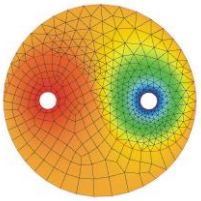
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE
Informatyka w Technologii Materiałów

Vol. 19, 2019, No. 1



PROJECTION-BASED INTERPOLATION ALGORITHM OVER TWO-DIMENSIONAL GRIDS WITH TRIANGULAR ELEMENTS

JAKUB RYZNER[1], ANNA PASZYŃSKA[2*]

[1]Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakow, Poland
[2]Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, ul. St. Łojasiewicza 11, 30-348 Krakow, Poland
**Corresponding author: anna.paszynska@uj.edu.pl*

Abstract

In this paper, we employ the projection-based interpolation algorithm for approximation of two-dimensional bitmaps resulting from photographs of multi-phase material samples. The algorithm uses h-adaptive two-dimensional finite element grid with triangular elements. It results in a continuous approximation of material data with Lagrange polynomial, which makes them more suitable for finite element method computations.

Key words: Projection-based interpolation, Material science simulations, Continuous material data representation

1. INTRODUCTION

The projection-based interpolation was invented by (Demkowicz, 2004), to implement efficient hp-adaptive finite element method computations, as described in (Demkowicz, 2006a). The mathematical properties of the method, including the convergence properties, are analyzed in (Demkowicz & Buffa 2004) and (Demkowicz, 2006b). The method was used in (Gurgul et al., 2013) for continuous approximation of material data using two-dimensional h-adaptive finite element method with rectangular elements. The agent-based parallel implementation was also proposed in (Sieniek et al., 2011). The method was extended to three-dimensional grids with hexahedral finite elements in (Goik et al., 2013).

We claim that the triangular meshes are more popular in engineering simulations, and thus we want to show that the PBI method works well with these non-regular grids as well.

In this paper, we propose the version of the method using two-dimensional triangular finite elements. For a non-regular structure of the material data bitmaps, that do not conform with horizontal and vertical lines, we expect to obtain better approximation properties of our method.

We assume that we want to compute the displacements, and thus the solution of the entire problem is needed. In the case when we are computing the strength, it is possible to apply the Saint-Venant principle, and the algorithm described in this paper may be not applicable.

It is possible to extend the code with *hp* adaptivity, but this is technically more complicated. In general *h* adaptation works well while recovering the large gradients, and *p* adaptation work well when we deal with smooth solutions. We compromise by using *h* adaptation with $p = 2$. We do not think that p-adaptation is suitable here since we are not approximating a smooth solution rather we are trying to adapt to jumps in material data.

 ISSN 1641-8581

2. PROJECTION-BASED INTERPOLATION WITH H-ADAPTIVE FINITE ELEMENT METHOD

We employ the following adaptive algorithm (**algorithm 1**) executed on two-dimensional initial grid constructed with triangular finite elements. The input to the algorithm is a uniform grid as well as the bitmap representing the material data.

How is the problem "solved" over the computational mesh? The projection solver performs the operations in **algorithm 2**.

```
function hadaptive_PBI(initial_mesh,desired_err,coef,BITMAP)
1 mesh = initial_mesh
2 repeat
3   u = solve on mesh using PBI (u=PBI_solver(mesh, BITMAP))
4   max_err = 0
5   for each element K of coarse mesh do
6     K_err = compute relative error on K
7           using the mesh solution and BITMAP data
8     if K_err > max_err then
9       max_err = K_err
10      end if
11    end do
12    adapted_mesh = new empty_mesh
13    for each element K of mesh do
14      if K_err > coef * max_err then
15        break element K
16      else
17        add K from mesh to adapted_mesh
18      end if
19    end do
20    mesh = adapted_mesh
21   until max_err < desired_err
22 return (u, mesh)
```

Algorithm 1. *h-adaptive finite element method on two-dimensional grid with triangular elements.*

```
function PBI_solver (mesh, BITMAP)
1 loop through all mesh vertices v_i
2   //compute the PBI coefficient at vertex v_i
3   a_{v_i} = BITMAP(v_i)
4 end loop
5 loop through all mesh edge nodes e_i
6   //compute the PBI coefficient at edge e_i
7   a_{e_i} = ∫_{e_i} U(x,y) φ_{e_i}(x,y) dxdy / ∫_{e_i} φ_{e_i}(x,y) φ_{e_i}(x,y) dxdy
8 end loop
9 loop through all mesh interior nodes I
10  //compute the PBI coefficient at interior I
11  a_I = ∫_I W(x,y) φ_I(x,y) dxdy / ∫_I φ_I(x,y) φ_I(x,y) dxdy
12 end loop
13 return (u)
```

