**Engineering with Computers manuscript No.**
(will be inserted by the editor)

# Comparison of the meccano method with standard mesh generation techniques

**J.M. Cascón** ·
**E. Rodríguez** ·
**J.M. Escobar** ·
**R. Montenegro**

July 9, 2013

**Abstract** The meccano method is a novel and promising mesh generation technique for simultaneously creating adaptive tetrahedral meshes and volume parameterizations of a complex solid. The method combines several former procedures: a mapping from the meccano boundary to the solid surface, a 3-D local refinement algorithm and a simultaneous mesh untangling and smoothing. In this paper we present the main advantages of our method against other standard mesh generation techniques. We show that our method constructs meshes that can be locally refined by using the Kossaczky bisection rule and maintaining a high mesh quality. Finally, we generate volume T-mesh for isogeometric analysis, based on the volume parameterization obtained by the method.

**Keywords** Tetrahedral mesh generation · Adaptive refinement · Nested meshes · Mesh untangling and smoothing · Surface and volume parameterization

## 1 Introduction

In the last decades, tetrahedral mesh generation has been a subject of intense research. The automatic and adaptive mesh generation is a crucial objective in finite element applications [5, 24, 25, 39, 41, 42, 44]. The challenge is to achieve high quality adaptive meshes of complex solids with minimal user intervention and low computational cost. On the one hand, most mesh generators are based on Delaunay triangulation and advancing front. Althought these techniques are widely extended, there is still some problem related to mesh quality and mesh conformity with the boundary of complex geometries [8]. If high element quality and mesh adaption is required, Steiner points are introduced for boundary recovery and an appropriate definition of element sizes is demanded. On the other hand, local adaptive refinement is a fundamental tool to adapt the mesh to singularities of numerical solution. For this purpose, the main strategies are based on remeshing or nested refinement.

We have recently introduced the meccano technique [6, 7, 35, 36] for constructing adaptive tetrahedral meshes of solids. The name of the method stems from the fact that the process starts from an outline of the solid composed by connected polyhedral pieces: a meccano. The main idea of the method is to build a volume mesh of the solid as a deformation of an appropriate tetrahedral mesh of the meccano.

Our strategy uses no Delaunay triangulation, nor advancing front technique, and it simplifies the geometrical discretization problem for three-dimensional complex domains, whose surfaces can be mapped to the meccano faces. The meccano method combines a local refinement of tetrahedral nested meshes [28], a parameterization of surface triangulations [20] and our simultaneous untangling and smoothing procedure [11, 13].

The meccano method requires a one-to-one mapping between the meccano boundary to the solid boundary. Mappings between physical and parametric spaces have been analyzed by several authors. Significant advances in surface parameterization have been done in [20, 21, 18, 29, 43, 46]. In [36] we present an automatic parameterization of a genus-zero solid surface triangulation to a cube boundary.

Although surface parameterization has been extensively studied in the literature, there are only a few works addressing the volume parameterization and they have in common the use of harmonic functions [31, 30, 33, 32, 46]. In addition, Floater et al [19] give a simple counterexample to show that convex combination mappings over tetrahedral meshes are not necessarily one-to-one. A crucial consequence of our method is the volume parameterization of a solid to the meccano. In [36, 15], we present an automatic procedure for the volume parameterization of a genus-zero solid. A similar work is proposed by Zhang in [49] and it is extended to arbitrary genus topology in [47].

Mesh optimization is the key for keeping mesh shape regularity and for avoiding a costly remeshing [26, 27]. In traditional mesh optimization, mesh moving is guided by the minimization of certain overall functions, but it is usually done in a local fashion. In general, this procedure involves two steps [23, 22]: the first is for mesh untangling and the second one for mesh smoothing. Each step leads to a dif-

J.M. Cascón
Department of Economics and History of Economics, Faculty of Sciences, University of Salamanca, Spain. E-mail: casbar@usal.es

J.M. Escobar · R. Montenegro · E. Rodríguez
University Institute for Intelligent Systems and Numerical Applications in Engineering (SIANI), University of Las Palmas de Gran Canaria, Spain
E-mail: {jmescobar,rmontenegro,erodriguez}@siani.es

ferent objective function. The meccano method uses the improvement proposed by [11, 13, 12, 16], where a simultaneous untangling and smoothing guided by the same objective function is introduced.

In this paper we analyze three advantages of the meccano method. Firstly, we compare our meccano meshes with triangulations generated by other well-known codes: TetGen [41, 42] and NETGEN [39]. Our procedure demands more CPU time than TetGen but it is competitive compared with NETGEN. Moreover the meccano meshes have higher quality than the other ones. Secondly, we solve adaptively a heat transfer problem starting from meccano meshes. The structure of our meshes allows us to refine and derefine locally by applying the simple and efficient Kossaczky algorithm. Therefore, we reach a fast local refinement and assure a high mesh quality along the whole adaptive procedure. Thirdly, we construct a trivariate T-spline representation of genus-zero solids based on the volume parameterization obtained by meccano method. The development of three-dimensional spline parameterizations from surface is an important challenge in the context of isogeometric analysis [2, 9, 3, 15].

Other advantages of the meccano method, that have been presented in previous papers, are: solid surface triangulation is automatically constructed, the final mesh is conforming with the object boundary, inner surfaces are automatically preserved (for example, interface between several materials), node distribution is adapted in accordance with the object geometry, and parallel computations can easily be developed for meshing the meccano pieces.

The rest of paper is organized as follow. In Section 2 we present a brief description of the meccano method. In Section 3 we compare the meshes produced by our method with the ones generated by TetGen and NETGEN. We analyze in Section 4 the behaviour of meccano meshes in an adaptive evolution problem. In Section 5 we generate a trivariate T-spline for solid isogeometric analysis. Finally some conclusions and future research are presented in Section 6.

## 2 The Meccano Method

The main steps of the *meccano tetrahedral mesh generation algorithm* are summarized in this section. The method has been previously introduced in [35, 36]. The input data of the algorithm is the definition of the solid boundary (for example a surface triangulation or CAD description) and a given precision (corresponding to the approximation of the solid boundary). The following algorithm describes our mesh generation approach.

Meccano tetrahedral mesh generation algorithm

1. *Meccano*: Construct a meccano, $\mathscr{M}$, approximation of the solid, $\Omega$, formed by polyhedral pieces.
2. *Mapping*: Define an admissible mapping, $\Pi$, between the meccano boundary faces, $\partial\mathscr{M}$, and the solid boundary, $\partial\Omega$, i.e. $\Pi : \partial\mathscr{M} \to \partial\Omega$.
3. *Coarse Mesh*: Construct a coarse tetrahedral mesh, $\mathscr{T}_0(\mathscr{M})$, of the meccano.
4. *Refined Mesh*: Generate a local refined tetrahedral mesh, $\mathscr{T}(\mathscr{M})$, from $\mathscr{T}_0(\mathscr{M})$, such that the surface triangulation, $\tau(\Omega)$, obtained after $\Pi$-mapping of $\mathscr{T}(\mathscr{M})$ boundary nodes, approximates the solid boundary $\partial\Omega$ for a given precision, $\varepsilon$.
5. *External Node Mapping*: Move the boundary nodes of $\mathscr{T}(\mathscr{M})$ to the solid surface according to $\Pi$.
6. *Relocation and Optimization*: Relocate the inner nodes of $\mathscr{T}(\mathscr{M})$ and optimize the resulting tetrahedral mesh by applying the simultaneous untangling and smoothing procedure to obtain the final tetrahedral mesh, $\mathscr{T}(\Omega)$, that approximates the solid.

The first step of the procedure is to construct a meccano approximation by connecting polyhedral pieces. The meccano and the solid must be equivalent from a topological point of view, i.e., their surfaces must have the same genus.

Once the meccano is assembled, we have to define an *admissible* one-to-one mapping between the boundary faces of the meccano and the boundary of the solid. If the solid is genus-zero and its boundary is given by a triangulation, we propose in [36] an automatic method to construct a parameterization of the solid surface triangulation to a cube boundary. For this purpose, we first divide the solid surface triangulation into six patches with the same topological connection as the cube faces. Then, a discrete mapping from each surface patch to the corresponding cube face is built by using the mean value parameterization proposed in [21].

At the moment, if the genus of the surface of the solid is greater than zero, the meccano should be defined by the user. An automatic construction of the meccano could be difficult

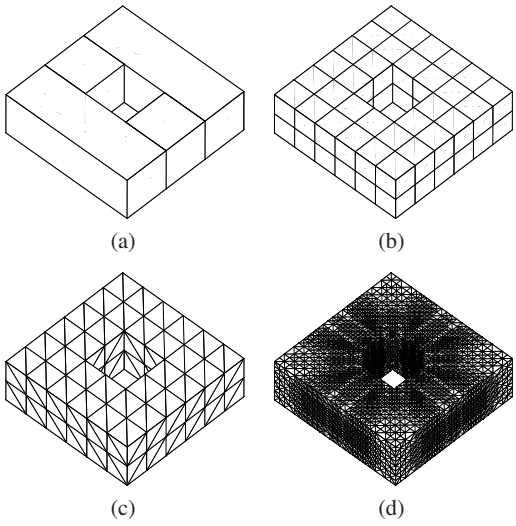



**Fig. 1** Steps of the meccano method for a toroidal solid on the parametric space: (a) meccano $\mathscr{M}$ formed by four cuboids, (b) cube mesh, (c) coarse tetrahedral mesh $\mathscr{T}_0(\mathscr{M})$, (d) refined mesh $\mathscr{T}(\mathscr{M})$.

when the topology of the solid is complex. We also remark that a non-optimal meccano can introduce large distortion in mesh generation. To avoid this issue an optimization of the boundary parameterization could be included [48].

