Programs and Algorithms of Numerical Mathematics 20
J. Chleboun, P. Kůs, P. Přikryl, M. Rozložník, K. Segeth, J. Šístek (Eds.)
Institute of Mathematics of the Czech Academy of Sciences, Prague 2021

RTIN-BASED STRATEGIES FOR LOCAL MESH REFINEMENT

Alexej Kolcun, Stanislav Sysala

Institute of Geonics of the Czech Academy of Sciences
Studentská 1768, Ostrava, Czech Republic
alexej.kolcun@ugn.cas.cz, stanislav.sysala@ugn.cas.cz

Abstract: Longest-edge bisection algorithms are often used for local mesh refinements within the finite element method in 2D. In this paper, we discuss and describe their conforming variant. A particular attention is devoted to the so-called Right-Triangulated Irregular Network (RTIN) based on isosceles right triangles and its tranformation to more general domains. We suggest to combine RTIN with a balanced quadrant tree (QuadTree) decomposition. This combination does not produce hanging nodes within the mesh refinements and could be extended to tetrahedral meshes in 3D.

Keywords: mesh refinement, longest-edge bisection, right-triangulated irregular network, balanced quadrant tree, homomorphic transformation

MSC: 65D17, 65D18

1. Introduction

Triangulation of an investigated domain is an essential part of the finite element method. It splits the domain into a union of non-overlapping elements with simple polygonal or polyhedral shapes and usually satisfies certain conformity conditions like that two neighboring elements have common vertex, edge or face. We shall consider the triangulation in 2D with the standard triangular finite elements P_k, $k = 1, 2, \dots$

The triangulation (the finite element mesh) could not contain triangles with too sharp or obtuse angles in order for the finite element approximation of a problem to be sufficiently accurate. Simple domains can be easily split into triangles with similar shapes and thus the mesh has a good quality. Mesh generation for complicated domain is more challenging, see, e.g., [1, 6]. One can use the Delaunay mesh generation [1] based on maximization of the minimum angle. A strategy of minimizing the maximum angle can be found in [4].

In practice, the original mesh has to be locally refined to achieve more accurate numerical results. To this end, longest-edge n-section algorithms and their variants are often used, see, e.g., [3, 7]. The longest-edge bisection algorithms are most known.

Our contribution is motivated by solution of slope, tunnel or foundation stability problems where one of the aim is to determine possible failure zones causing collapse

DOI: 10.21136/panm.2020.06

of a structure. This application requires a multiple usage of the local refinement in order to detect the failure zones more precisely, see [2, 8, 9]. We focus on simple domains that can be homomorphically transformed to a regular domain consisting of a union of squares. Therefore, it suffices to work with isosceles right triangles in order to keep the regularity of the refined meshes.

The contribution is organized as follows. In Section 2, a conforming variant of the longest-edge bisection algorithm is introduced. Section 3 is focused on a special case of the mesh, the so-called Right-Triangulated Irregular Network (RTIN) consisting of isosceles right triangles. In Section 4, we demonstrate that the RTIN approach can be also used for more complex geometries if a homomorphic transformation is applied. In Section 5, we suggest to build RTIN on a balanced quadrant tree (QuadTree) decomposition. Section 6 contains some concluding remarks.

2. Longest-edge bisection algorithm

The main idea of this algorithm is based on splitting a triangle according to the longest edge. To save the required property of the mesh we also have to split the neighboring triangle with common edge. It is illustrated in Figure 1 where the triangle ABC is chosen for refinement and its neighbor is denoted by DCB. In Figure 1 a), the common edge BC is the longest one for both the triangles and thus one can directly split these triangles. If BC is not the longest edge of DCB (see the cases b) and c)) then this neighbor triangle is split repeatedly until the common edge becomes the longest one. Figure 1 b) shows one-split decomposition of DCB, Figure 1 c) shows two-split decomposition of DCB. It is clear that after finite number of splittings, the edge BC is the longest one for both the neighbor triangles.