Algorithm 2. *Projection-based interpolation solver called from line 3 of* ***Algorithm 1****.*

The quadratic polynomials are used, thus we have one degree of freedom per vertex, one degree of freedom per edge, and one degree of freedom per element interior. From the point of view of a single triangular element, the algorithm computes the coefficients at all three vertices of the triangular element just by reading them from the bitmap:

$$a_{v_i} = BITMAP(v_i) \forall_{i=1,...,3} \tag{1}$$

Next, it minimizes the error in the *L*2 norm over three edges of the triangle The values read from *BITMAP*(v_i) corresponds to pixels as presented in figure 1. They are within [0,1] where the values close to 0 represent one phase of the material, and the values close to 1 represent the second phase of the material. They are mapped into the Young modulus in a way that one phase maps to 15 GPa and another phase maps to 208 GPa:

$$\left\| \overbrace{\left(BITMAT - \sum_{j=1}^{3} u_{V_j} \right)}^{U} - u_{e_i} \right\|_{L2(e_i)} \rightarrow \min \forall_{i=1,...,3} \tag{2}$$

We want to minimize distance from the space span by polynomial $u_{e_i} = a_i e_i$ and the remaining approximant *U*. This is attained by considering *U* - u_{e_i} orthogonal to the space span by u_{e_i}, which means that the scalar product in L2 norm is 0 there, which literally means:

$$a_{e_i} = \frac{\int_{e_i} U(x,y)\varphi_{e_i}(\mathbf{x},\mathbf{y})\,dxdy}{\int_{e_i} \varphi_{e_i}(\mathbf{x},\mathbf{y})\varphi_{e_i}(\mathbf{x},\mathbf{y})\,dxdy} \tag{3}$$

since the only basis function in the space span by polynomial u_{e_i} is e_i.

Next, it minimizes the error in the L2 norm over the interior of the triangle:

$$\left\| \overbrace{\left(U - \sum_{j=1}^{3} u_{V_j} - \sum_{j=1}^{3} u_{0,e_j} \right)}^{W} - u_{\mathbf{I}} \right\|_{L2(f_i)} \rightarrow \min \tag{4}$$

We want to minimize the distance from the space span by polynomial $u_{\mathbf{I}} = a_I e_I$ and the remaining approximant *W*. This is attained by considering $W - u_{e_I}$ orthogonal to the space span by u_{e_I}, which means that the scalar product in L2 norm is 0 there, which means:

$$a_{\mathrm{I}} = \frac{\int_{\mathrm{I}} W(\mathbf{x},\mathbf{y})\varphi_{\mathrm{I}}(\mathbf{x},\mathbf{y})\,dxdy}{\int_{\mathrm{I}} \varphi_{\mathrm{I}}(\mathbf{x},\mathbf{y})\varphi_{\mathrm{I}}(\mathbf{x},\mathbf{y})\,dxdy} \tag{5}$$

since the only basis function in the space span by polynomial u_{e_I} is e_I.

