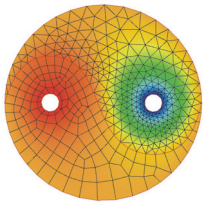
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 9, 2009, No. 2



OBJECT ORIENTED MULTI-SCALE *hp*-ADAPTIVE FINITE ELEMENT METHOD

PIOTR GURGUL, MARCIN SIENIEK, MACIEJ PASZYŃSKI

Department of Computer Science,
AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland
Corresponding Author: paszynsk@agh.edu.pl (M. Paszyński)

Abstract

This paper presents an object-oriented (O-O) project of the hp-adaptive Finite Element Method (hp-FEM) code, supporting multi-scale computations. The main goal of the O-O approach in hp-FEM is to simplify the code and make it reusable and easily extendable. Mesh implemented according to the Euler model makes it easy to switch between dimensions, changing only as much code, as it is really necessary. The hp-adaptation, including mesh refinements and unrefinements, is supported. The O-O approach allows to easily mixture the meso-scale with macro-scale elements, even different discretization methods are utilized. The O-O project has been prepared with Unified Modeling Language (UML). The structure of all classes in JAVA programming language has been automatically generated from the UML diagrams, and the method bodies have been implemented manually. The exemplary multi-scale problem, concerning the interaction of a meso-scale domain representing a polymer network modeled by the molecular statics technique, with macro-scale domain modeled by linear elasticity, has been solved. Different methods for coupling meso- and macro-scale elements have been implemented and tested.

Key words: multi-scale computations, finite element method

1. MOTIVATION

The paper presents the design and implementation of the object-oriented (O-O) *hp*-adaptive Finite Element Method (*hp*-FEM) application dedicated to the multi-scale problems. The mathematical foundations of the project base on the existing self-adaptive *hp*-FEM code developed by [5]. However, the O-O pattern is utilized in the design process to obtain clear, open, reusable, multi-component structure of the adaptive framework. Moreover, the application is predefined to support multi-scale computations, where some parts of the computational domain are modeled by the macro-scale FEM-based formulation, while other parts of the domain are modeled by the meso-scale particle interactions model. The preliminary 1D version of the application is discussed and tested over the challenging multi-scale problem resulting from the simulations of the nano-lithography process [7].

2. PROBLEM FORMULATION

In this section we focus on the formulation of the exemplary multi-scale problem, utilized for verification of the developed adaptive application. The problem is related to the simulation of the Step-and-Flash Imprint Lithography process [7]. The SFIL is a patterning process utilizing photopolymerization to replicate the topography of a template onto a substrate [1,4]. The original three dimensional problem [6] has been reduced to one dimension, as presented in figure 1. The problem consists in computing the displacement field of the polymer inside the tem-

 ISSN 1641-8581

plate, after the photopolymerization process. The displacement of the polymer can be described by the linear elasticity with thermal expansion coefficient, prescribing the volumetric shrinkage of the feature, resulting from the densification of the polymer. Since the problem is formulated in the meso-scale, there are some interactions between the polymer particles and the template. These interactions can be model well by using the molecular statics model [7]. Thus, we consider the multi-scale problem, with molecular statics applied close to the boundary, and the continuous linear elasticity formulation applied in the remaining part of the domain.

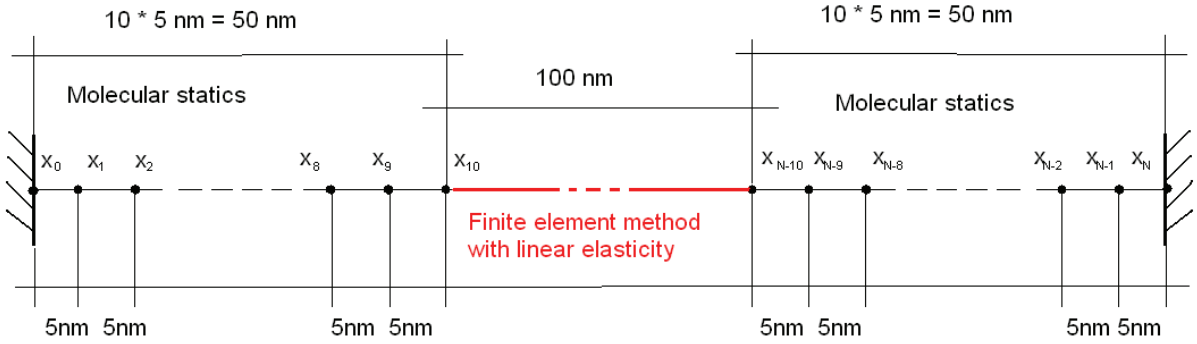


Fig. 1. *Geometry of the computational domain, with molecular statics model utilized close to the template, and the linear elasticity model utilized in the remaining part of the domain.*

