INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng (2012)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4419

Tetrahedral mesh generation based on space indicator functions

Hansmartin Friess[1], Sophia Haussener[2], Aldo Steinfeld[1,3] and Jörg Petrasch[4,*,†]

[1]*Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zurich, Switzerland*
[2]*Institute of Mechanical Engineering, EPFL, 1015 Lausanne, Switzerland*
[3]*Solar Technology Laboratory, Paul Scherrer Institute, 5232 Villigen, Switzerland*
[4]*Energy Research Center, Vorarlberg University of Applied Sciences, 6850 Dornbirn, Austria*

SUMMARY

An algorithm for the generation of tetrahedral volume meshes is developed for highly irregular objects specified by volumetric representations such as domain indicator functions and tomography data. It is based on red–green refinement of an initial mesh derived from a body-centered cubic lattice. A quantitative comparison of alternative types of initial meshes is presented. The minimum set of best-quality green refinement schemes is identified. Boundary conformity is established by deforming or splitting surface-crossing elements. Numerical derivatives of input data are strictly avoided. Furthermore, the algorithm features surface-adaptive mesh density based on local surface roughness, which is an integral property of finite surface portions. Examples of applications are presented for computer tomography of porous media.

Received 26 April 2011; Revised 13 October 2011; Accepted 10 August 2012

KEY WORDS: mesh generation; flow in porous media; multiscale; Navier–Stokes

1. INTRODUCTION

Algorithms for the automatic generation of unstructured volume meshes can be divided in two classes according to the formalism used to specify the problem geometry:

1. Boundary representation (B-Rep), which is particularly suitable to describe objects resulting from human design. A prominent algorithm using B-Rep input is the advancing front algorithm [1].
2. Volumetric representation (V-Rep),[‡] typically used to describe naturally occurring structures in medicine, geology, materials science, etc.

The mesh generator described in the present paper belongs to the V-Rep class. Its development was motivated by a study of transport properties of complex porous media, where it was used to transform X-ray tomography data into tetrahedral meshes for the purpose of CFD simulations [3, 4].

A volumetric representation of three-dimensional (3D) objects can be realized by different, nearly equivalent formalisms, which are referred to by a variety of terms, partly depending on the field of application:

1. Voxel data, raster data, imaging data, or tomography data, where the object boundary is defined by a threshold value of the (multivalued) voxel intensity.

*Correspondence to: Jörg Petrasch, Energy Research Center, Vorarlberg University of Applied Sciences, 6850 Dornbirn, Austria.
[†]E-mail: joerg.petrasch@fhv.at
[‡]The term is introduced here for lack of any widely accepted generic name. Constructive Solid Geometry (CSG) has occasionally been considered to be the counterpart of B-Rep [2]. In practice, however, the term is restricted to objects resulting from human design.

2. Interval volume [5,6], similar to (1), but the object is included between two surfaces, specified by a lower and an upper threshold.
3. Segmented image or binary image, obtained after the segmentation of voxel data. This concept is particularly important in medical imaging, where it is often impossible to define an object by thresholding; rather, a specialized preprocessing algorithm is needed [7].
4. Level Set Function or Signed Distance Function [6, 8].
5. Cut Function being positive (negative) inside (outside) the object [9].
6. CSG representation (Constructive Solid Geometry), where an object is defined by means of set-theoretical operations applied to CSG primitives [2, 8, 10, 11].
7. Indicator function [12] or characteristic function [13], being equal to unity (zero) for points inside (outside) the object.

The present mesh generator is based on a user-specified indicator function (7). This concept provides maximum versatility, because formalisms (1–6) can easily be converted into (7), but not necessarily vice versa.

For a characterization of our mesh generator in the context of previous work, we are going to compile the most relevant — similar or opposed — concepts and methods.

The vast majority of V-Rep mesh generators is grid-based [14–16] or iso-voluming [17]. These algorithms all feature the following intermediate grid generation steps:

1. A cubic or tetrahedral uniform mesh is generated as basis of subsequent transformations. Uniform tetrahedral meshes have been obtained by partitioning each unit cell of a cubic lattice into five tetrahedra [5, 6, 15, 17, 18] or into six tetrahedra [19], or by Delaunay triangulation of a hexagonal close packed lattice [20] or a body-centered cubic (BCC) lattice [8, 9].
2. A template mesh (terminology: [14]) or candidate mesh [8, 21], containing only well-shaped elements and a small number of similarity classes is derived from (1). Typically, the cell size is locally adapted to the required spatial resolution.[§] In general, the template mesh is not boundary-conforming; rather, it covers the object to be discretized, with certain elements straddling the boundary surface.
3. The final, boundary-conforming mesh is derived from (2).

To a certain extent, different types of uniform meshes (1), can be combined with various techniques to perform the transitions $(1 \rightarrow 2)$ and $(2 \rightarrow 3)$, as specified in the following two paragraphs.

$(1 \rightarrow 2)$: The template mesh can be obtained by coarsening a fine uniform mesh [23], or, in the majority of approaches, by refining a coarse uniform mesh, where the following techniques have been applied:

1. A uniform cubic mesh is refined by recursively partitioning cubes into eight octants to construct an octree approximation of the object. The octree structure is ultimately transformed into a mixed mesh containing cubes, pyramids and tetrahedra [2], or in a purely tetrahedral mesh [6, 23, 24].
2. A uniform tetrahedral mesh is refined by
   a. Red–green refinement [8, 15].
   b. Insertion of element centroids, each creating four children [17, 19].
   c. The LEPP algorithm (Longest Edge Propagation Path) [18].

$(2 \rightarrow 3)$: To make a template mesh boundary-conforming, the following manipulations can be applied to boundary-crossing volume elements:

1. Splitting and deforming, either in separate passes [19, 24] or in an interlaced fashion [18].
2. Splitting only [5, 17].
3. Deforming only [8, 15]. Compared with (1) and (2), this technique has the advantage of avoiding additional nodes resulting from splitting. On the other hand, it requires to process not only boundary-crossing elements, but also elements in a wider neighborhood.

[§]Template meshes with constant cell size have also been used [5, 11, 22].

DOI: 10.1002/nme

Table I. Properties of tetrahedral meshes based on cubic lattices.

Lattice	n_{tet}	SI	q	f_0	f_3	f_∞	n_{SE}	n_{SO}
SC	5	no	0.388	0	0.175	0	19	24
			0.515	0	0.375	4/5		
			0.536	4/5	0.250	0		
			0.646	0	0.131	2/15		
			1.000	1/5	0.069	1/15		
SC	6	yes	0.515	1	1	1	8	12
FCC	24	no	0.646	2/3	2/3	2/3	7	12
			1.000	1/3	1/3	1/3		
BCC	24h	yes	0.900	1	1	1	23	48

n_{tet}, number of tetrahedra in a cubic unit cell (BCC: half tetrahedra); SI, subdivision invariance; q, tetrahedral shape quality; f_i, corresponding fraction of occurrence after i regular refinements of all mesh elements; n_{SE}, n_{SO}, number of symmetry elements and symmetry operations, respectively, for the network of edges in a cubic unit cell of the unrefined mesh.