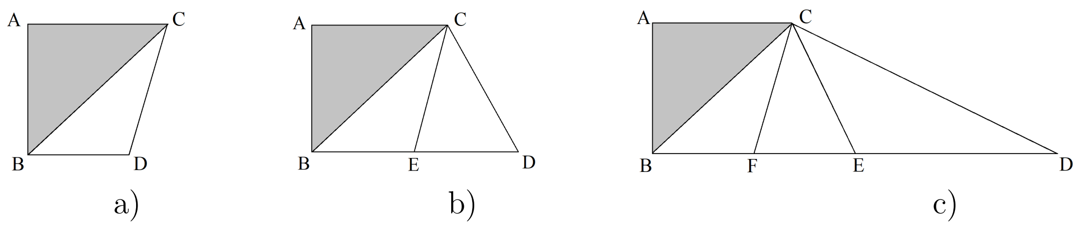

Figure 1: Refinement strategies of triangular mesh.

Let us consider the initial triangular decomposition $T = \{t_1, t_2, t_3, \ldots, t_m\}$ of a *bounded* domain and the array (stack) of distinct triangles $T4R = (t_{i_1}, t_{i_2}, \ldots, t_{i_n})$, $T4R \subset T$, chosen for the refinement. Then the conforming variant of the longest-edge bisection algorithm can be written as follows.

Algorithm 1: (Conforming longest edge bisection algorithm)

```
 1 WHILE (T4R is not empty}
 2 { Select the first element t from T4R, t=ABC,
     where a=BC is the longest edge.
 3   IF (a is on the boundary of the domain)
 4   { Split t to triangles q=DAB, r=DCA.
 5     Replace triangle t with q,r in T.
 6     Remove t from T4R.
 7   }
 8   ELSE
 9   { Select tt=BCD - neighboring triangle to ABC with common edge BC.
10     IF (tt is in T4R) Remove tt from T4R.
11     IF (a is the longest edge in tt)
12     { Split the pair (t,tt) into four triangles EAB,EBD,EDC,ECA.
13       Replace t,tt with these four triangles in T.
14       Remove t from T4R.
15     }
16     ELSE  T4R=(tt, T4R).
17   }
18 }
```

From Algorithm 1, we see that three possible scenarios can occur for any investigated triangle t. If the longest edge a of t is a part of the domain boundary or if a is also the longest edge of the neighboring triangle tt then one can simply split t and eventually tt, see steps 3–15. If a is not the longest edge of tt then we cannot directly split t. In this case, we postpone the splitting of t and analyze its neighbor tt at first such that the array T4R is used as a stack, see step 16.

It is important to note that the while-loop in Algorithm 1 is *finite*. Indeed, we can add only a finite number of triangles to the stack T4R in front of the triangle t because the lengths of their longest edges will increase and the domain is bounded. Once the last triangle (with the maximal longest-edge length) is added to $T4R$ in front of t, the splitting process will start. During this process, other triangles can be added to $T4R$ in front of t only if some triangle requires multi-split decomposition as in Figure 1 c). After finite number of steps, the triangle t will be on the first position in $T4R$ and can be split. It leads to the reduction of $T4R$.

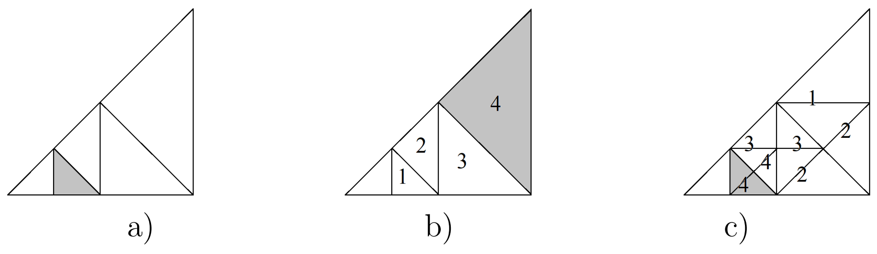

Figure 2: Illustration of Algorithm 1: a) initial stack T4R consists of the dark triangle, b) enlargement of the stack for local refinement, c) final refinement.

