Available online at www.sciencedirect.com
SCIENCE DIRECT
Finite Elements in Analysis and Design 42 (2006) 1053–1060
FINITE ELEMENTS IN ANALYSIS AND DESIGN
www.elsevier.com/locate/finel

Trimming of 3D solid finite element meshes using parametric surfaces: Application to sheet metal forming

A.J. Baptista[a,*], J.L. Alves[b], D.M. Rodrigues[a], L.F. Menezes[a]

[a]*Departamento de Engenharia Mecânica, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Pinhal de Marrocos-Polo II, 3030-788 Coimbra, Portugal*
[b]*Departamento de Engenharia Mecânica, Escola de Engenharia da Universidade do Minho, Campus de Azurém, 4800-058 Guimarães, Portugal*

Received 14 July 2005; received in revised form 25 January 2006; accepted 6 March 2006
Available online 18 April 2006

Abstract

In this paper we present a methodology for trimming 3D solid finite element meshes using a non-uniform rational B-spline (NURBS) surfaces representation. The algorithms applied in the identification of the spatial position of the finite element (FE) nodes and elements relative to the trimming surface are described, as well as, the explanation of the correction methods adopted to geometrically rearrange the trimmed elements. Three different strategies are proposed to adjust, with greater or lesser accuracy, the trimmed FE mesh to the trimming surface. The first consists only of an element elimination strategy, while the other two are based on a nodal stretching strategy, aiming for more accurate adjustment of the finite elements boundary to the trimming surface. Finally, to highlight the capabilities of the developed program, a mesh generation example is given by trimming a mesh with the shape of a blank automotive panel.

Keywords: Trimming; Cutting; Hexahedral meshes; Solid finite elements; NURBS

1. Introduction

In recent years the numerical simulation of multi-step deep-drawing processes has become a very important research field, in particular for the very strong automotive industry. Currently, great efforts are being taken towards the ambitious objective of entirely simulating all production stages of a deep-drawn part, including tasks between operations, to achieve so-called digital manufacturing [1,2].

To completely simulate a multi-stage deep-drawing process, it is necessary to numerically treat intermediate trimming operations, in which some part of the finite element mesh that models the blank sheet has to be eliminated. In the literature, few references are found that specifically treat the subject of trimming finite element meshes, particularly in the case of solid finite elements. One of the basic methods applied by some numerical codes is simply to eliminate the undesired finite elements from the FE mesh [1]. In case of methods where the trimmed FE mesh is adjusted to the trimming surface, the studies of Coelho et al. [3] for shell element meshes and Dhondt [4] for solid hexahedral meshes can be referenced. For the solid FE mesh case, Dhondt developed a method based on the subdivision of the affected (trimmed) elements, which usually produces a very significant increase in the number of finite elements added to the final trimmed mesh.

In this study, a new approach has been developed for the geometrical rearrangement of the trimmed solid brick elements. When the most accurate solution is chosen, those elements are modified by a selective adjustment of the spatial position of some nodes, in a so-called stretching node technique. The trimming algorithm was implemented in a code named DD3TRIM, which also includes specific remapping strategies for the variable transfer problem between meshes [5].

The use of non-uniform rational B-spline (NURBS) has nowadays become the most popular representation method in CAD/CAM systems, due to its generality and excellent properties [6]. This geometric modeller was originally introduced in the mid seventies and had a rapid proliferation for all sort of application fields, such as industrial design, electrical and mechanical engineering, biomedicine, etc. [7]. Part of the enormous success of this kind of parametric representation can

* Corresponding author. Tel.: +351 239 790700; fax: +351 239 790701.
E-mail address: antonio.baptista@dem.uc.pt (A.J. Baptista).

0168-874X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.finel.2006.03.005

also be justified by the fact that NURBS has been the major geometric element in international product data transfer standards since it was included, for instance, in STEP (standard for the exchange of product model data) and IGES (initial graphics exchange specification) file formats [7,8].

The application field of a numerical tool like DD3TRIM cannot just be used as a trimming tool for finite element meshes in the intermediate steps of a standard multi-stage deep-drawing process. Actually, it can be easily applied with important advantages, in the pre-processing stages of numerical simulations, for mesh generation purposes. This topic was the main subject of Dhondt's work, which developed an automatic and universal hexahedral FE mesher based on a trimming procedure starting from a given uniform and structured FE mesh [4]. It is nowadays well established that the usage of solid hexahedral elements in sheet metal forming simulations can be very difficult and time consuming when meshing and remeshing operations are not automatically performed. Nevertheless, the solution of starting from a structured and regular initial mesh, and subsequently removing the excess volume to obtain the final mesh model, can be both straightforward and efficient (from the regular element size point of view). In particular, the treatment algorithm of DD3TRIM does not strictly require the addition of elements for adjusting the trimmed elements towards the surface geometry, therefore saving computation time during the simulation stages.

The implemented algorithms were developed considering the geometry of the brick element and the particularities of the simulation of the deep-drawing process, in which thin sheet metal blanks are modelled. Thus, it is considered that the trimming surface dimensions are several times greater than the finite element dimensions themselves. Furthermore, the trimming surface should not have an orientation far from the perpendicular direction to the blank plane.

