Description of the Finite Element Program for Contact Problem

Kestřánek, Zdeněk
2001

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Description of the finite element program for contact problem

Zdeněk Kestřánek

Technical report No. 831

February 2001

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: 66053911 fax: (+4202) 8585789
e-mail: zdenda@cs.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Description of the finite element program for contact problem

Zdeněk Kestřánek [1]

Technical report No. 831
February 2001

Abstract

A description of the software package for analyzing contact problem in elasticity is presented.

Keywords
Finite Element Method, Mathematical programming

[1]Supported by Grant KIT COPERNICUS CP977006 - HIPERGEOS II of the European Comission and Grant of Ministry of Education, Youth and Sport of the Czech Republic OK-407(1999)

1 Introduction

We present here a finite element program with the working name FEC (Finite Element Contact) for solving the contact problem in elasticity. We use the contact problem formulation based on the Signorini type conditions and on the Coulomb model of friction. We allow the models with different material properties, however, for each defined subregion they have to be constant. The boundary conditions are assumed to be linear along each boundary segment. These things, together with the discretized geometry (i.e. triangulated region - list of nodes and elements), are defined in the input file for the FEC (USROT.TXT or binary version USROT.DAT). In another file, TEM.DAT, the values of temperature for each node are defined. Both theoretical background and the actual usage of the program is described in the report. The output files VALUESx.DAT, x=1,..,4, and VALPST.DAT contain the values of displacements, stresses and principal stresses. For the numerical results, we refer to [35, 36, 38, 39, 40, 41, 42, 43, 45, 46, 62, 63].

2 Formulation of the problem

2.1 Variational formulation of the problem

Let S bodies in the system occupy region $\Omega = \Omega^1 \cup \ldots \cup \Omega^S$. Let the boundary $\partial\Omega$ be divided into disjunct parts $\Gamma_u, \Gamma_\tau, \Gamma_c, \Gamma_0$, $\partial\Omega = \Gamma_u \cup \Gamma_\tau \cup \Gamma_c \cup \Gamma_0$. $\Gamma_c = \bigcup_{k,l} \Gamma_c^{kl}$ and Γ_0 represent unilateral and bilateral contact boundaries, respectively. Let $\mathbf{n}$ be outward normal to $\partial\Omega^k$ on Γ_c^{kl}. We define the sets

$$V \equiv \{\mathbf{v} \in H^1(\Omega) |\, \mathbf{v} = 0 \text{ on } \Gamma_u,\ v_n = 0 \text{ on } \Gamma_0\},$$

$$K \equiv \{\mathbf{v} \in V |\, v_n^k - v_n^l \leq 0 \text{ on } \Gamma_c^{kl}\}.$$

Let the potential energy functional be of the following form

$$\mathcal{L}(\mathbf{v}) \;=\; \mathcal{L}_0(\mathbf{v}) + j_0(\mathbf{v}) \quad , \tag{2.1}$$

where

$$\mathcal{L}_0(\mathbf{v}) \;=\; \frac{1}{2} A(\mathbf{v},\mathbf{v}) - L(\mathbf{v}) \quad , \tag{2.2}$$

$$A(\mathbf{u},\mathbf{v}) = \int_\Omega c_{ijkm} e_{ij}(\mathbf{u}) e_{km}(\mathbf{v}) d\mathbf{x} \quad , \tag{2.3}$$

$$L(\mathbf{v}) = \int_\Omega F_i v_i d\mathbf{x} + \int_{\Gamma_\tau} P_i v_i d\mathbf{s} + \int_\Omega \beta_{ij} e_{ij}(\mathbf{v}) \Theta d\mathbf{x} \quad , \tag{2.4}$$

$$j_0(\mathbf{v}) = \int_{\Gamma_c^{kl}} g^{kl} |\mathbf{v}_t^k - \mathbf{v}_t^l| d\mathbf{s} \quad . \tag{2.5}$$

F_i represents body loads, P_i surface tension, Θ temperature increment from an undeformed state and g_c^{kl} are prescribed friction forces. The coefficients c_{ijkl}, β_{ij} can be expressed by the constants E, σ, α, known from the thermoelasticity theory. The solution of our problem is the minimum of $\mathcal{L}(\mathbf{v})$ on the set K.

2.2 Finite element approximation

The problem defined in previous section cannot be solved exactly in general. Therefore it is replaced by a sequence of problems for which we can find a solution. We construct a finite dimensional aproximation of V and K. Let us consider a conform tetrahedral mesh corresponding to regions Ω^ι, $1 \le \iota \le S$, $\mathcal{T}_h$, with nodes a_i. Let Ω^ι be polyhedral and let h be the length of the longest side of tetrahedron in the mesh. It holds $\Gamma_c^{kl} = \bigcup_{j=1}^{J} \Gamma_{cj}^{kl}$, and $\Gamma_0 = \bigcup_{j=1}^{J'} \Gamma_{0j}$, where Γ_{cj}^{kl} and Γ_{0j} are faces of Γ_c^{kl} and of Γ_0, $J = J(k,l)$, respectively. Similarly $\Gamma_c^{kl} = \bigcup_{m=1}^{M} \Gamma_{cm}^{kl}$ and $\Gamma_0 = \bigcup_{m=1}^{M'} \Gamma_{0m}$, where Γ_{cm}^{kl} and Γ_{0m} are the tetrahedral faces after the mesh generation. Let us define on Γ_c^{kl} and Γ_0 for every node $a_i \in \Gamma_c^{kl} \cup \Gamma_0$

$$\mathcal{N}_i^{kl} \equiv \{j \in \{1,\dots,J\} |\, a_i \in \Gamma_{cj}^{kl}\}, \tag{2.6}$$

$$\mathcal{N}_i \equiv \{j \in \{1,\dots,J'\} |\, a_i \in \Gamma_{0j}\}. \tag{2.7}$$

Let $\mathbf{n}_j^k$ be the outward normal to Ω^k on Γ_{cj}^{kl}.

$$\begin{aligned} V_h \equiv\ & \{\mathbf{v}_h \in [C(\Omega^1)]^3 \times \dots \times [C(\Omega^S)]^3 |\, \mathbf{v}|_T \in [P_1(T)]^3 \quad \forall T \in \mathcal{T}_h; \\ & \mathbf{v}_h(a_i)\mathbf{n}_j = 0,\ j \in \mathcal{N}_i,\ a_i \in \Gamma_0; \\ & \mathbf{v}_h(a_i) = 0,\ a_i \in \Gamma_u\}, \end{aligned} \tag{2.8}$$

$$\begin{aligned} K_h \equiv\ & \{\mathbf{v}_h \in V_h |\, (v_{hn}^k + v_{hn}^l)(a_i) \equiv (\mathbf{v}_h^k - \mathbf{v}_h^l)(a_i)\mathbf{n}_j^k \le 0, \\ & j \in \mathcal{N}_i^{kl},\ a_i \in \Gamma_c^{kl},\ 1 \le k < l \le S\}\,. \end{aligned} \tag{2.9}$$

Let us define the approximation of $\mathcal{L}$

$$\mathcal{L}_h(\mathbf{v}_h) = \mathcal{L}_0(\mathbf{v}_h) + j_h(\mathbf{v}_h), \tag{2.10}$$

where $j_h(\mathbf{v}_h) : V_h \to R_+^1$ is convex, weakly lower semicontinuous functional.
DEFINITION 2.1 A function $\mathbf{u}_h \in K_h\, {}^1V_h$ is the approximate solution of the contact problem iff

$$\mathcal{L}_h(\mathbf{u}_h) \le \mathcal{L}_h(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in K_h\,. \tag{2.11}$$

Let

$$\Lambda \equiv \{\vec{\mu} \in [L^2(\Gamma_c^{kl})]^3 |\, |\vec{\mu}| \le 1 \text{ a.e. on } \Gamma_c^{kl}\}\,. \tag{2.12}$$

From the Schwartz's inequality for the product of vectors $\vec{\mu}$ and $g_c^{kl}\mathbf{v}_t$, it follows

$$\sup_{\vec{\mu}\in\Lambda} \int_{\cup\Gamma_c^{kl}} [\vec{\mu} g_c^{kl}(\mathbf{v}_t^k - \mathbf{v}_t^l)] d\mathbf{s} \le \int_{\cup\Gamma_c^{kl}} [g_c^{kl} |\mathbf{v}_t^k - \mathbf{v}_t^l|] d\mathbf{s}\,. \tag{2.13}$$

As the inverse inequality is obvious, it holds

$$j_0(\mathbf{v}) = \sup_{\vec{\mu}\in\Lambda} \int_{\cup\Gamma_c^{kl}} [\vec{\mu} g_c^{kl}(\mathbf{v}_t^k - \mathbf{v}_t^l)] d\mathbf{s}\,. \tag{2.14}$$

Let

$$\Lambda_h = \{\vec{\mu}_h \in \Lambda | \quad \vec{\mu}_h|_{\Gamma^{kl}_{cm}} \in [P_0(\Gamma^{kl}_{cm})]^3\}\,. \tag{2.15}$$

Then the natural approximation of j_0 is

$$j_h(\mathbf{v}) = \sup_{\vec{\mu}_h \in \Lambda_h} \sum_{k,l,m} \vec{\mu}_{hm} \int_{\Gamma^{kl}_{cm}} [g_c^{kl}(\mathbf{v}_t^k - \mathbf{v}_t^l)] d\mathbf{s} \equiv \sup_{\vec{\mu}_h \in \Lambda_h} j_{hh}(\mathbf{v}, \vec{\mu}_h)\,. \tag{2.16}$$

Moreover,

$$j_h(\mathbf{v}) = \sum_{k,l,m} | \int_{\Gamma^{kl}_{cm}} [g_c^{kl}(\mathbf{v}_t^k - \mathbf{v}_t^l)] d\mathbf{s} |\,. \tag{2.17}$$

Let us define the Lagrangian

$$\mathcal{H}(\mathbf{v}_h, \vec{\mu}_h) = \mathcal{L}_0(\mathbf{v}_h) + j_{hh}(\mathbf{v}_h, \vec{\mu}_h), \qquad \mathbf{v}_h \in K_h,\ \vec{\mu}_h \in \Lambda_h\,. \tag{2.18}$$

The problem (2.11) can now be formulated as the saddle point one.
Find $(\mathbf{u}_h, \vec{\lambda}_h) \in K_h \times \Lambda_h$ such that

$$\mathcal{H}(\mathbf{u}_h, \vec{\mu}_h) \le \mathcal{H}(\mathbf{u}_h, \vec{\lambda}_h) \le \mathcal{H}(\mathbf{v}_h, \vec{\lambda}_h) \qquad \forall (\mathbf{v}_h, \vec{\mu}_h) \in K_h \times \Lambda_h\,. \tag{2.19}$$

3 Assembling of the system

3.1 The input/output data structure

The data is stored in two global arrays, one integer and one real. Various data types which correspond to one of these arrays are distinguished by different addresses to their first element.

Geometrical model properties are stored especially in integer arrays $ITNODE$, $IBNDRY$, $OLDNODE$, $OLDBLST$, $ICCB$, ICP and in real arrays VX, VY, VZ, TR, CPX, CPY, CPZ.

In the $ITNODE(J,I)$ array, $1 \le I \le NT$, where NT is the number of elements in the mesh, the global nodal indices, $1 \le J \le 4$, of the tetrahedron I are stored. The material index is stored for $J = 5$. Similarly, in $IBNDRY(J,I)$, $1 \le I \le NB$, where NB is the number of boundary faces, the global nodal indices, $1 \le J \le 3$, of the face I are stored, for $J = 4$ the boundary type is stored (Γ_u, Γ_0, Γ_c, Γ_τ with $\mathbf{P} \not\equiv 0$, Γ_τ with $\mathbf{P} \equiv 0$), and for $J = 5$ the number of original boundary face on which the face I after the triangulation lies, is stored. The information about these original model faces, not necessarily triangular, is stored in the two arrays $OLDNODE$ and $OLDBLST$. In the $ICCB(J,I)$, $1 \le I \le NCB$, where NCB is the number of pairs of contact faces after the triangulation, the indices (to the $IBNDRY$ array) of the faces for each pair I are stored for $1 \le J \le 2$. In the $ICP(J,I)$, $1 \le I \le NCP$, where NCP is the number of pairs of contact nodes after the triangulation, the global indicies of nodes which form the contact pair I are stored for $1 \le J \le 2$.

In the arrays VX, VY, VZ the nodal coordinates of NV mesh nodes and in TR the temperature values in these nodes are stored. In the $CPX(I)$, $CPY(I)$, $CPZ(I)$, $1 \le I \le NCP$, the coordinates of normals for the contact pair I are stored, where

the normal is taken as the outward one with respect to the region containing the node $ICP(1,I)$.

The boundary conditions are characterized by the two real arrays $H1(J,I)$, $H2(J,K)$, $1 \le J \le 3$, $1 \le I \le NVR$, NVR is the number of model vertices before the triangulation, $1 \le K \le NMP$, NMP is the number of nodes on $\overline{\Gamma}_u \cap \overline{\Gamma}_\tau$. The body forces vector and elastic constants are store in the similar manner.

