 CMES, vol.95, no.2, pp.119-141, 2013

Novel graph-based adaptive triangular mesh refinement for finite-volume discretizations

Sanderson L. Gonzaga de Oliveira[1], **Mauricio Kischinhevsky**[2] **and João Manuel R. S. Tavares**[3]

Abstract: A novel graph-based adaptive mesh refinement technique for triangular finite-volume discretizations in order to solve second-order partial differential equations is described. Adaptive refined meshes are built in order to solve time-dependent problems aiming low computational costs. In the approach proposed, flexibility to link and traverse nodes among neighbors in different levels of refinement is admitted; and volumes are refined using an approach that allows straightforward and strictly local update of the data structure. In addition, linear equation system solvers based on the minimization of functionals can be easily used; specifically, the Conjugate Gradient Method. Numerical and analytical tests were carried out in order to study the required execution time and the data storage cost. These tests confirmed the advantages of the approach proposed in elliptic and parabolic problems.

Keywords: Adaptive mesh refinement, mesh generation, Sierpiński Curve, elliptic and parabolic problems, non-conformal mesh.

1 Introduction

The Finite Volume Method (FVM) represents and evaluates partial differential equations (PDEs) as algebraic equations. Using the FVM, the PDEs values are calculated at polytopes on a meshed geometry resulting in a linear equation system to be computed.

Techniques of adaptive mesh refinement (AMR) are commonly used to localize a large number of polytopes in specific regions of the mesh, allowing a coarse level

[1] Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, MG, Brazil, sanderson@dcc.ufla.br

[2] Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brazil, kisch@ic.uff.br

[3] Instituto de Engenharia Mecânica e Gestão Industrial, Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, tavares@fe.up.pt

of refinement in the remaining mesh. Increasing appropriately the number of polytopes in certain regions of the mesh to be used within the FVM can have two main advantages: *i)* it allows to maintain the accuracy of the original solution and; *ii)* the resulting linear equation system is smaller than a uniform refined mesh; hence, low computational costs and low storage costs are achieved. In evolutionary problems, such refinement becomes particularly important because of their dynamic nature; i.e., the variables may have either their value or domain modified only in certain regions, whereas may be relatively stable in other parts. This occurs, for example, in the study of turbulence in fluid flows or in solid modeling. In summary, techniques of AMR have been commonly used to provide numerical solutions of PDEs with low computational costs.

The studies about AMR began in the 1970s, but AMR remains a very active area of research; recent examples include the works by Nie, Li, and Wang (2012), Goffin, Baker, Buchan, Pain, Eaton, and Smith (2013) and Baker, Buchan, Pain, Tollit, Goffin, Merton, and Warner (2013). Burgarelli, Kischinhevsky, and Biezuner (2006) proposed a data structure for representing adaptive *square*-shaped meshes: the Autonomous Leaves Graph (ALG), which represents an adaptive square-mesh refinement coupled with a finite-volume solver of PDEs. Regarding triangulations, Velho, de Figueiredo, and Gomes (1999) described that a triangular mesh is a 2D simplicial complex, i.e. a simple structure with convenient combinatorial properties. In addition, simplicial meshes can be related to general meshes and the Finite Volume Method can be applied with triangular meshes to solve problems with complex geometries.

Here, the meshes addressed are related to simplicial complex structures in nonconform meshes for finite volume discretization, which means that hanging nodes are allowed. Additionally, the generation of the triangular discretization is not an independent step of the solution process, but a dynamic adaptive process. Moreover, the algorithms of AMR maintain the non-degeneracy of the triangular control-volumes and the meshes are always smooth.

A graph data structure was especially designed to represent the triangular AMR in the proposed solution. In relation to the total ordering, a Hamiltonian triangulation is generated with respect to the Hamiltonian ordering found. The aim of maintaining a linear ordering during the triangular AMR is especially important so that all control volumes (where the variables are evaluated) can be traversed in linear time; hence, the linear equation systems can be built with low computational cost.

Additionally, linear equation system solvers based on the minimization of functionals can be easily employed with the proposed solution. Furthermore, a graph-based triangular AMR scheme that provides all the requirements for solution of PDEs was developed. It is based on cell-centered triangular finite-volume approx-

imations through an extended finite-volume approach proposed by Schneider and Maliska (2002) for the solution of PDEs with low computational cost.

In the following, data structures and related algorithms that have been proposed to construct a dynamic grid in evolutionary problems are described. The 4-triangle longest-edge partition of triangular volumes, the triangular AMR scheme, and the graph data structure used, including the refinement and coarsening processes, are explained in Section 2. The space-filling curve employed in order to traverse the mesh nodes and the mesh total ordering process are depicted in Section 3. The experimental tests are presented in Section 4. The complexity of the space-filling curves applied in ALG and in the approach proposed are discussed in Section 5. The storage cost of the data structure used for representing adaptive meshes is addressed in Section 6. Finally, additional remarks are presented in Section 7.

2 Graph-based triangular meshes

A new graph-based triangular AMR technique with triangular cell-centered volumes based on ALG (Burgarelli, Kischinhevsky, and Biezuner (2006)) is described in this section. Additionally, the process of graph simplification that may occur during the computation, which includes refinement and coarsening of mesh volumes, is also presented. This graph-based triangular AMR technique extends the *square* finite volume discretization proposed by Burgarelli, Kischinhevsky, and Biezuner (2006) to the *triangular* finite-volume discretization.

2.1 Graph-based AMR technique with triangular cell-centered volumes

A graph explicitly provides all the relations among volumes, being the triangular volumes with their vertices, edges and other information represented by graph nodes. Two nodes are directly connected if they represent two triangles with a common edge. When a triangular volume is refined, an original volume node is replaced by a new subgraph comprised of: *i*) 4 *volume nodes*, *ii*) 3 *transition nodes* and *iii*) their links. In the local refinement of each triangular volume, the original nodes are discarded for memory saving, and only the generated volume nodes representing the 4 new triangular volumes and the 3 required transition nodes in the created subgraph pack are stored. In addition, the level of volume nodes of the new pack is increased of 1 (one).

Transition nodes indicate the refinement *level* of the volume in relation to their neighbor volumes. Such nested refinement process is reversible, i.e. it allows a coarsening operation. Transition nodes link triangles of different levels of an AMR, allowing a low number of local control volumes; consequently, a reduced linear equation system that assures the accuracy of the solution is obtained.

The refinement process proposed is depicted in Fig. 1. Suppose that a single graph node represents a triangular volume without partition. The triangular volume is refined and the single graph node is replaced by the subgraph pack shown on the right side of Fig. 1.