The remainder of this document is divided in two sections. In the first one, a brief mathematical description of NURBS is given and the trimming algorithm is detailed, describing the three main parts of the code: pre-processing, correction and post-processing. In the second section, a typical automotive example of trimming a solid finite element mesh is given. The chosen example highlights the advantages of using such a type of trimming algorithm associated to a NURBS representation, as a method to easily construct complex mesh geometries for deep-drawing simulation.

2. The trimming algorithm

Since in this study the trimming surface is defined by means of a NURBS representation, a brief review is given of its mathematic characterization and two basic operations: the projection of a point onto a surface and the intersection of a line with a surface. The notation convention used is the common one, with italic letters representing scalars and bold letters representing vectors or tensors.

2.1. NURBS surface characterization

A NURBS surface can be defined as the rational generalization of a tensor-product of a nonrational B-Spline surface defined as [9]

$$\mathbf{S}(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m}N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \tag{1}$$

where $\mathbf{P}_{i,j}$ are the $(n+1)\times(m+1)$ 3D control points that form a bi-directional control net, also known as the *control polygon* of the surface; $w_{i,j}$ are the related weights of every control point; $N_{i,p}(u)$ and $N_{j,q}(v)$ are the normalized B-spline basis functions or blending functions of degree p and q, associated with every node i and j of the *control polygon*, respectively. These functions can be calculated recursively. For the u-direction case, the formula is

$$N_{i,p}(u)=\begin{cases}1, & u_i\leqslant u\leqslant u_{i+1},\\ 0 & \text{otherwise},\end{cases}$$

$$N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u). \tag{2}$$

In these equations $\mathbf{U}=\{u_0,\ldots,u_{m_u}\}$ and $\mathbf{V}=\{v_0,\ldots,v_{m_v}\}$ are non-decreasing sequences of real numbers, where u_i and v_j are called knots, while $\mathbf{U}$ and $\mathbf{V}$ represent the knot vectors for the u and v directions, respectively. If one considers, for instance, the u-direction case, the mathematical degree (p), the number of knots (m_u+1) and the number of control points ($n+1$) are related by $m_u=n+p+1$. For the v-direction the formula is $m_v=m+q+1$.

In order to mathematically operate arbitrary 3D geometrical entities (points, lines, etc.) and 3D NURBS surfaces (such as to project points or to intersect lines with a NURBS surface), it is necessary to define and formulate the partial derivative equations for a generic point (u,v) of the surface: $\mathbf{S}_u(u,v)$, $\mathbf{S}_v(u,v)$, $\mathbf{S}_{uu}(u,v)$, $\mathbf{S}_{vv}(u,v)$ and $\mathbf{S}_{uv}(u,v)$. As an example, the first and second partial derivatives of $\mathbf{S}(u,v)$ with respect to u are given by [10]

$$\mathbf{S}_u(u,v)=\frac{\partial\mathbf{S}(u,v)}{\partial u}=\sum_{i=0}^{n}\sum_{j=0}^{m}\left(\frac{\partial R_{i,j}(u,v)}{\partial u}\right)\mathbf{P}_{i,j}, \tag{3}$$

$$\mathbf{S}_{uu}(u,v)=\frac{\partial^2\mathbf{S}(u,v)}{\partial u^2}=\sum_{i=0}^{n}\sum_{j=0}^{m}\left(\frac{\partial^2 R_{i,j}(u,v)}{\partial u^2}\right)\mathbf{P}_{i,j}. \tag{4}$$

Since all the formulas to the v-direction have similarities or can be simply deduced from the u-direction case, they will be omitted. For the mixed partial derivative, the equation is [10]

$$\mathbf{S}_{uv}(u,v)=\frac{\partial^2\mathbf{S}(u,v)}{\partial v\partial u}=\sum_{i=0}^{n}\sum_{j=0}^{m}\left(\frac{\partial^2 R_{i,j}(u,v)}{\partial v\partial u}\right)\mathbf{P}_{i,j}. \tag{5}$$

2.1.1. Projection of a point on a NURBS surface

The projection of a given point on a NURBS surface is a basic geometrical operation when dealing with 3D NURBS surfaces. In this investigation, the method followed is based on the review made by Stadler et al. [11].

Let us consider that a generic point $\mathbf{P}(x, y, z)$ is to be projected onto the surface $\mathbf{S}(u, v)$. The first steps are to determine the vector $\mathbf{r}(u, v)$ from an arbitrary point of $\mathbf{S}(u, v)$ to the generic point $\mathbf{P}(x, y, z)$,

$$\mathbf{r}(u, v) = \mathbf{S}(u, v) - \mathbf{P}(x, y, z) \tag{6}$$

and the partial derivative vectors $\mathbf{S}_u(u, v)$ and $\mathbf{S}_v(u, v)$. After this, the orthogonality condition is imposed based on the dot product functions

$$\begin{aligned} f(u, v) &= \mathbf{S}_u(u, v) \cdot \mathbf{r}(u, v), \\ g(u, v) &= \mathbf{S}_v(u, v) \cdot \mathbf{r}(u, v). \end{aligned} \tag{7}$$