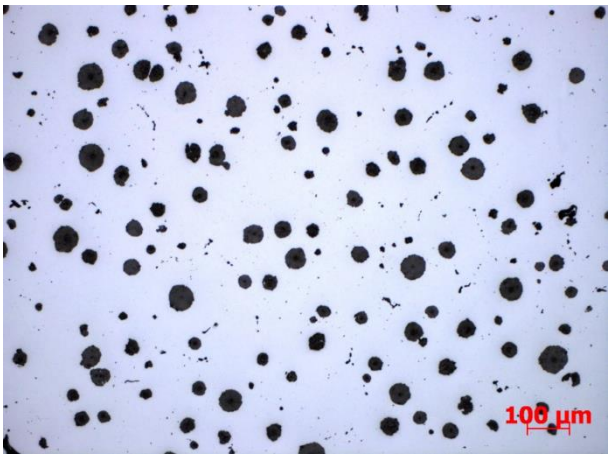


Fig. 1. *The original bitmap with photo of the two-phase material.*

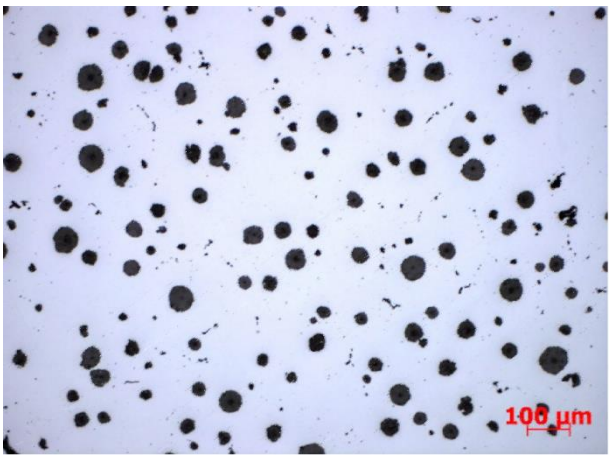


Fig. 2. *The continuous approximation of the bitmap.*

3. NUMERICAL RESULTS

We conclude the paper with numerical results presenting the generation of the continuous approximation of the material data representing the two-phase material.

The size of the input mesh is arbitrary, the smaller the better, but we need to be able to catch the material irregularities at the Gauss integration points, so either initial mesh is large enough, or

while computing the numerical error over an element we can perform the adaptive integration to catch all the irregularities of the bitmap.

We have performed now the linear elasticity simulations of the compressed material. We consider the two-phase material with graphite inclusions under the compressing force.

The properties of the graphite are E = 15 GPa Young modulus and ν = 0.3 Poisson ratio. The properties of the background material are E = 208 GPa Young modulus and ν = 0.3 Poisson ratio, which represent the steel.

We refer to Appendix A for the derivation of the linear elasticity equations. We solve the linear elasticity problem over the continuous approximation of material data using the modified hp2d code (Demkowicz, 2006a). The modifications are listed in Appendix B.

We put the compressing force at the top and the bottom of the domain, and the free boundary conditions on left and right sides, see figure 4.

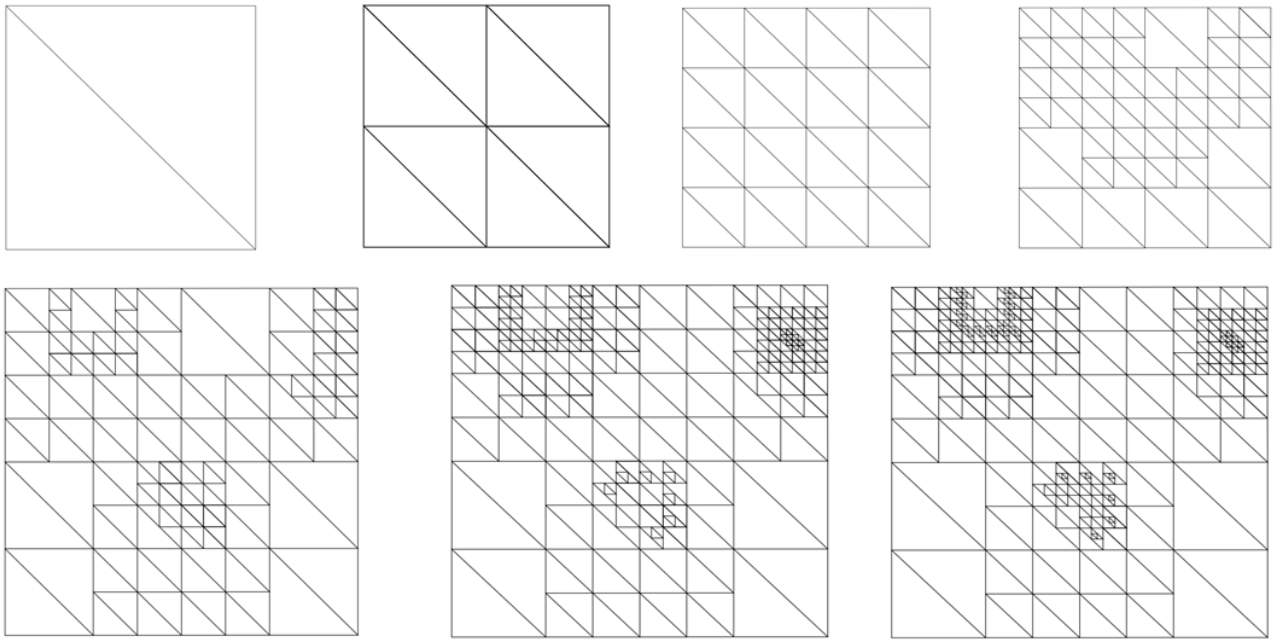


Fig. 3. *The fragment of the computational mesh with adaptivities resulting from application of algorithm 1.*

