Finite Element Method - 2D Mesh Generator - Metfem2D

Finite Element Method. Abstract.

The presented article contains a 2D mesh generation routine optimized with the Metropolis algorithm. The procedure enables to produce meshes with a prescribed size *h* of elements. These finite element meshes can serve as standard discrete patterns for the Finite Element Method (FEM). Appropriate meshes together with the FEM approach constitute an effective tool to deal with differential problems. Thus, having them both one can solve the 2D Poisson problem. It can be done for different domains being either of a regular (circle, square) or of a non – regular type. The proposed routine is even capable to deal with non – convex shapes.

Finite Element Method. Introduction.

The variety of problems in physics or engineering is formulated by appropriate differential equations with some boundary conditions imposed on the desired unknown function or the set of functions. There exists a large literature which demonstrates numerical accuracy of the finite element method to deal with such issues [1]. Historical development and present – day concepts of finite element analysis are widely described in references [1]. In Sec. 2 of the paper and in its Appendixes A – D, the mathematical concept of the Finite Element Method is presented. In presented article the well – known Laplace and Poisson equations will be examined by means of the finite element method applied to *an appropriate mesh*. The class of physical situations in which we meet these equations is really broad. Let us recall such problems like heat

conduction, seepage through porous media, irrotational flow of ideal fluids, distribution of electrical or magnetic potential, torsion of prismatic shafts, lubrication of pad bearings and others [2]. Therefore, in physics and engineering arises a need of some computational methods that allow to solve accurately such a large variety of physical situations.The considered method completes the above-mentioned task. Particularly, it refers to a standard discrete pattern allowing to find an approximate solution to continuum problem. At the beginning, the continuum domain is discretized by dividing it into a finite number of elements which properties must be determined from an analysis of the physical problem (e. g. as a result of experiments). These studies on particular problem allow to construct so – called *the stiffness matrix* for each element that, for instance, in elasticity comprising material properties like stress-strain relationships [3]. Then the corresponding *nodal loads* [4] associated with elements must be found.The construction of accurate elements constitutes the subject of a mesh generation recipe proposed by the author within the presented article. In many realistic situations, mesh generation is a time – consuming and error – prone process because of various levels of geometrical complexity. Over the years, there were developed both semi – automatic and fully automatic mesh generators obtained, respectively, by using the mapping methods or, on the contrary, algorithms based on the Delaunay triangulation method [5], the advancing front method [6] and tree methods [7]. It is worth mentioning that the first attempt to create fully automatic mesh generator capable to produce valid finite element meshes over arbitrary domains has been made by Zienkiewicz and Phillips [8].The advancing front method (AFM) starts from an *initial* node distribution formed on a basis of the domain boundary, and proceeds through a sequential creation of elements within the domain until its whole region is completely covered by them. The presented mesh algorithm takes advantage from the AFM method as it is demonstrated in Sec. 3. After a node generation along the domain boundary (Sec. 3.1), in next steps interior of the domain is discretized by adding *internal* nodes that are generated at the same time together with corresponding elements which is similar to Peraire *et al.* methodology [9], however, positions of these new nodes are chosen differently according to the manner described in Sec. 3.2. Further steps improve the quality of the mesh by applying the Delaunay criterion to triangular elements (Appendix E) and by a node shifting based on the Metropolis rule (Sec. 4).

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ O. C. Zienkiewicz and Y. K. Cheung, *Finite elements in the solution of field problems*, The Engineer, pp. 507-510 1965; O. C. Zienkiewicz, P. Mayer and Y. K. Cheung, *Solution of anisotropic seepage problems by finite elements*, J. Eng. Mech., ASCE, **92**, pp. 111-120, 1966; O. C. Zienkiewicz, P. L. Arlett, and A. K. Bahrani, *Solution of three – dimensional field problems by the finite element method*, The Engineer, 1967; L. R. Herrmann, *Elastic torsion analysis of irregular shapes*, J. Eng. Mech., ASCE, **91**, pp. 11-19, 1965; A. M. Winslow, *Numerical solution of the quasi-linear Poisson equation in a non-uniform triangle 'mesh'*, J. Comp. Phys., **1**, pp. 149-172, 1966; M. M. Reddi, *Finite element solution of the incompressible lubrication problem*, Trans. Am. Soc. Mech. Eng., **91**:524 1969

3.^ R. W. Clough, *The finite element method in structural mechanics*, In O. C. Zienkiewicz and G. S. Holister, editors, Stress Analysis, Chapter 7. John Wiley & Sons, Chichester, 1965

4.^ Nodes are mainly situated on the boundaries of elements, however, can also be present in their interior.

5.^ A. Bowyer, *Computing Dirichlet tessellations*, Comp. J., **24**(2), pp. 162-166, 1981; D. F. Watson, *Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes*, Comput. J., **24**(2), pp. 167-172 1981; J. C. Cavendish, D. A. Field and W. H. Frey, *An approach to automatic three-dimensional finite element mesh generation*, Int. J. Numer. Meth. Eng., **21**, pp. 329-347 1985;N. P. Weatherill, *A method for generating irregular computation grids in multiply connected planar domains*, Int. J. Numer. Meth. Eng., **8**, pp. 181-197 1988; W. J. Schroeder, M. S. Shephard, *Geometry-based fully automatic mesh generation and the Delaunay triangulation*, Int. J. Numer. Meth. Eng., **26**, pp. 2503-2515 2005; T. J. Baker, *Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation*, Eng. Comp., **5**, pp. 161-175 1989; P. L. Georgea, F. Hechta and E. Saltela, *Automatic mesh generator with specific boundary*, Comp. Meth. Appl. Mech. Eng., **92**, pp. 269-288 1991

6.^ S. H. Lo, *A new mesh generation scheme for arbitrary planar domains*, Int. J. Numer. Meth. Eng., **21**, pp. 1403-1426 1985; J. Peraire, J. Peiro, L. Formaggia, K. Morgan, O. C. Zienkiewicz, *Finite element Euler computations in three dimensions*, **26**, pp. 2135-2159 2005; R. Löhner, P. Parikh, *Three – dimensional grid generation by the advancing front method*, Int. J. Num. Meth. Fluids **8**, pp. 1135-1149 1988

7.^ M. A. Yerry, M. S. Shephard, *Automatic three-dimensional mesh generation by the modified-octree technique*, Int. J. Numer. Meth. Eng., **20**, pp. 1965-1990 1984; P. L. Baehmann, S. L. Wittchen, M. S. Shephard, K. R. Grice, and M. A. Yerry, *Robust, geometrically-based, automatic two-dimensional mesh generation*, Int. J. Numer. Meth. Eng., **24**, pp. 1043-1078 1987; M. S. Shephard and M. K. Georges, *Automatic three-dimensional mesh generation by the finite octree technique*, Int. J. Numer. Meth. Eng., **32**, pp. 709-749 1991

8.^ O. C. Zienkiewicz and D. V. Phillips, *An automatic mesh generation scheme for plane and curved surfaces by isoparametric coordinates*, Int. J. Numer. Meth. Eng., **3**, pp. 519-528 1971

9.^ J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, *Adaptative remeshing for compressible flow computations*, J. Comp. Phys. **72**, pp. 449-466, 1987

Finite Element Method. Mathematical Basis.

