AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

PRACA DOKTORSKA

MARCIN SIENIEK

STRATEGIE ADAPTACYJNE DLA PROBLEMÓW WIELOSKALOWYCH

Promotor:

dr hab. Maciej Paszyński, prof. AGH

Kraków, 2015

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie, i nie korzystałem ze źródeł innych niż wymienione w pracy.

AGH

UNIVERSITY OF SCIENCE AND TECHNOLOGY IN KRAKOW

Faculty of Computer Science, Electronics and Telecommunications

DEPARTMENT OF COMPUTER SCIENCE

PHD THESIS

MARCIN SIENIEK

ADAPTIVE STRATEGIES FOR MULTISCALE PROBLEMS

Supervisor:

Maciej Paszyński, Ph.D., D.Sc.

Krakow, 2015

I would like to herby express my utmost gratitude and true appreciation to Professor Maciej Paszyński, for all his guidance, assistance and persistence that were key to successful completion of my research concluded with this thesis.

CONTENTS

1 MOTIVATION

1.1 MAIN THESIS OF THIS BOOK

The main thesis of this work may be summarized as follows:

'*It is possible to develop adaptive algorithms for solving difficult multi-scale problems that allow for accurate representation of the material data resulting from MRI scans, delivering exponential convergence of the numerical error with respect to the problem size, resulting in linear computational cost.*'

1.2 STATE OF THE ART

1.2.1 DIGITAL MATERIAL REPRESENTATION

Digital material representation utilizes generic bitmaps for representation of e.g. morphology of the material during finite element (FE) analysis of material behavior under deformation and exploitation conditions [C19, C20, C21]. Due to the crystallographic nature of polycrystalline material, particular features are characterized by different properties that significantly influence material deformation. To properly capture FE solution gradients which are the results of mentioned material inhomogenities, specific refined meshes have to be created.

It can be generally stated that recently observed needs of the automotive and aerospace industries for new metallic materials that can meet strict requirements regarding weight/property ratio constitute a driving force for development of modern steel grades. Complicated thermomechanical operations are applied to obtain highly sophisticated microstructures with combination of e.g. large grains, small grains, inclusions, precipitates, multi-phase structures etc. These microstructure features and interaction between them at the micro-scale level during manufacturing or exploitation stages eventually result in elevated material properties at the macro-scale level.

To support experimental research on these materials, a numerical material model that can take mentioned microstructure features explicitly into account during FE analysis of processing and exploitation conditions needs to be used. One of the

solutions to deal with the explicit representation of microstructure features during numerical analysis is an approach based on the Digital Material Representation (DMR) [C20]. However, there are two major issues that have to be addressed in this methodology. The first is development of algorithms for creation of structures that can represent real morphology of single and two phase microstructures [C19]. The second that is addressed in this dissertation is a problem of meshing of the created DMR as due to the nature of obtained microstructure, significant solution gradients (strain, stress etc.) are expected during numerical modeling. Robust and reliable algorithms capable of proper refinement of finite elements along mentioned microstructure features have to be developed. One of the solution is developed within the work space projection approach.

1.2.2 Application of space projections

Space projections constitute an important tool, which can be used in diverse applications including finite element (FE) analysis [C14, C15, C1].

The operator can be applied iteratively on a series of increasingly refined meshes, resulting in an improving fidelity of the approximation. A proof of concept for a limited set of applications has been presented in earlier author's works: [C16, C17, C18].

The main goal of this part of the work is to apply the full HP-adaptive algorithm to the projection operation in order to observe the predicted exponential convergence. This idea is validated on the basis of mentioned digital microstructures, modeling of which constitutes an important problem among the material science community.

1.2.3 Adaptive algorithms for solving weak forms of PDEs

A number of adaptive algorithms for finite element mesh refinements are known. HP-adaptation is one of the most complex and accurate, as it results in an exponential convergence with the number of degrees of freedom [C14].

The quality of the interpolation can be improved by the expansion of the interpolation base. In FEM terms, this could be done thanks to some kind of mesh adaptation. Two methods of adaptation are being considered in the present work:

- P-adaptation – increasing polynomial approximation level. One approach is to increase order of the basis functions on the elements where the error rate is higher than desired. More functions in the base means smoother and more accurate solution but also more computations and the use of high-order polynomials.
- H-adaptation – refining the mesh. Another way is to split the element into smaller ones in order to obtain finer mesh. This idea arose from the observation that the domain is usually non-uniform and in order to approximate the solution fairly some places require more precise computations than others, where the acceptable solution can be achieved using small number of elements. The crucial factor in achieving optimal results is to decide if a given element should be split into two parts horizontally, into two parts vertically, into four parts (both horizontally and vertically on one side), into eight parts (both horizontally and vertically on the both sides) or not split at all. That is why the automated algorithm that decides after each iteration for the element if it needs h- or p-refinement or not was developed. The refinement process is fairly simple in 1D but the 2D and 3D cases enforce a few refinement rules to follow.
- Automated hp-adaptation algorithm. Neither the p- nor the h-adaptation guarantees error rate decreases in an exponential manner with the size of mesh. This can be achieved by combining together mentioned two methods under some conditions, which are not necessarily satisfied in the present case. Still, in order to locate the most sensitive areas at each stage dynamically, and improve the solution as much as possible, the self-adaptive algorithm can be applied. It decides if a given element should be refined or it is already properly refined for the satisfactory interpolation, in an analogical manner to the algorithm for Finite Elements adaptivity described by [C14].

1.2.4 Application of continuous representation of material data

The continuous data approximation is necessary in case of a non-continuous input data (called material data) representing continuous phenomena. Some examples may involve:

- MRI scans of non-homogenous material to be used as material data for finite element method simulations
- satellite images of topography of the terrain, when we have a non-continuous bitmap data representing a generally continuous terrain;
- input data obtained by using various techniques representing temperature distribution over the material, where the temperature is mostly a continuous phenomena.
- situations when we solve the non-stationary problems of the form $\frac{\partial u}{\partial t} - \nabla K \nabla u = f$, where u represents temperature, with initial conditions $u(x,0) = u_0$, where u_0 is represented by a non-continuous input data, it is usually necessary to perform a H^1 projection of the u_0 to get the required regularity of u.

1.2.5 Multi-frontal direct solvers and graph grammar systems

Multi-frontal solvers are considered some of the most advanced direct solvers suited for solving linear systems of equations [C26, C27, C28]. Graph transformation systems have been previously used to model mesh generation and for multi-frontal solvers for example in [C29, C30, C31, C32, C33, C34].

The graph transformation system proposed by the above works for reuse of identical sub-trees has been also utilized for modeling mesh generation and adaptation with CP-graphs [C29, C30, C35] and hyper-graphs [C38]. This graph model can be also used for expressing the classical multi-frontal solver algorithm [C36, C37].

Finally, it is also possible to obtain the linear computational cost direct solver using some other topological features of the refined meshes [C39, C40].

1.3 Open problems and main scientific results

1.3.1 Open problems

- There are no self-adaptive algorithms for generation of continuous representation of material data based on three dimensional MRI scans.
- There are no numerical tests showing how the generation of continuous representation of material data based on discrete MRI scans improves the

convergence of the finite element method simulations in the field of material science.

- Traditional solvers deliver $O(N^2)$ computational cost when solving global three dimensional projection problem over regular 3D grids.
- There are no algorithms speeding up the multi-frontal solver algorithm over sub-parts of the computational mesh, when material data coefficients are identical and the geometry of the sub-meshes is identical.
- There is a graph grammar model of the classical multi-frontal solver algorithm. However, there is no graph grammar expression of the multi-frontal solver algorithm for multi-scale problems.

1.3.2 Main scientific results of the dissertation

- I have proposed the application of three dimensional fully automatic adaptive algorithms for generation of continuous representation of material data based on three dimensional MRI scans.
- I have performed a number of numerical experiments showing how generation of continuous representation of material data improves the convergence of the finite element method.
- I have proposed a linear computational cost adaptive algorithms for generation of continuous representation of material data based on three dimensional MRI scans, using projection based interpolation algorithm.
- I have derived the projection based interpolation algorithm for the case of three dimensional hexahedral and tetrahedral grids .
- I have proposed a new algorithm for re-use of identical parts of the mesh, with identical material data.
- I have expressed the multi-frontal solver algorithm allowing for reuse techniques with graph grammar productions which allow for dynamic construction of elimination trees for multi-scale problems.
- I have also proposed a framework for automatic switching of the scales from macro-scale to nano-scale during multi-scale simulations.

2 ALGORITHMS

In this section I present the proposed algorithms that solve the open problems enumerated above and constitute key findings of my PhD work.

2.1 SELF-ADAPTIVE ALGORITHM FOR SOURCE DATA PRE-PROCESSOR

2.1.1 SELF-ADAPTIVE ALGORITHM FOR SOURCE DATA PRE-PROCESSOR USING GLOBAL PROJECTION SOLVER

Material data pre-processor algorithm constructs three-dimensional coarse mesh along with base functions allocated on it, which enables constant approximation of discrete material data obtained e.g. from tomography.

Before entering the algorithm, we define three-dimensional finite elements, which the algorithm uses, on the basis of [A1]:

We start with defining one-dimensional base functions in the interval [0,1] (compare Figure 1).

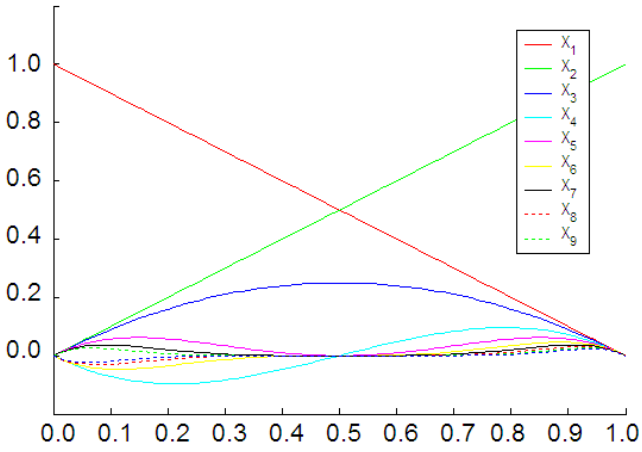

Figure 1 Base functions in the interval $[0,1]$

$$\hat{\chi}_1(\xi) = 1-\xi$$

$$\hat{\chi}_2(\xi) = \xi \qquad (1)$$

$$\hat{\chi}_3(\xi) = (1-\xi)\xi$$

$$\hat{\chi}_l(\xi) = (1-\xi)\xi(2\xi-1)^{l-3} \text{ for } l = 4, \dots, p+1$$

Subsequently, we define three-dimensional base functions connected with vertices, edges, faces and interiors of three-dimensional cubic elements. Vertex functions, which are related to 8 vertices of the element (compare Figure 2), are defined as tensor product of three one-dimensional linear functions:

$$\begin{aligned}
\hat{\phi}_1(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3)\\
\hat{\phi}_2(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3)\\
\hat{\phi}_3(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3)\\
\hat{\phi}_4(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3)\\
\hat{\phi}_5(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3)\\
\hat{\phi}_6(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3)\\
\hat{\phi}_7(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3)\\
\hat{\phi}_8(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3)
\end{aligned} \tag{2}$$

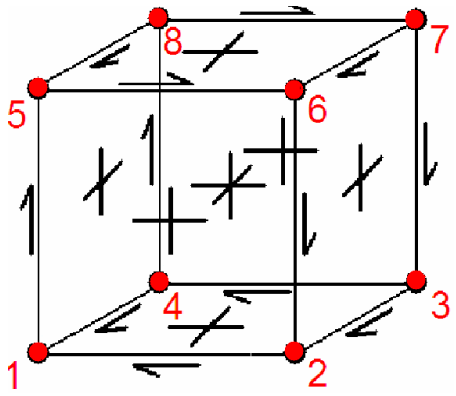

Figure 2 Base functions connected with vertices of the element

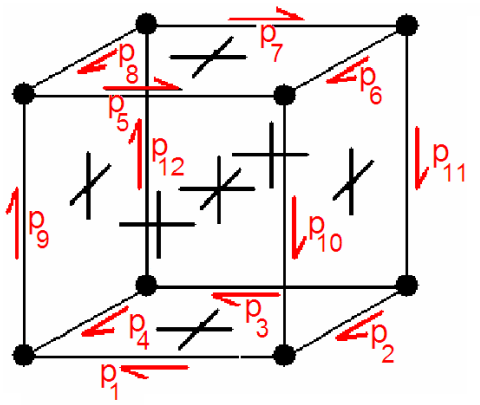

Figure 3 Base functions connected with edges of the element

Then we define functions related to edges of the element, as shown in Figure 3. They are defined as follows:

$$\begin{aligned}
\hat{\phi}_{9,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_1(\xi_3) \quad j=1,\ldots,p_1-1\\
\hat{\phi}_{10,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_1(\xi_3) \quad j=1,\ldots,p_2-1\\
\hat{\phi}_{11,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_1(\xi_3) \quad j=1,\ldots,p_3-1\\
\hat{\phi}_{12,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_1(\xi_3) \quad j=1,\ldots,p_4-1\\
\hat{\phi}_{13,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_2(\xi_3) \quad j=1,\ldots,p_5-1\\
\hat{\phi}_{14,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_2(\xi_3) \quad j=1,\ldots,p_6-1\\
\hat{\phi}_{15,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_{2+j}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_2(\xi_3) \quad j=1,\ldots,p_7-1\\
\hat{\phi}_{16,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_2(\xi_3) \quad j=1,\ldots,p_8-1\\
\hat{\phi}_{17,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j=1,\ldots,p_9-1\\
\hat{\phi}_{18,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_1(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j=1,\ldots,p_{10}-1\\
\hat{\phi}_{19,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j=1,\ldots,p_{11}-1\\
\hat{\phi}_{20,j}(\xi_1,\xi_2,\xi_3) &= \hat{\chi}_2(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+j}(\xi_3) \quad j=1,\ldots,p_{12}-1
\end{aligned} \tag{3}$$

In particular, it is possible to define p_i base functions on each edge, where p_i stands for polynomial approximation on the edge.

Then, we define functions related to faces of the element (compare Figure 4).

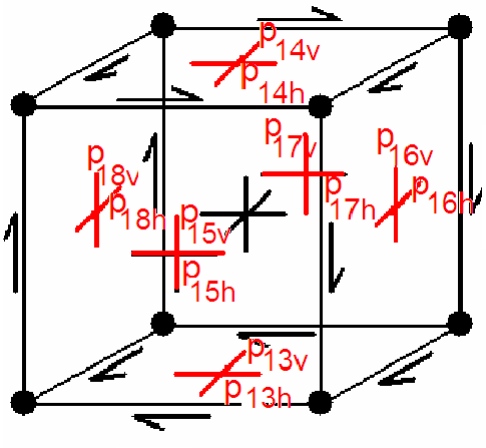

Figure 4 Base functions connected with faces of the element

$$\hat{\phi}_{21,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_{2+i}(\xi_1)\hat{\chi}_{1+j}(\xi_2)\hat{\chi}_1(\xi_3)$$
$$i=1,\ldots,p_{h1}-1,\ j=1,\ldots,p_{v1}-1$$

$$\hat{\phi}_{22,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_{2+i}(\xi_1)\hat{\chi}_{1+j}(\xi_2)\hat{\chi}_2(\xi_3)$$
$$i=1,\ldots,p_{h2}-1,\ j=1,\ldots,p_{v2}-1$$

$$\hat{\phi}_{23,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_{2+i}(\xi_1)\hat{\chi}_1(\xi_2)\hat{\chi}_{2+j}(\xi_3)$$
$$i=1,\ldots,p_{h3}-1,\ j=1,\ldots,p_{vv}-1 \qquad (4)$$

$$\hat{\phi}_{24,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_{2+i}(\xi_1)\hat{\chi}_2(\xi_2)\hat{\chi}_{2+j}(\xi_3)$$
$$i=1,\ldots,p_{h4}-1,\ j=1,\ldots,p_{v4}-1$$

$$\hat{\phi}_{25,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_1(\xi_1)\hat{\chi}_{2+i}(\xi_2)\hat{\chi}_{2+j}(\xi_3)$$
$$i=1,\ldots,p_{h5}-1,\ j=1,\ldots,p_{v5}-1$$

$$\hat{\phi}_{26,i,j}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_2(\xi_1)\hat{\chi}_{2+i}(\xi_2)\hat{\chi}_{2+j}(\xi_3)$$
$$i=1,\ldots,p_{h6}-1,\ j=1,\ldots,p_{v6}-1$$

Again, we have (pv_i, ph_j) base functions on each face, where pv_i and ph_j stand for approximation levels in vertical and horizontal directions.

Eventually, we define $px * py * pz$ base functions connected with interior of the element, see Figure 5.

$$\hat{\phi}_{27,i,j,k}(\xi_1,\xi_2,\xi_3)=\hat{\chi}_{2+i}(\xi_1)\hat{\chi}_{2+j}(\xi_2)\hat{\chi}_{2+k}(\xi_3)$$
$$i=1,\ldots,p_x-1,\ j=1,\ldots,p_y-1,\ k=1,\ldots,p_z-1 \qquad (5)$$

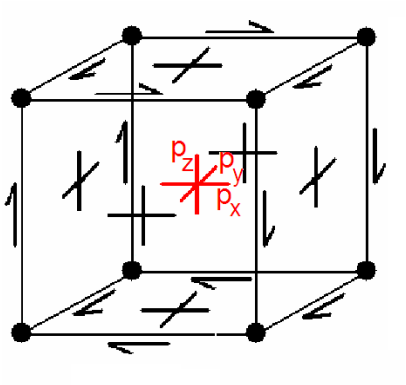

Figure 5 Base functions connected with interior of the element

After that, we define the problem of source data discrete projections using base functions $(a_i)_{i=1,\dots,dim\,V}$ allocated on three-dimensional mesh of cubic elements.

In particular, we define space V allocated on base functions $V = span\,\{(a_i)_{i=1,\dots,dim\,V}\}$.

In this subchapter, I refer to space L^2 and H^1 and their norms. Space L^2 is defined as $L^2 = \left\{ f: \Omega \to R : \int_\Omega |f^2| dx < \infty \right\}$. The case of L^2 projection in space $V_h \subset L^2$ can be referred to as the following optimization problem:

Given a function $f(x)$, find $u^V(x) \in V$, for which $\|f(x) - u^V(x)\|_{L^2(\Omega)}$ is minimum.

As $u^V(x) = \sum_{i=1}^{dim\,V} a_i\,\varphi_i(x)$ where $span(\{a_i\}_{i=1..dim\,V}) = V$, then we have to define $\{\varphi_i\}_{i=1..dim\,V}$, coefficients of linear combination of base functions.

Since $\|f(x)\|_{L^2(\Omega)} = \int_\Omega f(x)^2 d\Omega$, if we differentiate the equation over coefficients φ_j and compare the result with 0, in order to find minimum, we obtain:

$$\frac{\partial}{\partial \varphi_j}\left[\int_\Omega \left(f(x) - \sum_{i=1}^{\dim V} a_i\,\varphi_i(x)\right)^2 d\Omega\right] \tag{6}$$

which can later be converted into a matrix equation:

$$A \cdot \Phi = F \tag{7}$$

where:

$$A_{j,i} = \int_\Omega a_j(x) a_i(x) d\Omega \tag{8}$$

$$F_i = \int_\Omega f(x) a_i(x) d\Omega \tag{9}$$

in which the solution is a projection L^2 z $f(x)$ in the space V, marked as $u^V(x)$. The above method minimizes interpolation error square in function itself. Information on derivatives can be included through minimizing the sum error of the function as well as its gradients.

This approach is called the H^1 projection. Space H^1 is defined as $H^1 = \left\{ f: \Omega \to R : \int_\Omega |f'^2| dx + |f^2| < \infty \right\}$ and can be expressed by a similarly formulated problem:

Given function $f(x)$, find $u^V(x) \in V$ in which $\|f(x) - u^V(x)\|_{H^1(\Omega)}$ is minimum

which leads to:

$$\frac{\partial}{\partial \varphi_j}\left[\int_\Omega \left(f(x) - u^V(x)\right)^2 d\Omega + \beta \int_\Omega \left(\nabla f(x) - \nabla u^V(x)\right)^2 d\Omega\right] = 0 \qquad (10)$$

where β is a scalar parameter, which scales the part related to derivatives. Thus, the elements of equations system (1) take the following form:

$$A_{j,i} = \int_\Omega a_j(x) a_i(x) d\Omega + \beta \int_\Omega \nabla a_j(x) \cdot \nabla a_i(x) d\Omega \qquad (11)$$

$$F_i = \int_\Omega f(x) a_i(x) d\Omega + \beta \int_\Omega \nabla f(x) \cdot \nabla a_i(x) d\Omega \qquad (12)$$

Such a reformulated problem can be solved with any method of solving linear equations system.

The aforementioned projection problem is subsequently solved in every step of adapting algorithm, which generates coarse mesh approximating discrete source data.

```
1  function adaptive_fem(initial_mesh, desired_err, coef)
2  coarse_mesh = initial_mesh
3  repeat
4    coarse_u = solve the problem on coarse_mesh
5    fine_mesh = copy coarse_mesh
6    divide each element K of fine mesh into 8 new elements (K1 .. K8)
7    increase polynomial order of shape functions on each element of
8       fine mesh by 1
9    fine_u = solve the problem on fine_mesh
10   max_err = 0
11   for each element K of fine mesh do
12      K_err = compute relative decrease error rate on K
```

```
13      if K_err > max_err then
14         max_err = K_err
15      end if
16    end do
17    adapted_mesh = new empty_mesh
18    for each element K of coarse_mesh do
19      if K_err > coef * max_err then
20         choose a combination of refinements on element K
21            from fine_mesh to adapted_mesh
22      else
23         add K from coarse_mesh to adapted_mesh
24      end if
25   end do
26   coarse_mesh = adapted_mesh
27   output fine_u
28 until max_err < desired_err
29 return (fine_u, fine_mesh)
```

Algorithm 1 Algorithm of source data adaptive pre-processor

2.1.2 PROPERTIES OF SELF-ADAPTIVE ALGORITHM OF MATERIAL DATA ADAPTIVE PRE-PROCESSOR USING GLOBAL PROJECTION SOLVER

Properties of this algorithm were investigated in works [D8, D9]. This algorithm provides globally optimal approximation of function given at the beginning in the base of V, which is a subspace of H_1, in which basic computational problem is solved.

Presented projection algorithm was tested on three dimensional example. The example concerns the approximation of the material data representing 6 balls of one material inside another material. Such a case study can represent a simplified version of a two phase microstructure with clearly defined fractions of subsequent materials.

The initial microstructure morphologies as well as the resulting 3D grids and projection results are presented in Figure 6. The numerical results deliver exponential convergence rates summarized in Table 1. N denotes the number of degrees of freedom and the convergence is measured in the relative energy norm. These experiments imply also that three dimensional problems are difficult and even after several iterations of hp-adaptive code we reach 50% relative error, starting from almost 100% error. This proves the necessity for preprocessing of three dimensional material data by using adaptive algorithms.

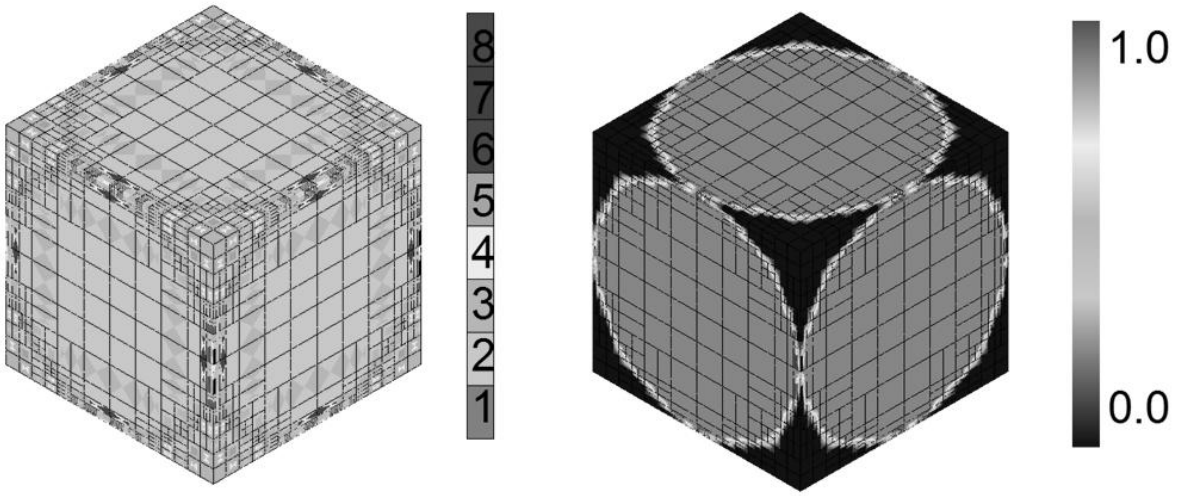

Figure 6 3D balls problem: mesh after the sixth iterations and solution over the mesh

step	N	$\frac{\lvert u_{hP}-u_{Hp}\rvert}{\lvert u_{hp}\rvert}$ [%]
1	125	71.3
2	2197	66.3
3	5197	62.4
4	12093	63.9
5	22145	57.7
6	41411	51.03

Table 1 Convergence rate for the problem of projections of 3D balls

The advantage of the algorithm lies in its simplicity, and in the fact that the same method can be used to solve the main problem (boundary problem) in pre-processing of source data (approximation problem).

Its main disadvantage is relatively high cost of computation (the time of single solver iteration). In particular, the following solution of the above system of equations (line 4 and 9) on three-dimensional mesh shall have complexity $O(N^2)$ [D3, D8, D9].The time of each step iteration in pre-processing algorithm is shorter than main

computational problem iteration only by the difference resulting from easier generation of equations system for source data pre-processing problem.

New pre-processor algorithm proposed by the author is described in next subchapter solves this problem.

2.1.3 New algorithm for adaptive source data pre-processor based on projection-based interpolation

The applied methodology of mesh pre-conversion with the use of interpolation with projection-based interpolation (PBI) was briefly introduced in [C10].

Initially, the PBI method was presented by L. Demkowicz in order to control hp-adaptation (for optimal adaptation selection on particular finite elements). The author summarizes algorithm modifications applicable for solution of a global projection problem in material data pre-processing.

The main goal of PBI is to find a continuous representation of U in the base $(\varphi_i)_{i=1..\dim V}$ of space $V \subset H_0^1(\Omega)$. In other words, we are looking for coefficients $\{a_i\}_{i=1..\dim V}$ of a linear combination $u^V = \sum_{i=1}^{\dim V} a_i \varphi_i$, which interpoles function on vertices ($u^V(v) = U(v)$ for each vertex v) and minimizes the norm H_0^1 of a difference between the interpolating function and the interpolated one $\|U - u^V\|_{H_0^1(\Omega)} \rightarrow \min$ on the remaining nodes.