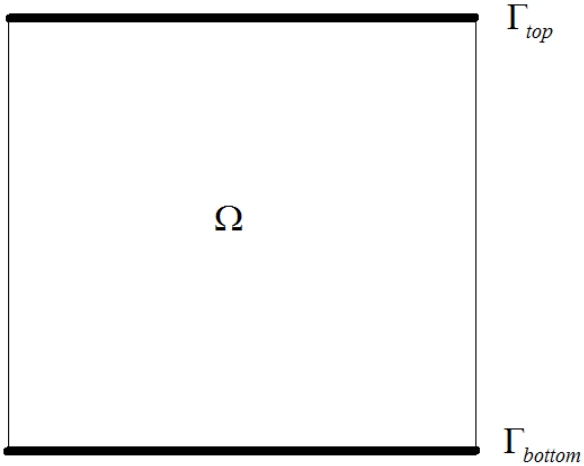


Fig. 4. *The continuous approximation of the bitmap.*

In our formulation we use displacements, and we model the compression by the following trick. We fix the displacements at the top and at the bottom of the domain, and we define the displacements as (0,-1) at the top of the boundary, and (0,1) at the bottom of the boundary. Thus, we do not model the force itself, but rather the displacement of the material on the boundary of the domain, following from the compression force. From mathematical point of view, the formulation is stable and well-posed. The problem is elliptic, and on the part of the boundary we have Dirichlet boundary conditions.

We utilize the compressing force:

$$g_i : \Gamma_{D_i} \ni x \to g_i(x) = \begin{cases} (0,-1) & for \quad x \in \Gamma_{top} \\ (0,0) & for \quad x \in \Omega \setminus \left(\Gamma_{top} \cup \Gamma_{bottom} \right) \\ (0,1) & for \quad x \in \Gamma_{bottom} \end{cases} \tag{6}$$

and zero initial stress $\sigma_{ij}^0 = 0$.

The elliptic computational problem, such as the linear elasticity, where some part of the boundary included the Dirichlet boundary conditions for all the components of the solution derive a unique solution. Here we fix the displacements on the top and bottom of the domain, as expressed by the formula for function gi from (7) the Dirichlet b.c. as described in Appendix. In other words, the function gi is the Dirichlet boundary condition.

We processed the piece of the material as presented in figure 5. Young modulus map in the material is presented in figure 6. The resulting displacement fields are presented in figures 7 and 8.