The finite elements method (FEM) is based on the idea of division the whole domain Ω into a number of finite sized elements or subdomains Ω^i in order to approximate a continuum problem by a behavior of an equivalent assembly of discrete finite elements [1].

In the presence of a set of elements Ω^i the total integral over the domain Ω is represented by the sum of integrals over individual subdomains Ω^i

$$\int_{\Omega} L\,(u, \partial u/\partial x, ...)d\Omega = \sum_i \int_{\Omega^i} L(u, \partial u/\partial x, ...)\, d\Omega^i$$

$$\int_{\Gamma} L(u, \partial u/\partial x, ...)\, d\Gamma = \sum_i \int_{\Gamma^i} L(u, \partial u/\partial x, ...)d\Gamma^i,$$

where L(u, ∂u/∂x, ...) denotes a differential operator.
The continuum problem is posed by appropriate differential equations (e. g. Laplace or Poisson equation) and boundary conditions that are imposed on the unknown solution ϕ. The general procedure of FEM is aimed at finding an approximate solution ϕ_A given by the expansion:

(*)

$$\phi \sim \phi_A = \sum_j \phi_A{}^j N_j = \phi_A{}^j N_j,$$

where N_j (j = 1, ..., n) are shape functions (basis functions or interpolation functions) [1, 2] and all or the most of the parameters $\phi_A{}^j$ remain unknown.

After dividing the domain Ω, the shape functions are defined locally for elements Ω^I. A typical finite element is triangular in shape and thus has three main nodes. It is easy to demonstrate that triangular subdomains fit better to the boundary Γ than others e. g. rectangular ones. Among the triangular elements family one can find linear, quadratic and cubic elements [1] (see also Appendix A). A choice of an appropriate type of subdomains depends on a desired order of approximation and thus arises directly from the continuum problem. The higher order of element, the better approximation. Each triangular element can be described in terms of its *area* coordinates L^i_1, L^i_2 and L^i_3.

There are general rules that govern the transformation from *area* coordinates to Cartesian coordinates

$$x = L_1x_1 + L_2x_2 + L_3x_3$$
$$y = L_1y_1 + L_2y_2 + L_3y_3$$
$$1 = L_1 + L_2 + L_3$$

$$\rightarrow \begin{pmatrix} L_1 \\ L_2 \\ L_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where set of pairs (x_1, y_1), (x_2,y_2), (x_3, y_3) represents Cartesian nodal coordinates. In turn the *area* coordinates are related to shape functions in a manner that depends on the element order. In further analysis only the linear triangular elements will be used. For them, the shape functions are simply the area coordinates (see Appendix A). Therefore, each pair of shape functions $N^i_k(x,y)$, $N^i_l(x,y)$ for k,l=1,2,3 could be thought as a natural basis of the Ω^i triangular element.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ A. Kendall, H. Weimin, *Theoretical Numerical analysis, A Functional Analysis Framework, Third Edition.*, Springer 2009

3.^ the Einstein summation convention

Finite Element Method. Mathematical Basis.

Integration upon Elements.

We shall consider the linear expression derived in the Appendix B[1, 2, 3]

$$\delta \phi_A{}^T(K\phi_A + f) = \int_\Omega \delta\phi(-\varepsilon \partial^2/\partial x^2 - \varepsilon\partial^2/\partial y^2)\phi(x, y)\, dxdy + \int_\Omega \delta\phi\rho(x, y)\, dxdy = 0$$

with the boundary condition $\phi = \gamma$ on Γ. In such a simply case of integral – differential problems with a differential operator $L = -\varepsilon\partial^2/\partial x^2 - \varepsilon\partial^2/\partial y^2$, the variable ϕ in the above – written equation only consists of one scalar function ϕ

which is the sought solution, while the constant vector *f* is represented by the last term in that expression. To find the solution for such a problem means to determine the values of $\phi(x,y)$ in the whole domain Ω. The values of ϕ on its boundary Γ are already prescribed to γ. On the other hand, at the very beginning (see Eq. (2) in the previous page) we have postulated that a function ϕ could be approximated by an expansion ϕ_A given by means of some basis functions $N_m(x,y)$, m = 1, ...,n (for more details see Appendix A). Thus another possibility to deal with the Poisson problem is just to start from the functional Π and build a set of Euler equations $\partial\Pi/\partial\phi_m = 0$ where *m = 1, ..., n* and ϕ_m approximates value of the solution ϕ calculated at the *m*-th mesh node.

$$\Pi = \int_{\Omega} \left[1/2\varepsilon \sum_l (\partial N_l/\partial x\, \phi^l)^2 + 1/2\varepsilon \sum_l (\partial N_l/\partial y\, \phi^l)^2 + \rho \sum_l N_l\, \phi^l \right] dxdy +$$

$$+ \int_{\Gamma} (\gamma - 1/2 \sum_l N_l \phi^l) \sum_k N_k\, \phi^k\, d\Gamma$$

and after that we calculate the derivative $\partial\Pi/\partial\phi_{A,m}$. Moreover, let's simplify our problem by neglecting the last term in the above – presented equation and imposing $\phi = \gamma$ on the boundary Γ instead. In that manner, one obtains the expression

$$\partial\Pi/\partial\phi_{A,m} = \int_{\Omega} [\varepsilon \sum_l (\partial N_l/\partial x\, \phi^l) \sum_k \partial\, N_k/\partial x\, \delta^k{}_m$$

$$+ \varepsilon \sum_l (\partial\, N_l/\partial y\, \phi^l) \sum_k \partial N_k/\partial y\, \delta^k{}_m + \rho \sum_l N_l\, \delta^l{}_m]\, dxdy = 0$$

or in a simplified form

$$\partial\Pi/\partial\phi_{A,\,m} = \sum_l \left(\int_\Omega \varepsilon(\partial N_l/\partial x\, \partial N^m/\partial x + \partial N_l/\partial y\, \partial N^m/\partial y)\, dxdy \right) \phi^l + \int_\Omega \rho N^m dxdy = 0.$$

It is worth mentioning that some requirements must be imposed on the shape functions *N*. Namely, if *n*-th order derivatives occur in any term of*L* then the shape functions have to be such that their *n-1* derivatives (pay an attention to the above – presented equation) are continuous and finite. Therefore, generally speaking C_{n-1} continuity of shape functions must be preserve.