This problem could be solved as a global system of equations in cubic time with the number of unknowns, or – regarding complex structure of the problem – by the use of a Finite Element (FEM) solver in square time. Instead, I propose the use of algorithm with linear complexity, which provides globally suboptimal approximation, which, however, is sufficient to further solution conversion as source data for the Finite Element solver.

As the main assumption is that in the end of the PBI step, there is a mesh compatible with the FEM solver, used in the later step in paper outline, then in the PBI step we also use the mesh composed of finite elements based on base functions used in the FEM solver. The PBI algorithm consists of the following sub-stages, connected respectively with vertices, edges, faces and interiors. For each finite element, there are a_i coefficients indicated in the appropriate order.

First the vertices are investigated, since their coefficients are the most straightforward to compute. There is only one function per vertex with a support on it and the interpolating function is required to be equal to the interpolant which yields:

$$a_{v_i} = \frac{U(v_i)}{\varphi_i(v_i)} \forall_{i=1..8} \tag{13}$$

On nodes other than vertices, the input function cannot be represented exactly, so instead of approach to minimize the representation error is employed. First, on each one of the 12 edges:

$$\left\| \left(U - \sum_{j=1}^{8} u_{V_j} \right) - \sum_{l=1}^{\dim V|e_i} u_{l,e_i} \right\|_{H_0^1(e_i)} \to \min \quad \forall_{i=1..12} \tag{14}$$

where $\dim V|e_i$ signifies the number of edge shape functions in space V with supports on edge e_i. Such a problem can be reduced to a linear system and solved with a linear solver, but if we assume the adaptation order $p = 2$ on each node, for each edge there exists only one shape function with a support on it. Not only is this restriction justified performance-wise (one local equation instead of a system), but it also suffices in most cases, according to my experiments. Thus equation (10) reduces to:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{8} u_{V_j} \right)}^{\dot{U}} - u_{0,e_i} \right\|_{H_0^1(e_i)} \to \min \quad \forall_{i=1..12} \tag{15}$$

where $\dot{U}$ vanishes on the element's vertices. After rewriting the norm:

$$\sqrt{\int_{e_i} \sum_{k=1}^{3} \nabla \cdot \left(\dot{U} - u_{0,e_i} \right)^2 dx} \to \min \quad \forall_{i=1..12} \tag{16}$$

$$\int_{e_i} \sum_{k=1}^{3} \left(\frac{d\dot{U}}{dx_k} - \frac{du_{0,e_i}}{dx_k} \right)^2 dx \to \min \quad \forall_{i=1..12} \tag{17}$$

and as $u_{0,e_i} = a_{e_i} \varphi_i$ where a_{e_i} is constant:

$$\int_{e_i} \sum_{k=1}^{3} \left(\frac{d\dot{U}}{dx_k} \right)^2 dx - 2 \int_{e_i} \sum_{k=1}^{3} \frac{d\dot{U}}{dx_k} \frac{du_{0,e_i}}{dx_k} dx + \int_{e_i} \sum_{k=1}^{3} \left(\frac{du_{0,e_i}}{dx_k} \right)^2 dx \to \min \quad \forall_{i=1..12} \tag{18}$$

which leads to:

$$\int_{e_i} \sum_{k=1}^{3} \frac{du_{0,e_i}}{dx_k} \frac{du_{0,e_i}}{dx_k} dx - 2 \int_{e_i} \sum_{k=1}^{3} \frac{d\dot{U}}{dx_k} \frac{du_{0,e_i}}{dx_k} dx \to \min \quad \forall_{i=1..12} \tag{19}$$

since $\int_{e_i} \sum_{k=1}^{3} \left(\frac{d\dot{U}}{dx_k}\right)^2 dx$ is constant and can be omitted in minimization.

Let be $b(u, v)$ a bilinear, symmetric form defined as:

$$b_\Omega(u, v) = 2 \int_\Omega \sum_{k=1}^{3} \frac{du}{dx_k} \frac{dv}{dx_k} dx \tag{20}$$

and $l(u)$ be a linear form:

$$l_\Omega(v) = 2 \int_\Omega \sum_{k=1}^{3} \frac{d\dot{U}}{dx_k} \frac{dv}{dx_k} dx \tag{21}$$

It is proven that minimizing $\frac{1}{2} b_\Omega(u, u) - l_\Omega(u)$ is reducible to solving $b_\Omega(u, v) = l_\Omega(v) \forall_{v \in V}$. By applying this lemma to problem (19) for $u = a_{e_i} u_{0,e_i}$, we obtain:

$$2 \int_{e_i} \sum_{k=1}^{3} \frac{du_{0,e_i}}{dx_k} \frac{dv}{dx_k} dx = 2 \int_{e_i} \sum_{k=1}^{3} \frac{d\dot{U}}{dx_k} \frac{dv}{dx_k} dx \ \forall_{i=1..12} \forall_{v \in V} \tag{22}$$

and finally as $u_{0,e_i} = a_{e_i} \varphi_i$ and $\varphi_i \in V$:

$$a_{e_i} = \frac{\int_{e_i} \sum_{k=1}^{3} \frac{d\dot{U}}{dx_k} \frac{d\varphi_{e_i}}{dx_k} dx}{\int_{e_i} \sum_{k=1}^{3} \frac{d\varphi_{e_i}}{dx_k} \frac{d\varphi_{f_i}}{dx_k} dx} \ \forall_{i=1..12} \tag{23}$$

The next step consists in an optimization on faces:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{8} u_{V_j} - \sum_{j=1}^{12} u_{0,e_j}\right)}^{\ddot{U}} - u_{0,f_i} \right\|_{H_0^1(f_i)} \to \min \quad \forall_{i=1..6} \tag{24}$$

where $\ddot{U}$ vanishes on vertices and edges. This leads to:

$$a_{f_i} = \frac{\int_{f_i} \sum_{k=1}^{3} \frac{d\ddot{U}}{dx_k} \frac{d\varphi_{f_i}}{dx_k} dx}{\int_{f_i} \sum_{k=1}^{3} \frac{d\varphi_{f_i}}{dx_k} \frac{d\varphi_{f_i}}{dx_k} dx} \ \forall_{i=1..6} \tag{25}$$

Finally, an analogical optimization in the interior of the finite element:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{8} u_{V_j} - \sum_{j=1}^{12} u_{0,e_j} - \sum_{j=1}^{6} u_{0,f_j}\right)}^{\ddot{U}} - u_{0,I} \right\|_{H_0^1(I)} \to \min \tag{26}$$

(where $\ddot{U}$ vanishes everywhere except from the interior) yields:

$$a_I = \frac{\int_I \sum_{k=1}^{3} \frac{d\ddot{U}\, d\varphi_I}{dx_k dx_k} dx}{\int_I \sum_{k=1}^{3} \frac{d\varphi_I d\varphi_I}{dx_k dx_k} dx}$$

The algorithm specified above is applicable, instead of traditional one, to solving a projection task in lines 4 and 9.

This algorithm can also be easily paralleled by the use of OpenMP mechanism, which was the subject of work [D19].

```
1  #pragma omp parallel for
2  for vertex in (vertices of elements of the mesh)
3     compute values of coefficient a_{v_i} given by (13) for PBI for vertex
4     store a_{v_i} at vertex
5  #pragma omp parallel for
6  for edge in (edges of elements of the mesh)
7     compute values of coefficient a_{e_i} given by (23) for PBI for edges
8     store a_{e_i} at edge
9  #pragma omp parallel for
10 for face in (faces of elements of the mesh)
11    compute values of coefficient a_{f_i} given by (25) for PBI for face
12    store a_{f_i} at face
13 #pragma omp parallel for
14 for interior in (interiors of elements of the mesh)
15    compute values of coefficient a_I given by (27) for PBI for  interior
16    store a_I at interior
```

Algorithm 2 Modification of lines 4 and 9 in the algorithm of source data adaptive pre-processor

It is worth mentioning, that application of this method does not construct any global equation systems but merely singular and individually independent equations for particular mesh nodes.

Figure 7 presents an exemplary usage of the PBI algorithm for continuous representation of a human head in order to carry out simulation of propagation of electromagnetic waves, which are emitted from a mobile phone antenna [D11, D4], whereas Figure 8 shows simulation of human head heated by the phone on

generated mesh, by means of the Pennes equation, under the assumed radiation of the cell-phone antenna. For more detais we refer to [D11,D4].

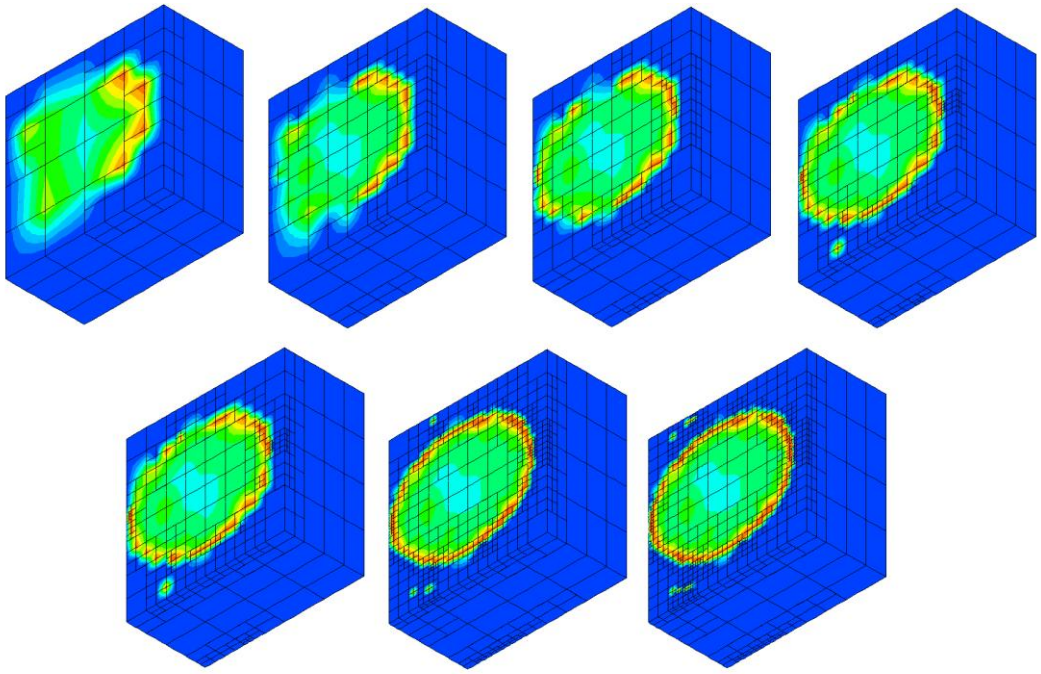

Figure 7 Sequence of meshes generated by the adaptive algorithm for continuous representation of human head

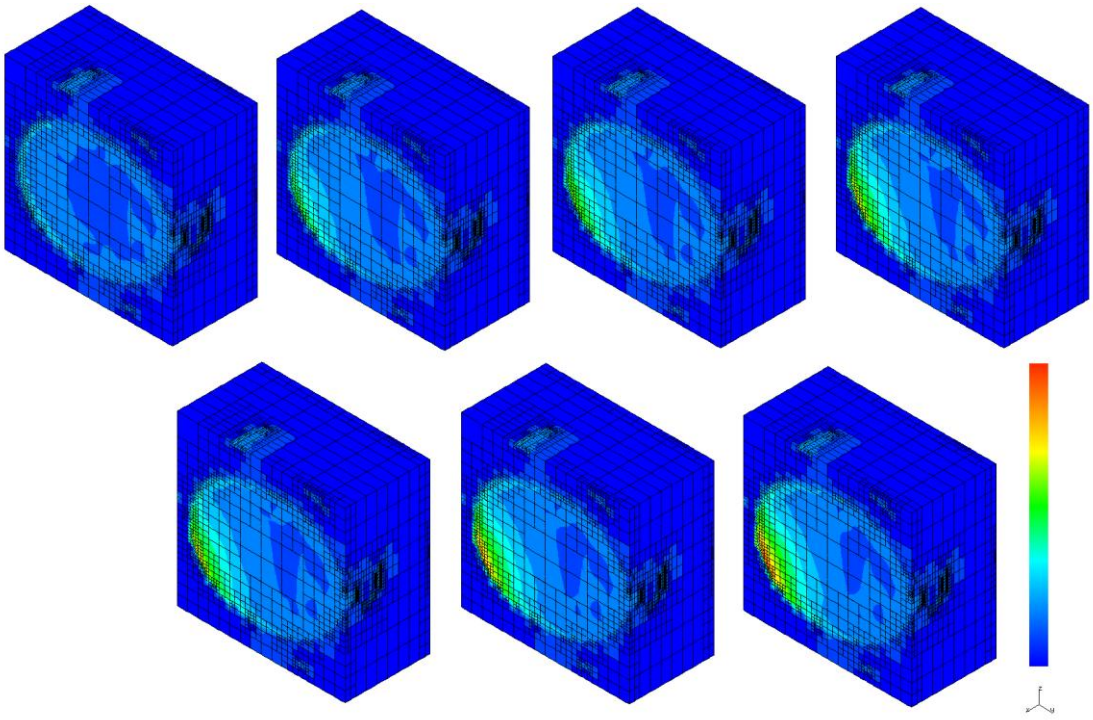

Figure 8 Computations sequence from particular simulation steps of human head heated by the phone [D11,D4]

2.1.4 Properties and applicability of the modified algorithms of material data adaptive pre-processor

The main advantage of the algorithm of material data adaptive pre-processor (Algorithm 2) is linear complexity O(N) in sequential version and complexity O(N/c) in parallel version, where c stands for number of cores.

PBI method has been investigated in papers [D3, D10, D5, D19, D13]. In contrast with the method described in 2.1.1, the PBI method does not provide globally optimal approximation, since optimization acts locally for each finite element separately. However, local optimization requires constant amount of operations on each element of the mesh (vertex, edge, face or interior), which means, that one PBI iteration has linear complexity with the number of finite elements (not square number as it occurs in method introduced in 2.1.1).

As one can conclude from the results obtained in [D3] for the investigated problems, nearly every iteration of h-adaptive PBI method reduces the number of more expensive iterations of h-adaptive method of finite elements, which results in a substantial minimization of computational costs.

Also the local character of PBI method has significant implications – it enables much easier, than in the case of the method investigated in 2.1.1, parallelization algorithm with the use of agent techniques (which was conducted in [D5]).

2.1.5 Numerical experiment

The applicability of the method has been tested on the following example. I was looking for the solution of the linear elasticity problem with thermal expansion coefficient over the three dimensional non-regular material [D3], presented in Figure 9. For derivation of the weak formulation of the linear elasticity with thermal expansion coefficient we refer to Appendix A.

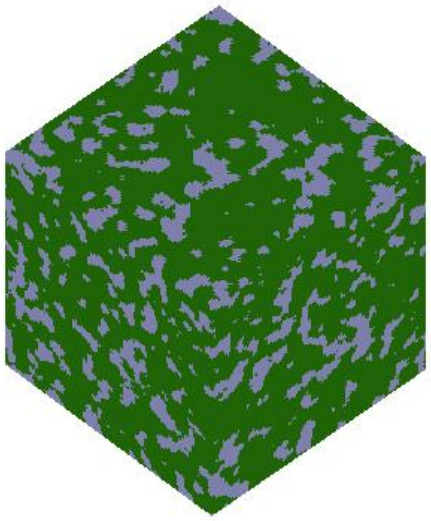

Figure 9 Digital representation of a two-phase material [D3]

We compare efficiency of two numerical algorithms based on the conventional and modified workflows for solving linear elasticity problem with thermal expansion coefficient [C7]. A complex dual phase steel under thermal loading conditions is selected for investigation, as these steels represent modern materials often used in automotive industry. The problem of thermal loading and related thermal stresses is often underestimated as usually material behavior under mechanical deformation is considered in metal forming. However, due to higher demands for better in use properties under exploitation conditions, the issue of thermal loading needs to be precisely investigated especially when complex multiphase materials are considered. Although multiphase materials can provide highly elevated properties, due to the fact that these materials are often a combination of several significantly different materials/phases, the local deformation inhomogenities can occur and they influence macro scale behavior of a final product. As mentioned, to model these inhomogenities, the digital material representation can be incorporated into the sophisticated h-adaptation finite element model by using conventional and modified workflow approaches.

2.1.5.1 BASIC WORKFLOW

To simulate the micro scale behavior of a complex materials under deformation, one needs typically at least two steps. In the first, a digital representation of the material (DMR) with the required parameters is required. State of the art in the DMR models

is presented in the following section. In the second step of a work flow, an advanced h-adaptation finite element method is applied to precisely investigate material behavior.

Figure 10 The conventional two-stage workflow that uses an h-FEM approach

This conventional two stage workflow can be also modified in order to obtain higher computational efficiency by incorporating more stages. One of the possibilities of such a modification is proposed as a result of the present research.

2.1.5.2 PROPOSED WORKFLOW IMPROVEMENT

However, such a conventional two stage work flow has limitations mainly related with the h-adaptive Finite Element method. Despite the fact that this method is very precise and converges exponentially, unfortunately it is also quite time-consuming. Each iteration of this algorithm requires solving the linear elasticity on two meshes, each in $O(N^2)$ asymptotic time with N being proportional to the number of finite elements. Previous experiments show that usually a few initial steps of the Finite Element algorithm focus on reproducing the material structure with the computational mesh[21], regardless of the boundary problem solved. This observation leads to the question, whether these iterations could be replaced with a less expensive pre-processing. Thus, the work flow approach was extended by a third intermediate stage as seen in Figure 11.

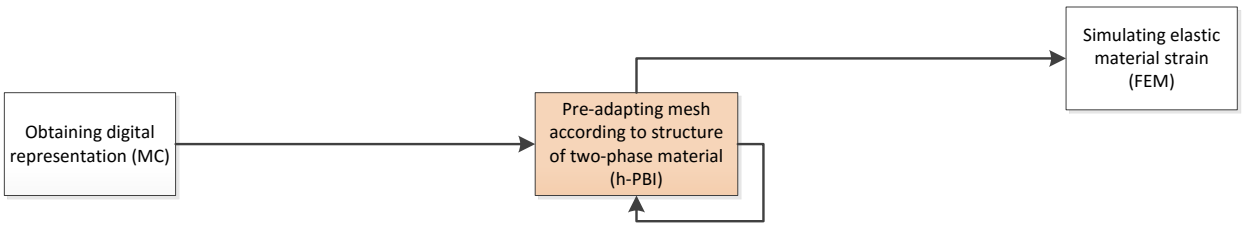

Figure 11 The modified workflow, a new intermediate stage marked in orange

The Projection-Based Interpolation method comes to the aid in the case of reducing computational time. Although unaware of the elasticity problem to be solved, this method adapts the computational mesh around the interface between material phases and around boundaries with linear rather than quadratic computational complexity. After a couple of steps, Finite Elements still need to be used to adapt the mesh around solution peculiarities (problem-specific!) and to output the actual solution of the boundary problem, but the process is much shorter now. The detailed algorithm behind the PBI is described below.

2.1.5.3 COMPARISON OF THE METHODS IN 3D

In the first investigated case (traditional workflow), 7 iterations of the h-adaptive algorithm were required. Subsequent mesh sizes are given in Table 2. A series of results for subsequent iterations is presented for X, Y and Z axis in Figure 13, Figure 14 & Figure 15 respectively. The scale for all the figures is shown in Figure 12.

h-FEM iteration #	size of the coarse mesh (along a single axis)	size of the fine mesh (along a single axis)	elasticity error estimate
1	125	1000	106.8249
2	729	5832	65.8873
3	1331	9261	46.9618
4	7793	58833	41.1811
5	16377	130289	40.5419
6	75997	630865	35.8563

Table 2 Mesh sizes and error decrease rate for pure h-FEM

Figure 12 Scale used in subsequent figures

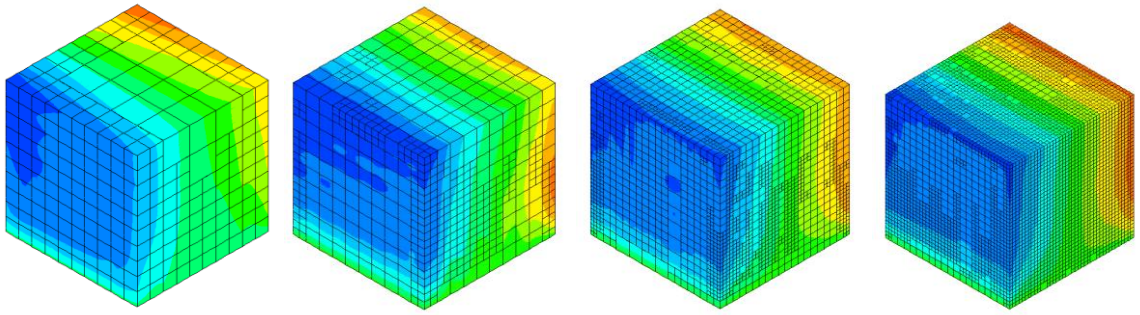

Figure 13 X component of the thermal deformation vector of the squeezed dual phase material from Figure 9 (last 4 iterations of a pure h-FEM algorithm)

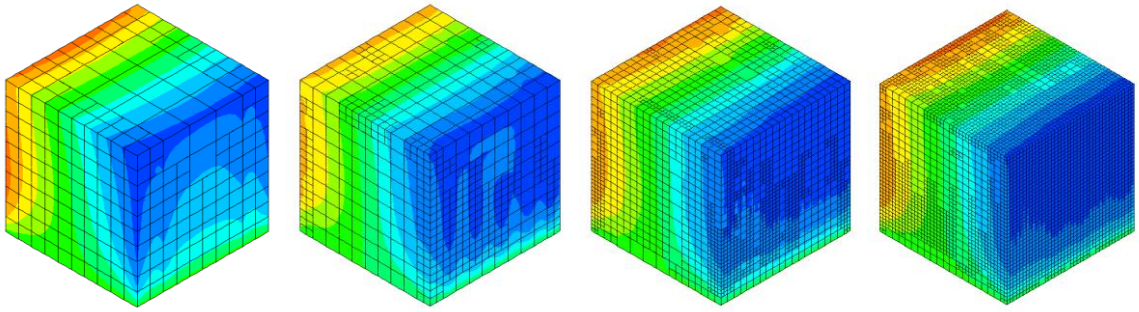

Figure 14 Y component of the thermal deformation vector of the squeezed dual phase material from Figure 9 (last 4 iterations of a pure h-FEM algorithm)

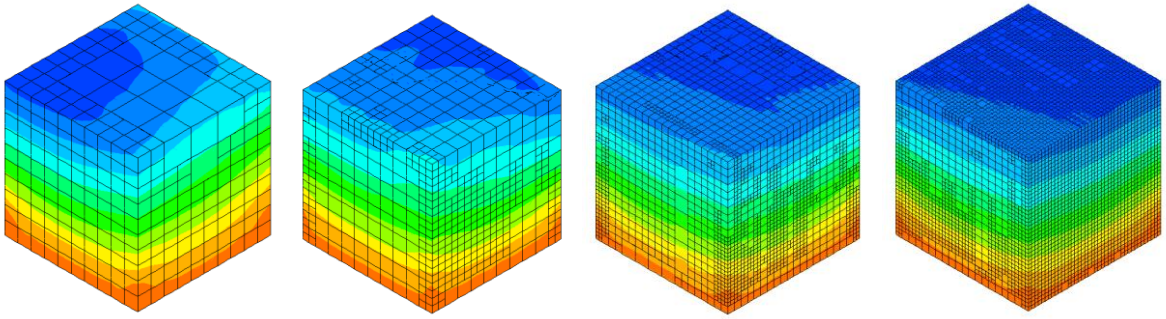

Figure 15 Z component of the thermal deformation vector of the squeezed dual phase material from Figure 9 (last 4 iterations of a pure h-FEM algorithm)

Each iteration incurred constructing and solving the linear elasticity problem on two meshes - coarse and fine. Solving such a problem with FEM requires $O(N^2)$ operations over the regular 3D mesh, and between $O(N)$ and $O(N^2)$ on highly refined 3D mesh, depending on the topology of the mesh [C44]. The error decrease is illustrated in Figure 16.

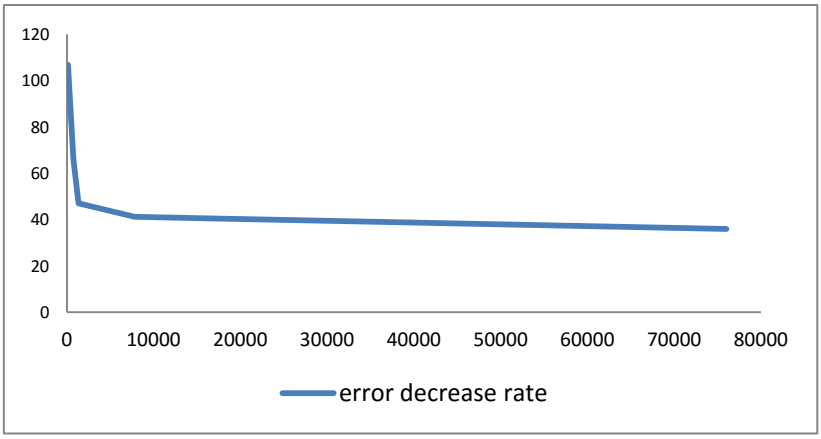

Figure 16 Error decrease for pure h-FEM

In the second case study (modified work flow), first the mesh which well approximates the DMR was generated using the adaptive PBI procedure, as presented in Figure 17.