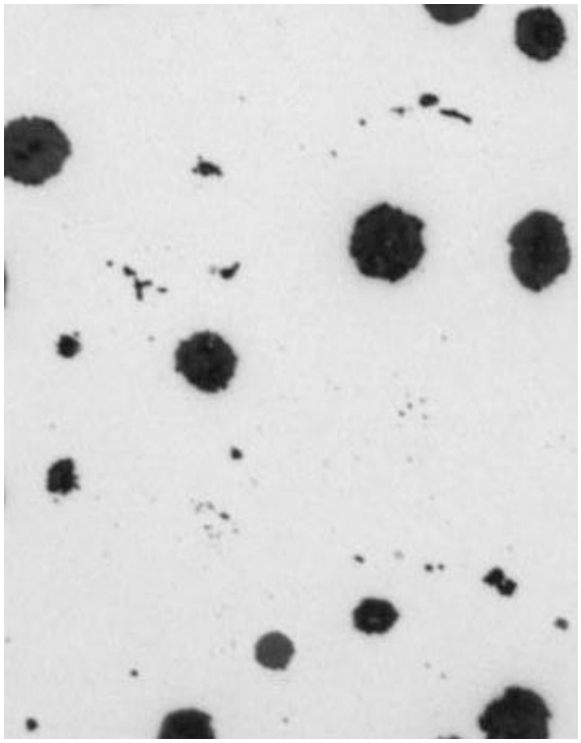


Fig. 5. *A Piece of the bitmap used for the compression simulations.*

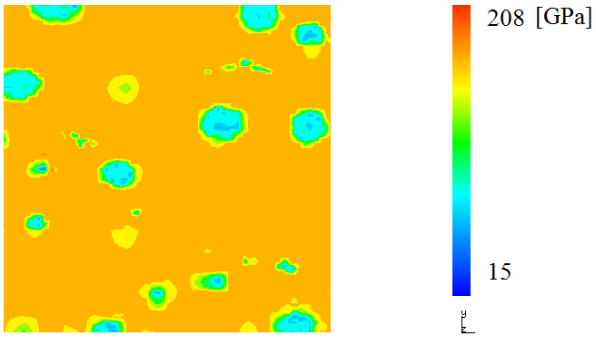


Fig. 6. *Young modulus map in the material.*

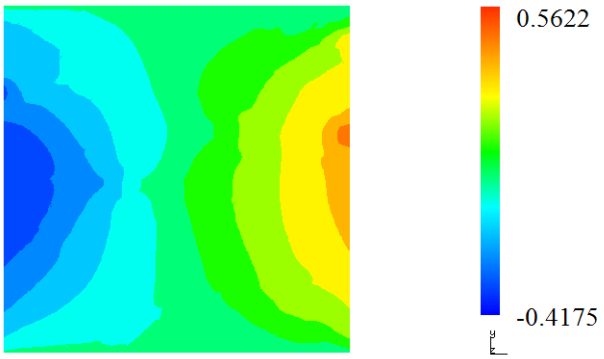


Fig. 7. *Displacement in the x direction.*

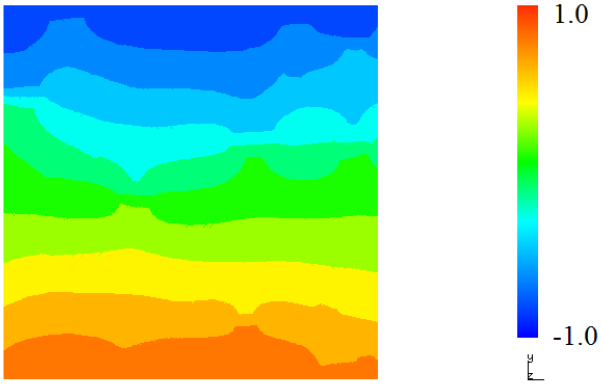


Fig. 8. *Displacement in the y direction.*

4. CONCLUSIONS

The adaptive bitmap compressions with projection-based interpolation algorithm on triangular grids results in the continuous approximation of material data suitable for finite element method computations. We have shown in this paper that the material science data can be successfully approximated with PBI using triangular mesh and subject to finite element method linear elasticity simulations. The continuous approximation of material data removes singularities on the border of materials.

When there are jumps in material coefficients, it is possible to apply the finite element method without continuous approximation of material data obtained by computing some projections, e.g., projection-based approximation as proposed in this paper. However, to converge to the high accuracy of the numerical solution, we need to perform several adaptations of the computational mesh, each time solving a computational problem (linear elasticity in our case). The computational cost of solution of a computational problem in two dimensions is $O((kN)^{3/2})$ where k is the number of equations ($k = 2$ in case of linear elasticity), and N is the number of basis functions over the mesh. In three-dimensions (not considered in this paper) this cost grows higher to $O((kN)^2)$. In other words, the computational cost of

creating high accuracy adaptive grid based on adaptive solving of a linear elasticity problem requires a sequence of O((*kN*)*3/2*) cost solutions. On the other hand, the projection based algorithm employed in this paper generates a smooth approximation of material data, by employing a similar adaptive process, with the linear computations cost O(*N*). We can even employ the adapted computational mesh generated by projection based interpolation algorithm in a linear computational cost and solve the linear elasticity problem with the original distribution of coefficients, resulting in a similar accuracy than the one obtained from the more costly standard adaptive procedure.

APPENDIX A

We start with classical linear elasticity with u_i displacement vector, $u_{i,j}$ displacement gradients, $u_{(i,j)}$ symmetric part of the displacement gradients $u_{(i,j)} = (u_{i,j} + u_{j,i})/2$ and $u_{[i,j]}$ skew-symmetric part of the displacement gradients $u_{[i,j]} = (u_{i,j} - u_{j,i})/2$.

We have the property that $u_{i,j} = u_{(i,j)} + u_{[i,j]}$.

