

UNIVERSITY OF ZAGREB
FACULTY OF MECHANICAL ENGINEERING AND NAVAL
ARCHITECTURE

**AN AUTOMATIC FINITE VOLUME MESH
GENERATOR FOR SYNTHETIC
MICROSTRUCTURE DISCRETIZATION**

MASTER THESIS

Supervisor:
Prof. dr. sc. Željko Tuković

Student:
Alen Cukrov

Zagreb, 2015

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Automatic Mesh Generation	1
1.2.1	The Delaunay triangulation method	2
1.2.2	The Advancing Front Method	2
1.2.3	Tree Methods – Quadtree (2D), Octree (3D)	3
1.3	Outline of the Thesis	3
2	An Automatic Mesh Generator	4
2.1	Introduction	4
2.2	The Algorithm	4
2.3	The Level-set Function of a Convex Polygon	8
2.4	Internal Boundaries	13
2.5	Conclusion	14
3	Examples	15
3.1	Introduction	15
3.2	Discussion	15
3.3	Conclusion	21
4	Conclusion and Perspectives	28
4.1	Conclusion	28
4.2	Perspectives	28
	References	30

I hereby certify that this thesis has been written by me using the knowledge obtained during my studies and the cited literature.

I would like to express my sincere appreciation to my thesis supervisor, Professor Željko Tuković. It has been a honour and a pleasure to work under the supervision of such exceptional professional, teaching and personal qualities.



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE



Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
procesno-energetski, konstrukcijski, brodstrojarski i inženjersko modeliranje i računalne simulacije

Sveučilište u Zagrebu	
Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

DIPLOMSKI ZADATAK

Student: **Alen Cukrov**

Mat. br.: 0035174572

Naslov rada na hrvatskom jeziku: **Automatski generator mreže kontrolnih volumena za diskretizaciju sintetičke mikrostrukture**

Naslov rada na engleskom jeziku: **An Automatic Finite Volume Mesh Generator for Synthetic Microstructure Discretisation**

Opis zadatka:

Advanced ceramics are a class of materials which exhibit superior properties to traditional ceramics, including abrasion resistance and high hardness. They have a two-phase structure composed of grains of a primary hard phase together with either a metallic or ceramic matrix material. The strength and toughness of these materials are affected by both the size of the primary phase and percentage matrix content. Therefore it is desirable to optimize these parameters to produce stronger or tougher materials. By specifying material microstructure virtually, the influence of individual material parameters on the microstructural scale can easily be investigated and altered to change bulk material properties.

In the scope of this thesis one has to develop an automatic finite volume mesh generation procedure for discretization of statistically representative two-dimensional microstructure produced using Voronoi tessellation. The simplified microstructure represented by a set of polygonal Voronoi cells should be discretized using triangular or polygonal finite volumes. The geometry of the microstructure will be defined by a distance function (implicit description) and non-uniform finite volume mesh resolution will be defined by in a sizing function specified in advance. The automatic mesh generation procedure should be implemented in OpenFOAM® and the representative example of a generated microstructure mesh should be used to perform a micromechanical finite volume analysis of the bulk material properties. This work will be performed in the scope of the research project entitled "Damage Prediction and Repair Reliability of Composite Structures and Adhesive Joints (DAMPREP)" supervised by professor Alojz Ivanković at the University College Dublin.

It is advised to list references used in this work, as well as to acknowledge help and support possibly received during the course of this study.

Zadatak zadan:

Rok predaje rada:


Predviđeni datumi obrane:

13. studenog 2014.

15. siječnja 2015.

21., 22. i 23. siječnja 2015.

Zadatak zadao:


