

# A two-dimensional linear elasticity problem for anisotropic materials, solved with a parallelization code

Mihai-Victor PRICOP\*, Cornelia NIȚĂ\*

\*Corresponding author

\*INCAS – National Institute for Aerospace Research “Elie Carafoli”

Bdul Iuliu Maniu 220, Bucharest 061136, Romania

vpricop@incas.ro, nitac@incas.ro

DOI: 10.13111/2066-8201.2010.2.3.8

**Abstract:** The present paper introduces a numerical approach of static linear elasticity equations for anisotropic materials. The domain and boundary conditions are simple, to enhance an easy implementation of the finite difference scheme. SOR and gradient are used to solve the resulting linear system. The simplicity of the geometry is also useful for MPI parallelization of the code.

**Key Words:** elasticity, finite difference.

## 1. INTRODUCTION

The problem discussed in this paper does not have an immediate application. We try only to experiment the numerical methods and their algorithm, in order to obtain the numerical optimization of composite structures. The statement of the two-dimensional, anisotropic and static elasticity on a simple geometry is a good starting point for an immediate experimentation of various algorithms and parallel computing.

## 2. THE TWO-DIMENSIONAL ELASTICITY EQUATIONS

The equations of equilibrium, general constitutive law and deformation equations are:

$$\begin{cases} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} = 0 \\ \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} = 0 \end{cases} \quad (1)$$

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \quad (2)$$

$$\begin{aligned} \varepsilon_x &= \frac{\partial u}{\partial x}, \varepsilon_y = \frac{\partial v}{\partial y}, \\ \gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{aligned} \quad (3)$$

For an isotropic material, the matrix  $\mathbf{e}$  is:

$$\mathbf{e} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \quad (4)$$

Replacing the deformations in the equations of equilibrium two partial derivative equations can be obtained, where the unknowns are the displacements.

With the following notation:

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix} \quad (5)$$

The equations in displacements are:

$$\begin{cases} e_1 \frac{\partial^2 u}{\partial x^2} + e_2 \frac{\partial^2 v}{\partial x \partial y} + e_3 \left( \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2} \right) + e_7 \frac{\partial^2 u}{\partial x \partial y} + e_8 \frac{\partial^2 v}{\partial y^2} + e_9 \left( \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} \right) = 0 \\ e_4 \frac{\partial^2 u}{\partial x \partial y} + e_5 \frac{\partial^2 v}{\partial y^2} + e_6 \left( \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x \partial y} \right) + e_7 \frac{\partial^2 u}{\partial x^2} + e_8 \frac{\partial^2 v}{\partial x \partial y} + e_9 \left( \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2} \right) = 0 \end{cases} \quad (6)$$

The domain is a rectangle, corresponding to an infinite thickness plate, with clamped edges and with imposed displacements on upper side for an easier implementation.

The grid is  $[1, n_{x_0}] \times [1, n_y]$ .

The imposed Dirichlet boundary conditions are:

1. on the left side,  $x=1, y \in [1, n_y]$  where  $u=0; v=0$ ;
2. on the right side,  $x=n_{x_0}, y \in [1, n_y]$  where  $u=0; v=0$ ;
3. on the upper side,  $x \in [1, n_{x_0}], y=n_{x_0}$  where  $u=0; v=-0.2n_yx(1-x)$ ;
4. on the lower side,  $x \in [1, n_{x_0}], y=1$  where  $u=0; v=0$ ;

The stress boundary conditions are stated as follows:

We exemplify on the left side, where imposed pressure (linear distribution load in 2D or on surface in 3D) gives:

$$\sigma_x = e_1 \varepsilon_x + e_2 \varepsilon_y + e_3 \gamma_{xy} = e_1 \frac{\partial u}{\partial x} + e_2 \frac{\partial v}{\partial y} + e_3 \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \quad (7)$$

We are looking for finite difference with a second order accurate approximation. If the first three points  $i = \{1, 2, 3\}$ , with unitary spacing, are considered the support is  $x = \{0, 1, 2\}$ .