In the molecular statics model, the positions of particles are considered, with x_i being the unknown – the position of the i^{th} particle. In the linear elasticity model, the displacement field $R \ni x \to u(x) \in R$ is considered. Both variables can be related on the discrete level:

$$u_i = 5 * i - x_i \tag{1}$$

where u_i is the displacement of i^{th} particle, and 5 is the initial distance between particles.

2.1. Meso-scale model

The interaction between the polymer and the template is prescribed by the molecular statics meso-scale model. The model assumes the quadratic potentials

$$V(r) = k(r - r_{eq})^2 \quad \Rightarrow \quad F(r) = 2k(r - r_{eq}) \tag{2}$$

between each pair of interacting particles, resulting in linear forces. The equillibrium distance between particles $r_{eq} = 5.34$ and the stiffness coefficients $k = 0.1$ are based on the literature [7]. Thus, the force between i^{th} and $i+1^{\text{th}}$ particles is equal to

$$F_{i,i+1} = 2k_{i,i+1}\left(x_{i+1} - x_i - r_{i,i+1}^{eq}\right) \tag{3}$$

where $k_{i,i+1}$ is the stiffness coefficient of the "spring" between i^{th} and $i+1^{\text{th}}$ particles, $r_{i,i+1}^{eq}$ is the equillibrium length of the "spring". Since the problem has been reduced to 1D, we assume that the displacement of the neighboring layers of particles results in additional load force

$$f(x) = cx(x-100)(x-200) \tag{4}$$

with constant c resulting in the maximum enforced horizontal displacement. The second Newton law implies the equillibrium equation to be formulated at each particle

$$F_{i,i+1} - F_{i-1,i} + f(x_i) = 0 \tag{5}$$

which, under the definition of the interparticle forces (3), leads to the following equation

$$x_{i-1}\left(k_{i-1,i}\right) + x_i\left(-k_{i-1,i} - k_{i,i+1}\right) + x_{i+1}\left(k_{i,i+1}\right) =$$
$$f(x_i) + r_{i,i+1}^{eq}k_{i,i+1} - r_{i-1,i}^{eq}k_{i-1,i} \tag{6}$$

Notice, that x_i are unknown equillibrium positions of particles. Moreover, it is assumed that the first and last particles, representing the template particles, are actually fixed:

$$x_0 = 0 \;\; x_N = 200 \tag{7}$$

2.2. Macro-scale formulation – strong form

The displacement of the polymer inside the template can be described the continuous model. We start from the standard one dimensional linear elasticity model [5]

$$-\frac{d}{dx}\left(EA\frac{du}{dx}\right) = f \tag{8}$$

where E represents the Young modulus, A represents the cross-sectional area and f stands for an additional load force. The interface conditions are the following

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

where $F_{9,10}$ and $F_{N-10,N-9}$ are the forces between particles 9 and 10, and N-10 and N-9.

$$EA\frac{du(50)}{dx} = -2k_{9,10}\left(x_{10} - x_9 - r^{eq}_{9,10}\right)$$
$$EA\frac{du(150)}{dx} = 2k_{N-10,N-9}\left(x_{N-9} - x_{N-10} - r^{eq}_{N-10,N-9}\right) \tag{10}$$

Moreover, the position of these particles is identified with the discrete displacement field, with the values u_l, u_r of the first order shape function at borders of the macro-scale FEM area

$$u_l = x_{10} - 50 \quad u_r = x_{N-10} - 150 \tag{11}$$

2.3. Macro-scale formulation – weak form

The strong formulation transforms into the following weak (variational) formulation

$$\int_{50}^{150} EAu'v'\,dx - EAu'(150)v(150) + EAu'(50)v(50) = \int_{50}^{150} f\,v\,dx + \int_{50}^{150} \alpha\,v'\,dx \tag{12}$$