Therefore, the problem of projecting a generic point on a NURBS surface is reduced to solving a nonlinear system of equations, given by

$$\begin{aligned} f(u, v) &= 0, \\ g(u, v) &= 0. \end{aligned} \tag{8}$$

A possible way to solve the previous problem is, for instance, by applying the Newton–Raphson iterative method. In this case one can write

$$\mathbf{X}^{(i+1)} = \mathbf{X}^{(i)} - [\mathbf{J}^{(i)}]^{-1}\mathbf{F}(\mathbf{X}^{(i)}), \tag{9}$$

where, for the iteration i,

$$\mathbf{F}(\mathbf{X}^{(i)}) = \begin{bmatrix} f(\mathbf{X}^{(i)}) \\ g(\mathbf{X}^{(i)}) \end{bmatrix}, \quad \mathbf{X}^{(i)} = \begin{bmatrix} u^{(i)} \\ v^{(i)} \end{bmatrix} \tag{10}$$

being $\mathbf{J}^{(i)}$ the Jacobi matrix of $\mathbf{F}(\mathbf{X})$ at $\mathbf{X}^{(i)}$:

$$\begin{aligned} \mathbf{J}^{(i)} &= \begin{bmatrix} \dfrac{\partial f}{\partial u} & \dfrac{\partial f}{\partial v} \\ \dfrac{\partial g}{\partial u} & \dfrac{\partial g}{\partial v} \end{bmatrix}^{(i)} \\ &= \begin{bmatrix} |\mathbf{S}_u|^2 + \mathbf{r} \cdot \mathbf{S}_{uu} & \mathbf{S}_u \cdot \mathbf{S}_v + \mathbf{r} \cdot \mathbf{S}_{uv} \\ \mathbf{S}_u \cdot \mathbf{S}_v + \mathbf{r} \cdot \mathbf{S}_{uv} & |\mathbf{S}_v|^2 + \mathbf{r} \cdot \mathbf{S}_{vv} \end{bmatrix}^{(i)}. \end{aligned} \tag{11}$$

The convergence rate of the proposed iterative procedure, or even the achievement of a solution, are greatly affected by the initial solution $\mathbf{X}^{(0)}$, as well as the surface's dimensions and its particular geometrical characteristics. To obtain a faster convergence rate it is assigned as initial solution the closest knot of the surface to the point $\mathbf{P}$.

2.1.2. Intersection of a line with a NURBS surface

The so-called ray-patch intersection, i.e., the identification of the intersection point between a generic line and a NURBS surface, can in some cases be a difficult and very time consuming problem to solve, since it is greatly dependent on the initial solution if an iterative procedure is used [12].

However, in the particular case of trimming solid finite elements, once the FE edges (virtual line that connects two nodes) are clearly defined, the identification of a good initial solution is straightforward, and thus, the classical iterative Newton–Raphson method can be easily applied to solving the problem of the intersection between a line (the trimmed edge of a FE) and a NURBS surface (the trimming surface).

Let us consider that r is the line that intersects the trimming surface, defined by the point $\mathbf{A}(a_1, a_2, a_3)$ and the vector $\mathbf{v}(v_1, v_2, v_3)$. The intersection point between r and $\mathbf{S}(u, v)$ can be determined by solving the following nonlinear system of equations:

$$\begin{aligned} S^x(u, v) &= a_1 + kv_1, \\ S^y(u, v) &= a_2 + kv_2, \\ S^z(u, v) &= a_3 + kv_3, \quad k \in \mathbb{R}. \end{aligned} \tag{12}$$

Rearranging the previous equations in order to determine their roots (solution vector (u, v, k), which allows one to identify the intersection point), the system of equations can be rewritten as

$$\begin{aligned} f(u, v, k) &= S^x(u, v) - (a_1 + kv_1) = 0, \\ g(u, v, k) &= S^y(u, v) - (a_2 + kv_2) = 0, \\ h(u, v, k) &= S^z(u, v) - (a_3 + kv_3) = 0. \end{aligned} \tag{13}$$

Finally, the Jacobi matrix $\mathbf{J}$ is given by

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f}{\partial u} & \dfrac{\partial f}{\partial v} & \dfrac{\partial f}{\partial k} \\ \dfrac{\partial g}{\partial u} & \dfrac{\partial g}{\partial v} & \dfrac{\partial g}{\partial k} \\ \dfrac{\partial h}{\partial u} & \dfrac{\partial h}{\partial v} & \dfrac{\partial h}{\partial k} \end{bmatrix} = \begin{bmatrix} S_u^x(u, v) & S_v^x(u, v) & -v_1 \\ S_u^y(u, v) & S_v^y(u, v) & -v_2 \\ S_u^z(u, v) & S_v^z(u, v) & -v_3 \end{bmatrix}. \tag{14}$$

2.2. Pre-processing stage

The pre-processing stage begins by reading the mesh file (using a standard finite element mesh file format), the trimming NURBS surface (by means of an IGES file) and a third file where the trimming options are defined (trimming strategy, type of correction/adjustment to be applied to the trimmed finite elements).

Additionally, some geometrical information, related to the 8-node solid FE mesh (Fig. 1) is internally generated and stored in the form of connectivity tables for nodes ($\mathbf{N}_i$), edges ($\mathbf{A}_i$) and facets ($\mathbf{F}_i$).