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ -\frac{3}{2} & 2 & -\frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \rightarrow \begin{pmatrix} u_1 \\ 2 \cdot u_2 - \frac{3 \cdot u_1}{2} - \frac{u_3}{2} \\ \frac{u_1}{2} - u_2 + \frac{u_3}{2} \end{pmatrix}$$

Fig. 1 VanderMonde matrix of linear system from which results the interpolation polynomial

Fig. 2 The inversion and multiplication with nodal values give the polynomial coefficients. We consider:

$$u_i = u_1, u_2 = u_{i+1,j}, u_3 = u_{i+2,j}.$$

Thus the first order derivative of the interpolation polynomial in the point  $x = 0$  gives:

$$\frac{\partial u}{\partial x} = -1.5u_i + 2u_{i+1,j} - 0.5u_{i+2,j} \quad (8)$$

Therefore, finite difference equation for the boundary condition is:

$$e_1(-1.5u_{i,j} + 2u_{i+1,j} - 0.5u_{i+2,j}) + e_2 \frac{v_{i,j+1} - v_{i,j-1}}{2} + e_3 \frac{u_{i,j+1} - u_{i,j-1}}{2} + e_3(-1.5v_{i,j} + 2v_{i+1,j} - 0.5v_{i+2,j}) = \sigma_x$$

Similarly we can apply stress boundary condition where we have a linear combination of strain, including those which are not parallel with the system axis.

For the interior points of grid we use a centered finite difference scheme of second-order accuracy and for boundary points we use a non-centered finite difference scheme of the same order of accuracy.

Thus, considering unitary step of grid we use the formulas:

$$\left\{ \begin{array}{l} \frac{\partial^2 u}{\partial x^2} \approx u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \\ \frac{\partial^2 u}{\partial y^2} \approx u_{i,j+1} - 2u_{i,j} + u_{i,j-1} \\ \frac{\partial^2 u}{\partial x \partial y} \approx u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1} \end{array} \right. \quad (9)$$

The following finite difference equation is obtained:

$$\left\{ \begin{array}{l} u_{i,j} = \frac{1}{2(e_1 + e_9)}(e_1(u_{i+1,j} + u_{i-1,j}) + (e_2 + e_9)v_{xy} + (e_3 + e_7)u_{xy} + e_5v_{xx} + e_8v_{yy} + e_9(u_{i,j+1} + u_{i,j-1})) \\ v_{i,j} = \frac{1}{2(e_5 + e_9)}(e_9(v_{i+1,j} + v_{i-1,j}) + (e_4 + e_9)u_{xy} + (e_6 + e_8)v_{xy} + e_6u_{yy} + e_7u_{xx} + e_5(v_{i,j+1} + v_{i,j-1})) \end{array} \right.$$

Where:

$$\begin{aligned}
 v_{xy} &= \frac{v_{i+1,j+1} - v_{i-1,j+1} - v_{i+1,j-1} + v_{i-1,j-1}}{4} \\
 u_{xy} &= \frac{u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1}}{4} \\
 v_{xx} &= v_{i+1,j} - 2v_{i,j} + v_{i-1,j} \\
 v_{yy} &= v_{i,j+1} - 2v_{i,j} + v_{i,j-1} \\
 u_{xx} &= u_{i+1,j} - 2u_{i,j} + u_{i-1,j} \\
 u_{yy} &= u_{i,j+1} - 2u_{i,j} + u_{i,j-1}
 \end{aligned} \tag{10}$$

### 3. POST PROCESSING

The results are obtained for any boundary conditions in terms of displacement. For the post processing the stress and the strain are needed in the entire domain. The computation is very simple for the inner nodes  $i = 2 : n_{x0} - 1$   $j = 2 : n_y - 1$ .

Because of the computational inefficiency, Jacobi and simple gradient methods were implemented serial only. Conjugated gradient methods will be approach in the future.

### 4. RESULTS

#### *Serial SOR Gauss-Seidel*

This method is an order of magnitude more efficient than Jacobi in terms of iterations number or computational time.