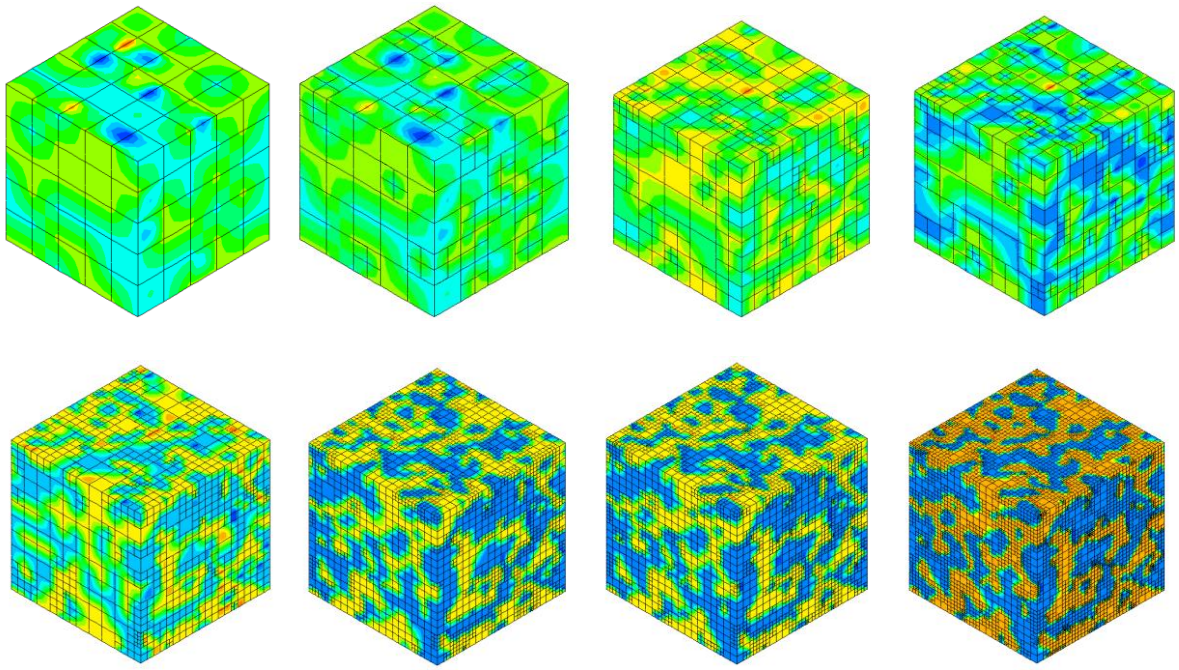

Figure 17 A series of pre-adapted meshes along with their corresponding approximations of the initial microstructure from Figure 9 in subsequent steps of PBI

Table 3 illustrates the error of the elasticity solution after pre-adapting mesh with 1-5 PBI iterations. Figure 18, Figure 19 & Figure 20 illustrate the numerical results.

h-PBI iteration #	size of the coarse mesh (along a single axis)	size of the fine mesh (along a single axis)	elasticity error estimate (for a single FEM iteration after the PBI)
1	3407	26997	42.46
2	8457	67913	38.63
3	10187	83301	38.51
4	25961	210753	37.87
5	36233	602631	35.87

Table 3 Mesh sizes and error decrease rate for h-PBI+FEM

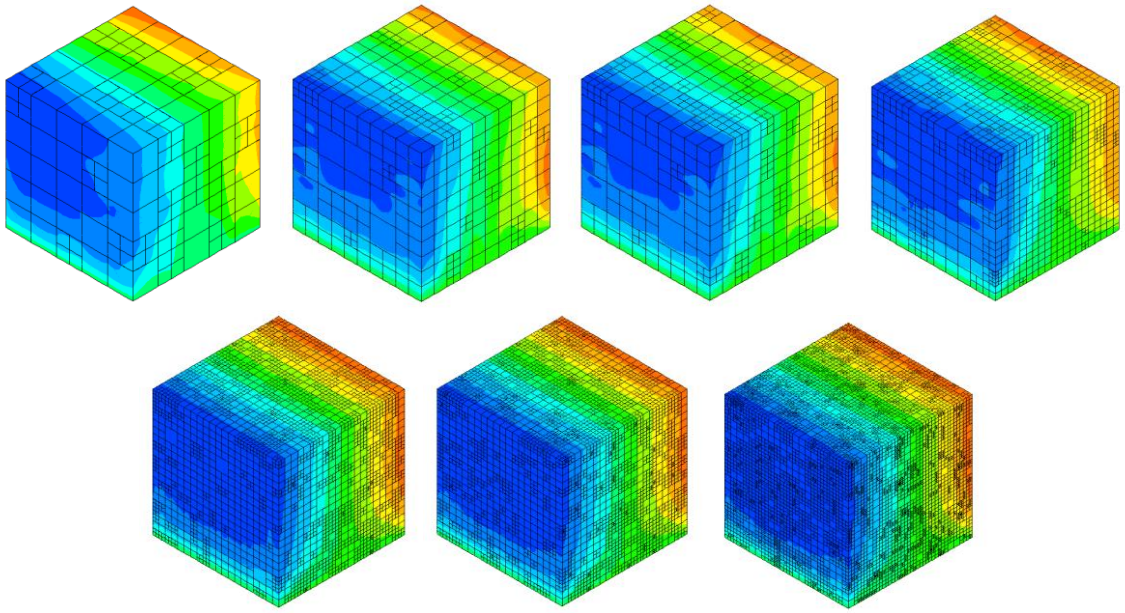

Figure 18 X component of the thermal deformation vector of the squeezed dual phase material from Figure 9 for h-PBI + FEM

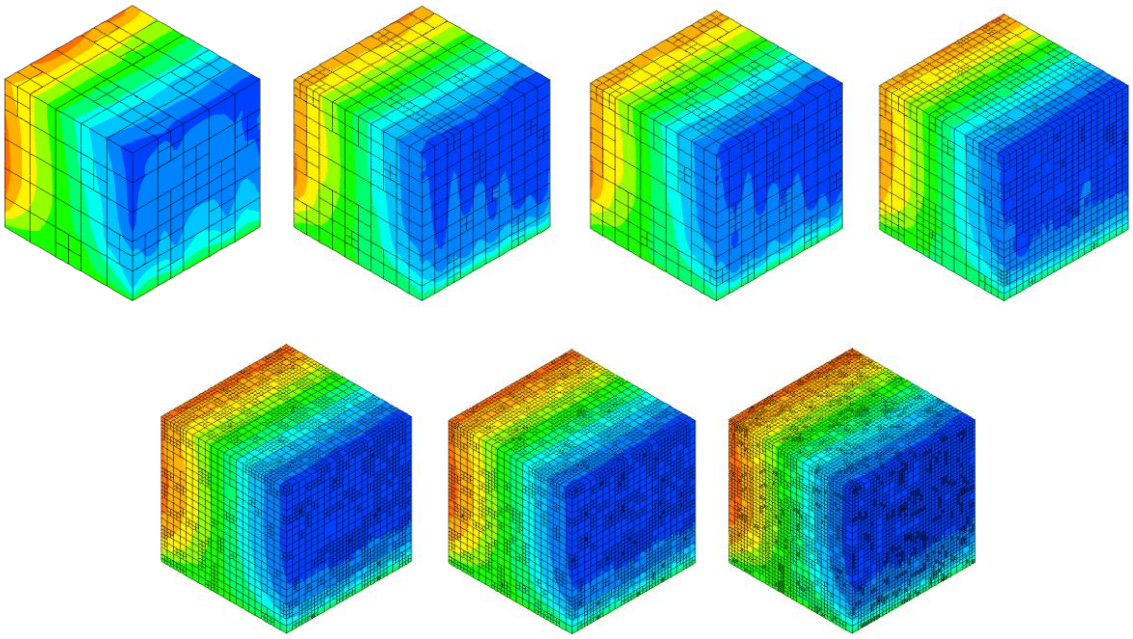

Figure 19 Y component of the thermal deformation vector of the squeezed dual phase material from Figure 9 for h-PBI + FEM

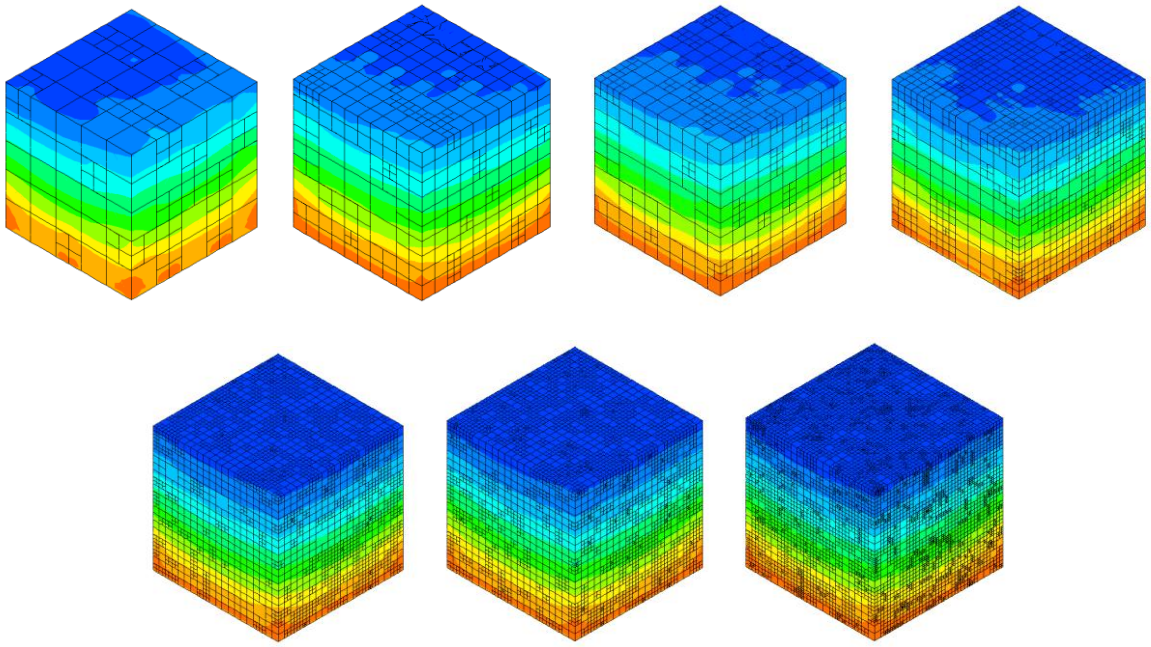

Figure 20 Z component of the thermal deformation vector of the squeezed dual phase material from Figure 9 for h-PBI + FEM

The proposed hybrid method not only delivered the same precision (error estimate of ~35) but it also required less iterations (5 vs. 7) of a cheaper ($O(N)$) algorithm. This hybrid technique required a total of $5 * O(N) + O(N^2)$ to achieve the same precision as the pure FEM. Moreover if the precision of ~40 is acceptable, the benefit from this optimization is even more prevalent - pure h-FEM would have required 5 steps to get

there, while the hybrid method achieved a superior precision after 2 steps of h-PBI and 1 step of the FEM.

2.1.5.4 COMPARISON OF THE METHODS IN 2D

As a proof of concept we also tested our solution on a few 2D bitmaps representing selected microstructures described below. Three kinds of digital material representations have been created for the purpose of the current experiment. The first one depicts a commonly used simplified representation of a two phase microstructure. Using this approach, an influence of different volume fractions of hard martensitic phase in soft ferritic matrix can be investigated. The second one represents a single phase polycrystalline microstructure, where subsequent grains are clearly distinguished. The third one represents a multi-layer composite material.

To properly capture grain morphology with homogenous FE meshes, very fine finite element meshes have to be used. This, in turn, leads to excessive computational time. The finer the FE mesh is, the better description of the phase boundary shape is obtained. With coarser meshes some microstructure features can be even neglected.

However, to reduce computational time and maintain high accuracy of the solution along mentioned phase/grain boundaries, proposed hp adaptation technique can be used to obtain specific non-uniform FE meshes that are refined along the phase/grain boundaries.

Since the described algorithm operates over [0, 1]×[0, 1] square, each bitmap required some extra preprocessing. We converted .bmp files into input .dat files by performing the following steps:

• Since the value of material data is encoded in the bitmap's luma Y , we obtain it according to the formula below:

$$Y = 0.299 \times Red + 0.587 \times Green + 0.114 \times Blue$$

This kind of encoding is frequently used, as it makes visually neighbouring pixels to be neighbours also numerically.

• Normalize the output value set to the interval [0, 1].

• Save bitmap as a 2D matrix of floating-point numbers.

• Compute value of f(x, y):(x, y) ∈ [0, 1] × [0, 1] by scaling and mapping (x, y) to the nearest pixel on the bitmap.

• Compute value of $\partial f(x,y)/\partial x$ and $\partial f(x,y)/\partial y$ using differential quotient.

The hp-adaptive algorithm is executed for 20 iterations until the error decrese rate falls below the desired 1%. Behaviour of the mesh after 10 and 20 iterations for all three cases has been depicted in Figures 22, 23 and 24. In the first case, adaptation focuses on 5 squares surrounding the distinct pieces of a different material. As the adaptation proceeds, the interface between materials becomes visible more clearly. The refinement of meshes is also reflected in the solutions. The results after 20 iterations for the second and third microstructure have been presented in Figures 25 and 26. These figures may be thought of as normalized continuous representations of the initial microstructure images (Figure 21).

Figure 21 Microstructure images used for computations

For the first bitmap we obtained an acceptable solution only after 10 iterations. This is presented on Figure 25. All microstructure images passed to the algorithm had a few features which made them especially tricky to interpolate continuously – namely irregular shapes, sharp edges and a wide range of values. While shapes turned out to be reflected in quite a faithful manner with h-adaptation, sharp edges required elevated degrees of p-adaptation. This, by nature, leads to the Runge effect, where interpolant oscillates at the edges of an interval. To avoid this effect, one should supply the upper bound for p-adaptation level in some applications.

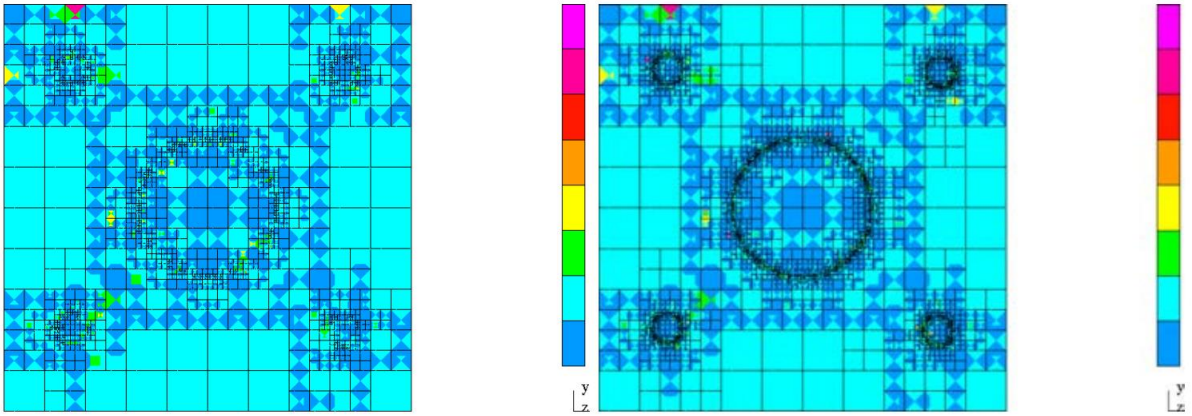

Figure 22 Mesh density and p-refinement levels after 10 and 20 iterations for the first microstructure

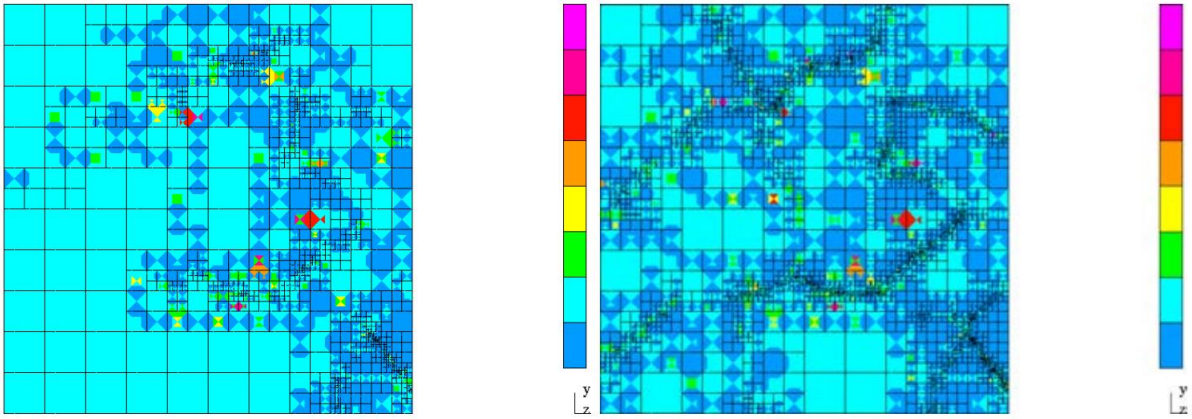

Figure 23 Mesh density and p-refinement levels after 10 and 20 iterations for the second microstructure

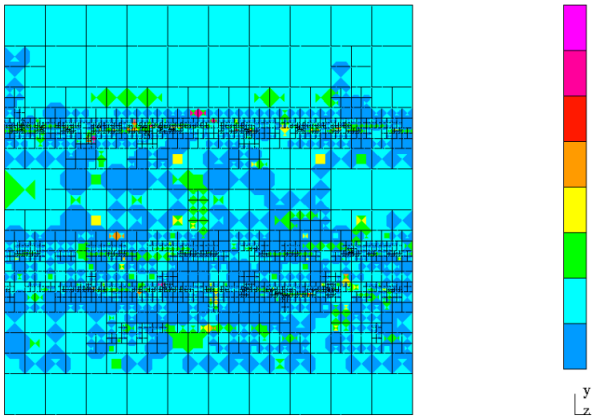

Figure 24 Mesh density and p-refinement levels after 30 iterations for the third microstructure

The obtained meshes can be used as an input for the FE modeling of material behavior during processing or exploitation stages. We tested the feasibility of this application using examples 2 and 3, for which we consider the heat transfer and linear elasticity problems respectively. In general, the applicability of the grids for the finite element method simulations depends on the sensitivity of the physical phenomena to the changes of material properties.

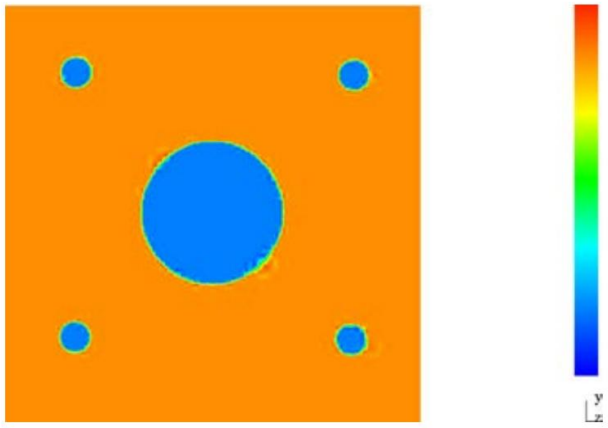

Figure 25 The approximation of material data after 10 iterations for the first microstructure

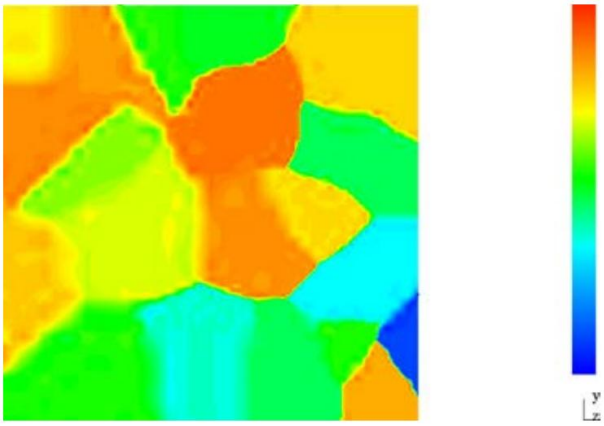

Figure 26 The approximation of material data after 20 iterations for the second microstructure.

First test concerned the heat transfer problem. We solved the heat transfer equation over the second mesh with Dirichlet boundary condition on the bottom and Neuman boundary condition on top, left and right sides. We assumed that the heat transfer coefficient K changes for different material. We also assumed that the temperature

over the Dirichlet boundary varies with different materials. From the numerical results presented in Figure 27 it follows that the heat transfer is not sensitive to the changes in material data. The adaptivity are only necessary for the Dirichlet boundary.

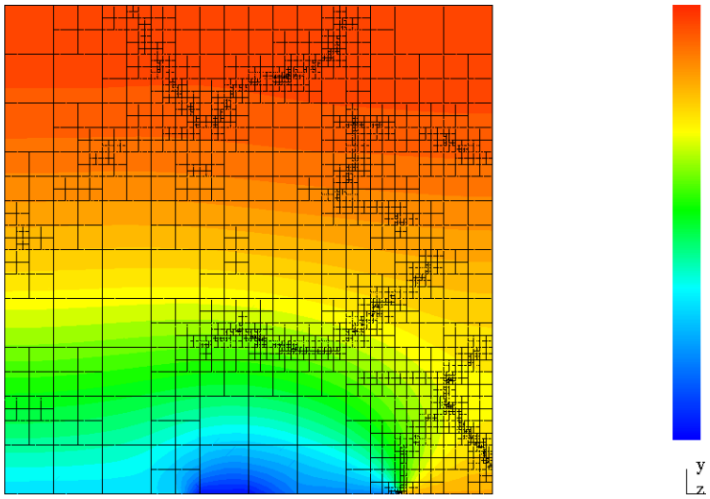

Figure 27 The solution to the heat transfer problem over that mesh (temperature scalar field).

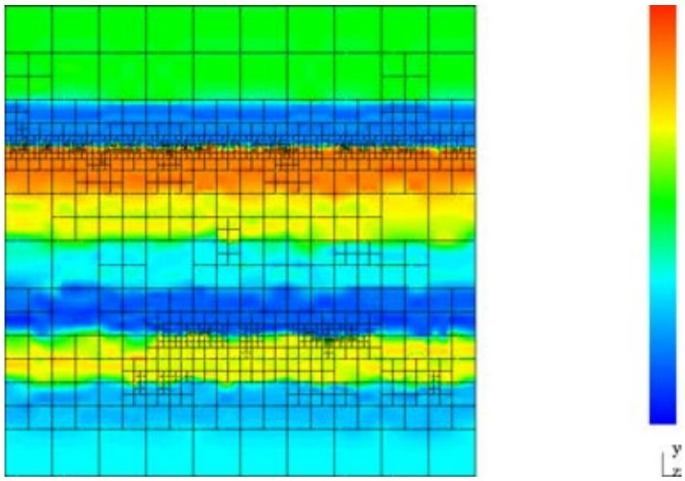

Figure 28 The approximation of material data after 20 iterations for the third microstructure.

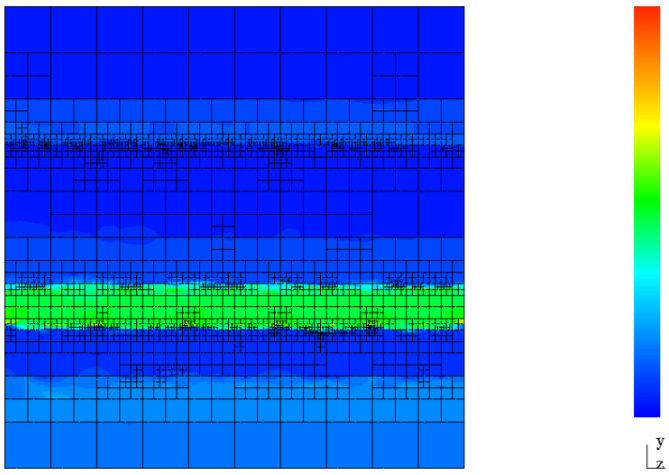

Figure 29 The solution to the linear elasticity problem (norm of the displacement vector) over a further refined mesh.

The second numerical problem concerned the linear elasticity. We subjected the third multi-layered mesh to the crushing forces. Namely, we assumed fixed zero Dirichlet boundary conditions on the bottom, the crushing force on the top, with zero Neumann boundary conditions on both sides. We assumed different Young modulus for different layers denoted by different colors. Our adaptive procedure has improved the accuracy of the linear elasticity problem significantly. The linear elasticity problem solved on the uniform unrefined grid suffers from 50 percent numerical error (measured as relative error in H1 norm). The same problem solved over our hp adapted mesh (Figures 28 and 29) delivers 10 percent numerical error. In other words our adaptive procedure has reduced the numerical error almost one order of magnitude.

Finally, in Figure 30 we present exponential convergence curves for the first and the second case.

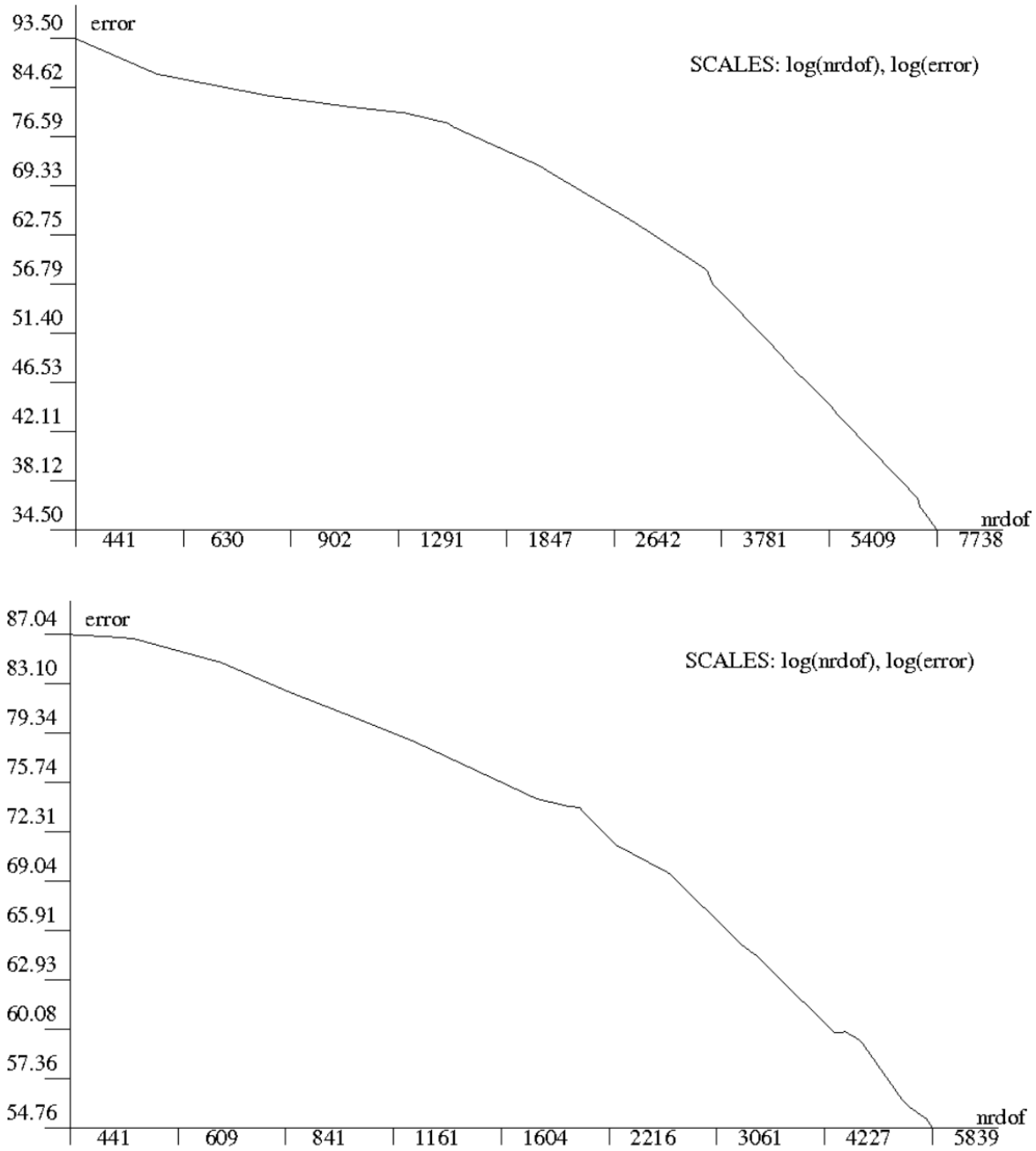

Figure 30 Convergence curves, nrdof – number of degrees of freedom, error – absolute error decrease rate in terms of H1 norm.