Using the previously introduced terminology, our algorithm can be briefly characterized as follows:

1. An initial uniform mesh is derived from a BCC lattice.
2. The template mesh is obtained by red–green refinement, guided by a user-specified sizing function.
3. Boundary conformity is established by both deforming and splitting surface-crossing elements.

Section 2 provides quantitative arguments to justify the use of the BCC mesh. Section 3 describes our mesh generating algorithm step by step. Section 4 describes a method for surface-adaptive mesh density that does not use any differential operations. Section 5 presents application examples and results. Section 6 discusses potential achievements of the present work.

2. QUANTITATIVE COMPARISON OF UNIFORM TETRAHEDRAL MESHES

Most uniform tetrahedral meshes used as a starting configuration for mesh generation can be obtained by Delaunay triangulation of one of the following point lattices: Simple cubic (SC) [17–19, 25], face-centered cubic (FCC) [26], and BCC [8, 9, 27, 28].[¶] Table I compares these mesh types with respect to subdivision invariance, tetrahedral shape quality, and symmetry properties.

The advantage of subdivision invariance has been pointed out in [8]. For the mesh types compiled in Table I, either all or none of the tetrahedra are subdivision-invariant in the unrefined state ($i = 0$).

Among the large number of tetrahedral shape quality measures proposed in the literature [29], we use

$$q = 9 \cdot \frac{\text{insphere area}}{\text{circumsphere area}} \tag{1}$$

implying $q = 1$ for an equilateral tetrahedron.[‖] To analyze briefly the impact of red–green refinement on q, one can restrict oneself to regular refinement, because an element in the final mesh may be the end product of repeated regular refinements, but at most one irregular refinement. Table I (sharing some results with [31]) displays q-spectra resulting from i regular refinements of each mesh type, $i = 0, 3, \infty$. For a subdivision-invariant mesh, the q-spectrum is (trivially) independent of i. The same property holds for the FCC mesh, which is not subdivision-invariant. By contrast, the

[¶]The present discussion does not include the hexagonal close packed lattice used in [20].

[‖]Our q is also used by default in the commercial mesh generator [30]. It is equal to the square of the widely used radius ratio. It has the particular property to assume a rational number for a BCC tetrahedron, $q = 9/10$.

DOI: 10.1002/nme

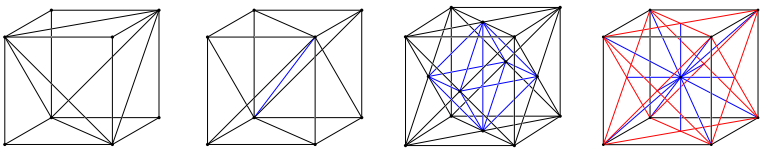

Figure 1. Network of triangulation edges in a unit cell of cubic lattices. Black dots: lattice points. Blue lines: interior edges. (a) SC, 5 tetrahedra per unit cell; (b) SC, 6 tetrahedra; (c) FCC, 24 tetrahedra; (d) BCC, 24 half tetrahedra (auxiliary red lines indicate face centers).

q-spectrum of the SC mesh with $n_{\text{tet}} = 5$ depends strongly on i, with newly emerging congruence classes ($q = 0.388; 0.515$) and congruence classes dying out when $i \to \infty$ ($q = 0.388; 0.536$).

We conclude from Table I that the BCC mesh is superior to the other mesh types with respect to q. Within the class of subdivision-invariant meshes, the BCC mesh even realizes the theoretically highest possible q-value — a fact that can be derived from a theorem by Fuchs [28].

The symmetry properties of a mesh can be important with respect to anisotropy biases of numerical solutions [8, 32]. For a simple quantitative assessment, the (unrefined) network of edges contained in a cubic unit cell (Figure 1) is considered, and the numbers n_{SE} and n_{SO} of symmetry elements and symmetry operations, respectively, mapping this network onto itself (Table I). The theoretical maximum is given by the point group of a cube, characterized by $n_{\text{SE}} = 23$ and $n_{\text{SO}} = 48$, and this maximum is realized by the BCC mesh, but by none of the other mesh types.

3. ALGORITHM

Our mesh-generating algorithm consists of four steps:

1. Construction of an initial uniform mesh
2. Red–green refinement of the initial mesh
3. Establishing boundary conformity
4. Smoothing

Sections 3.1–3.4 describe the four steps in detail.

3.1. Construction of the initial body-centered cubic mesh

The initial BCC mesh covers a user-specified bounding box of the object, similar to [8]. Apart from brute-force Delaunay triangulation, there are two explicit ways to construct a BCC mesh. They are based on

1. The two interlaced cubic grids associated with a BCC lattice [8]. The tetrahedra of the BCC mesh are spanned by pairs of nearest edges of the two grids, as shown in Figure 2.
2. A triangulation template consisting of six tetrahedra contained in a (crystallographic) primitive unit cell (Figure 3(a)). The mesh can be obtained by translational repetition of the template.

Compared with SC and FCC, the bounding box cannot be filled up exactly with regular tetrahedra (Figure 2) for the BCC lattice. In particular, if the bounding box has the same orientation as the lattice (box edges parallel to edges of cubic unit cells), then this 'matching deficiency' manifests itself on all six faces of the bounding box. The matching deficiency can be reduced, however, to two opposed faces of the bounding box by a suitable relative orientation between the box and the lattice. This improvement is particularly desirable in the case where fluid flow in a square channel containing a sample of porous material is simulated [3, 4, 33, 34]. The matching deficiency can then be limited to the inlet and outlet area of the channel, where the flow is relatively homogeneous and perpendicular to the surface.

An improved orientation between box and lattice is illustrated in Figure 3. The natural coordinate systems of the box, xyz, and the lattice, $x'y'z'$, respectively, are rotated with respect to each other through 45° around the y axis, along which the primitive unit cell has its longest extension, $2a$.

DOI: 10.1002/nme

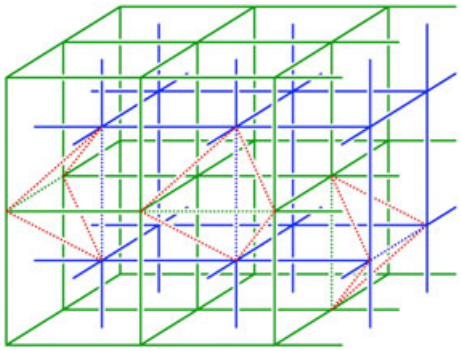

Figure 2. Triangulation of a BCC lattice based on pairs of closest edges (green/blue broken lines) in the two interlaced cubic grids (green and blue) associated with a BCC lattice. Examples of BCC tetrahedra (green/blue/red broken lines).