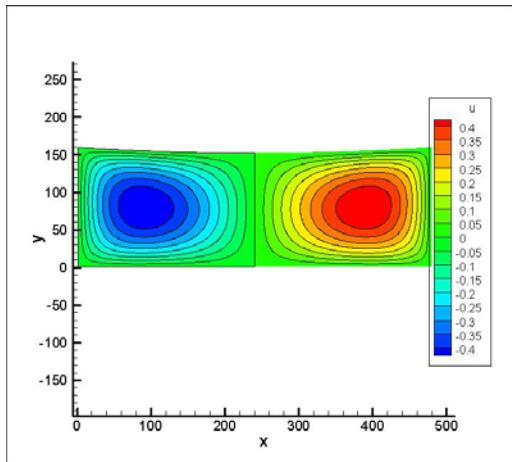


Fig. 3 Horizontal displacement  $u$

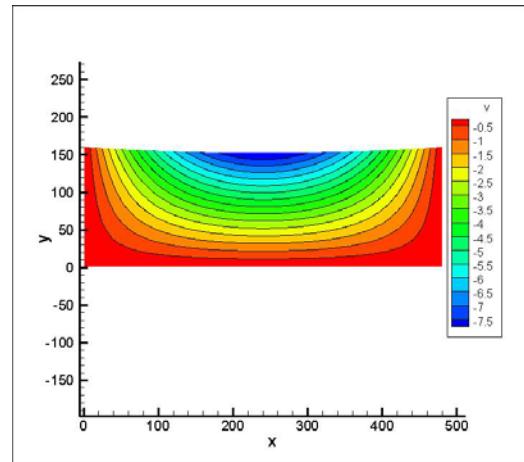
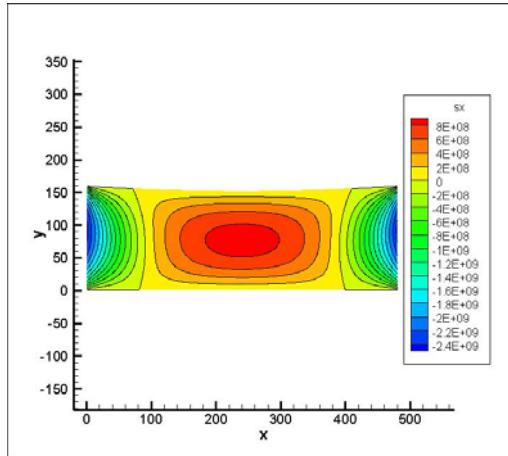
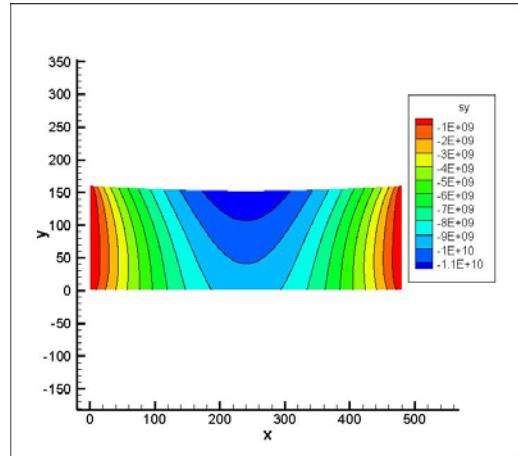
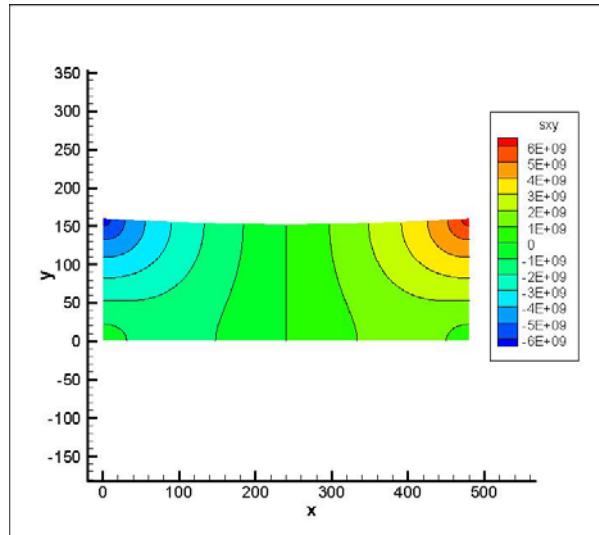


Fig. 4 Vertical displacement  $v$

Fig. 5 Strain  $\sigma_x$ Fig. 6 Strain  $\sigma_y$ Fig. 7 Strain  $\sigma_{xy}$ 

### Parallel SOR Gauss-Seidel

The parallelization for computers with distributed memory architecture, calls the one-dimensional partition.

The displacements matrices are divided in blocks horizontally aligned. Any computing node array (displacements) is one column extended in both directions, in order to store neighbors values used as boundary conditions.

The parallel computing was made on a two quad core workstation. Open MPI and gfortran public domain compiler were used.

## Serial SOR Gauss-Seidel

## Serial Jacobi

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	5575	$10^{-8}$	18.84
240 x 80	16600	$1.5 \times 10^{-7}$	235.28

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	475	$10^{-8}$	1.77
240 x 80	1719	$10^{-8}$	25.46
360x 120	3573	$10^{-8}$	120.28
480x 160	5982	$10^{-8}$	363.64

## Parallel SOR Gauss-Seidel 2 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	378	$10^{-8}$	0.6
240 x 80	1477	$10^{-8}$	7.38
360x 120	3229	$10^{-8}$	38.06

## Parallel SOR Gauss-Seidel 3 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	389	$10^{-8}$	0.53
240 x 80	1487	$10^{-8}$	5.27
360x 120	3228	$10^{-8}$	25.52

