NON-STRUCTURED GRID REFINEMENT USING GENETIC ALGORITHMS

Elton F. Doehnert[1], Luciano Araki[2]

[1]*Dept. of Mechanical Engineering, University of Paraná*
eltonfd@gmail.com, lucianoaraki@gmail.com

Abstract. Refinement of non-structured grids is an important topic related to the accuracy of numerical solutions of the most used methods to solve engineering problems, such as the finite elements method and the finite volumes method. Considering this fact, the current paper presents a methodology to optimize triangular non-structured meshes using a floating point genetic algorithm to choose the best position for grid vertices. Such position is based on a fitness function evaluated to each internal grid vertex; it is based on the average of the quality of volumes to which the vertex belongs to. The algorithm is executed to a grid firstly generated with the Delaunay's triangularization and then disturbed to obtain a non-optimal grid, which needs to be improved. Results show that the quality parameters were improved, although the method needs high computational efforts.

Keywords: Genetic algorithm; evolutional method; non-structured grids; mesh refinement

1 Introduction

Although automatic meshing tools are widely used, these tools may not guarantee the quality of meshes. Not only in the tessellation process, but also in mesh refinement, it is possible that some severely distorted or out of shape elements are created. Even when a uniform mesh is desired, the tessellation tool can generate elements that are too small or too large compared to the desired elements. [9]

There are several types of mesh smoothing schemes, such as Laplacian smoothing and optimization-based smoothing. Typically, each method has a compromise between computational quality and cost. For example, Laplacian smoothing requires very low computational cost, but often results in poor quality mesh on elements or even invalid elements. On the other hand, while optimization-based smoothing is more likely to avoid invalid elements and achieve higher mesh quality, the computational cost is much higher than Laplacian smoothing. [9]

Genetic Algorithms (GAs) are adaptive methods that can be used to solve problems involving search and optimization. They are based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and survival of the fittest, first enunciated by Charles Darwin in the book *The Origin of Species*. By imitating this process, the genetic algorithm is able to evolve solutions to real-world problems, as long as they have been correctly coded. For example, GAs can be used to design bridge structures, for the greatest strength/weight ratio, or to determine the least amount of scrap when cutting tissue. It can also be used for online process control, such as in a chemical plant, or balancing multi-processor computer systems. [1]

GA uses a direct analogy with nature, where individuals from a population, represented in the algorithm by possible solutions to a problem, receive a level of adaptability according to their performance in solving the problem, such level is called *fitness* . Individuals with a high *fitness* value are more likely to reproduce, that is, to pass on their characteristics to the new population. On the other hand, members of the good low *fitness* population will have less chance of reproduction, so that over time their genetic characteristics will become extinct.

In the genetic algorithm, a potential solution to the problem can be represented by a set of parameters. These parameters (known as *genes*) are joined together to form a string of values (often called *chromosomes*). In the article [4] it is shown that the ideal is to use a binary alphabet for this value chain. For example, if our problem is to maximize a function of three variables, $F(x, y, z)$, we can represent each variable by a 10-bit binary number. Our chromosomes, therefore, would contain three genes, and would have 30 binary numbers. However, according

to [5] these variables can be represented using a floating point encoding, and with this representation the algorithm becomes faster, more consistent and with greater precision especially in large domains, where the encoding binary would become too long.

In genetic terms, the set of parameters represented by a given chromosome is called the *genotype*. The genotype contains information needed to build an organism - which is called a *phenotype*.

The objective of this paper is to describe how to use the Genetic Algorithm as described above to rearrange the position of the internal vertices of a non-structured grid in a way to make the numerical solution more accurate.

2 Method

The method used consists of generating a mesh using conformal Delaunay triangulation, a method that generates high quality triangulations. This method is described in [8] and [7]. This mesh is then optimized using the genetic algorithm in each of its internal vertices, in order to establish the best position with respect to a quality criterion, this quality is defined in [3] and is calculated for each mesh volume according to the equation 1:

$$SQ_i = \frac{4\sqrt{3}A_i}{\sum_{i=1}^{3} l_i^2} \tag{1}$$

In the equation 1 the parameters are:

- A_i: For a given control volume i this is the area of that given volume;
- l_i: This is the length of a tringulation side

In the problem of determining the ideal position of the mesh vertices, a set of Cartesian coordinates can be defined to determine this position, such set can be called by the *genotype* of the solution while the *phenotype* is the the average quality of the volumes that share this vertex. The adaptation of each individual depends on the performance of their phenotype. This can be inferred from your *genotype* - i.e. it can be computed from your chromosome using the fitness function.