In turn, after substituting

$$K^m_l = \varepsilon \int_\Omega (\partial N_l/\partial x\, \partial N^m/\partial x + \partial N_l/\partial y\, \partial N^m/\partial y)dxdy$$

$$f^m = \int_\Omega \rho N^m dxdy$$

finally one obtains a set of equations

$$\partial\Pi/\partial\,\phi_m = \sum_l K^m_l \phi^l + f^m = 0$$

for *m, l = 1, ..., n* or in matrix description

$$\partial\Pi/\partial\phi = K\phi + f = 0.$$

It is worth noticing that the matrix *K* is a symmetric one because of the symmetry in exchange of subscripts *l* and *m* in the equation.
Now, we are obliged to employ the division of our domain Ω into a set of subdomains Ω^i. It gives that

$$K^m_l = \sum_i K^{im}_l = \sum_i \int_{\Omega^i} \varepsilon(x,y)\, (\partial N^i_l(x,y)/\partial x \partial N^{im}(x,y)/\partial x + \partial N^i_l(x,y)/\partial y \partial N^{im}(x,y)/\partial y)\, dxdy$$

$$f^m = \sum_i f^{im} = \sum_i \int_{\Omega^i} \rho(x,y)\, N^{im}(x,y)\, dxdy.$$

Therefore, after the transformation to *I* subdomains the expression becomes

$$\sum_i K^{im}{}_l\, \phi^l + \sum_i f^{im} = 0$$

for $i = 1, ..., I$ and $m = 1, ..., n$. Or in matrix notation

$$\phi_A = -K^{-1} f$$

In fact, the summation written above takes into account only these elements Ω^i which contribute to *m*-th node, however, because of the consistency in notation all elements are included in the sum with the exception that those $N^i{}_m$ functions for which node *m* does not occur in *i*-th element are put equal zero. From now, the whole story is to calculate integrals

$$K^{im}{}_l = \int_{\Omega^i} \varepsilon(x, y)\, (\partial N^i{}_l(x, y)/\partial x\, \partial N^{im}(x, y)/\partial x + \partial N^i{}_l(x, y)/\partial y\, \partial N^{im}(x, y)/\partial y)dxdy$$

$$= \int_0^1 dL_1 \int_0^{1-L_1} dL_2\, \varepsilon(L_1, L_2, L_3)\, |\det J^i|\, (\nabla N_{il} T\, T^T\, \nabla^T N^{im}),$$

$$f^{im} = \int_{\Omega^i} \rho(x,y)\, N^{im}(x,y)\, dxdy = \int_0^1 dL_1 \int_0^{1-L_1} dL_2\, |\det J^i|\, \rho(L_1,L_2,L_3)\, N^{im}(L_1, L_2, L_3)$$

where $N^i{}_m = L^i{}_m$, $L_3 = 1 - L_1 - L_2$ (see Appendix A) whereas $\det(J^i)$ – the Jacobian of *i*-th element, T matrix together with ∇ operator in new coordinates

are evaluated in Appendix C.
An integration over the *i*-th subdomain Ω^i, which is a triangular element with three nodes, enforces the transformation from *n*-dimensional global interpolation to the local interpolations given by means of $N_{ik}(x,y)$ functions

where $i_k = 1, 2, 3$. That is why in equations new indices i_l, i_m appear which

further are allowed to take three possible values 1,2 and 3 for each element *i* (the local subspace).
As the next step, the Gauss quadrature is employed to compute above-written integrals numerically as it is described in the Appendix D.
And finally, after incorporating boundary conditions to equations by inserting appropriate boundary values of ϕ, the system of equations can be solved.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ A. Kendall, H. Weimin, *Theoretical Numerical analysis, A Functional Analysis Framework, Third Edition.*, Springer 2009

3.^ R. Courant, D. Hilbert, *Methods of Mathematical Physics, Volume 1*, Interscience Publisher, New York, 1953

Finite Element Method. Mesh generation.

Initial mesh

The domain Ω is a set of points in two – dimensional Euclidean plane $\Re^2$. The initial mesh should define the shape of the domain Ω or more precisely its boundary $\partial\Omega$. Let us denote the bd(Ω) as Γ. It could form a smooth curve (like a circle) or be a polygon. At the beginning it is necessary to assign the initial set of nodes belonging to Γ. Taking into account polygon it is obvious that the initial mesh must consist of its vertices, however, in the case of a smooth curve one can choose the initial mesh differently. In the article, the author concentrate on the polygonal domains (see Fig. 1) that can be formed from a smooth curve after placing some initial nodes on its boundary Γ and connecting them by line segments (chords).

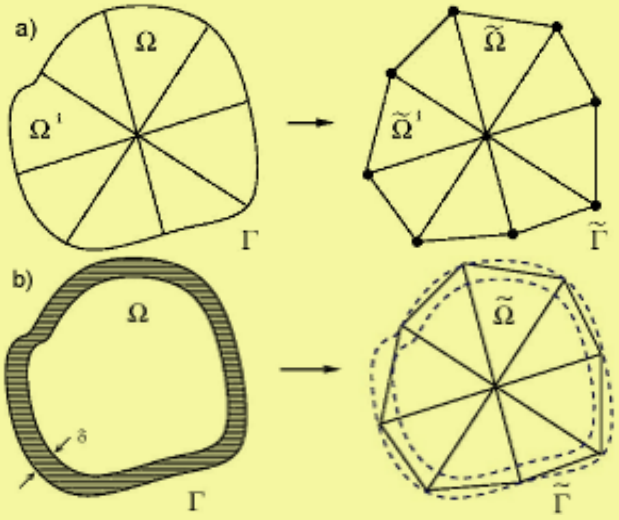


Fig. 1. Figure presents the domain Ω_A and its boundary Γ_A after projection to the polygonal domain. It has eight boundary nodes and one central node. Comparing both the initial Ω and the polygonal Ω_A domain one can notice that such a simple projection gives rather rough correspondence between them a), however, in some cases it could be a sufficient one i. e. when an integrated function changes very slowly in some δ – thick neighborhood of the boundary Γ b).

Let us start with determining the principal rectangular super domain as a Cartesian product

$$[x_{min}, x_{max}]\times[y_{min}, y_{max}]:=\{(x, y):x\in [x_{min}, x_{max}], y\in [y_{min}, y_{max}]\}$$

where the domain Ω is embedded in this super domain. This object is introduced due to a mesh creation procedure presented here.

$$\Omega\in [x_{min}, x_{max}]\times[y_{min}, y_{max}]$$

The following function $mesh_{init}$ (vertices, radius) where the variable *vertices* determines the number of its sides and the second one gives the *radius* of its circumscribed circle. For instance, one can make use of the Octave GNU project (free open source) and create both the initial nodes *p* and the initial triangles *t* arrays in the case of regular polygon of *N* vertices and lying within a circumscribed circle with a given radius. Following further steps of the algorithm presented in next sections, one can obtain meshes for different domains Ω (see few examples in Fig. 2).

Let us introduce a measure that estimates an element area in respect to the prescribed element area S designed by the element size h. The measure $S_N = S_{elem}/S$ gives a normalized area for each element. An estimation of the average deviation from assumed value of the element area provides information of mesh quality in the case for their fairly uniform distribution.