Prof. dr. sc. Željko Tuković


Predsjednik Povjerenstva:
Prof. dr. sc. Zvonimir Guzović

List of Figures

1	Diskretizirana kružna domena.	xiv
2	Domena diskretizirana koristeći Booleove operacije.	xiv
3	Ploča s rupom.	xv
4	Mreža sa gradacijom.	xv
5	Geometrija (a) i diskretizacija (b) jedinične ćelije sa 6 vlakana.	xvi
6	Geometrija (a) i diskretizacija (b) ćelije sa 20 slučajno raspoređenih vlakana.	xvi
7	Geometrija (a) i diskretizacija (b) sintetičke mikrostrukture.	xvii
8	Raspodjela normalnog naprezanja σ_{xx} u materijalu sa mikrostrukturom.	xvii
2.1	Satisfied (a) and not satisfied (b) Delaunay criterion (Owen, 1998).	6
2.2	Forces: (a) internal forces and (b) external forces (Persson, 2006).	7
2.3	Repulsive forces acting on the edge endpoints.	7
2.4	Computing the signed distance to a convex polygon.	10
2.5	Only the point which has all distances negative is inside the polygon.	11
2.6	Points outside the polygon.	12
2.7	An edge which crosses the internal boundary. Right edge endpoint is closer to internal boundary and is thus projected.	14
3.1	Discretization of the unit circle.	15
3.2	Refined discretization of the unit circle.	16
3.3	Discretized ellipse.	16
3.4	Domain defined using union scheme.	17
3.5	Discretized surface with hole.	17
3.6	Domain created by intersecting two circles of constant radii.	18
3.7	Polygonal domains.	19
3.8	Square plate with a circular hole.	19
3.9	Graded mesh at specific zones.	20
3.10	The geometry (a) and the discretization (b) of the basic case with internal interface.	22
3.11	The geometry (a) and the discretization (b) of the unit square with 6 fibres.	23

3.12 The geometry (a) and the discretization (b) of the cell with 20 randomly distributed fibres.	24
3.13 The geometry (a) and the discretization (b) of the two-phase synthetic microstructure.	25
3.14 Shear stresses σ_{xx} (a) and σ_{yy} (b) distribution in the numerical microstructure.	26
3.15 Shear stress τ_{xy} (a) and displacement (b) distribution in the numerical microstructure.	27

List of Tables

3.1	Parameters of the unit circle meshes	16
3.2	Parameters of the ellipse mesh	17
3.3	Characteristics of the meshes defined by set operations	18
3.4	Material properties	21

List of Algorithms

1	DISTMESH algorithm by Persson and Strang	5
2	Signed distance function of a polygon	9
3	Internal boundaries generation on an existing mesh	13

Nomenclature

Symbol	Description	Unit
c	value of the counter	
d_Ω	distance function	m
dI_Ω	internal distance function	m
e	element's edge	
f	linear repulsive force	N
\mathbf{F}	total force	N
\mathbf{F}_{ext}	exterior force	N
\mathbf{F}_{int}	interior force	N
h_Ω	size function	1
$\hat{\mathbf{k}}$	unit vector pointing in the positive z direction	1
l_0	desired edge length	m
l_e	actual edge length	m
\mathbf{n}	normal vector	1
$N_{\mathbf{P}_{fix}}$	number of fixed points	
\mathbf{p}	point field	m
\mathbf{P}_{fix}	fixed point field	m
\mathbf{r}_P	domain boundary's edge vector	m
Ω	computational domain	m ²
$\partial\Omega$	computational domain's external boundary	m

Abstract

This thesis deals with development of an automatic mesh generator for discretization of materials with internal interfaces. The mesh generator developed in this thesis can create triangular finite volume meshes. The user is required to specify the signed distance function and mesh size function. The applied algorithm is based on a conjunction between the Delaunay triangulation and mechanical analogy. Extensions related to dealing with polygonal domains and handling the internal boundaries were done. Obtained results shown reliability of the method.

Extended Abstract (Croatian)

Kako bi se sustav parcijalnih diferencijalnih jednadžbi mogao numerički riješiti potrebno je najprije zadanu domenu diskretizirati, odnosno podijeliti na konačan broj kontrolnih volumena – mrežu. Prema navodu Versteega i Malalasekera (Versteeg & Malalasekera, 2007) na projektima vezanim uz računalnu dinamiku fluida preko 50% vremena se utroši na definiranje geometrije domene i generiranje mreže. Ukoliko domena u sebi sadrži unutarnje domene koje također treba diskretizirati, primjerice materijal sa svojom mikrostrukturom, generiranje mreže postaje znatno složenije. Stoga se ovim radom razvija automatski generator mreže konačnih volumena u svrhu izrade numeričkih studija vezanih uz materijale sa diskontinuitetima.

Metode automatskog generiranja numeričke mreže svrstavaju se u tri glavne skupine (Zienkiewicz *et al.*, 2005):

- metoda Delaunayjeve triangulacije,
- metoda napredujuće fronte i
- metode stabla (kvartalno stablo u 2D i oktalno stablo u 3D).