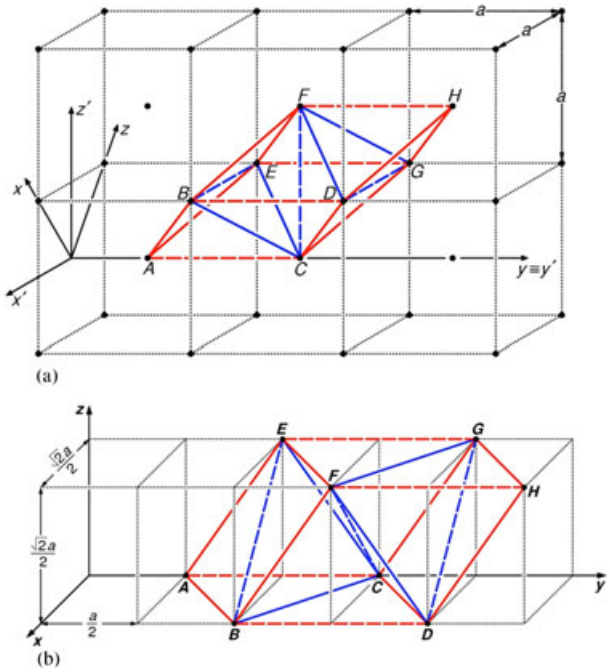

Figure 3. Triangulation of a BCC lattice by means of a template of 6 tetrahedra. (a) Template within a portion of the BCC lattice, filling up a primitive unit cell; natural coordinate system of the lattice, $x'y'z'$, and of the template, xyz. (b) Template replotted in the system xyz. Black dots: BCC lattice points. Red lines: Edges of the primitive unit cell. Blue lines: Edges added for triangulation. Long-dashed lines: Edges with dihedral angle $= 90^o$ and length $= a =$ edge length of a traditional unit cell of the BCC lattice. Solid lines: Edges with dihedral angle $= 60^o$ and length $= \sqrt{3}a/2$.

3.2. Red–green refinement of the initial body-centered cubic mesh

The technique of red–green refinement, applied to triangular meshes, goes back to [35], where red and green are synonyms for regular and irregular, respectively. Corresponding techniques in 3D have been introduced in [36, 37], and later applied or modified in [8, 15, 28, 38]. The basic purpose is to locally refine a tetrahedral mesh according to a stipulated sizing function. The transformation repeatedly applies two types of elementary operations:

1. Regular or red refinement, where a tetrahedron is partitioned into eight children with best possible shape quality. Red refinement is typically applied recursively to match the sizing function, and to impose the one level difference rule, also known as 2 : 1 rule [39].

DOI: 10.1002/nme

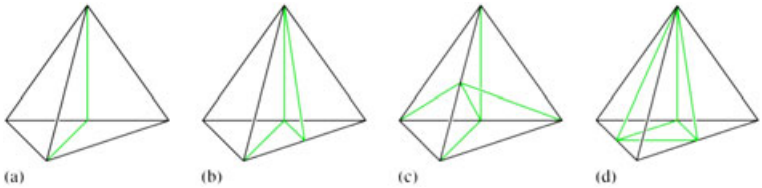

Figure 4. Green refinement patterns. (a) One edge midpoint, two tetrahedra. (b) Two midpoints on adjacent edges, three tetrahedra. (c) Two midpoints on opposed edges, four tetrahedra. (d) Three midpoints on the same face, four tetrahedra.

2. Irregular or green refinement, where a tetrahedron is partitioned into less than eight tetrahedra with reduced shape quality compared with red refinement. Green refinement eliminates hanging nodes[**] previously created by red refinements.

Green refinement schemes handling all 62 possible patterns of hanging nodes have been implemented in [38]. Most other algorithms, however, use a restricted repertoire of green refinement schemes, which requires extra red refinements — in addition to those needed to match the sizing function and the 2 : 1 rule — to reach a configuration where all patterns of hanging nodes can be handled by the restricted repertoire. The four schemes shown in Figure 4 have been proposed in [36, 37], while [8, 15] only use schemes (a), (c), and (d).

3.2.1. Red–green refinement procedure. A compact formulation of our refinement procedure is presented in Figure 5. The procedure assumes a given restricted repertoire of green refinement schemes, and it uses the following input entities:

V, T: Set of nodes, and set of tetrahedra, respectively, forming the initial BCC mesh according to Section 3.1.

n: Number of refinement levels.

TooBig: User-specified sizing function controlling the coarseness of the mesh. For any tetrahedron t, *TooBig*(t) returns a value of true if and only if t must be refined. The returned value typically depends on the following two quantities:

1. The distance of t from the boundary surface, estimated by the shortest path composed of tetrahedral edges connecting t with a boundary-crossing edge. This approximation is easy to compute, and it turns out in practice to be a satisfactory surrogate for a Level Set Function.
2. The surface structure, quantified by the local surface roughness, σ, as discussed later in Section 4.

On output, the sets V and T contain the nodes and tetrahedra, respectively, of the final mesh.

The code listed in Figure 5 internally uses the following definitions and easy-to-compute auxiliary functions:

Green Node Pattern: a set of edge midpoints of a tetrahedron corresponding to one of the green refinement schemes of the restricted repertoire. The repertoire of Figure 4 generates $6+12+3+4 = 25$ Green Node Patterns for a given tetrahedron.

HalfPoints(t) = set of six edge midpoints of a tetrahedron t (Figure 6).

QuarterPoints(t) = set of 24 edge midpoints of the triangles partitioning each face of t into four similar parts (Figure 6).

RedChildren(t) = the set of eight tetrahedra resulting from red refinement of tetrahedron t.

GreenChildren(t, G) = the set of tetrahedra resulting from green refinement of t according to a given Green Node Pattern G.

The code listed in Figure 5 can be divided into subprocedures as indicated by the upper case comment lines. The subsequent paragraphs outline each of these subprocedures in plain text, with some added comments.

[**] A hanging node is a vertex of a volume element located on the surface of a neighboring element, but different from all vertices of that element.

DOI: 10.1002/nme

```
do i = 1,n
    ! IMPOSE USER-SPECIFIED MESH GRADATION
    T_new := ∅
    do for all t ∈ T
        if diameter of t equals a/2^(i-1) then
            if TooBig(t) then
                V := V ∪ HalfPoints(t) ; T_new := T_new ∪ RedChildren(t)
            else
                T_new := T_new ∪ {t}
            end if
        end if
    end do
    T := T_new
    ! IMPOSE THE 2:1 RULE
    repeat
        V_old := V ; T_new := ∅
        do for all t ∈ T
            if QuarterPoints(t) ∩ V ≠ ∅ then
                V := V ∪ HalfPoints(t) ; T_new := T_new ∪ RedChildren(t)
            else
                T_new := T_new ∪ {t}
            end if
        end do
        T := T_new
    until V = V_old
    ! ESTABLISH GREEN CLOSABILITY
    repeat
        V_old = V ; T_new := ∅
        do for all t ∈ T
            H = V ∩ HalfPoints(t)
            if H = ∅ or H is a Green Node Pattern of t then
                T_new := T_new ∪ {t}
            else if H is a subset of some Green Node Pattern of t then
                G := the smallest Green Node Pattern of t which includes H
                V := V ∪ G ; T_new := T_new ∪ {t}
            else
                V := V ∪ HalfPoints(t) ; T_new := T_new ∪ RedChildren(t)
            end if
        end do
        T = T_new
    until V = V_old
end do
! PERFORM GREEN CLOSURE
Tnew := ∅
do for all t in T
    H := HalfPoints(t) ∩ V
    if H = ∅ then
        T_new := T_new ∪ {t}
    else
        T_new := T_new ∪ GreenChildren(t, H)
    end if
end do
T := T_new
```

Figure 5. Procedure for red–green mesh refinement.