As the variable number of degrees of freedom can occur in each node, ranging from 0 to 3, it is necessary to store this information, together with global indices of corresponding degrees of freedom. For this purpose, an integer array $KOD(J,I)$, $1 \le J \le 3$, $1 \le I \le NV$, is defined. If the degree of freedom J is missing in the node I, i.e. I belongs to $\Gamma_u \cup \Gamma_0$, then $KOD(J,I) < 0$.

The output parameters, i.e. displacements and stresses in nodes and on elements, are stored similarly, moreover, their values are written to files for the graphical output. Due to symmetry, only six components per node or element are stored for the stress tensor.

3.2 Assembling of local systems

The contributions to the Lagrangian $\mathcal{H}$ will be assembled by single elements and boundary faces. Let us consider a tetrahedron $T_n \in \mathcal{T}_h$ with vertices Q_1, Q_2, Q_3, Q_4 and linear mapping

$$\varphi : R^3 \to R^3, \quad \xi \to \varphi(\xi) = \mathbf{x}\,, \tag{3.1}$$

which maps the reference tetrahedron T_{ref} with vertices $\overline{Q}_1 = [0,0,0]$, $\overline{Q}_2 = [1,0,0]$, $\overline{Q}_3 = [0,1,0]$, $\overline{Q}_4 = [0,0,1]$ on T_n. Let us denote for brevity

$$x_{lm} \equiv x_l(Q_m), \qquad l = 1,2,3, \quad m = 1,\ldots,4\,. \tag{3.2}$$

On T_n let us define the functions v_i^* by the formula

$$v_i^*(\xi) = v_i(\varphi(\xi))\,, \quad i = 1,2,3\,, \tag{3.3}$$

where $v_i = v_i(\mathbf{x})$ are linear polynomials on T_n. The matrix of partial derivatives of the mapping φ has the form

$$\mathbf{J}_\varphi(\xi) = \begin{bmatrix} x_{12}-x_{11} & x_{13}-x_{11} & x_{14}-x_{11} \\ x_{22}-x_{21} & x_{23}-x_{21} & x_{24}-x_{21} \\ x_{32}-x_{31} & x_{33}-x_{31} & x_{34}-x_{31} \end{bmatrix}\,. \tag{3.4}$$

The matrix of partial derivatives of the inverse mapping φ^{-1}, $\mathbf{J}_{\varphi^{-1}} = [\mathbf{J}_\varphi]^{-1}$. The corresponding inverse can be obtained by Cramer's rule for the system with right hand sides $(\delta_{i1}, \delta_{i2}, \delta_{i3})$. Let us define

$$\mathbf{v}_{i,}^T(\mathbf{x}) \equiv \left[\frac{\partial v_i}{\partial x_1}, \frac{\partial v_i}{\partial x_2}, \frac{\partial v_i}{\partial x_3}\right](\mathbf{x})\,. \tag{3.5}$$

Similarly

$$(\mathbf{v}_{i,}^*)^T(\xi) \equiv \left[\frac{\partial v_i^*}{\partial \xi_1}, \frac{\partial v_i^*}{\partial \xi_2}, \frac{\partial v_i^*}{\partial \xi_3}\right](\xi)\,. \tag{3.6}$$

It follows from the chain rule

$$\mathbf{v}_{i,}^T(\mathbf{x}) = (\mathbf{v}_{i,}^*)^T(\xi)[\mathbf{J}_\varphi]^{-1} . \tag{3.7}$$

Let us introduce vectors

$$\begin{aligned} \Delta_i &= [\, v_i(Q_1),\, v_i(Q_2),\, v_i(Q_3),\, v_i(Q_4)\,]^T & (3.8)\\ \Delta_i^* &= [\, v_i^*(\overline{Q}_1),\, v_i^*(\overline{Q}_2),\, v_i^*(\overline{Q}_3),\, v_i^*(\overline{Q}_4)\,]^T . & (3.9) \end{aligned}$$

It holds

$$\Delta_i = \Delta_i^* . \tag{3.10}$$

Therefore, on every tetrahedron we seek the solution in the form of linear polynomial (3.3). As φ is linear, v_i^* is also linear polynomial and therefore

$$v_i^*(\xi) = (\mathbf{a}_i^*)^T\mathbf{G}^*(\xi) = \mathbf{G}^*(\xi)^T\mathbf{a}_i^* , \tag{3.11}$$

$$\begin{aligned} \mathbf{a}_i^* &= [\,(a_i^1)^*,\,(a_i^2)^*,\,(a_i^3)^*,\,(a_i^4)^*\,]^T & (3.12)\\ \mathbf{G}^*(\xi) &= [\,1,\,\xi_1,\,\xi_2,\,\xi_3\,]^T . & (3.13) \end{aligned}$$

Analogically to (3.13), let us introduce the vectors of derivatives of monomials

$$\mathbf{p}_i^*(\xi) = \frac{\partial \mathbf{G}^*}{\partial \xi_i}(\xi) \qquad , \tag{3.14}$$

i.e.

$$(\mathbf{v}_{i,}^*)^T = (a_i^*)^T\mathbf{P}^* , \tag{3.15}$$

where

$$\mathbf{P}^* = [\mathbf{p}_1^*, \mathbf{p}_2^*, \mathbf{p}_3^*] \tag{3.16}$$

is the matrix 4×3. Furthermore, we introduce the matrix $\mathbf{S}$, 4×4,

$$\mathbf{S} = [\,\mathbf{G}^*(\overline{Q}_1),\,\mathbf{G}^*(\overline{Q}_2),\,\mathbf{G}^*(\overline{Q}_3),\,\mathbf{G}^*(\overline{Q}_4)\,]^T . \tag{3.17}$$

Therefore, we can express the coefficients of the sought polynomial $\mathbf{a}_i^*$ in dependance on nodal parameters Δ_i^*

$$\mathbf{a}_i^* = \mathbf{S}^{-1}\Delta_i^* . \tag{3.18}$$

Finally, let us introduce the vectors

$$\begin{aligned} \underline{\mathbf{v}} &= [\, v_1,\, v_2,\, v_3\,]^T , & (3.19)\\ \underline{\mathbf{v}}^* &= [\, v_1^*,\, v_2^*,\, v_3^*\,]^T , & (3.20)\\ \underline{\Delta} &= [\, \Delta_1^T,\, \Delta_2^T,\, \Delta_3^T\,]^T , & (3.21)\\ \underline{\Delta}^* &= [\,(\Delta_1^*)^T,\,(\Delta_2^*)^T,\,(\Delta_3^*)^T\,]^T , & (3.22)\\ \underline{\mathbf{a}}^* &= [\,(\mathbf{a}_1^*)^T,\,(\mathbf{a}_2^*)^T,\,(\mathbf{a}_3^*)^T\,]^T , & (3.23)\\ \underline{\mathbf{v}}_{,} &= [\,(\mathbf{v}_{1,})^T,\,(\mathbf{v}_{2,})^T,\,(\mathbf{v}_{3,})^T\,]^T , & (3.24)\\ \underline{\mathbf{v}}_{,}^* &= [\,(\mathbf{v}_{1,}^*)^T,\,(\mathbf{v}_{2,}^*)^T,\,(\mathbf{v}_{3,}^*)^T\,]^T & (3.25) \end{aligned}$$

and block variants of already defined matrices $\mathbf{M} = \mathbf{S}$, $\mathbf{G}^*$, $\mathbf{P}^*$, $[\mathbf{J}_\varphi]^{-1}$

$$\underline{\mathbf{M}} \;=\; diag(\mathbf{M})_i \qquad . \tag{3.26}$$

Therefore,

$$\underline{\mathbf{v}}(\mathbf{x}) \;=\; (\underline{\mathbf{G}}^*)^T(\xi)\underline{\mathbf{S}}^{-1}\underline{\Delta}\,, \tag{3.27}$$

$$\underline{\mathbf{v}}_{,}(\mathbf{x}) \;=\; (\underline{\mathbf{J}}_\varphi)^{-T}(\underline{\mathbf{P}}^*)^T(\xi)\underline{\mathbf{S}}^{-1}\underline{\Delta}\,. \tag{3.28}$$

Let us define

$$\overline{\mathbf{e}} \equiv \overline{e}_{ij} = (2-\delta_{ij})e_{ij}\,. \tag{3.29}$$

Then we can write

$$\sum_{i,j,k,l=1}^{3} c_{ijkl}e_{ij}e_{kl} = \sum_{\substack{i,j,k,l=1\\ i\le j,\; k\le l}}^{3} c_{ijkl}\overline{e}_{ij}\overline{e}_{kl}\,. \tag{3.30}$$

Moreover, we express the dependence $\overline{e}_{ij} = \overline{e}_{ij}(\mathbf{u})$ in the matrix form. Let $\mathbf{T}$ be the 6×9 matrix

$$\mathbf{T} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0\\ 0&1&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&0&1 \end{bmatrix}\,. \tag{3.31}$$

Thus,

$$\overline{\mathbf{e}} = \mathbf{T}\underline{\mathbf{v}}_{,}(\mathbf{x})\,. \tag{3.32}$$

The bilinear form (2.3) can on the element T_n be therefore written as

$$\frac{1}{2}A(\mathbf{v}_h,\mathbf{v}_h)|_{T_n} = \frac{1}{2}\underline{\Delta}^T\underline{\mathbf{S}}^{-T}\left(\int_{T_{ref}}(\underline{\mathbf{P}}^*\underline{\mathbf{D}}^*(\underline{\mathbf{P}}^*)^T)(\xi)d\xi\right)\underline{\mathbf{S}}^{-1}\underline{\Delta}\,, \tag{3.33}$$

where $\underline{\mathbf{D}}^*$ is the matrix 9×9

$$\underline{\mathbf{D}}^* = (\mathbf{J}_\varphi^{-1}\mathbf{T}^T\mathbf{D}\mathbf{T}\mathbf{J}_\varphi^{-T})|\mathbf{J}_\varphi|\,. \tag{3.34}$$

$\mathbf{D}$, the matrix 6×6, represents the coefficients of the Hooke's law, in which the substitution (3.29) is used, and $|\mathbf{J}_\varphi|$ is the jacobian of the mapping φ. If the constant field of body forces is assumed on every element, $\mathbf{F} = (F_1, F_2, F_3)$, we obtain for the body forces in (2.4)

$$\int_{T_n} F_i v_{hi}d\mathbf{x} = \underline{\Delta}^T\underline{\mathbf{S}}^{-T}\left(\int_{T_{ref}}\underline{\mathbf{G}}^*(\xi)d\xi\right)\mathbf{F}|\mathbf{J}_\varphi|\,. \tag{3.35}$$

Similarly,

$$\int_{T_n}\beta_{ij}e_{ij}(\mathbf{v}_h)\theta_h d\mathbf{x} = \underline{\Delta}^T\underline{\mathbf{S}}^{-T}\left(\int_{T_{ref}}(\underline{\mathbf{P}}^*\underline{\mathbf{B}}^*(\mathbf{G}^*)^T)(\xi)d\xi\right)\mathbf{S}^{-1}\nabla\,, \tag{3.36}$$

where $\underline{\mathbf{B}}^*$ is the 9×1 matrix

$$\underline{\mathbf{B}}^* = (\underline{\mathbf{J}}_\varphi^{-T} \mathbf{T}^T \mathbf{b}) |\mathbf{J}_\varphi| \tag{3.37}$$

and the vector $\mathbf{b}$, 6×1, represents the coefficients of thermal expansion. Furthermore, the vector ∇, 4×1, represents (known) nodal values of the piecewise linear function of the temperature. To express the Neumann boundary conditions and the influence of the frictional forces, i.e. the term (2.16), it is necesarry to parametrize the corresponding boundary triangular face $S_{\underline{n}}$.

$$\mathbf{x} = \mathbf{Z}\mathbf{t} + \mathbf{z}\,, \tag{3.38}$$

where $\mathbf{x} = [x_1, x_2, x_3]$, $\mathbf{t} = [t_1, t_2]$,

$$\mathbf{Z} = \begin{bmatrix} x_{12} - x_{11} & x_{13} - x_{11} \\ x_{22} - x_{21} & x_{23} - x_{21} \\ x_{32} - x_{31} & x_{33} - x_{31} \end{bmatrix}\,, \quad \mathbf{z} = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix}\,. \tag{3.39}$$

Let the functions P_i be linear on $S_{\underline{n}}$, let be again determined by the nodal parameters ∇_{P_i}, where ∇_{P_i} is the 3×1 vector.

$$\int_{S_{\underline{n}}} P_i v_{hi} d\mathbf{s} = \underline{\Delta}_0^T \underline{\mathbf{S}}_0^{-T} \left(\int_{S_{ref}} (\underline{\mathbf{G}}_0 \underline{\mathbf{G}}_0^T)(\mathbf{t}) d\mathbf{t} \right) \underline{\mathbf{S}}_0^{-1} \underline{\nabla}_P J_{S_{\underline{n}}}\,. \tag{3.40}$$