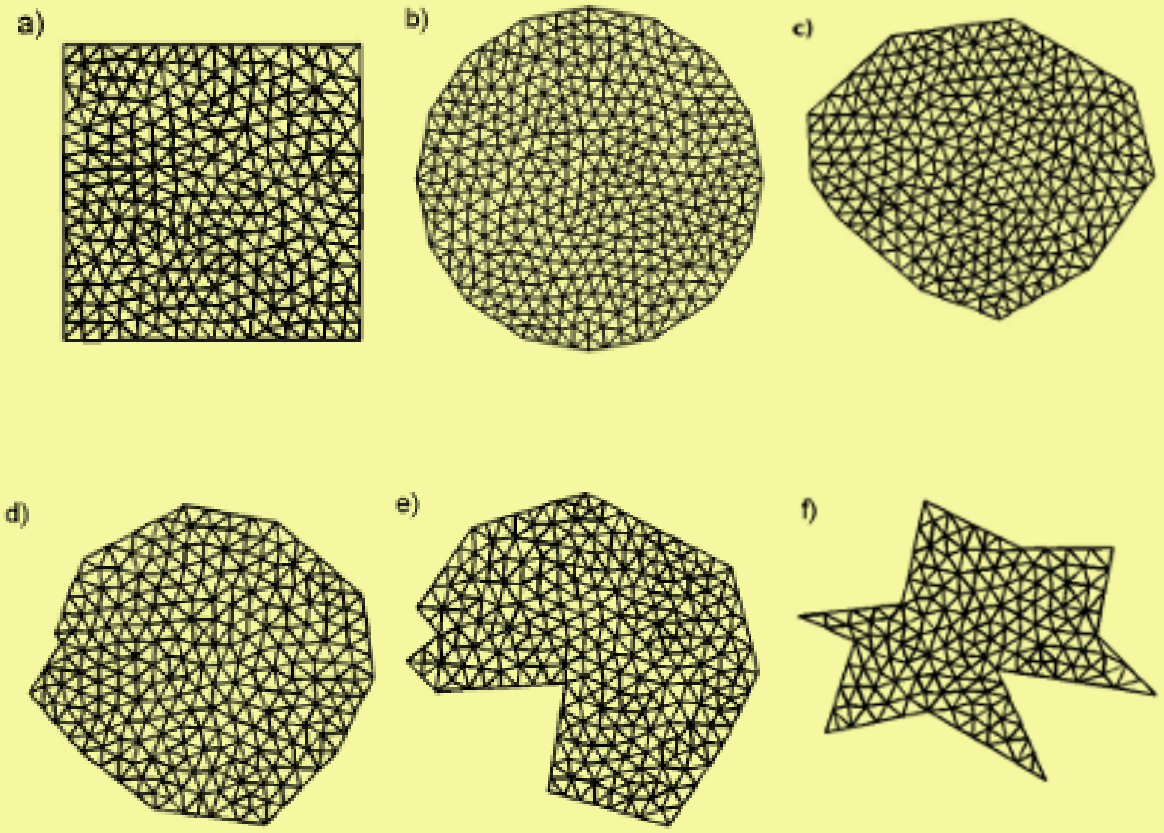


Fig. 2. Figure shows four domains Ω having different shapes. In brackets, finally established set of parameters is written: N_p – number of mesh points, $N_{divisions}$ – number of divisions (according to Sec. 3.2), $S_{A,\,N}$ – a normalized average element area are presented; a) regular polygon – square (258, 8, 1.002); b) regular polygon with 16 nodes (376, 6, 1.026) which approximates circular shape well; c) non – regular, convex figure (315, 8, 1.01); d) non – regular, semi – convex figure (247, 6, 1.071); and two non – regular, non – convex figures e) (245, 7, 0.993) and f) (164, 6, 1.0003) both with weight = [0.25, 0.75].

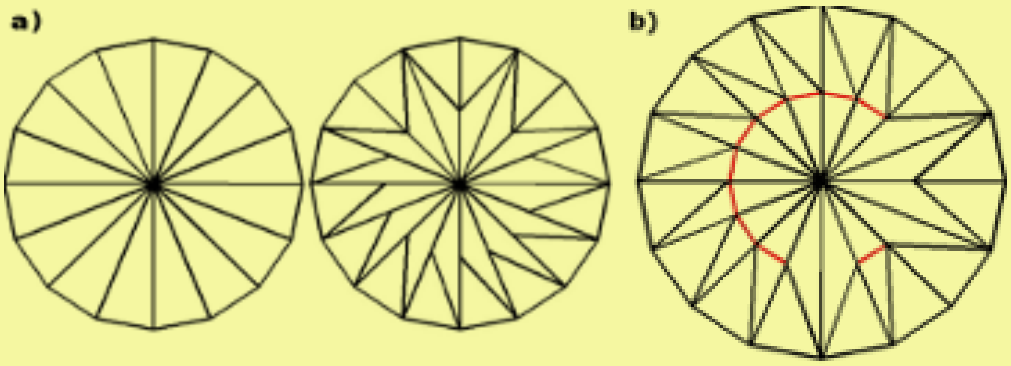
References

1.^ The source code of Octave is freely distributed GNU project, for more info please go to the following web page of Octave.

Mesh generator.

Adding new nodes to the mesh

In this section, let us start with the procedure that allows us to add new mesh nodes to the existing ones.The initial configuration of the nodes were already defined. It **must** define well the shape of the divided area in aspects explained in the description of the Figure 3. These *initial* nodes are called the *constant nodes* and are kept immobile through the rest of the algorithm steps. Each triangle could be split up into two new triangles by adding a new node to its longest bar. To avoid producing triangles much smaller than defined by the element *size h* only part of them could be broken up i. e. these for which the triangle area is one and half times bigger than A. That condition is set in the algorithm by introducing a new control parameter C_{split}. The new node is added in the middle of the triangle longest bar.



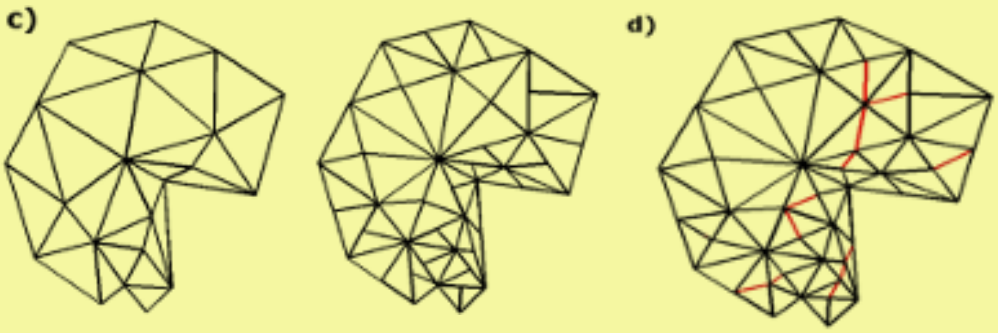


Fig. 3 presents a division process of non – regular and circular domains together with their boundaries. Pictures a) and c) show meshes with new nodes. Some of them are of the *illegal* type (defined in Sec. 3.2). These nodes constitute starting points for next complementary division that transforms such not well – defined elements into the correct ones, see pictures b) and d).
There is a need to underline that presented above algorithm is not quite optimal because some of the new nodes could produce triangles with one edge divided by a node resulting from splitting up an adjacent triangle (see Fig. 3). Such triangles are not desirable [1] and are denoted as *illegal* (see Figure 3a) and Figure 3c)). Thus the previous procedure needs to be improved. Let us add a few extra steps to it:
For each triangle $T_k \in \Omega$ perform checking whether it is of *illegal* type. If so,

split it up into two new properly defined triangles by connecting so – called illegal node with the vertex of T_k lying oppositely to it. Remove the old T_k

triangle. That is it.
Figures Fig 3b) and Fig 3d) show meshes having only desired elements.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

Mesh generator.