The extension and the consequent reduction of the stack $T4R$ within Algorithm 1 is illustrated in Figure 2. The initial triangulation and the stack consisting of the grey triangle is depicted in Figure 2 a). The sequence of triangles inserted into $T4R$ is shown in Figure 2 b). We see that three triangles were added to $T4R$ and their longest-edge lengths increase. The last added triangle has the longest edge on the boundary and thus can be divided according to steps 3–7. The final locally refined mesh is in Figure 2 c). The remaining triangles from $T4R$ are split according to steps 9–15. The edges signed by numbers indicate the order of the refinement process. In this particular case, any triangle from $T4R$ was split into two triangles (1-split decomposition).

Starting from the initial mesh, the longest-edge bisection algorithm can be applied multiply leading to a family of refined and nested meshes. The following theorems summarize important properties of the family, see, e.g., [3].

Theorem 1 (Rosenberg)**.** *Let α_{min} be the smallest angle in the initial triangulation. Then longest-edge bisection algorithm produces the family of triangulations where any angle α of any triangle from any triangulation is such that*

$$\alpha \geq \frac{\alpha_{\min}}{2}.$$

Theorem 2 (Adler)**.** *The longest-edge bisection algorithm produces only a finite number of different triangular shapes.*

From these properties, it follows that the locally refined meshes produced by the longest-edge bisection method do not degenerate even if a large number of refinements is applied.

3. Right-triangle irregular network

If the investigated domain is a union of square subdomains then one can choose the initial triangulation consisting of isosceles right triangles. Then the splitting process is always one-step as in Figure 2, that is, any triangle from $T4R$ is split just into two triangles, unlike the case from Figure 1c). It means that any refined triangle is again isosceles and right-angled and thus the triangles have the same shape and do not degenerate. These facts simplify implementation and properties of Algorithm 1. The corresponding locally refined meshes create the so-called *Right-Triangle Irregular Network* (RTIN).

The construction of RTIN based on a family of nested local refinements is illustrated on the example depicted in Fig. 3. It is considered a quarter of the circle Q inside a square domain and the initial mesh, see Fig. 3 a). Further, we consider the parameters $0 < h_1 < h_2 < 1$, $\varepsilon > 0$ and the following criterion:

$$h_1 \leq \frac{p(t \cap Q)}{p(t)} \leq h_2 \quad \text{and} \quad l(t) > \varepsilon, \tag{1}$$

where $p(t)$ denotes the area of triangle t and $l(t)$ is the length of its hypotenuse.

The purpose of the first part of the criterion is not to refine the triangle if its intersection with prescribed domain is negligibly small. Such small intersections we can see in Figure 1 b) – black. The choice of the values h_1, h_2 depends on compexity of the shape of analyzed domain. Experience leads to values of $h_1 = 0.1$, $h_2 = 0.9$. Nevertheless, the resulting triangulation is sufficiently fine in the given area. This is enforced by the condition of conformity in the decomposition scheme used – see steps 9–17 of Algorithm 1.

The second part of criterion (1) is a simple stopping criterion based on the smallest size of the considered element.

We construct a family of locally refined nested meshes by the following automatically generated process:

Algorithm 2: (RTIN)

```
Based on criterion (1) generate T4R.
WHILE (T4R is not empty)
{ Apply Algorithm 1.
  Based on criterion (1) generate T4R.
}
```

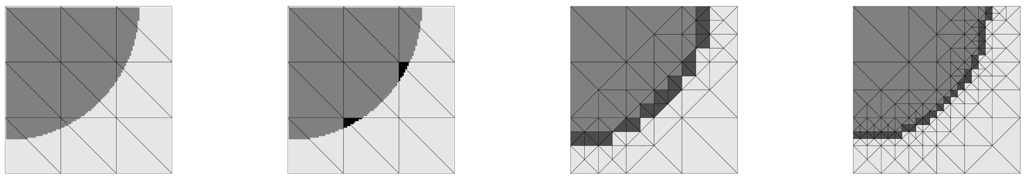
a) b) c) d)

Figure 3: a) The initial triangulation and the subdomain Q (in grey), b) highlighted negligible small intersections of domain with triangles, c) and d) local refinement for different values of ε. Dark triangles form a stack for further refinement.