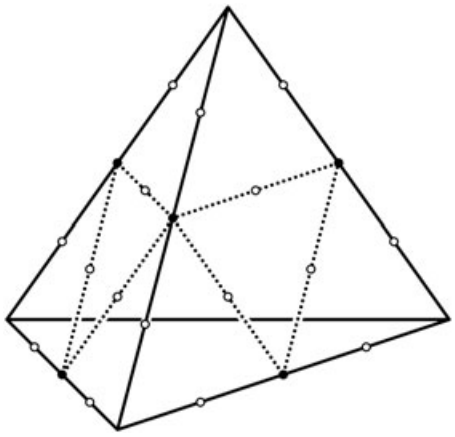

Figure 6. 'Half points' (filled circles) and 'quarter points' (empty circles) of a tetrahedron. (The points are indicated only on the two 'visible' faces.). The two sets of points are returned by the auxliary functions *HalfPoints* and *QuarterPoints*, respectively, of the red–green refinement procedure listed in Figure 5.

3.2.1.1. Impose user-specified mesh gradation. At the beginning of the i-th refinement pass, the smallest tetrahedra have a diameter $a/2^{i-1}$ (a = length of a cubic unit cell of the BCC lattice,

DOI: 10.1002/nme

see Figure 3). For every tetrahedron t with this particular diameter, the i-th refinement pass calls up the sizing function, *TooBig*(t), and t is refined regularly if the function returns a value of true.

3.2.1.2. Impose the 2 : 1 rule. All tetrahedra are inspected iteratively. If a tetrahedron t has an 'undersized' neighbor (i.e., a tetrahedron sharing more than one point with t, and having a diameter less than half the diameter of t), then t is regularly refined. The iteration proceeds until diameters of neighboring tetrahedra never differ by more than a factor 2. The presence of an undersized neighbor of t always manifests itself by a quarter point of t (Figure 6) being contained in the current set of nodes, V.

3.2.1.3. Establish green closability. All tetrahedra are inspected iteratively. For every tetrahedron t, the hanging nodes appearing on its surface are determined. A hanging node manifests itself by the presence of an edge midpoint of t in the current set of nodes, V. Three cases are distinguished:

1. There are no hanging nodes at all, or the hanging nodes represent a Green Node Pattern. In this case, no action is required.
2. The set of hanging nodes can be extended to a Green Node Pattern. In this case, the required additional nodes are added to the current set of nodes, V (without any other action).
3. None of the previous two conditions is true. In this case, t is refined regularly (contributing to the 'extra red refinements' mentioned in Section 3.2.1).

The iteration proceeds until every tetrahedron either has no hanging nodes on its surface at all, or the hanging nodes represent a Green Node Pattern.

3.2.1.4. Perform green closure. Every tetrahedron is inspected once. If there are hanging nodes on its surface, these nodes are guaranteed to represent a Green Node Pattern, and the corresponding green refinement scheme is applied.

Note that our algorithm does not need any data structure for connectivity. Rather, all required connectivity information is inferred simply from the presence (or absence) of 'half points' or 'quarter points' (Figure 6) in the current set of nodes, V. The corresponding code is simplified by the fact that all occurring coordinates are integer multiples of $a/2^{n+1}$ or $\sqrt{2}a/2^{n+1}$ (see Figure 3). This fact does not exclude, however, a generalization of the algorithm to arbitrary, maybe nonuniform initial meshes.

3.2.2. Identification of the minimum, best-quality green refinement repertoire. According to [8,37], the following reasons suggest the use of a restricted repertoire of green refinement schemes: (a) simplicity, (b) enhanced quality of 'green children', and (c) implicit smoothing of the mesh gradation. On the basis of these arguments, it seems logical to look for the repertoire with a minimum number of best-quality schemes.

The need for extra red refinements as a consequence of a restricted repertoire (Section 3.2.1; Section 3.2.2, subprocedure Section 3.2.1.3) has been addressed as domino effect [37, 38]. The question to be clarified here is whether a restricted repertoire entails an unacceptably strong domino effect, or even a catastrophic domino effect producing a final mesh of constant cell size, corresponding to the finest level of refinement.

In a very large number of computational experiments with highly irregular objects, we have found that a restricted repertoire of only two green refinement schemes, (a) and (d) in Figure 4, has never created a significant domino effect. On the other hand, a repertoire with only one scheme almost always leads to a catastrophic domino effect.

Table II displays tetrahedral shape qualities resulting from the green refinement schemes of Figure 4 applied to a BCC tetrahedron. The fraction of occurrence indicated in the table is based on the assumption that all Green Node Patterns generated by one of the schemes in Figure 4 are equally probable.

Table II and the preceding remarks about the domino effect suggest that schemes (a) and (d) in Figure 4 represent a minimum, best-quality set of green refinement schemes for a BCC mesh.

DOI: 10.1002/nme

Table II. Green refinement patterns applied to a regular BCC tetrahedron —-corresponding to the patterns (a–d) in Figure 4.

n_M	Constraint	n_{tet}	q	f
2	none	2	0.474	1/3
			0.491	2/3
2	same face	3	0.206	1/9
			0.296	2/9
			0.324	1/9
			0.474	1/9
			0.491	2/9
			0.508	2/9
2	opposed edges	4	0.304	1/3
			0.414	1/3
			0.515	1/3
2	same face	4	0.324	1/4
			0.508	1/2
			0.513	1/4

n_M, number of edge midpoints; n_{tet}, number of green tetrahedra; q, f, shape quality of green tetrahedra and corresponding fraction of occurrence.

3.3. Establishing boundary conformity

To make a template mesh boundary-conforming, a frequently used technique is to split boundary-crossing elements. The method is liable to sliver elements, because nodes of the template mesh can fall arbitrarily close to the boundary surface. To suppress sliver elements, splitting has been combined with various flanking measures [18, 22, 24].

Our algorithm reaches boundary conformity in two successive passes. Pass 1 shifts selected nodes to the boundary surface; it largely eliminates configurations where splitting would produce sliver elements. Pass 2 splits boundary-crossing elements left over from Pass 1. The two passes are described below in more detail.

3.3.1. Pass 1. All nodes in the mesh are initially flagged by IN or OUT according to the indicator function. Edges with differently flagged end points are considered as boundary-crossing. The end points of these edges are considered as candidates for shifting. The candidates are processed in the order of ascending distance d from the boundary. Each candidate is shifted to the nearest boundary point, provided that the quality of the incident tetrahedra does not drop below a certain limit. Shifted nodes are flagged as NIO (neither in nor out, similar notation as in [2]).[††]

3.3.2. Pass 2. Each boundary-crossing tetrahedron is split according to one of the schemes depicted in Figure 7. As opposed to [17, 19], we take into account that the four edge intersections in Case 6, C, D, E, F, are not coplanar in general. A reasonably simple and accurate representation of the true surface is achieved by using the center, M, of the four intersection points as an additional node.