The boundary of the domain

The one of the most important issues to define is the domain boundary. After determining the boundary Γ_A by the initial *constant* nodes (lines 1-18 of the

presented below algorithm), the next task is to determine which new nodes are lying on boundary line segments Γ (as it is visible in Fig. 4). These selection is done with a help of the following algorithm:

1.For an initial node table *p* (nodes from 1 to N) find all pairs of neighbouring vertices.

2.Connect them by a segment line. If $x_1 - x_2 \neq 0$ then a function $y=ax+b$

exists and one can find pairs *a, b* for each such a line segment otherwise a vertical line $x = a$ together with limits $[y_1, y_2]$ must be found.

3.Establish the table of coefficients *a, b*.

4.For each new node check whether its coordinates (*x, y*) fulfill any of $y=ax + b$ equations or $x = a$ where $y < y_2$ and $y > y_1$

5.If yes classify it as a boundary node else classify it as an internal node.

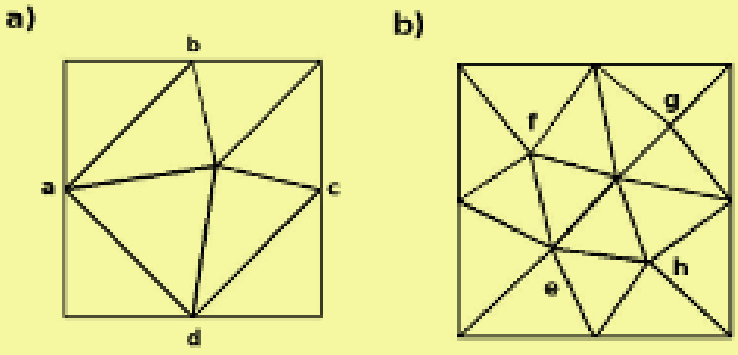


Fig. 4 presents the square domain divided into a set of new elements Ω^i with corresponding set of line segments Γ^i being its boundary. A way of finding new nodes constitutes the main point of the mesh generation process (see Sec. 3.2) while a selection of nodes is perform according to the algorithm from Sec. 3.3 a) nodes *a,b,c,d* have been classified as boundary nodes whereas b) nodes *e,f,g* have been determined as internal nodes.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

Generator Optimization via the Metropolis method

Let us define the set of mesh triangles $\Omega = \{ T_j, j = 1, 2, ..., M \}$ and a set T^i of triangle mesh elements to which a node p_i belongs. The *closest neighbors* $C(p_i)$ of the mesh point p_i are defined as a subset of mesh points $p_j \in T^i$

$$\forall\, p_i \,\exists\, T^i \subset \Omega: p_i \in T^i \qquad C(p_i) = \{ p_j: p_j \in T^i \text{ where } p_j \neq p_i \}.$$

Note, that the *closest region* is not the same what *the Voronoi region* [1]. Presented definition is needed to proceed with the Metropolis algorithm [2] which will be applied in order to adjust triangle's area to the desired size given by the element *size h*.
In turn, a proper triangulation is the essence of the finite element method as it is stated in the Sec. 2. Let us divide the whole problem into two different tasks. The first one focuses on finding an optimization for mesh elements being *the internal elements* whereas the second one is developed for so – called *the edge elements*. They are the elements for which one triangle's bar belongs to the boundary Γ of the domain Ω. It is assumed that a proper triangulation gives a discrete set of triangles T_j which approximates the domain Ω well.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^N. Metropolis, S. Ulam, *The Monte Carlo Method*, J. Amer. Stat. Assoc., **44**, No. 247., pp. 335-341, 1949

Generator Optimization via the Metropolis method

Internal mesh elements

Presented method is based on the following algorithm[1, 2]:

1. Define the element *size h* and consequently the element *area A*.
2. Initialize the configuration of triangles and then select the *internal* nodes $P_{int} = \{ p_i : p_i \in P \wedge p_i \notin \Gamma \}$ i. e. these nodes does not belong to the domain boundary Γ.
3. For each node p_i in P_{int} find its subdomain Ω^i defined as a set of triangles T_i to which the node p_i belongs.
4. Perform the Metropolis approach to every *internal* node p_i within its subdomain Ω^i. The Metropolis algorithm is adopted in order to adjust an area of each triangle in the node's subdomain to prescribed value *A* by shifting the position of the node p_i (Fig. 5 demonstrates robustness of the Metropolis approach; compare the node distribution in **a)** and in **b)**). That adjustment is governed by the following rules:
   4.1. Find an area of each triangle A_k (where $k = 1, 2, ..., K$) in Ω^i together with the vectors $\mathbf{r_{ji}} = \mathbf{p_i} - \mathbf{p_j}$ for each $p_j \in \Omega^i$ connected to node p_i
   4.2. Calculate the length of each triangle edge $|\mathbf{r_{ji}}|$ and its deviation $\delta |\mathbf{r_{ji}}|$ from the designed element size *h* i. e. $\delta |\mathbf{r_{ji}}| = |\mathbf{r_{ji}}| - h$
   4.3. Calculate the new position of the node $\mathbf{p^{new}_i}$ as

$$\mathbf{p^{new}_i} = \mathbf{p_i} - \sum_j F_j \, \delta |\mathbf{r_{ji}}| \, \mathrm{versor}(\mathbf{r_{ji}})$$

where $\mathrm{versor}(\mathbf{r_{ji}})$ (i. e. an unit vector) has the standard meaning as $\mathbf{r_{ji}}/|\mathbf{r_{ji}}|$ and F_j are weights corresponding to magnitude of *j*-th force applying to node p_i. Finally, they were set to the constant value *F*.