Algoritam koji je korišten u radu najbliži je metodi Delaunayjeve triangulacije. Riječ je o DISTMESH algoritmu, kojeg su razvili Persson i Strang (Persson & Strang, 2004). Algoritam se temelji na kombinaciji analogije vlačnih opruga sa Delaunayjevom triangulacijom, i rezultira kvalitetnim trokutnim konačnim volumenima.

Kako bi se diskretizirala domena Ω algoritam zahtjeva od korisnika definiranje:

- funkcije udaljenosti d_Ω za definiranje geometrije domene
- funkcije veličine h_Ω za definiranje gradacije mreže.

Za proizvoljnu točku \mathbf{p}_i vrijedi:

$$d_\Omega(\mathbf{p}_i) = \begin{cases} \min\{d(\mathbf{p}_i, \partial\Omega)\} & \text{za } \mathbf{p}_i \notin \Omega \\ 0 & \text{za } \mathbf{p}_i \in \partial\Omega \\ -\min\{d(\mathbf{p}_i, \partial\Omega)\} & \text{za } \mathbf{p}_i \in \Omega \end{cases} \quad (1)$$

gdje $\partial\Omega$ označava vanjsku granicu domene. Iz navedenog slijedi da funkcija udaljenosti vraća udaljenost između bilo koje točke u unaprijed definiranom graničnom pravokutniku (engl. bounding box) i najbliže vanjske granice domene. Negativan predznak vrijedi za točke unutar domene, dok pozitivan upućuje da je točka izvan domene. Pri $d_\Omega(\mathbf{p}_i) = 0$ točka se nalazi na vanjskoj granici domene.

Dakle, najprije se distribuira N početnih točaka unutar prethodno zadanog graničnog pravokutnika, te se iste pohranjuju u dvodimenzijско polje \mathbf{p} :

$$\mathbf{p} = [\mathbf{x} \ \mathbf{y}] \quad (2)$$

Točke se pomiču, a nakon toga se provjerava funkcija udaljenosti d_Ω za svaku točku. One točke koje imaju $d_\Omega(\mathbf{p}_i) \leq 0$ se zadržavaju. Zatim se funkcija veličine h_Ω ispituje za svaku od zadržanih točaka, te se odbacuju one točke koje pokazuju vjerojatnost proporcionalnu $1/h_\Omega(\mathbf{p}_i)^2$. Ovaj konačan skup točaka predaje se Delaunayjevom triangulacijskom algoritmu koji spaja zadržane točke.

Delaunayjeva triangulacija jest triangulacija skupa točaka pri čemu opisana kružnica pojedinog trokuta prolazi jedino kroz točke istog. Time Delaunayjeva triangulacija ujedno ostvaruje i maksimiziranje najmanjeg kuta na cijeloj triangulaciji što daje elemente visoke kvalitete. U sklopu procesa triangulacije, težišta svih elemenata su ispitana, te oni sa $d_\Omega > 0$ bivaju izbrisanimi.

Dobivena mreža fiktivno postaje mehanička struktura. Pritom su točke spojevi a spojnice dviju točaka (engl. edges) vlačne opruge. Kako su koordinate točaka zapravo nepoznanice, tako se iste dobivaju postizanjem statičke ravnoteže ovog sustava.

U svakoj točki mreže \mathbf{p}_i , može se definirati polje sile $\mathbf{F}(\mathbf{p}_i)$ kao suma svih sila koje djeluju na spomenutu točku:

$$\mathbf{F}(\mathbf{p}_i) = \mathbf{F}_{int}(\mathbf{p}_i) + \mathbf{F}_{ext}(\mathbf{p}_i) \quad (3)$$

gdje \mathbf{F}_{int} i \mathbf{F}_{ext} označavaju unutarnje i vanjske (reakcije na granicama) sile. Unutarnje sile koje djeluju na pojedinu točku se odnose na odbojne sile od svih spojnica povezanih na tu točku. Vanjske pak sile prisutne su jedino kod točaka na granicama domene, koje se nakon djelovanja unutarnjih sila nađu izvan domene. Valja istaknuti da su kod unutarnjih točaka ove sile jednake nuli, a kod fiksnih točaka su sve sile jednake nuli.