satisfied $\forall v \in V = H^1(50,150)$. We have also added the thermal expansion coefficient term $\int_{50}^{150} \alpha\,v'\,dx = \alpha\left[v(150) - v(50)\right]$ by the analogy with the three dimensional variational formulation [7]. By utilizing the interface conditions, we end up with the following weak equations

$$\int_{50}^{150} EAu'v'\,dx - 2k_{N-10,N-9}\left(x_{N-9} - x_{N-10} - r^{eq}_{N-10,N-9}\right)v(150) - 2k_{9,10}\left(x_{10} - x_9 - r^{eq}_{9,10}\right)v(50) = \int_{50}^{150} f\,v\,dx + \int_{50}^{150} \alpha v'\,dx \tag{13}$$

3. DESIGN OF THE OBJECT-ORIENTED APPLICATION

In the following part of the paper, we focus on the particular aspects of the O-O design of the *hp*-FEM application. The Unified Modeling Language [2] is utilized as a tool for the description of the modeled system.

3.1. 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.

3.2. Mesh – implementation

Mesh is implemented quite straight: there are separate classes for each kind of node, all of them derived from the abstract base class `Node`, as illustrated in figure 2. 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.

3.3. 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,

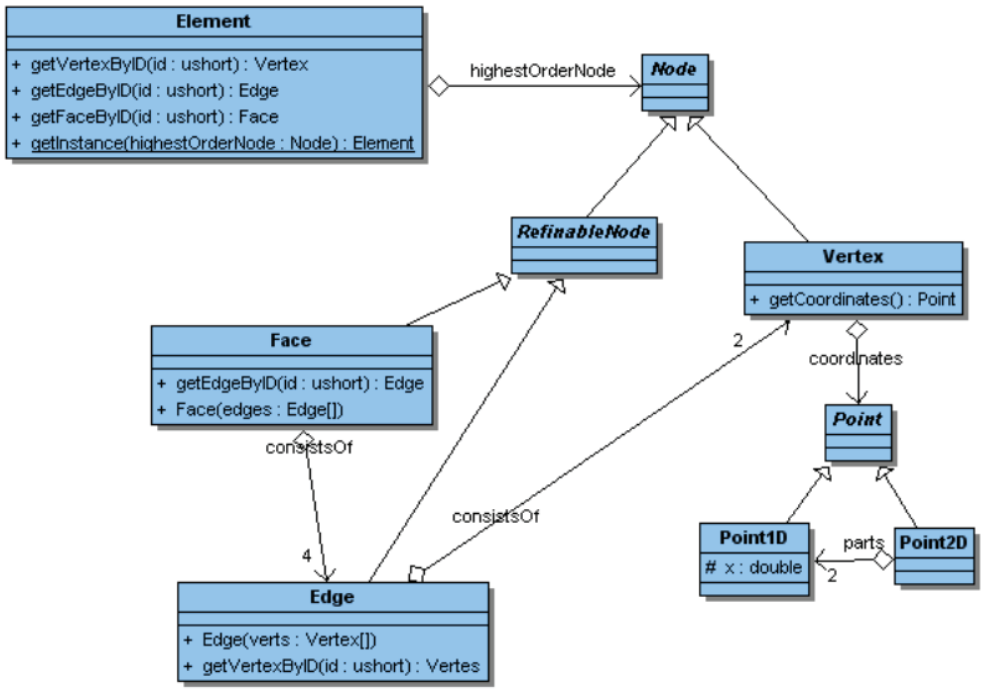


Fig. 2. *Mesh class diagrams.*

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 have associated one or more 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.

3.4. Shape functions – implementation

Our application is dedicated to the special kind of hierarchical shape functions, defined in [5], where subsequent functions are tensor product of the previous ones. The one-dimensional shape functions are defined as

$$\kappa_1(\xi)=1-\xi \quad \kappa_2(\xi)=\xi \quad \kappa_3(\xi)=(1-\xi)\xi$$
$$\kappa_n(\xi)=\kappa_{n-1}(\xi)(\kappa_2(\xi)-\kappa_1(\xi)) \qquad (14)$$

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:

$$\phi_2(\xi_1,\xi_2)=\kappa_1(\xi_1)\kappa_2(\xi_2)=(1-\xi_1)\xi_2 \qquad (15)$$

shape functions of the bottom edge:

$$\phi_{5,j}(\xi_1,\xi_2)=\kappa_{2+j}(\xi_1)\kappa_1(\xi_2), \quad j=1,\ldots,p_1-1 \qquad (16)$$

and bubble shape functions

$$\phi_{9,ij}(\xi_1,\xi_2)=\kappa_{2+i}(\xi_1)\kappa_{2+j}(\xi_2),$$
$$i=1,\ldots,p_h-1, j=1,\ldots,p_v-1 \qquad (17)$$

The full definition for all shape functions can be found in [5]. `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 3. 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.