   4.4. Find an area of each triangle $A^{new}_k \in \Omega^i$ after shifting $p_i \rightarrow p^{new}_i$

4.5.Apply an energetic measure *E* to a sub – mesh Ω^i. That quantity could be understand in terms of a square deviation of a mesh element area from the prescribed element *area A*. Therefore, in the presented paper the δE is defined as a sum of a discrepancy between each triangle *area* A_k and *A* after moving node p_i and prior it, respectively

$$\delta E = \sum_k ((A^{new}_k - A)^2 - (A_k - A)^2) \; k = 1, 2, ..., K.$$

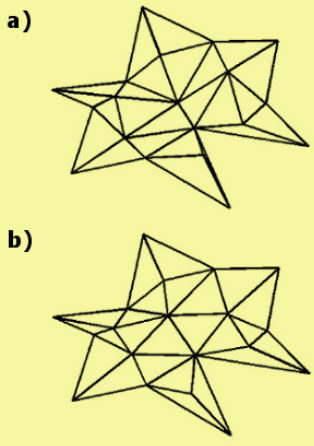
If the obtained value of an *energetic* change is lower than zero the change is accepted. Otherwise, the Metropolis rule is applied i. e. the following condition is checked

$$e^{-\delta E/T} > r$$

where *r* is an uniformly distributed random number on the unit interval *(0, 1)* and *T* denotes temperature.

4.6.The above – presented algorithm is repeated unless an assumed tolerance will be achieved.

In order to reach a better convergence of the presented method several other improvements could be adopted. For instance, the change in the length of the triangle edge could be an additional measure of mesh approximation goodness. That condition will ensure a lack of elongated mesh elements i. e. elements with very high ratio of its edge lengths (to see such *skinny* elements look at Fig. 5a)).



NODES	$N_{METROPOLIS}$	
	$F_j = 0.006$	$F_j = 0.1$
1	121	4
2	168	28
3	98	23
4	328	22
5	149	29
6	537	35
7	377	25

Figure shows an application of *the Metropolis algorithm*. Picture a) presents initial positions of new nodes just after generating them whereas picture b) shows their positions after node shifts according to the procedure described in Sec. 4.1 with the following two values of the force strength F_j: *0.006* and *0.1*

applied to each internal node *j = 1, ..., 7* and temperature set as *T = 0.01*. The table presents the total number of Metropolis steps that was required to obtain the final result shown in b).

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ N. Metropolis, S. Ulam, *The Monte Carlo Method*, J. Amer. Stat. Assoc., **44**, No. 247., pp. 335-341, 1949

Generator Optimization via the Metropolis method

Boundary mesh elements

The Metropolis algorithm[1, 2] applied to boundary nodes slightly differs from the above– described case and could be summarize in the following steps:

1.Find all the boundary or edge nodes i. e. nodes for which $p_{k,\,edge} \in \Gamma$.

2.Find triangles in the closest neighborhood of the considered $p_{k,edge}$ node. Then calculate an area of each triangle $A_{l,edge}$.

3.Calculate the force acting on each boundary node and coming **only** from nodes connected to it (as previously).

$$\mathbf{F_k} = -\sum_{j=1}^{J} F_{boundary}\,\delta\,|\mathbf{r_{jk}}|\,\mathrm{versor}(\mathbf{r_{jk}})$$

where J denotes the total number of nodes linked to the *k-th* node and $\delta|\mathbf{r_{jk}}|$ is defined as previously. Let us impose the following constrain on the motion of the *k-th* node in order to keep it in the boundary Γ. The force must be tangential to the boundary Γ so the boundary projection of the force $\mathbf{F_k}$ must be found:

$$\mathbf{F_{k,\Gamma}} = \text{versor}(\mathbf{L_\Gamma})\,(\mathbf{L_\Gamma}\cdot\mathbf{F_k}) / |\mathbf{L_\Gamma}|$$

where $\mathbf{L_\Gamma}$ denotes a vector lying along boundary Γ.

4. Similarly, find an area of each triangle $A^{new}_{l,edge}$ after shifting $p_{k,edge} \rightarrow p^{new}_{k,edge}$ according to the force F_k.

5. Adopt the Metropolis energetic condition to the boundary case i.e.

$$\delta E = \sum_l ((A^{new}_{l,\,edge} - A)^2 - (A_{l,\,edge} - A)^2)\ l = 1, 2, ..., L.$$

If

$$e^{-\delta E/T} > r$$

the new configuration is accepted otherwise is rejected. *T* denotes temperature and a random number $r \in U(0; 1)$ as previously.

6. The main point of this part is to ensure that the boundary nodes are moved just along the boundary Γ.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ N. Metropolis, S. Ulam, *The Monte Carlo Method*, J. Amer. Stat. Assoc., **44**, No. 247., pp. 335-341, 1949

The Lagrange polynomials.

The Lagrange polynomials $p_k(x)$ are given by the general formula [1, 2]

$$p_k(x) = \prod_{\substack{i=1 \\ i \neq k}}^{n} (x - x_i)/(x_k - x_i)$$

for *k = 1, ..., n*.
It is clearly seen from the above – given expression that for $x = x_k$ $p_k(x_k) = 1$ and for $x = x_j$ such that $j \neq k$ $p_k(x_j) = 0$. Between nodes values of $p_k(x)$ vary according to the polynomial order i. e. n-1 which is the order of interpolation. Making use of these polynomials one can represent an arbitrary function $\phi(x)$ as

$$\phi(x) = \sum_k p_k(x)\, \phi_k$$

On the other hand, when the interpolated function ϕ depends on two spacial coordinates one can define basis polynomials in the form

$$p_m(x, y) \equiv p_{IJ}(x, y) = p_I(x)\, p_J(y),$$

where *I J* describe row and column number for the *m-th* node in a rectangular lattice (rows align along x and columns along y direction, respectively). And consequently, the set $\{p_1, ..., p_m, ..., p_n\}$ is a basis of a n – dimensional functional space because each function p_m for *m = 1, ..., n* equals 1 at the interpolation node (x_m, y_m) and 0 at others. It is easy to demonstrate that such functions are orthogonal[2]. Instead of spacial coordinates any other coordinates can be considered. In the case of mesh elements the natural coordinates are the area coordinates *L* defined already in the Sec. The mathematical concept of FEM. On that basis the shape functions could be constructed as a composition of these three basis polynomials i. e. $N_m(L_1,L_2,L_3) = p^a{}_I(L_1)p^b{}_J(L_2)p^c{}_K(L_3)$ where the values of *a, b and c* assign the polynomial order in each L_k*-th* coordinate and *I, J and K* denote the *m-th* node

position in a triangular lattice (i. e. in the coordinates L_1, L_2 and L_3, respectively).
In the [1] could be found a comprehensive description of various elements belonging to the triangular family starting from a linear through quadratic to cubic one. For simplicity, in the article only the linear case is looked on. It explicitly means that shape functions $N_k = L_k(x, y)$, where *k = 1, 2, 3,* change between two nodes linearly (see Eq. (3)).