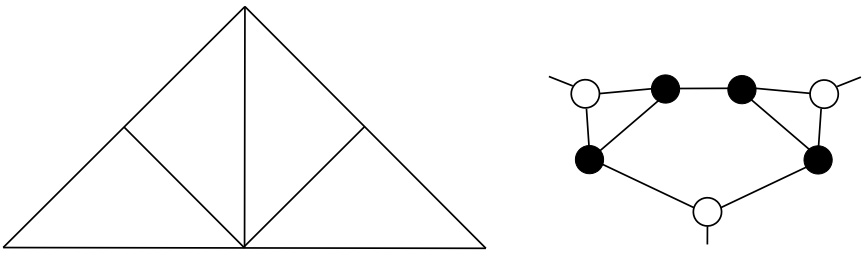

Figure 1: A single graph node that represents an original triangular volume is replaced by a subgraph pack (on the right) created after the refinement of the triangular volume on the left. (The opposite path is performed in the coarsening process.)

In the subgraph on the right side of Fig. 1, two types of nodes are showed: volume nodes (black circles) and transition nodes (white circles). Each volume node represents the respective triangular volume in the mesh. Additionally, graph node pointers are represented by lines and the directed graphs are represented as undirected for clarity, i.e. a line between nodes in the graph means that the nodes point to each other. Furthermore, the volume nodes point to transition nodes by pairs. In addition, each transition node also points to two nodes in a subgraph pack. Finally, a node has 3 pointers, and an unused pointer occurs in the boundary and receives the null value.

An example of an initial discretization and the corresponding graph are shown in Fig. 2. In this figure, the triangle barycenters are represented by black circles, and white circles are used in the graph to link the boundary of the domain. The scheme proposed leads to high-quality meshes provided that high-quality polygons are presented in the initial mesh, as shown in Fig. 2: all the polygons are isosceles right triangles.

During the AMR process, the *four triangle longest-edge* (4T-LE) partition of Rivara (1984) is applied. A mesh with an initial triangular volume refined and the resulting graph are shown in Fig. 3. It should be noted that the boundaries of the domain are omitted in this figure.

Considering a volume node v with its links to nodes n_a, n_b, and n_c as presented on the left side of Fig. 4, the volume node v can be replaced by a pack consisted of the

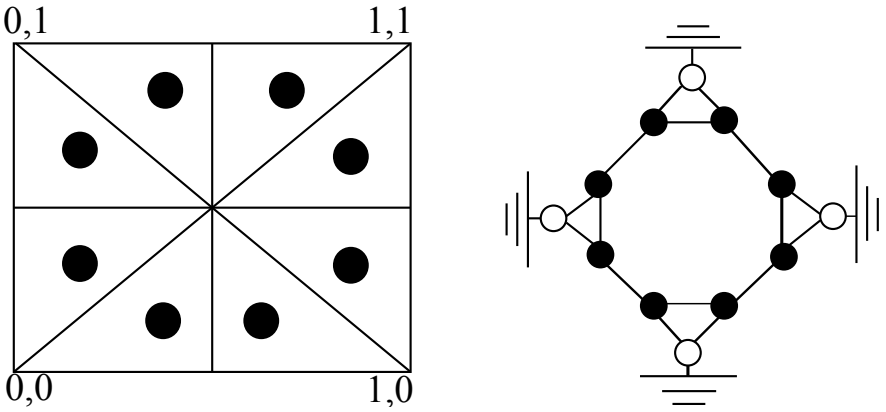

Figure 2: Unit square as an example of the problem domain, and links of the graph data structure; the nodes represent the initial refinement level, i.e. a refinement of level 0 (zero).

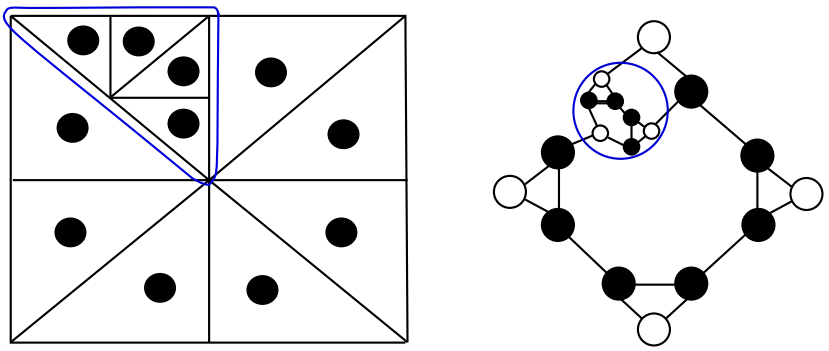

Figure 3: A refinement example with triangular volumes, and its graph representation with transition and volume nodes (the subgraph pack is identified by a blue circle).

volume nodes v_1, v_2, v_3, v_4 and the transition nodes t_1, t_2, and t_3. The links among the nodes of the subgraph pack are set accordingly, including in relation to n_a, n_b and n_c. This is exemplified on the right side of Fig. 4.

Another refinement of the discretized domain visible in Fig. 3 is shown in Fig. 5; besides, its graph representation and its simplification are also illustrated. As can be verified in this figure, a graph is simplified if the involved volumes have common refinement levels, i.e. their vertices in an interface are coincident.

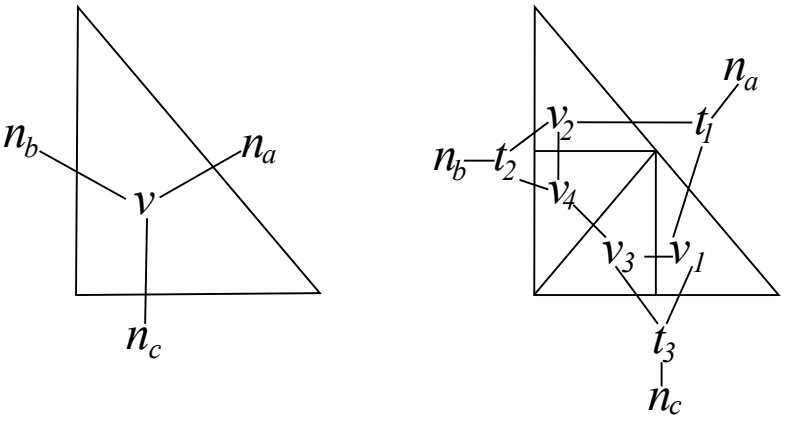

Figure 4: The volume node v with links to nodes n_a, n_b, and n_c on the left side is replaced by the pack consisted of volume nodes v_1, v_2, v_3, v_4 and transition nodes t_1, t_2, and t_3, presented on the right side.

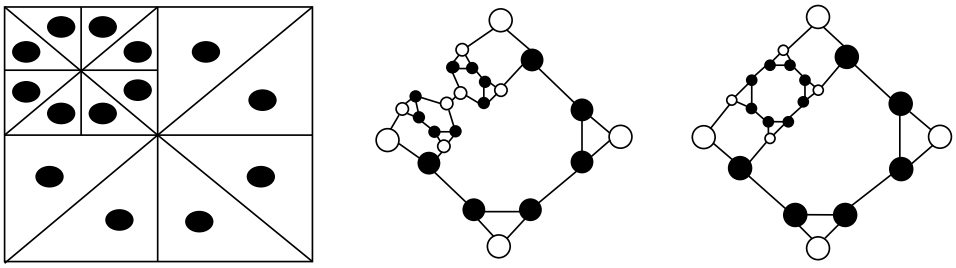

Figure 5: Refinement of another volume depicted in Fig. 3, its graph representation and its graph simplification, from left to right, respectively.