## Parallel SOR Gauss-Seidel 4 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	395	$10^{-8}$	0.46
240 x 80	1493	$10^{-8}$	4.34
360x 120	3247	$10^{-8}$	20.03

## Parallel SOR Gauss-Seidel 5 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	401	$10^{-8}$	0.25
240 x 80	1512	$10^{-8}$	3.10
360x 120	3268	$10^{-8}$	16.77

## Parallel SOR Gauss-Seidel 6 CPU

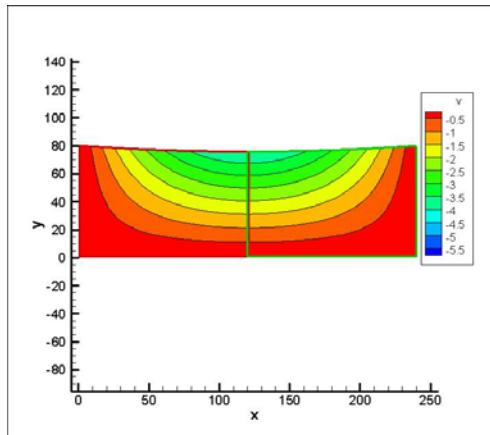
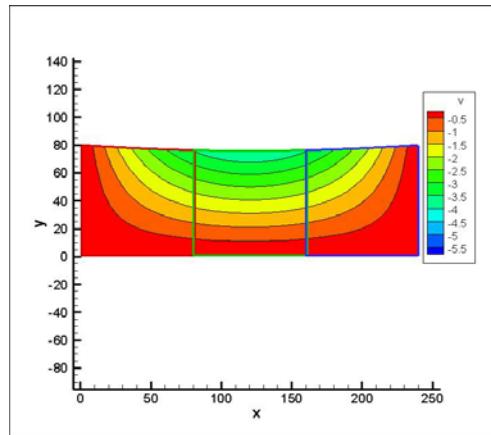
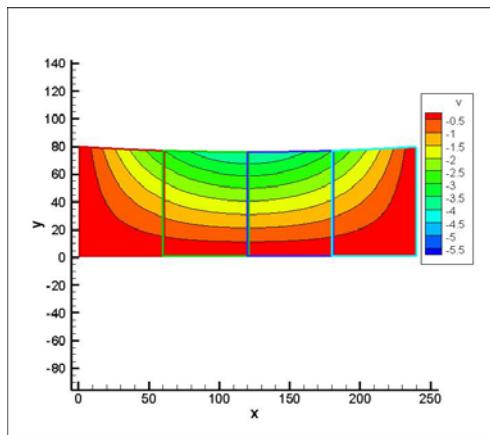
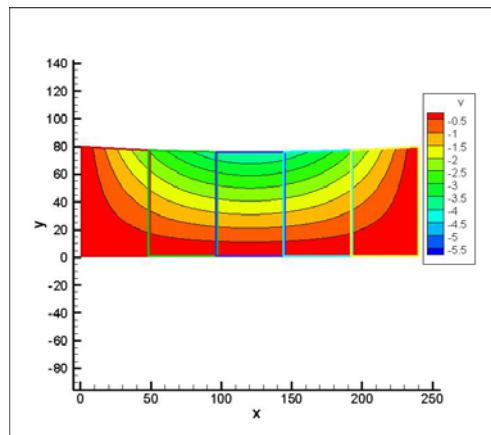
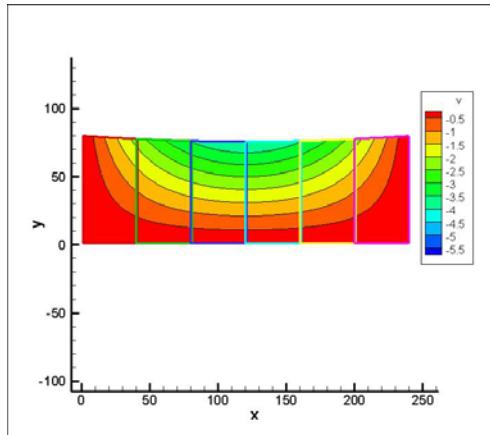
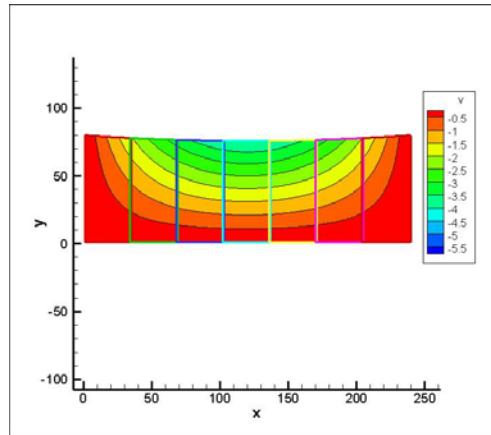
$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	411	$10^{-8}$	0.36
240 x 80	1523	$10^{-8}$	2.93
360x120	3283	$10^{-8}$	12.99