To convert pyramids and wedges making up the clipped volume elements into tetrahedra, quadrilateral faces are always divided along the shorter diagonal. This rule preserves mesh consistency without needing any connectivity information. With given diagonals of quadrilateral faces, there is a chance of 3/4 that a wedge can be partitioned into three tetrahedra. If this is not possible, the wedge is partitioned into eight tetrahedra by insertion of a Steiner point, like in [17, 19].

[††]Processing the candidates in the order of ascending d improves the final mesh quality significantly. The strategy avoids situations where a candidate X with very small d — liable to produce sliver elements when split — is 'blocked' by a candidate Y with moderate d. If Y were processed prior to X, the local mesh quality could be degraded so that X cannot be shifted.

DOI: 10.1002/nme

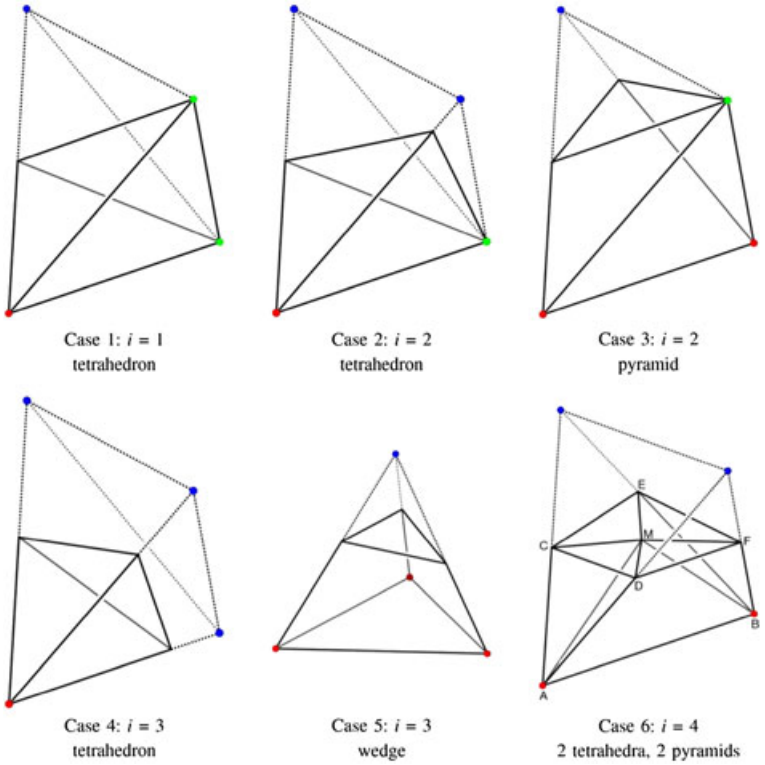

Figure 7. Splitting schemes used to establish boundary conformity. Red/green/blue dots: vertices with attribute IN/NIO/OUT. Solid and broken lines: edge portions inside and outside the boundary surface, respectively. i = number of edge intersection points. The constituents of the clipped volume element (tetrahedra, pyramids, wedges) are also indicated for each case.

3.4. Smoothing

In the the last step of our mesh generation algorithm, optimization-based smoothing is applied, which is one of several methods to improve the mesh quality [6, 40]. The corresponding iterative procedure minimizes the objective function

$$P = \sum_t \frac{1}{q(t)^m} \tag{2}$$

by variation of all node coordinates, without affecting the connectivity. Appropriate constraints for nodes on surfaces are used. Relocation of a node X is suppressed whenever it would lead to degeneration or inversion of any element connected to X. The summation in Equation (2) is extended over all tetrahedra t of the mesh. The parameter m is empirical. An analysis of quality statistics suggests an optimal value of $m \approx 3$. The function $q(t)$ is defined by Equation (1). We have found empirically that the present optimization-based smoothing procedure yields significantly better results than Laplacian smoothing [15, 17, 24] in accordance with previous work [41, 42] based on objective functions different from Equation (2).

4. SURFACE-ADAPTIVE MESH DENSITY

To adapt the mesh density locally to the length scale of relevant surface structures, surface curvature can be used to control mesh refinement [8, 18]. Because this concept involves differential operations, it is liable to unspecific fluctuations when applied to tomography data. To avoid this problem, our algorithm uses an integral property, σ, of finite portions of the surface, rather than a

DOI: 10.1002/nme

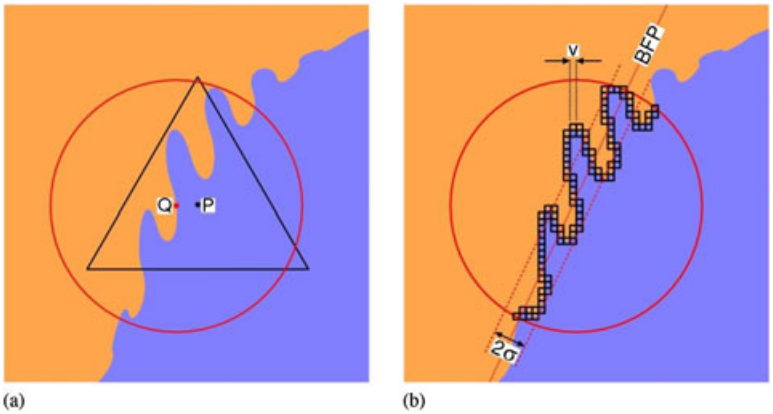

Figure 8. Algorithmic steps in the computation of the local surface roughness, σ. (a) Tetrahedron t being considered for red refinement (black triangle), with center P. Sphere (red circle), with center Q on the surface (blue–brown boundary), with same diameter as t, and minimized distance between Q and P. (b) Representation of the surface sample inside the sphere by a set of voxels (black squares) of size v. Best fitting plane, BFP, of the voxel set. Root mean square deviation, σ, of voxel positions relative to BFP.

differential expression, to control the mesh density on surfaces.[‡‡] The quantity σ is associated with every tetrahedron t being considered for red refinement (Section 3.2.1); it can be referenced by the user-specified sizing function, *TooBig*(t), which may return a value of true if σ exceeds a critical threshold.

The quantity σ is defined as follows: If a tetrahedron t is not intersected by the surface, then $\sigma = 0$. Otherwise, σ results from the following algorithmic steps (illustrated in Figure 8):

1. Find a sphere having its center Q located on the surface, as close as possible to the center of t, and having the same diameter as t.
2. Represent the surface portion inside the sphere by a set of n voxels with positions x_i and a user-specified size v.
3. Find the best fitting plane (BFP) of the voxel set (in the sense of minimum root mean square (RMS) deviation). Represent BFP by the equation

$$\mathbf{e} \cdot \mathbf{x} = g, \tag{3}$$

   where $\mathbf{e}$ is a unit vector perpendicular to BFP.
4. Let σ be equal to the RMS deviation of voxel positions from BFP

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\mathbf{e} \cdot \mathbf{x}_i - g)^2}. \tag{4}$$

This computational recipe (in particular, the constraint for Q in Step 1) provides a smooth dependence of σ on the position and orientation of surface-crossing tetrahedra. In the special case of a spherical surface, σ assumes a constant value for equally sized surface-crossing tetrahedra.