Since they are drawn in logarithmic scale, error decrease rate is virtually exponential with respect to the number of degrees of freedom (marked as ”nrdof”). In both cases it took the similar amount of required steps to obtain a satisfactory solution.

2.2 GENERALIZATION ON MESHES WITH FOUR-FACED ELEMENTS

The algorithm of material data adaptive pre-processor was designed and implemented for mesh with four-faced elements [D1, D2, D19]. The necessary modifications to the derivation procedure are listed below:

The location of material data inside the human head model has been obtained by using the projection based interpolation algorithm that consists the following sub-steps, related to tetrahedral finite element vertices, edges, faces and interiors. For each finite element, we are looking for a_i a_i coefficients in a particular order. The computational mesh can be generated by using a linear computational cost projection based interpolation routine, first proposed by [4]. We start with vertices, since their coefficients are the most straightforward to compute. There is only one function per each of four vertexes with a support on it and the interpolating function is required to be equal to the interpolant which yields

$$a_{v_i} = \frac{U(v_i)}{\varphi_i(v_i)} \quad \forall i = 1,...,4 \tag{26}$$

On nodes other than vertices, the input function cannot be represented exactly, so instead we are trying to minimize the representation error. First, on each one of the 6 edges of tetrahedral element:

$$\left\| \left(U - \sum_{j=1}^{4} u_{v_i} \right) - \sum_{l=1}^{\dim V|_{e_i}} u_{l,e_i} \right\|_{H_0^1(e_i)} \to \min \quad \forall i = 1,...,6 \tag{27}$$

where $\dim V|e_i$ $\dim V|e_i$ signifies the number of edge shape functions in space V V with supports on edge e_i. Such a problem can be reduced to a linear system and solved with a linear solver, but if we assume the adaptation order $p = 2$ on each node, for each edge there exists only one shape function with a support on it. Not only is this restriction justified performance-wise (one local equation instead of a system), but it also suffices in most cases, according to our experiments. Thus equation (27) reduces to:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{4} u_{v_i} \right) - u_{0,e_i}}^{\dot{U}} \right\|_{H_0^1(e_i)} \to \min \quad \forall i = 1,...,6 \tag{28}$$

where $\dot{U}$ vanishes on the element's vertices. After rewriting the norm:

$$\sqrt{\int_{e_i} \sum_{k=1}^{3} \nabla \cdot \left(\dot{U} - u_{0,e_i} \right)^2 dx} \to \min \quad \forall i = 1,...,6 \tag{29}$$

$$\int_{e_i} \sum_{k=1}^{3} \left(\frac{dU}{dx_k} - \frac{du_{o,e_i}}{dx_k} \right)^2 dx \to \min \quad \forall i = 1,...,6 \text{, and } u_{o,e_i} = a_{e_i} \varphi_i \tag{30}$$

we have

$$\int_{e_i} \sum_{k=1}^{3} \left(\frac{dU}{dx_k} \right)^2 dx - 2 \int_{e_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{du_{o,e_i}}{dx_k} dx + \int_{e_i} \sum_{k=1}^{3} \left(\frac{du_{o,e_i}}{dx_k} \right)^2 dx \to \min \quad \forall i = 1,...,6 \tag{31}$$

which leads to

$$\int_{e_i} \sum_{k=1}^{3} \left(\frac{du_{o,e_i}}{dx_k} \right)^2 dx - 2 \int_{e_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{du_{o,e_i}}{dx_k} dx \to \min \quad \forall i = 1,...,6 \tag{32}$$

since the other term is constant and can be omitted in minimization.

Let be $b(u, v)$ is a bilinear, symmetric form and l(u) is a linear form defined as:

$$b(u,v) = 2 \int_{e_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{du_{o,e_i}}{dx_k} dx \quad l(v) = \int_{e_i} \sum_{k=1}^{3} \left(\frac{du_{o,e_i}}{dx_k} \right)^2 dx \tag{33}$$

It is proven that minimizing $\frac{1}{2} b(u, v) - l(u)$ is reducible to solving $b(u, v) = l(u)$ for all test functions v. By applying this lemma we obtain

$$\int_{e_i} \sum_{k=1}^{3} \frac{du_{o,e_i}}{dx_k} \frac{dv}{dx_k} dx = 2 \int_{e_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{du_{o,e_i}}{dx_k} dx \to \min \quad \forall i = 1,...,6 \tag{34}$$

which leads to

$$a_{e_i} = \int_{e_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{d\varphi_{e_i}}{dx_k} dx \Big/ \int_{e_i} \sum_{k=1}^{3} \frac{d\varphi_{e_i}}{dx_k} \frac{d\varphi_{e_i}}{dx_k} dx \quad \forall i = 1,...,6 \tag{35}$$

The next step consists in an optimization on four faces of tetrahedral element:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{4} u_{v_j} - \sum_{j=1}^{6} u_{0,e_j} \right)}^{\ddot{U}} - u_{0,f_i} \right\|_{H_0^1(f_i)} \rightarrow \min \quad \forall i = 1,...,4 \tag{36}$$

where $\ddot{U}$ vanishes on vertices and edges. This leads to:

$$a_{f_i} = \int_{f_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{d\varphi_{f_i}}{dx_k} dx \Bigg/ \int_{f_i} \sum_{k=1}^{3} \frac{d\varphi_{f_i}}{dx_k} \frac{d\varphi_{f_i}}{dx_k} dx \quad \forall i = 1,...,4 \tag{37}$$

Finally, an analogical optimization in the interior of the finite element:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{4} u_{v_j} - \sum_{j=1}^{6} u_{0,e_j} - \sum_{j=1}^{4} u_{0,f_j} \right)}^{\ddot{U}} - u_{0,I} \right\|_{H_0^1(I)} \rightarrow \min \tag{38}$$

(where $\ddot{U}$ vanishes everywhere except from the interior) yields:

$$a_I = \int_{f_i} \sum_{k=1}^{3} \frac{dU}{dx_k} \frac{d\varphi_I}{dx_k} dx \Bigg/ \int_{f_i} \sum_{k=1}^{3} \frac{d\varphi_I}{dx_k} \frac{d\varphi_I}{dx_k} dx \tag{39}$$

It is worth noting that using this method the global matrix is not constructed at all. Thanks to the p=2 restriction, we have a single equation over each vertex, edge, face and interior. This algorithm requires a computational cost linear with respect to the mesh size, because it involves constant number of operations for each vertex, edge, face and interior and the number of respective nodes is proportional to n - the number of finite elements.

Exemplary results of this simulation are presented in works [D1, D2], see Figure 21.

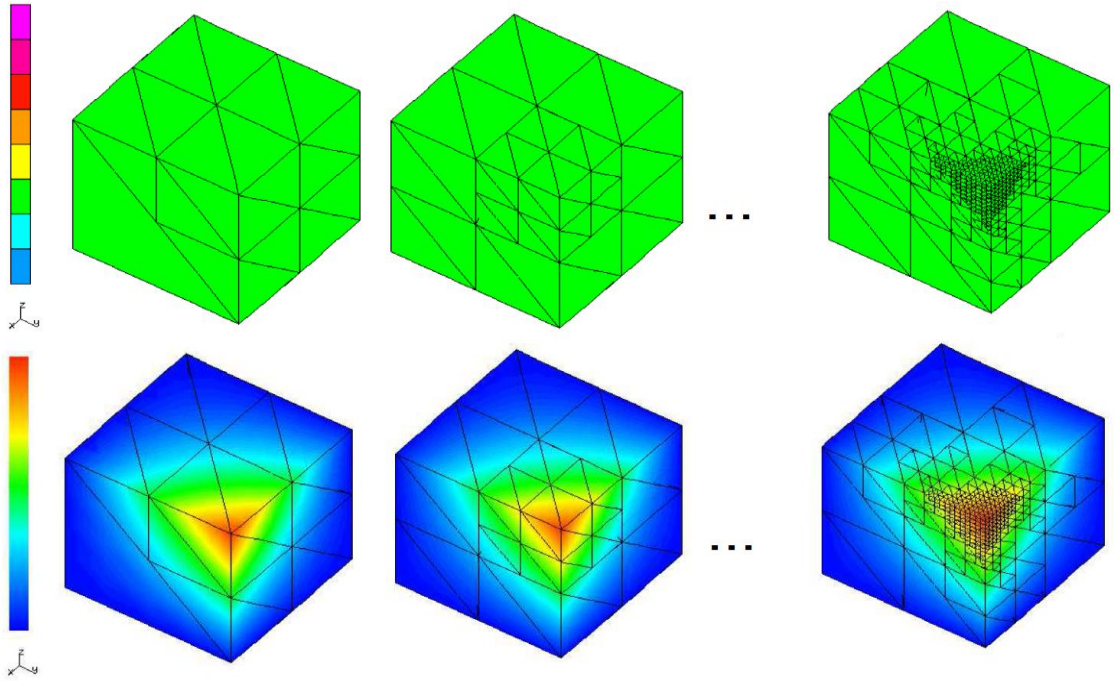

Figure 21 Computation sequence for projection pre-processor algorithm for the function with maximum at one corner of the cube

2.3 Elimination subtree re-use algorithm for multi-frontal direct solver

In papers [D6, D11] I propose an improvement to the multifrontal solver, which is the subject of current studies in the domain of direct solvers for modeling with the finite element method.

2.3.1 Classical algorithms of the multi-frontal solver

To fully understand the significance of the improvements for classic multifrontal solver algorithm, one may need the explanation of how it works. Multifrontal solvers were used in [C3, C4]. Their task is to solve linear equations system. The multifrontal solver algorithm execution is controlled by the so-called elimination tree, which can be constructed with the help of many available algorithms e.g. nested-dissections algorithm.

General algorithm for selected type of adaptive mesh algorithms can be presented similarly as in Algorithm 3 pseudo-code.

```
1 function multi-frontal solver (root node)
2   root problem  <- forward elimination (root node)
3   solution <- solve root problem
```

```
4    backward substitution (root node, solution)
5  end function
```

Algorithm 3 Classical algorithm of multi-frontal solver

Multifrontal solver algorithm starts with execution of a first elimination stage, it analyses an elimination tree from its leaves down to root.

This stage is concluded with the Algorithm 4 Classic algorithm of elimination stagepseudocode.

```
1  function forward elimination(node)
2    if node is a leaf then
3       generate local system assigned to node,
4       including boundary conditions
5    else
6       schur matrix1 <- multi-frontal (first son node)
7       schur matrix2 <- multi-frontal (second son node)
8       merge schur matrices into new system
9    end if
10   find fully assembled nodes
11   eliminate fully assembled nodes
12   return schur complement matrix
13 end function
```

Algorithm 4 Classic algorithm of elimination stage

In each node of the solver tree there is a 'subsystem' which represents the equation being solved on the part of domain (e.g. single finite element in the case elimination tree leaves). Subsystems can divide some variables, nevertheless, careful separation is recommended. Gauss elimination is carried out for each subsystem on these degrees of freedom, which are completely aggregated. The remaining part of the subsystem constitutes the so-called Schur complement matrix [C5]. Schur complement matrix of subsystems divide some variables, which are subsequently merged and partly eliminated, to a possible degree. The rest becomes another Schur complement matrix. Assembly and merger is continued for as long as there are variables left to eliminate.

In each node of the elimination tree, the part of local matrix which can be eliminated is identified (lines 10 and 11), and the remaining matrix referred to as Schur complementation (line 12) goes to next computation stage in the parent node. Then

the multi-frontal solver algorithm carries out the stage of backward substitution by analysis of the elimination tree from root to its leaves.

This stage is concluded with the Algorithm 5 pseudocode.

```
1  function backward substitution (node, solution)
2    execute backward substitution using solution from node local system
3    if node is a leaf then
4       restrict solution into node
5       execute backward substitution using solution
6    else
7       solution son1 <- restrict solution into first son node
8       backward substitution (first son node, solution son1)
9       solution son2 <- restrict solution into second son node
10      backward substitution (second son node, solution son2)
11   end if
12 end function
```

Algorithm 5 Classic algorithm of backward substitution

This stage is based on the reverse of elimination stage, solver analyses the tree from root to leaves. This stage has lower computational costs as we do not have to conduct partial elimination in elimination tree nodes, just merely a partial backward substitution.

The main aim of papers [D6, D11] is to generate and optimize the multifrontal solver algorithm for such trees.

2.4 Algorithm of elimination subtrees reuse algorithm for regular mesh

Work [D6] specifies modifications of multifrontal solver algorithm for regular meshes with constant values of material functions. For such meshes, operations carried out on various branches of multifrontal solver elimination tree are identical. Thus, it is possible to reuse the results from equations on similar subtrees. To eliminate asymmetry resulting from the position of a given element one should delay introduction of boundary conditions on the boundaries of the domain to the root of a tree.

Modified algorithm is presented with Algorithm 6.

```
1  function multi-frontal solver (root node)
2    root problem <- forward elimination (root node)
3    enforce boundary conditions at root problem
4    solution <- solve root problem
5    backward substitution (root node, solution)
6  end function
```

```
1  function forward elimination (node)
2    if node is a leaf then
3       generate local system assigned to node, incl. boundary conditions
4       mark boundary nodes as not fully assembled
5    else
6       schur matrix1 <- multi-frontal (first son node)
7       schur matrix2 <- schur matrix1
8       merge schur matrices into new system
9    end if
10   find fully assembled nodes
11   eliminate fully the boundary nodes
12   eliminate fully assembled nodes
13   return schur complement matrix
14 end function
```

```
1  function backward substitution (node, solution)
2    execute backward substitution using solution
3    from first node on this level
4    if node is a leaf then
5       restrict solution into node
6    else
7       solution <- son1
8       restrict solution into first son node
9       backward substitution (first son node, solution son1)
10      solution <- son2
11      restrict solution into second son node
12      backward substitution (second son node, solution son2)
13   end if
14 end function
```

Algorithm 6 Algorithm of elimination tree subtrees reuse

The algorithm is illustrated in Figure 22.

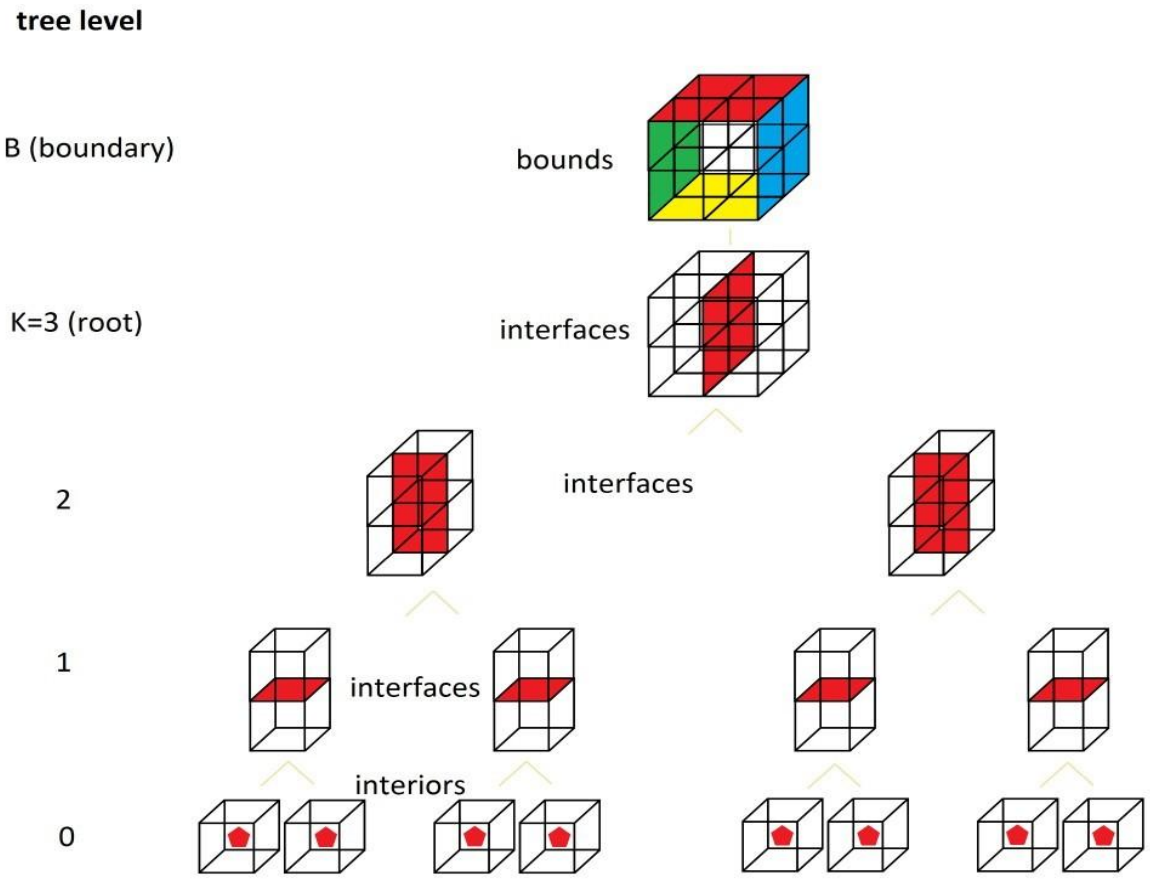

Figure 22 Execution of the solver over the regular cubic mesh results in identical sub-branches of the elimination tree

2.5 Properties and applications of elimination subtree re-use algorithm

In this chapter we estimate the benefit of computing a single path in the above tree instead of the full tree for a regular 3D mesh.

For a given solver level $0 \leq k \leq K$, the partial Gaussian elimination complexity depends on two coefficients:

- M_k - the total number of degrees of freedom on the level k,
- E_k - the number of degrees of freedom that are eliminated on the level k

where $k = K$ signifies the root level of the tree (and consequently the height of the whole tree) and $k = 0$ stands for lowest bottom nodes. Note that the Schur complement on the level k is an $(M_k - E_k) \times (M_k - E_k)$ submatrix. For such a matrix, Gaussian elimination and substitution require exactly:

$$T_G(M_k, E_k) = \sum_{n=1}^{E_k}(M_k - n - 1)(M_k - n) + 2 = \frac{1}{3}\left(3{M_k}^2 E_k - 3{E_k}^2 M_k + {E_k}^3 + 6M_k - E_k\right) \quad (40)$$

operations. The required memory, in turn is proportional to:

$$S(M_k, E_k) = {M_k}^2 + {E_k}^2 + M_k + E_k \quad (41)$$

The number of degrees of freedom per node is given by:

$$D_i = (p-1)^3 \quad (42)$$

$$D_f = (p-1)^2 \quad (43)$$

$$D_e = (p-1)^1 \quad (44)$$

$$D_v = (p-1)^0 = 1 \quad (45)$$

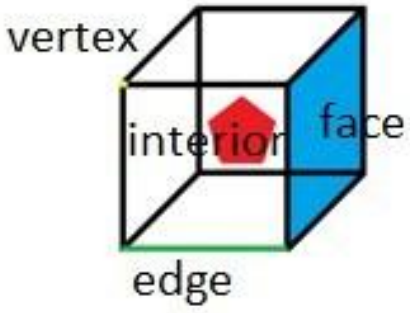

Figure 23 Naming convention for an element

where i, f, e, v mean respectively interiors, faces, edges and vertices of a finite element, as illustrated on Figure 23. In the above formulas p stands for the polynomial order of finite element shape functions used for the Galerkin discretization base.

Thus, M_k can be expressed as

$$M_k = \sum_{x\in\{i,f,e,v\}} M_k^x \, D_x \quad (46)$$

and E_k can be written as

$$E_k = \sum_{x\in\{i,f,e,v\}} E_k^x \, D_x \quad (47)$$

To compute the solver complexity we need to find a formula for M_k^x and E_k^x for $x \in \{i, f, e, v\}$ and $k \in 0..K$. For $x = i$, this is trivial, since all interiors are eliminated at level 0. Hence, $M_0^i = 1$, $E_0^i = 1$, $M_k^i = 0 \forall_{k>0}$ and $E_k^i = 0 \forall_{k>0}$.

For other nodes the reasoning is a bit more complex. Let us introduce a helper variable R_k^x, defined as the number of the nodes of type x, on the interface of level k(they are repeated on both submeshes under the merger, hence "R") . Note that $R_k^x \geq E_k^x$ as it is always a subset of the interface (repeated) nodes that is eliminated. Also, The following formula is true for all $x \in \{f, e, v\}$ and $0 < k \leq K$:

$$M_k^x = 2(M_{k-1}^x - E_{k-1}^x) - R_k^x \tag{48}$$

k	M_k^f	E_k^f	R_k^f
0	6	0	0
1	11	1	1
2	18	2	2
3	28	4	4
4	44	4	4
5	72	8	8
6	112	16	16

Table 4 Coefficients over the face

2.5.1 FACES

The first several values of these sequences are enumerated in Table 4. It is easy to observe that $M_0^f = 6$, $R_0^f = E_0^f = 0$, $R_1^f = E_1^f = 1$ (required for the recursive formula) and

$$R_k^f = \begin{cases} R_{k-1}^f & k \,\%\, 3 = 1 \\ 2R_{k-1}^f & k \,\%\, 3 = 2 \\ 2R_{k-1}^f & k \,\%\, 3 = 0 \end{cases} \tag{49}$$

$$E_k^f = \begin{cases} E_{k-1}^f & k \,\%\, 3 = 1 \\ 2E_{k-1}^f & k \,\%\, 3 = 2 \\ 2E_{k-1}^f & k \,\%\, 3 = 0 \end{cases} \tag{50}$$

which is, in a non-recursive form:

$$E_k^f = R_k^f = 2^{2[(k-1)/3]+(k-1)\%3} \tag{51}$$

where / is the integer division (rounding down) and % is the modulo division operator.

2.5.2 EDGES

The first few values for the coefficients corresponding to edges are presented in Table 5.

Here, $M_0^e = 12$, $R_0^e = 0$, $E_0^e = 0$, $R_1^e = 4$, $E_1^e = 0$ and:

$$R_k^e = \begin{cases} R_{k-1}^e & k\,\%\,3 = 1 \\ 2R_{k-1}^e - 2^{k/3} & k\,\%\,3 = 2 \\ 2R_{k-1}^e - 2^{k/3} & k\,\%\,3 = 0 \end{cases} \tag{52}$$

$$E_k^e = \begin{cases} E_{k-1}^e & k\,\%\,3 = 1 \\ 2E_{k-1}^e + 2^{k/3} & k\,\%\,3 = 2 \\ 2E_{k-1}^e + 2^{k/3} & k\,\%\,3 = 0 \end{cases} \tag{53}$$

k	M_k^e	E_k^e	R_k^e
0	12	0	0
1	20	0	4
2	33	1	7
3	52	4	12
4	84	4	12
5	138	10	22
6	216	24	40

Table 5 Coefficients over edges

2.5.3 VERTICES

The same coefficients for vertices are outlined in Table 6.

k	M_k^v	E_k^v	R_k^v
0	8	0	0
1	12	0	4
2	18	0	6
3	27	1	9
4	43	1	9
5	69	3	15
6	107	9	25
7	171	9	25
8	279	21	45

Table 6 Coefficients over vertices

Here, $M_0^v = 8$, $E_0^v = 0$, $R_0^v = 0$, $E_0^v = 0$, $R_1^v = 4$ and

$$R_k^v = \begin{cases} R_{k-1}^v & k \,\%\, 3 = 1 \\ 2R_{k-1}^v - 2^{k/3} - 1 & k \,\%\, 3 = 2 \\ 2R_{k-1}^v - 2^{k/3} - 1 & k \,\%\, 3 = 0 \end{cases} \tag{54}$$

$$E_k^v = \begin{cases} E_{k-1}^v & k \,\%\, 3 = 1 \\ 2E_{k-1}^v + 2^{k/3} - 1 & k \,\%\, 3 = 2 \\ 2E_{k-1}^v + 2^{k/3} - 1 & k \,\%\, 3 = 0 \end{cases} \tag{55}$$

which simplifies to the following non-recursive formulas:

$$R_k^v = \left(2^{k/3} + 1\right)\left(2^{(k+1)/3} + 1\right) \tag{56}$$

$$E_k^v = \left(2^{k/3} - 1\right)\left(2^{(k+1)/3} - 1\right) \tag{57}$$

2.5.4 BENEFITS FROM THE OPTIMIZATION

The absolute benefit from applying the subtree reutilization optimization to the polymer strain modeling during the SFIL process can be computed by comparing the cost (T) / memory consumption (S) of the usual, full tree computation:

$$T = \sum_{k=0}^{K} 2^{K-k} T_G(M_k, E_k) + T_G(M_B, E_B) \tag{58}$$

$$S = \sum_{k=0}^{K} 2^{K-k} S(M_k, E_k) + S(M_B, E_B) \tag{59}$$

to the optimized computing cost (T') and space requirements (S')

$$T' = \sum_{k=0}^{K} T_G(M_k, E_k) + T_G(M_B, E_B) \tag{60}$$

$$S' = \sum_{k=0}^{K} S(M_k, E_k) + S(M_B, E_B) \tag{61}$$

In both cases, $T_G(M_B, E_B)$ and $S(M_B, E_B)$ represent additional time and space spent on processing boundary conditions. The expression $T_G(M_k, E_k)$ can be computed for any k by supplying M_k and E_k (computed using formulas (46) and (47), respectively) to equation (40).

Conversely, we get relative savings by referencing the absolute savings to the cost of the non-optimized version:

$$\Delta T = \frac{T - T'}{T} \tag{62}$$

$$\Delta S = \frac{S - S'}{S} \tag{63}$$

Since $T_G(M_k, E_k)$ and $S(M_k, E_k)$ are functions of k and p and consequently T, T', S, S' are functions of K and p, so are ΔT and ΔS. Theoretical benefit (time-wise and memory-wise) from using the described optimization is decapitated in Figure 34 and Figure 35, respectively. The values of ΔT and ΔS have been computed by a Python application which basically expands all the recursive formulas introduced in this paper.

As one can see time and memory savings increase with the size of the problem (n). For small problems both time and memory savings raise, whereas for larger problems both measures decrease. They also converge asymptotically to, respectively, 31% and 66%.

Overall, the optimization seems to have moderate effect on the computation time, but memory savings of order of 70% should translate to tangible reductions in hardware expenditures.