4. RTIN-based decomposition of more general domains

Let $\Omega \subset \mathbb{R}^2$ be an investigated bounded domain which can be one-to-one transformed to a domain $\widetilde{\Omega} \subset \mathbb{R}^2$ consisting of a union of square subdomains. In particular, we assume that there exists a Lipschitz continuous and piecewise smooth function $\chi\colon \widetilde{\Omega} \to \Omega$ satisfying

$$c_1 \geq \det \nabla\chi(\xi) \geq c_2 > 0 \quad \text{for almost all } \xi \in \widetilde{\Omega}, \tag{2}$$

where c_1 and c_2 are positive constants independent of $\xi \in \widetilde{\Omega}$. Further, let $\widetilde{T}$ be a given RTIN-based triangulation of $\widetilde{\Omega}$. Then, using the function χ, one can transform any nodal point of $\widetilde{T}$ to Ω and create the corresponding triangulation T of the domain Ω.

In addition, we assume that χ is a *homomorphic* transformation in the following sense: there exists a minimal and maximal angles α and β, $0 < \alpha < \beta < \pi$, such that for any RTIN-based triangulation $\widetilde{T}$ of $\widetilde{\Omega}$ the corresponding triangulation T of Ω has the same structure as $\widetilde{T}$ and any angle of any triangle from T has the size between α and β. In general, it is not clear whether the natural assumption (2) is a sufficient condition for satisfying the homomorphic assumption on χ.

Using the transformation χ one can construct a family of locally refined meshes in Ω as in Algorithm 1 applied on RTIN. It is worth noticing that this family cannot be interpreted as the longest-edge bisection, nevertheless its triangles cannot degenerate due to the assumptions on χ.

The transformation χ and RTIN-based meshes are illustrated in Figures 4 and 5, where model examples on stability of a tunnel and a slope are considered, respectively.

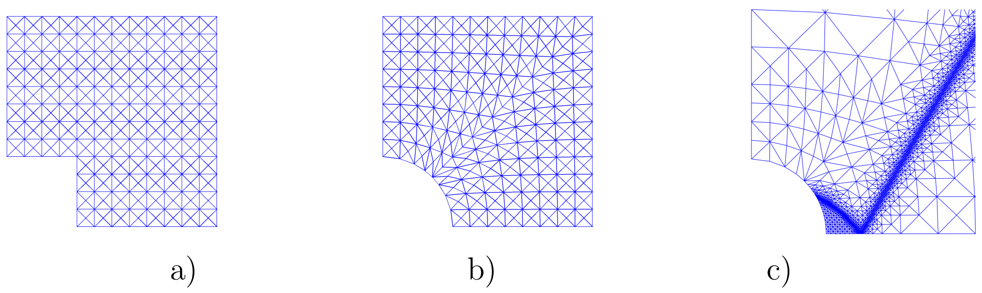

Figure 4: RTIN-based meshes for a tunnel profile (a symmetric quarter of the geometry is considered): a) initial triangulation of $\widetilde{\Omega}$, b) its transformation to Ω, c) adaptively refined RTIN-based mesh (detail).

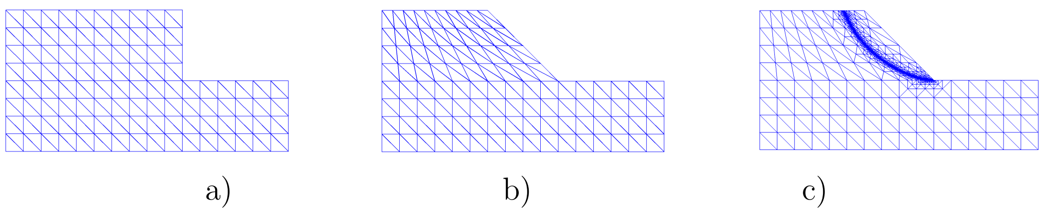

Figure 5: RTIN-based meshes for a slope profile: a) initial triangulation of $\widetilde{\Omega}$, b) its transformation to Ω, c) adaptively refined RTIN-based mesh.

For the slope geometry, the angle characteristic has been done, see Figure 6. Distributions of initial triangulation are in red, distributions of final triangulations are in green. We see that the mesh does not degenerate.