Equation (4) is formally identical with the usual definition of surface roughness, if BFP is considered as a reference plane. Thus, σ can be interpreted as local surface roughness (where 'local' means: in a region close to, and similar in size with a tetrahedron considered for refinement).

The computation of σ in terms of the $\mathbf{x}_i = (x_i, y_i, z_i)$ does not require explicit knowledge of $\mathbf{e}$. Rather, σ can be obtained by

$$\sigma = \sqrt{\lambda_{\min}} \quad ,$$

[‡‡]To some extent, unspecific fluctuations can also be suppressed by filtering, with a corresponding degradation of surface representation. The present approach reduces the need for filtering.

DOI: 10.1002/nme

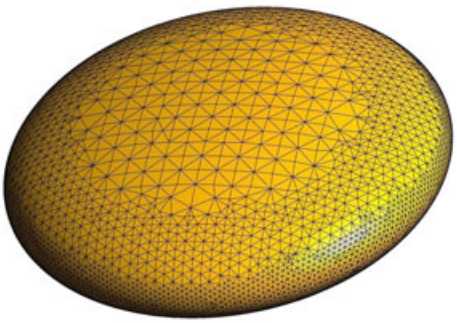

Figure 9. Surface-adapted mesh density controlled by local surface roughness.

where $\lambda_{\min}$ is the least eigenvalue of the matrix

$$C = \begin{pmatrix} a & d & f \\ d & b & e \\ f & e & c \end{pmatrix}$$

with

$$a = \sum_i (x_i - \bar{x})^2 \quad b = \sum_i (y_i - \bar{y})^2 \quad c = \sum_i (z_i - \bar{z})^2$$

$$d = \sum_i (x_i - \bar{x})(y_i - \bar{y}) \quad e = \sum_i (y_i - \bar{y})(z_i - \bar{z}) \quad f = \sum_i (x_i - \bar{x})(z_i - \bar{z})$$

$\bar{x}, \bar{y}, \bar{z} =$ average values of x_i, y_i, z_i, respectively.

To illustrate the effect of our procedure, Figure 9 shows mesh edges on the surface of a spheroid. The result is similar to what one would expect from curvature-controlled mesh density.

A weakness of the current implementation is its relatively large computation time, being proportional to $(c/\mathrm{v})^3$, where c is the linear cell size and v is the linear voxel size (see Figure 8).

5. APPLICATIONS AND RESULTS

Three applications of the mesh generator and the resulting meshes are presented. The morphology of (i) reticulate porous ceramics (RPC), (ii) anisotropic ceramic foams (ACF), and (iii) packed beds of highly porous particles (PB) are obtained by computed tomography (CT) and used in combination with the mesh generator to produce computational meshes of the complex fluid phase structure (on the pore-level) for direct numerical simulations with commercial CFD solvers such as ANSYS

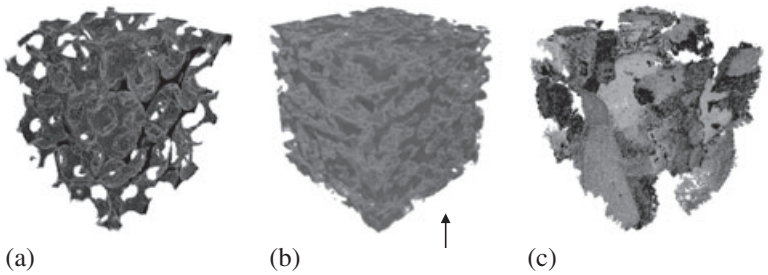

Figure 10. 3D rendering of computed tomography data of (a) a cubical sample of RPC with edge length of 6 mm, (b) an anisotropic ceramic foam with edge length of 0.37 mm, where the arrow indicates the direction of uniaxial pressing, and (c) a packed bed of highly porous media with an edge length of 3.42 mm.

DOI: 10.1002/nme

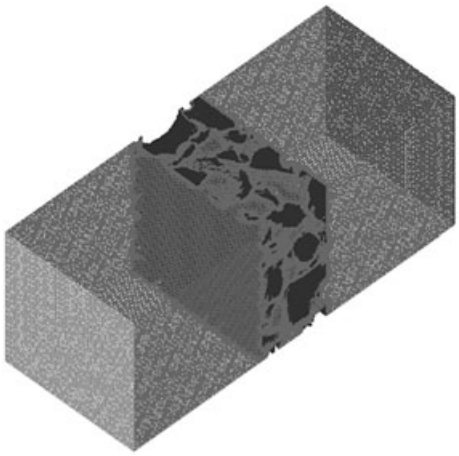

Figure 11. Schematics of the computational domain, consisting of a square channel containing a sample of porous material.

Table III. CT resolution, volume of mesh (including inlet and outlet region) V, sample's porosity, number of tetrahedral elements n_{tet}, mean tetrahedral shape quality, $\bar{q}$, and mean mesh element diameter, $\bar{D}$, for the RPC, ACF, and PB samples.

	CT resolution (μm)	V (mm^3)	Sample porosity	n_{tet}	$\bar{q}$	$\bar{D}$ (μm)
RPC	30	3462	0.91	$26.2{\cdot}10^6$	0.7263	103.3
ACF	0.37	0.047	0.51	$105.9{\cdot}10^6$	0.7185	1.8
PB	3.7	238	0.86	$33.5{\cdot}10^6$	0.7195	34.4

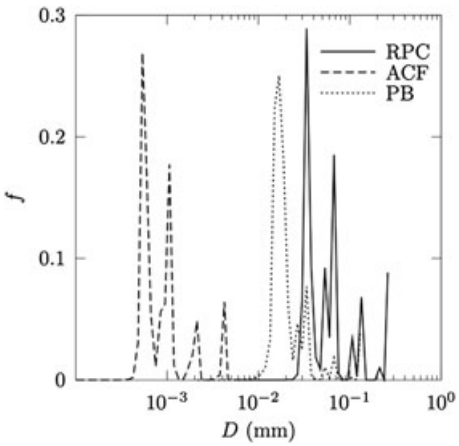

Figure 12. Mesh element size distribution for the RPC, ACF, and PB computational meshes in terms of the element diameter D (largest edge length).

and Fluent [30,43]. This approach allowed for a complete characterization of the morphological and effective heat/mass transport properties of RPC [44], ACF [45], and PB [46]. A microfocus X-ray tube was used to obtain low-resolution CT data (voxel size[§§] $= 30\mu$m) of silicon carbide RPC with nominal pore sizes of 1.27 mm. A 3D rendering of a subvolume is depicted in Figure 10(a). High-resolution computed tomography data (voxel size $= 0.37$ and 3.7μm) obtained with synchrotron radiation was used for the 3D renderings of ACF made of ceria with pore diameters between 10

[§§]Voxel size corresponds to the edge length of the smallest resolvable, cubical (3D) volume

DOI: 10.1002/nme

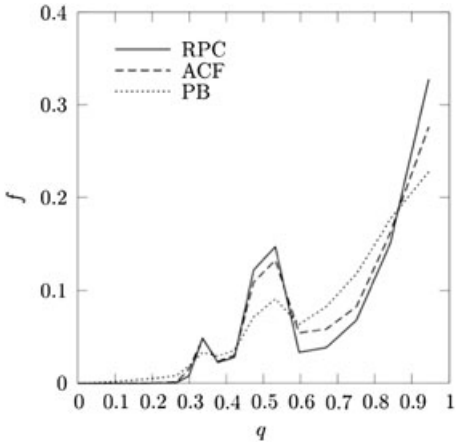

Figure 13. Distribution of tetrahedral shape quality, q, of the RPC, ACF, and PB computational meshes.