We also introduce f_i body force per unit volume.

We introduce ε_{ij} strain tensor, defined to be the symmetric part of the displacement gradients $\varepsilon_{ij} = (u_{i,j} - u_{j,i})/2$, as well as σ_{ij} stress tensor, defined in terms of the generalized Hooke's law: $\sigma_{ij} = c_{ijkl}\varepsilon_{kl}$, with c_{ijkl} elastic coefficients (known for a given material).

We derive first the strong form of the boundary-value problem:

Given f_i: $\Omega \to R$, g_i: $\Gamma_{D_i} \to R$, h_i: $\Gamma_{N_i} \to R$, find u_i: $\bar{\Omega} \to R$ such that:

$$\sigma_{ij,j} + f_i = 0 \ \text{ in } \ \Omega \tag{7}$$

$$u_i = g_i \ \text{ in } \ \Gamma_{D_i} \tag{8}$$

$$\sigma_{ij} n_j = h_i \ \text{ in } \ \Gamma_{N_i} \tag{9}$$

where: $\sigma_{ij} = c_{ijkl}\varepsilon_{kl}$.

The weak form is derived in the following way:

Multiply $\sigma_{ij,j} = 0$ by $w_i \in V^i$ and integrate over Ω to get:

$$\int_\Omega w_i \sigma_{ij,j} d\Omega = 0 \tag{10}$$

We integrate by parts:

$$-\int_\Omega w_{i,j}\sigma_{ij}\, d\Omega + \int_\Gamma w_i \sigma_{ij} n_j\, d\Omega = 0 \tag{11}$$

We have the property $w_{i,j}\sigma_{ij} = w_{(i,j)}\sigma_{ij}$ since σ_{ij} is symmetric tensor, thus:

$$-\int_\Omega w_{(i,j)}\sigma_{ij}\, d\Omega + \int_\Gamma w_i \sigma_{ij} n_j\, d\Omega = 0 \tag{12}$$

We get the weak form of the boundary-value problem:

Given f_i: $\Omega \to R$, g_i: $\Gamma_{D_i} \to R$, h_i: $\Gamma_{N_i} \to R$, find $u_i \in V_i$ such that:

$$\int_\Omega w_{(i,j)}\sigma_{ij}\, d\Omega = \int_\Omega w_i f_i\, d\Omega + \sum_{i=1}^{nd}\left(\int_{\Gamma_{N_i}} w_i h_i\, d\Gamma_{N_i}\right) \quad \text{for all } w_i \in V_i \tag{13}$$

We use $\sigma_{ij} = c_{ijkl}\varepsilon_{kl}$ to get:

$$\int_\Omega w_{(i,j)} c_{ijkl}\varepsilon_{kl}\, d\Omega = \int_\Omega w_i f_i\, d\Omega + \sum_{i=1}^{nd}\left(\int_{\Gamma_{N_i}} w_i h_i\, d\Gamma_{N_i}\right) \tag{14}$$

We also use $\varepsilon_{ij} = u_{(i,j)}$ to get:

$$\int_\Omega w_{(i,j)} c_{ijkl} u_{(k,l)}\, d\Omega = \int_\Omega w_i f_i\, d\Omega + \sum_{i=1}^{nd}\left(\int_{\Gamma_{N_i}} w_i h_i\, d\Gamma_{N_i}\right) \tag{15}$$

Following Hughes (2000) we introduce the following abstract notation:

Given **f**, **g**, **h** find $\mathbf{u} \in \mathbf{V}$, such that:

$$\mathbf{a}(\mathbf{w},\mathbf{u}) = (\mathbf{w},\mathbf{f}) + (\mathbf{w},\mathbf{h})_\Gamma \ \text{ for all } \mathbf{w} \in \mathbf{V} \tag{16}$$

$$\mathbf{a}(\mathbf{w},\mathbf{u}) = \int_\Omega w_{(i,j)} c_{ijkl} u_{(k,l)}\, d\Omega \tag{17}$$

$$(\mathbf{w},\mathbf{f}) = \int_\Omega w_i f_i\, d\Omega \tag{18}$$

$$(\mathbf{w},\mathbf{h})_\Gamma = \sum_{i=1}^{nd}\left(\int_{\Gamma_{N_i}} w_i h_i d\Gamma_{N_i}\right) \tag{19}$$

For the implementation purposes we notice that:

$$\mathbf{a}(\mathbf{w},\mathbf{u}) = \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w})^{\mathrm{T}}\, \mathbf{D}\boldsymbol{\varepsilon}(\mathbf{u})\, d\Omega \tag{20}$$

since $w_{(i,j)} c_{ijkl} u_{(i,j)} = \boldsymbol{\varepsilon}(\mathbf{w})^{\mathrm{T}}\mathbf{D}\boldsymbol{\varepsilon}(\mathbf{u})$ and we have defined:

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

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

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

where

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}$$

and E - the Young modulus, ν - the Poisson ratio.

