordering [7] is utilized within MUMPS.

The simulations have been performed over a linux cluster from our Department of Computer Science, with architecture summarized in Table 8. The solver has been implemented in fortran90 with MPI.

We start with presenting in Table 6 the execution time and memory usage for sequential MUMPS solver executed over 1 processor as well as for parallel MUMPS solver, executed over 4, 8, 12 and 16 processors. The measured execution times include the matrix assembly, time spend on analysing the connectivity data by METIS in order to construct a proper ordering, and time spend on factorization.

Next, we present the detailed execution times and memory usage for our multi-level solver, executed for 8 and 16 processors. This is reported in Tables 7

Table 6
Total execution time and memory usage per processor for MUMPS solver

Cores	Execution time [s]	Memory usage per cores [MB]
1	2221	3998
4	1426	4516, 4516, 4417, 4156
8	812	1761, 1857, 2117, 1804, 1845, 1752, 1612, 2117
16	644	1183, 872, 873, 1151, 1165, 866, 1005, 992, 1120, 957, 1143, 954, 814, 968, 934, 1183
20	789	570, 723, 935, 940, 737, 906, 736, 840, 1166, 858, 797, 609, 1080, 964, 767, 998, 905, 821, 1166, 797

and 8 for 8 and 16 processors, respectively. The tables report the total execution time spend by each processor (or set of processors) at particular nodes of the elimination tree. The time includes the assembly, the reordering of nodes, and factorization. It also reports the memory usage. Each table is summarized with *Total* line, which contains the sum of maximum execution times for levels of the elimination tree as well as maximum memory usage. This is the total execution time and memory usage for the parallel solver.

Table 7
Linux cluster details

Node	Number of cores	Processor clock [GHz]	Total memory [GB]	Memory per core [GB]
Node 0	2	3.0	8	4
Node 1	2	3.0	8	4
Node 2	2	2.8	8	4
Node 3	2	2.8	8	4
Node 4	2	2.8	8	4
Node 5	2	3.0	6	3
Node 6	2	2.8	6	3
Node 7	12	2.4	16	1.33

Our multi-level solver obtained 7.76 speedup (0.97 efficiency) for 8 processors and 6.20 speedup (0.38 efficiency) for 16 processors. The parallel MUMPS solver obtained 2.73 speedup (0.34 efficiency) for 8 processors and 3.44 speedup (0.21 efficiency) for 16 processors.

The maximum memory usage per core for our solver was 1873 [MB] for 8 processors and 1916 [MB] for 16 processors. The maximum memory usage per core for parallel MUMPS solver was 2117 [MB] for 8 processors and 1183 [MB] for 16 processors.

Table 8
Solver execution statistics per 8 subdomains elimination tree

Sub-domains	Problem size	Number of non zero entries	Execution time [s]	Memory usage [MB]
Subdomain 0	7652	830707	14	265
Subdomain 1	5020	401356	5	262
Subdomain 2	7289	777229	12	232
Subdomain 3	7971	860175	13	311
Subdomain 4	7392	747797	13	341
Subdomain 5	7290	789420	13	224
Subdomain 6	8076	865398	29	373
Subdomain 7	7626	821297	17	251
Sub-domains 0,1	4327	11434513	16	881
Sub-domains 2,3	4468	11536477	10	962
Sub-domains 4,5	4060	11507491	21	765
Sub-domains 6,7	4371	12484213	24	896
Sub-domains 0,1,2,3	6441	30409371	103	1873
Sub-domains 4,5,6,7	5971	24384651	95	1640
Sub-domains 0,1,2,3,4,5,6,7,8	4930	24304900	130	960
Total			286	1873

Table 9
Solver execution statistics per 16 subdomains elimination tree

Sub-domains	Problem size	Number of non zero entries	Execution time [s]	Memory usage [MB]
Sub-domain 0	3557	332191	3	87
Sub-domain 1	3911	372952	3	137
Sub-domain 2	4416	442464	4	166
Sub-domain 3	4414	402744	4	152
Sub-domain 4	3769	345943	3	151
Sub-domain 5	3382	301744	3	162
Sub-domain 6	5033	510041	5	301
Sub-domain 7	4429	444631	4	186
Sub-domain 8	3956	352765	3	166
Sub-domain 9	4205	426477	4	161
Sub-domain 10	4454	475695	5	138
Sub-domain 11	4043	392659	4	145
Sub-domain 12	4112	430856	4	120
Sub-domain 13	3562	279616	3	162
Sub-domain 14	4372	431209	5	159
Sub-domain 15	4212	416574	5	140
Sub-domains 0,1	2910	4667527	4	392
Sub-domains 2,3	3234	6531273	11	462
Sub-domains 4,5	3676	5543848	3	379
Sub-domains 6,7	4496	10012208	19	747
Sub-domains 8,9	3447	7165240	13	534
Sub-domains 10,11	3206	6010750	9	444
Sub-domains 12,13	3318	5749926	6	438
Sub-domains 14,15	3310	6446964	11	484
Sub-domains 0,1,2,3	4817	13832607	22	1003
Sub-domains 4,5,6,7	6147	16623332	52	1327