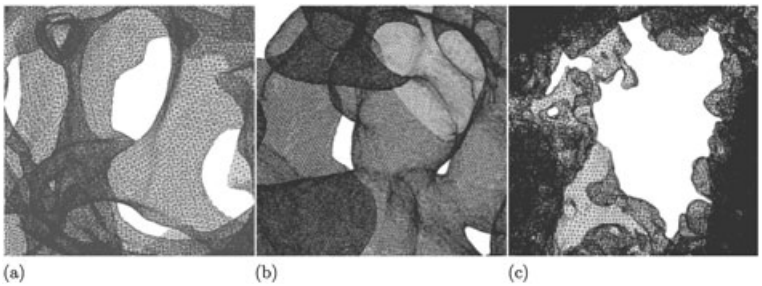

Figure 14. Surface mesh of the solid–fluid interphase of (a) the RPC with strut thickness of approximately 150 μm and mean mesh element length of 103 μm, (b) the ACF with pore size of approximately 30 μm and mean mesh element length of 2 μm, and (c) PB with particle sizes of 500 μm radius and mean mesh element length of 34 μm.

and 100 μm, and packed beds of highly porous 1-mm-diameter carbonaceous particles. These two material samples are depicted in Figures 10(b) and (c), respectively.

Each computational mesh represents a square channel containing a sample of porous material and an inlet and an outlet region, both having approximately twice the volume of the porous sample (Figure 11). Grid convergence studies were conducted to determine the mesh density for which the discretization error of computational results becomes negligible. Corresponding mesh characteristics are given in Table III and by Figures 12 and 13. In the element size distribution (Figure 12), the four distinct peaks reflect the setting $n = 4$ (number of refinement levels, Section 3.2.1). In the distributions of tetrahedral shape quality (Figure 13), the peaks at $q \approx 0.3$ and $q \approx 0.5$ correspond to the dominant q values resulting from green refinement (Table II), while the peak at $q = 0.9$ belongs to the bulk of unrefined BCC tetrahedra.

Close-ups of mesh edges on the solid–fluid phase boundary for RPC, ACF, and PB samples are shown in Figure 14. The solid–fluid phase boundary is accurately resolved (within the limits given by the CT resolution).

6. SUMMARY AND CONCLUSIONS

A basic design concept of our mesh generation algorithm was a reasonable compromise between mesh quality on one hand, and robustness and simplicity on the other hand. Among all published algorithms, that of [8] is perhaps the one that is most similar to ours, given that it also uses a BCC mesh in combination with red–green refinement.

DOI: 10.1002/nme

Important qualitative arguments for giving preference to the BCC mesh have been put forward in [8]. The present paper provides additional, quantitative arguments by comparing the most frequently used mesh types with respect to tetrahedral shape quality and degree of symmetry (Section 2).

Our algorithm differs from that of [8] in four major points: (i) In the applications shown in [8], an object to be meshed has virtually no surface region in common with the object's bounding box covered by the initial, uniform BCC mesh. In this situation, it is both natural and efficient to have the edges of the bounding box parallel to the edges of the cubic unit cells of the BCC lattice. In contrast, our typical application is a sample of porous material, where the volume to be discretized may have a large surface fraction in common with the volume's bounding box. To improve the mesh quality along this surface fraction, we have identified the optimal relative orientation between the BCC lattice and the bounding box (Section 3.1). (ii) Instead of three green refinement schemes used in [8], we use only two. We have shown that these two schemes represent a minimum, best-quality repertoire green refinement scheme (Section 3.2.2). (iii) Given that our typical input data originate from tomographic scans affected by fluctuating errors, our algorithm completely avoids operations that imply numerical differentiation. Instead of a Level Set Function [8], a fast and robust approximation is used to have the mesh density controlled by the distance from the object surface. Instead of curvature [8], we use the concept of local surface roughness to adapt the mesh density to the surface structure (Section 4). (iv) To establish boundary conformity, our algorithm deals only with surface-crossing elements, while the algorithm of [8] deforms the mesh in a wider neighborhood of the surface. The relative simplicity of our algorithm entails the necessity of splitting some of the boundary-crossing elements, which introduces additional nodes and degrades tetrahedral shape quality.

Currently, isotropic meshes are considered desirable because this represents the most standard case (which also applies to all mesh-generating algorithms cited in the present paper). Anisotropic elements close to boundaries that are advantageous for certain flow conditions have not been considered within this work.

Coupling this mesh generator to CT data allows for the construction of accurate, high quality, and robust meshes, which have been successfully used for numerically solving the governing mass, momentum and energy conservation equations on the pore-level and the subsequent extraction of the effective heat and mass transport properties of the porous media to be used in volume averaged (continuum) models.

LIST OF SYMBOLS

a	Edge length of a cubic unit cell of a BCC lattice; largest tetrahedral edge in the initial BCC mesh
D	Diameter (largest edge) of a tetrahedron
n	Number of refinement levels in the procedure for red–green refinement
q	Tetrahedral shape quality (insphere to circumsphere ratio)
T	Set of tetrahedra maintained in the procedure for red–green refinement
V	Set of nodes maintained in the procedure for red–green refinement
σ	Local surface roughness

ACKNOWLEDGEMENTS

This work was financially supported in part by the Swiss National Science Foundation under contract No. 200021-115888 and by the European Commission under contract No. 212470.

REFERENCES

1. Lohner R, Parikh P. Generation of 3D unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
2. Perucchio R, Saxena M, Kela A. Automatic mesh generation from solid models based on recursive spatial decompositions. *International Journal for Numerical Methods in Engineering* 1989; **28**:2469–2501.