2.2 Coarsening process

All nodes in a common pack have an attribute indicating that the nodes belong to the same subgraph pack. Provided that the neighbor volume nodes are at the same level of the resulting volume, pointers of the resulting volume node and its neighbors directly point to each other. Otherwise, transition nodes are created in order to link neighbors with different levels of refinement. In the coarsening process, the level of the resulting volume node is decreased of 1 (one) in relation to the old pack.

A subgraph pack selected to be coarsened is showed in Fig. 6, which can be replaced by a single volume node as shown in Fig. 7. The final configuration of the discretization followed by the intermediate graph representation is presented in the

graph on the right side of Fig. 7.

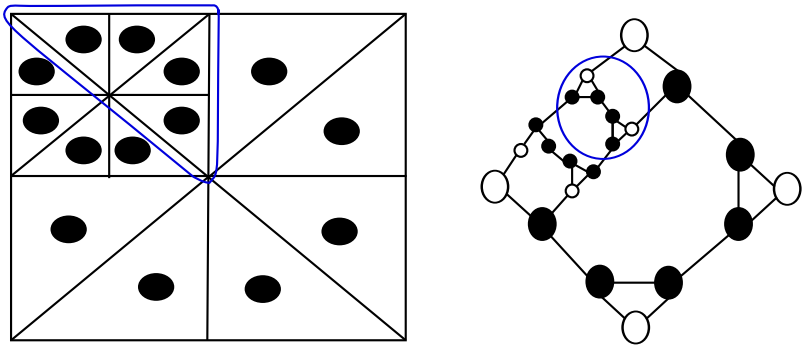

Figure 6: Example of a discretized domain and its graph representation with a pack selected to be coarsened.

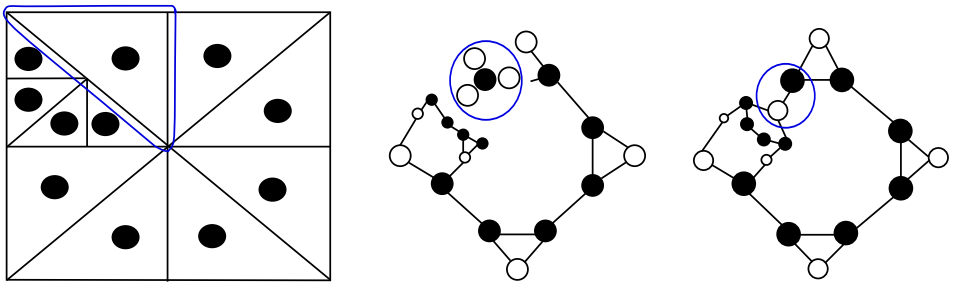

Figure 7: Collapse of the subgraph pack shown in Fig. 6.

3 Mesh total ordering

Mesh volumes should have some ordering so that each volume is related to a line of the resulting linear equation system in an implicit formulation. On the other hand, a triangular sequence defines a total-order relation on the triangles set of a mesh.

Velho, de Figueiredo, and Gomes (1999) explained that a generalized sequential triangulation, or Hamiltonian triangulation, is a triangulation in which there is an ordered set $t_1, \ldots, t_N$ of all its triangles so that two consecutive triangles t_i and t_{i+1} share an edge. A path in a graph is said to be Hamiltonian if all nodes included are visited only once. Each triangle in a Hamiltonian triangulation has an entry edge and an exit edge with respect to the Hamiltonian ordering, and the knowledge

of these edges completely characterizes the sequence. Geometrically, a generalized triangular sequence can be represented by building an oriented path on the mesh domain that visits each triangle traversing its entry and exit edges, called a sequential path. Moreover, the problem of finding the best triangle sequences of a mesh is NP-hard [Evans, Skiena, and Varshney (1996)]. However, by adding new points, it is always possible to refine a triangle mesh into a Hamiltonian triangulation [Arkin, Held, Mitchel, and Skiena (1994)]. As such, the scheme presented here always generates a generalized sequential triangulation since the refinement is strictly local.

In the scheme proposed here, the 4T-LE partition of Rivara (1984) allows the use of the Sierpiński-like Curve, which has a low computational cost [Velho, de Figueiredo, and Gomes (1999); Gonzaga de Oliveira and Kischinhevsky (2008); Bader, Schraufstetter, Vigh, and Behrens (2008); Bader, Bock, Schwaiger, and Vigh (2010)] in order to traverse the graph nodes. The space-filling curve used is a Pólya curve since right isosceles triangles are involved.

The dynamic Sierpiński-like Curve (SlC) for the total-order relation on triangular volumes of a mesh was implemented as a double-linked list. Only *local updates* are carried out during the refinement and coarsening processes. Examples of successive adaptive refinements by volume bisection in a quadrangular domain with triangular volumes ordered by the SlC are shown in Fig. 8. This scheme allows the straightforward updating of the double-linked list for mesh node ordering.

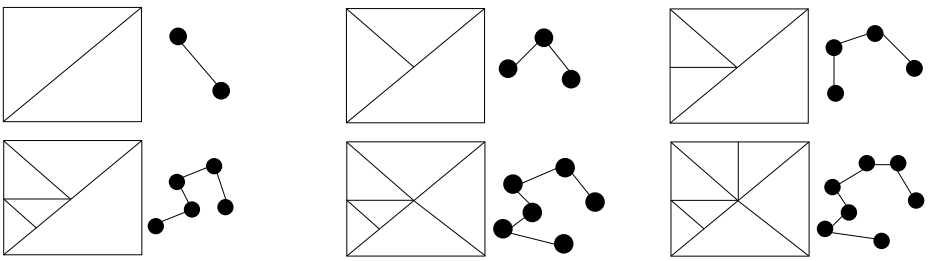

Figure 8: Successive adaptive refinements with the volumes ordered by a SlC shown on the right side of each discretization.

Clearly, each volume node must be taken into account when setting up the linear equation system to be solved. Since triangular volumes are adaptively refined in this approach, the SlC is an efficient way to number the mesh volumes. The Rivara's refinement is applied so that the SlC can be used. The original Rivara's refinement is propagated to adjacent triangles to maintain the mesh conforming. However, this propagation is not considered here since conformity is not required in the FVM.

Since the 4T-LE is applied, a Hamiltonian triangulation is obtained and each update of the double-linked list is strictly local. This follows the strictly local updating of the graph, leading to a subgraph pack, in the refinement or coarsening process. Moreover, when a volume is refined or a subgraph pack is coarsened, only the local pointers of the SlC are updated. This $\Theta(1)$ method only locally updates the SlC already defined.