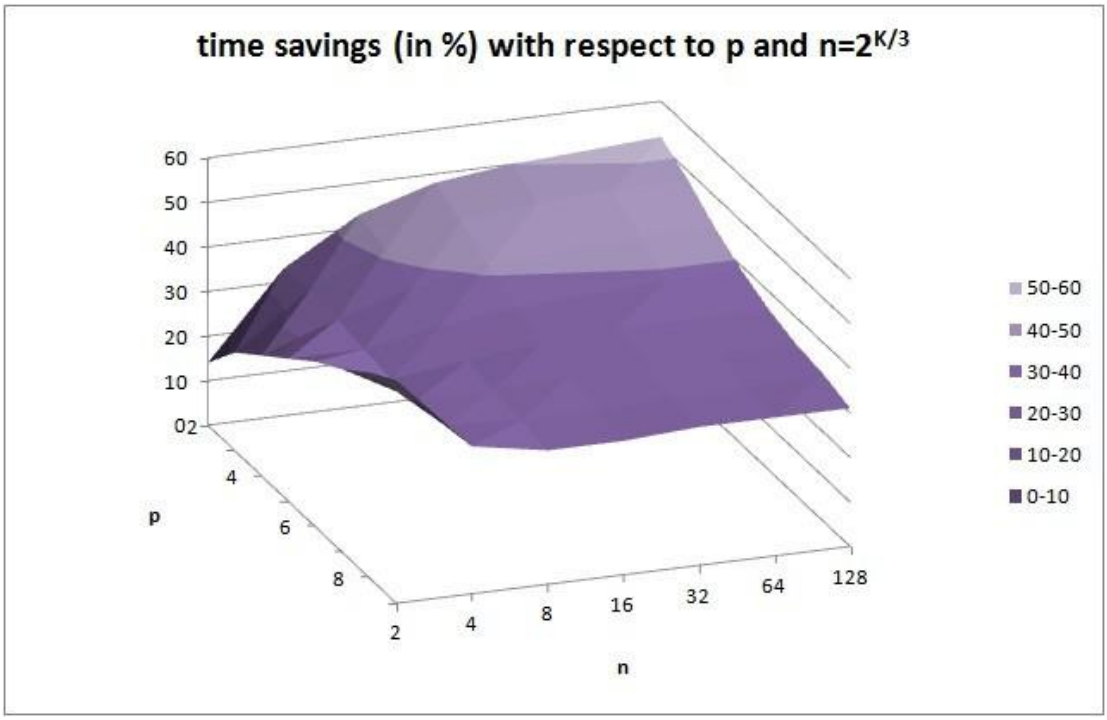

Figure 24 Theoretical estimations of a proportion of computational time saved thanks to the subtree reuse optimization on the exemplary problem

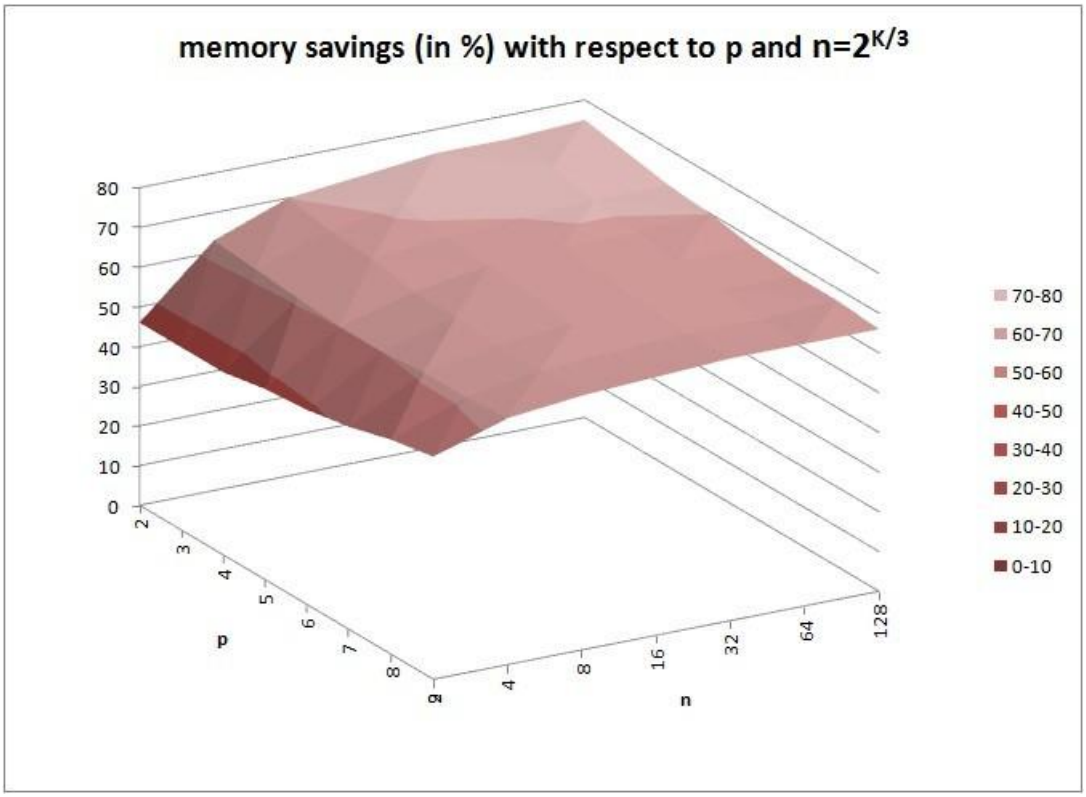

Figure 25 Theoretical estimations of a proportion of memory usage saved thanks to the subtree reuse optimization on the exemplary problem

2.6 Algorithm of elimination subtrees reuse for irregular meshes

I have also proposed an algorithm for re-use of the elimination subtrees for irregular meshes, on which we apply both macro- and nano-scale approach [D13]. The algorithm is based on grammar graphs which attribute subtrees of an elimination tree and check the possibility to reuse these subtrees during change of the elimination tree generated for a given coarse mesh, on which macro-scale model was used, whereas some parts of coarse mesh are based on use of nano-scale model. Reuse is only possible for these subtrees which represent parts of a mesh on which macro-scale model was applied. The algorithm was tested on multi-physics Step-and-Flash Imprint Lithography problem described in Appendix B.

2.6.1 GRAPH GRAMMAR BASED FORMULATION OF THE MULTI-FRONTAL SOLVER ALGORITHM WITH REUSE TECHNIQUE

In this section a multi-frontal solver is described. In order to present the algorithm performed by the solver, graph transformation systems are used. Such a formalism allows for describing steps of an algorithm as a set of transformations that alter a graph representing a computational mesh and attribute it with computational data. As a result it's easier to determine the algorithm's properties, time complexity and determine whether a given algorithm is legal.

The first step of the solver algorithm is to generate the computational mesh. It is done by executing a sequence of graph transformations that generate a graph structure representing computational mesh. The first graph transformation is presented on left panel in Figure 26. The productions replace the starting graph containing only a single vertex **S** with a graph representing a single hexahedral element with eight nodes. The following graph transformations replace some nodes by sub-graphs that represent smaller elements. Graph nodes as well as graph transformations are attributed by the location over a rectangular domain. The transformation $\mathbf{(P)^{TNW}}$ from right panel in Figure 37 is replicated for different locations, for **{TNW,TNE,TSW,TSE,BNW,BNE,BSW,BSE}** where **T** and **B** stands for top and bottom, and **N**, **S**, **W**, **E** stand for north, south, west, east.

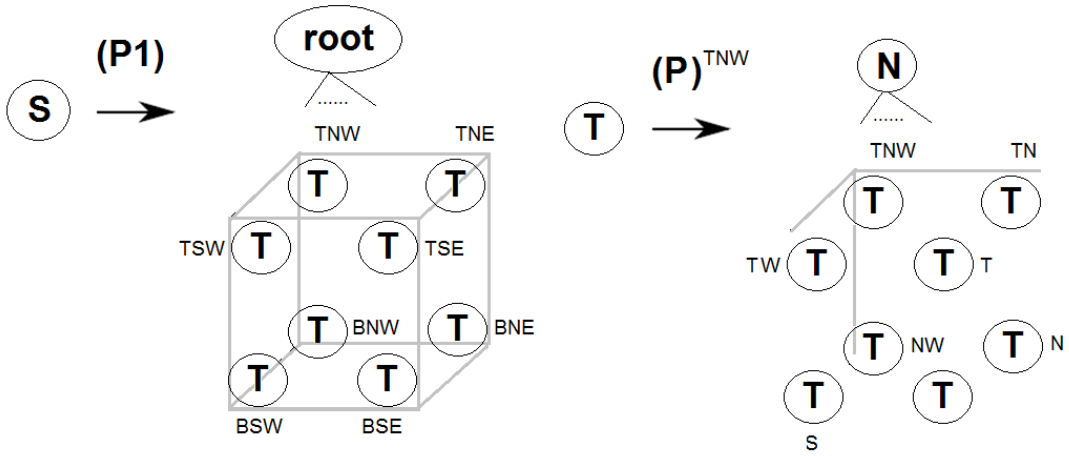

Figure 26 Exemplary graph transformations for generating of the structure of the mesh

The exemplary derivation of an eight-element mesh is presented in Figure 27. In the first step, production **(P1)** is executed, in the second step, productions **(P)^{TNW} - (P)^{TNE} - (P)^{TSW} - (P)^{TSE} - (P)^{BNW} - (P)^{BNE} - (P)^{BSW} - (P)^{BSE}** are executed to obtain the eight finite element mesh. The graph representing the mesh has hierarchical tree-like structure storing the history of graph transformations derivation. To obtain larger meshes, it is necessary to add graph transformations for locations like **{T,B,N,S,W,E,TN,TS,TW,TE,BN,BS,BW,BE,NE,NW,SE,SW}**, compare labels of the left bottom sub-graph at Figure 39.

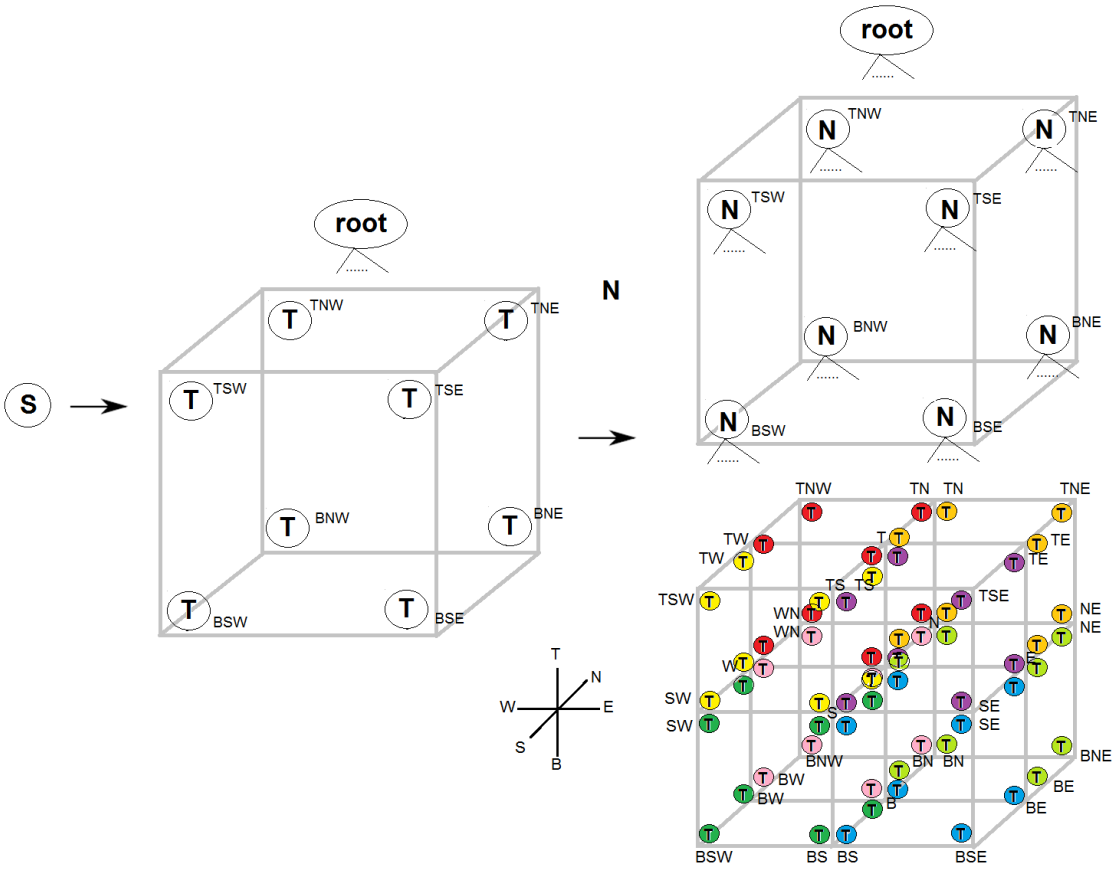

Figure 27 Derivation of eight finite element mesh

The next step of the solver algorithm consists in identification of macro-scale and nano-scale elements. Notice that graph nodes labeled with ***N*** actually represent particles (over nano-scale elements) or finite element method nodes (over macro-scale elements). Thus, the elements are represented by patches of eight nodes.

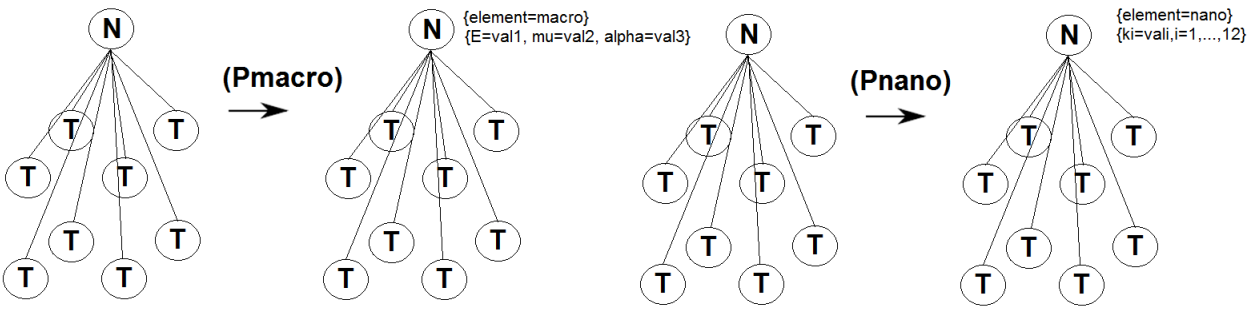

Figure 28 Graph transformations for identification of macro- and nano-scale elements

This identification is performed by graph transformation presented in Figure 28. The macro-scale elements are attributed by Young modulus and Poisson ratio values. The nano-scale elements are attributed by parameters of the spring force parameters $k_{\alpha\beta}$.

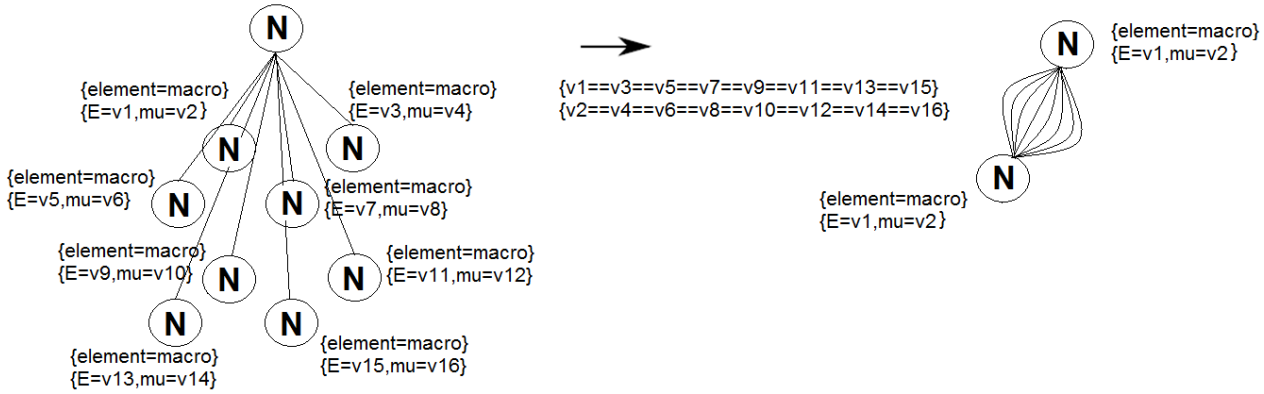

Figure 29 Exemplary graph transformation for identification of macro-scale elements with identical material data

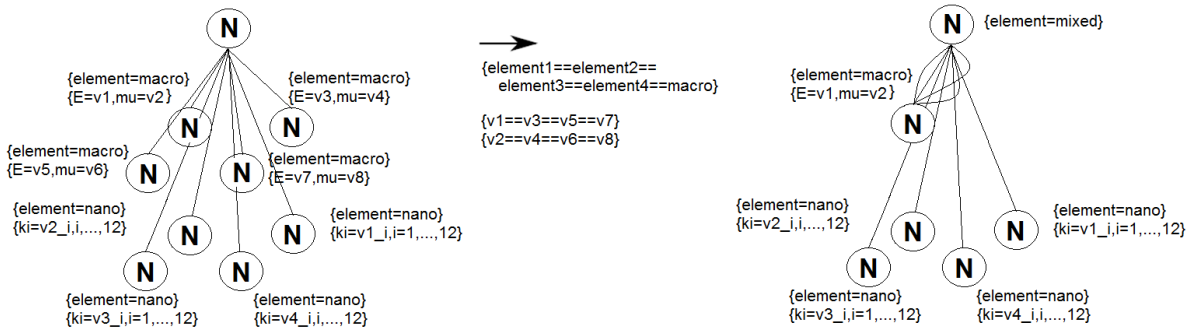

Figure 30 Exemplary graph transformation for partial identification of macro-scale with identical material data

The resulting tree structure can be directly utilized by the multi-frontal solver algorithm. However, in this work a more sophisticated approach, featuring a re-use optimization technique, is proposed. It is based on an observation, that if we

postpone the resolution of the domain boundary to the top of the elimination tree, given a regular mesh with equal coefficients, the LU factorized local matrices for different finite elements are the same and hence can be reused.

Thus, the third step of the solver algorithm consists in an identification of identical sub-branches of the elimination tree, for the reuse of partially LU factorized matrices. An exemplary graph transformation for such an identification is presented in Figure 40. Such a graph transformation checks if all eight son elements are macro-scale elements and whether the corresponding Young modulus and Poisson ratios are identical. If this is the case, the eight son element nodes are reduced to one representative node, so the LU factorization can be performed only once and father node can merge eight identical matrices from the same representative son node.

Another, more complicated case for the identification is presented in Figure 30. In this example only four son elements are macro-scale with identical Young modulus and Poisson ratio values. The four identical macro-scale elements are reduced to one representative element, however the nano-scale elements are stochastic in their nature and cannot be reduced.

Finally, on a modified elimination tree, the multi-frontal solver algorithm can be executed:

```
1  function frontal_elimination(node)
2     if new_schur_matrix already computed for the node then
3        return schur_matrix
4     if node is a leaf then
5        generate local system assigned to node
6        excluding boundary conditions
7     else
8        loop through son_nodes
9           schur_matrix = frontal_elimination(son_node)
10          merge schur_matrix into new_system
11       end loop
12    end if
13    find fully assembled nodes and eliminate them
14    return new_schur_matrix
15 end function
```

Algorithm 7 Frontal elimination algorithm

Notice that in case of representative nodes in line 9, the same node of the elimination tree is actually called many times and line 2 prevents from re-computing the identical Schur complement matrices many times. The forward elimination algorithm is followed by analogous backward substitution.

2.6.2 NUMERICAL EXPERIMENTS

In this section numerical results presenting the shrinkage of the feature after removal of the template. It is assumed that the polymer network has been damaged during the removal of the template, and thus the inter-particle forces are weaker in one part of the mesh.

The problem has been solved first by using pure nano-scale approach, with non-linear model allowing for large deformations, with quadratic potentials, as defined in Appendix B). The resulting equilibrium configurations of polymer network particles are presented in Figure 31 and Figure 32. The damage has been modeled here by assuming smaller values of the spring stiffness coefficients $k_{\alpha\beta}$.

Then, the problem has been solved again by using the multi-scale approach. The part of the mesh with undamaged polymer has been modeled by the macro-scale approach, with Finite Element Method. The part of the mesh with the damaged polymer, denoted in Figure 33 by red color, has been modeled by the nano-scale approach with linear model assuming small deformations and quadratic potentials. The damage of the polymer, modeled by weakening the inter-particle forces results in slight lean of the feature, illustrated in Figure 32 for the nano-scale model, and in Figure 31, for the macro-scale model. The displacement fields are similar in both nano-scale and macro-scale simulations.

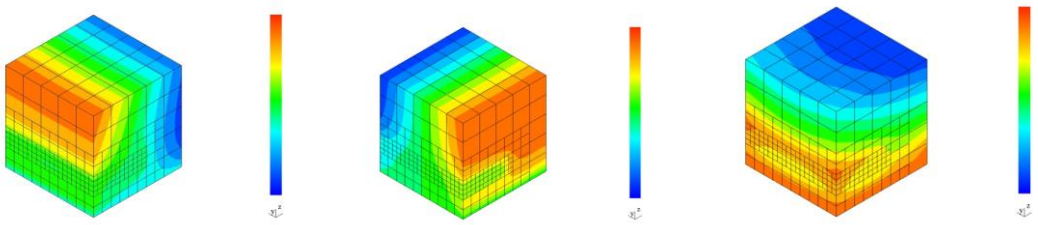

Figure 31 X, Y and Z components of the displacement vector field for the interior modeled by linear elasticity with thermal expansion coefficient

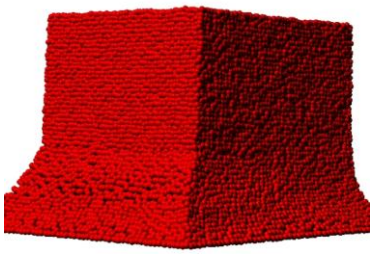

Figure 32 Results of the non-linear model allowing for large deformations, with quadratic potentials

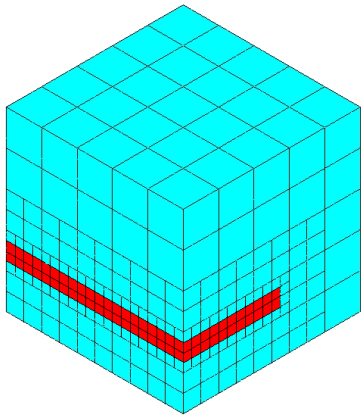

Figure 33 3D mesh for multi-scale simulations. The blue color denotes the macro-scale domain with FEM model, the red color denotes the nano-scale domain with MS model

To conclude this section, let us look at the comparison of the execution times of the graph transformation based multi-frontal solver executed with and without the reuse technique. The computations have been performed sequentially on a cluster node with Dual-Core AMD Opteron processor clocked at 2.6 GHz with 32 GB using a Fortran 90 implementation.

The results are presented in Figure 34-Figure 35. The horizontal axis denotes different polynomial orders of approximations utilized over the macro-scale domain (p parameter). Different lines correspond to different number of elements in each direction (n parameter). The resulting speedup of the reuse solver algorithm is presented in Figure 35.

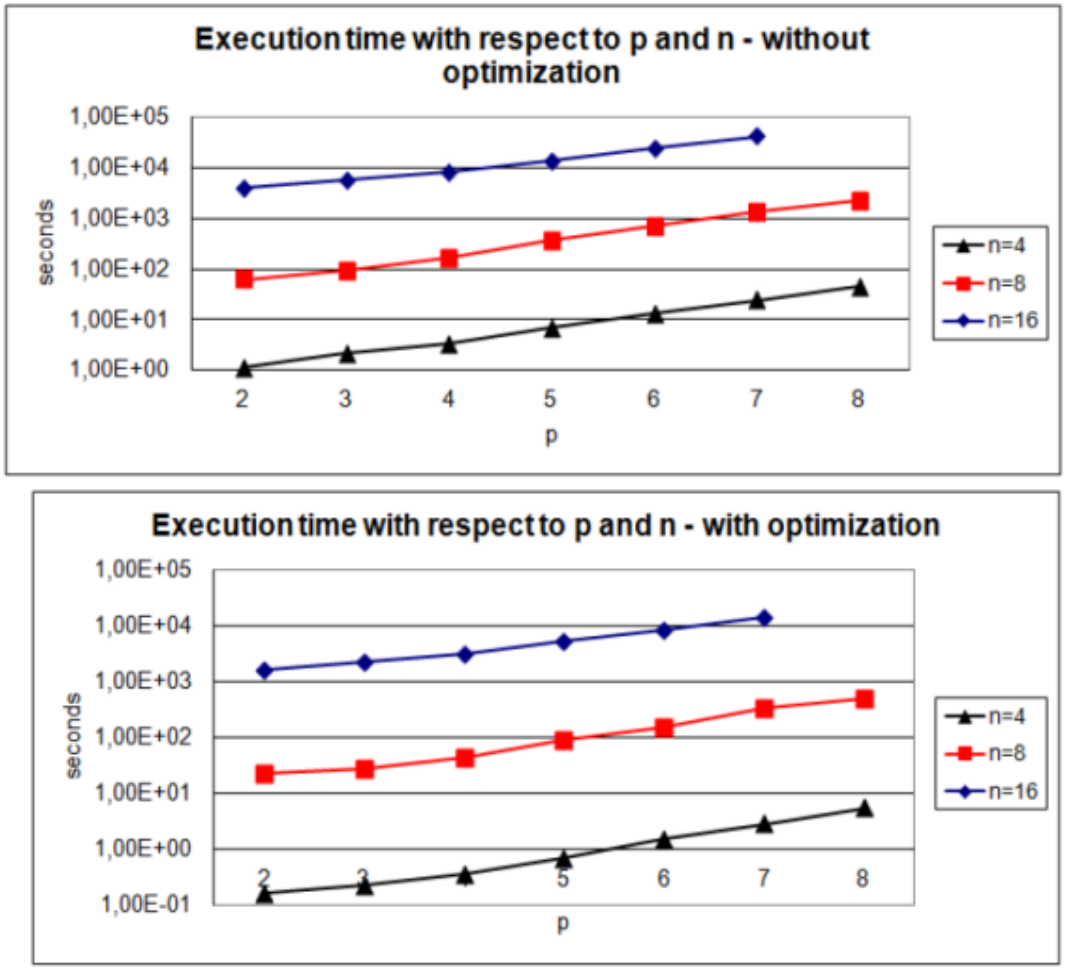

Figure 34 Top panel: Execution time of the solver without reuse Bottom panel: Execution time of the solver with reuse

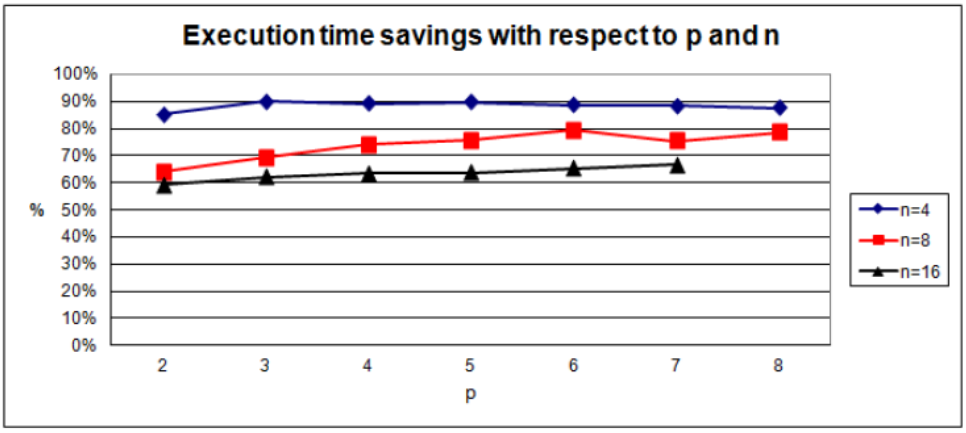

Figure 35 Speedup of the reuse solver

2.7 Adaptive algorithm of automatic scale change for multi-scale methods

In papers [D7, D12] (and also [D20, D28, B3, D25, D26]) I have developed an algorithm of multiscale modelling with the use of dynamic change of model scales. The algorithm proposed in [C6] was implemented and tested for one- and three-dimensional problems. In particular, work [D7] illustrates, on a simple example, the proof of concept, whereas [D12] tackles the same problem in 3D.