After allocating all the necessary connectivity tables, a procedure is followed in order to identify and store the relative positions of both nodes and elements to the trimming surface oriented with the normal $\mathbf{n}$ (Fig. 2). Such information is stored in two status tables. For nodes, the $\text{STATUS}^{(\text{nodes})}$ table is generated, and for each node one of the following three statuses is assigned: "*keep*", "*eliminate*" or "*on surface*". For the elements, the $\text{STATUS}^{(\text{elements})}$ table is created and for each element one of the following statuses is assigned: "*to treat*", "*keep*" or "*eliminate*" (Fig. 3).

To determine the status of each node, an algorithm based on the inner product calculus between two vectors normal to the NURBS surface is used. The first step is to project a given node $\mathbf{N}_i$ (see Fig. 4) onto the trimming surface, and subsequently to

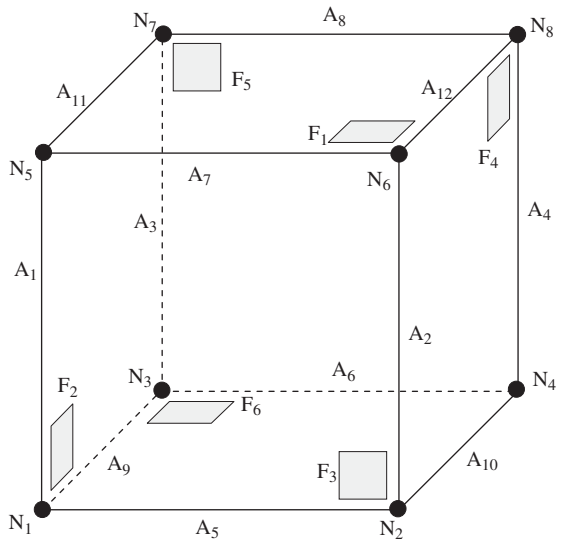

Fig. 1. Nodes, edges and facets connectivity for the 8-node brick element.

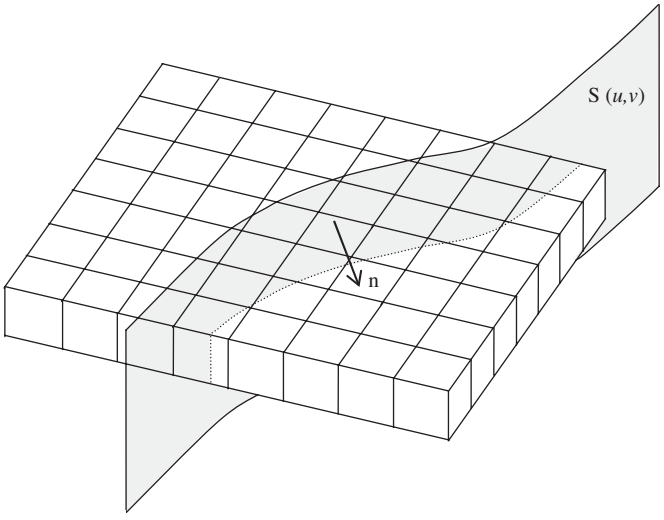

Fig. 2. Initial mesh to be trimmed by the NURBS surface.

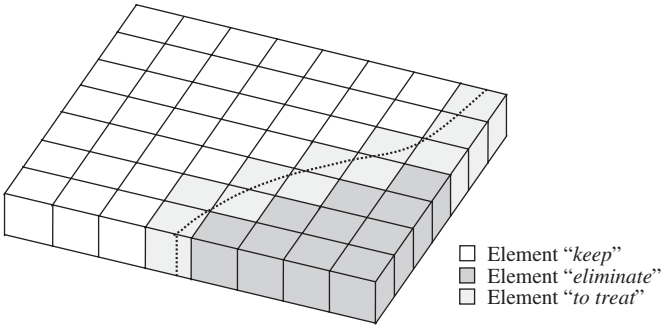

Fig. 3. Finite element statuses according to the trimming surface position.

compute the vector ($\mathbf{n}_i$) normal to the surface at the projection point $\mathbf{P}_i$:

$$\mathbf{n}_i^{(\mathbf{P}_i)} = \mathbf{S}_u \otimes \mathbf{S}_v. \tag{15}$$

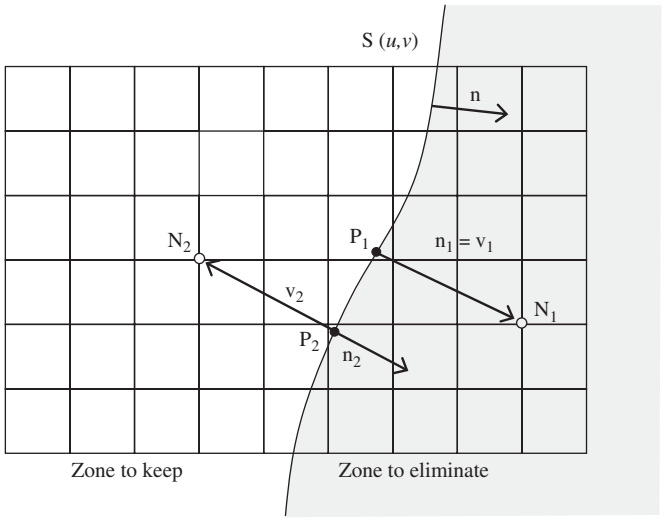

Fig. 4. Schematic 2D representation of the statuses determination for nodes.