4 Experimental tests

The following tests concern numerical results for the Laplace equation (elliptic problem) and for the heat conduction equation (parabolic problem). In all tests accomplished, the linear equation systems presented sparse, symmetric and positive-definite coefficient matrices; therefore, a linear equation system solver based on the minimization of functionals was employed; specifically, the Conjugate Gradient Method. A pre-conditioner was not used. In relation to the convective terms of PDEs, an upwind scheme was used. In relation to the diffusive terms, the vertex-centered scheme proposed by Schneider and Maliska (2002) was extended to a cell-centered scheme.

In the tests, the criterion used for the refinement was based on the flux across the interface of neighboring volumes. More precisely, if the gradient of the flux in an edge was higher than a refinement value defined by the user, both volumes that share the edge were refined. The choice of this threshold value was based on the requirements of a large or small refinement at regions of the domain where there were large or small variation in the behavior of the solution, respectively. This criterion overestimates the number of refinements needed, since a large difference of fluxes does not necessarily mean that there is a large variation in the difference between the approximate and the exact solutions. However, it should be noted that variations can also occur with the exact solution.

4.1 Heat conduction equation (parabolic problem)

Let's consider the 2D heat conduction equation with continuous initial and boundary data

$$\begin{aligned} \phi_t &= \nabla^2\phi, \\ \phi(u,0) &= f(u), u \in \Omega, f(u) \equiv 0, \\ \phi(u,t) &= g(u), u \in \partial\Omega, t \geq 0, \end{aligned} \tag{1}$$

in which $\Omega \subset \mathbb{R}^2$ (thus, $u = (x,y)$), and f is a smooth function limited in Ω. This problem is well-defined in the Hadamard sense [Zauderer (1989)]. Taking into

consideration the approximation in a unit square and the FVM basic implicit formulation $M_P^{k+1}\phi_P^{k+1} - \Delta t \oint_{\partial\Omega} \nabla\phi^{k+1} \cdot \vec{n}d(\partial\Omega) = M_P^k\phi_P^k$, in which M represents the area of $\triangle P$, we set the boundary conditions on top, bottom and left sides of the unit square with a unique prescribed boundary value f and the right side with a different value. An initial test [Gonzaga de Oliveira and Kischinhevsky (2009)] allowed to conclude that: after 10 time steps and 7 maximum refinement levels, the mesh consisted of 1496 volumes and the Conjugate Gradient Method converged after 519 iterations.

In another test with the same boundary conditions, the final mesh configuration was achieved after 10 time steps using 1.0 as the refinement criterion and 7 maximum refinement levels for each volume. The mesh obtained, shown in Fig. 9, consists of 5285 volumes and the Conjugate Gradient Method converged after 578 iterations.

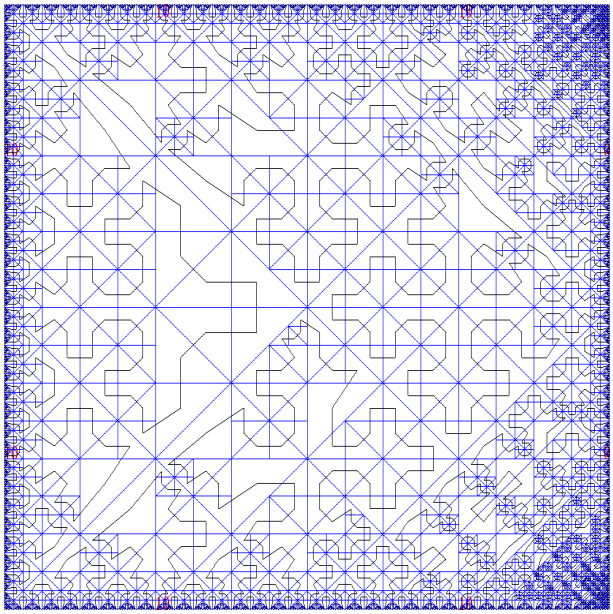

Figure 9: Final mesh configuration of a triangular AMR approximation to the solution of the heat conduction equation.

The result of the test with boundary condition $x - y$ and refinement criterion equal to 0.05 is illustrated in Fig. 10. In this case, the final discretization was achieved after 10 time steps with 6 maximum levels of refinement for each volume resulting in 6110 volumes.

The result of the test with boundary condition $x^2 - y^2$, 0.07 as the refinement criterion and 10 time steps is illustrated in Fig. 11. In this example, the discretization

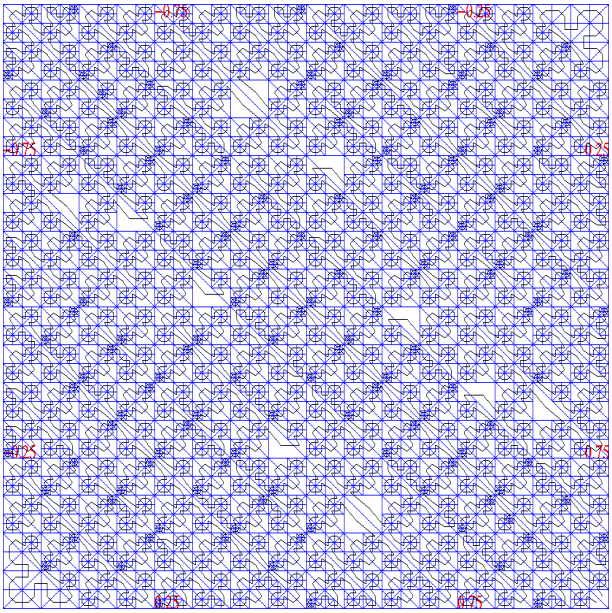

Figure 10: Triangular AMR approximation to the solution of the heat conduction equation with boundary condition $x - y$.

was until 7 refinement levels for each volume resulting in 4472 volumes.

A comparison between the time spent during the triangular AMR stage and the time spent for all numerical computations is presented in Tab. 1 for the experiments conducted for this problem using the boundary conditions suggested in Burgarelli, Kischinhevsky, and Biezuner (2006). In this table, $g(x,y)$ is the boundary condition of the test, l is the maximum refinement level reached, N is the number of volumes attained in the grid, *RefTime* is the normalized time spent in the refinement process, *Time* is the total time spent for all numerical computations, and % indicates $100 \cdot \frac{RefTime}{Time}$.

The tests described were performed with $\Delta t = 0.1$ and the simulations were executed until $t = 10 \cdot \Delta t$. In the same tests, the adopted boundary condition was chosen according to the steady-state solution intended, and the initial conditions were set as $f(x,0) \equiv 0$. Additionally, it should be noted that at level l, the minimum volume size attained was 2^{-l}.