The idea of multiscale modeling is presented below on an example of elastic deformation of a substrate in the process of lithography by the method of Step-and-Flash Imprint Lithography specified in [C9]. In this problem, substrate deforms as a result of irradiation with UV rays and takes shape of the template applied thereto.

2.7.1 One dimensional multi-scale model of Step-and-Flash Imprint Lithography

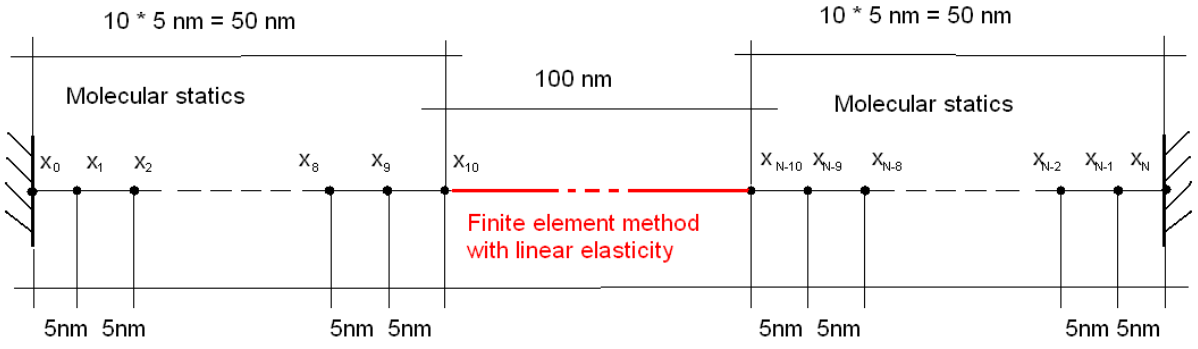

Figure 36 The scheme of problem domain division between applicable models

In [D7] we used spot interface, domain division between models is presented in Figure 36. It is worth mentioning, that it is not the only method of joining models of different scales. In particular, better results, regarding the problem of linear elasticity, have been achieved with the use of Arlequin Interface proposed in [C13].

2.7.2 Defining discreet problem

Discreet model, which shows interaction between polymer and template, constitutes one of the foundations applied to multiscale model. The model assumes square potentials shown by formula:

$$V(r) = k\left(r - r_{eq}\right)^2 \rightarrow F(r) = 2k(r - r_{eq}) \tag{28}$$

between each pair of interacting elements, which results in linear forces. Model parameters (the distance between elements $r_{eq} = 5.34$ and elastic modulus $k = 0.1$) were extracted from [C7]. The interaction force between ith and i+1th equals:

$$F_{i,i+1} = 2k_{i,i+1}\left(x_{i+1} - x_i - r_{i,i+1}^{eq}\right) \tag{29}$$

where $k_{i,i+1}$ is an elastic modulus for the model of 'spring' between the ith and i+1th element, whereas $r_{i,i+1}^{eq}$ is a equilibrium length for the 'spring'.

In one-dimension, movement between adjacent layers of molecules triggers an additional force, which is described in formula:

$$f(x) = cx(x - 100)(x - 200) \tag{30}$$

where constant c stands for maximum horizontal deviation.

Second Newton's law implies that for each cell the following balance equations can be formulated:

$$F_{i,i+1} - F_{i-1,i} + f(x_i) = 0 \tag{31}$$

which, after substitution of the definition of intermolecular forces (3), leads to the following equation:

$$x_{i-1}\left(k_{i-1,i}\right) + x_i\left(-k_{i-1,i} - k_{i,i+1}\right) + x_{i+1}\left(k_{i,i+1}\right) = f(x_i) + r_{i,i+1}^{eq}k_{i,i+1} - r_{i-1,i}^{eq}k_{i-1,i} \tag{32}$$

x_i are unknown equlibrium positions of individual elements. Of course, the locations of boundary elements are given:

$$x_0 = 0\ x_N = 200 \tag{33}$$

2.7.3 Definition of continuous problem

Alternatively, polymer deformation in a template can be explained with the following continuous problem. We start with standard linear elasticity problem (C12) with external force f

$$-\frac{d}{dx}\left(EA\frac{du}{dx}\right) = f \tag{34}$$

where E represents Young modulus, whereas A stands for cross section area. Therefore, modulus boundary conditions on the interface equals:

$$EA\frac{du(50)}{dx} = -F_{9,10} \quad EA\frac{du(150)}{dx} = -F_{N-10,N-9} \tag{35}$$

where $F_{9,10}$ is the effective intermolecular force between molecules 9 and 10, whereas $F_{N-10,N-9}$ is force between molecules $N-10$ and $N-9$. When we substitute discreet problem, we get:

$$EA\frac{du(50)}{dx} = -2k_{9,10}\left(x_{10} - x_9 - r_{9,10}^{eq}\right)$$

$$EA\frac{du(150)}{dx} = -2k_{N-10,N-9}\left(x_{N-9} - x_{N-10} - r_{N-10,N-9}^{eq}\right) \tag{36}$$

Moreover, in this model the molecules are identified in discreet displacements field, with values u_l, u_r equivalent to coefficients in the first row shape functions on the domain's boundary, where continuous problem is modeled with finite elements method.

$$u_l = x_{10} - 50 \;\; u_r = x_{N-10} - 150 \tag{37}$$

The strong formula presented above leads to the following weak (variational) formula:

$$\int_{50}^{150} EAu'v'dx - EAu'(150)v(150) + EAu'(50)v(50) = \alpha[v(150) - v(50)] + \int_{50}^{150} fvdx \tag{38}$$

which is correct for each test function $\forall_{v\in V\subset H^1(50,150)}$, while α stands for a coefficient of thermal expansion that satisfies $\int_{50}^{150} \alpha v'dx = \alpha[v(150) - v(50)]$.

When we substitute interface conditions, we get:

$$\int_{50}^{150} EAu'v'dx - 2k_{N-10,N-9}\left(x_{N-9} - x_{N-10} - r_{N-10,N-9}^{eq}\right)v(150) - 2k_{9,10}\left(x_{10} - x_9 - r_{9,10}^{eq}\right)v(50) = \int_{50}^{150} fvdx + \alpha[v(150) - v(50)] \tag{39}$$

2.8 Adaptive algorithm of automatic scale change for multiscale models

Adaptive algorithm of automatic scale change for multiscale models can be summarized in the form of pseudo-code presented in Algorithm 8.

```
1 function adaptive_multiscale_fem(initial_mesh, desired_err, coef)
2 coarse_mesh = initial_mesh with all macro-scale elements
3 repeat
4    coarse_u = solve the problem on coarse_mesh  using the solver algorithm
with re-use of identical branches
5    fine_mesh = copy coarse_mesh
6    divide each element K of fine mesh into 8 new elements (K1 .. K8)
7    increase polynomial order of shape functions on each element of fine
mesh by 1
8    fine_u = solve the problem on fine_mesh using the solver algorithm with
re-use of identical branches
9    max_err = 0
10   for each element K of fine mesh do
11      K_err = compute relative decrease error rate on K
12      if K_err > max_err then
13         max_err = K_err
14      end if
15   end do
16   adapted_mesh = new empty_mesh
17    for each element K of coarse_mesh do
18      if K_err > coef * max_err then
19         execute h refinement of element K
20         if size of new element < scale threshold then
21            change the element scale from macro-scale to nano-scale
22         end if
23      else
24         add K from coarse_mesh to adapted_mesh
25      end if
26   end do
27   coarse_mesh = adapted_mesh
28   output fine_u
29 until max_err < desired_err
30 return (fine_u, fine_mesh)
```

Algorithm 8 Adaptive algorithm for automatic scale change

Proposed algorithm of automatic scale change prevents application of macro-scale models in circumstances, where size of a local element requires nano-scale model.

The object-oriented implementation of the algorithm is discussed in Appendix C. We have generated the Java code from the discussed UML diagrams, and implemented the bodies of class methods manually. We conclude the presentation with the

numerical experiments presenting the solution of the multi-scale problem defined in the "Problem formulation" section. First, we have solved the problem in the meso-scale, to provide the "exact" solution presented on the first panel in Figure 47. Then, the internal part of the domain has been modeled by the macro-scale linear elasticity formulation (13), while the external parts of the domain keep the meso-scale model, to express the interactions of the polymer with the template. This fully multi-scale problem has been solved on the manually designed non-uniform hp mesh, with the order of approximation varying from p=1 to p=5, as it is presented on second panel in Figure 47.

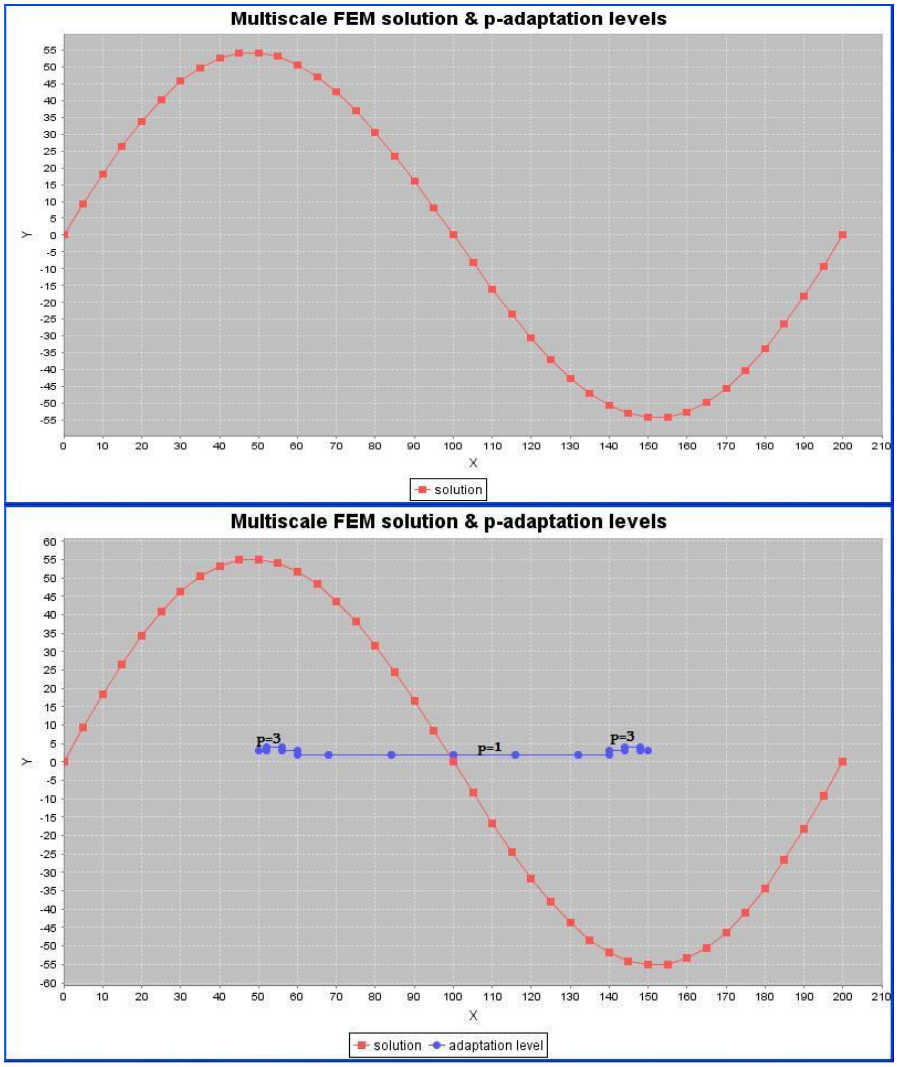

Figure 37 The solutions of the multi-scale problem. The red lines denote the solution, the blue lines denote the order of approximation on the FEM domain. First panel: The solution obtained with the meso-scale model defined over the entire domain. Second panel: The solution obtained over the manually hp refined mesh.

3 CONCLUSIONS

In this dissertation I have proposed a family of algorithms that - in my opinion - taken together contribute considerably to the field of numerical computer science, allowing for effective processing of boundary problems for which material data are given in a non-continuous way (e.g. with an MRI scan bitmap). The algorithms considered included:

- an application of three dimensional fully automatic adaptive algorithms for generation of continuous representation of material data based on three dimensional MRI scans,
- a linear computational cost adaptive algorithms for generation of continuous representation of material data based on three dimensional MRI scans, using projection based interpolation algorithm,
- a projection based interpolation algorithm for the case of three dimensional hexahedral and tetrahedral grids,
- an algorithm that re-uses identical parts of the mesh, with identical material data

I have expressed also expressed the multi-frontal solver algorithm allowing for reuse techniques with graph grammar productions which allow for dynamic construction of elimination trees for multi-scale problems.

For the proposed algorithms I have evaluated their performance and relative trade-offs:

- I have performed a number of numerical experiments showing how generation of continuous representation of material data improves the convergence of the finite element method.
- I have also proposed a framework for automatic switching of the scales from macro-scale to nano-scale during multi-scale simulations.

In my opinion these results are significant enough to prove the initial thesis behind this work.

4 ACKNOWLEDGEMENTS

My research was funded by:

- Preludium grant from National Science Centre Prelude, awarded with the decision DEC-2011/03/N/ST6/01397, of which I was the Director
- Harmony grant from National Science Centre, awarded with the decision DEC-2011/03/N/ST6/01397, of which I was an Executor
- 2 Dean's grants from Faculty of Computer Science, Electronics and Telecommunications

During my studies I has also been a beneficiary of Małopolski Fundusz Stypendialny dla Doktorantów 'Doctus' (Lesser Poland Scholarship Fund for PhD candidates 'Doctus'), co-financed by European Union from the European Social Fund.

I am deeply grateful to these institutions as well as to the other organizations that supported me with grants & scholarships throughout my tertiary education:

- Ministry of Science and Higher Education
- The city of Krakow
- Małopolska Agencja Stypendialna "Sapere Auso" (Lesser Poland Scholarship Foundation 'Sapere Auso')
- AGH University of Science and Technology
- Cracow University of Economics

Appendix A. DERIVATION OF LINEAR ELASTICITY MODEL WITH THERMAL EXPANSION COEFFICIENT

A.1 STRONG FORM

Let:

- u_i be the displacement vector
- $u_{i,j}$ - displacement gradients
- $u_{(i,j)}$ - a symmetric part of the displacement gradients $u_{(i,j)} = \frac{u_{i,j} + u_{j,i}}{2}$
- $u_{[i,j]}$ - a skew-symmetric part of the displacement gradients $u_{[i,j]} = \frac{u_{i,j} - u_{j,i}}{2}$
- $u_{i,j} = u_{(i,j)} + u_{[i,j]}$
- ε_{ij} be a strain tensor defined as the symmetric part of the displacement gradient

  $\varepsilon_{ij} = u_{(i,j)} = \frac{u_{i,j} + u_{j,i}}{2}$
- σ_{ij} stress tensor, defined in terms of the generalized Hooke's law

  $\sigma_{ij} = c_{ijkl}\left(\varepsilon_{kl} + \theta\,\alpha_{kl}\right) + \sigma_{ij}^0$

where

- c_{ijkl} elastic coefficients (known for a given material)
- σ_{ij}^0 initial stress
- ε_{ij}^0 initial strain
- $\varepsilon_{kl}^0 = -\theta\,\alpha_{kl}$
- θ temperature
- α_{kl} thermal expansion coefficients

Strong form of the boundary-value problem

Given $g_i : \Gamma_{D_i} \ni x \to g_i(x) = 0 \in R$, θ, α_{kl} *and* σ_{ij}^0, *find* $u_i : \overline{\Omega} \to R$ *such that*

$\sigma_{ij,j} = 0$ *in* Ω

$u_i = g_i$ in Γ_{D_i}

where $\sigma_{ij} = c_{ijkl}\left(\varepsilon_{kl} + \theta\,\alpha_{kl}\right) + \sigma_{ij}^{0}$

A.2 WEAK FORM

Let us multiply $\sigma_{ij,j} = 0$ by $w_i \in V_i$ and integrate over Ω $\int_{\Omega} w_i \sigma_{ij,j}\, d\Omega = 0$

(integration by parts)

$-\int_{\Omega} w_{i,j}\sigma_{ij}\, d\Omega + \int_{\Gamma} w_i \sigma_{ij} n_j\, d\Omega = 0$ ($w_{i,j}\sigma_{ij} = w_{(i,j)}\sigma_{ij}$ since σ_{ij} is symmetric tensor)

$-\int_{\Omega} w_{(i,j)}\sigma_{ij}\, d\Omega + \int_{\Gamma} w_i \sigma_{ij} n_j\, d\Omega = 0$ ($w_i = 0$ on Γ)

$\int_{\Omega} w_{(i,j)}\sigma_{ij}\, d\Omega = 0$ ($\sigma_{ij} = c_{ijkl}\left(\varepsilon_{kl} + \theta\,\alpha_{kl}\right) + \sigma_{ij}^{0}$)

$\int_{\Omega} w_{(i,j)} c_{ijkl}\varepsilon_{kl}\, d\Omega = -\theta\int_{\Omega} w_{(i,j)} c_{ijkl}\alpha_{kl}\, d\Omega - \int_{\Omega} w_{(i,j)}\sigma_{ij}^{0}\, d\Omega$ ($\varepsilon_{ij} = u_{(i,j)}$)

$\int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)}\, d\Omega = -\theta\int_{\Omega} w_{(i,j)} c_{ijkl}\alpha_{kl}\, d\Omega - \int_{\Omega} w_{(i,j)}\sigma_{ij}^{0}\, d\Omega$

Weak form of the boundary-value problem

Given $g_i : \Gamma_{D_i} \ni x \to g_i(x) = 0 \in R$, θ , α_{kl} *and* σ_{ij}^{0} , *find* $u_i \in V_i$ *such that*

$\int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)}\, d\Omega = -\theta\int_{\Omega} w_{(i,j)} c_{ijkl}\alpha_{kl}\, d\Omega - \int_{\Omega} w_{(i,j)}\sigma_{ij}^{0}\, d\Omega$ *for all* $w_i \in V_i$

A.3 IMPLEMENTATION ISSUES

Let us introduce the following abstract notation:

Given $\mathbf{f, g, h}$ *find* $\mathbf{u} \in \mathbf{V}$ *such that* $\mathbf{a}(\mathbf{w},\mathbf{u}) = -\mathbf{A}(\mathbf{w}) - \boldsymbol{\Sigma}(\mathbf{w})$ *for all* $\mathbf{w} \in \mathbf{V}$

$\mathbf{a}(\mathbf{w}, \mathbf{u}) = \int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)}\, d\Omega$

$\mathbf{A}(\mathbf{w}) = \theta\int_{\Omega} w_{(i,j)} c_{ijkl}\alpha_{kl}\, d\Omega$

$\boldsymbol{\Sigma}(\mathbf{w}) = \int_{\Omega} w_{(i,j)}\sigma_{ij}^{0}\, d\Omega$

$\boldsymbol{\varepsilon}(\mathbf{u}) = \begin{Bmatrix} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \end{Bmatrix}$

$$\boldsymbol{\varepsilon}(\mathbf{w}) = \begin{Bmatrix} w_{1,1} \\ w_{2,2} \\ w_{1,2} + w_{2,1} \end{Bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix}$$

where

- $\lambda = \dfrac{\nu E}{(1+\nu)(1-2\nu)}$
- $\mu = \dfrac{E}{2(1+\nu)}$
- $E = 1GPa$ is Young modulus
- $\nu = 0.3$ is Poisson ratio

$\boldsymbol{\alpha} = \begin{Bmatrix} \alpha_{11} \\ \alpha_{22} \\ 2\alpha_{12} \end{Bmatrix}$ I assume symmetry $\alpha_{12} = \alpha_{21}$, thus $\alpha_{12} + \alpha_{21} = 2\alpha_{12}$

$$\boldsymbol{\sigma}^{\mathbf{0}} = \begin{Bmatrix} \sigma_{11}^0 \\ \sigma_{22}^0 \\ \sigma_{12}^0 \end{Bmatrix}$$

$$\mathbf{a}(\mathbf{w}, \mathbf{u}) = \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w})^{\mathbf{T}} \mathbf{D} \boldsymbol{\varepsilon}(\mathbf{u}) d\Omega$$

$$\mathbf{A}(\mathbf{w}) = \theta \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w})^{\mathbf{T}} \mathbf{D} \boldsymbol{\alpha}\, d\Omega$$

$$\boldsymbol{\Sigma}(\mathbf{w}) = \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w})^{\mathbf{T}} \boldsymbol{\sigma}^{\mathbf{0}}\, d\Omega$$

I assume $g_i : \Gamma_{D_i} \ni x \to g_i(x) = 0 \in R$, $\theta = 1$ (temperature gradient), $\{\alpha_{kl}\}_{k,l=1,2} = \begin{bmatrix} -0.06115 & 0 \\ 0 & -0.06115 \end{bmatrix}$ and $\sigma_{ij}^0 = 0$ (zero initial stress).

Appendix B. STEP-AND-FLASH IMPRINT LITHOGRAPHY

Step-and-Flash Imprint Lithography (SFIL) constitutes an important patterning framework used in silicon industry [D5]. The process consists of the following phases, decapitated in Figure B.1.

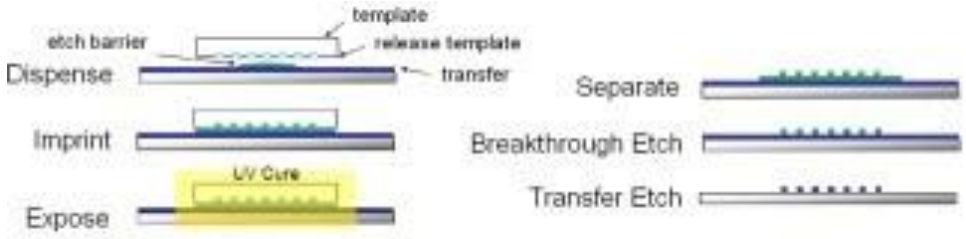

Figure B1. Steps of SFIL

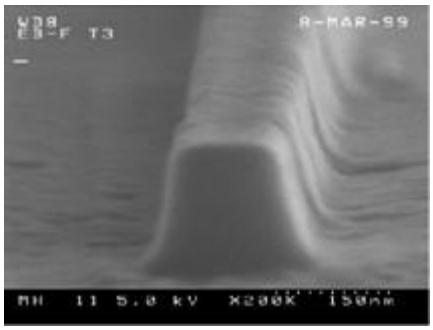

Figure B.2 Shrinkage of the feature after removal of the template. the picture courtesy of prof. Grant C. Wilson from University of Texas at Austin

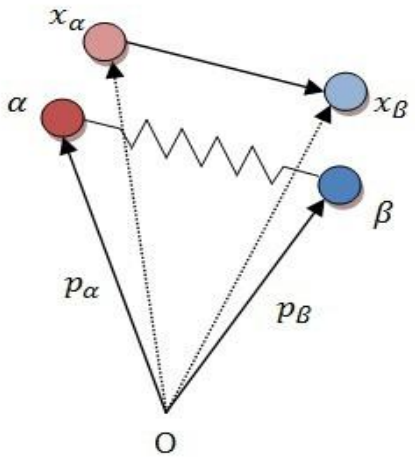

Figure B.3 A pair of interacting particles

- dispense - depositing a low viscosity silicon containing photocurable etch barrier onto a substrate,
- imprint - bringing the template into contact with the etch barrier,
- expose - exposing the etch barrier to UV in order to cure it,
- separate - releasing the template,
- breakthrough etch - a short, halogen etch,
- transfer etch - an anisotropic reactive ion etch to yield high aspect ratio.

Photopolymerization is often accompanied by densification. The interaction potential between photopolymer precursors undergoing free radical polymerization changes from van der Waals' to covalent. The average distance between molecules decreases and causes volumetric contraction. Densification of the SFIL photopolymer (the etch barrier) may affect both the cross sectional shape of the feature and the placement of relief patterns. The exemplary shrinkage of the feature measured after removing the template is presented in Figure B.2.

The macro-scale part of the computational domain is modeled by the linear elasticity with thermal expansion coefficient. Following [C23] the strong and weak formulations for the linear elasticity problem with thermal expansion coefficient are given as follows.

B.1 STRONG FORMULATION

Given $g_i : \Gamma_D \ni x \rightarrow g_i(x) \in R$, θ and α_{kl}, find the displacement vector field

$u_i : \Omega^- \ni x \rightarrow u_i(x) \in R, i = 1,2,3$, such that

$$\sigma_{ij,j} = 0 \; in \; \Omega \tag{B.1}$$

$$u_i = g_i \; on \; \Gamma_D \tag{B.2}$$

where σ_{ij} is the stress tensor, defined in terms of the generalized Hook's law

$$\sigma_{ij} = c_{ijkl}(\epsilon_{kl} + \theta\alpha_{kl}) \tag{B.3}$$

here c_{ijkl} are elastic coefficients (known for given material), θ is the temperature, α_{kl}

are the thermal expansion coefficients, and $\epsilon_{kl} = u_{(i,j)} = \frac{u_{i,j}+u_{j,i}}{2}$ is the strain tensor,

where $u_{i,j}$ are displacement gradients.

B.2 WEAK FORMULATION

The weak formulation is obtained by multiplying (1) by test functions $w_i \in H_0^1(\Omega)$ and integrating by parts over Ω:

$$-\int_\Omega w_{i,j}\sigma_{ij}d\Omega + \int_\Gamma w_i\sigma_{ij}n_j d\Omega = 0 \tag{B.4}$$

Since σ_{ij} is symmetric tensor, then $w_i\sigma_{ij} = w_{(i,j)}\sigma_{ij}$ and

$$\int_\Omega w_{(i,j)}\sigma_{ij}d\Omega = 0 \tag{B.5}$$

where we have also used the fact that $w_i = 0$ on Γ. Finally, by utilizing (3) we get

$$-\int_\Omega w_{(i,j)}c_{ijkl}u_{(k,l)}d\Omega = -\theta\int_\Gamma w_{(i,j)}c_{ijkl}\alpha_{kl}d\Omega \tag{B.6}$$

B.3 REFORMULATION FOR THE SFIL MODELING

For the convenient implementation of the algorithm, we utilize the following equivalent weak formulation. Find $u \in V$, such that

$$a(u,w) = -A(w)\forall w \in V \tag{B.7}$$

$$a(u,w) = \int_\Omega \epsilon(w)^T D\epsilon(w)d\Omega \tag{B.8}$$

$$A(w) = \theta\int_\Omega \epsilon(w)^T D\alpha d\Omega \tag{B.9}$$

Where $V = \left\{v \in \left(H^1(\Omega)\right)^3 : tr\ v = 0\ on\ \Gamma_D\right\}$, and Γ_D is defined as the bottom of the 3D cube.