The second vector ($\mathbf{v}_i$) is the one that connects the projection point $\mathbf{P}_i$ to the nodal position $\mathbf{N}_i$. Since the surface normal $\mathbf{n}$ is previously oriented towards the side of the mesh to be eliminated, the position and status of any node relative to the trimming surface can easily be identified by evaluating the value of the inner product between $\mathbf{n}_i$ and $\mathbf{v}_i$: if $\mathbf{n}_i \cdot \mathbf{v}_i < 0$, then $\text{STATUS}^{(\text{nodes})}$ is "*keep*"; otherwise, if $\mathbf{n}_i \cdot \mathbf{v}_i > 0$, then $\text{STATUS}^{(\text{nodes})}$ is "*eliminate*". The status "*on surface*" is assigned if the Euclidean distance between point $\mathbf{P}_i$ and node $\mathbf{N}_i$ is close to zero.

The $\text{STATUS}^{(\text{elements})}$ table is filled-in by summing the nodes with "*eliminate*" status for each finite element: if the number of nodes of the element to be eliminated is equal to 8, all the nodes of the element are on the side of the mesh to be eliminated, and consequently the assigned status is "*eliminate*"; otherwise, if no node of the element has "*eliminate*" status, then the element status is "*keep*"; finally, for cases where there are both nodes to eliminate and to keep in the same finite element, it is assigned the status "*to treat*".

2.3. Correction stage

At the correction stage each element with "*to treat*" status is again verified to settle if the element is to be eliminated or kept in the mesh. The decision is based on the percentage of element that has to be eliminated (V^{e}). The value of V^{e} is estimated by subdividing the hexahedral element into six tetrahedrons and then calculating, according to the nodes statuses and intersections of the edges with the trimming surface, the volume to be eliminated in each tetrahedron. If $V^{\text{e}} \leqslant 50\%$, the element is assigned with "*keep*" status, otherwise, if $V^{\text{e}} > 50\%$, the element is assigned with "*eliminate*" status.

Three different correction strategies were implemented in the DD3TRIM code: one based merely on element elimination and two others where the trimming geometry is accurately described in the final "new" mesh.

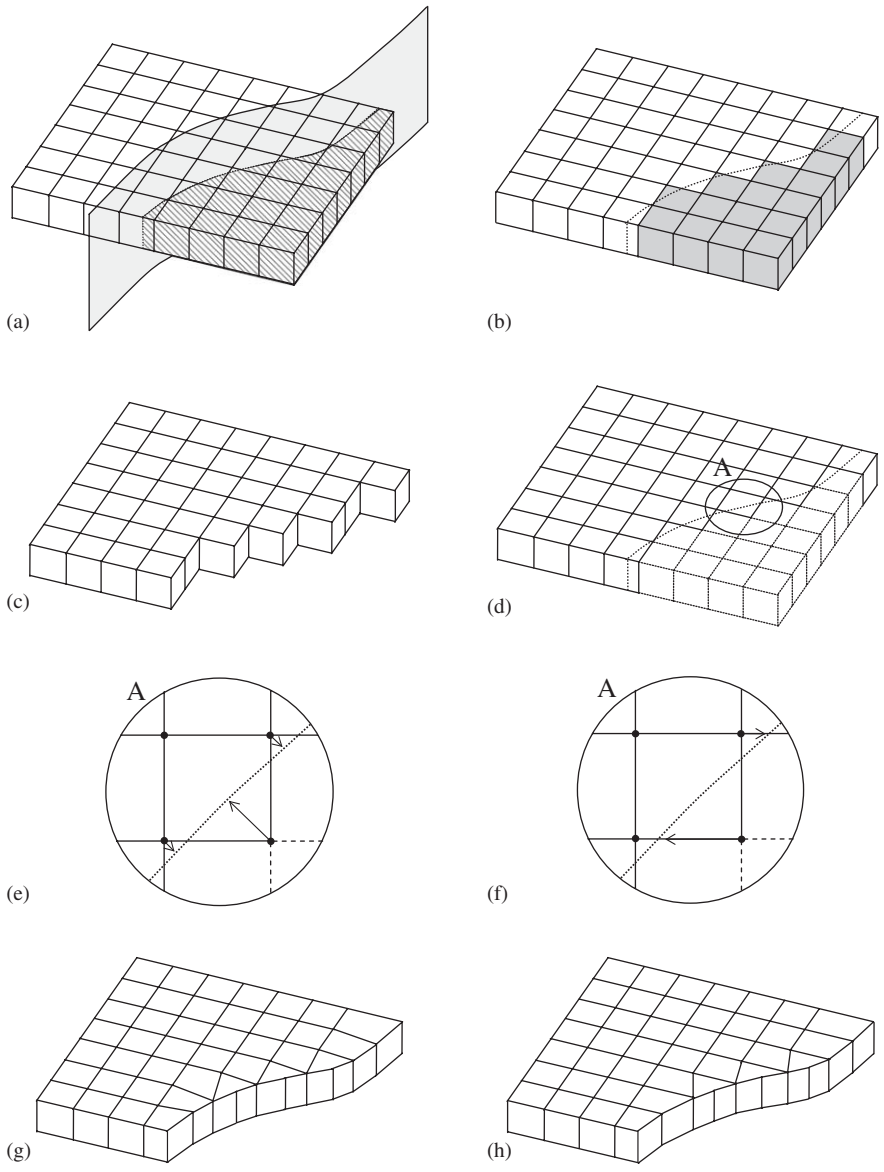

Fig. 5. Representation of trimming procedure and nodal correction strategies: (a) untrimmed mesh and trimming surface; (b) elements to eliminate (in grey); (c) final mesh using correction type I; (d) mesh before node adjustment; (e) node adjust for correction type II; (f) node adjust for correction type III; (g) final mesh, using correction type II; (h) final mesh, using correction type III.