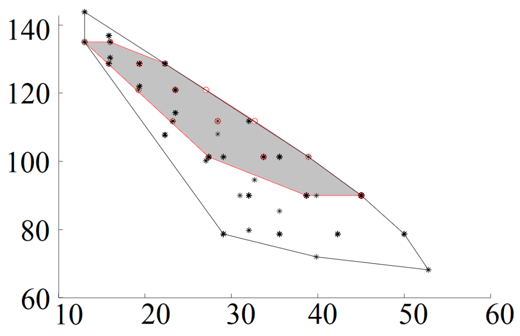

Figure 6: Angle characteristics of the initial (in dark area) and adaptively refined meshes for the slope geometry. x-coordinate means the minimal angle of any triangle, y-coordinate means the maximal one.

5. RTIN strategy based on a balanced quadrant tree

The RTIN approach cannot be directly extended to 3D. At least, we are not able to keep the same shape of all tetrahedrons during the mesh refinement procedure. Motivated by the eventual extension of RTIN to 3D, we suggest the following strategy based on a balanced quadrant tree presented in 2D.

2D extension of recursive dichotomous interval division is called a *quadrant tree* (QuadTree). It leads to a non-conforming decomposition of a rectangular domain into rectangles. For any rectangular element e of the decomposition, one can determine its recursion depth $\sigma := \sigma(e)$. The degree of nonconformity denoted by η is the maximal difference $\sigma(e_1) - \sigma(e_2)$ of the recursion depths between two adjacent elements e_1, e_2. The determination of η is illustrated in Figure 7 a), b). We see that the recursion depths σ of the squares vary from 1 to 4. We have $\sigma = 1$ for the largest squares, while $\sigma = 4$ for the smallest ones. Therefore, $\eta = 3$ in Fig 7 a), while $\eta = 1$ in Figure 7 b). If $\eta \leq 1$ then we say that QuadTree is *balanced.* The balanced QuadTree in 2D and also in 3D has been used, for example, in [5].

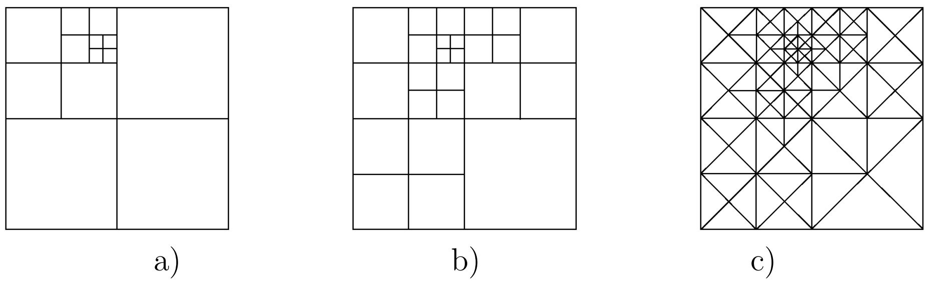

Figure 7: a) QuadTree with $\eta = 3$. b) QuadTree with $\eta = 1$ (balanced QuadTree). c) RTIN based on the balanced QuadTree (QRTIN).

RTIN strategy based on balanced QuadTree (or, briefly, QRTIN) has two steps:

1. refine locally QuadTree and keep $\eta = 1$, see Figure 7 b);
2. create QRTIN from the refined QuadTree, see Figure 7 c).

The local refinement procedure of the balanced QuadTree is similar to Algorithm 1. We prescribe the set E of all rectangular elements defining the initial QuadTree and the stack $E4R$ of elements to be refined. We sketch the corresponding algorithm.

Algorithm 3: (Balancing)

```
1 WHILE (E4R is not empty)
2 { Select first element e from E4R.
3   refin=TRUE.
4   Find the set S of all elements adjacent with e.
5   FOR (s in S)
6   { IF (sigma(e) > sigma(s)) { E4R = (s, E4R). refin=FALSE. }
7   }
8   IF (refin) { Refine e and any s in S. Remove e from E4R.}
9 }
```

The balance of refined QuadTree is kept due to step 6 of Algorithm 3: if an adjacent element s has the recursion depth $\sigma(s)$ less than $\sigma(e)$ then we postpone the split of e and add s at the beginning of the stack $E4R$.

The conforming QRTIN can be created from the balanced QuadTree, for example, by the following algorithm.