In step 3, the meccano is decomposed into a coarse tetrahedral mesh $\mathscr{T}_0(\mathscr{M})$ by an appropriate subdivision of its initial polyhedral pieces. Although any tetrahedralization algorithm could be used, we propose a partition of meccano compatible with the Kossaczky refinement algorithm [28].

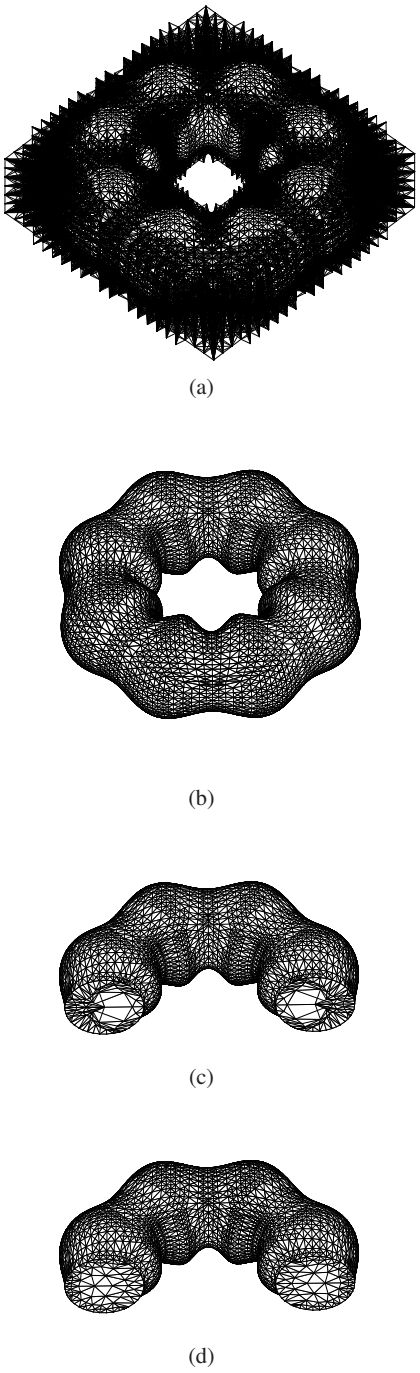
This mesh is locally refined in step 4 to obtain an approximation of the solid boundary within a given precision. To be more precise we have to introduce some notations. Given a tetrahedral mesh of the meccano $\mathscr{T}(\mathscr{M})$, we denote as $\tau(\mathscr{M})$ its boundary triangulation and $\tau(\Omega)$ the surface triangulation obtained after $\Pi$-mapping of $\tau(\mathscr{M})$ nodes. Note that $\tau_0(\Omega)$ is a coarse approximation of $\partial\Omega$. In order to improve this approximation we build a refined mesh $\mathscr{T}(\mathscr{M})$ of $\mathscr{T}_0(\mathscr{M})$ such that the distance between $\tau(\Omega)$ and $\partial\Omega$ is less than a prescribed tolerance $\varepsilon$. The concept of distance between surfaces can be defined and implemented in several ways. In our case is as follows: Let $T = \langle a,b,c\rangle$ be a triangle of $\tau(\mathscr{M})$, where $a$, $b$ and $c$ are their vertices, and let $p_k \in \{p_i\}_{i=1}^{N_q}$ be a Gauss quadrature point of $T$. We define the distance, $d(T)$, between the triangle $\langle \Pi(a),\Pi(b),\Pi(c)\rangle \in \tau(\Omega)$ and $\partial\Omega$ as the maximum of the volumes of the tetrahedra formed by $\Pi(a)$, $\Pi(b)$, $\Pi(c)$ and $\Pi(p_k)$. Then, the distance between $\tau(\Omega)$ and $\partial\Omega$ , $d(\tau(\Omega),\partial\Omega)$, is the maximum of all $d(T)$, that is

$$d(\tau(\Omega),\partial\Omega) = \max_{T\in\tau(\mathscr{M})} d(T) \tag{1}$$

We recall that local refinement stops when $d(\tau(\Omega),\partial\Omega) < \varepsilon$. Note that this is an approximation of the maximum missed (or overestimed) volume per face of $\tau(\Omega)$. A more accurate approach of distance based on Hausdorff envelope can be found in [4].

Then, we construct a mesh of the solid $\mathscr{T}(\Omega)$ by mapping the boundary nodes of $\mathscr{T}(\mathscr{M})$ to $\partial\Omega$ and by relocating the inner nodes at a *reasonable* position. After these two steps, the resulting mesh is generally tangled. Therefore, a simultaneous untangling and smoothing procedure [11, 13] is applied and a valid adaptive tetrahedral mesh of the solid is obtained. In short, this last procedure finds the new positions of the inner nodes of $\mathscr{T}(\Omega)$ optimizing an objective function. Such a function is based on a certain measurement of the quality of the local submesh $N(q)$, formed by the set of tetrahedra connected to the *free node* $q$. In fact, we use a suitable modification of the objective function such that it is regular over all $\mathbb{R}^3$, [11].

We illustrate the meccano algorithm with a toroidal solid that is generated by revolving a circle of variable radius, see Figure 2(b). This is a genus-one solid. The first algorithm step constructs a simple meccano composed of four cuboids as shown in Figure 1(a). In this case, the definition of a one-to-one mapping between the meccano and toroid boundaries is straightforward. In the third step of the mesh generator procedure, the meccano is splited into a coarse and uniform



(a) (b) (c) (d)

**Fig. 2** Steps of the meccano method for a toroidal solid on the physical space: (a) tangled mesh after the mapping of meccano boundary nodes to the toroid surface and (b) resulting tetrahedal mesh $\mathscr{T}(\Omega)$ after inner node relocation and mesh optimization. Cross sections of the toroid before (c) and after (d) the application of the mesh optimization process.

cube mesh (i.e., a polycube), see Figure 1(b), and each cube is divided into six tetrahedra. It results in a coarse tetrahedral mesh of the meccano shown in Figure 1(c). In the following step, we reach the local refined mesh that is shown in Figure 1(d). A tangled tetrahedral mesh, see Figure 2(a), is obtained after the mapping of the meccano boundary nodes to the toroid surface. The relocation of the meccano inner nodes reduces the number of inverted tetrahedra, but the tangling problem is not solved completely; see Figure 2(c). So, in the final algorithm step is necessary the application of the tetrahedral mesh optimization presented in [11, 13]. The final mesh can be seen in Figure 2(b) and its cross section in Figure 2(d).

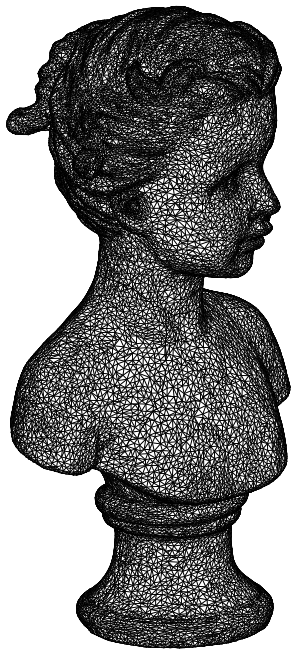
We have implemented the meccano technique using several freely-available libraries:

- The module of 3-D refinement of ALBERTA code [38].
- Our mesh optimization library: SUS Code [13].
- The parameterization toolbox of the geometry group at SINTEF ICT, Department of Applied Mathematics.

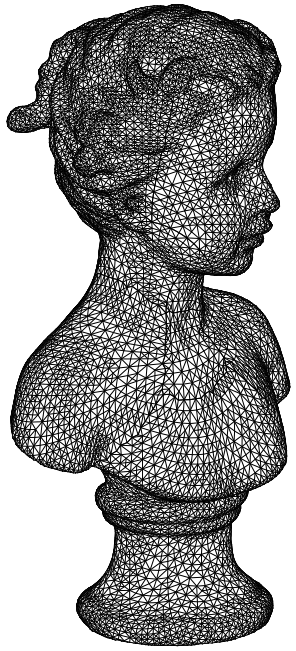
ALBERTA is an adaptive multilevel finite element toolbox developed in C. This software can be used for solving several types of 1-D, 2-D or 3-D problems. ALBERTA uses the Kossaczky refinement algorithm [28] and requires an initial mesh topology [37]. The recursive refinement algorithm could not terminate for general meshes. However, meshes constructed using the meccano method verify the topology and structure restrictions imposed by ALBERTA. They can be refined by its recursive algorithm, because they are loop-free, and the degeneration of the resulting triangulations after successive refinements is avoided.