To give a clearer picture of the implemented correction strategies, the main correction steps are sketched in Fig. 5, from the initial untrimmed mesh (Fig. 5a), the element status evaluation (Fig. 5b) and the node adjusting methods (Figs. 5d–f) until the final trimmed mesh (Figs. 5c,g,h).

Using correction type I all the nodes and elements with "*eliminate*" status are suppressed from the mesh, creating a lacy mesh boundary as shown in Fig. 5c. In order to precisely adjust the trimmed elements to the trimming surface two strategies are implemented, by means of correction types II and III.

In correction type II, the nodes that have to be moved from their original positions are projected onto the NURBS surface. Nevertheless, to assure that the domain of the mesh is kept in the thickness direction, additional corrections are taken to determine the new node position. The procedure starts by projecting the node to be moved, see for instance $\mathbf{N}_6$ of Fig. 6a, in $\mathbf{S}(u, v)$, according the previously presented formulation (Eqs. (6) and (7)). Thus, point $\mathbf{P}_1$ is found, such that

$$\mathbf{P}_1 = \mathrm{proj}_{\mathbf{S}(u,v)}\mathbf{N}_6. \tag{16}$$

Subsequently $\mathbf{P}_1$ is projected onto the element facet $\{\mathbf{N}_5, \mathbf{N}_6, \mathbf{N}_7, \mathbf{N}_8\}$ producing a new position $\mathbf{P}_2$ (Fig. 6b). Finally, the intersection of the line defined by $\mathbf{N}_6$ and $\mathbf{P}_2$ with the trimming surface $\mathbf{S}(u, v)$ is determined, to obtain point $\mathbf{P}_3$

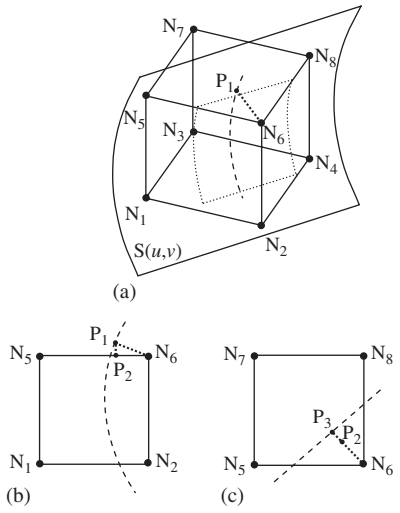

Fig. 6. Node adjustments applied in correction type II: (a) projection of node $\mathbf{N}_6$ in the surface (point $\mathbf{P}_1$); (b) projection of point $\mathbf{P}_1$ on element facet $\{\mathbf{N}_5, \mathbf{N}_6, \mathbf{N}_7, \mathbf{N}_8\}$ to create $\mathbf{P}_2$ (2D view); (c) intersection of line $\mathbf{N}_6\mathbf{P}_2$ with $\mathbf{S}(u, v)$ to create $\mathbf{P}_3$ (2D view).

(Fig. 6c), which will be the new position of the node $\mathbf{N}_6$:

$$S^x(u, v) = N_6^x + k(P_2^x - N_6^x),$$
$$S^y(u, v) = N_6^y + k(P_2^y - N_6^y),$$
$$S^z(u, v) = N_6^z + k(P_2^z - N_6^z), \quad k \in \mathbb{R}. \tag{17}$$

A variant to the abovementioned type II procedure could be adopted. This procedure would consist of the projection of node $\mathbf{N}_6$ on the NURBS curve defined by the intersection between the trimming surface and finite element facet $\{\mathbf{N}_5, \mathbf{N}_6, \mathbf{N}_7, \mathbf{N}_8\}$. However, the analysis made showed that this solution was more complex and time consuming, leading to results that are quite similar to the previous ones.

In correction type III, the new position of node $\mathbf{N}_6$ is obtained by the intersection of the finite element edge with the trimming surface, for a predetermined edge correction direction. Thus, if the predetermined direction is, for instance $\overrightarrow{\mathbf{N}_5\mathbf{N}_6}$, the problem relies on solving the following system of equations:

$$S^x(u, v) = N_6^x + k(N_6^x - N_5^x),$$
$$S^y(u, v) = N_6^y + k(N_6^y - N_5^y),$$
$$S^z(u, v) = N_6^z + k(N_6^z - N_5^z), \quad k \in \mathbb{R}. \tag{18}$$

Even when correction type II is chosen to rearrange the node position with respect to the trimming surface, correction type III is always applied to the nodes that define the boundary of the model, in order to preserve the border surface continuity.