DOI: 10.1002/nme

3. Petrasch J. Multi-scale analyses of reactive flow in porous media. *Dissertation*, Eidgenössische Technische Hochschule Zürich, Diss. ETH No. 17192, 2007.
4. Petrasch J, Meier F, Friess H, Steinfeld A. Tomography based determination of permeability, Dupuit-Forchheimer coefficient, and interfacial heat transfer coefficient in reticulate porous ceramics. *International Journal of Heat and Fluid Flow* 2008; **29**:315–326.
5. Nielson GM, Sung J. Interval volume tetrahedrization. *Proceedings of IEEE Visualization'97*, Phoenix AZ, Oct 19–24, 1997; 221–228.
6. Zhang Y, Bajaj C, Sohn B-S. 3D finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering,* Pittsburg PA*,* Oct 12-15, 2005; **194**:5083–5106.
7. Fedorov A, Chrysochoides N. Tetrahedral mesh generation for brain MRI. *Proceedings of the 17th International Meshing Roundtable*, 2008; 55–72.
8. Teran J, Molino N, Fedkiw R, Bridson R. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers* 2005; **21**:2–18.
9. Labelle F, Shewchuck JR. Isosurface Stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics* 2007; **26**:57-1–57-10. No. 3, Article 57.
10. Requicha AAG, Voelcker HB. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications* 1982; **2**:9–24.
11. Frey P, Sarter B, Gautherie M. Fully automatic mesh generation for 3D domains based upon voxel sets. *International Journal for Numerical Methods in Engineering* 1994; **37**:2735–2753.
12. Torquato SD, Pham DC. Optimal bounds on the trapping constant and permeability of porous media. *Physical Review Letters* 2004; **92**.
13. Manwart C, Aaltosalmi U, Koponen A, Hilfer R, Timonen J. Lattice-boltzmann and finite-difference simulation for the permeability for three-dimensional porous media. *Physical Review E* 2002; **66**:016702–016713.
14. Ho-Le K. Finite element mesh generation methods: a review and classification. *Computer-Aided Design* 1988; **20**(1):27–38.
15. Sullivan JM, Charron G, Paulsen KD. A three-dimensional mesh generator for arbitrary multiple material domains. *Finite Elements in Analysis and Design* 1997; **25**:219–241.
16. Owen SJ. A Survey of unstructured mesh generation technology. *Proceedings of the 7th International Meshing Roundtable*, Dearborn MI, Oct 26–28, 1998.
17. Ferrand M. Registration of 3-D intraoperative MR images of the brain using a finite-element biomechanical model. *IEEE Transactions on Medical Imaging* 2001; **20**:1384–1397.
18. Mohamed A, Davatzikos C. Finite element mesh generation and remeshing from segmented medical images. *Proceedings of IEEE International Symposium on Biomedical Imaging*, Arlington, VA, USA, April 15–18, 2004; 420–423.
19. Hamann B, Thornburg HJ, Hong G. Automatic unstructured grid generation based on iterative point insertion. *Computing* 1995; **55**:135–161.
20. Mello UT, Cavalcanti PR. A point creation strategy for mesh generation using crystal lattices as templates. *Proceedings of the 9th International Meshing Roundtable*, New Orleans, Louisiana, USA, October 2-5, 2000.
21. Molino N, R Bridson, Teran J, Fedkiw R. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. *Proceedings of the 12th International Meshing Roundtable*, Santa Fe NM, Sept 14–17, 2003.
22. Wang K, Denney TS, Morrison EE, Vodyanoy VJ. Construction of volume meshes from computed tomography data. *Proceedins of the 2005 IEEE Engineering in Medicine and Biology. 27th Annual Conference*, Shanghai, China, September 1-4, 2005.
23. Hartmann U, Kruggel F. A fast algorithm for generating large tetrahedral 3D finite element meshes from magnetic resonance tomograms. *Proceedings of the IEEE Workshop on Biomedical Image Analysis*, Santa Barbara, California, June 1988.
24. Yerry MA, Shephard MS. Automatic 3D mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering* 1984; **20**:1965–1990.
25. Montenegro R, Cascon JM, Escobar JM, Rodriguez E, Montero G. An automatic strategy for adaptive tetrahedral mesh generation. *Applied Numerical Mathematics* 2009; **59**:2203–2217.
26. Radovitzky R, Ortiz M. Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-Delaunay triangulation. *Computer Methods in Applied Mechanics and Engineering* 2000; **187**:543–569.
27. Field DA, Smith WD. Graded tetrahedral finite element meshes. *International Journal for Numerical Methods in Engineering* 1991; **31**:413–425.
28. Fuchs A. Automatic grid generation with almost regular delaunay tetrahedra. *Proceedings of the 7th International Meshing Roundtable*, Dearborn MI, Oct 26–28, 1998; 133–147.
29. Shewchuk J. What is a good linear element? Interpolation, conditioning, and quality measure. *Proceedings of the 11th International Meshing Roundtable*, Ithaca NY, Sept 15–18, 2002; 115–126.
30. Ansys Inc. Ansys 12.1, 2009.
31. Naylor DJ. Filling space with tetrahedra. *International Journal for Numerical Methods in Engineering* 1999; **44**:1383–1395.
32. Jirasek M, Grassl P. Evaluation of directional mesh bias in concrete fracture simulations using continuum damage models. *Engineering Fracture Mechanics* 2008; **75**:1921–1943.

DOI: 10.1002/nme

33. Fourie W, Said R, Young P, Barnes DL. The simulation of pore scale fluid flow with real world geometries obtained from x-ray computed tomography. *Proceedings COMSOL Conference*, Boston, 2007.
34. Tabor G, Yeo O, Young P, Laity P. CFD simulation of flow through an open cell foam. *International Journal of Modern Physics C* 2008; **19**:703–715.
35. Bank RE, Sherman AH, Weiser A. Refinement algorithms and data structures for regular local mesh refinement. In *Scientific Computing*, Stepleman R (ed.). IMACS/North Holland Publishing Company: Amsterdam, 1983; 3–17.
36. Bornemann F, Erdmann B, Kornhuber R. Adaptive multilevel methods in three space dimensions. *International Journal for Numerical Methods in Engineering* 1993; **36**:3187–3203.
37. Bey J. Tetrahedral grid refinement. *Computing* 1995; **55**:355–378.
38. Grosso R, Greiner G. Hierarchical meshes for volume data. *Computer Graphics International 1998 (CGI'98)*, Hannover, Germany, June 22-June 26.
39. Shephard MS, Georges MK. Automatic 3D mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**:709–749.
40. Teng S-H, Wong CW. Unstructured mesh generation: theory, practice and perspectives. *International Journal of Computational Geometry & Applications* 2000; **10**(3):227–266.
41. Freitag LA, Ollivier-Gooch C. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering* 1997; **40**:3979–4002.
42. Chen Z, Tristano JR, Kwok W. Construction of an objective function for optimization-based smoothing. *Engineering with Computers* 2004; **20**:184–192.
43. Ansys Inc. Fluent 12.0.16, 2009.
44. Haussener S, Coray P, Lipiński W, Wyss P, Steinfeld A. Tomography based heat and mass transfer characterization of reticulate porous ceramics for high-temperature processing. *Journal of Heat Transfer* 2010; **132**:023305.
45. Haussener S, Steinfeld A. Effective heat and mass transport properties of porous ceria for solar-thermal fuel generation. *Materials* 2012; **5**:192–209.
46. Haussener S, Jerjen I, Wyss P, Steinfeld A. Tomography-based determination of effective transport properties of reacting porous media. *Journal of Heat Transfer* 2012; **134**:012601.

DOI: 10.1002/nme