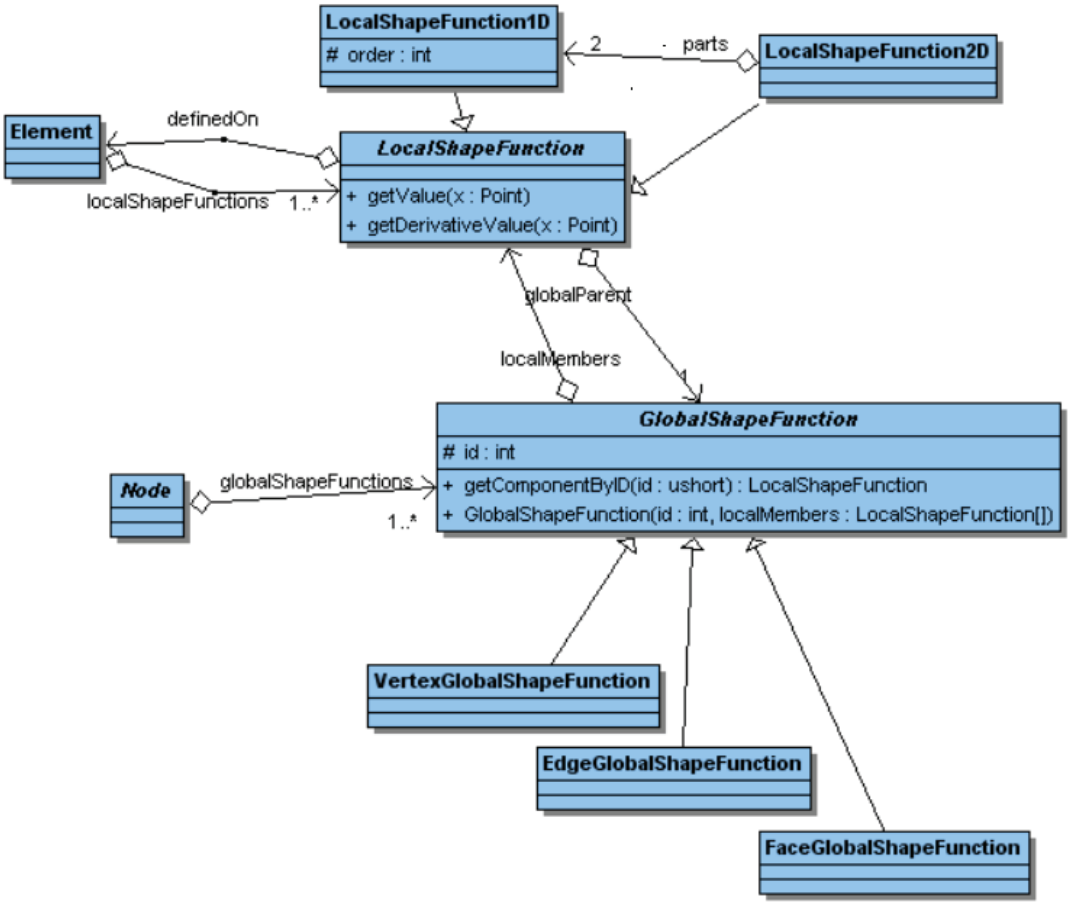


Fig. 3. Shape functions class diagrams.

3.5. 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`.

3.6. 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).

3.7. 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.