2.1 Algorithm

Each generation will be defined by a new population, which will have, on average, a better average *fitness* and will therefore converge towards a solution to the problem. A summary of the methodology using genetic algorithms is given by [6] and is given by:

- **Startup:** Initially, many solutions representing individuals are randomly generated in order to form the initial population. The size of this population depends on the nature of the problem, but it typically contains many hundreds of thousands of possible solutions. Traditionally, the population is randomly generated, covering the entire possible space of solutions. Occasionally, one may prioritize a particular area of that space perceived as having a higher probability of having the ideal solution.
- **Selection:** For each new generation, a set of the current population is selected in some way so as to generate the new generation. Such individuals are selected by their *fitness* function. Some methods calculate the fitness of the entire generation while others select just a few random individuals, however the latter method can be much more costly in terms of time. Usually the chosen fitness function is stochastic and made in such a way that even solutions with low fitness have a small chance of being selected, such a strategy helps to keep the genetic diversity of the population high, preventing the premature emergence of bad solutions. The most studied and used selection forms include selection *roulette whell* and *tournament selection*.
- **Reproduction:** The next step is the generation of new individuals who will be part of the new population, which will replace the old population. Such reproduction is done using the *genotype* of the selected parents and recombining them through the process of *crossover* and *mutation*. As the child generated in this process has genotypes from both parents, he will share many characteristics with his parents. New parents are selected for each new child and this process continues until the entire new population is generated. Although typically only two parents are chosen for each child, some researchers [2] suggest that more than two parents can produce a child with a better genotype.

The evolution process of new solutions (individuals) continues until some stopping criterion is established, which can be:

- A solution is found satisfying a minimum criterion;
- A fixed number of generations is reached;
- A computational time is reached;
- The solution with the greatest fitness of the population reaches a plateau such that new generations cannot improve the result;

- A combination of the above.

3 Results

The optimization algorithm based on the presented method is implemented in Python programming language for two complex geometries and the results are presented in the figures 1 and 2. For each mesh is calculated some statistical values of its interval triangulation angles and its volume qualities as defined in equation 1, before the algorithm those values are presented in the tables 1 and 3, after the GA algorithm the new values of the meshes statistics are found in the tables 2 and 4. The statistics calculated in those tables are of its highest angle, smallest angle, standard deviation between angles, highest quality between volumes, lowest quality, standard deviation of quality and the mean of quality, which can be used as a parameter of the overall quality of a mesh.

Table 1. Original Mesh 1

Valor	Ângulo	Qualidade
Média	60	0.830759
Desvio Padrão	22.868034	0.193268
Mínimo	9.554770	0.076371
Máximo	157.010795	0.999923

Table 2. Mesh 1 with GA

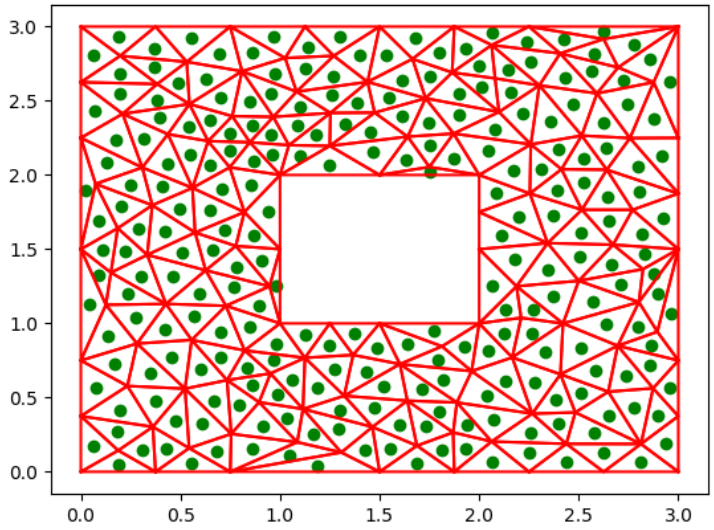
Valor	Ângulo	Qualidade
Média	60	0.848804
Desvio Padrão	21.527223	0.189446
Mínimo	5.504031	0.041763
Máximo	161.853212	0.999185

Table 3. Original Mesh 2

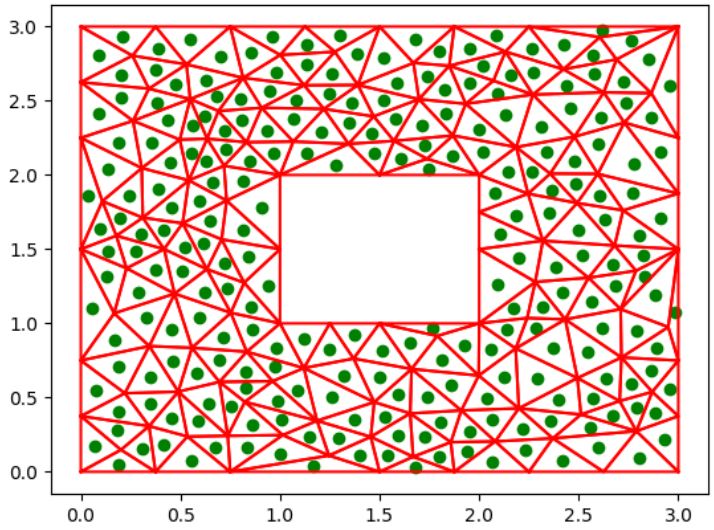
Valor	Ângulo	Qualidade
Média	60	0.873503
Desvio Padrão	17.606637	0.102456
Mínimo	26.888982	0.604936
Máximo	109.030637	0.999390