Ukupna unutarnja sila \mathbf{F}_{int} koja djeluje na točku \mathbf{p}_i glasi:

$$\mathbf{F}_{int}(\mathbf{p}_i) = \sum_e f(l_e, l_0) \quad (4)$$

gdje e označava sve spojnice povezane sa točkom \mathbf{p}_i , l_0 je željena duljina spojnice, a l_e je duljina spojnice koja spaja susjedne točke \mathbf{p}_i i \mathbf{p}_j :

$$l_e = \|\mathbf{p}_i - \mathbf{p}_j\| \quad (5)$$

Član pod znakom sumacije na desnoj strani jednadžbe (4) odnosi se na ranije spomenutu odbojnu silu koja djeluje na krajevima spojnica (tj. djeluje na točke):

$$f(l_e, l_0) = f^{ij} = \begin{cases} (l_0 - l_e)n_{i,j} & \text{if } l_e < l_0 \\ 0 & \text{if } l_e \geq l_0 \end{cases} \quad (6)$$

gdje je $n_{i,j}$ vektor normale:

$$n_{i,j} = \frac{\mathbf{p}_i - \mathbf{p}_j}{l_e} \quad (7)$$

Sila f^{ij} koja djeluje na točku \mathbf{p}_i jednaka je po iznosu, a suprotna po smjeru sili koja djeluje na susjednu točku \mathbf{p}_j :

$$f^{ij} = -f^{ji} \quad (8)$$

Naglasak na odbojnim silama je u svrhu distribucije točaka po cijeloj domeni. U prilog tome, željena duljina spojnice dviju točaka se postavlja na malo veću vrijednost od one koja je definirana funkcijom veličine h_Ω :

$$l_0 = h_\Omega(\mathbf{p}_{mid,e}) F_{scale} \sqrt{\frac{\sum_e l_e^2}{\sum_e h_\Omega(\mathbf{p}_{mid,e})^2}} \quad (9)$$

gdje $\mathbf{p}_{mid,e}$ označava točku na sredini spojnice, a F_{scale} je fiksni faktor. Persson i Strang (Persson & Strang, 2004) preporučaju vrijednost 1,2 za F_{scale} .

Kao što je naglašeno ranije, koordinate točaka mreže \mathbf{p} se dobivaju postizanjem statičke ravnoteže uvedenog fiktivnog mehaničkog sustava koja glasi:

$$\mathbf{F}(\mathbf{p}) = 0 \quad (10)$$

U tu svrhu, definirana je pseudo-nestacionarnost:

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}_{int}(\mathbf{p}), \quad t \geq 0 \quad (11)$$

Stacionarno rješenje jednadžbe (11) zadovoljava jednadžbu (10) i daje dobru raspodjelu točaka. Obična diferencijalna jednadžba (11) se rješava koristeći Eulerov eksplicitni algoritam:

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \Delta t \mathbf{F}_{int}(\mathbf{p}^n) \quad (12)$$

gdje \mathbf{p}^{n+1} su nove, a \mathbf{p}^n poznate koordinate točaka. Pseudo-vremenski korak je Δt , a \mathbf{F}_{int} je ukupna unutarnja sila.

Nakon djelovanja unutarnjih sila, neke točke završavaju izvan granica domene. Te točke se vraćaju na granicu domene postupkom projiciranja. Projiciranje proizvoljne točke \mathbf{p}_i na granicu glasi:

$$\mathbf{p}_i = \mathbf{p}_i - \frac{d_\Omega(\mathbf{p}_i)}{|\nabla d_\Omega|^2} \nabla d_\Omega \quad (13)$$

te se postupak projiciranja ponavlja dok se ne postigne $d_\Omega(\mathbf{p}_i) = 0$.

Postupak Delaunayjeve triangulacije i djelovanja sila, ponavlja se sve dok pomaci točaka ne spuste ispod neke unaprijed zadane vrijednosti.

Ovaj bazni algoritam implementiran je u C++ biblioteku OpenFOAM®. Samom implementacijom postignuta je mogućnost defniranja domena jedino preko kontinuiranih funkcija poput jednadžbe kružnice ili elipse. Kako je mikrostruktura koju se želi diskretizirati zapravo skup konveksnih poligona, tako se javlja potreba za razvojem algoritma koji bi davao udaljenost proizvoljne točke od najbliže granice. Izvorna MATLAB® implementacija u tu svrhu koristi ugrađenu "inpolygon" funkciju.