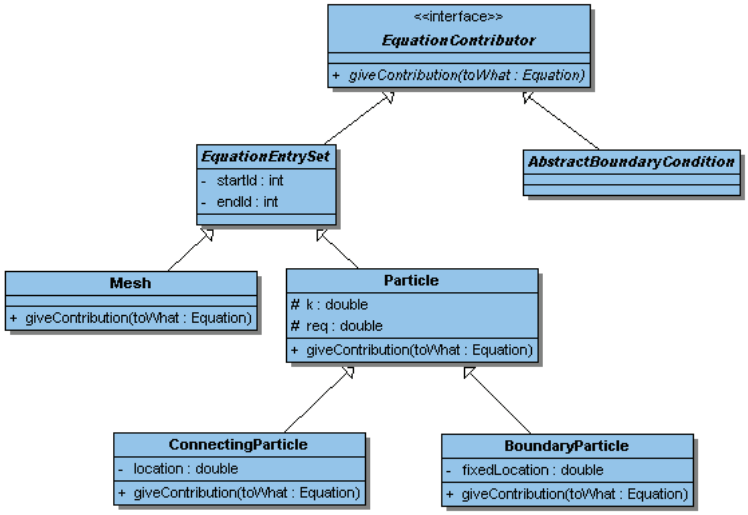


Fig. 4. Dependencies between particle classes of different types.

3.8. 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 [5].

3.9. 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 4. 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:

1. `Particle` located inside the meso-scale part, with the following attributes r_{eq}, k – interparticle interaction coefficients, with unknown location x_i to be computed according to formula (6).
2. `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, defined according to formulae (10-11)
3. `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, as it is expressed in the equation (7)

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.

4. NUMERICAL RESULTS

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 5. 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 5.

5. CONCLUSIONS AND FUTURE WORK

We presented the design and implementation of the O-O *hp*-FEM multi-scale application. The application has been tested on the challenging multi-scale problem, related to the simulations of the nanolithography process. The obtained results clearly illustrate the power of the developed *hp* adaptive framework. The best approximation is obtained when *h* and *p* adaptations are combined together. The crucial task is to identify the parts of the domain, where more precise approximation is required for the high accuracy of the solution. This can be achieved by employing the automatic *hp*-adaptation algorithm [5], to be implemented in the future version of the code. We are also going to extend the application to two and three dimensional problems.

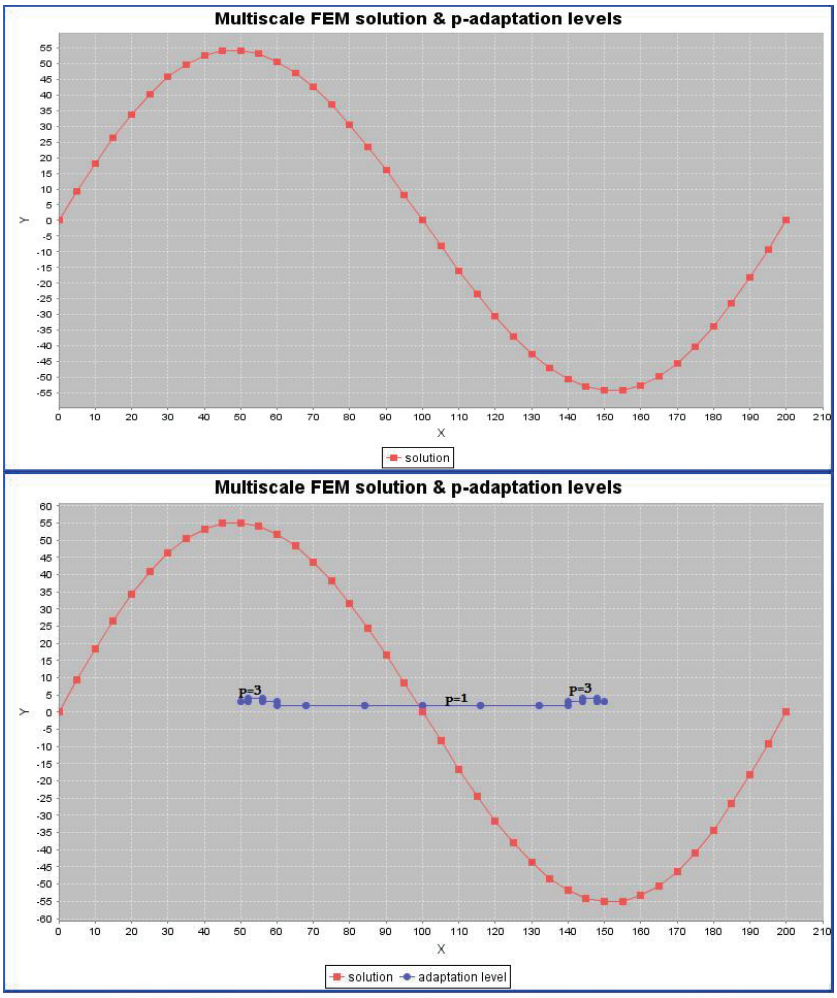


Fig. 5. *The solutions of the multi-scale problem. The red lines denote the solution, the blue lines denote the order of approximation on the FEM domain.* ***First panel:*** *The solution obtained with the meso-scale model defined over the entire domain.* ***Second panel:*** *The solution obtained over the manually hp refined mesh.*