At the same time, $\underline{\mathbf{S}}_0$, $\underline{\mathbf{G}}_0$ and $\underline{\Delta}_0$ are defined in the same manner as $\underline{\mathbf{S}}$, $\underline{\mathbf{G}}$ and $\underline{\Delta}$ in the case of body forces vector discretization. The dimension differs, as the face is defined by the three nodes only. Moreover, it is a planar object, i.e. the corresponding approximation polynomial is defined by only two independent variables. $J_{S_{\underline{n}}} = 2\ meas\, S_{\underline{n}}$ can be computed from the knowledge of x_{kl}, $k, l = 1, 2, 3$. S_{ref} is the reference triangle with vertices $[0, 0]$, $[1, 0]$ and $[0, 1]$, $\underline{\nabla}_P = [\nabla_{P_1}^T, \nabla_{P_2}^T, \nabla_{P_3}^T]^T$. The integrand in (2.16) can be for $S_{\underline{n}} \subset \Gamma_c^{kl}$ expressed in the form

$$\overline{\mu}^T \mathbf{H}_i (v_i^k - v_i^l) g_c^{kl}\,, \tag{3.41}$$

where $\mathbf{H}_i$ is 3×1 vector, $i = 1, 2, 3$,

$$\mathbf{H}_i = [\,\delta_{i1} - n_i n_1,\, \delta_{i2} - n_i n_2,\, \delta_{i3} - n_i n_3\,]^T \tag{3.42}$$

and $\overline{\mu}$ is the 3×1 vector. Let us define a 3×3 matrix

$$\mathbf{H} = [\,\mathbf{H}_1,\, \mathbf{H}_2,\, \mathbf{H}_3\,]^T\,. \tag{3.43}$$

As the components $\mathbf{H}_i$ are constant on given face, therefore determined by only one parameter, we obtain by discretizing of the term (2.16)

$$\int_{S_{\underline{n}}} g_c^{kl} \vec{\mu}_h (\mathbf{v}_{ht}^k - \mathbf{v}_{ht}^l) d\mathbf{s} = \underline{\Delta}_0^T \underline{\mathbf{S}}_0^{-T} \left(\int_{S_{ref}} \underline{\mathbf{G}}_0(\mathbf{t}) d\mathbf{t} \right) \mathbf{H} \overline{\mu} \cdot (J_{S_{\underline{n}}} g_c^{kl})\,. \tag{3.44}$$

For every element T_n with possible boundary faces $S_{\underline{n}}$ we obtain a local functional in the form

$$f_n(y_n, \mu_n) = \frac{1}{2} y_n^T C_n y_n - y_n^T d_n + y_n^T G_n^T \mu_n\,, \tag{3.45}$$

where C_n is the 12×12 matrix, y_n, d_n are the 12×1 vectors. The dimension μ_n and G_n depends on the number $S_{\underline{n}} \subset \overline{T}_n \cap \overline{\Gamma}_c^{kl}$, let us denote it as p_n. Then μ_n is of type $3p_n \times 1$ and G_n of type $3p_n \times 12$.

During the assembling of f_n it is necessary to implement the block matrix multiplication efficiently, including the matrices with integrations, i.e. the terms (3.33) and (3.36). For instance, in (3.33) the elements of the matrix $(\underline{\mathbf{P}}^* \underline{\mathbf{D}}^* (\underline{\mathbf{P}}^*)^T)$, i.e. the sums

$$\sum_{k,l=1}^{9} \underline{d}^*_{kl} \underline{p}^*_{ik}(\xi) \underline{p}^*_{jl}(\xi) \quad i,j = 1, \ldots, 12 \tag{3.46}$$

are integrated, and then it is suitable to employ the block diagonal structure of $\underline{\mathbf{P}}^*(\xi)$, where all three blocks are the same matrices $\mathbf{P}^*$ (3.16, 3.26). The essential part of integrations in (3.46) are therefore computations of integrals of monomials

$$\int_{T_{ref}} (p^*_{ik} p^*_{jl})(\xi) d\xi \,, \quad i,j = 1, \ldots, 4, \quad k,l = 1,2,3, \tag{3.47}$$

which are independent of the element T_n. After the assembling of f_n, the degrees of freedom which correspond to the nodes on $(\overline{\Gamma}_u \cup \overline{\Gamma}_0) \cap \overline{T}_n$ are eliminated.

The approximation of the set K_h v (2.9) leads in the case of linear approximation to the constraints

$$A_{cp} y_{cp} \leq 0 \,, \tag{3.48}$$

where the vectors A_{cp} and y_{cp}, which are of type 1×6 and 6×1, respectively, correspond to a given contact pair of nodes.

3.3 Assembling of global system

The resulting global matrices have a sparse structure, therefore the matrix format $SPARSE$ [36], is used, when in general the $M \times N$ matrix A is represented by two integer and one real array. In the array $IA(I)$, $I = 1, \ldots, M+1$ such indices of the arrays JA, A are stored, which correspond to the first element of the I-th row (or I-th column). In the $JA(J)$, $J = 1, \ldots, LJA$, the column (or row) index of the given element and in $A(J)$ its value is stored. In the case of square, symmetric matrix it is possible to store only its lower (or upper) triangular part.

During the assembling of the discretized global functional (2.18), the node numbering is taken into account, together with the information concerning the number of degrees of freedom in particular nodes, i.e. with the array $KOD(3, NV)$, defined above. The block insertion of local element matrices turned out to be efficient. Let $j = j(i)$ be the permutation of local degrees of freedom x_{ni} of element T_n, such that

$$j(i_1) < j(i_2) \Leftrightarrow g(i_1) < g(i_2) \quad \forall 1 \leq i_l \leq 12, \; l = 1,2, \tag{3.49}$$

where $g(i)$ is the global index of the $i-$th local degree of freedom. With this definition, we first assemble "subglobal" matrix having the same dimension as the local matrix. Then the columns of this matrix, from the highest to the lowest, are inserted into the global matrix. By doing this, the number of shifts in the global matrix is minimized.

We finally arrive at the discretized version of the problem (2.19):
Find $(x, \lambda) \in K_d \times L_d$, such that

$$\mathcal{H}(x, \mu) \leq \mathcal{H}(x, \lambda) \leq \mathcal{H}(y, \lambda) \qquad \forall (y, \mu) \in K_d \times L_d, \tag{3.50}$$

where

$$\mathcal{H}(y, \mu) = \tfrac{1}{2} y^T C y - y^T d + y^T G^T \mu \,, \tag{3.51}$$

$$K_d = \{y \in R^N | \, Ay \leq 0\}, \tag{3.52}$$

$$L_d = \{\mu \in R^{3P} | \quad \mu_{3i-2}^2 + \mu_{3i-1}^2 + \mu_{3i}^2 \leq 1, \; i = 1, \ldots, P\} \tag{3.53}$$

and the matrices C, A, G and the vector d are respectively: the positive semidefinite stiffness matrix of type $N \times N$, the constraint matrix of type $M \times N$ expressing the inpenetration conditions (3.48), the frictional forces matrix of type $3P \times N$ and the external forces vector of type $N \times 1$. For M and $3P$, it holds $M \ll N$ and $3P \ll N$, respectively.

3.4 Uzawa's method

We use the Uzawa's algorithm for the problem (3.50) ([29, 38]):
ALGORITHM 3.1

λ^0 ...initial approximation
If λ^k is known, we solve the minimization problem
$\mathcal{H}(x, \lambda^k) \to \min$,
obtaining x^k.
Then we correct
$\lambda^{k+1} = \underline{\Pi}_{L_d}(\lambda^k + \rho {x^k}^T G^T), \quad \rho > 0,$

where $\underline{\Pi}_{L_d}$ is the projection of R^{3P} onto L_d , defined as:

$$\underline{\Pi}_{L_d}(y) = [(\Pi_{L_d}(y))_1^T, \ldots, (\Pi_{L_d}(y))_P^T]^T, \tag{3.54}$$

for $i = 1, \ldots, P$

$$(\Pi_{L_d}(y))_i = \begin{cases} (y_{3i-2}, y_{3i-1}, y_{3i})^T & \text{for} \quad |y|_i \leq 1, \\ (y_{3i-2}, y_{3i-1}, y_{3i})^T / |y|_i & \text{for} \quad |y|_i > 1, \end{cases} \tag{3.55}$$

and $|y|_i \equiv \sqrt{y_{3i-2}^2 + y_{3i-1}^2 + y_{3i}^2}$.
REMARK 3.2 In planar case both definition of L_d

$$L_d = \{\mu \in R^P | \quad |\mu_i| \leq 1, \; i = 1, \ldots, P\} \,, \tag{3.56}$$

and definition of $\underline{\Pi}_{L_d}(y)$

$$(\underline{\Pi}_{L_d}(y))_i = \begin{cases} y_i & \text{for} \quad |y_i| \leq 1, \\ \operatorname{sgn} y_i & \text{for} \quad |y_i| > 1. \end{cases} \tag{3.57}$$

are simplified.

REMARK 3.3 The existence of such ρ_1, ρ_2, that for $\rho \in (\rho_1, \rho_2)$ the component x^k generated by the Uzawa's algorithm converges to the first component x of the saddle point is studied in [22, 29, 36, 38, 53]. During computations, the acceptable value of ρ, i.e. the value, for which the convergence is achieved in real time, has to be chosen experimentally. The dependance between this value and the magnitude of the elements of the matrices C and G can be observed.

The minimization algorithm in the Uzawa's method is the most expensive step of the whole algorithm. For the methods for solving this step, we refer e.g. to [35, 36, 38, 40, 45, 62, 63].

4 H-version of the FEM for the contact problem

4.1 Introduction

The adaptive mesh refinement has proved to be an efficient tool for the numerical solution of partial differential equations. The reduction of the discretization error is accomplished by as few additional degrees of freedom as possible. The mesh refinement represents the h-version of the finite element method, whereas in the p-version the order of the approximation space is increased. The refinement is made only in those parts of the region where the estimated error is great. In structural mechanics this should contain the parts near clamping, in contact regions or near nonconvex boundaries.

The adaptive local mesh refinement has the following general structure:

(1) Construction of the initial coarse grid T_0; $k = 0$

(2) Solution of the problem on T_k

(3) The A-posteriori error estimate calculation for all elements in T_k and construction of $S_k \subset T_k$, where S_k are the elements with great error estimate

(4) If $S_k \neq \{\emptyset\}$, then
    division of the elements in S_k;
    construction of the next "conform" mesh T_{k+1}; $k = k + 1$
    return to (2)
    else
    finish

The steps (2)-(4) for fixed k will be called the refinement iteration. The iteration in (4) for the conform mesh construction will be called the conformity iteration. For the steps (3) and (4) it is necessary to choose such data structures and algorithms which can be relatively simply inserted to existing programs for solving the steps (1) and especially (2). The space and time requirements as well as the robustness have to be taken into account.

The next part of the step (3) determines the elements with great error estimate. There exist two main strategies. In the first strategy we require that the total error, i.e.

the sum of errors η_T^2 on particular elements, is smaller than some prescribed tolerance ε. The sufficient condition for the acceptance of the mesh with N elements reads

$$\eta_T \leq \frac{\varepsilon}{\sqrt{N}} \qquad \forall T \in T_k. \tag{4.1}$$

The division is performed for the elements with

$$\eta_T > \frac{\varepsilon}{\sqrt{N}}. \tag{4.2}$$

However, this strategy is implicit if (4.1) is not fulfilled as N depends on the mesh refinement result. From numerical point of view the choice of ε can also be doubtful. In the second strategy (fixed fraction) the elements are ordered with respect to their error magnitudes and the division is performed for certain part of the elements with greatest error value, i.e.

$$\eta_T > \gamma \cdot \max_{T' \in T_k} \eta_{T'}, \qquad 0 < \gamma < 1. \tag{4.3}$$

This strategy equilibriates the error between the elements and it is used in our calculations.

Similarly to two dimensions case, the tetrahedron can be divided either by octasection or by the bisection. We concentrate on the bisection in what follows. The first algorithm of this type was published in [65] and it is essentially equivalent to that of [1]. The algorithm implemented in our work proceeds from [1] and contains necessary modifications for the conformity preserving on contact boundaries. The numerical performance of both the error estimators and the local refinement algorithm are published in [43, 44, 45, 46].

4.2 Error estimate calculation

We proceed from linear problems for all estimators. The influence of the contact boundary is naturally included for the second and third estimators.