Here

$$\epsilon(u) = \begin{pmatrix} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ u_{2,3}+u_{3,2} \\ u_{1,3}+u_{3,1} \\ u_{1,2}+u_{2,1} \end{pmatrix} \tag{B.10}$$

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix} \tag{B.11}$$

The linear elasticity with thermal expansion coefficient can be applied to SFIL process modeling with the Young modulus $E = 1GPa$ and Poisson ratio $\nu = 0.3$, as provided by [C24]. We also assume that $g_i : \Gamma_D \ni x \rightarrow g_i(x) = 0$ (the feature is fixed at the bottom and the other sides are treated as either free boundary conditions for the simulations outside the template or as the nano-scale model interface for the simulations inside the template), $\theta = 1$ (the thermal expansion coefficient α expresses the volumetric contraction of the feature when the temperature gradient is equal to 1^oC), $\alpha_{ij} = -\alpha\delta_{ij}$ where $\alpha = -0.0615$ is based on inverse analysis [C25].

B.4 MOLECULAR STATICS MODELS

We consider a regular rectangular 3D grid with interacting particles, as presented in Figure B.3. Each particle interacts with its 26 neighbors. For each pair of interacting particles α and β we can distinguish their initial configurations p_α , p_β and (unknown) equilibrium configurations x_α, x_β (compare Figure B.3).

Over the nano-scale domain, we employ a non-linear model assuming large deformations and quadratic potentials for networks of particles forming polymer chains [C7]. In this model the force between pair of interacting particles α and β is given by

$$F_{\alpha\beta} = k_{\alpha\beta}\left(r_{\alpha\beta} + \Delta r_{\alpha\beta} - r^0_{\alpha\beta}\right)\frac{(p_\beta - p_\alpha)}{\|p_\beta - p_\alpha\|} \tag{B.12}$$

where $k_{\alpha\beta}$ is the spring stiffness coefficient, $r_{\alpha\beta} + \Delta r_{\alpha\beta} = kx_\beta - x_\alpha k$ is the length of the sprig in the equilibrium configuration, x_α , x_β represent the (unknown) equilibrium configuration of particles, $r^0_{\alpha\beta}$ is the length of the unscratched spring and p_α, p_β represent the initial configuration of particles.

The remaining interactions of particles (not along the polymer chain) are given by a non-linear model assuming large deformations and Lennard-Jones potential. In this model the force between pair of interacting particles α and β is given by

$$V_{\alpha\beta}(r_{\alpha\beta}) = \left(\frac{\sigma_{\alpha\beta}}{r_{\alpha\beta}}\right)^{n_{\alpha\beta}} - \left(\frac{\sigma_{\alpha\beta}}{r_{\alpha\beta}}\right)^{m_{\alpha\beta}} \tag{B.13}$$

where $\sigma_{\alpha\beta}$, $n_{\alpha\beta}$ and $m_{\alpha\beta}$ are parameters of the potential and the inter-particle forces are computed with

$$F_{\alpha\beta}(r_{\alpha\beta}) = \frac{\partial V_{\alpha\beta}(r_{\alpha\beta})}{\partial r} \tag{B.14}$$

Large deformations are observed here, which implies the direction of the inter-particle forces along the resulting spring alignments $x_\beta - x_\alpha$. The Molecular Statics problem consists in finding the equilibrium configuration of particles satisfying

$$\sum_\beta F_{\alpha\beta} = 0 \tag{B.15}$$

for $\alpha = 1..N$ (total number of particles). More mathematical details of the molecular statics model formulation are presented in [C7].

B.5 Coupling macro-scale and nano-scale models

Macro-scale and nano-scale models are coupled by identifying the particles located on the interface of the nano-scale domain with the corresponding nodes of the FEM mesh, located on the interface.

In such case, we solve Molecular Statics equations inside the nano-scale domain, and Linear Elasticity with Thermal Expansion coefficient discretized by FEM inside the macro-scale domain, while the interface between macro- and nano-scales is treated in the special way. The nodes of the FEM mesh, from the viewpoint of the nano-scale model are identified with particles represented by their positions x_α, while from the viewpoint of the macro-scale model, the nodes are understood in the traditional way as the degrees of freedom, representing the displacements u_α. It implies the additional coupling equations

$$u_\alpha = \Delta p_\alpha = x_\alpha - p_\alpha \tag{B.16}$$

for each particle (FEM mesh node) α located on the interface between nano- and macro-scales.

In practice, it is not necessary to add the coupling equation (16) to the system, but only to aggregate the FEM and MS equations to the same global matrix.

Appendix C. DESIGN OF THE OBJECT-ORIENTED APPLICATION FOR MULTI-SCALE SIMULATIONS

In this appendix, we focus on the particular aspects of the O-O design of the hp-FEM application. The Unified Modeling Language [C41] is utilized as a tool for the description of the modeled system.

C.1 MESH – CONCEPT

The major part of the problem domain is solved with the finite element method. For this purpose the domain is divided into numerous finite elements creating a mesh. A mesh is being built according to the Euler's model. The model assumes that the element consists of many nodes. A node is such a part of mesh on which we can define a shape function. Vertices are the most primitive kind of them. An edge is delimited by two vertices, a face – by four edges, an interior – by six faces and so on, depending on the domain's dimension. By the order of the node we mean the polynomial order of approximation utilized over the node, An element contains several first order vertex nodes. 1D element contains one higher order node, associated with an edge, 2D element contains four higher order edge nodes, and one higher order interior node, and so on.

C.2 MESH – IMPLEMENTATION

Mesh is implemented quite straightforward: there are separate classes for each kind of node, all of them derived from the abstract base class *Node*, as illustrated in Figure C.1. All nodes but vertices are called refinable – they can undergo the h-adaptation process. Node classes are designed in such a way, that no matter the dimension of the problem is, the core of hp-FEM code remains the same. Not surprisingly, the basic difference lies in the way of storing of nodes coordinates. For this reason there is an extra subclass of *Point* class, for each spatial dimension. The *Element* class stays unchanged too, thanks to the polymorphic calls of the *highestOrderNode*'s methods.

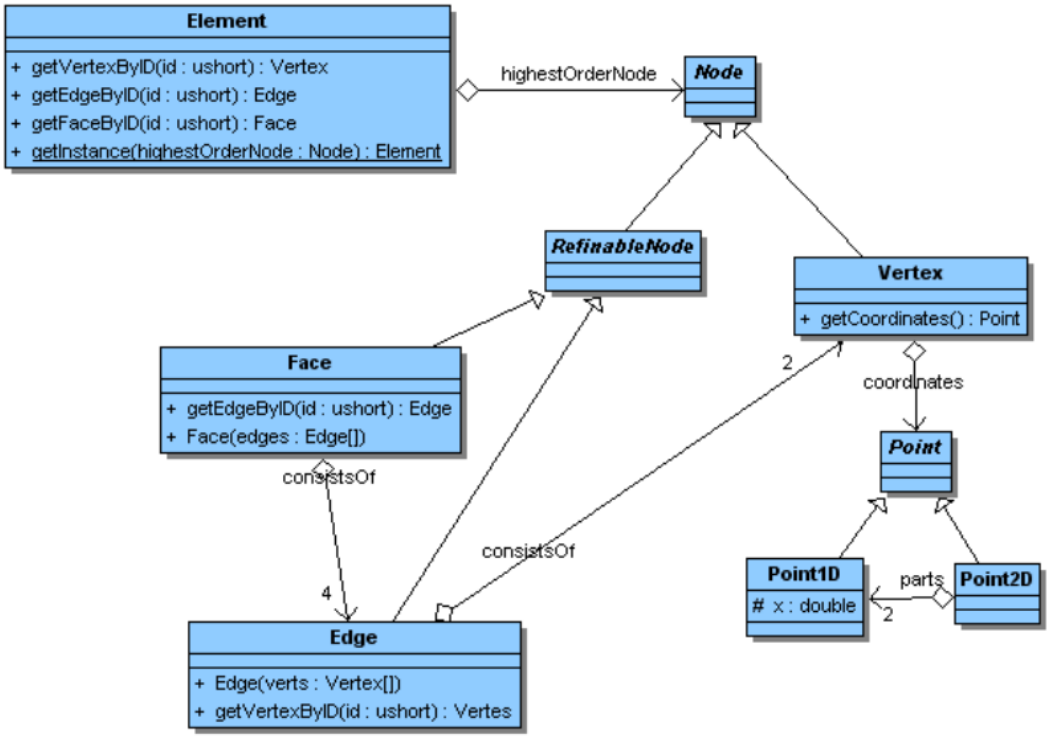

Figure C.1. Mesh class diagrams.

C.3 SHAPE FUNCTIONS - CONCEPT

The solution of the variational problem is approximated in the base of so called global shape functions. Global shape functions are associated with some nodes and have support over one or several neighboring elements. Global shape function's restriction into a single finite element is called a local shape function. These single multi-dimensional polynomials (no longer splines) are, in contrast, linked to finite elements (not nodes). Typically in 1D we use shape functions of orders up to 9. Shape functions are related with node objects. Each vertex node has associated one global shape function that restricts into one local shape function over each element having the vertex. Each edge node has associated one or more global shape functions, that restrict into one local shape function for each element having the edge. All higher order nodes (faces or interiors in 2D and 3D) has associated one or more global shape functions, that restrict to one local shape function over each element having the node.

C.4 SHAPE FUNCTIONS – IMPLEMENTATION

Our application is dedicated to the special kind of hierarchical shape functions, defined in [C12], where subsequent functions are tensor product of the previous ones. The one-dimensional shape functions are defined as

$$\kappa_1(\xi)=1-\xi \quad \kappa_2(\xi)=\xi \quad \kappa_3(\xi)=(1-\xi)\xi \quad \kappa_n(\xi)=\kappa_{n-1}(\xi)(\kappa_2(\xi)-\kappa_1(\xi)) \tag{C.1}$$

Two-dimensional shape functions are defined as tensor products of the 1D shape functions. Here we present some examples, e. g. the shape function of the left-bottom element vertex:

$$\phi_2(\xi_1,\xi_2)=\kappa_1(\xi_1)\kappa_2(\xi_2)=(1-\xi_1)\xi_2 \tag{C.2}$$

shape functions of the bottom edge:

$$\phi_{5,j}(\xi_1,\xi_2)=\kappa_{2+j}(\xi_1)\kappa_1(\xi_2), \quad j=1,\ldots,p_1-1 \tag{C.3}$$

and bubble shape functions

$$\phi_{9,ij}(\xi_1,\xi_2)=\kappa_{2+i}(\xi_1)\kappa_{2+j}(\xi_2), \quad i=1,\ldots,p_h-1,\ j=1,\ldots,p_v-1 \tag{C.4}$$

The full definition for all shape functions can be found in [C12]. *LocalShapeFunction2D* is composed of *LocalShapeFunction1Ds* to reflect the fact that each multi-dimensional polynomial originates from the tensor product of simple polynomials. This dependence is illustrated in Figure C.2. Note analogical composition in the Point class hierarchy. *LocalShapeFunctions* and *GlobalShapeFunctions* are required in our O-O implementation. The *globalParent/localMemebers* mapping is crucial in the process of equation generation. In the O-O implementation, it is most natural to define the *LocalShapeFunctions* and *GlobalShapeFunctions* objects, since all procedures, like computing the derivative of the function, or value of the function at given point, are located inside these objects. Also, the higher dimension shape function objects can be composed by tensor products of lower dimension shape function, which can be directly expressed on the UML diagrams. The design of the O-O application is different from the procedural approach.

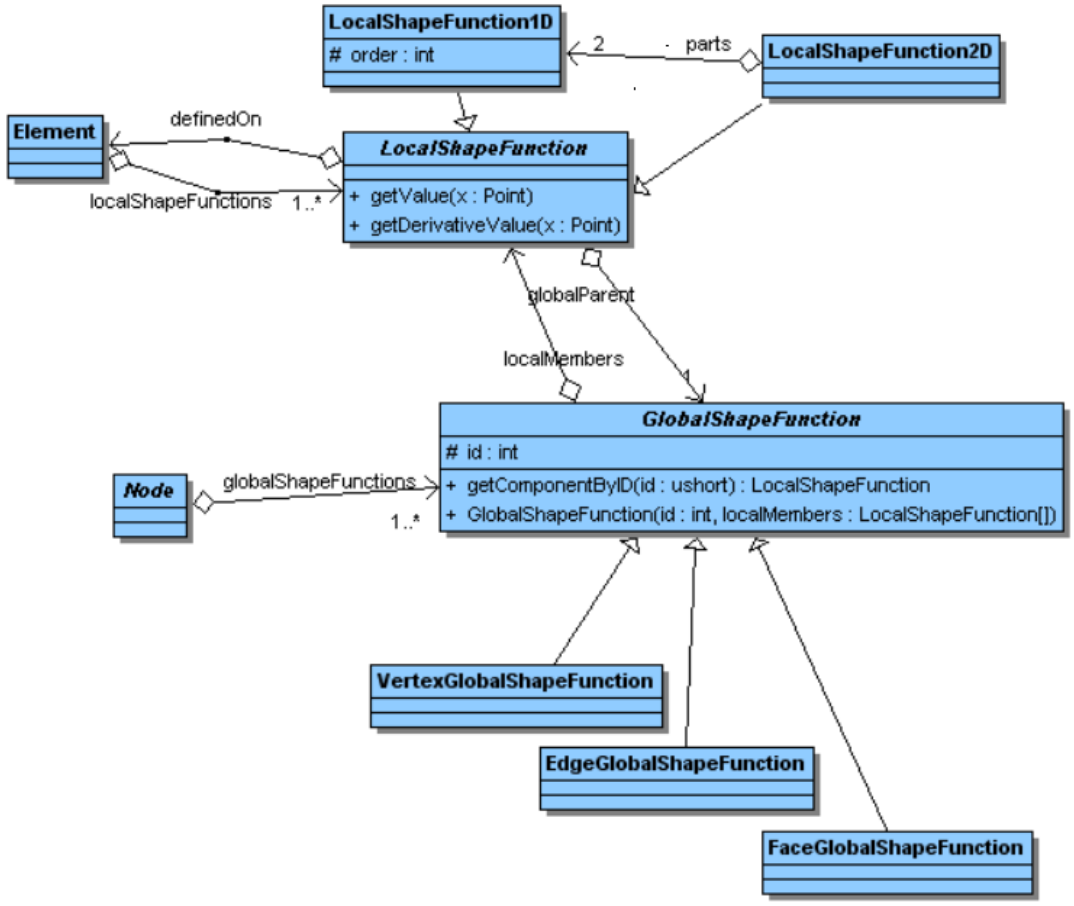

Figure C.2. Shape functions class diagrams.

C.5 Collections

All the introduced objects are kept inside the *Mesh* class, in several collections. Nodes are stored in a set, which helps identifying those with the same coordinates (only one object exists for each combination of coordinates). Note that there is no need to keep *LocalShapeFunction* objects in separate collection, since these objects are accessed only from Elements.

C.6 Adaptation

The quality of hp-FEM solution can be improved in two ways: p-adaptation – by increase of the maximal order of shape functions defined on an element, and h-adaptation – by breaking selected elements (and though – nodes).

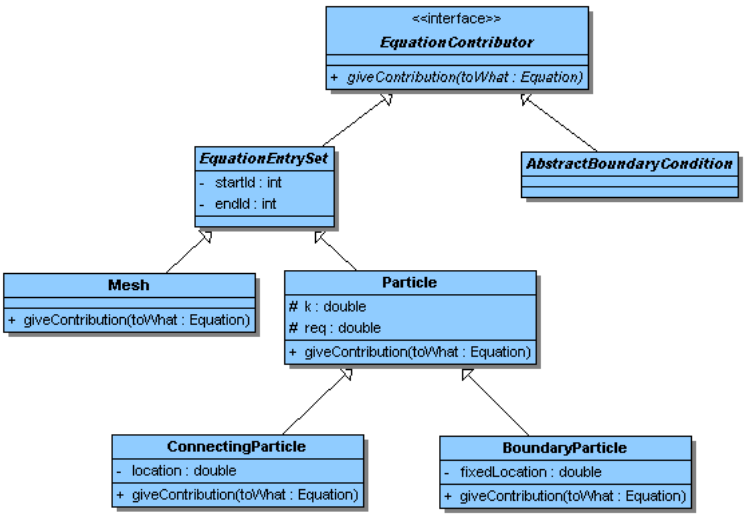

Figure C.3. Dependencies between particle classes of different types.

C.7 P-ADAPTATION

The key idea behind the p-adaptation is to extend number of the shape functions for the adapted element. Thanks to the increase of the polynomial approximation level in the base, the solution becomes smoother. In our code the p-adaptation is as simple as creating a new *LocalShapeFunction* object of an appropriate order and registering it with the element.

C.8 H-ADAPTATION

The error of the approximate solution can be also decreased by breaking some elements into smaller elements (by increasing the number of elements, or by reducing their sizes). Some sensitive places may require a lot of elements for accurate approximation of the solution, whereas relatively sparse mesh is acceptable form some other places. The key factor in achieving the satisfactory results is to find these sensitive places, which demand to be approximated with the use of more elements. In can be done manually by predicting solution features or automatically by refining some elements based on the evaluation of the error decrease rate [C12].

C.9 MOLECULAR STATICS PART (MESO-SCALE)

In some places, which normally would require very intensive adaptation (because of enormous error rate of the FEM), it is much more sensible to switch from the macro-scale to the meso-scale. This leads to the formulation of the computational problem in

terms of the particle interaction models. Considering the number of particles in the average problem domain it is usually impossible to solve the whole problem in such a way, however it turns out perfect when it comes to some local peculiarities. Fortunately, molecular computations do not mean any profound changes in our model. It is enough to adapt some of the existing components to their new roles and introduce several new ones.

In addition to the classes already introduced for the hp-FEM model, we have created the Particle class tree, which – similarly to the Mesh class inherits from the *EquationEntrySet* abstract class. The tree is illustrated in Figure C.3. Its aim is to add some additional information to the *EquationContributor* interface, pointing its precise location in the equation matrix. There are several types of Particles, depending on where the particle is located:

StandardParticle located inside the meso-scale part, with the following attributes r_{eq}, k – interparticle interaction coefficients, with unknown location x_i

ConnectingParticle – located on the macro- / meso- scales interface. These particles have to manage both macro-scale and meso-scale variables, thus the location variable (from the meso-scale) and the displacement degrees of freedom (for the macro-scale) must be stored in such particles. The macro- / meso- scale interface condition is stored in a form of *AbstractBoundaryCondition* subclass

BoundaryParticle which represents the Dirichlet boundary condition enforced on the non-scale level. In such the particles, the location x_i of the particles is fixed, with nothing to be computed

All of the equations are stored in the common matrix. Given the solution vector, we have to transform the meso-scale results, defined as the location of particular particles, into the displacement field, utilized in the macro-scale computations.

C.10 Generalization to higher dimensions

C.10.1 0-0 hierarchical data structures

The computational mesh is designed based on the Euler's hierarchical model. Higher dimension objects are composed with several lower dimension objects, as it is

presented in Figure 6. An *edge* consists in two *vertices*, a *face* consists in four edges, an *interior* consists in six faces. Technically speaking *Vertex2D* is not the same as *Vertex1D*, since e.g. it contains more coordinates. However, the general spatial dimension independent *Vertex* template can be created, and dimension dependent object can be created by parameterizing the general template. The *Element* object consists in several *Node* objects, stored at *NodesList* object as it is presented in Figure C.6. The *Element1D* consists in two vertices and one edge, the *Element2D* consists in four vertices, four edges and one face, the *Element3D* consists in eight vertices, twelve edges, six faces and one interior. There are also *ParticleElements* classes representing a nano-scale elements filled with particles. All vertices, edges and interiors inherit from the *Node* object.

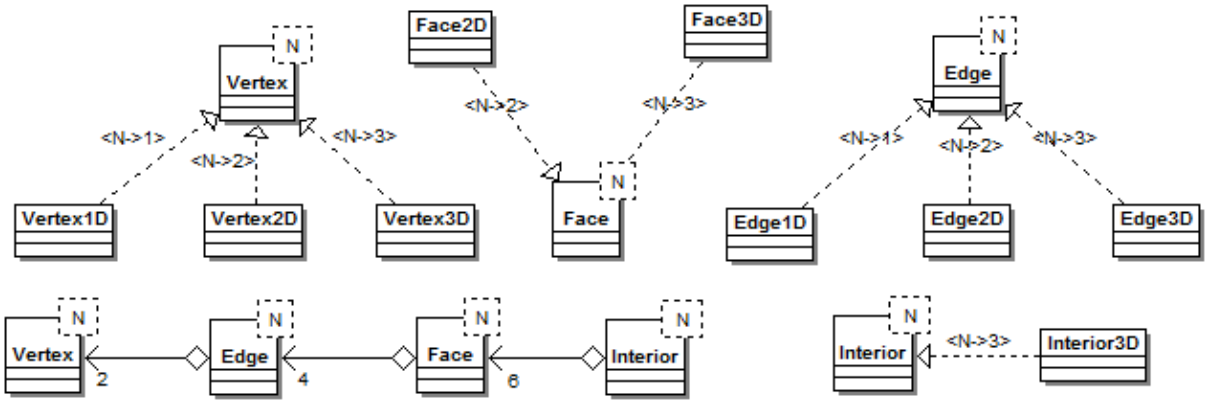

Figure C.4. Class diagram illustrating relations between nodes: vertices, edges, faces and interiors.

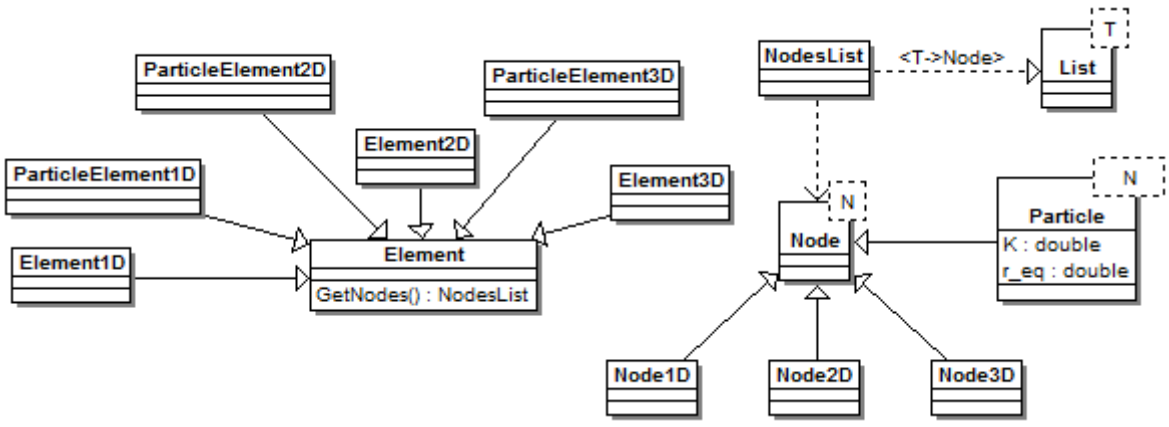

Figure C.5. Class diagram illustrating general relation between *Element* and *Node* classes.

The solution of the variational problem is approximated in the base of so called *global shape functions*. Global shape functions are associated with nodes and have supports over one or several neighboring elements. A restriction of a global shape function

into an element is called *local shape function*. For the purpose of the UML project we utilize special kind of hierarchical local shape functions [C14], where higher spatial dimension functions are tensor products of several lower dimension functions. The basic 1D shape functions are defined by the following recursive formulae:

$$\kappa_1(\xi) = 1-\xi \text{ , } \kappa_2(\xi) = \xi \text{ , } \kappa_3(\xi) = (1-\xi)\xi \text{ , } \kappa_n(\xi) = \kappa_{n-1}(\xi)(\kappa_2(\xi) - \kappa_1(\xi)) \quad \text{(C5)}$$

We distinguish κ_1 and κ_2 as VertexLocalShapeFunction1D, and κ_n n>=3 as EdgeLocalShapeFunction1D. The VertexLocalShapeFunction2D are defined as tensor products of two VertexLocalShapeFunction1D

$$\phi_1(\xi_1,\xi_2) = \kappa_1(\xi_1)\kappa_1(\xi_2) = (1-\xi_1)(1-\xi_2) \text{ , } \phi_2(\xi_1,\xi_2) = \kappa_2(\xi_1)\kappa_1(\xi_2) = \xi_1(1-\xi_2) \quad \text{(C6)}$$

$$\phi_3(\xi_1,\xi_2) = \kappa_2(\xi_1)\kappa_2(\xi_2) = \xi_1\xi_2 \text{ , } \phi_4(\xi_1,\xi_2) = \kappa_1(\xi_1)\kappa_2(\xi_2) = (1-\xi_1)\xi_2$$

The EdgeLocalShapeFunction2D are defined as tensor products of one VertexLocalShapeFunction1D, and one VertexLocalShapeFunction1D.

$$\phi_{5,j}(\xi_1,\xi_2) = \kappa_{2+j}(\xi_1)\kappa_1(\xi_2) \text{ j=1,…p1-1; } \phi_{6,j}(\xi_1,\xi_2) = \kappa_2(\xi_1)\kappa_{2+j}(\xi_2) \text{ j=1,…p2-1; } \quad \text{(C7)}$$

$$\phi_{7,j}(\xi_1,\xi_2) = \kappa_{2+j}(1-\xi_1)\kappa_1(\xi_2) \text{ j=1,…p3-1; } \phi_{8,j}(\xi_1,\xi_2) = \kappa_1(\xi_1)\kappa_{2+j}(1-\xi_2) \text{ j=1,…p4-1}$$

where p_k is the polynomial order of approximation over an edge. The *FaceLocalShapeFunction2D* are defined as tensor products of two *EdgeLocalShapeFunction1D*

$$\phi_{9,ij}(\xi_1,\xi_2) = \kappa_{2+i}(\xi_1)\kappa_{2+j}(\xi_2) \text{ i=1,…,p}_h\text{-1, j=1,…,p}_v\text{-1} \quad \text{(C8)}$$

The relations are expressed in the class diagram presented in Figure C.7. The local shape functions are again created by instantiating template *LocalShapeFunction*, as it is presented in Figure C.9.