Table 4. Mesh 2 with GA

Valor	Ângulo	Qualidade
Média	60	0.900385
Desvio Padrão	15.643499	0.098648
Mínimo	24.665694	0.621293
Máximo	106.827269	0.998796

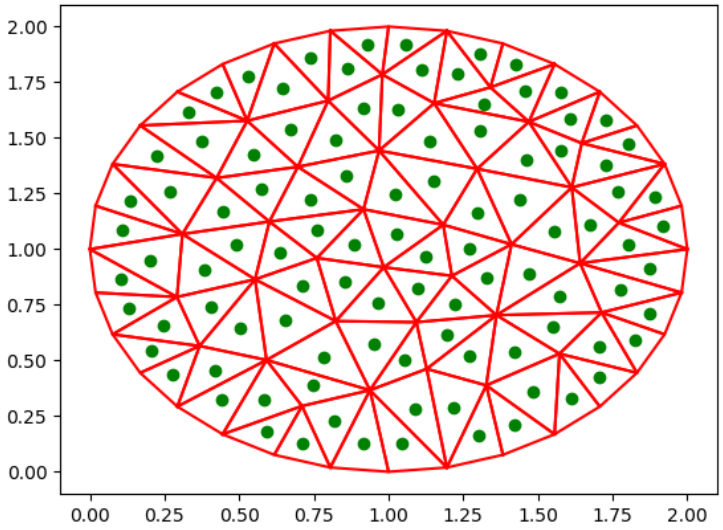


(a) Original Mesh

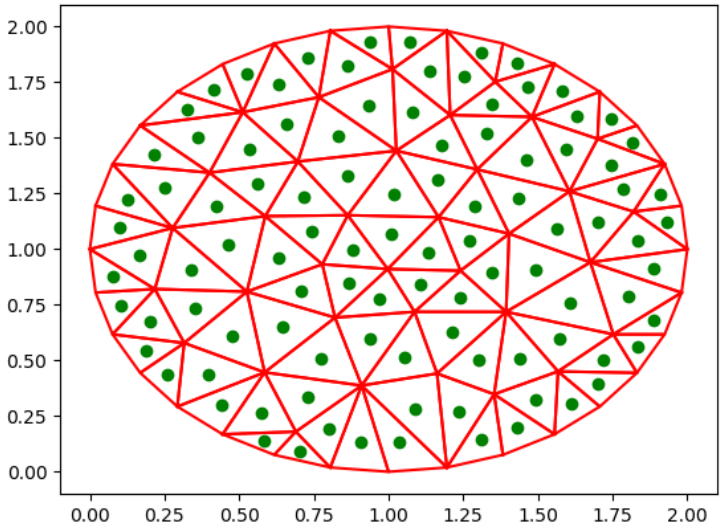


(b) Mesh after GA

Figure 1. Malha 1



(a) Original Mesh



(b) Mesh after GA

Figure 2. Mesh 2

Conclusion

The optimization of unstructured meshes is of great importance in certain types of applications, this optimization can be done with the use of some type of heuristic.

In this work, two unstructured triangular meshes were generated using Delaunay triangulation, a method based on the genetic algorithm was developed to optimize the position of the mesh's internal vertices based on a quality factor. The results show that even for a mesh generated by Delaunay triangulation, which is a very efficient method of generating triangular meshes, the algorithm was able to improve the average volume quality of this mesh, as can be seen in the 2 and 4 where quality is measured by a quality factor defined in 1.

Although the algorithm manages to improve the quality of the meshes, it is known that other smoothing methods are more efficient in relation to computational cost as well as generating a better result.

References

[1] David Beasley. An Overview of Genetic Algorithms : Part 1 , Fundamentals. 1993.

[2] A. E. Eiben, P. E. Raué, and Zs. Ruttkay. Genetic algorithms with multi-parent recombination. pages 78–87. 2012.

[3] Mehdi Falsafioon, Sina Arabi, Ricardo Camarero, and Francois Guibault. Comparison of Two Mesh Smoothing Techniques for Unstructured Grids. *IOP Conference Series: Earth and Environmental Science*, 22, 2013.

[4] John H. Holland. Adaptation in natural and artificial systems, 1992.

[5] Cezary Z. Janikow and Zbigniew Michalewicz. An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms. *undefined*, 1991.

[6] Manoj Kumar, Mohammad Husian, Naveen Upreti, and Deepti Gupta. GENETIC ALGORITHM: REVIEW AND APPLICATION. *International Journal of Information Techonology and Knowledge Management*, 2(2):451–454, 2010.

[7] Ming C. Lin, Dinesh N. Manocha, and Pa.) ACM Workshop on Applied Computational Geometry (1st : 1996 : Philadelphia. *Applied computational geometry : towards geometric engineering : FCRC '96 Workshop, WACG '96, Philadelphia, PA, May 27-28, 1996 : selected papers*. Springer, 1996.

[8] Jonathan Richard Shewchuk. Unstructured Mesh Generation. pages 259–299, 1992.

[9] Tian Zhou and Kenji Shimada. An angle-based approach to two-dimensional mesh smoothing. *Proceedings of the 9th International Meshing Roundtable*, (412):373–384, 2000.