2.4. Post-processing stage

In the post-processing stage a new mesh file containing the tables for nodes coordinates and finite element connectivity is sequentially constructed by selecting the nodes and elements that were assigned with "*keep*" status.

Node-numeration can have a great impact on the pattern of the sparse matrices used to solve the systems of linear equations of a simulation and, in this way, on the computation time. Thus, it is necessary to optimize node-numeration. The optimization algorithm used consists of minimizing for all the elements, the maximum value of the difference between the maximum node number and the minimum node number of each element.

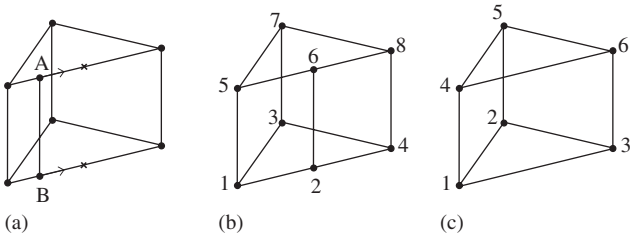

Fig. 7. Optimization of the pentahedral element shape: (a) node moving operation; (b) element after node distribution; (c) degenerated element.

During the correction stage, according to the trimming surface orientation, the re-positioning of the adjusted nodes may generate elements with pentahedral shapes (see for instance Figs. 5e and 5g). To minimize the distortion of these elements, a local optimization treatment is performed, that consists of moving the nodes (see nodes A and B of the exterior face of the element in Fig. 7a) to the middle edge position (Fig. 7b). Another option that can be applied is to degenerate the element (Fig. 7c), transforming the standard eight-node brick element into a six-node element, but keeping the eight-node connectivity by repeating some nodes.

Despite these distorted elements belonging only to the border of the mesh, their numerical behaviour during the subsequent simulation stages can somehow affect the overall results. To investigate and evaluate this fact, a study was conducted by simulating simple mechanical tests (a uniaxial tensile test and a simple bending test) using trimmed specimens meshes [13]. Several orientations of the trimming surfaces were used over a regular mesh in order to obtain different element shapes. Based on these mechanical tests simulations and other practical examples, it can be stated that the element distortion imposed by the proposed local trimming treatment is minor, when considering its impact on the element response behaviour with respect to the solicitations that occur in a standard deformation process simulation.

The general algorithm of DD3TRIM code, as described in the above discussion, is displayed in Fig. 8.

3. Trimming example

Figs. 9a and b show a typical automotive trimming application, where a mesh is trimmed to be subsequently used in the numerical simulation of the deep-drawing of a side panel. Four trimming operations are performed on a regular size base mesh (Fig. 9a). Firstly, an outside trim is carried out to remove the elements which exceed the main design geometry. Secondly, three holes are opened in the zones where after the deep-drawing operation, cavities such as doors or wheel arches will be placed (Fig. 9b). Correction method type II was used to execute all the trimming operations, with the pentahedral

GENERAL PROCEDURE TRIMMING

BEGIN

- Pre-Processing Stage -

- Read mesh, trimming surface and correction type option data
- Generate connectivity of Nodes, Edges and Facets
- Generate tables containing the status of nodes and elements:

  STATUS $^{(node)}$ = {*keep; eliminate; on surface*}

  STATUS $^{(element)}$ = {*to treat; keep; eliminate*}

- Correction Stage -

DO for all the elements with status: STATUS $^{(element)}$ = {*to treat*}

- Calculate the volume of the element to be eliminated

  IF $V^{eliminate} > 50\ \%$ **THEN** STATUS $^{(element)}$ = {*eliminate*}

  IF $V^{eliminate} \leq 50\ \%$ **THEN** STATUS $^{(element)}$ = {*keep*}

- Type of correction to apply:

  CASE 1 Simply eliminate the nodes and elements with status {*eliminate*}

  CASE 2 Project nodes in the normal direction to trimming surface and apply corrections towards the element facet

  CASE 3 Project nodes along element edge direction

- Project the affected nodes towards the trimming surface:

  IF STATUS $^{(element)}$ = {*eliminate*} **THEN** move node with STATUS $^{(node)}$ = {*keep*}

  IF STATUS $^{(element)}$ = {*keep*} **THEN** move node with STATUS $^{(node)}$ = {*eliminate*}

END DO

- Post-Processing Stage -

- Update nodal coordinates and connectivity tables
- Correct elements with pentahedral shapes:

  CASE 1 Redistribute the border nodes to the middle of the exterior face

  CASE 2 Degenerate the element

END

Fig. 8. DD3TRIM trimming algorithm.

element shapes being locally optimized by the node distribution scheme.

In this example it was assumed that the NURBS surface dimensions were much greater than the mean element ones. Furthermore, the trimming geometry does not include strong curvature variations such as, for instance, strong convex and/or concave profiles. In such conditions the DD3TRIM program should be carefully used and, if necessary, a previous refinement (local or generalized) operation must be performed to avoid large element distortions or even rendering impossible any description of the desired trimming geometry.