One can conclude from Tab. 1 that the most of the computing time was demanded by the computation core, i.e. to solve the linear equation system, with the refinement stages demanding a considerably low computational time. Clearly, the higher

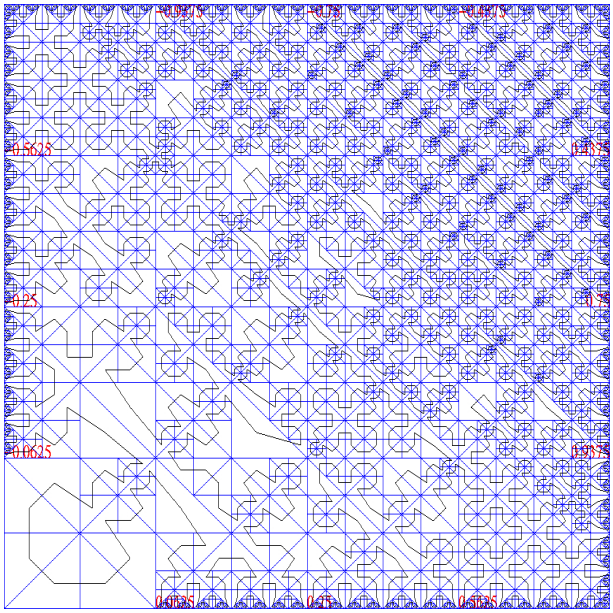

Figure 11: Triangular AMR approximation of the heat conduction equation with boundary condition $x^2 - y^2$.

resolution of a mesh, the higher the total computation time. However, the refinement computational time increases much slower than the total computation time when increasing the number of volumes in the mesh.

4.2 Laplace equation (elliptic problem)

Let's consider now the Dirichlet problem given by

$$\nabla^2 \phi = 0 \; in \; \Omega \in \mathbb{R}^2, \; \phi = f \; on \; \partial\Omega, \tag{2}$$

in which ϕ is the dependent variable of the PDE, Ω is a limited domain in $\mathbb{R}^2$, and f is a defined smooth function on the boundary $\partial\Omega$. This problem is well-posed in the Hadamard sense [Zauderer (1989)]. Considering that one has equal and constant prescribed boundary conditions on top, bottom, and left sides of a unit square, and on the other hand, the right side of the unit square presents a different value for f, the initial test [Gonzaga de Oliveira, Kischinhevsky, Burgarelli, and Biezuner (2008)] allowed to conclude that this triangular AMR for finite-volume discretizations is as simple and straightforward as using the quadrangular ALG. The

Table 1: Numerical results for the heat conduction equation.

$g(x,y)$	l	N	*RefTime (s)*	*Time (s)*	%
	4	860	$1.0 \cdot 10^{-2}$	0.23	4.348
	5	1007	$1.1 \cdot 10^{-2}$	0.29	3.448
10	6	3029	$4.0 \cdot 10^{-2}$	4.86	0.823
	7	3566	$5.0 \cdot 10^{-2}$	8.71	0.574
	8	34541	$2.3 \cdot 10^{-1}$	448.29	0.051
	5	956	$1.0 \cdot 10^{-2}$	0.26	3.846
	6	4028	$2.0 \cdot 10^{-2}$	10.81	0.185
$x-y$	7	16040	$1.3 \cdot 10^{-1}$	93.34	0.139
	8	66260	$6.6 \cdot 10^{-1}$	1036.06	0.064
	9	174410	1.5	7490.62	0.020
	6	3563	$5.0 \cdot 10^{-2}$	15.74	0.318
	7	12602	$1.0 \cdot 10^{-1}$	81.38	0.123
x^2-y^2	8	55811	$4.1 \cdot 10^{-1}$	1013.84	0.040
	9	175499	1.5	7147.79	0.021
	10	283604	3.68	24862.50	0.015

mesh and the SlC obtained by the proposed triangular AMR scheme are illustrated in Fig. 12. In this case, the final mesh configuration was achieved using a maximum of 8 refinement levels for each volume and 1.5 as the refinement criterion. The mesh was comprised of 4427 volumes, whose Conjugate Gradient Method converged after 731 iterations.

ALG and the proposed scheme were applied to the Laplace equation and tested using the same machine. In Tab. 2, the computational times required in each experiment are indicated. The linear equation systems of both schemes were solved using the same implementation of the Conjugate Gradient Method. The number of volumes (N column) in the tests at the same refinement level are similar for both schemes, except in the test with 10 refinement levels. The linear equation systems were solved faster when the triangular scheme was used (see *Time* column), even in the test with 10 refinement levels.

In the test with 8 refinement levels, the refinement time required by each scheme (*RefTime* column) were similar. In the other tests, the refinement time of the triangular scheme was slower than the one of ALG. It is worth to emphasize that in both schemes the refinement time increases much slower than the numerical computation time (see % column).

Besides, one can conclude from the values in Tab. 2 that the refinement process of

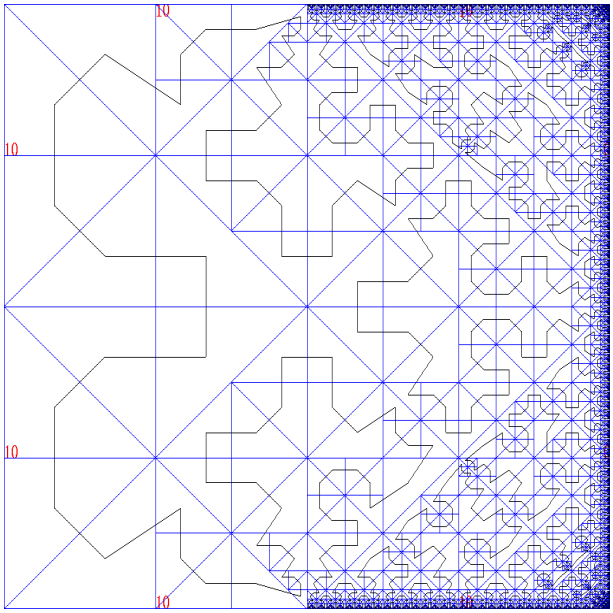

Figure 12: Sierpiński-like Curve of the triangular AMR scheme applied to the Laplace equation.

Table 2: Numerical results for the Laplace equation (△ - triangular scheme).

l	scheme	*N*	*Time (s)*	*RefTime (s)*	*%*
7	ALG	3169	1.12	0.04	3.57
	△	3188	0.89	0.01	1.12
8	ALG	38398	57.39	0.17	0.30
	△	38402	42.81	0.19	0.44
9	ALG	169831	650.01	1.54	0.24
	△	170258	429.17	0.43	0.10
10	ALG	247246	1809.30	3.67	0.20
	△	261260	836.27	0.60	0.07

the proposed scheme scales well for medium-size meshes:

- an increment of $170258/38402 \cong 4.4$ times in the number of volumes resulted in an increment of $0.43/0.19 \cong 2.3$ of the refinement process time;
- an increment of $261260/170258 \cong 1.5$ times in the number of volumes resulted in an increment of $0.60/0.43 \cong 1.4$ of the refinement process time.