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^ A. Kendall, H. Weimin, *Theoretical Numerical analysis, A Functional Analysis Framework, Third Edition.*, Springer 2009

Variational principles

We shall now look on the left hand of the Eq. (1) i. e. the integral expression Π

$$\Pi = \int_{\Omega} L(u, \phi, \partial\phi/\partial x, \partial\phi/\partial y, ...)\, d\Omega.$$

We are aimed at determining the appropriate ϕ continuous function for which the first variation $\delta\Pi$ vanish. If

$$\delta\Pi = \kappa\, (d/d\kappa\, \Pi[\phi + \kappa\eta])_{\kappa = 0} = 0$$

for any $\delta\phi$ then we can say that the expression Π is made to be *stationary* [1].
The function ϕ is embed in a family of functions
$\phi+\delta\phi=\phi(x, y) +\kappa\eta(x, y)$
with the parameter κ. The variational requirement (equation above) gives vanishing of the first variation for any arbitrary η. In the presented article, the variational problem is limited to the case in which values of desired function ϕ at the boundary of the region of integration i. e. at the boundary curve Γ are assumed to be prescribed. Generally, the first variation of has the form

$$\delta\Pi = \partial\Pi/\partial\phi\ \delta\phi + \partial\Pi/\partial\phi_x\ \delta(\phi_x) + \partial\Pi/\partial\phi_y\ \delta(\phi_y) + \ldots$$

and vanishes when

$$\partial\Pi/\partial\phi = 0,\ \partial\Pi/\partial\phi_x = 0,\ \partial\Pi/\partial\phi_y = 0,\ \ldots$$

The condition above – presented gives the Euler's equations. Moreover, if the functional is *quadratic* i.e. if all its variables and their derivatives are in the maximum power of 2, then the first variation of has a standard linear form

$$\delta\Pi \equiv \delta\,\phi^T(K\,\phi + f) = 0,$$

which represents, though, in matrix notation. A vector ϕ denotes all variational variables i. e. ϕ and its derivatives as it is written in Euler eqs. **K** denotes stiffness matrix (the FEM nomenclature [2]) and **f** is a constant vector (does not depend on ϕ). We are interested in finding solutions to the Poisson and the Laplace differential equations under some boundary conditions. These classes of differential problems can be represent in such a general linear form. Now, we construct a functional Π which the first variation gives the Poisson – type equation. Firstly, we define the functional Π in the form:

$$\Pi = \int_\Omega [\varepsilon/2\ (\partial\phi/\partial x)^2 + \varepsilon/2\ (\partial\phi/\partial y)^2 + \rho\phi]\ dxdy + \int_\Gamma (\gamma - 1/2\phi)\phi d\Gamma,$$

where $d\Gamma = (dx^2 + dy^2)^{1/2}$, ρ, γ and ε can be functions of spacial variables *x* and *y*. Secondly, we find the first variation of Π

$$\delta\Pi = \int_\Omega [\ \varepsilon\ \partial\phi/\partial x\ \delta\phi_x + \varepsilon\ \partial\phi/\partial y\delta\phi_y + \rho\delta\phi]\ dxdy + \int_\Gamma (\gamma - \phi)\delta\phi d\Gamma,$$

where $\delta\phi_x = \partial(\delta\phi)/\partial x$. And after integration by parts and taking advantage of the Green's theorem [2] one can simplify the above – written equation to the form

$$\delta\Pi = \int_{\Omega} [\, -\varepsilon\, \partial^2\phi/\partial x^2 - \varepsilon\, \partial^2\phi/\partial y^2 + \rho]\, dxdy + \int_{\Gamma} \varepsilon\, \delta\phi\, \partial\phi/\partial n\, d\Gamma + \int_{\Gamma} (\gamma - \phi)\delta\phi d\Gamma = 0,$$

where $\partial\phi/\partial n$ denotes the normal derivative to the boundary Γ. The expression within the first integral constitutes the Poisson equation

$$-\varepsilon\, \partial^2\phi/\partial x^2 - \varepsilon\, \partial^2\phi/\partial y^2 + \rho = 0 \text{ in } \Omega$$

whereas the second term in the main equation gives the Neumann boundary condition

$$\varepsilon\, \partial\phi/\partial n = 0 \text{ on } \Gamma$$

and the third one represents the Dirichlet boundary condition

$$\phi = \gamma \text{ on } \Gamma$$

Note. The above – presented calculation demonstrates a way in which one can incorporate the boundary conditions of Neumann or Dirichlet type into a variational expression Π. However, an appropriately formulated boundary – value problem must include only one kind of b.c. (Neumann or Dirichlet b.c.) defined on the whole boundary Γ or it is permitted to mix them but only in not self – overlapping way.

References

1.^ R. Courant, D. Hilbert, *Methods of Mathematical Physics, Volume 1, Interscience Publisher,* New York, 1953

2.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

Transformation in local *L* -coordinates

Let us compute the determinant of the Jacobian transformation between the global *x, y* and a local L_1, L_2, L_3 coordinate frame. One notices immediately that the problem is degenerate. That is why, we introduce a new coordinate *z*

as a linear combination of L_1, L_2, L_3 i. e. $z = L_1 + L_2 + L_3$. Note that z is not an independent coordinate and has a constant value equal 1. After taking into account relations Eq. (3) we find the Jacobian matrix in the form

$$J(L_1, L_2, L_3) \equiv \partial(x, y, z)/\partial(L_1, L_2, L_3) =$$

$$\begin{pmatrix} \partial x/\partial L_1 & \partial x/\partial L_2 & \partial x/\partial L_3 \\ \partial y/\partial L_1 & \partial y/\partial L_2 & \partial y/\partial L_3 \\ \partial z/\partial L_1 & \partial z/\partial L_2 & \partial z/\partial L_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}$$

Furthermore, we have the relation between the Jacobian and an element area

$$\det(J) \equiv 2\Delta,$$

where Δ denotes the area of a triangle which is based on vertices $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. And finally, we obtain the coordinates transformation rule

$$dxdy = 2\Delta dL_1 dL_2, \text{ and } L_3 = 1 - L_1 - L_2.$$

The relation between the gradient operator Δ in cartesian and in new coordinates is given by:

$$[\partial/\partial x, \partial/\partial y] = [\partial L_1/\partial x\, \partial/\partial L_1 + \partial L_2/\partial x\, \partial/\partial L_2 + \partial L_3/\partial x\, \partial/\partial L_3,$$

$$\partial L_1/\partial y\, \partial/\partial L_1 + \partial L_2/\partial y\, \partial/\partial L_2 + \partial L_3/\partial y\, \partial/\partial L_3]$$

$$[\partial/\partial x, \partial/\partial y] = 1/(2\Delta)\, [\partial/\partial L_1, \partial/\partial L_2, \partial/\partial L_3] \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{pmatrix} = [\partial/\partial L_1, \partial/\partial L_2, \partial/\partial L_3]\, T$$

where $L_k = (a_k x + b_k y + c_k)/(2\Delta)$ $(k = 1, 2, 3)$ and $a_1 = y_2 - y_3$, $b_1 = x_3 - x_2$, $c_1 = x_2 y_3 - x_3 y_2$, the rest of coefficients is obtained by cyclic permutation of

indices *1, 2* and *3*.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

Numerical integration - Gauss's quadrature

The l.h.s integral **I** can be approximated by the *Q - point* Gauss quadrature [1, 2, 3, 4]

$$I = \int_0^1 dL_1 \int_0^1 dL_2 \, |\det J| \, f(L_1, L_2, L_3) \sim \sum_{q=1}^{Q} f_q(L_1, L_2, L_3) W^q$$