## Parallel SOR Gauss-Seidel 7 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	419	$10^{-8}$	0.34
240 x 80	1541	$10^{-8}$	2.16
360x 120	3309	$10^{-8}$	11.66

## Parallel SOR Gauss-Seidel 8 CPU

$n_{x0} \times n_y$	iterations	accuracy	Time[s]
120 x 40	426	$10^{-8}$	0.36
240 x 80	1555	$10^{-8}$	2.42
360x 120	3326	$10^{-8}$	6.24

Fig. 8 Vertical displacement  $v$  - 2 CPUFig. 9 Vertical displacement  $v$  - 3 CPUFig. 10 Vertical displacement  $v$  - 4 CPUFig. 11 Vertical displacement  $v$  - 5 CPUFig. 12 Vertical displacement  $v$  - 6 CPUFig. 13 Vertical displacement  $v$  - 7 CPU

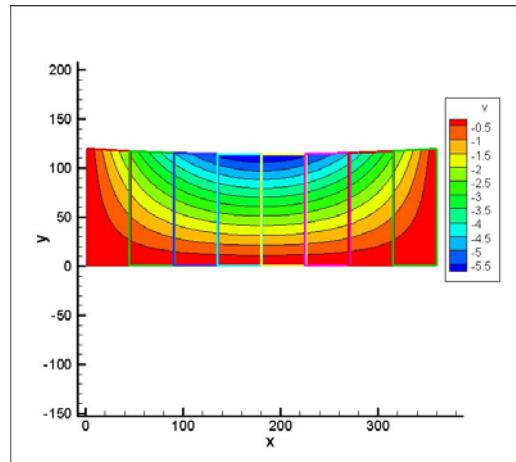
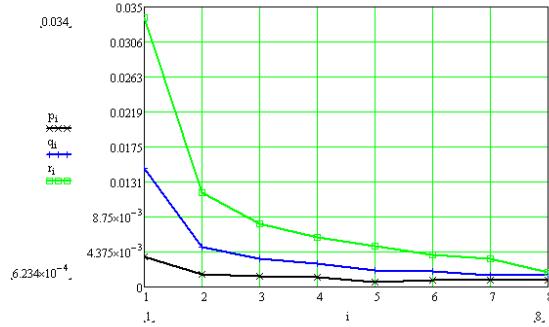
Fig. 14 Vertical displacement  $V$  - 8 CPU

Fig. 15 Computational time for one iteration, for the 3 discretization types: p(120x40), q(240x80), r(360x120), according to the processor number.

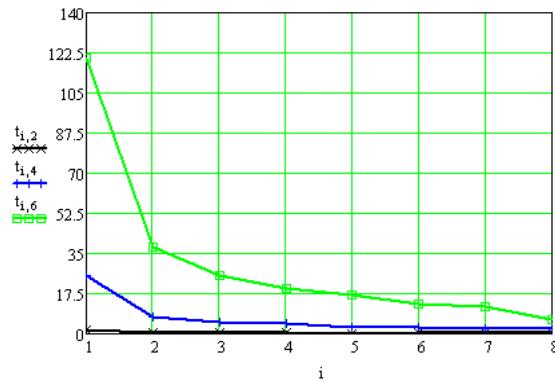


Fig. 16 Effective computational time for the 3 discretization types according to the processors number.

## 5. CONCLUSIONS

Parallel computing is a useful tool for solving large problems. The reduction in memory due to partition of array not only allows for the solution of much larger systems, but faster run times.

Future work will focus on utilization of finite difference scheme of fourth order accuracy and also parallel form of conjugate gradient algorithm.

It is also interesting to implement this code for the three dimensional elasticity problems for both isotropic and anisotropic materials.

## REFERENCES

- [1] A.Neculai, *Convergenta algoritmilor de optimizare*, Editura Tehnica, 2004.
- [2] M. Snir, S. Otto, S. Huss-Lederman, J. Dongara, *MPI: The Complete Reference*, the MIT Press, Cambridge, Massachusetts, 1996.
- [3] C. Berbente, S. Mitran, S. Zancu, *Metode numerice*, Editura Tehnica, 1998.