5 Complexities of the space-filling curves in ALG and in the approach proposed

In the quadrangular ALG approach, Burgarelli, Kischinhevsky, and Biezuner (2006) extended the Hilbert Curve for the total ordering of the mesh, and the Modified Hilbert Curve (MHC) was used in order to number the volumes. The algorithm that implements the local update of the MHC and its complexity analysis are addressed in the following.

The algorithm for the local update of the MHC requires the definition of the local shape of the Hilbert Curve for each local refinement. Briefly, the local curve can have 4 shapes: $\sqsubset$, $\sqsupset$, $\sqcap$ and $\sqcup$. The *HilbertShape* method is used and according to its output, the MHC, a double-linked list, is updated. *Pseudocode 1* is an example of how to call the *HilbertShape* method. The attributions in *Pseudocode 1* allow that each pointer of the MHC is updated without any loop.

Pseudocode 1: UpdateMHC;
// calls the *HilbertShape* method and locally updates the MHC
input: the Hilbert coordinate *volHilbertCoord* of the volume and its refinement level *volRefLevel*;
output: update the MHC in the pack;
begin
  integer: numberOfHilbertShape;
  numberOfHilbertShape ← *HilbertShape* (*volHilbertCoord*, *volRefLevel*+1);
  if (numberOfHilbertShape = 0) then { 11 attributions + 1 condition } // shape: $\sqsubset$
  else if (numberOfHilbertShape = 1) then { 11 attributions + 1 condition } // shape: $\sqcup$
  else if (numberOfHilbertShape = 2) then { 11 attributions + 1 condition } // shape: $\sqsupset$
  else if (numberOfHilbertShape = 3) then { 11 attributions + 1 condition } // shape: $\sqcap$
  end-if;
end;

The number of operations of the local update of the MHC depends on the *HilbertShape* method described in *Pseudocode 2*. It defines the local shape of the new subgraph pack, and for that the inputs are: the Hilbert coordinate of the volume being refined, i.e. an attribute kept in each volume object that is used only to evaluate the local Hilbert shape, and the refinement level of this volume. The output of *Pseudocode 2* is: 0, 1, 2 or 3, representing the local shapes $\sqsubset$, $\sqcup$, $\sqsupset$ and $\sqcap$, respectively.

Pseudocode 2: HilbertShape; // evaluates the local Hilbert shape
input: the Hilbert coordinate *volHilbertCoord* of the volume being refined;

and the refinement level l of this volume;
output: 0, 1, 2 or 3 that represents the local shapes $\sqsubset$, $\sqcup$, $\sqsupset$ and $\sqcap$, respectively;
begin
1. integer: i $\leftarrow$ 0, j, k;
2. integer: hilbTab[4][4];
3. hilbTab[0][0] $\leftarrow$ 1; hilbTab[0][1] $\leftarrow$ 0; hilbTab[0][2] $\leftarrow$ 0; hilbTab[0][3] $\leftarrow$ 3;
4. hilbTab[1][0] $\leftarrow$ 0; hilbTab[1][1] $\leftarrow$ 1; hilbTab[1][2] $\leftarrow$ 1; hilbTab[1][3] $\leftarrow$ 2;
5. hilbTab[2][0] $\leftarrow$ 3; hilbTab[2][1] $\leftarrow$ 2; hilbTab[2][2] $\leftarrow$ 2; hilbTab[2][3] $\leftarrow$ 1;
6. hilbTab[3][0] $\leftarrow$ 2; hilbTab[3][1] $\leftarrow$ 3; hilbTab[3][2] $\leftarrow$ 3; hilbTab[3][3] $\leftarrow$ 0;
7. for k from 1 to l-1 do
8. j $\leftarrow$ volHilbertCoordinate mod 4;
9. i $\leftarrow$ hilbTab[i][j];
10. volHilbertCoordinate $\leftarrow$ vollHilbertCoordinate / 4;
11. end-for;
12. return i;
end-*HilbertShape* method.

Lines from 1 to 6 and 12 in *Pseudocode 2* are executed once, line 7 is executed l times, and each line from 8 to 11 is executed l-1 times. Thus, $\Theta(l)$ is the asymptotic number of operations of the *HilbertShape* method, in which l is the number of refinement levels of the *pack* built. Clearly, the number of operations of the local update of the MHC is given by calling the *HilbertShape* method in *Pseudocode 2*. It should be noted that *volRefinementLevel* in this algorithm is the number of refinement levels of the volume just refined, and l in the *HilbertShape* method is the number of refinement levels of the *pack* built.

In the approach proposed here, a Sierpiński-like Curve is employed. The complexity of the algorithm that implements the local update of the SlC, $\Theta(1)$, was indicated in Section 3. One can conclude that the Sierpiński-like Curve with its $\Theta(1)$ running time surpasses the MHC implemented within ALG, which computational time is $\Theta(l)$, in which l is the refinement level of the refined pack.

6 Analyses about storage costs in mesh generation

In this section, the storage costs of data structures used in mesh generation are discussed: the graph data structure used in the approach proposed and ALG are analysed in subsections 6.1 and 6.2, respectively. Finally, the storage costs of the data structures for triangle meshes are addressed in subsection 6.3.

6.1 Storage cost of the graph data structure proposed

Let's consider an initial discretization with 8 triangular finite volumes as shown in Fig. 2, that has some triangles in the boundary not refined so that transition nodes at this boundary do not occur, and the sequence $t_0, t_1, \cdots, t_{n-1}$ of triangles in the mesh with refinement level 0 (zero). A possible case occurs when t_i, with $i = 1,3,5,\cdots$, is not refined, but t_j, with $j = 0,2,4,\cdots$, is refined for all triangles at the refinement level 0 (zero). If one continues refining the finite volumes in each level in this way, no transition nodes vanishes because t_i and t_{i+2} are not refined and t_{i+1} and t_{i+3} are for the 4 triangles in a subgraph pack. That is, no adjacent triangles are refined so that no inner transition node vanishes in the data structure. One can conclude by induction that $\sum_{i=1}^{l} 2^{i+1} = 2^{l+2} - 4$ $(\forall l > 0)$, in which l is the level of refinement, and also that there are $2^{l+2} - 4 + 2^{l+3} = 3 \cdot 2^{l+2} - 4$ volume nodes in this scenary. In addition, there are $3 \cdot 2^{l+2} - 8$ transition nodes.