ACKNOWLEDGEMENTS

The work reported in this paper was supported by Polish MEiN grant no. 3 T08B 055 29.

REFERENCES

1. Bailey, T. C., Colburn, M. E., Choi, B. J., Grot, A., Ekerdt, J. G., Sreenivasan, S. V., Willson, C. G., Step and Flash Imprint Lithography: A Low-Pressure, Room Temperature Nanoimprint Patterning Process, Alternative Lithography, Unleashing the Potentials of Nanotechnology, eds, Sotomayor Torres, C., Elsevier, 2002.
2. Booch, G., Rumbaugh, J., Jacobson, I., The Unified Modeling Language User Guide, Addison-Wesley Professional, 1st edition, 1998.
3. Colburn, M. E., Step and Flash Imprint Lithograpy: A Low Pressure, Room Temperature Nonoimprint Lithography, PhD. Thesis, The University of Texas in Austin, 2001.
4. Colburn, M. E., Suez, I., Choi, B. J., Meissi, M., Bailey, T., Sreenivasan, S. V., Ekerdt, J. E., Willson, C. G., Characterization and modeling of volumetric and mechanical properties for SFIL photopolymers, Journal of Vacuum Science and Technology, B 19(6), 2001.
5. Demkowicz, L., Computing with hp-Adaptive Finite Elements, Vol. I. One and Two Dimensional Elliptic and Maxwell Problems, Chapman & Hall / CRC Applied Mathematics & Nonlinear Science, 2006.
6. Paszyński, M., Modelowanie wieloskalowe w zastosowaniach do nanotechnologii, Hutnik-Wiadomości Hutnicze, 71, 2006, 188-196 (in Polish).
7. Paszyński, M., Romkes, A., Collister, E., Meiring, J., Demkowicz, L., Willson, C. G., On the Modeling of Step-and-Flash Imprint Lithography using Molecular Statics Models, ICES Report 05-38, The University of Texas in Austin, 2005.

WIELOSKALOWA *HP* ADAPTACYJNA METODA ELEMENTÓW SKOŃCZONYCH ZORIENTOWANA OBIEKTOWO

Streszczenie

Artykuł prezentuje projekt systemu obiektowego hp adaptacyjnej metody elementów skończonych, przeznaczonego do rozwiązywania problemów wieloskalowych.

Celem opisu aplikacji hp adaptacyjnej za pomocą paradygmatu obiektowego jest uproszczenie kodu źródłowego, umożliwienie wielokrotnego wykorzystywania podobnych funkcjonalnie fragmentów kodu, oraz uczynienie kodu łatwo rozszerzalnym. Element siatki obliczeniowej zaprojektowany został zgodnie z ideą modelu Eulera, w którym obiekty wyższego wymiaru przedstawione są jako kompozycja obiektów niższego wymiaru. Umożliwia to łatwe rozszerzanie modelu na wyższe wymiary, z wykorzystaniem wyników uzyskanych na elementach składowych niższego wymiaru. Projekt opisuje algorytm hp adaptacji, wraz z procedurą usuwania zbędnych stopni swobody. Podejście obiektowe pozwala na łatwe łączenie elementów nanoskalowych z elementami makroskalowymi, pomimo wykorzystania różnych metod dyskretyzacji. Projekt obiektowy przygotowany został w języku UML (Unified Modeling Language). Diagramy klas w języku UML umożliwiły automatyczną generacje struktury klas w języku JAVA. Proponowany projekt aplikacji adaptacyjnej przetestowany został na modelowym problemie wieloskalowym, dotyczącym symulacji procesu nanolitografii poprzez wyciskanie i naświetlanie. W artykule przedstawione są również różne metody łączenia modeli makro i nano skalowych.

Submitted: October 8, 2008
Submitted in a revised form: November 7, 2008
Accepted: November 20, 2008