APPENDIX B

In this appendix we describe the modificiations of the hp2d linear elasticity code. The Dirichlet and Naumann boundary conditions have been implemented in the following routines:

Getg – uniform Neumann b.c.
Getmat – material data, in particular Young modulus and Poisson ratio
Getf – the compressing force

Source code of the moodification of the hp2d linear elasticity code.

```
c-------------------------------------------------------------c
c    routine name       - getmat
c
c-------------------------------------------------------------c
c    computer           - machine independent
c
c    latest revision    - Jan 06
c
c    purpose            - routine specifies coefficients of the
c                         differential equation to be solved
c                         (material data)
c
c    arguments :
c      in:
c                Mdle    - element (middle node) number
c                Xi      - master element coordinates of a point
c                Xp      - physical coordinates of the point
c      out:
c                ZAcoef,Zbcoef,ZCcoef - operator coefficients
c
c    required  routines -
c
c-------------------------------------------------------------c
      subroutine getmat(Mdle,Xi,Xp, ZAcoef,Zbcoef,ZCcoef)
c
      use data_structure2D
c
#include "syscom.blk"
```


```
c
      dimension Xi(2),Xp(NDIMEN)
      dimension ZAcoef(1:MAXEQNS,1:MAXEQNS,2,2),
     .          Zbcoef(1:MAXEQNS,1:MAXEQNS,2),
     .          ZCcoef(1:MAXEQNS,1:MAXEQNS)
c
c  ...elasticities
      dimension  ee(2,2,2,2)
c
      double precision Xval,Xder(2)
c
      ZAcoef(1:MAXEQNS,1:MAXEQNS,1:2,1:2) = ZERO
      ZBcoef(1:MAXEQNS,1:MAXEQNS,1:2) = ZERO
      ZCcoef(1:MAXEQNS,1:MAXEQNS) = ZERO
c
      EYOUNG = 1.d0; RNI = 0.3d0
      call get_bitmap(Xp,Xval,Xder)
      if(Xval.lt.0.5d0)then
        EYOUNG=INV_E1*1.d9 !graphite
      else
        EYOUNG=INV_E2*1.d9 !background
      endif
c
      g  = EYOUNG/2.d0/(1.d0+RNI)
      rl = EYOUNG*RNI/(1.d0+RNI)/(1.d0-2.d0*RNI)
      aux = 2.d0*g+rl
c
c  ...define the tensor of elasticities, see Section 15.2
c     in the book
c
c  ...E_ijkl = g(d_ik d_jl + d_il d_jk) + rl d_ij d_kl
      ee(1:2,1:2,1:2,1:2) = 0.d0
      ee(1,1,1,1) = aux; ee(2,2,2,2) = aux
      ee(1,1,2,2) = rl;  ee(2,2,1,1) = rl
      ee(1,2,1,2) = g;   ee(1,2,2,1) = g
      ee(2,1,1,2) = g;   ee(2,1,2,1) = g
c
c  ...translate into the format for the PDE's coeeficients
      do ivar2=1,2
      do ivar1=1,2
        do k=1,2
        do l=1,2
          ZAcoef(ivar2,ivar1,k,l) = ee(ivar2,k,ivar1,l)
        enddo
        enddo
```


```
      enddo
      enddo
c
c      write(*,*)'getmat_elas: ZAcoef',ZAcoef
c
      return
      end
c-------------------------------------------------------------c
c   routine name       - getf
c
c-------------------------------------------------------------c
c   computer           - machine independent
c
c   latest revision    - Jan 06
c
c   purpose            - routine defines a rhs for the linear
elasticity
c                        problem
c
c   arguments :
c     in:
c               Xp      - physical coordinates of a point
c     out:
c               Zfval    - value of the rhs
c
c   required  routines -
c
c-------------------------------------------------------------c
      subroutine getf(Mdle,Xi,Xp, Zfval)
c
      use data_structure2D
      use control
c
#include "syscom.blk"
c
      dimension Xi(2),Xp(NDIMEN),Zfval(MAXEQNS)
      dimension zacoef(1:MAXEQNS,1:MAXEQNS,2,2),
     .          zbcoef(1:MAXEQNS,1:MAXEQNS,2),
     .          zccoef(1:MAXEQNS,1:MAXEQNS)
      dimension zval(MAXEQNS),zdval1(MAXEQNS,2),zdval2(MAXEQNS,2,2)
      save nflag_getf
      data nflag_getf /0/
      Zfval=0.d0
      end
c-------------------------------------------------------------c
```


```
c   routine name       - dirichlet
c
c-------------------------------------------------------------c
c   computer           - machine independent
c
c   latest revision    - Jan 06
c
c   purpose            - routine returns values and the first
c                        order derivatives for the Dirichlet data
c                        for the elastic plate example
c
c   arguments :
c     in:
c               Xp       - physical coordinates of a point
c     out:
c               Zval     - values of the Dirichlet data
c               Zdval1   - values of the first order derivatives
c
c-------------------------------------------------------------c
      subroutine dirichlet(Xp, Zval,Zdval1)
c
      use data_structure2D
      use control
c
#include "syscom.blk"
c
      dimension Xp(NDIMEN),Zval(MAXEQNS),Zdval1(MAXEQNS,2)
c
c  ...second order derivatives of the exact solution
      dimension zdval2(MAXEQNS,2,2)
      save nflag_dirichlet
      data nflag_dirichlet/0/
c
      Zval = ZERO
      Zdval1 = ZERO
c
        if(IEQFLAG.eq.1)then
          if (abs(Xp(2)-1.d0).lt.1.d-6) then
            Zval(1) = 0.d0; Zval(2) = -1.d0
          elseif (abs(Xp(2)).lt.1.d-6) then
            Zval(1) = 0.d0; Zval(2) =  1.d0
          endif
        endif
      return
      end
```