Each volume node presents 5 pointers: 3 neighbors and the SlC double-linked list: $5 \cdot (3 \cdot 2^{l+2} - 4)$. As such, the number of transition nodes is relevant in relation to the number of volume nodes. Defining the class of transition nodes without the two pointers of the double-linked list results in memory savings, as each transition node presents 3 pointers: $3 \cdot (3 \cdot 2^{l+2} - 8)$. Summing up the two terms divided by the number of triangular finite volumes, yields $\frac{24 \cdot 2^{l+2} - 44}{3 \cdot 2^{l+2} - 4} \cong 8$ nodes in the data structure per triangle.

On the other hand, the number of vertices is $3 \cdot 2^{l+2} - 3$. It should be noted that, in this case, two edges are inserted to create one triangle, yielding $e = 2t$, in which e is the number of edges. Clearly, the number of vertices is just about the number of triangles. More precisely, from the Euler's formula, $n = t + 1$, in which n is the number of *vertices* and t is the number of *triangles*. Thus, the number of pointers in the data structure proposed is just about $8n$.

6.2 Storage cost of the ALG structure

Let's consider now 4 quadrangular finite volumes as the initial discretization and refine each finite volume in an arbitrary diagonal in each refinement level. That is, no adjacent volumes are refined so that no inner transition node vanishes in the data structure.

One can conclude by induction that $\sum_{i=1}^{l} 2^i = 2^{l+1} - 2$, and that there are $6 \cdot 2^{l+1} - 4$ nodes in the data structure for $3 \cdot 2^{l+1} - 2$ quadrangular finite volumes. It should be noted that there are just about 2 nodes in the data structure per square in the meshes. Since there are 6 pointers for each node, the number of pointers in the data

structure is $12q$, in which q is the number of *quadrangular finite volumes*.

In this case, there are 5 new vertices for 4 new quadrangular finite volumes in a pack. Thus, the number of vertices is $5 \cdot 2^{l+1} - 1$, yielding $\frac{6 \cdot (6 \cdot 2^{l+1} - 4)}{5 \cdot 2^{l+1} - 1} \cong 7$, and the number of pointers in the data structure is about $7n$, in which n is the number of *vertices*.

As such, one can conclude that the number of transition nodes is relevant in relation to the number of volume nodes. Defining the class of transition nodes without the 2 pointers of the double-linked list results in memory savings. Specifically, the volume nodes would be $6 \cdot (3 \cdot 2^{l+1} - 2)$ and the transition nodes would be $4 \cdot (3 \cdot 2^{l+1} - 2)$ for $3 \cdot 2^{l+1} - 2$ quadrangular finite volumes. Additionally, the number of pointers in the graph data structure is just about $6n$. This scheme is called improved ALG in Tab. 3.

6.3 Comparison among storage costs

In Tab. 3, storage costs of common data structures for mesh generation are indicated as found in De Floriani, Kobbelt, and Puppo (2004) and De Floriani and Hui (2005). It should be noted that the size of an index was assumed as 1 (one).

Table 3: Storage costs of data structures used in mesh generation (adapted from De Floriani, Kobbelt, and Puppo (2004) and De Floriani and Hui (2005)).

Data structure	Indices per vertice ($\cong$)	Graph nodes per polygon
winged-edge [Baumgart (1975)]	27	-
half-edge data structure [Mäntylä (1988)]	27	-
Doubly-Connected Edge List [Mueller and Preparata (1978)]	21	-
data structure of Botsch, Steinberg, Bischoff, and Kobbelt (2002)	21	-
indexed data structure with adjacencies of Lawson (1977)	13	-
IA [Nielson (1997); Paoluzzi, Bernardini, Attani, and Ferruci (1993)]	13	-
directed edge data structure [Campagna, Kobbelt, and Seidel (1998)]	13	-
Triangle-Segment [De Floriani, Magillo, Puppo, and Sobrero (2004)]	13	7
graph data structure proposed	8	8
improved ALG (for quadrangular meshes)	6	12
indexed data structure	6	-

The Triangle-Segment (TS) data structure [De Floriani, Magillo, Puppo, and Sobrero (2004)] extends the indexed data structure with adjacencies (IA) [Nielson (1997); Paoluzzi, Bernardini, Attani, and Ferruci (1993)] to general simplicial complexes [De Floriani and Hui (2005)]. For a manifold domain, the TS also encodes

$6t + n$ patches of information [De Floriani and Hui (2005)]. And, using the Euler's formula, $t = 2n$, both data structures encode $13n$ in the worst case. Related to triangle meshes, the Triangle-Segment (TS) encodes $7t$ in the worst case. In addition, De Floriani and Hui (2005) claimed that the TS data structure presents lower storage cost when compared to other data structures used for the same purpose.

One observes in Tab. 3 that the indexed data structure is the one that presents the lowest storage cost. However, Lawson (1977) extended this data structure in order to support mesh traversal through edges and increased the storage cost to $13n$ (disregarding geometry and attributes), in which n is the number of indexes [De Floriani, Kobbelt, and Puppo (2004)].

The graph data structure proposed here supports efficient mesh traversal. It is worth to note that a volume node v_i can reach its adjacent volume node v_j traversing $|v_i.level - v_j.level|$ transition nodes. As so, it can be concluded that the new data structure is competitive against the data structures reviewed by De Floriani, Kobbelt, and Puppo (2004) in terms of storage cost.

The data structure proposed is also competitive against the ALG in terms of running time (see Tab. 2). In addition, the $8n$ storage cost of the data structure proposed is also competitive with the $6n$ storage cost of both the improved ALG and the indexed data structures.

Both graphs for triangular AMR and ALG for quadrangular-meshes employ an encoding scheme that represents *finite volumes*, i.e. triangles and quadrangular finite volumes, respectively. This graph data structure for triangular AMR presents a storage cost of $8v$ and the ALG storage cost is equal to $12v$, in which v is the number of finite volumes (see subsections 6.1 and 6.2).

7 Conclusions

A triangular discretization for the Finite Volume Method to solve partial differential equations with a graph-based adaptive mesh refinement based on a cell-centered scheme was proposed. The required data for solving the PDE is stored in a graph data structure that represents the triangular AMR.

The data structure proposed was especially designed in order to minimize the number of operations required in the triangular AMR process. The scheme developed enables the replacement of a single triangle by a pack of triangles in any region of interest. Furthermore, as long as a triangle is refined, a graph node is replaced by a subgraph that represents the new triangles inserted. Similarly to ALG, low memory storage is achieved since the refined nodes in the local refinement are discarded.

The refinement process proposed allows that the created new pack from a single triangle may be coarsened, i.e. it may return to a previous stage. The connection

between triangular neighbors is either direct or there are few steps to reach volume nodes among neighbors in different refinement levels. Furthermore, only strict local changes in the data structure occur.

The strategy based on ALG belongs to a group of methods that deals with the AMR technique and allows triangular local mesh refinements with low computational costs. The implementation of the strategy allows flexibility in reaching triangular neighbors with different refinement levels and, since it was not conceived for any particular problem or geometry, it can be applied to study several phenomena and classes of PDEs.