The mesh optimization algorithm proposed in [11], which is a simultaneous untangling and smoothing technique, is implemented in the SUS Code [13] using C++ programming language.

If the solid is genus-zero and its boundary is given by a triangulation, its parameterization can be automatically obtained following the procedure presented in [36] that uses GoTools core and parameterization modules from SINTEF ICT, (http://www.sintef.no/math_software). This code implements Floater's parameterization in C++ [20, 21]. However, at present, if the genus of solid is bigger than zero a one-to-one mapping is required by the method.



(a)



(b)

**Fig. 3** (a) Original surface triangulation of the Bust with 32002 nodes and 64000 triangles, (b) resulting tetrahedral mesh generated by the meccano method for $\varepsilon = 0.1$ (34524 nodes, 147352 tetrahedra, 16129 boundary nodes and 32254 boundary faces).

## 3 Comparison with NETGEN and TetGen

In this section we compare the meshes generated by our meccano code versus other standard ones: NETGEN and TetGen.

NETGEN is an automatic mesh generation tool for two and three dimensions that was developed by J. Schöberl [39] in C++. Surface and volume mesh generation are based on the advancing front method. The input of this method is a boundary mesh. Starting with the original boundary, element by element is cut off to reduce the domain iteratively. NETGEN allows to combine the advancing front procedure with a fast Delaunay algorithm. In this case, the domain is firstly filled with tetrahedra by using the Delaunay technique and

when it fails then the slower advancing front algorithm takes over. NETGEN also contains modules for mesh optimization and hierarchical mesh refinement.

TetGen is a program developed by H. Si [41, 42] in C++, that generates tetrahedral meshes for three-dimensional domains. The algorithms used are of Delaunay type. Given the domain boundary as a surface triangulation, TetGen can produce the boundary constrained tetrahedralization (which exactly preserves the input surface mesh), constrained or conforming Delaunay tetrahedralization (which may add some extra points into the input surface triangulation). The quality of the mesh can be improved by the standard Delaunay refinement algorithm, such as the Shewchuk's algorithm [40]. It adds vertices to the mesh to ensure that no tetrahedron has a radius-edge ratio greater than a given value. The default value is 2.0, which is equivalent to bound the smallest face angle c.a. 15 degrees. TetGen also provides options to use other quality measures or a combination of them, such as the smallest and the largest dihedral angle bounds.

In order to compare the meshes, we have to introduce some quality metric. In the following we have considered three quality metrics: mean ratio ($q_1$), condition number ($q_2$), and the minimal dihedral angle ($q_3$). The first two are algebraic shape quality metrics [26, 27], and for a valid tetrahedron $T$ are given by

$$q_1(T) = \frac{3\sigma^{\frac{2}{3}}}{|\mathbf{S}|^2}, \qquad q_2(T) = \frac{3}{|\mathbf{S}||\mathbf{S}^{-1}|},$$

being $|\mathbf{S}| = \sqrt{\mathrm{tr}(\mathbf{S}^T\mathbf{S})}$ the Frobenius norm of the weighted Jacobian matrix $\mathbf{S}$ that is associated to the affine map from the ideal tetrahedron (usually equilateral one) to the physical one $T$, and $\sigma = \det(\mathbf{S})$.

To compare our method with the other ones, we take the same surface triangulation of the Bust, Figure 3(a), as input data for the three methods. It has been obtained from the AIM@SHAPE Shape Repository. The surface triangulation has 32002 nodes and 64000 triangles, and its bounding box is defined by the points $(x,y,z)_{\min} = (-120,-30.5,-44)$ and $(x,y,z)_{\max} = (106,60,46)$. As meccano we consider a cube with an edge length equal to 20. Its center is placed inside the solid at the point $(5,-3.4)$. The automatic parameterization of the solid surface is done with the procedure that was introduced in [36]. This procedure is based on the construction of a compatible decomposition of the surface triangulation of Figure 3(a) into six patches and then it uses the Floater's parameterization [21] on each patch. A detailed description of the application of the meccano method to the Bust test example is presented in [7].

We report in Tables 1, 2, 3 and 4 some features of the meshes generated by Meccano, NETGEN, TetGen v.1.4.3 and TetGen v.1.5.0, respectively. The CPU time has been measured on a Dell precision 690, 2 Dual Core Xeon processor and 8 Gb RAM memory and using standard optimization in all cases.

Table 1 contains information about three meshes constructed by the meccano method for several values of the approximation parameter $\varepsilon$. We recall that the meccano method does not maintain the input data. It creates its own boundary triangulation that approximates the original one. Note that the meccano technique generates a high quality tetrahedral meshes.

Table 2 shows information about two meshes generated by NETGEN. The first mesh is generated combining an advancing front method and a Delaunay algorithm. The second mesh is produced using only an advancing front technique. Both meshes have been optimized by making five iterations of the optimization module that is included in NETGEN.

Table 3 reports information about six meshes generated by TetGen v.1.4.3. We use this code to produce boundary constrained and conforming Delaunay tetrahedralizations, imposing several restrictions about quality. In fact, we employ the -Y switch to suppress the creation on Steiner points on the boundary (constrained tetrahedralization), and the -q switch to ensure that no tetrahedra have radius-edge ratio greater than a prescribed value.

Table 4 contains information about six meshes generated by TetGen v.1.5.0. We build again boundary constrained and conforming Delaunay tetrahedralizations. As quality parameter we now use -qq to set a minimum dihedral angle bound, and -qo to fix a maximum dihedral angle bound. Note that this new TetGen version includes a new algorithm for boundary recovery that generates a "strictly" constrained tetrahedralization, i.e. no Steiner points are included into the input surface mesh. However a great number of Steiner points are introduced in the conforming Delaunay mesh.

The minimal values of the quality metrics are shown in Fig. 4. In addition, we present in Fig. 5 the 'cumulative frequency polygon' of the generated meshes; for a given value of $x \in (0.1,1)$ each line represents the percentage of elements that have a quality less than $x$. We note that the selected criterion to compare the quality is not relevant from a qualitative point of view.

The following comments summarize the comparison between the methods:

- NETGEN and TetGen v.1.5.0 (constrained strategy) maintain the input surface triangulation, i.e. they do not introduce any Steiner point. TetGen v.1.4.3 (constrained strategy) introduces only a few boundary nodes to construct the Delaunay tetrahedralization, but it does not modify the original surface. However, Meccano generates a new boundary triangulation that is an approximation of the original one. This is an important handicap for the two fist methods that could have an effect in term of quality.

**Table 1** Meccano: Features of meccano meshes produced for values of $\varepsilon$: 0.05, 0.01, and 0.005. The quality is measured with the shape mean metric ($q_1$), with the metric based on the condition number of the weighted Jacobian matrix ($q_2$), and with the minimal dihedral angle ($q_3$) in grades.

| Strategy | | Nodes | Tetrahedra | Boundary Nodes | Boundary Faces | CPU time |
|---|---|---|---|---|---|---|
| $\varepsilon = 0.05$ | | 49427 | 211662 | 22954 | 45904 | 45.0 s |
| $\varepsilon = 0.01$ | | 109878 | 471728 | 50579 | 101154 | 81.3 s |
| $\varepsilon = 0.005$ | | 152229 | 656064 | 69337 | 138670 | 114.1 s |
| Strategy | | $\min q_i(T)$ | mean $q_i(T)$ | $\#\{T : q_i(T) < 0.1\}$ | $\#\{T : q_i(T) < 0.2\}$ | $\#\{T : q_i(T) < 0.3\}$ |
| $\varepsilon = 0.05$ | $q_1$ | 0.16 | 0.74 | 0 (0.000%) | 12 (0.005%) | 369 (0.17%) |
| | $q_2$ | 0.09 | 0.73 | 1 (0.000%) | 63 (0.030%) | 437 (0.20%) |
| | $q_3$ | 3.98 | 41.88 | – | – | – |
| $\varepsilon = 0.01$ | $q_1$ | 0.14 | 0.75 | 0 (0.000%) | 35 (0.007%) | 722 (0.15%) |
| | $q_2$ | 0.07 | 0.74 | 8 (0.002%) | 106 (0.022%) | 780 (0.16%) |
| | $q_3$ | 3.13 | 42.37 | – | – | – |
| $\varepsilon = 0.005$ | $q_1$ | 0.13 | 0.75 | 0 (0.000%) | 40 (0.006%) | 889 (0.13%) |
| | $q_2$ | 0.06 | 0.75 | 12 (0.002%) | 138 (0.021%) | 919 (0.14%) |
| | $q_3$ | 2.75 | 42.59 | – | – | – |

**Table 2** NETGEN: Features of two meshes generated by NETGEN. The first one is produced combining advancing front method and Delaunay algorithm. The second is generated using only advancing front technique. The quality is measured with the mean ratio metric ($q_1$), with the metric based on the condition number of the weighted Jacobian matrix ($q_2$), and with the minimal dihedral angle ($q_3$) in grades.