Table 9. cont.

Sub-domains 8,9,10,11	4772	15046955	38	984
Sub-domains 12,13,14,15	5389	15534227	45	1329
Sub-domains 0,1,2,3,4,5,6,7,8	7040	35658739	170	1916
Sub-domains 9,10,11,12,13,14,15	5389	15534227	45	1329
Sub-domains 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15	4753	22576729	112	892
Total			358	1916

7. Conclusions

In this paper we analyzed the possibility of using a concurrent solver applied to challenging multi-physics problems resulting from FEM simulations, with varying number of unknowns in different parts of the mesh and possibly different polynomial orders of approximation. We overcome the difficulties by utilizing the hypermatrix concept based on the notion of a node. We also designed and implemented a new concurrent direct solver, tested its efficiency and compared it with the MUMPS solver. Our solver obtained 7.7 speedup and turned out to be up to 1.8 times faster than MUMPS solver.

Acknowledgements

The work reported in this paper was supported by the Polish MNiSW grant no. NN 519 405737. The work of the third author was partially funded by the Spanish Ministry of Science and Innovation under the project MTM2010-16511. The first author would like also to acknowledge the Foundation for Polish Science for providing funds necessary for purchasing the linux cluster.

References

[1] Amestoy P. R., Duff I. S., L'Excelent J.-Y.: *Multifrontal parallel distributed symmetric and unsymmetric solvers*. Computer Methods in Applied Mechanics and Engineering, vol. 184, 2000, pp. 501–520.

[2] Amestoy P. R., Duff I. S., Koster J., L'Excelent J.-Y.: *A fully asynchronous multifrontal solver using distributed dynamic scheduling*. SIAM Journal of Matrix Analysis and Applications, vol. 23, 2001, pp. 15–41.

[3] Amestoy P. R., Guermouche A., L'Excelent J.-Y., Pralet S.: *Hybrid scheduling for the parallel solution of linear systems*. Parallel Computing, vol. 32(2), 2006, pp. 136–156.

[4] Demkowicz L., Gatto P., Kurtz J., Paszyński M., Rachowicz W., Bleszyński E., Bleszyński M., Hamilton M., Champlin C., Pardo D.: *Modeling of bone conduction of sound in human head using hp finite elements. Part I. Code design and modificaton*. Computer Methods in Applied Mechanics and Engineering, vol. 200, 2011, pp. 1757–1773.

[5] Demkowicz L., Kurtz J., Pardo D., Paszyński M., Rachowicz W., Zdunek A.: *Computing with hp-Adaptive Finite Element Method. Vol. II. Frontiers: Three Dimensional Elliptic and Maxwell Problems*. Chapmann & Hall / CRC Applied Mathematics & Nonlinear Science, 2007.

[6] Glut B., Jurczyk T., Kitowski T.: *Anisotropic Volume Mesh Generation Controlled by Adaptive Metric Space*. AIP Conf. Proc. Volume 908 NUMIFORM 2007, Materials Processing and Design: Modeling, Simulation and Applications, June 17–21, Porto, Portugal, 2007, pp. 233–238.

[7] Karypis G., Kumar V.: *A fast and high quality multilevel scheme for partitioning irregular graphs*. SIAM Journal on Scientific Computing, vol. 20, issue 1, 1999, pp. 359–392.

[8] Michler Ch., Demkowicz L., Kurtz J., Pardo J.: *Improving the performance of perfectly matched layers ny means of hp-adaptivity*. ICES-Report 06-17, The University of Texas at Austin, 2006.

[9] `http://www.enseeiht.fr/lima/apo/MUMPS`

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

[11] Schroeder W., Martin K., Lorensen B.: *The Visualization Toolkit An Object--Oriented Approach To 3D Graphics*. 3rd Edition. Kitware, Inc. `www.kitware.com`