where W_q denotes the weights for *q - points* of the numerical integration, and can be found in the Table 5.3 in [1]. As it was already said, a set of $N_k(L_1, L_2, L_3)$ shape functions where *k = 1, 2, 3* can be used to evaluate each **f** function in the interpolation series which, for instance, in the highest order 10 – nodal cubic triangular element has the following form

$$f(L_1, L_2, L_3) = \sum_{k=1}^{3} N_k(L_1, L_2, L_3) f^k + \sum_{k=4}^{9} N_k(L_1, L_2, L_3) \Delta f^k + N_{10} \Delta\Delta f^{10}$$

where f_k are nodal values of **f** function, f_k denote departures from linear interpolation for mid – side nodes, and f_{10} is departure from both previous orders of approximation for the central node[C] [1]. For linear triangular elements only the first term is important which gives an approximation

$$f(L_1, L_2, L_3) = \sum_{k=1} L_k f^k$$

Note, that the r.h.s sum does not include the Jacobian j det J_j that should be incorporated by the weights W_q but it is not (in their values given in Table 5.3 from [1]). Thus let's add the triangle area to the above – recalled formula

$$|\det J|/(2\Delta) \sum_{q=1} f_q(L_1, L_2, L_3) W^q$$

and in that way we end up with the final expression for the Q – *point* Gauss quadrature.

References

1.^ O. C. Zienkiewicz, R. L. Taylor and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals, Sixth edition.*, Elsevier 2005

2.^R. Radau, *Etude sur les formules d'approximation qui servent calculer la valeur d'une integrate definie*, Journ. de Math. 6(3), pp. 283-336, 1880

3.^P. C. Hammer and O. J. Marlowe and A. H. Stroud, *Numerical Integration Over Simplexes and Cones*, Math. Tables Aids Comp., 10, pp. 130-137, 1956

4.^F. R. Cowper, *Gaussian quadrature formulas for triangles*, Int. J. Numer. Meth. Eng., 7, pp. 405-408, 1973

Delaunay triangulation algorithm

Let's remind briefly the main points of the Delaunay triangulation method [1] together with their numerical implementation using Octave and Matlab software [2]. Let $P = \{p_i, i = 1, 2, ... ,N\}$ be a set of points in two – dimensional Euclidean plane $\Re^2$. They are called *forming points of mesh* [1]. Let us define the triangle T as a set of three mesh points

$$T = \{t_j \in P, j = 1,2,3\}.$$

Definition of Delaunay zones

Using the Delaunay criterion one can generate triangulation where no four points from the set of forming points P are co – circular:

$$\forall\, p_i \in P \wedge p_i \notin T \quad ||p_i - u|| > \rho^2$$

where *u* is the center of the *T* triangle and ρ is its radius. The proposed algorithm consists of the following steps:

- The triangle's bars are given by the following vectors $\mathbf{t_{12}}$, $\mathbf{t_{13}}$, $\mathbf{t_{23}}$ where $\mathbf{t_{ij} = t_j - t_i}$, $\mathbf{t_i} = [t^x_i, t^y_i, 0]$ for each $i \neq j$ and $i, j \in \{1, 2, 3\}$.

- The cross product of each triangle bars defines a plane. The pseudo vector **A** together with its projection on the normal to the plane *n* – direction A_n are found

$$A = t_{12} - t_{13}$$

$$A_n = n \cdot A$$

in order to determine the triangle orientation. If the quantity $A_n > 0$ the triangle orientation is clockwise, otherwise is counterclockwise.

- The determinant of the square matrix *D(T)* is built on the basis of the set of triangle's nodes given by the equation

$$D(T) = \det \begin{pmatrix} t^x_1 & t^y_1 & (t^x_1)^2 + (t^y_1)^2 \\ t^x_2 & t^y_2 & (t^x_2)^2 + (t^y_2)^2 \\ t^x_3 & t^y_3 & (t^x_3)^2 + (t^y_3)^2 \end{pmatrix}$$

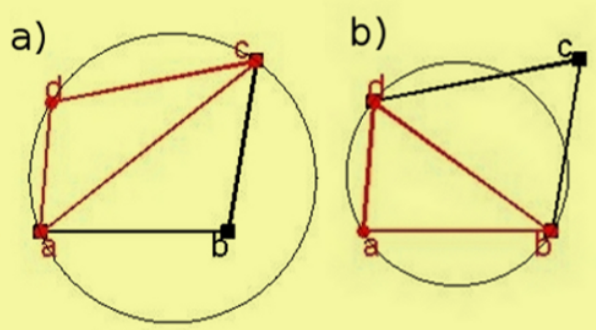


Figure shows the main idea of the Delaunay criterion. a) Two triangles (with nodes **abc** and **acd**, respectively) are not Delaunay triangles, b) after exchange of the edge **ac** to the edge **bd** two new triangles **abd** and **bcd** replace the old ones. They are both of the Delaunay type. Circles represent the Delaunay zones.
next the following determinant is calculated in order to find out whether a mesh point p_i is outside or inside the Delaunay zone (see Fig. 13)

$$D(T)_i = \det \begin{pmatrix} t^x_1 & t^y_1 & (t^x_1)^2 + (t^y_1)^2 & 1 \\ t^x_2 & t^y_2 & (t^x_2)^2 + (t^y_2)^2 & 1 \\ t^x_3 & t^y_3 & (t^x_3)^2 + (t^y_3)^2 & 1 \\ p^x_i & p^y_i & (p^x_i)^2 + (p^y_i)^2 & 1 \end{pmatrix}$$

for each $p_i \in P \wedge p_i \notin T$.

●If for any point p_i its $D(T)_i < 0$ the triangle T is not the Delaunay triangle (see Fig. 13a). Then one need to find other triangles in the closest neighborhood of the triangle T corresponding to the number of p_i inside the Delaunay zone and recursively exchange the bars between T and each of them (see Fig. 13b).

●Finally, checking whether the new two triangles are the Delaunay triangles takes its place. If so, new ones are accepted unless the change is canceled.

The algorithm ends up with the new triangular mesh Ω_{new}.

If you wish you can have an insight into the program delaunayTakeTechEase.m that implements the above – presented algorithm using the Octave and Matlab software (it is a part of a free Octave and Matlab course). One can use also the appropriate Octave and Matlab library function.
Help. In order to find an orientation of a triangle *T* one can check the sign of A_n (see equation). If it is greater than 0 the triangle orientation is clockwise unless counterclockwise. In the latter case, to ensure the clockwise orientation one can once flip up and down matrix in equation then the triangle orientation turns into the opposite one. Obviously, this flipping results in the change of the sign of the matrix determinant $D(T) \rightarrow -D(T)$.

References

1.^ B. Delaunay, *Sur la sphre vide*, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7, pp. 793-800, 1934

2.^The source code of Octave is freely distributed GNU project, for more info please go to the following web page http://www.gnu.org/software/octave/.

Metfem2D

Last update: April 2, 2019