The first estimator was published in [75]. Let

$$\mathbf{e}_\tau = \tau - \tau_h \,, \tag{4.4}$$

where τ is the exact value of the stress tensor and τ_h is its computed aprroximation. The error $\mathbf{e}_\tau$ is measured in the L_2 norm, i.e.

$$||\mathbf{e}_\tau||^2 = \sum_{T \in \mathcal{T}_h} ||\mathbf{e}_\tau||_T^2 \tag{4.5}$$

where

$$||\mathbf{e}_\tau||_T^2 = \int_T (\mathbf{e}_\tau)^T (\mathbf{e}_\tau) d\mathbf{x} \,. \tag{4.6}$$

However, the value τ is not known and therefore it is necessary to replace it by some approximation τ^*, "better" than τ_h. We assume, that the stresses are approximated by the polynomials of the same order as the displacements, i.e. by linear polynomials

$$\tau_{ij}^* = \mathbf{N}\overline{\tau}_{ij}^* \,, \quad i,j = 1,2,3, \tag{4.7}$$

where $\mathbf{N}$ are the basis (shape) functions, cf. (3.27), and $\overline{\tau}^*_{ij}$ are the nodal values of the stress components. The parameters $\overline{\tau}^*$ can be obtained by suitable projection $\underline{\mathbf{P}} = diag\ (\mathbf{P})_{ij}$ with the requirement

$$\int_\Omega \underline{\mathbf{P}}(\tau^* - \tau_h)d\mathbf{x} = 0\,. \tag{4.8}$$

The form $\mathbf{P} = \mathbf{N}^T$ can be chosen, but the simple averaging $\mathbf{P} = \delta_k$, where k are the mesh nodes and δ_k Dirac's function, gives acceptable results [45]. Let us denote the suggested estimator as η_1, i.e.

$$\eta_{T,1} \equiv ||\tau^* - \tau_h||_T\,. \tag{4.9}$$

In the second estimator [67] we assume that we have computed $\mathbf{u}_h$, the approximation of the displacement field and also corresponding strain and stress tensors, $\varepsilon(\mathbf{u}_h)$ and $\tau_h = \Lambda\varepsilon(\mathbf{u}_h)$, respectively. Similarly as in the first estimator we introduce the smoothened stress tensor $\tilde{\tau}_h = A_h\tau_h$, where $\tilde{\tau}_h$ has square summable divergence. Let the terms $g(\varepsilon(\mathbf{u}_h))$ and $g^*(\tilde{\tau}_h)$ denote primal and dual potential energy functionals, respectively [67]. The estimator will consist of local error indicators for the constitutive relations $\eta_{T,\Lambda}$, for the equilibrium equations $\eta_{T,F}$ and for the boundary conditions $\eta_{T,P}$:

$$\eta^2_{T,2} = \eta_{T,\Lambda} + C(\eta_{T,F} + \eta_{T,P}), \tag{4.10}$$

where

$$\eta_{T,\Lambda} = \int_T [g(\varepsilon(\mathbf{u}_h)) + g^*(\tilde{\tau}_h) - \tilde{\tau}_h\varepsilon(\mathbf{u}_h)]d\mathbf{x} + \int_T |\varepsilon(\mathbf{u}_h) - \Lambda^{-1}\tilde{\tau}_h|^2 d\mathbf{x} \quad , \tag{4.11}$$

$$\eta_{T,F} = \int_T |div\,\tilde{\tau}_h + \mathbf{F}|^2 d\mathbf{x} \quad , \tag{4.12}$$

$$\eta_{T,P} = \int_{T\cap\Gamma_\tau} |\tilde{\tau}_h\mathbf{n} - \mathbf{P}|^2 d\mathbf{x} \quad , \tag{4.13}$$

and the constant C is sufficiently large. The $\mathbf{n}$ is the outer normal to $T \cap \Gamma_\tau$ and $\mathbf{F}$ represent external body forces density. From the numerical point of view, the choice of C is not critical since the terms $\eta_{T,F}$ and $\eta_{T,P}$ strongly predominate the term $\eta_{T,\Lambda}$ and since the fixed fraction strategy is used.

The third estimator proceeds from [7]. The elasticity problem can be written as the variational equation

$$A(\mathbf{u},\mathbf{v}) = \int_\Omega F_i v_i d\mathbf{x} + \int_{\Gamma_\tau} P_i v_i d\mathbf{s}\,, \tag{4.14}$$

where $\mathbf{u} \in V$ is the exact solution, $\mathbf{v} \in V$ arbitrary and the non-emptyness of Γ_u is assumed. Let $\mathbf{u}_h \in V_h$ be the approximate solution of (4.14). Putting

$$\mathbf{e} = \mathbf{u} - \mathbf{u}_h \tag{4.15}$$

and integrating by parts over all elements, we get the residual equation

$$\begin{aligned} A(\mathbf{e},\mathbf{v}) \;&=\; \int_\Omega F_i v_i d\mathbf{x} + \int_{\Gamma_\tau} [P_i - \tau_{ij}(\mathbf{u}_h)n_j]v_i d\mathbf{s} \\ &\quad + \sum_{S\subset\Gamma_I} \int_S [[\tau_{ij}(\mathbf{u}_h)n_j]]v_i d\mathbf{s} \quad \forall\mathbf{v} \in V\,, \end{aligned} \tag{4.16}$$

where $\Gamma_I = \bigcup_{k<m} T_k \cap T_m$, T_k, T_m, $k < m$, are the mesh tetrahedrons, $\mathbf{n} = (n_i)_{i=1,2,3}$ is the outward normal with respect to T_k and $[[\tau]] = \tau|_{T_m} - \tau|_{T_k}$. Define local projections of input data

$$\Pi_T \mathbf{F} = \frac{\int_T \mathbf{F} d\mathbf{x}}{meas\ T} \qquad \text{where } meas\ T \text{ is the tetrahedron volume,} \tag{4.17}$$

$$\Pi_S \mathbf{P} = \frac{\int_S \mathbf{P} d\mathbf{s}}{meas\ S} \qquad \text{where } meas\ S \text{ is the area of the face on } \Gamma_\tau. \tag{4.18}$$

Let

$$\mathbf{J}_S = \begin{cases} [[\tau_{ij}(\mathbf{u}_h) n_j]]_S & S \subset \Gamma_I \\ 2\{\Pi_S \mathbf{P} - (\tau_{ij}(\mathbf{u}_h) n_j)|_S\} & S \subset \Gamma_\tau \\ 0 & S \subset \Gamma_u \end{cases} \tag{4.19}$$

The estimator is suggested in the following form

$$\eta_{T,3} \equiv \left[(meas\ T)^2 |\Pi_T \mathbf{F}|^2 + \frac{1}{2} \sum_{S \in E_T} (meas\ S)^2 |\mathbf{J}_S|^2 \right]^{\frac{1}{2}}, \tag{4.20}$$

where $E_T = T \cap (\Gamma_\tau \cup \Gamma_u \cup \Gamma_I)$.

The fourth estimator is based on solutions of local problems. and it was analyzed for the Helmholtz's equation in the plane [8]. Thus

$$l(u) \equiv -\nabla a \nabla u + bu = f \text{ on } \Omega \subset R^2 \tag{4.21}$$

$$a \frac{\partial u}{\partial n} = g \text{ on } \partial\Omega \tag{4.22}$$

with the assumptions $a \in C^1(\overline{\Omega})$, $b \in C^0(\overline{\Omega})$, there exist constants $\underline{a}$, $\overline{a}$, $\underline{b}$, $\overline{b}$, such that

$$0 < \underline{a} \le a(\mathbf{x}) \le \overline{a} \tag{4.23}$$

$$0 < \underline{b} \le b(\mathbf{x}) \le \overline{b} \quad . \tag{4.24}$$

Let $(\cdot,\cdot)_\omega$ denote a scalar product on ω for $\omega \subseteq \Omega$. Similarly for $\gamma \subseteq \partial\Omega$, let $< \cdot, \cdot >_\gamma$ denote a scalar product on γ.
The variational formulation of (4.21-4.22) is: Find $u \in H^1(\Omega)$

$$\underline{A}(u,v) = (f,v)_\Omega + < g, v >_{\partial\Omega} \quad \forall v \in H^1(\Omega), \tag{4.25}$$

where

$$\underline{A}(u,v) = \int_\Omega (a \nabla u \nabla v + buv) d\mathbf{x}. \tag{4.26}$$

Denote

$$|||u|||^2 \equiv \underline{A}(u,u). \tag{4.27}$$

DEFINITION 4.1 Let us consider a tetrahedral mesh on Ω. Let H be the space of piecewise linear polynomials, in which we find the solution u_h and $\overline{H}$ be the space of piecewise quadratic polynomials, where the approximation of the error is sought. Let H_T and $\overline{H}_T$ be the restrictions of H and $\overline{H}$ onto the elements of the mesh T.

DEFINITION 4.2 $\check{H}_T \equiv \{v \in \overline{H}_T,\ Iv = 0\ \}$, where I is the Lagrange's interpolation operator. $\check{H}_T$ is the "bubble functions" space.

The estimator is defined as the solution of the following residual equation

$$\underline{A}(\check{e}, v)|_T = F_T(v) \quad \forall v \in \check{H}_T\,, \tag{4.28}$$

where

$$\begin{aligned} F_T(v) &= (f - l(u_h), v)_T + < g - a\frac{\partial u_h}{\partial n}, v >_{E_T \cap \partial\Omega} \\ &\quad + \frac{1}{2} < [[a\frac{\partial u_h}{\partial n}]], v >_{E_T \cap \Gamma_I}\ . \end{aligned} \tag{4.29}$$

In our case, the bilinear form corresponding to elasticity problem is assumed in (4.28), and instead of the scalar function u_h, we seek the vector function $\mathbf{u}_h$. The operator $l(\mathbf{u}_h)|_T = 0$, as the elasticity problem is described by the Lame's equations [53], where only second derivatives of the solution appear and $\mathbf{u}_h$ is piecewise linear. The substitution of remaining terms of (4.29) by terms from (4.16) is obvious. Thus

$$\eta_{T,4} \equiv \check{\mathbf{e}}\,. \tag{4.30}$$

4.3 Implementation of estimators for contact problems

The approximation of the error is based on the knowledge of element values of stresses (the vector $\underline{\Delta}$ of type 6×1) and nodal values of stresses (the vector $\dot{\underline{\Delta}}$ of type 24×1). Here, we assume the relation

$$\tau^* = (\underline{\mathbf{G}}^*)^T \underline{\mathbf{S}}^{-1} \dot{\underline{\Delta}}\,, \tag{4.31}$$

cf. (4.7), where $\underline{\mathbf{G}}^*$ is the block diagonal matrix, consisting of 6 blocks, each block is the matrix of monomials $\mathbf{G}^*$ of type 4×1. The similar block structure has also $\underline{\mathbf{S}}$, the blocks of which are 4×4 matrices. In the case of η_1 estimator, the value $||\mathbf{e}_\tau||_T^2$ with $\tau \approx \tau^*$, has to be computed.

Similarly to the section 3.2 we obtain the dependence of this value on the nodal and element parameters

$$\begin{aligned} (\dot{\underline{\Delta}})^T \underline{\Delta} \left(\int_{T_{ref}} |\mathbf{J}_\varphi| d\mathbf{x} \right) &- 2(\dot{\underline{\Delta}})^T \underline{\mathbf{S}}^{-T} \left(\int_{T_{ref}} \underline{\mathbf{G}}^* |\mathbf{J}_\varphi|\ d\mathbf{x} \right) \underline{\Delta} \\ &+ (\dot{\underline{\Delta}})^T \underline{\mathbf{S}}^{-T} \left(\int_{T_{ref}} \underline{\mathbf{G}}^* (\underline{\mathbf{G}}^*)^T |\mathbf{J}_\varphi|\ d\mathbf{x} \right) \underline{\mathbf{S}}^{-1} \dot{\underline{\Delta}}\,, \end{aligned} \tag{4.32}$$

where the notation of section 3.2 is used.

In the estimator (4.20), we concentrate on the computation of $\mathbf{J}_S$ (4.19). We assume constant values of stresses on all the faces. It is not necessary to search all the neighbours for a given element, as the nodal values are known, together with the values on the whole element. Let us introduce the average value of the stress tensor on each

face $\check{\tau}$ by averaging the values in all nodes of the given face. Now, the stress jump can be replaced in the natural way by

$$\frac{1}{2}[[\tau_{ij}(\mathbf{u}_h)n_j]] \approx \check{\tau}_{ij}n_j - \tau_{ij}n_j|_{T_k}\,, \tag{4.33}$$

where T_k is the tetrahedron, on which we compute the error.

During the insertion of conditions on contact part of the boundary, we will try to transform the problem to the certain linear elasticity problem. It can be seen that $\eta_{T,3}$ explicitly depends on the boundary conditions with dual quantities (stresses). The residual between the computed and prescribed stress is measured. Therefore, we consider those conditions on contact boundary which contain only stresses [43, 45]. We measure the values $\alpha = [\tau_n]^+$ and $\beta = [|\vec{\tau}_t| - g_c]^+$. Let us introduce the vector $\alpha\mathbf{n} + \delta\vec{\tau}_t$ in (4.19) for Γ_c, where $\delta = \frac{\beta}{\gamma}$, $\gamma = \max(|\vec{\tau}_t|, \varepsilon_T)$ and $\varepsilon_T > 0$. Furthermore, the conditions of action and reaction on the contact are transformed to the stress jump measurement as for the faces on Γ_I.