| Strategy | | Nodes | Tetrahedra | Boundary Nodes | Boundary Faces | CPU time |
|---|---|---|---|---|---|---|
| With Delaunay | | 57965 | 242875 | 32002 | 64000 | 80.0 s |
| Without Delaunay | | 130210 | 683534 | 32002 | 64000 | 602.1 s |
| Strategy | | min q(T) | mean q(T) | $\#\{T : q(T) < 0.1\}$ | $\#\{T : q(T) < 0.2\}$ | $\#\{T : q(T) < 0.3\}$ |
| With Delaunay | $q_1$ | 0.04 | 0.81 | 6 (0.002%) | 27 (0.011%) | 71 (0.029%) |
| | $q_2$ | 0.01 | 0.80 | 24 (0.010%) | 62 (0.026%) | 265 (0.110%) |
| | $q_3$ | 0.55 | 45.07 | – | – | – |
| Without Delaunay | $q_1$ | 0.04 | 0.71 | 8 (0.001%) | 231 (0.033%) | 2321 (0.340%) |
| | $q_2$ | 0.01 | 0.70 | 141 (0.021%) | 1365 (0.200%) | 9038 (1.322%) |
| | $q_3$ | 0.50 | 38.87 | – | – | – |

- Meccano meshes have better quality than meshes generated by TetGen and NETGEN.
- Meccano meshes have good structures because they are constructed by simple nested refinements. Moreover, they can be easily refined by using the Kossaczky algorithm. TetGen and NETGEN produce meshes without this type of structure. Therefore, they require more complex refinement algorithms.
- TetGen is the fastest code. Meccano is competitive versus NETGEN from a computational cost point of view.

We now study if the quality of NETGEN and TetGen meshes can be improved by using the SUS Code. We have applied an improved version of SUS Code to the meshes generated by NETGEN and TetGen v.1.5.0. The results are presented in Tables 5 and 6, respectively. Note that, the application of five smoothing iterations of SUS Code on NETGEN meshes does not produce a significant improvement in the mesh quality. However, the quality of TetGen meshes is significantly improved after five iterations of SUS Code.

As we have commented above, NETGEN and TetGen maintain the input surface triangulation, and therefore the quality of the final tetrahedral meshes depends on the quality of the data. In order to eliminate the effect of the initial boundary triangulation in the comparison of the codes, we finally use the boundary mesh produced by the Meccano as input data for the other codes. Table 7 summarizes the information of the resulting meshes. Note that NETGEN produces a mesh of extraordinary quality, and Meccano meshes are generally better than TetGen ones. We remark that the quality of the Meccano meshes is linked to the structure imposed by the Kossaczky refinement.

## 4 Adaptive Local Refinement of the Meccano Meshes

Our volume meshes can be utilized in adaptive finite element applications by using the Kossaczky's algorithm. The local refinement steps are very fast because the sequence of solid nested meshes is defined from the coarse mesh of the meccano, i.e., the dividing edge for tetrahedron bisection is known straightforward. In addition, we note that the minimum mesh quality is bounded during all the mesh adaptation process, because similar elements appear in the meccano mesh after three consecutive bisections. The minimum quality of refined meshes is function of the initial mesh quality [34, 45].

In order to assess the quality of meccano meshes within an adaptive procedure we propose the following example. In this Section we have used the mean ratio quality metric ($q_1$).

**Table 3** TetGen v.1.4.3: Features of meshes generated by TetGen. The -Y switch suppresses the creations of Steiner points on the boundary. The -q switch ensures that no tetrahedra have radius-edge greater than a prescribed value. The quality is measured with the mean ratio metric ($q_1$), with the metric based on the condition number of the weighted Jacobian matrix ($q_2$), and with the minimal dihedral angle ($q_3$) in grades.

| Strategy | | Nodes | Tetrahedra | Boundary Nodes | Boundary Faces | CPU time |
|---|---|---|---|---|---|---|
| Constrained | | | | | | |
| -Yq2 | | 45021 | 179056 | 32005 | 64006 | 9.4 s |
| -Yq1.5 | | 53918 | 237736 | 32005 | 64006 | 11.5 s |
| -Yq1.1 | | 95512 | 504069 | 32005 | 64006 | 21.2 s |
| Conforming | | | | | | |
| -q2 | | 68242 | 271188 | 48059 | 96112 | 11.4 s |
| -q1.5 | | 79096 | 341901 | 48233 | 96462 | 15.0 s |
| -q1 | | 278904 | 1596154 | 48863 | 97722 | 50.2 s |
| Strategy | | min q(T) | mean q(T) | $\#\{T : q(T) < 0.1\}$ | $\#\{T : q(T) < 0.2\}$ | $\#\{T : q(T) < 0.3\}$ |
| Constrained | | | | | | |
| -Yq2 | $q_1$ | 0.04 | 0.67 | 43 (0.02%) | 517 (0.29%) | 3028 (1.70%) |
| | $q_2$ | 0.01 | 0.65 | 221 (0.12%) | 2583 (1.14%) | 8289 (4.62%) |
| | $q_3$ | 0.48 | 35.49 | – | – | – |
| -Yq1.5 | $q_1$ | 0.04 | 0.71 | 44 (0.02%) | 440 (0.18%) | 2437 (1.00%) |
| | $q_2$ | 0.01 | 0.70 | 200 (0.08%) | 2119 (0.89%) | 7975 (3.35%) |
| | $q_3$ | 0.48 | 37.92 | – | – | – |
| -Yq1.1 | $q_1$ | 0.03 | 0.78 | 76 (0.02%) | 451 (0.09%) | 2459 (0.49%) |
| | $q_2$ | 0.01 | 0.76 | 208 (0.04%) | 2218 (0.44%) | 10476 (2.07%) |
| | $q_3$ | 0.48 | 42.77 | – | – | – |
| Conforming | | | | | | |
| -q2 | $q_1$ | 0.03 | 0.67 | 36 (0.01%) | 415 (0.15%) | 3599 (1.30%) |
| | $q_2$ | 0.02 | 0.65 | 215 (0.08%) | 3039 (1.12%) | 11206 (4.13%) |
| | $q_3$ | 0.79 | 35.38 | – | – | – |
| -q1.5 | $q_1$ | 0.03 | 0.71 | 34 (0.01%) | 372 (0.11%) | 2906 (0.85%) |
| | $q_2$ | 0.02 | 0.70 | 204 (0.06%) | 2652 (0.78%) | 10422 (3.05%) |
| | $q_3$ | 0.79 | 37.92 | – | – | – |
| -q1 | $q_1$ | 0.03 | 0.81 | 34 (0.00%) | 421 (0.03%) | 3114 (0.19%) |
| | $q_2$ | 0.02 | 0.80 | 224(0.01%) | 3365 (0.21%) | 21737 (1.36%) |
| | $q_3$ | 0.79 | 45.59 | – | – | – |

Let us consider the linear heat evolution equation with null initial and Dirichlet boundary conditions:

$$\begin{aligned} \partial_t u - \Delta u &= f && \text{in } \Omega \times (0,T), \\ u &= 0 && \text{on } \partial\Omega \times (0,T), \\ u &= 0 && \text{on } \Omega \times \{0\}, \end{aligned}$$

where the solid domain $\Omega$ is the Stanford Bunny (see Figure 11), $t \in [0,4]$ and $f(x,t)$ is a space-time depending function. In this application we have considered as $f(x,t)$ a gaussian-type function in space, that is moving in time around a circle inside the domain. Our space-time discretization consists of piecewise linear element and backward Euler method with variable time step. The adaptive algorithm is driven by the residual estimator proposed in [10]. We use the implicit adaptive strategy of type A described in [1], that means, for each time step we start from the previous step mesh and repeat the process: SOLVE $\rightarrow$ ESTIMATE $\rightarrow$ MARK $\rightarrow$ REFINE/DEREFINE, until the estimated error is below the fixed tolerance.

The initial tetrahedral mesh is generated applying the meccano method to a surface triangulation that has been obtained from the Stanford Computer Graphics Laboratory. The resulting mesh has 13105 nodes and 54906 tetrahedra; its minimum quality is 0.106 and the mean quality 0.677. Figure 11(a) and Figure 6(c) show the initial mesh and a cross section, respectively.

In Figure 6(a) and (b) the initial null solution and the mesh element quality curve are shown. The quality curve is obtained by sorting the mesh elements in increasing order of quality. We consider again the mean ratio as quality metric.

In Figure 7 we present several solutions and the corresponding adaptive meshes at times 0.5, 1 and 1.5. Note that the adaptive algorithm captures the solutions and refines accordingly.

The evolution of the minimum and mean mesh quality in the course of the evolution process are shown in Figure 8(a). The minimum mesh quality is bounded during all adaptive procedure. In fact, the worst tetrahedron quality is 0.066. The mean quality is always bigger than 0.5. Moreover, the percentage of elements with quality less than 0.1 is negligible (less than 0.14%). Meanwhile, the percentage of elements with quality less than 0.2 and 0.3 is around 5% and 15%, respectively (see Figure8(b)).