4. Conclusions

This paper presents an algorithm for performing trimming operations in solid finite element meshes composed of 8-node brick elements. The trimming surface is represented by means of NURBS definition. One of the greatest achievements of such a numerical tool, when associated with a deep-drawing simulation code, is to allow the execution of multi-step deep-drawing simulations (where parts of the mesh that models the blank sheet can be removed). Nevertheless, the DD3TRIM code can be used as a meshing tool, in order to produce meshes to

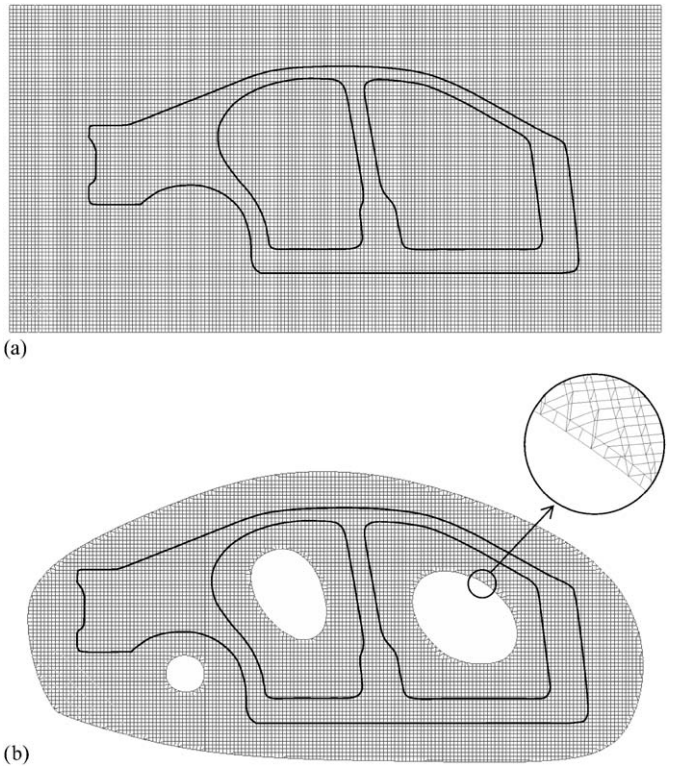

Fig. 9. Standard example of trimming a mesh using a free surface geometry: (a) base mesh and the desired final panel geometry; (b) trimmed mesh using correction method type II.

initiate the production simulation sequence. This last issue is very relevant, since this ability can solve several problems related to the production of complex mesh domains. The implemented algorithms prove to be robust and efficient in the several tests conducted to trim meshes with difficult geometry designs.

Acknowledgements

The authors are grateful to the Portuguese Foundation for Science and Technology (FCT) who financially supported this work, through the Program POCTI (Portuguese Government and FEDER).

References

[1] M. Kawka, T. Kakita, A. Makinouchi, Simulation of multi-step sheet metal forming process by a static explicit FEM code, J. Mat. Proc. Tech. 80–81 (1998) 54–59.
[2] T.H. Choi, S. Choi, K.H. Na, H.S. Bae, W.J. Chung, Application of intelligent design support system for multi-step deep drawing process, J. Mat. Proc. Tech. 130–131 (2002) 76–88.
[3] L.C. Coelho, M. Gattass, L.H. Figueiredo, Intersecting and trimming parametric meshes on finite element shells, Int. J. Numer. Meth. Eng. 47 (2000) 777–800.
[4] G. Dhondt, A new automatic hexahedral mesher based on cutting, Int. J. Numer. Meth. Eng. 50 (2001) 2109–2126.
[5] A.J. Baptista, J.L. Alves, M.C. Oliveira, D.M. Rodrigues, L.F. Menezes, in: J. Cao, M.F. Shi, T.B. Stoughton, C.T. Wang, L. Zhang (Eds.), Application of the Incremental Volumetric Remapping Method in the Simulation of Multi-Step Deep Drawing Processes, NUMISHEET'2005, On the Cutting Edge of Technology, American Institute of Physics, CP 778, vol. A, 2005, pp. 173–178.
[6] C.C.L. Wang, Y. Wang, M.M.F. Yuen, On increasing the developability of a trimmed NURBS surface 2004, Eng. Comp. 20 (2004) 54–64.
[7] E. Dimas, D. Briassoulis, 3D geometric modelling based on NURBS: a review, Adv. Eng. Software 30 (1999) 741–751.
[8] L. Piegl, On NURBS: a survey, IEEE Comput. Graph. Appl. 17 (1991) 55–71.
[9] L. Piegl, W. Tiller, The NURBS Book, Springer, London, 1995.
[10] M.-C. Tsai, C.-W. Cheng, M.-Y. Cheng, A real-time NURBS surface interpolator for precision three-axis CNC machining, Int. J. Mach. Tools Manuf. 43 (2003) 1217–1227.
[11] M. Stadler, G.A. Holzapfel, J. Korelc, C^n-continuous modelling of smooth contact surfaces using NURBS and application to 2D problems, Int. J. Numer. Meth. Eng. 57 (2003) 2177–2203.
[12] K. Qin, M. Gong, Y. Guan, W. Wang, A new method for speeding up ray tracing NURBS surfaces, Comput. Graph. 31 (5) (1997) 577–586.
[13] A.J. Baptista, J.L. Alves, M.C. Oliveira, D.M. Rodrigues, L.F. Menezes, Trimming of 3D solid elements in numerical simulation of sheet metal forming: tests and applications, Comput. Struct., submitted for publication.