In the last estimator, the piecewise quadratic functions with zeros in the element vertices in 3D are chosen as bubble functions space. The right hand side is similar to that of the quantity $\mathbf{J}_S$ appearing in the second estimator. The quadratic polynomials are determined by the nodal values and the values in the midpoints of all tetrahedron edges. After the local assmbling, it is necessary to eliminate the degrees of freedom which correspond to the vertices of the tetrahedron. Thus, for every element it is necessary to assemble and solve systems with up to 18 unknowns.

4.4 Local refinement algorithm

Let us return to the step (4) of the overall adaptive mesh refinement algorithm. The bisection algorithm accomplishes the mesh conformity by the iterative bisection of those tetrahedra which do not meet the conformity condition. During the bisection, a certain edge, called the refinement edge, is chosen and the new vertex is placed to its centre. With this new vertex, two new tetrahedrons are created (Fig. 4.1). The selection of the refinement edge is therefore a key aspect of the algorithm. In addition to the algorithm itself, we introduce a new data structure called "marked tetrahdron". It will be used for the geometrical data of the tetrahedron together with a small amount of informations which are needed for the bisection.

DEFINITION 4.3 For tetrahedron T, let $V(T)$,$E(T)$ and $S(T)$ denote the set of its vertices, edges and faces, respectively. Let $E(\varphi)$ denote the set of edges of the face $\varphi \in S(T)$, let r_T be the refinement edge. Furthermore, let $m_\varphi \in E(T)$ be the marked edge of the face φ. Finally, let $f_T \in \{0, 1\}$ be a boolean flag. Then a marked tetrahedron is a 4-tuple $(V(T), r_T, (m_\varphi)_{\varphi\in S(T)}, f_T)$. For the refinement faces, i.e. for the two faces that intersect at the refinement edge, the marked edge coincides with the refinement edge.

Marked tetrahedrons will be divided into four classes by a mutual position of their marked edges (Fig. 4.2).

P - marked edges are coplanar. A type P is further classified as type P_u or P_f, according to whether its flag is set or not.

A - the marked edges intersect the refinement edge, but are not coplanar,

O - the marked edges of the non-refinement faces do not intersect the refinement edge. In this case, a pair of opposite edges are marked in the tetrahedron,

M - the marked edge of just one of the non-refinement faces intersects the refinement edge.

DEFINITION 4.4 Let T_1 and T_2 be the child tetrahedra created by the bisection of the T. A face $\varphi \in S(T_i)$ is inherited (i), if $\varphi \in S(T)$. The faces $\varphi \in S(T_i)$, for which $\exists \varphi' \in S(T)$, $\varphi \subset \varphi'$, are cut faces (c). A face, which is neither inherited nor cut is called a new face (n). Cf. Fig. 4.3.

ALGORITHM 4.5
$\{T_1, T_2\} = BisectTet(T)$
Input: A marked tetrahedron T
Výstup: Marked tetrahedra T_1, T_2
Cf. Fig. 4.4

1. Bisect T by joining the midpoint of its refinement edge to each of the two vertices not lying on the refinement edge. This defines new sets $V(T_1)$ and $V(T_2)$.
2. Mark the faces of the children as follows:
   a) The inherited face inherits its marked edge from the parent, and this marked edge is the refinement edge of the child.
   b) On the cut faces of the children mark the edge opposite the new vertex with respect to this face.
   c) The new face is marked the same way for both children. If the parent is type P_f, the marked edge is the edge connecting the new vertex to the new refinement edge. Otherwise it is the edge opposite the new vertex.
3. The flag is set in the children if and only if the parent is type P_u.

It is proved in [1], that the repeated application of the Algorithm 4.5 on an arbitrary initial tetrahedron cannot create degenerate tetrahedra for any of its succesors.

DEFINITION 4.6 Let $V(\mathcal{T})$ be a set of mesh nodes which belong to the region $\Omega \subset R^3$. The node $\nu \in V(\mathcal{T})$ is a hanging node of T, if $\nu \in T - V(T)$.

DEFINITION 4.7 A mesh is marked if each tetrahedron in it is marked. A marked conforming mesh is conformingly marked if each face has a unique marked edge.

In a marked conforming mesh the face shared by two tetrahedra defines for both of them the same marked edge. Let a conforming mesh be given. Let the edges be ordered in certain manner. Let us choose as the refinement edge of the tetrahedron its longest edge and as the marked edge (of its two remaining faces) the longest edge of this face. Assume that $f_T = 0$ holds for all tetrahedra in the initial mesh. The following algorithm represents the process of local adaptive mesh refinement.

ALGORITHM 4.8
$\mathcal{T}' = LocalRefine(\mathcal{T}, \mathcal{S})$
Input: Conformingly marked mesh $\mathcal{T}$, $\mathcal{S} \subset \mathcal{T}$ and $f_T = 0 \ \forall T \in \mathcal{T}$
Output: Conformingly marked mesh $\mathcal{T}'$

1. $\overline{\mathcal{T}} = BisectTets(\mathcal{T}, \mathcal{S})$
2. $\mathcal{T}' = RefineToConformity(\overline{\mathcal{T}})$

At the same time $BisectTets(\mathcal{T}, \mathcal{S}) = (\mathcal{T} - \mathcal{S}) \cup \bigcup_{\mathcal{T} \in \mathcal{S}} BisectTet(\mathcal{T})$ and $RefineToConformity$ is described in the following algorithm.

ALGORITHM 4.9
$\mathcal{T}' = RefineToConformity(\mathcal{T})$
Input: Marked mesh $\mathcal{T}$
Output: Marked mesh $\mathcal{T}'$ without hanging nodes

1. $\mathcal{S} = \{T \in \mathcal{T} |\ T$ has a hanging node $\}$
2. if $\mathcal{S} \neq \emptyset$ then

   $\overline{\mathcal{T}} = BisectTets(\mathcal{T}, \mathcal{S})$

   $\mathcal{T}' = RefineToConformity(\overline{\mathcal{T}})$

   else

   $\mathcal{T}' = \mathcal{T}$

This recursive algorithm can be easily transformed to the non-recursive one. Therefore, every recursive call of $RefineToConformity$ represents one iteration of conformity.

4.5 Implementation of the local refinement algorithm

For the adaptive mesh refinement, it is convenient to define an array of edges $IEDGE(J,I)$, $1 \le I \le NED$, $1 \le J \le 3$, where NED is the number of edges in the mesh. Additionally to the endpoints' indices, the information about the child edge, $IE3 = IEDGE(3,I)$, will be stored. If $IE3 = 0$ then the edge I has not a descendant. If $IE3 < 0$ then a descendant was already created in the current conformity iteration and the value $-IE3$ is the pointer to the heap array, where the first child edge is temporarily stored. If $IE3 > 0$ then $IE3$ is the pointer to the $IEDGE$ array, where the first child edge is stored. The $IEDGE$ array is sorted in the lexicographical order.

DEFINITION 4.10 The edge e_j is greater then e_i, if
$IEDGE(2,e_j) > IEDGE(2,e_i)$ or
$IEDGE(2,e_j) = IEDGE(2,e_i)$ and $IEDGE(1,e_j) > IEDGE(1,e_i)$.

This ordering is defined uniquely. The edge search in the $IEDGE$ array can be performed by the bisection method. The subroutine $FINDEDGE$ finds for a pair of endpoints I, J a corresponding edge L in the $IEDGE$ array. If $L < 0$, then the edge is greater than the edge $-L$, smaller than the edge $-L+1$ and is not yet inserted. The cases $L = 0$ and $L = NED$ are also taken into account.

The edges of the tetrahedron are characterized by a local vertex (node) numbering ($LOCEDGE(2,6)$). Furthermore, we consider a list of vertices and edges with respect to particular faces of the tetrahedron ($LOCFACN(3,4)$ and $LOCFACE(3,4)$). In Fig. 4.5 the local vertex indices, the local edge indices (underlined) and the local face indices (italics) are depicted. This is the case when the refinement edge connects the vertices 2 and 3. The tetrahedron type can then be written as a four digit number which express the relation face - its marked edge. The particular digits represent local edge indices, being therefore between 1 and 6. If the tetrahedron is not of type O, the index of the refinement edge is determined uniquely. If it is of type O and the local edge index of the edge opposite the refinement edge is lower then the one of refinement edge itself, then the uniquenes is achieved by the change of sign of the four digit number. The cases P_u and P_f are distinguished in the same manner. As it follows from the Algorithm 4.5 and Algorithm 4.9, that the types P_f and O cannot occur in the mesh simultaneously, the information concerning the tetrahedron type can be represented by an integer variable $IETYPE$. We add the sixth component to the $ITNODE$, i.e. $ITNODE(6,I) = IETYPE$. It holds for each tetrahedron

$$1122 \le |IETYPE| \le 6655\,. \tag{4.34}$$

When generating the initial conform mesh, we have to decide which edge will be the marked one. The requirement is that the tetrahedrons sharing this face have the same marked edge for this face. In the case of a pair of faces which form the contact, we extend this requirement by the condition that the marked edges have to be geometrically equivalent. The endpoints of these edges have different global nodal indices but the same coordinates. By doing this, the problem is transformed to the above case, together with various subroutines which "balance" possible incompleteness

of newly defined pairs of contact nodes. It is necessary to choose such ordering which fulfils these requirements. At the same time we cannot assume anything about the node numbering on contact faces and edges.

DEFINITION 4.11 Consider the edges e_i and e_j with lengths d_i and d_j and with midpoint coordinates $c_i = [x_i, y_i, z_i]$ and $c_j = [x_j, y_j, z_j]$. The edge e_j is geometrically greater than e_i if $d_j > d_i$ or $d_j = d_i$ and $x_j > x_i$ or $d_j = d_i$, $x_j = x_i$ and $y_j > y_i$ or $d_j = d_i$, $x_j = x_i$, $y_j = y_i$ and $z_j > z_i$.

The marked edge of the face in the initial mesh is the edge geometrically greater than all other edges of this face and therefore is determined uniquely. The refinement edge of the tetrahedron is defined in the same manner.

The implementation of $LocalRefine$ is realized by the subroutine $BISTETS$, where the conformity iterations are performed. In the initial conformity iteration the bisection of the element is based on the value resulting from an error estimator. In the next iterations, the element is bisected if there exists an edge of this element for which

$$IEDGE(3, L_i) > 0,\ i = 1, \ldots, 6,\ 1 \leq L_i \leq NED\,. \tag{4.35}$$

The bisections of particular elements are performed by the subroutine $BISECTTET$. From the Algorithmu 4.3 it follows that for bisection the explicit determination of the tetrahedron type (P_u, P_f, A, O, M) is not necessary. It is only necessary to distinguish coplanar cases (P_u, P_f) and other cases, where the particular types A, O and M need not be distinguished. Each child of the parent tetrahedron inherits one vertex of the refinement edge. The second vertex is replaced by a new one. In Fig. 4.5 the refinement edge is the edge connecting vertices 2 and 3. The local index of the missing vertex of the refinement edge is assigned to the local index of a new vertex. The type of face of the child tetrahedron and then also the choice of the marked edge is determined by the mutual position of the face, nodes of the refinement edge and the new node. In the case of Fig. 4.5 the face *2* is inherited, the faces *1* and *4* are cut faces and the face *3* is new for the left child. For the right child, the face *3* is inherited, the faces *1* and *4* are cut faces and the face *2* is new.

As the data structures of the original program were stored in two global arrays, it is convenient in every conformity iteration to store newly defined entities first on the heap. The heap is represented by those parts of global arrays which were not yet allocated. After the bisection of a given entity is done, the number of newly defined entities is known and their insertion to the corresponding entity, i.e. to a certain place in the global array can be performed at one time.

The first entity for which new elements are created are simultaneously tetrahedrons ($ITNODE(6,*)$) and edges ($IEDGE(3,*)$). The index of a new node is represented by a negative whole number IG in $ITNODE$ and $IEDGE$. One successor of the tetrahedron I is stored in the place of original tetrahedron, i.e. in $ITNODE(J, I)$, $J = 1, \ldots, 6$. At the same time, in the material entry, i.e. in the $ITNODE(5, I)$ resides the negative value of the heap index NHP, where the information about the second successor and a possible information about new edges created from the refinement one is stored. With these edges, a list of newly created nodes is uniquely defined. The

other pair of edges can be created during the bisection, however, their vertices are not stored as they can be obtained from both new tetrahedrons. Thus on the heap the new entity occupies at least six items: the vertices of a new tetrahedron, material type of a new tetrahedron and $IETYPE$, the type of a new tetrahedron.