Finally, we show in Figure 9 the quality curves at several times. In conclusion, the mesh generated by our method has good quality even after being locally refined.

**Table 4** TetGen v.1.5.0: Features of meshes generated by TetGen. The -Y switch suppresses the creations of Steiner points on the boundary. The -qq parameter sets a minimum dihedral angle bound, and -qo parameter sets a maximum dihedral angle bound. The quality is measured with the mean ratio metric ($q_1$), with the metric based on the condition number of the weighted Jacobian matrix ($q_2$), and with the minimal dihedral angle ($q_3$) in grades.

| Strategy | | Nodes | Tetrahedra | Boundary Nodes | Boundary Faces | CPU time |
|---|---|---|---|---|---|---|
| Constrained | | | | | | |
| -Yqq15 | | 61957 | 284858 | 32002 | 64000 | 10.4 s |
| -Yqo165 | | 44692 | 177043 | 32002 | 64000 | 7.9 s |
| -Yqq15o165 | | 61957 | 284858 | 32002 | 64000 | 10.4 s |
| Conforming | | | | | | |
| -qq15 | | 279279 | 1061273 | 200021 | 400038 | 46.3 s |
| -qo165 | | 250753 | 880106 | 199962 | 399920 | 40.1 s |
| -qq15o165 | | 279279 | 1061237 | 200021 | 400038 | 46.3 s |
| Strategy | | min q(T) | mean q(T) | $\#\{T : q(T) < 0.1\}$ | $\#\{T : q(T) < 0.2\}$ | $\#\{T : q(T) < 0.3\}$ |
| Constrained | | | | | | |
| -Yqq15 | $q_1$ | 0.04 | 0.74 | 28 (0.01%) | 260 (0.09%) | 1383 (0.49%) |
| | $q_2$ | 0.01 | 0.72 | 113 (0.04%) | 1127 (0.40%) | 3875 (1.36%) |
| | $q_3$ | 0.48 | 40.04 | – | – | – |
| -Yqo165 | $q_1$ | 0.04 | 0.67 | 40 (0.02%) | 333 (0.18%) | 2676 (1.51%) |
| | $q_2$ | 0.01 | 0.65 | 141 (0.08%) | 2283 (1.29%) | 7895 (4.46%) |
| | $q_3$ | 0.48 | 35.33 | – | – | – |
| -Yqq15o165 | $q_1$ | 0.04 | 0.74 | 28 (0.01%) | 260 (0.09%) | 1383 (0.49%) |
| | $q_2$ | 0.01 | 0.72 | 113 (0.04%) | 1127 (0.40%) | 3875 (1.36%) |
| | $q_3$ | 0.48 | 40.04 | – | – | – |
| Conforming | | | | | | |
| -qq15 | $q_1$ | 0.01 | 0.63 | 2988 (0.28%) | 22299 (2.10%) | 72291 (6.80%) |
| | $q_2$ | 0.02 | 0.62 | 3498 (0.33%) | 30025 (2.83%) | 91878 (8.66%) |
| | $q_3$ | 0.52 | 36.31 | – | – | – |
| -qo165 | $q_1$ | 0.01 | 0.59 | 2950 (0.33%) | 22540 (2.56%) | 73755 ( 8.38%) |
| | $q_2$ | 0.003 | 0.57 | 3527 (0.40%) | 31222 (3.55%) | 97561 (11.09%) |
| | $q_3$ | 0.26 | 34.04 | – | – | – |
| -q15o165 | $q_1$ | 0.01 | 0.63 | 2989 (0.28%) | 22317 (2.10%) | 72331 (6.81%) |
| | $q_2$ | 0.02 | 0.62 | 3498 (0.33%) | 30025 (2.83%) | 91878 (8.66%) |
| | $q_3$ | 0.52 | 36.31 | – | – | – |

**Table 5** NETGEN + SUS: Mean ratio quality of resulting meshes after five smoothing iterations of SUS Code on NETGEN meshes.

| Strategy + SUS | min $q_1(T)$ | mean $q_1(T)$ | $\#\{T : q_1(T) < 0.1\}$ | $\#\{T : q_1(T) < 0.2\}$ | $\#\{T : q_1(T) < 0.3\}$ |
|---|---|---|---|---|---|
| With Delaunay | 0.04 | 0.81 | 6 (0.003%) | 27 (0.011%) | 71 (0.029%) |
| Without Delaunay | 0.04 | 0.71 | 7 (0.001%) | 245 (0.035%) | 2362 (0.340%) |

**Table 6** TetGen v.1.5.0 + SUS: Mean ratio quality of resulting meshes after five smoothing iterations of SUS Code on TetGen v.1.5.0 meshes.

| Strategy + SUS | min $q_1(T)$ | mean $q_1(T)$ | $\#\{T : q_1(T) < 0.1\}$ | $\#\{T : q_1(T) < 0.2\}$ | $\#\{T : q_1(T) < 0.3\}$ |
|---|---|---|---|---|---|
| Constrained | | | | | |
| -Yqq15 | 0.035 | 0.77 | 17 (0.006%) | 223 (0.08%) | 932 (0.33%) |
| -Yqo165 | 0.035 | 0.70 | 31 (0.002%) | 281 (0.16%) | 1346 (0.76%) |
| -Yqq15o165 | 0.035 | 0.77 | 17 (0.006%) | 223 (0.08%) | 932 (0.33%) |
| Conforming | | | | | |
| -qq15 | 0.015 | 0.68 | 197 (0.019%) | 3417 (0.32%) | 21364 (2.01%) |
| -qo165 | 0.009 | 0.63 | 202 (0.023%) | 3831 (0.43%) | 24716 (2.81%) |
| -qq15o165 | 0.015 | 0.68 | 197 (0.019%) | 3417 (0.32%) | 21364 (2.01%) |

## 5 Application of Volume Parameterization in Isogeometric Analysis

One of the most important contributions of the meccano method is the resulting volume parameterization of the solid. It can have interesting applications in solid modeling and numerical simulation. Particularly, the construction of volume T-mesh for isogeometric analysis [2, 9, 3] can be easier.

The key lies in using the mapping, provided by the volume parameterization, to transform a T-mesh defined on the parametric domain (meccano) into the physical domain. The T-mesh of the parametric domain is the parametric space where the set of T-splines are defined [3].

In order to make the construction of the T-mesh easier, we apply an octree subdivision of a cube enclosing the initial polycube decomposition of the meccano. The dimension of

**Table 7** Features of meshes generated by Meccano, TetGen and NETGEN, when the input surface triangulation is the output one constructed by Meccano. The quality is measured with the mean ratio metric ($q_1$), with the metric based on the condition number of the weighted Jacobian matrix ($q_2$), and with the minimal dihedral angle ($q_3$) in grades.

| Strategy | Nodes | Tetrahedra | Boundary Nodes | Boundary Faces | CPU time |
|---|---|---|---|---|---|
| Meccano: $\varepsilon = 0.05$ | 49427 | 211662 | 22954 | 45904 | 45.0 s |
| TetGen 1.4.3: -Yq1.1 | 56039 | 278970 | 22954 | 45904 | 8.6 s |
| TetGen 1.4.3: -q1 | 102861 | 562376 | 26307 | 52610 | 15.0 s |
| TetGen 1.5.0: -Yqq15 | 39706 | 172632 | 22954 | 45904 | 5.9 s |
| TetGen 1.5.0: -qq15 | 164712 | 588907 | 127108 | 254212 | 25.0 s |
| NETGEN: (Delaunay) | 37777 | 149230 | 22954 | 45904 | 41.0 s |

| Strategy | | min q(T) | mean q(T) | $\#\{T : q(T) < 0.1\}$ | $\#\{T : q(T) < 0.2\}$ | $\#\{T : q(T) < 0.3\}$ |
|---|---|---|---|---|---|---|
| Meccano: $\varepsilon = 0.05$ | $q_1$ | 0.16 | 0.74 | 0 (0.00%) | 12 (0.005%) | 369 (0.17%) |
| | $q_2$ | 0.09 | 0.73 | 1 (0.00%) | 63 (0.030%) | 437 (0.20%) |
| | $q_3$ | 3.98 | 41.88 | – | – | – |
| TetGen 1.4.3: -Yq1.1 | $q_1$ | 0.09 | 0.79 | 3 (0.00%) | 20 (0.007%) | 377 (0.13%) |
| | $q_2$ | 0.05 | 0.78 | 8 (0.00%) | 468 (0.168%) | 4279 (1.53%) |
| | $q_3$ | 2.44 | 43.84 | – | – | – |
| TetGen 1.4.3: -q1 | $q_1$ | 0.22 | 0.81 | 0 (0.00%) | 0 (0.000%) | 446 (0.08%) |
| | $q_2$ | 0.14 | 0.80 | 0 (0.00%) | 692 (0.123%) | 6913 (1.23%) |
| | $q_3$ | 5.95 | 45.90 | – | – | – |
| TetGen 1.5.0: -Yqq15 | $q_1$ | 0.02 | 0.74 | 1 (0.00%) | 24 (0.014%) | 143 (0.08%) |
| | $q_2$ | 0.004 | 0.73 | 19 (0.01%) | 177 (0.102%) | 1450 (0.84%) |
| | $q_3$ | 0.21 | 40.90 | – | – | – |
| TetGen 1.5.0: -qq15 | $q_1$ | 0.02 | 0.59 | 1848 (0.31%) | 18761 (3.186%) | 62612 (10.63%) |
| | $q_2$ | 0.02 | 0.58 | 1233 (0.21%) | 20399 (3.464%) | 73011 (12.40%) |
| | $q_3$ | 0.80 | 35.50 | – | – | – |
| NETGEN:(Delaunay) | $q_1$ | 0.36 | 0.81 | 0 (0.00%) | 0 (0.000%) | 0 (0.00%) |
| | $q_2$ | 0.31 | 0.81 | 0 (0.00%) | 0 (0.000%) | 0 (0.00%) |
| | $q_3$ | 12.62 | 46.18 | – | – | – |