The proposed data structure also seeks to have low computational cost and flexibility in ordering the triangular mesh. For this, the mesh total-ordering is performed using a Sierpiński-like Curve that was implemented by means of a double-linked list. The refinement and coarsening schemes of the triangular volumes enable straightforward updating of both in the linked list for volume meshes total ordering and graph nodes as well.

Acknowledgement: This work was performed with the support of the CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico (National Council for Scientific and Technological Development - Brazil), CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Coordination for Enhancement of Higher Education Personnel - Brazil), and FAPEMIG - Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Minas Gerais Research Support Foundation - Brazil). The authors also acknowledge D. Burgarelli and R. Biezuner for making available the ALG applied to the Laplace equation.

References

Arkin, E.; Held, M.; Mitchel, J.; Skiena, S. (1994): Hamiltonian triangulations for fast rendering. In *The 2nd Annual European Symposium on Algorithms, (LNCS)*, volume 855 of *Lecture Notes in Computer Science*, pp. 36–47. Springer, Berlin Heidelberg New York.

Bader, M.; Bock, C.; Schwaiger, J.; Vigh, C. (2010): Dynamically adaptive simulations with minimal memory requirement - solving the shallow water equations using Sierpinski curves. *SIAM Journal of Scientific Computing*, vol. 2, no. 1, pp. 212–228.

Bader, M.; Schraufstetter, S.; Vigh, C. A.; Behrens, J. (2008): Memory efficient adaptive mesh generation and implementation of multigrid algorithms using

sierpinski curves. *International Journal of Computational Science and Engineering*, vol. 4, no. 1, pp. 12–21.

Baker, C. M. J.; Buchan, A. G.; Pain, C. C.; Tollit, B. S.; Goffin, M. A.; Merton, S. R.; Warner, P. (2013): Goal based mesh adaptivity for fixed source radiation transport calculations. *Annals of Nuclear Energy*, vol. 55, pp. 169–183.

Baumgart, B. G. (1975): A polyhedron representation for computer vision. In *Proceedings AFIPS National Computer Conference*, volume 44, pp. 589–596.

Botsch, M.; Steinberg, S.; Bischoff, S.; Kobbelt, L. (2002): Openmesh - a generic and efficient polygon mesh data structure. In *Proceedings of the OpenSG Symposium*.

Burgarelli, D. D.; Kischinhevsky, M.; Biezuner, R. J. (2006): A new adaptive mesh refinement strategy for numerically solving evolutionary PDE's. *Journal of Computational and Applied Mathematics*, vol. 196, pp. 115–131.

Campagna, S.; Kobbelt, L.; Seidel, H.-P. (1998): Directed edges – A scalable representation for triangle meshes. *Journal of Graphics, GPU, and Game Tools*, vol. 3, no. 4, pp. 1–12.

De Floriani, L.; Hui, A. (2005): Data structures for simplicial complexes: an analysis and a comparison. In Desbrun, M.; Pottmann, H.(Eds): *Eurographics Symposium on Geometry Processing*. The Eurographics Association and Blackwell Publishing.

De Floriani, L.; Kobbelt, L.; Puppo, E. (2004): A survey on data structures for level-of-detail models. In N.Dodgson; Floater, M.; Sabin, M.(Eds): *Advances in Multiresolution for Geometric Modelling, Series in Mathematics and Visualization*, pp. 49–74. Springer Verlag.

De Floriani, L.; Magillo, P.; Puppo, E.; Sobrero, D. (2004): A multi-resolution topological representarion for non-manifold meshes. *Computer-Aided Design Journal*, vol. 36, no. 2, pp. 141–159.

Evans, F.; Skiena, S.; Varshney, A. (1996): Completing sequential triangulations is hard. Technical report, State University of New York, Stony Brook, NY, Department of Computer Science, 1996.

Goffin, M. A.; Baker, C. M. J.; Buchan, A. G.; Pain, C. C.; Eaton, M. D.; Smith, P. N. (2013): Minimising the error in eigenvalue calculations involving the boltzmann transport equation using goal-based adaptivity on unstructured meshes. *Journal of Computational Physics*, vol. 242, pp. 726–752.

Gonzaga de Oliveira, S. L.; Kischinhevsky, M. (2008): Sierpiński curve for total ordering of a graph-based adaptive simplicial-mesh refinement for finite volume discretizations. In *31o. Congresso Nacional de Matemática Aplicada e Computacional (CNMAC)*, pp. 581–585.

Gonzaga de Oliveira, S. L.; Kischinhevsky, M. (2009): A graph-based adaptive triangular-mesh refinement applied to classical elliptic and parabolical problems. In *32o. Congresso Nacional de Matemática Aplicada e Computacional (CNMAC)*, pp. 607–612.

Gonzaga de Oliveira, S. L.; Kischinhevsky, M.; Burgarelli, D.; Biezuner, R. (2008): Graph-based adaptive simplicial-mesh refinement for finite volume discretizations. In *Congresso Ibero-Latino-Americano de Métodos Computacionais em Engenharia (CILAMCE)*.

Lawson, C. L. (1977): Software for C^1 surface interpolation. In Rice, J. R.(Ed): *Mathematical Software III, Academic Press*, pp. 161–164.

Mäntylä, M. (1988): *An Introduction to Solid Modeling*. Computer Science Press.

Mueller, D. E.; Preparata, F. P. (1978): Finding the intersection of two convex polyhedra. *SIAM Theoretical Computer Science*, vol. 7, pp. 217–236.

Nie, Y. F.; Li, Y. Q.; Wang, L. (2012): Parallel node-based local tetrahedral mesh generation. *Computer Modeling in Engineering & Sciences (CMES)*, vol. 83, no. 6, pp. 575–597.

Nielson, G. M. (1997): Tools for triangulation and tetrahedralizations and constructing functions defined over them. In G. M. Nielson, H. Hagen, H. M.(Ed): *In: Scientific Visualization: overviews, Methodologies and Techniques, ch. 20*, pp. 429–525, Silver Spring, MD. IEEE Computer Society.

Paoluzzi, A.; Bernardini, F.; Attani, C.; Ferruci, V. (1993): Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, vol. 12, no. 1, pp. 56–102.

Rivara, M. C. (1984): Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International Journal for Numerical Methods in Engineering*, vol. 20, pp. 745–756.

Schneider, F. A.; Maliska, C. R. (2002): Numerical solution of bidimensional convective-diffusive problems by the Finite Volume Method using unstructed

meshes (in Portuguese). In *IX Congresso Brasileiro de Engenharias e Ciências Térmicas*, volume 0346.

Velho, L.; de Figueiredo, L. H.; Gomes, J. (1999): Hierarchical generalized triangle strips. *The Visual Computer*, vol. 15, pp. 21–35.

Zauderer, E. (1989): *Partial Differential Equations of Applied Mathematics*. Wiley-Interscience, 2nd edition.