However, the case when a new node and a pair of child edges of the refinement edge are created, has to be distinguished. This can be achieved by substituting of $IETYPE$ by $10*IETYPE$. The original information in $IETYPE$ remain unchanged and regarding to (4.34) it holds for an arbitrary tetrahedron

$$|10 * IETYPE| \geq 11220 > 6655 \geq |IETYPE| . \tag{4.36}$$

Let now the refinement edge have an index L. Then $IEDGE(3, L) = -(NHP + 6)$ and therefore the list of newly defined nodes is indeed easily accessible through the array $IEDGE$. In the case of newly defined node it is also desirable to define a pointer $NRHP$ to the real heap, where its coordinates and the interpolation of temperature values to this node are stored. In Fig. 4.6 we assume that the refinement edge is $I2-I3$ and that a node IG is created.

We require on the index IG

$$|IG| > |IE_i|, \, i = 1, 2 \text{ and } |IG| > |IG_0| , \tag{4.37}$$

where IE_i are the vertex indices of the refinement edge and IG_0 are the indices already created nodes in the current conformity iteration. On the other hand, we naturally minimize $\max\limits_{i=1,2} (|IG| - IE_i)$.

After one conformity iteration we get $NTHEAP$ new tetrahedrons, $NEHEAP$ new edges and $NVHEAP$ new nodes on the heap. During the insertion of nodes, it is necessary to change the entries of all the arrays which point to the nodes, i.e. the array of tetrahedrons $ITNODE$, the array of boundary elements $IBNDRY$, the array of edges $IEDGE$ and also the array of pairs of contact nodes ICP and the array of bilateral contact nodes. The corresponding entries have to be changed also on the heap itself. Then new tetrahedrons and edges are inserted into the arrays $ITNODE$ and $IEDGE$.

Let us suppose that the bisection process influences given pair of contact nodes. Then the algorithm generates one of the cases a) - e) in the Fig. 4.7. The balancing subroutines transform the cases b) and c) to the case a) and the case e) to the case d). All three edges of the faces are checked. At this time, also new pairs of contact nodes can be inserted into the array ICP.

The next entity for which new elements are created due to the bisection are the boundary faces, i.e. the array $IBNDRY$. We proceed in the similar manner as in the case of tetrahedrons. Both child faces have the same orientation as the parent face. A pair of newly created faces has to be distinguished from the old ones, due to the necessary modification of the array of pairs of contact faces $ICCB$. This can be achieved e.g. by setting the entry $IBNDRY(5, I)$ negative, where I is the first of the newly created faces.

5 Instructions for users of FEC

5.1 Compiling and linking of the FEC

We use the environment of the Digital Visual Fortran 5.0 compiler under Win98/NT (Microsoft Developer Studio - MSDEV). Let the directory where the current version (6.0) of the three-dimensional FEC resides be called Fec3dv06.

To create a corresponding project, run MSDEV and click New, then select Projects tab. Here choose Win32 Console Application and as the Project name choose Fec3dv06. After that, select project options by clicking on the Project Menu and Project Settings. Various settings are predefined for the tabs General, Debug, Fortran, Link, etc. For the debugging phase, we recommend to leave current settings. For the release phase, the following can be e.g. changed. On the Fortran tab, change for the General Category the Debugging Level from Full to None and the Optimization level from None to Optimize for Speed. On the Link tab, change the Output file name from Debug/Fec3dv06.exe to Fec3dv06.exe and uncheck the box Generate debug info. Then we have to add the files into the project by clicking on Project and Add to Project menus. The following Fortran source files have to be included into the project: fec06.f, utils3.f, cg10.f, proj0.f and adapt.f. After inserting them, save the workspace by clicking File and Save Workspace.

The compilation and linkage can be performed at one step by clicking Build Fec3dv06.exe. When correcting a particular source file, the compilation of this file can be invoked by opening (or activating if already opened) the corresponding window and clicking Compile. The link can then be performed again by clicking on Build or if we wish to be sure that all updated files are really compiled by clicking Rebuild All. The program can be run in the Debugging mode by clicking Build, Start Debug and e.g. Run to Cursor, where the cursor is placed on the line of the source code which we wish first to analyze. Normal run can be invoked by clicking Build and Execute Fec3dv06.exe or directly outside the MSDEV studio by clicking on the file Fec3dv06.exe.

5.2 The input/output files

The input file has the name usrot.dat. It is a binary file, which can be obtained from its text form, usrot.txt, by running the utility a2b.exe (ascii to binary). However, for some concrete models we may use transformation programs that create directly a binary version. To understand the format of input file, it is advisable to read section 3.1 of this report and the comments in the INDATA subroutine (a2b.for) or in the INDATA, INDATA2 subroutines (fec06.f). Generally speaking, the input values consist of integer and real arrays and their dimensions. In integer arrays, pointers to real or other integer ones are stored. In real arrays, the nodal coordinates, the normal coordinates, material parameters and the values of external forces and of boundary conditions are stored. The program creates 5 binary output files for further graphical post-processing. These files are

values1.dat - values of displacements on every element. The order of components is u_1, u_2, u_3 (obtained by the averaging of nodal values for each element). This file

is created in the subroutine TRANSF1.

values2.dat - values of displacements in every node. The order of components is u_1, u_2, u_3. This file is created in the subroutine TRANSF1.

values3.dat - values of the stress tensor on every element. The order of components is τ_{11}, τ_{22}, τ_{33}, τ_{12}, τ_{23}, τ_{13}, $\overline{\tau}$, where

$$\overline{\tau} = \sqrt{\tau_{11}^2 + \tau_{22}^2 + \tau_{33}^2 - \tau_{11}\tau_{22} - \tau_{11}\tau_{33} - \tau_{22}\tau_{33} + 3(\tau_{12}^2 + \tau_{13}^2 + \tau_{23}^2)}.$$

This file is created in the subroutine ELSTRESS.

values4.dat - values of the stress tensor in every node. The order of components is τ_{11}, τ_{22}, τ_{33}, τ_{12}, τ_{23}, τ_{13}, $\overline{\tau}$ (obtained by the averaging of element values in each node). This file is created in the subroutine TRANSF2.

valpst.dat - scalar values of principal stresses (first 3 items) and the coordinates of two principal vectors (last 6 items) on every element. The third principal vector can then be already computed. This file is created in the subroutine TRANSF2.

When the adaptation process is switched on (IADAPT=1/2), the additional file usrota.dat is created. The format of this file is the same as of the usrot.dat. It contains the model geometry after performing one refinement iteration. In this version of the program, this file has to be copied to the usrot.dat and the program has to be run again, if we wish to continue with the adaptation process.

5.3 Running the FEC

When running, the program interactively asks only for a few data. IADAPT=0/1/2. 0 - no adaptive refinement is performed, 1 - adaptive refinement is performed, 2 - same as 1, but the value $\max_{T' \in T_k} \eta_{T'}$ is substituted by $\max_{T' \in T_0} \eta_{T'}$, see section 4.1. By doing this, the number of bisected elements decreases when increasing the number of refinement iteration (relative comparison to the error on coarser grid, which is expected to be greater).

The program writes on the output the numbers maxw, maxiw that express how many entries of global arrays are used. The next query of the program is SOLV-RO=?, for the parameter ρ of the Uzawa algorithm, see section 3.4. If RO=0, no friction is assumed. INPUT MODE 1/2 - select the number of removed multipliers in the active set method [36, 38, 62, 63], 1 - one, with the greatest negative value, 2 - all with negative values. INPUT IDIAG=0/1 - select the method for the computation of the projected gradient, 1 - assumes diagonal form of the constraint matrix product [36, 62, 66], 0 - general case. LOWER/HIGHER TOLERANCE L=1/H=-1, this is controlled by the EPS2 in CG1 and CG2 subroutines. "nextit 0/!0" - input zero, if the user wishes the Uzawa iterations run automatically, nonzero number, if he or she wishes to pause after next Uzawa's iteration.

5.4 Graphical postprocessing of the results

The data in the files usrot.dat, values?.dat and valpst.dat can be visualized by a simple graphical tool scene1.exe. Due to the permanent development of FEC, it is assumed for clarity that the graphics which include the executable scene1.exe and the files for a possible rebuild of the project (i.e. the files *.cpp, *.c, *.h, *.dfm and scene1.mak) reside in the directory fecg3b. When making possible changes in the source files, the project has to be rebuilt using the Borland C++ Builder compiler (BCB) under Win98/NT. The execution itself, outside the BCB, naturally requires the same operating systems. Moreover, the palette files, palets.txt, paletp.txt and paletp2.txt, are required for the execution.

After starting the scene1.exe, the first query concerns the type of output device on which the data is to be displayed. The option 0 (scr) means the output only on the screen, when the black background is set. The option 1 (prtscr) corresponds to printing the exported picture (see below) on the printer, therefore with white background. This option, however, displays lower number of contour levels and therefore the option 2 (prtscrall) should be preferred. This option is especially convenient when publishing figures in papers and reports. Each picture displayed in the scene1 is saved in the file timage.bmp until a next picture is displayed. Timage.bmp can be processed e.g. by the Windows utility Paintbrush. After accepting a value, the user then selects by the Tab key the EnterButton and presses it.

The second query (auto=1/man=0) concerns the issue whether the user lets the program plot the whole model automatically (auto=1, suitable at the beginning of the analysis) or whether he or she determines the position of the observer and the direction of the view manually (man=0).

The third query controls the zooming of the view. If the option 1 was selected in the previous (i.e. second) query, one may use the value 0, which means automatic zoom with fitting the model on the whole screen. Otherwise, the user enters a floating point number corresponding to required magnification.

The 4-th query is conditional, it appears only if zooming>0. It concerns shifts of the screen in horizontal and vertical directions. The standard values are dx=dy=0.

In the 5-th query, "barveni 1-coarse 2-smooth", the smooth (2) is highly recommended, whereas (1) corresponds to the constant values on elements.

The 6-th query "Select region/0". If the user wishes to plot only one selected region of the model, then he or she enters its corresponding number, 0 otherwise.

The 7-th query "Type in eye position" is conditional, it appears only if the second query is man=0. The same holds for the 8-th query, "And direction and angle of view". The standard value of angle is 0.67 (fraction of the right angle).

Entering an integer i in the 9-th query will ensure that only nodes (and corresponding elements) that are farther from the observer than i-th node, the so-called cutpoint, will be plotted. The zero value plots all the nodes.

The 10-th query offers the option of filtering. It filters out the faces fully inside the model, thus accelerating the plotting algorithms.

Finally, remaining queries concern the plot itself. The 11-th one offers options whether to plot all node numbers (1), to plot only the numbers of keypoints (i.e. the

nodes which correspond to some model vertex before the triangulation), to plot only the selected points (3 - the indices and offsets are then entered), to finish and proceed to the results view (0) or to exit the program (-1).

The 12-th query offers options concerning type of contours (surface=1, isolines=0). For the 3D model, the "surface=1" option is highly recommended.

In the 13-th query, the user can choose between two plotting algorithms - the option "view=0" is much faster, however, "final=1" can produce nicer pictures.

The 14-th query selects which quantity will be displayed. If the user wishes to proceed to the next step, enters 0.

The 15-th query concerns principal stresses. If the user enters 0.0 for the scale factor, the program immediately proceeds to the 16-th query.

This final query concerns the deformed shape plot. The coefficient of deflection and then a type of plot are entered. The option "element=1" is the fastest, but does not use a hidden line/surface removal. The program will first display the initial configuration and then after pressing an arbitrary key, e.g. the spacebar, the deformed geometry. With the next press of the spacebar, the program is finished and the corresponding window can be closed.