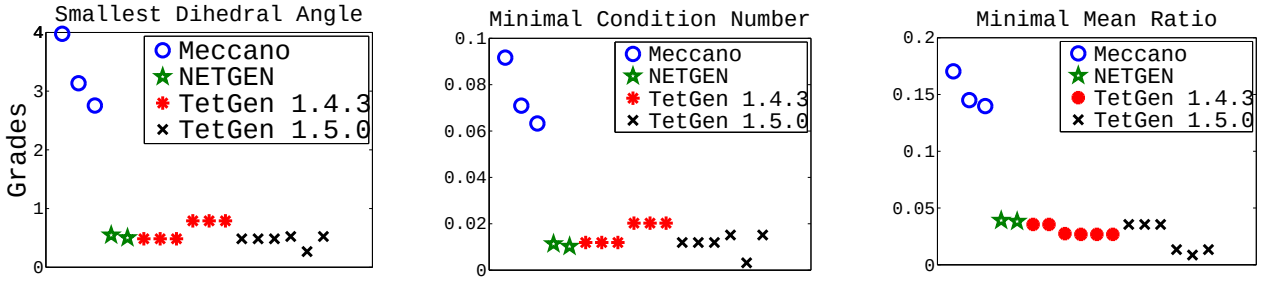



**Fig. 4** Minimal values of quality metrics for meshes generated by Meccano (blue), NETGEN (green), TetGen v.1.4.3 (red) and TetGen v.1.5.0 (black). Meshes have been ordered by appearance in Tables 1,2,3 and 4.

the cube must satisfy a certain restriction so that this polycube is nested inside the octree. Each leaf of the octree is divided in eight children (eight cubes). The division continues until each terminal cube of the octree does not contain a node of the Kossaczky's meccano mesh in its inner. This subdivision produces vertices both inside and outside the meccano, but only the inner vertices must be considered as *anchors*. The external vertices can be used to complete the unclamped *knot vectors*.

Thus, the image of a point $(u,v,w)$ in the parametric domain is given by

$$\mathbf{S}(u,v,w) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(u,v,w)$$

where $R_\alpha(u,v,w) = \frac{B_\alpha(u,v,w)}{\sum_{\beta \in A} B_\beta(u,v,w)}$ is the T-spline blending function and $B_\alpha$ the corresponding B-spline associated to the $\mathbf{s}_\alpha$ *anchor*. The index set $A$ runs over all the anchors of the T-mesh. The control points $\mathbf{P}_\alpha$ are calculated by imposing the conditions $\mathbf{S}(\mathbf{s}_\beta) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(\mathbf{s}_\beta)$ for all the anchors of the T-mesh. Here, we have also used the anchors as interpolation points. The image of each interpolation point $\mathbf{s}_\beta$ in the physical space, $\mathbf{S}(\mathbf{s}_\beta)$, is determined by the volume parameterization.

A complete description of the procedure for genus-zero solids can be found at [15]. An introduction to solids with surface of genus greater than zero is presented in [14].

In [17] we present some results of the application of isogeometric analysis with T-splines to the resolution of Pois-

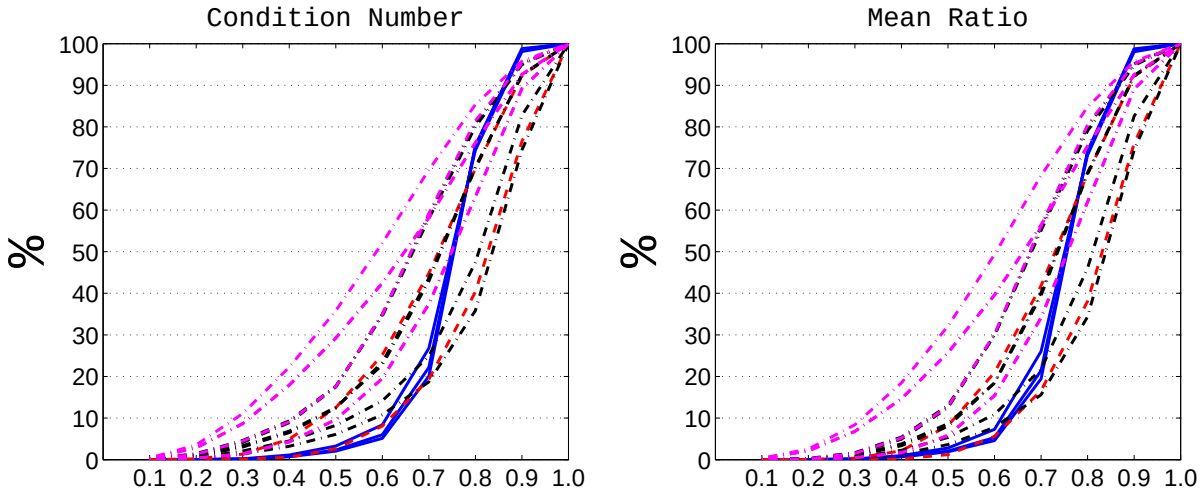



**Fig. 5** 'Cummulative frequency polygon' for meshes generated by Meccano (blue), NETGEN (red), TetGen v.1.4.3 (black) and TetGen v.1.5.0 (magenta), for condition number and mean ratio metrics. That is, for a $x \in (0.1, 1)$ each line represents the percentage of elements that have quality less than $x$.

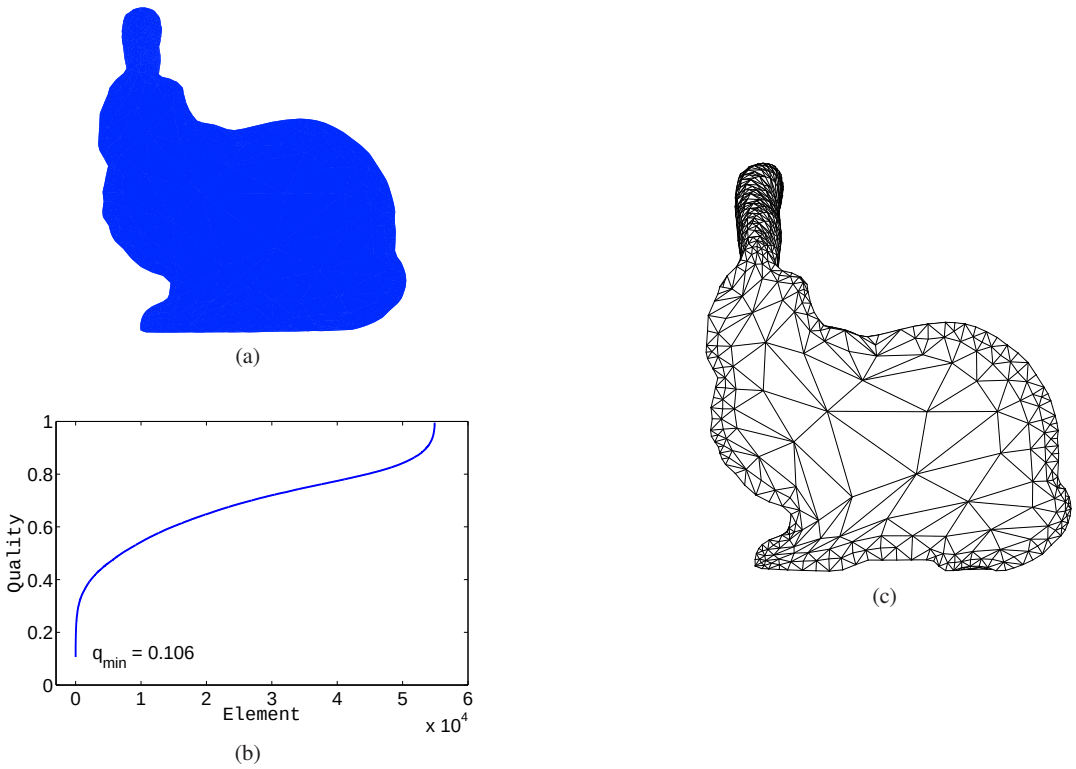



**Fig. 6** Solution of the linear heat evolution problem at time $t = 0$ (a), quality curve (b) and cross section of the initial tetrahedral mesh.

son equation in a three-dimensional solid that is parameterized with this technique.