C.10.2 HIERARCHICAL ALGORITHMS

In the proposed UML project, the basic objects are created by instantiating general templates. Thus, they methods must be defined within these templates as spatial dimension independent. The exemplary dimension independent algorithm for

computing a value of the shape function is listed below. The *N* stands for the spatial dimension, and *Value* stands for either double precision or complex arithmetic.

```
template <int N, class Value>
class LocalShapeFunction{
LocalShapeFunction1D mLocalShapeFunction1D[N];
Value GetValue(Point<N> P){
   value=1.0;
   for(int i=1; i<=N; ++i)
  value*=mLocalShapeFunction1D[i].GetValue(P[i]);
```

We have analyzed algorithms utilized in [C14] in the existing procedural version of 1D, 2D and 3D *hp*-FEM. Most of the algorithms are actually designed in a hierarchical manner, since they call procedures implemented for lower spatial dimension objects. Let us discuss the issue on the *h* adaptation algorithm example. The process of refining e.g. a 2D face consists in two processes of refining 1D edges and then linking them to make a new faces. The process of refining an edge is, consequently described as the process of making new vertices and linking them into new edges. Newly created nodes are linked on father / sons lists from existing nodes, as it is presented in Figure C.8. The active finite elements are dynamically reconstructed after the refinement process is finished. In addition to the classes already introduced, we have created the *Particle* class, which is a special kind of node, storing the nano-scale particle, utilized during modeling of the inter-particle interactions in the nano-scale. The particles are collected within element objects. Thus, there are macro-scale elements collecting multiple *Nodes*, and nano-scale elements, collecting multiple particles, as it has been already illustrated in Figure C.5.

C.10.3 NODE-BASED SOLVER

The next essential part of the multiscale application is an efficient multi-frontal solver. We have developed a node based solver, suitable for the adaptive multi-physics computations, utilizing the hyper-matrix, working on the level of *Nodes*, independent on the spatial dimension and the polynomial order of approximation. The new solver has been described in [C42] based on the previous version [C43].

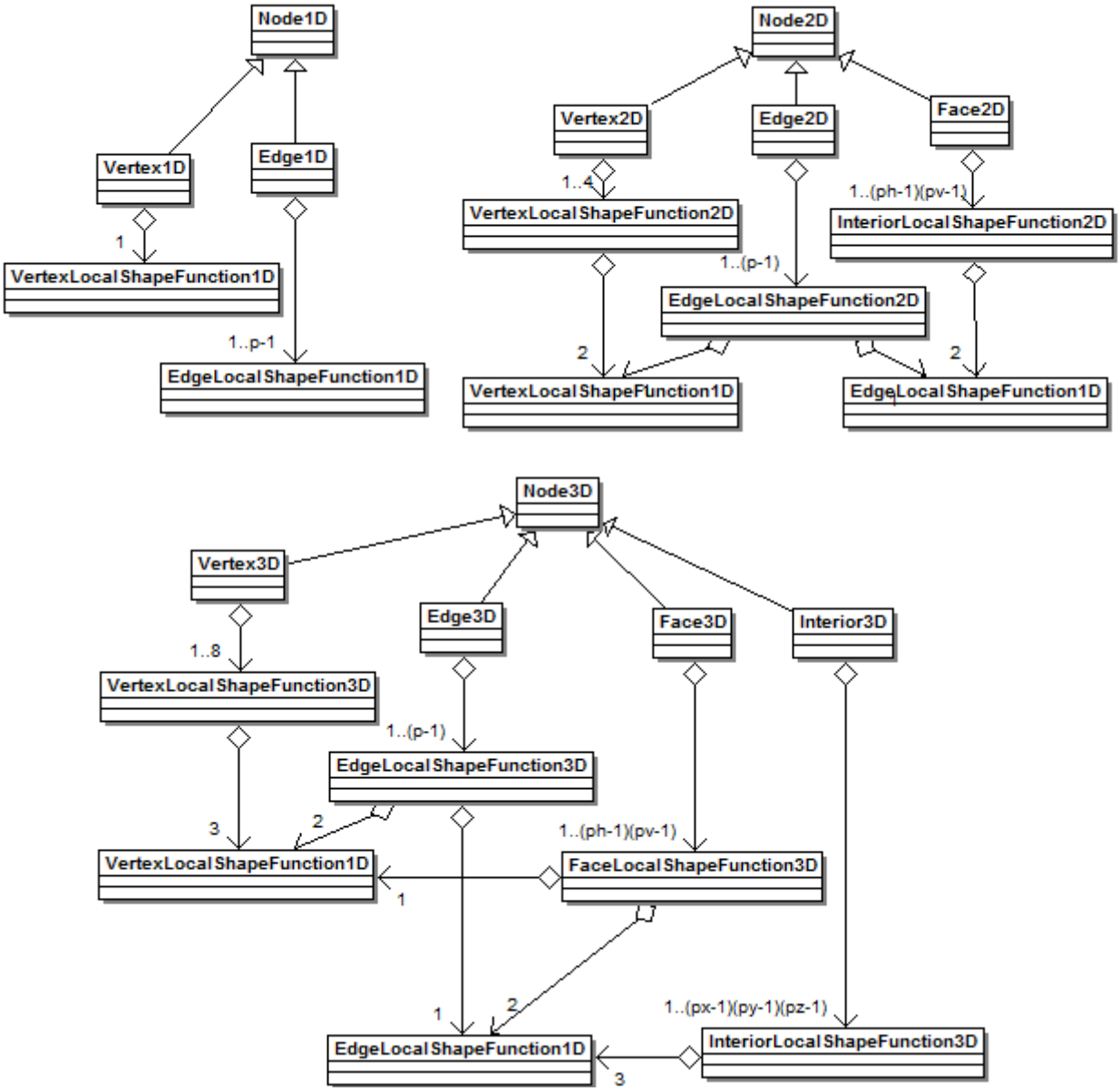

Figure C.6 Class diagrams illustrating general relations between local shape function and node classes.

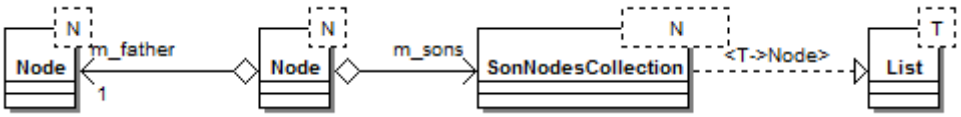

Figure C.7 Class diagram illustrating relations between father and son nodes in the refinement tree

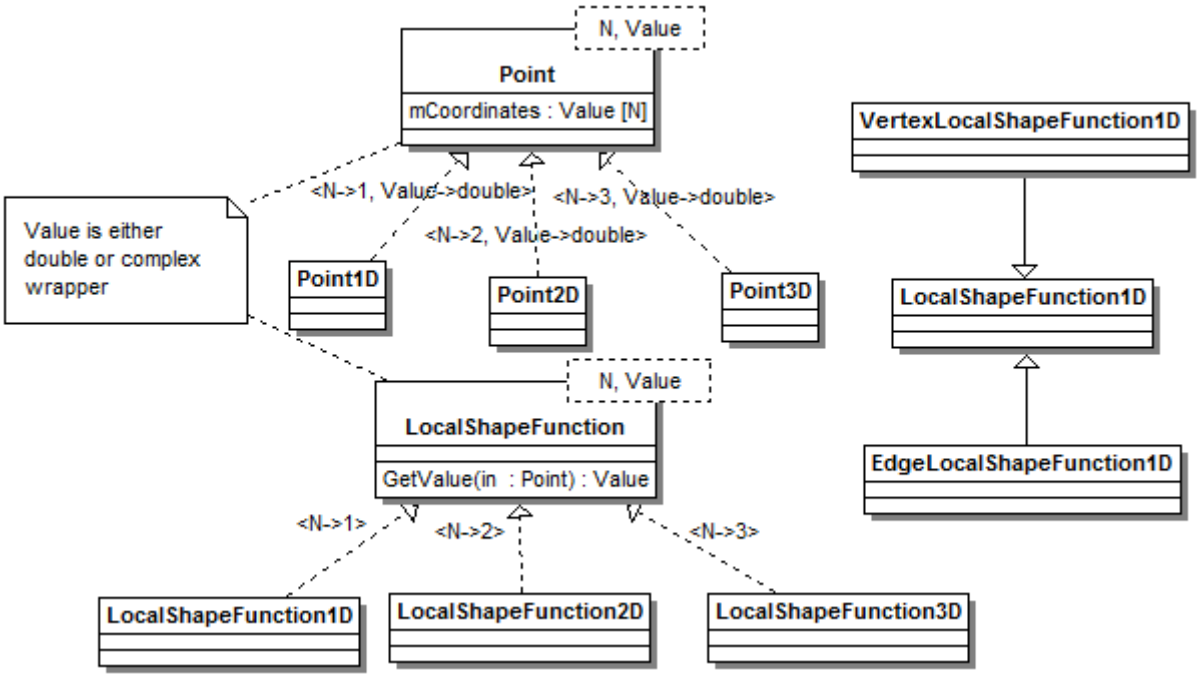

Figure C.8 Class diagram illustrating relations between father and son nodes in the refinement tree

5 PhD Candidate Accomplishments

Below are enlisted publications co-authored by the author of this thesis along with their MNiSW (Polish Ministry of Science and Higher Education) classification. IF stands for Impact Factor as indexed on the Thomson Journal Citation Reports list. LF indicates that the journal is indexed by the Thomson Journal Master List.

5.1 List A

[D1] Ryszka I., Paszyńska A., Grabska E., Paszyński M., **Sieniek M.**, 2015, Graph grammar systems for modeling three dimensional finite element method Part I, accepted by **Fundamenta Informaticae**, IOS Press

LF, IF: 0.479, MNiSW: 15pts (List A)

[D2] Ryszka I., Paszyńska A., Grabska E., Paszyński M., **Sieniek M.**, 2015, Graph grammar systems for modeling three dimensional finite element method, Part II. accepted by **Fundamenta Informaticae**, IOS Press

LF, IF: 0.479, MNiSW: 15pts (List A)

[D3] **Sieniek M.**, Paszyński M., Madej Ł., Goik D., 2014, Adaptive Projection-Based Interpolation as a pre-processing tool in the Finite Element workflow for elasticity simulations of the dual phase microstructures, **Steel Research International**, vol. 85, issue 6, pp. 1109-1119, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim

LF, IF: 1.023, MNiSW: 25pts (List A)

[D4] **Sieniek M**., Woźniak M., Paszyński M., Demkowicz L., 2015, Quasi-linear computational cost adaptive solvers for three dimensional modeling of the heating of the human head induced by cell-phone, in review to **Journal of Computational Science**, Elsevier

LF, IF: 1.567, MNiSW: 30pts (Web of Science)

[D5] Gurgul P., **Sieniek M.**, Magiera K., Skotniczny M., 2013, Application of multi-agent paradigm to hp-adaptive projection-based interpolation operator, **Journal of Computational Science**, Elsevier, vol. 4(3), pp. 164–169

LF, IF: 1.567, MNiSW: 30pts (Web of Science)

[D6] **Sieniek M.**, Paszyński M., 2014, Subtree reuse in multi-frontal solvers for regular grids in Step-and-Flash Imprint Nanolithography modelling, **Advanced Engineering Materials**, ISSN 1438-1656, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, vol. 16(2), pp. 231-240

LF, IF: 1.508, MNiSW: 30pts (List A)

5.2 List B

[D7] Gurgul P., **Sieniek M.**, Paszyński M., 2009, Object-oriented Multiscale HP-Adaptive Finite Element Method, Computer Methods in Materials Science, ISSN 1641-8581, Paragraph, vol. 9, pp. 289-295

MNiSW: 10pts (List B)

[D8] Gurgul P., **Sieniek M.**, Paszyński M., Madej Ł., 2013, Three-dimensional adaptive algorithm for continuous approximations of material data using space projection, Computer Methods in Materials Science, ISSN 1641-8581, Paragraph, vol. 13(2), pp. 245-250

MNiSW: 10pts (List B)

[D9] Gurgul P., **Sieniek M.**, Paszyński M., Madej Ł., Collier N., 2013, Two dimensional hp-adaptive algorithm for continuous approximations of material data using space projections, Computer Science, AGH University of Science and Technology Press, vol. 14(1), pp. 97-112

MNiSW: 8pts (List B)

5.3 Web of Science Conferences

[D10] Goik D., **Sieniek M.**, Paszyński M., Madej Ł., 2013, Employing an adaptive projection-based interpolation to prepare discontinuous 3D material data for finite element analysis, Procedia Computer Science, Elsevier, vol. 18, pp. 1535–1544, conference proceedings of: International Conference on Computational Science, Barcelona, Spain, 05-07.06.2013

MNiSW: 10pts (Web of Science)

[D11] Goik D., **Sieniek M.**, Woźniak M., Paszyńska A., Paszyński M., Hypergraph Grammar based Adaptive Linear Computational Cost Projection Solvers for Two and Three Dimensional Modeling of Brain, Procedia Computer Science, Elsevier, vol. 29, pp. 1002–1013, materiały pokonferencyjne z: International Conference on Computational Science, Cairns, Australia, 10-12.06.2014

MNiSW: 10pts (Web of Science)

[D12] **Sieniek M.**, Fast graph transformation based direct solver algorithm for regular three dimensional grids, Procedia Computer Science, Elsevier, vol. 29, pp. 1027–1038, materiały pokonferencyjne z: International Conference on Computational Science, Cairns, Australia, 10-12.06.2014

MNiSW: 10pts (Web of Science)

[D13] Gurgul P., **Sieniek M.**, Magiera K., Skotniczny M., Paszyński M., 2011, Agent-oriented image processing with the hp-adaptive projection-based interpolation method, Procedia Computer Science, Elsevier, vol. 4, pp. 1844-1853, proceedings of: International Conference in Computational Science, Singapur, 01-03.06.2011

MNiSW: 10pts (Web of Science)

[D14] Collier N., **Sieniek M.**, 2012, Agent-based algorithm for spatial distribution of objects, Procedia Computer Science, Elsevier, vol. 9, pp. 1494–1502, proceedings of: International Conference in Computational Science, Omaha, Stany Zjednoczone, 04-06.06.2012

MNiSW: 10pts (Web of Science)

[D15] **Sieniek M.**, Gurgul P., Kołodziejczyk P., Paszyński M., 2010, Agent-based parallel system for numerical computations, Procedia Computer Science, Elsevier, vol. 1, pp. 1971-1981; proceedings of: International Conference in Computational Science, Amsterdam, Holandia, 01-03.05.2010

MNiSW: 10pts (Web of Science)

[D16] Matuszyk P., **Sieniek M.**, Paszyński M., 2015, Fully automatic 2D hp-adaptive Finite Element Method for Non-Stationary Heat Transfer, Procedia Computer Science, Elsevier, proceedings of: International Conference on Computational Science, Reykjavik, Islandia, 1-3.06.2015

MNiSW: 10pts (Web of Science)

[D17] Paszyńska A., Gurgul P., **Sieniek M.**, Paszyński M., 2013, Linear Computational Cost Graph Grammar Based Direct Solver for 3D Adaptive Finite Element Method Simulations, International Journal of Materials, Mechanics and Manufacturing, vol. 1(3), ISSN 1793-8198, IACSIT Press, referat z International Conference on Nano and Materials Engineering, Bangkok, Thailand, 08-09.04.2013

MNiSW: 10pts (Web of Science)

[D18] **Sieniek M.**, Gurgul P., Paszyński M., 2013, Employing adaptive finite elements to model squeezing of a layered material in 3D, vol. 1(4), ISSN 1793-8198, IACSIT Press, an essay from International Conference on Nano and Materials Engineering, Bangkok, Thailand, 08-09.04.2013

MNiSW: 10pts (Web of Science)

5.4 OTHER PUBLICATIONS

[D19] Goik D., **Sieniek M.**, Gurgul P., Paszyński M., 2014, Modeling of the Absorption of the Electromagnetic Wave Energy in the Human Head Induced by Cell Phone, Journal of Applied Mathematics and Physics, vol. 2, pp. 1079-1084

[D20] **Sieniek M.**, Gurgul P., Paszyński M., 2013, Fast multiscale simulations of the Step-and-Flash Imprint Lithography, Asia Pacific Congress on Computational Mechanics, Singapur, 11-14.12.2013

[D21] Gurgul P., **Sieniek M.**, Paszyński M., Madej Ł., 2013, Three-dimensional hp-adaptive algorithm for continuous approximations of material data using space projection, ISSN 1641-8581, vol. 13(2), pp. 245–250, an essay from: XX Conference Computer Methods in Materials Technology, KomPlasTech 2013, Zakopane, Polska, 13-16.01.2013

[D22] **Sieniek M.**, Distributed Agent Platform for Adaptive Modeling in Nanolithography, 2013, Lap Lambert, ISBN 978-3-659-39300-6

[D23] **Sieniek M.**, Zarządzanie reputacją pracodawcy z branży IT na przykładzie Google, 2013, Bezkresy Wiedzy, ISBN 978-3-639-88994-9

[D24] **Sieniek M.**, Gurgul P., Paszyński M., 2010, Application of agent-based approach for multiscale hp-adaptive Finite Element Method, proceedings of: XV Conference Computer Methods in Materials Technology, KomPlasTech 2010, Białka Tatrzańska, Polska 16-19.01.2010

[D25] Gurgul P., **Sieniek M.**, Paszyński M., 2009, Object-oriented Multiscale HP-Adaptive Finite Element Method, materiały pokonferencyjne z: XIV Conference Computer Methods In Material Technology KomPlasTech2009, Krynica Zdrój, Polska 11-14.01.2009

[D26] Paszyński M., Gurgul P., **Sieniek M.**, Wrzeszcz M., Schaefer R., 2008, Object-oriented HP-Adaptive Finite Element Method System for Multiscale Problems, proceedings of: 8th World Congress on Computational Mechanics oraz 5th European Congress on Computational Methods in Applied Sciences and Engineering, Wenecja, Włochy, 30.06-04.07.2008

[D27] **Sieniek M.**, Gurgul P., Paszyński M., 2011, Application of agent-based approach for multiscale hp-adaptive Finite Element Method, materiały proceedings of: International Conference on Complex, Intelligent and Software Intensive Systems, Seul, Korea Południowa; pp. 458-461

The sum of the index Impact Factor (IF) of my achievements of the PhD student equals **6.623,** whereas the total MNiSW score is **263.** Scoring method was prepared in compliance with the list of Ministry of Science and Higher Education from 31st December 2014. The Impact Factor results were taken from journals' websites.

6 REMAINING BIBLIOGRAPHY

Papers that I am referring to but have not contributed to.

[C1] Demkowicz L., 2004, Projection-based interpolation, ICES Report, The University of Texas in Austin, 04-03

[C2] Babuska I., Guo B., 1986, The hp-version of the finite element method, Part I: The basic approximation results, Computational Mechanics, vol. 1 (21-41)

[C3] Reid K. Duff I. S., The multifrontal solution of indefinite sparse symmetric linear systems. ACM Transactions on Mathematical Software.

[C4] Reid K. Duff I. S., The multifrontal solution of unsymmetric sets of linear systems. SIAM Journal on Scientific and Statistical Computing.

[C5] Zhang F., 2005, The Schur Complement and Its Applications; Springer

[C6] Paszyński M., 2006, Modelowanie wieloskalowe w zastosowaniach do nanotechnologii, Hutnik-Wiadomości Hutnicze, 71, pp. 186-196

[C7] Paszyński M., Romkes A., Collister E., Meiring J., Demkowicz L., Willson C. G., 2005, On the Modeling of Step-and-Flash Imprint Lithography using Molecular Statics Models, ICES Report 05-38, University of Texas in Austin

[C8] Baumann P. T., Ben Dhia H., Elkhodja N., Oden J. T., 2008, Application of Arlequin Method to the Coupling of Particle and Continuum Models, Computational Mechanics, vol. 42 pp. 511-530

[C9] Bailey T., Smith B., Choi B. J., Colburn M., Meissl M., Sreenivasan S. V., Ekerdt J. G., Willson C. G., 2001, Step and flash imprint lithography: Defect analysis, Journal of Vacuum Science and Technology (B), vol. 19 (6) pp. 2806-2810

[C10] Demkowicz L., Buffa A., 2005, H1, H(curl), and H(div)-conforming projection-based interpolation in three dimensions Quasi-optimal p-interpolation estimates, Computer Methods in Applied Mechanics and Engineering, 194, 267-296

[C11] Grochowski M., Schaefer R., Smołka M., 2006, Architectural Principles and Scheduling Strategies for Computing; Fundamenta Informaticae, vol. 71; IOS Press, pp. 15–26

[C12] Demkowicz L., 2006, Computing With Hp-adaptive Finite Elements. Vol. 1: One and Two Dimensional Elliptic and Maxwell Problems, Chapman & Hall CRC, Texas

[C13] Bauman P.T., Dhia H. B. , Elkhodja N., Oden J. T., Prudhomme S., On the application of the Arlequin method to the coupling of particle and continuum models, Computational mechanics 42 (4), 511-530

[C14] Demkowicz, L., Kurtz, J., Pardo, D., Paszyński, M., Zdunek, A., 2006, Computing With Hp-adaptive Finite Elements, Part II Frontiers, Three Dimensioanl Elliptic and Maxwell Problems with Applications CRC Press, Taylor and Francis.

[C15] Demkowicz, L., Buffa, A., 2004, H^1, H(curl) and H(div) – conforming projection-based interpolation in three dimensions, ICES-Report 04-24, The University of Texas in Austin.

[C16] Gurgul, P., Sieniek, M., Magiera, K., Skotniczny, M., 2011, Application of multi-agent paradigm to hp-adaptive projection-based interpolation operator, accepted to Journal of Computational Science.

[C17] Gurgul, P., Sieniek, M., Paszyński, M., Madej, Ł., Collier, N., 2012, Two dimensional hp-adaptive algorithm for continuous approximations of material data using space projections; accepted to Computer Science.

[C18] Sieniek, M., Gurgul, P., Kołodziejczyk, P., Paszyński, M., 2010, Agent-based parallel system for numerical computations, Procedia Computer Science, 1, 1971-1981.

[C19] Paszyński, M., Romkes, A., Collister, E., Meiring, J., Demkowicz, L., Wilson, C.G., 2005, One the modeling of Step-and-Flash imprint lithography using molecular static models, ICES-Report 05-38.

[C20] Madej, L., 2010, Development of the modeling strategy for the strain localization simulation based on the Digital Material Representation, DSc dissertation, AGH University Press, Krakow.

[C21] Madej, L., Rauch, L., Perzyński, K., Cybułka P., 2011, Digital Material Representation as an efficient tool for strain inhomogeneities analysis at the micro scale level, Archives of Civil and Mechanical Engineering, 11, 661-679.

[C22] Perzyński, K., Major, Ł., Madej, Ł., Pietrzyk, M., 2011, Analysis of the stress concentration in the nanomultilayer coatings based on digital Representation of the structure, Archives of Metallurgy and Materials, 56, 393-399.

[C23] T. J. R. Hughes, The Finite Element Method. Linear Statics and Dynamics Finite Element Method Analysis, 2000

[C24] M. E. Colburn, Step and Flash Imprint Lithograpy: A Low Pressure, Room Temperature Nonoimprint Lithography. PhD Thesis, 2000

[C25] M. Paszynski, B. Barabasz, R. Schaefer, Lecture Notes in Computer Science, 2007, 4488, 342-349

[C26] Geng P., Oden T. J., van de Geijn R. A. A Parallel Multifrontal Algorithm and Its Implementation, Computer Methods in Applied Mechanics and Engineering, vol. 149, pp.289-301.2006

[C27] Duff I. S., Reid J. K. The multifrontal solution of unsymmetric sets of linear systems, SIAM Journal of Scientific and Statistical Computing, vol. 5, pp.633-641, 1984

[C28] Duff I. S., Reid J. K. The multifrontal solution of indefinite sparse symmetric linear equations, ACM Transations on Mathematical Software, vol. 9, pp. 302-325, 1983

[C29] Paszynska, A., Grabska, E. and Paszynski, M. A graph grammar model of the hp adaptive three dimensional finite element method. part I, Fundamenta Informaticae, vol. 114 (2), pp. 149-182. 2012

[C30] Paszynska, A., Grabska, E. and Paszynski, M. A graph grammar model of the hp adaptive three dimensional finite element method. part II, Fundamenta Informaticae vol. 114 (2), 183-201. 2012

[C31] Paszynska, A., Paszynski, M., Grabska, E. (2008). Graph transformations for modeling hp-adaptive finite element method with triangular elements, Lecture Notes in Computer Science, vol. 5103 pp. 604–613.

[C32] Paszynska, A., Paszynski, M., Grabska, E. (2009). Graph Transformations for Modeling hp-Adaptive Finite Element Method with Mixed Triangular and Rectangular Elements, Lecture Notes in Computer Science, vol. 5545, pp.875-884

[C33] Paszynski, M. On the parallelization of self adaptive hp-finite element methods part I. composite programmable graph grammar model, Fundamenta Informaticae, vol. 93 no. 4, pp. 411–434, 2009

[C34] Paszynski, M. On the parallelization of self-adaptive hp-finite element methods part ii. partitioning communication agglomeration mapping (PCAM) analysis, Fundamenta Informaticae, vol. 93 no. 4, pp. 435–457, 2009

[C35] Strug, B., Paszynska, A., Paszynski, M. Using a graph grammar system in the finite element method, International Journal of Applied Mathematics and Computer Science, vol. 23(4), pp.839-853, 2013

[C36] Paszynski, M., Schaefer R., Graph grammar driven partial differential eqautions solver, Concurrency and Computations: Practise and Experience, vol. 22, no. 9, pp.1063-1097, 2010

[C37] Paszynski, M., Pardo, D., Paszynska, A. Parallel multi-frontal solver for p adaptive finite element modeling of multi-physics computational problems, Journal of Computational Science, vol. 1, no. 1, pp. 48-54. 2010

[C38] Slusarczyk, G., Paszynska, A. Hypergraph Grammars in hp-adaptive Finite Element Method, Procedia Computer Science, vol. 18 pp.1545-1554, 2013

[C39] Szymczak, A., Paszynska, A., Gurgul, P. Graph Grammar Based Direct Solver for hp-adaptive Finite Element Method with Point Singularities, Procedia Computer Science, vol. 18, pp. 1594-1603, 2013

[C40] Paszynski, M., Pardo, D., Calo, V. M. A direct solver with reutilization of LU factorizations for h-adaptive finite element grids with point singularities, Computers & Mathematics with Applications, vol. 65 (8), pp.1140-1151, 2013

[C41] Booch, G., Rumbaugh, J., Jacobson, I., The Unified Modeling Language User Guide, Addison-Wesley Professional, 1st edition, 1998.

[C42] Paszyński M., Pardo D., Paszyńska A. 2010 Parallel Multi-Frontal Solver for Multi-Physics p Adaptive Problems, accepted to International Conference on Computational Science May 2010, Amsterdam, Proceedia Computer Science

[C43] Paszyński M., Pardo D., Torres-Verdín C., Demkowicz L., Calo V. 2010 A Parallel Direct Solver for the Self-Adaptive hp Finite Element Method, Journal of Parallel and Distributed Computing 70 270-281

[C44] Paszynski M. Pardo D., Calo V., 2015, Direct solvers performance on *h* adaptive grids, Computers & Mathematics with Applications, 70(3) 282-295.

7 List of Figures

8 List of Algorithms