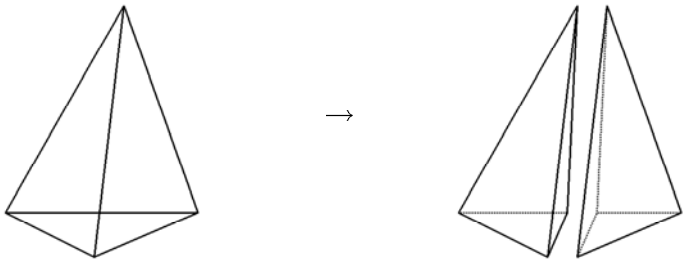


Figure 4.1:

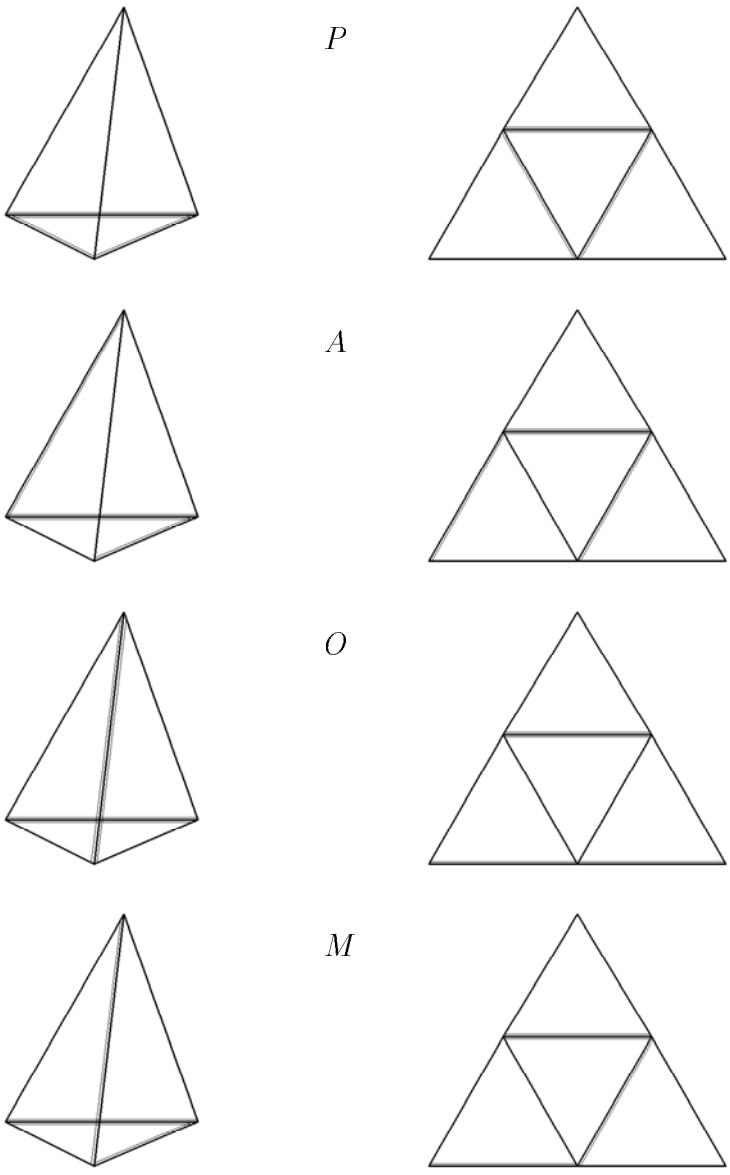


Figure 4.2:

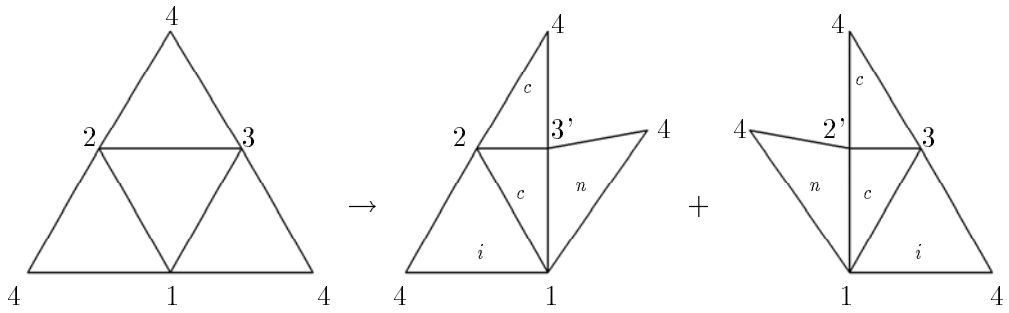


Figure 4.3:

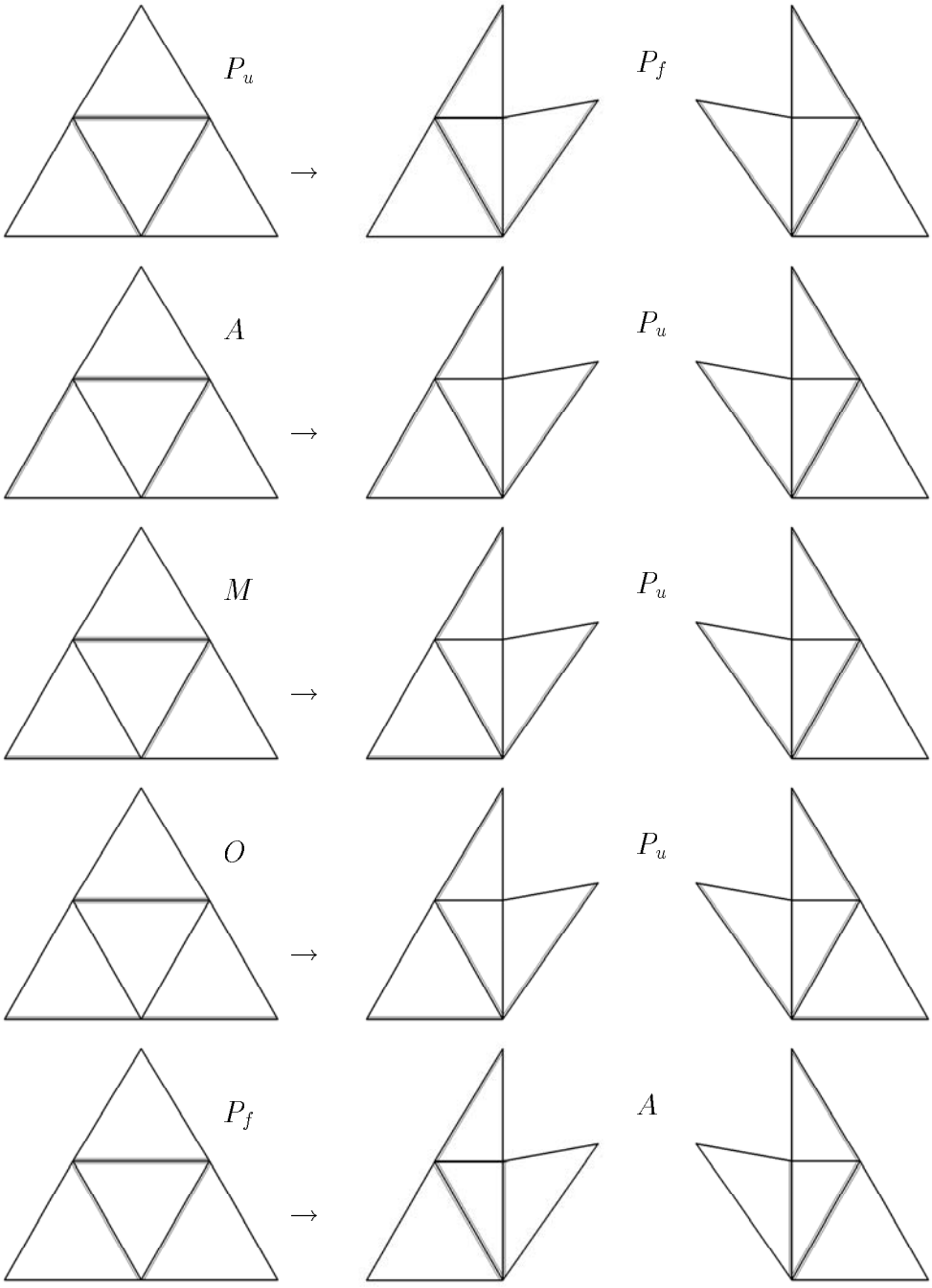


Figure 4.4:

$$
\begin{array}{rcccccc}
 & & 1 & 2 & 1 & 1 & 2 & 3 \\
LOCEDGE(2,6) & = & 2 & 3 & 3 & 4 & 4 & 4
\end{array}
$$

$$
\begin{array}{rcccc}
 & & 1 & 1 & 1 & 2 \\
LOCFACN(3,4) & = & 2 & 4 & 3 & 4 \\
 & & 3 & 2 & 4 & 3
\end{array}
$$

$$
\begin{array}{rcccc}
 & & 1 & 1 & 3 & 2 \\
LOCFACE(3,4) & = & 2 & 4 & 6 & 5 \\
 & & 3 & 5 & 4 & 6
\end{array}
$$

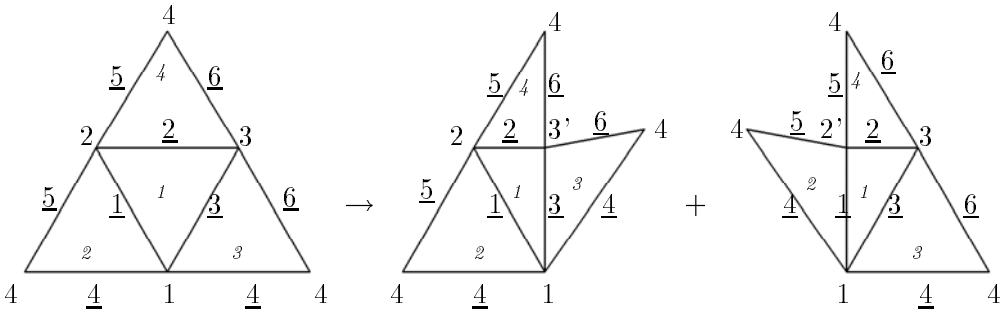
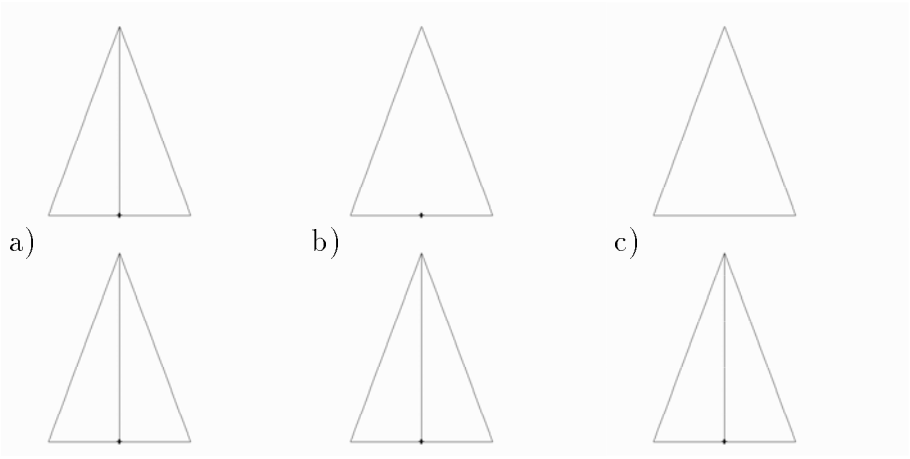


Figure 4.5:

	$I1$	IG	$I3$	$I4$	$M.$	$IET.$	$I2$	IG	$I3$	$NRHP$
	NHP						NHP+6			

Figure 4.6:



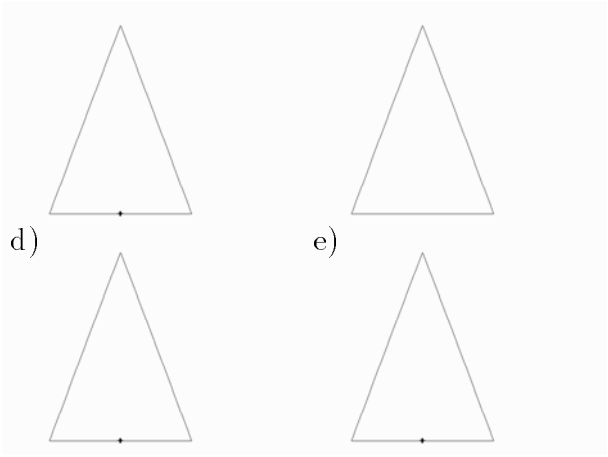


Figure 4.7:

Bibliography

[1] D.N. Arnold, A. Mukherjee and L. Pouly, Locally adapted tetrahedral meshes using bisection, *SIAM J.Sci.Comp.* 22 (2000) 431-448.

[2] O. Axelsson, V.A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation* (Academic Press, New York, London, 1984).

[3] I. Babuška, A. Miller, *A-Posteriori Error Estimates and Adaptive Techniques for the Finite Element Method.* (Technical Note BN-968, University of Maryland, 1981).

[4] I. Babuška, W.C. Rheinboldt, Error estimates for adaptive finite element computations, *SIAM J.Num.Anal.* 15 (1978) 736-754.

[5] I. Babuška, W.C. Rheinboldt, A-posteriori error estimates for the finite element method, *Int.J.Num.Meth.Eng.* 12 (1978) 1597-1615.

[6] I. Babuška, M. Suri, The h-p version of the FEM: An overview, *Comp.Meth.Appl.Mech.Eng.* 80 (1990) 5-26.

[7] I. Babuška, R. Durán and R. Rodríguez, Analysis of the efficiency of an a posteriori error estimator for linear triangular finite elements, *SIAM J.Num.Anal.* 29 (1992) 947-964.

[8] R.E. Bank and A. Weiser, Some a posteriori error estimators for elliptic partial differential equations, *Math.Comp.* 44 (1985) 283-301.

[9] R.E. Bank, *PLTMG User's Guide - Ed. 5.0* (University of California at San Diego, 1988).

[10] E. Bänsch, Local mesh refinement in 2 and 3 dimensions, *Imp.Comp.Sci.Eng.* 3 (1991) 181-191.

[11] R. Becker, R. Rannacher, *A feed-back approach to error control in finite element methods: Basic analysis and examples* (Technical Report 359, Universität Heidelberg, 1996) 27p.