We summarize here an application of our method to the Stanford bunny. The original surface triangulation has been obtained from the Stanford Computer Graphics Laboratory, *http://graphics.stanford.edu/data/3Dscanrep/*.

Figure 10(a) shows the cube tetrahedral mesh obtained by the meccano method. Figure 10(b) shows the parametric T-mesh. Figure 11(a) shows the tetrahedral mesh of the bunny constructed by the meccano method. Figure 11(b) shows the T-mesh of the Standford bunny generated by the application of the mapping $\mathbf{S}(u,v,w)$ to the parametric T-mesh.

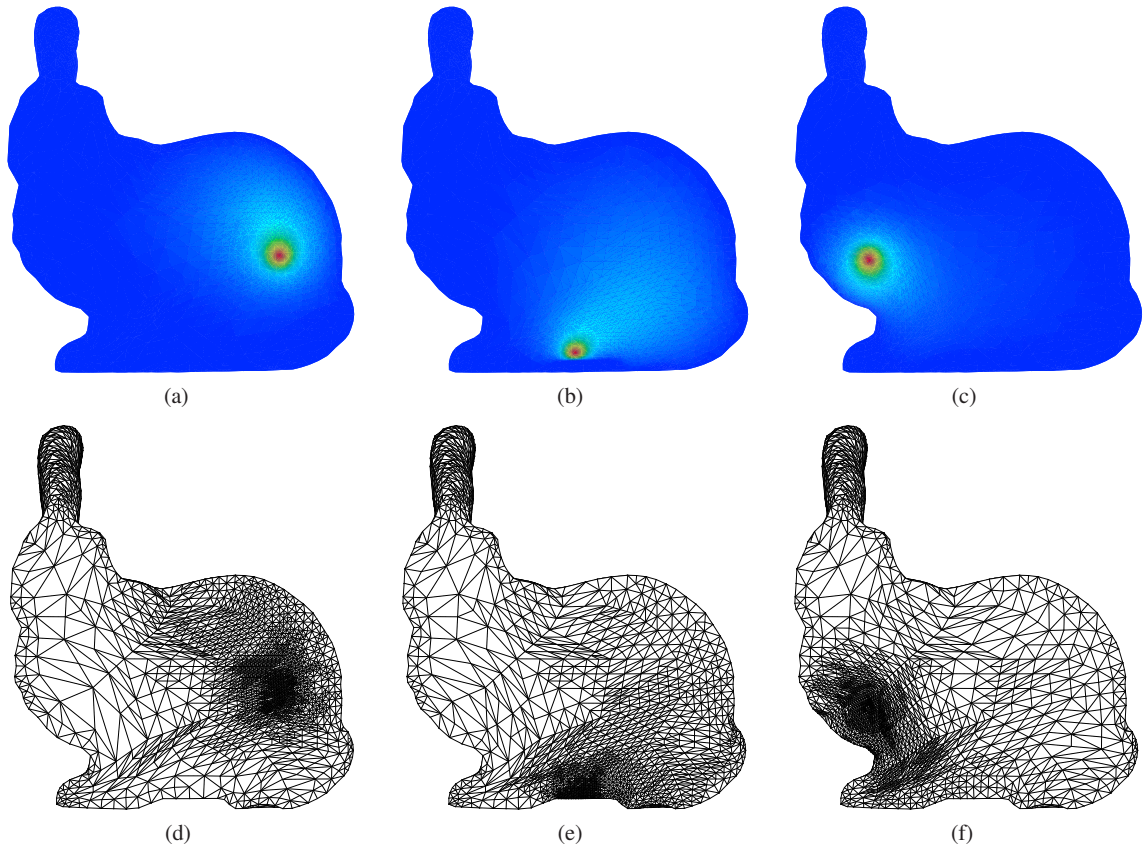
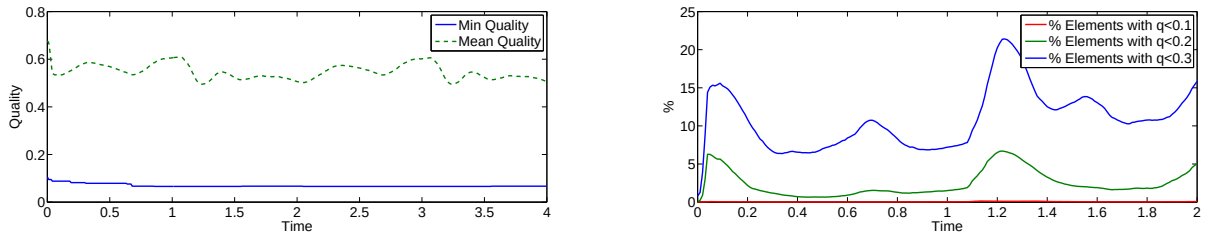



**Fig. 7** Solutions of the linear heat evolution problem at time $t = 0.5$ (a), $t = 1.0$ (b) and $t = 1.5$ (c). Cross sections of the corresponding tetrahedral meshes, (d), (e) and (f), respectively.




(a) (b)

**Fig. 8** Minimum and mean quality versus time (a) and percentage of elements with poor quality versus time (b).

The combination of the surface parameterizations (quasi-conforming [21]), the adaptive strategy and the mesh optimization generally entails low distortion and values of mapping scaled Jacobian close to one. Nevertheless, in some regions of the surface, especially those close to the patch boundaries, the distortion can become high. In order to improve the mapping quality, we propose in [15] an iterative procedure that combines the refinement of the T-mesh cells with negative scaled Jacobian and the mesh optimization procedure. In this case, after four iterations we obtain a T-mesh whose all cells have positive scaled Jacobian at their center and only the 6% of cells have a scaled Jacobian less than 0.5. See [15] for more details.

## 6 Conclusions and Future Research

In this paper, we have shown important performance advantages of the meccano method. The mesh generation technique is an efficient mesh generator for creating adaptive tetrahedral meshes.

We highlight the fact that the method requires minimum user intervention and has a low computational cost to mesh

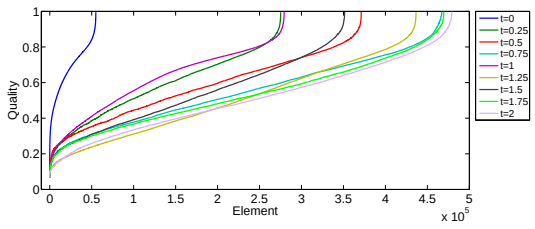
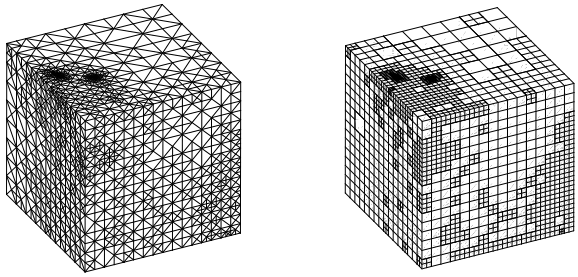
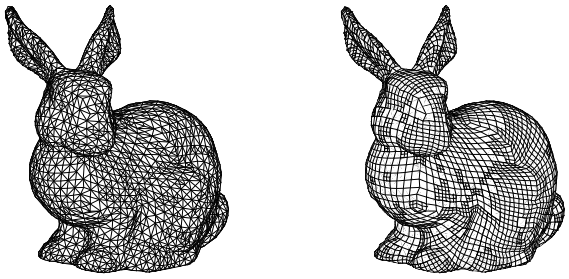



**Fig. 9** Evolution of the mesh quality curves for several times.



(a) (b)

**Fig. 10** Meccano tetrahedral mesh (a) and corresponding parametric T-mesh (b).



(a) (b)

**Fig. 11** Tetrahedral mesh of the Stanford bunny (a) and corresponding physical T-mesh (b).

solids whose boundary is a surface of genus 0. In this case, the procedure is fully automatic and it is only defined by a surface triangulation of the solid, a cube and a tolerance that fixes the desired approximation of the solid surface. We have presented a comparison between the tetrahedral meshes generated by the meccano method and other standard techniques. Particularly, a detailed comparison have been done with NETGEN and TetGen meshes. Our mesh generation strategy automatically achieves a good mesh adaption to the geometrical characteristics of the domain and the quality of the resulting meshes is high.

We have shown that the meccano meshes can be locally refined by using the Kossaczky's algorithm. Specifically, we have applied this technique in a heat evolution problem. We have noted that the local refinement/derefinement steps are very fast and the minimum mesh quality is bounded in adaptive finite element applications.

We also remark that the method simultaneously constructs a volume parameterization of complex solids. On this direction, we have introduced an application of the meccano method for the construction of volume T-mesh for isogeometric analysis.

In future works, the meccano technique can be extended to automatically mesh a complex solid whose boundary is a surface of genus greater than zero. In this case, the meccano can be a polycube or can be built by polyhedral pieces with compatible connections. At present, the user has to define the meccano associated to the solid, but we have implemented a special CAD package for more general input solid.