ACKNOWLEDGMENTS

We would like to acknowledge Maciej Paszyński for help in setup of the hp2d code interface.

REFERENCES

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

Demkowicz, L. Buffa, A., 2004, H1, H(curl) and H(div) – conforming projection-based interpolation in three dimensions, *ICES-Report 04-24*, The University of Texas in Austin.

Demkowicz, L., 2006a, *Computing With Hp-adaptive Finite Elements*, Chapman & Hall CRC Press.

Demkowicz, L., 2006b, Polynomial exact sequences and projection – based interpolation with application to Maxwell equations, *ICES-Report 06-12*, The University of Texas in Austin.

Goik, D., Sieniek, M., Paszynski, M., Madej, Ł., 2013, Employing an Adaptive Projection-based Interpolation to Prepare Discontinuous 3D Material Data for Finite Element Analysis, *Procedia Computer Science*, 18, 1535-1544.

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*, 4(3), 164-169.

Hughes, T.J.R., 2000, *The finite element method: Linear statics and dynamics finite element method analysis*, Dover Publications.

Sieniek, M., Gurgul, P., Skotniczny, M., Magiera, K., Paszynski, M., 2011, Agent-oriented image processing with the hp-adaptive projection-based interpolation operator, *Procedia Computer Science*, 4, 1844-1853.

ALGORYTM INTERPOLACJI BAZUJĄCEJ NA PROJEKCJI DLA DWU-WYMIAROWYCH SIATEK OBLICZENIOWYCH Z ELEMENTAMI TRÓJKĄTNYMI

Streszczenie

Artykuł przedstawia zastosowanie algorytmu interpolacji bazującej na projekcji do aproksymacji dwuwymiarowych bitmap reprezentujących zdjęcia struktury wielo-fazowych materiałów. Algorytm ten używa *h*-adaptacyjnej dwuwymiarowej metody elementów skończonych z elementami trójkątnymi. W wyniku jego zastosowania dostajemy ciągłą aproksymację materiału z wykorzystaniem wielomianów Lagrange'a rozpiętych na siatce obliczeniowej, na której można następnie przeprowadzić obliczenia metodą elementów skończonych.

Received: December 2, 2018.
Received in a revised form: April 23, 2019.
Accepted: June 14, 2019.