Algorithm 4: (Converting quadrilaterals to triangles)

```
1 For (e in E)
1 { IF (e contains a hanging node)
2   { Split e into four triangles using both diagonals.
3     Split all triangles containing the hanging nodes. }
4   ELSE
5   { Split e into two triangles by one of the diagonals.}
6 }
```

Resulting algorithm we obtain as a modification of the Algorithm 2, where instead of step 3 (Algorithm 1) the Algorithm 3 and Algorithm 4 are used.

To illustrate QRTIN we use the same example as in Section 3. In Figure 8, one can see: a) A locally refined balanced QuadTree structure, b) the corresponding QRTIN, and c) RTIN refinement (for comparision). In Table 5, the number of triangles for RTIN and QRTIN strategies and their ratios n_Q/n_H depending on selected values of ε are compared.

It is readily seen that the QRTIN strategy can be also transformed to more general domains as in Section 4.

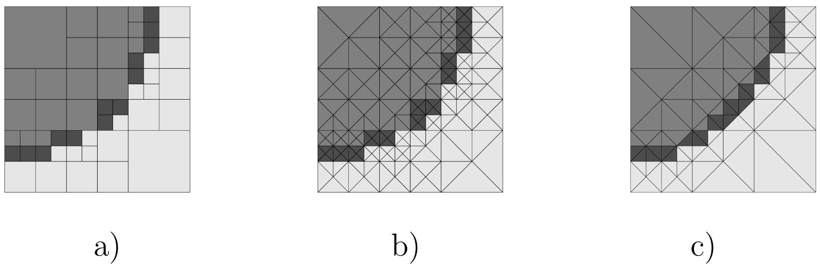

Figure 8: a) QuadTree, b) the corresponding QRTIN, c) RTIN refinement (for comparison).

ε	n(RTIN)	n(QRTIN)	n_Q/n_H
8	271	572	2.11
4	616	1208	1.96
2	1282	2824	2.20
1	2666	5792	2.17
0.5	5664	12022	2.12

Table 1: Comparison of numbers of triangles for RTIN and QRTIN strategies.

6. Conclusion

In the paper, we have started from the longest-edge bisection algorithm in 2D and applied it on Right-Triangulated Irregular Network (RTIN) consisting of isosceles right triangles. RTIN-based refinements has been extended to more general domains by the homomorphic transformation. We have also suggested a combination of RTIN and the balanced quadrant tree (QRTIN) for purposes of extension to 3D in future. This research has been motivated by solution of geotechnical stability problems.

Acknowledgements

This work was supported by grant No. 19-11441S of the Czech Science Foundation.

References

[1] Cheng, S-W., Dey, T. K. and Shewchuk, J. R.: *Delaunay mesh generation.* CRC Press 2013.

[2] Haslinger, J., Repin, S. and Sysala, S.: Inf-sup conditions on convex cones and applications to limit load analysis. Math. Mech. Solids **24** (2019), 3331–3353.

[3] Korotov, S., Plaza, A. and Suárez, J. P.: Longest-edge n-section algorithms: Properties and open problems. J. Comput. Appl. Math. **293** (2016), 139–146.

[4] Křížek, M.: On the maximum angle condition for linear tetrahedral elements. SIAM J. Numer. Anal. **29 (2)** (1992), 513–520.

[5] Kůs, P. and Šístek, J.: Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver. Adv. Eng. Softw. **110** (2017), 34–54.

[6] Lo, D. S. H.: *Finite element mesh generation.* CRC Press 2015.

[7] Rivara, M. C. and Irribaren, G.: The 4-triangles longest-side partition of triangles and linear refinement algorithms. Math. Comp. **65 (216)** (1996), 1485–1502.

[8] Sysala, S., Blaheta, R., Kolcun, A., Ščučka, J., Souček, K. and Pan, P.: Computation of composite strength by limit analysis. Key Engineering Materials **810** (2019), 137–142.

[9] Sysala, S., Haslinger, J. and Repin, S.: Reliable computation and local mesh adaptivity in limit analysis. In: J. Chleboun, P. Kůs, P. Přikryl, M. Rozložník, K. Segeth, J. Šístek, and T. Vejchodský (Eds.), PANM 19: Proceedings of 19th conference, Programs and Algorithms of Numerical Mathematics, Prague: Czech Academy of Sciences, 2019, 149–158.