**Acknowledgements** This work has been supported by the Spanish Government, "Secretaría de Estado de Universidades e Investigación", "Ministerio de Economía y Competitividad", and FEDER, grant contracts: CGL2011-29396-C03-01 and CGL2011-29396-C03-03; "Junta Castilla León", grant contract: SA266A12-2. It has been also supported by CONACYT-SENER ("Fondo Sectorial CONACYT SENER HIDROCARBUROS", grant contract: 163723). Particularly, the authors thank Dr. Hang Si for providing them the new TetGen v.1.5.0 and for his kind comments and suggestions. They also thank to Dr. Joachim Schöberl for developing the NETGEN freely available code.

## References

1. Bänch E (1993) Adaptive finite element techniques for the Navier-Stokes equations and other transient problems. In: Brebbia A, Aliabad M (eds) Adaptive Finite and Boundary Elements, Computational Mechanics Publications and Elsevier, pp 47–76
2. Bazilevs Y, Calo VM, Cottrell JA, Evans J, Hughes TJR, Lipton S, Scott MA, Sederberg TW (2008) Isogeometric analysis: Toward unification of computer aided design and finite element analysis. In: Trends in Engineering Computational Technology, Saxe-Coburg Publications, Stirling, pp 1–16
3. Bazilevs Y, Calo V, Cottrell J, Evans J, Hughes T, Lipton S, Scott M, Sederberg T (2010) Isogeometric analysis using T-splines. Comput Meth Appl Mech Eng 199:229–263
4. Borouchaki H, Frey P (2005) Simplication of surface mesh using Hausdorff envelope. Comput Meth Appl MechEng 194:4864–4884
5. Carey GF (1997) Computational Grids: Generation, Adaptation and Solution Strategies. Taylor & Francis, Washington
6. Cascón JM, Montenegro R, Escobar JM, Rodríguez E, Montero G (2007) A new *meccano* technique for adaptive 3-D triangulation. In: Proc. of the 16th Interna-

tional Meshing Roundtable, Springer, Berlin, pp 103–120
7. Cascón JM, Montenegro R, Escobar JM, Rodríguez E, Montero G (2009) The meccano method for automatic tetrahedral mesh generation of complex genus-zero solids. In: Proc. of the 18th International Meshing Roundtable, Springer, Berlin, pp 463–480
8. Chen J, Zhao D, Huang Z, Zheng Y, Gao S (2011) Three-dimensional constrained boundary recovery with an enhanced Steiner point suppression procedure original research article. Computers and Structures 89:455–466
9. Cottrell J, Hughes T, Bazilevs Y (2009) Isogeometric Analysis: Towad Integration of CAD and FEA. John Wiley & Sons, Chichester
10. Eriksson K, Johnson C (1991) Adaptive finite elment methods for parabolic problems I: A linear model problem. SIAM J Numer Anal 28:43–77
11. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2003) Simultaneous untangling and smoothing of tetrahedral meshes. Comput Meth Appl Mech Eng 192:2775–2787
12. Escobar JM, Montero G, Montenegro R, Rodríguez E (2006) An algebraic method for smoothing surface triangulations on a local parametric space. Int J Num Meth Eng 66:740–760
13. Escobar JM, Rodríguez E, Montenegro R, Montero G, González-Yuste JM (2010) SUS Code: Simultaneous mesh untangling and smoothing code. http://www.dca.iusiani.ulpgc.es/proyecto2012-2014/html/Software.html
14. Escobar JM, Cascón JM, Rodríguez E, Montenegro R (2011) The meccano method for isogeometric solid modeling. In: Proc. of the 20th International Meshing Roundtable, Springer, Berlin, pp 551–568
15. Escobar JM, Cascón JM, Rodríguez E, Montenegro R (2011) A new approach to solid modeling with trivariate T-splines based on mesh optimization. Comput Meth Appl MechEng 200:3210–3222
16. Escobar J, Montenegro R, Rodríguez E, Montero G (2011) Simultaneous aligning and smoothing of surface triangulations. Engineering with Computers 27:17–29
17. Escobar J, Montenegro R, Rodríguez E, Cascón J (2012) The meccano method for isogeometric solid modeling and applications. Engineering with Computers pp 1–13, DOI 10.1007/s00366-012-0300-z, URL `http://dx.doi.org/10.1007/s00366-012-0300-z`
18. Floater MS, Hormann K (2005) Surface parameterization: a tutorial and survey. In: Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization. Springer, Berlin, pp 157–186
19. Floater MS, Pham-Trong V (2006) Convex combination maps over triangulations, tilings, and tetrahedral meshes. Advances in Computational Mathematics 25:347–356
20. Floater MS (1997) Parametrization and smooth approximation of surface triangulations. Comput Aid Geom Design 14:231–250
21. Floater MS (2003) Mean value coordinates. Comput Aid Geom Design 20:19–27
22. Freitag LA, Knupp PM (2002) Tetrahedral mesh improvement via optimization of the element condition number. Int J Num Meth Eng 53:1377–1391
23. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Num Meth Eng 49:109–125
24. Frey PJ, George PL (2000) Mesh Generation. Hermes Science Publishing, Oxford
25. George PL, Borouchaki H (1998) Delaunay Triangulation and Meshing: Application to Finite Elements. Editions Hermes, Paris
26. Knupp PM (2000) Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II - A frame work for volume mesh optimization and the condition number of the Jacobian matrix. Int J Num Meth Eng 48:1165–1185
27. Knupp PM (2001) Algebraic mesh quality metrics. SIAM J Sci Comput 23:193–218
28. Kossaczky I (1994) A recursive approach to local mesh refinement in two and three dimensions. J Comput Appl Math 55:275–288
29. Lin J, Jin X, Fan Z, Wang CCL (2008) Automatic polycube-maps. In: Lecture Notes in Computer Science, Springer, Berlin, vol 4975, p 316
30. Li B, Li X, Wang K, Qin H (2010) Generalized polycube trivariate splines. In: Proc. of the 2010 International Conference on Shape Modeling and Applications, IEEE Computer Society, pp 261–265
31. Li X, Guo X, Wang H, He Y, Gu X, Qin H (2007) Harmonic volumetric mapping for solid modeling applications. In: Proc. of ACM Solid and Physical Modeling Symposium, Association for Computing Machinery, Inc., pp 109–120
32. Martin T, Cohen E (2010) Volumetric parameterization of complex objects by respecting multiple materials. Computers and Graphics 34:187–197
33. Martin T, Cohen E, Kirby RM (2009) Volumetric parameterization and trivariate b-spline fitting using harmonic functions. Comput Aid Geom Design 26:648–664
34. Maubach J (1995) Local bisection refinement for n-simplicial grids generated by reflection. SIAM J Sci Comput 16:210–227

35. Montenegro R, Cascón JM, Escobar JM, Rodríguez E, Montero G (2009) An automatic strategy for adaptive tetrahedral mesh generation. Appl Num Math 59:2203–2217
36. Montenegro R, Cascón JM, Rodríguez E, Escobar JM, Montero G (2010) The meccano method for automatic three-dimensional triangulation and volume parametrization of complex solids. In: Developments and Applications in Engineering Computational Technology, Saxe-Coburg Publications, Stirling, pp 19–48
37. Schmidt A, Siebert KG (2005) Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA, Lecture Notes in Computer Science, vol 42. Springer, Berlin
38. Schmidt A, Siebert KG, Koster D, Kriessl O, Heine CJ (2007) ALBERTA - an adaptive hierarchical finite element toolbox, http://www.alberta-fem.de/
39. Schöberl J (1997) NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules. Comput Visual Sci 1:41–52
40. Shewchuk JR (1998) Tetrahedral mesh generation by Delaunay refinement. In: Proc. of the Fourteenth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, SCG '98, pp 86–95
41. Si H (2008) Adaptive tetrahedral mesh generation by constrained Delaunay refinement. Int J Num Meth Eng 75:857–880
42. Si H (2009) Tetgen: A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator, http://tetgen.berlios.de, v. 1.4.3. Tech. rep., Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr 39, 10117 Berlin, Germany
43. Tarini M, Hormann K, Cignoni P, Montani C (2004) Polycube-maps. ACM Trans Graph 23:853–860
44. Thompson JF, Soni B, Weatherill N (1999) Handbook of Grid Generation. CRC Press, London
45. Traxler CT (1997) An algorithm for adaptive mesh refinement in n dimensions. Computing 59:115–137
46. Wang H, He Y, Li X, Gu X, Qin H (2008) Polycube splines. Comput Aid Geom Design 40:721–733
47. Wang W, Zhang Y, Liu L, Hughes TJR (2013) Trivariate solid t-spline construction from boundary triangulations with arbitrary genus topology. Comput Aid Design 45:351–360
48. Wan S, Yin Z, Zhang K, Zhang H, Li X (2011) A topology-preserving optimization algorithm for polycube mapping. Computers and Graphics 35:639–649
49. Zhang Y, Wang W, Hughes TJR (2012) Solid t-spline construction from boundary representations for genus-zero geometry. Comput Meth Appl MechEng 249-252:185–197