[12] J. Berrocal, Z. Kestřánek, J. Nedoma, Numerical modelling of tectonic evolution of the Wadati-Benioff zone beneath the Andean region, in: Michel Deville, Robert Owens, Eds., *Proc.16th IMACS World Congress on Scientif.Comp., Appl.Math. and Simulation* (EPFL, Lausanne, 2000) CD.

[13] R. Blaheta, R. Kohut, *Benchmarks and programs for testing iterative solvers for 3D problems of geomechanics.* (Report 9502, Dept. Appl. Math., Institute of Geonics, Ostrava, 1995).

[14] J. Céa, *Optimisation. Théorie et Algorithmes* (Dunod, Paris, 1970).

[15] Z. Dostál, Domain decomposition methods for modelling of deformable block structure, in: Z.Rakowski, Ed., *Proc.Int.Conf. Geomechanics'93* (Balkema Publishers, Rotterdam, 1994) 185-188.

[16] Z. Dostál, Duality based domain decomposition with proporting for the solution of free boundary problem, *J.Comp.Appl.Math.* 63 (1995) 203-208.

[17] R. Durán, M.A. Muschietti, R. Rodríguez, On the asymptotic exactness of error estimators for linear triangular elements, *Numer. Math.* 59 (1991) 107-127.

[18] G. Duvaut, J.L. Lions, *Inequalities in Mechanics and Physics* (Springer Verlag, Berlin, 1976).

[19] J. Dvořák, Some aspects of the numerical solution of 3D Stefan problem in hetergenous media, in: J.R. Whiteman, Ed., *MAFELAP 1996, Summaries of Papers* (BICOM, London, 1996) 38.

[20] C. Eck, S.A. Nazarov, W.L. Wendland, An Adaptive Method for Contact Problems,
`http://www.uni-stuttgart.de/ibs/program.html`, (1997).

[21] R. Fletcher, *Practical Methods of Optimization, Vol.2: Constrained Optimization* (John Wiley&Sons, New York, 1981).

[22] R. Glowinski, J.L. Lions, R. Tremolièrs, *Numerical Analysis of Variational Inequalities* (North Holland, Amsterdam, 1981).

[23] J. Haslinger, Finite element analysis for unilateral problems with obstacles on the boundary, *Appl.Math.* 22 (1977) 180-188.

[24] J. Haslinger, I. Hlaváček, Contact between elastic bodies. II. Finite element analysis, *Appl.Math.* 26 (1981) 263-290.

[25] J. Haslinger, M. Tvrdý, Approximation and numerical realization of contact problems with friction, *Appl.Math.* 28 (1983) 55-71.

[26] J. Haslinger, Approximation of the Signorini Problem with Friction, Obeying the Coulomb Law, *Math.Met.Appl.Sci.* 5 (1983) 422-437.

[27] I. Hlaváček, Convergence of dual finite element approximations for unilateral boundary value problems, *Appl.Math.* 25 (1980) 375-376.

[28] I. Hlaváček, J. Lovíšek, Finite element analysis of the Signorini problem in semicoercive cases, *Appl.Math.* 25 (1980) 273-285.

[29] I. Hlaváček, J. Haslinger, J. Nečas, J. Lovíšek, *Solution of variational inequalities in mechanics* (Alfa, Bratislava, 1982).

[30] J. Horák, H. Netuka, Numerical realization of contact problem with friction - semicoercive case, in: *Proc.Int.Conf. Mathematical Methods in Engineering* (Czech.Sci.Techn.Soc.Centr.Res.Inst.ŠKODA Concern, Plzeň, 1991) 147-152.

[31] D. Janovská, *Numerical solution of contact problems.* (Dissertation, MFF UK, Praha, 1980).

[32] J. Jarušek, Contact problems with friction - coercive case, *Czech.Math.J.* 33 (1983) 237-261.

[33] J. Jarušek, Contact problems with bounded friction. Semicoercive case, *Czech.Math.J.* 34 (1989) 619-624.

[34] C. Johnson and P. Hansbo, Adaptive finite element methods in computational mechanics, *Comp.Meth.Appl.Mech.Eng.* 101 (1992) 143-181.

[35] Z. Kestřánek, Comparison of methods for solving the contact problem in thermoelasticity, in: V. Kompiš, Ed., *Proc.Int.Conf. Numerical Methods in Continuum Mechanics* (Editional Centre VSDS, Žilina-Slovakia, 1995) 128-138.

[36] Z. Kestřánek, *Numerical Analysis of the Contact Problem. Comparison of Methods for Finding the Approximate Solution.* (Techn. Rep. V-648, ICS AS CR, Praha, 1995).

[37] Z. Kestřánek, Algorithms for contact problem with friction in thermoelasticity, in: J.R.Whiteman, Ed., *MAFELAP 1996, Summaries of Papers* (BICOM, London, 1996).

[38] Z. Kestřánek, J. Nedoma, *The Conjugate Projected Gradient Method - Numerical Tests and Results.* (Techn. Rep. V-677, ICS AS CR, Praha, 1996).

[39] Z. Kestřánek, J. Dvořák, J. Nedoma, Iterative Solvers for Coupled Contact-Stefan-like Problems. In: Achim Sydow, Ed., *Proc.15th IMACS World Congress on Scientif.Comp., Modell. and Appl.Math. Vol.3 Comp.Physics, Chem. and Biol.* (Berlin, 1997) 467-472.

[40] Z. Kestřánek and J. Nedoma, Numerical simulation of a bridge on a non-stable slope. Comparison of results based on contact problem without and with friction, in: Z.Rakowski, Ed., *Proc.Int.Conf. Geomechanics'96* (A.A.Balkema, Rotterdam, 1997) 169-174.

[41] Z. Kestřánek, Numerical Modelling of Some Particular Cases in Contact Problem. In: M. Feistauer, R. Rannacher and K. Kozel, Eds., *Proc. 3rd Summ.Conf. NMICM'97* (Praha, 1997) 321-328.

[42] Z. Kestřánek, Iterative Methods for Contact Problems, *Mathematics and Computers in Simulation* 50 (1999) 199-204.

[43] Z. Kestřánek, H-version of the FEM for the contact problem in elasticity, in: M. Griebel, S. Margenov and P. Yamalov, Eds., *Large-Scale Scientific Computations of Engineering and Environmental Problems II, Notes on Numerical Fluid Mechanics, Vol. 73* (Vieweg, Wiesbaden, 2000) 363-370.

[44] Z. Kestřánek, *H-version of the contact problem in thermoelasticity,* (in: Technical Report V-784, ICS AS CR, Prague, 1999) 40-44.

[45] Z. Kestřánek, *Numerical analysis of the 3D contact problem of Signorini type with friction in thermoelasticity. H-version of the finite element approximation,* (Ph.D. Thesis, Technical University Prague, 1999).

[46] Z. Kestřánek, h-version of the FEM for the contact problem in elasticity, in: Michel Deville, Robert Owens, Eds., *Proc.16th IMACS World Congress on Scientif.Comp., Appl.Math. and Simulation* (EPFL, Lausanne, 2000) CD.

[47] R. Kohut, *Software Modulus: Code for Implementation of Preconditioners.* (Techn. Rep. 9604, IGN AS CR, Ostrava, 1996).

[48] V. Kolář, J. Kratochvíl, F. Leitner, A. Ženíšek, *Výpočet plošných a prostorových konstrukcí metodou konečných prvků* (SNTL, Praha, 1979).

[49] R. Krejčí, *Preprocessing a postprocessing statických a dynamických úloh.* (Dipl.Thesis, MFF UK, Praha, 1995).

[50] R. Krejčí, J. Dvořák, J. Nedoma, On the 2D and 3D FEM Pre- and Post-Processing, in: J.R. Whiteman, Ed., *MAFELAP 1996, Summaries of Papers* (BICOM, London, 1996).

[51] S. Míka, J. Šulcová, Algoritmy sedlového bodu, in: M. Práger, P. Přikryl and K. Segeth, Eds., *Programy a algoritmy numerické matematiky* (MÚ ČSAV, Praha, 1990) 114-144.

[52] W.F. Mitchell, A Comparison of Adaptive Refinement Techniques for Elliptic Problems, *ACM Transactions on Mathematical Software* 15 (1989) 326-347.

[53] J. Nečas, I. Hlaváček, *Mathematical Theory of Elastic and Elasto-Plastic Bodies* (Elsevier, Amsterdam, 1981).

[54] J. Nečas, J. Jarušek, J. Haslinger, On the solution of the variational inequality to the Signorini problem with small friction, *Boll.Unione Mat.Ital.* 17-B (1980) 796-811.

[55] J. Nedoma, On one type of Signorini problem without friction in linear thermoelasticity, *Appl.Math.* 28 (1983) 393-407.

[56] J. Nedoma, On the Signorini problem with friction in linear thermoelasticity. Quasicoupled 2D case. *Appl.Math.* 32 (1987) 186-199.

[57] J. Nedoma, Plate tectonics and analysis of fracture mechanics in a collision zone. Part I. Formulation of the problem, *Contr.Geoph.Inst.Slov.Acad.Sci.* 20 (1990) 53-69.

[58] J. Nedoma, Plate tectonics and analysis of fracture mechanics in a collision zone. Part II. Crack mechanics, *Contr.Geoph.Inst.Slov.Acad.Sci.* 21 (1991) 38-57.

[59] J. Nedoma, Finite element analysis of contact problems in thermoelasticity. The semi-coercive case, *J.Comp.Appl.Math.* 50 (1994) 411-423.

[60] J. Nedoma, J. Dvořák, On the FEM solution of a coupled contact-two phase Stefan problem in thermoelasticity. Coercive case, *J.Comp.Appl.Math.* 63 (1995) 411-420.

[61] J. Nedoma, Numerical solutions of a coupled two-phase Stefan-contact problem with friction in thermoelasticity by variational inequalities, in: J.R. Whiteman, Ed., *MAFELAP 1996, Summaries of Papers* (BICOM, London, 1996) 101.

[62] J. Nedoma, *Numerical Modelling in Applied Geodynamics* (John Wiley & Sons, Chichester, New York, Weinheim, Brisbane, Singapore, Toronto, 1998).

[63] J. Nedoma, Z. Kestřánek, *Preconditioners for contact problems in elasticity* (Technical Report V-739, ICS AS CR, Prague, 1998) 16p.

[64] J.T. Oden, L. Demkowicz, W. Rachowicz and T.A. Westerman, Toward a Universal h-p Adaptive Finite Element Strategy, Part 2. A Posteriori Error Estimation, *Comp.Meth. Appl.Mech.Eng.* 77 (1989) 113-180.

[65] A. Plaza, M.A. Padrón, G.F. Carey, A 3D refinement/derefinement algorithm for solving evolution problems. In: Achim Sydow, Ed., *Proc.15th IMACS World Congress on Scientif.Comp., Modell. and Appl.Math. Vol.2 Num.Math.* (Berlin, 1997) 197-202.

[66] B.N. Pshenichnyj, J.M. Danilin, *Numerical Methods in Extremal Problems.* (Mir Publishers, Moscow, 1978).

[67] S.I. Repin, L.S. Xanthis, A Posteriori Error Estimation for Elasto-Plastic Problems Based on Duality Theory, *Comp.Meth. Appl.Mech.Eng.* 138 (1996) 317-339.

[68] M.C. Rivara, Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques, *Int.J.Num.Meth.Eng.* 20 (1984) 745-756.

[69] M.C. Rivara, Selective Refinement/Derefinement Algorithms for Sequences of Nested Triangulations, *Int.J.Num.Meth.Eng.* 28 (1989) 2889-2906.

[70] I.G. Rosenberg and F. Stenger, A Lower Bound on the Angles of Triangles Constructed by Bisecting the Longest Side, *Math.Comp.* 29 (1975) 390-395.

[71] P. Saint-Georges, G. Warzee, *Efficient Iterative Solution of Constrained FE Analyses.* (Techn.Rep. SMC96/03, Université Libre de Bruxelles, 1996).

[72] R. Verfürth, A review of a-posteriori error estimation and mesh-refinement techniques, *J.Comp.Appl.Math.* 50 (1994) 67-83.

[73] J.Z. Zhu, O.C. Zienkiewicz, Superconvergence recovery technique and a posteriori error estimates. Part I. The recovery technique, *Int.J.Num.Meth.Eng.* 33 (1992) 1331-1364.

[74] J.Z. Zhu, O.C. Zienkiewicz, Superconvergence recovery technique and a posteriori error estimates. Part II. Error estimates and adaptivity, *Int.J.Num.Meth.Eng.* 33 (1992) 1365-1382.

[75] O.C. Zienkiewicz, J.Z. Zhu, A simple error estimator and adaptive procedure for practical engineering analysis, *Int.J.Num.Meth.Eng.* 24 (1987) 337-357.

[76] O.C. Zienkiewicz, R.L. Taylor, *The Finite Element Method* (McGraw-Hill, London, 4th Edn., 1989).