Stoga je razvijen algoritam kvadratne vremenske složenosti $O(n^2)$ koji koristeći vektorski račun daje udaljenost točke od najbliže granice. Pritom se podrazumijeva da je poligon konveksan, te da je definiran fiksnim točkama u smjeru suprotnom od smjera kazaljke na satu.

Razvijeni algoritam dostupan je u sklopu drugog poglavlja zajedno sa pripadajućim skicama, jednadžbama i objašnjenjima.

Nadalje, bazni algoritam u stanju je diskretizirati domenu načinjenu bez diskontinuiteta. Kako kompozitni materijali sadrže vlakna, tako diskretizacija takvih domena obvezuje na proširivanje algoritma u tom smjeru. Algoritam za izradu domene sa unutarnjim granicama dan je u četvrtom odjeljku drugog poglavlja.

Na generiranim mrežama uočavaju se gotovo izotropni elementi:

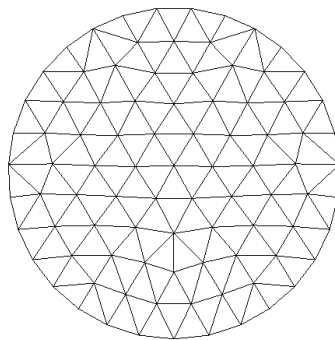


Figure 1: Diskretizirana kružna domena.

Kvalitetni trokutni elementi posljedica su primjenjene Delaunayjeve triangulacije, ali i djelovanja fiktivnih sila. Posebno se djelovanjem sila omogućava kvalitetna mreža na granicama domene što inače slovi kao problem Delaunayjeve triangulacije.

Algoritam podržava i definiranje geometrije koristeći Booleove operacije:

- uniju:

$$d_{\Omega_1 \cup \Omega_2}(\mathbf{p}) = \min(d_{\Omega_1}(\mathbf{p}), d_{\Omega_2}(\mathbf{p})) \quad (14)$$

- razliku:

$$d_{\Omega_1 \setminus \Omega_2}(\mathbf{p}) = \max(d_{\Omega_1}(\mathbf{p}), -d_{\Omega_2}(\mathbf{p})) \quad (15)$$

- presjek:

$$d_{\Omega_1 \cap \Omega_2}(\mathbf{p}) = \max(d_{\Omega_1}(\mathbf{p}), d_{\Omega_2}(\mathbf{p})) \quad (16)$$

Domena definirana koristeći Booleove operacije prikazana je na sljedećoj slici:

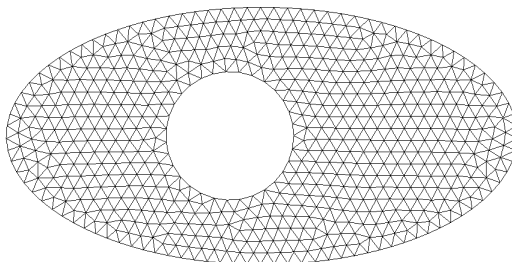


Figure 2: Domena diskretizirana koristeći Booleove operacije.

Koristeći algoritam za generiranje poligona i njegovom kombinacijom sa funkcijom udaljenosti za kružnicu preko Booleovih operacija, omogućeno je generiranje domene koja se može, primjerice, iskoristiti za analizu naprezanja:

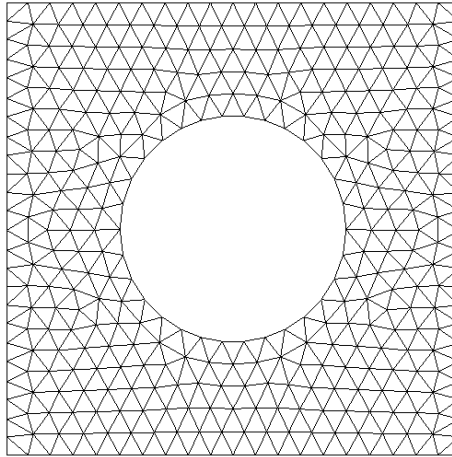


Figure 3: Ploča s rupom.

Na priloženim diskretiziranim domenama, uočava se gotovo uniformna mreža, tj. $h_{\Omega}(\mathbf{p}) = 1$ jest zadano. Kako funkcija veličine h_{Ω} opisuje relativni razmak između točaka mreže, tako je omogućena rafinacija pojedinih zona.

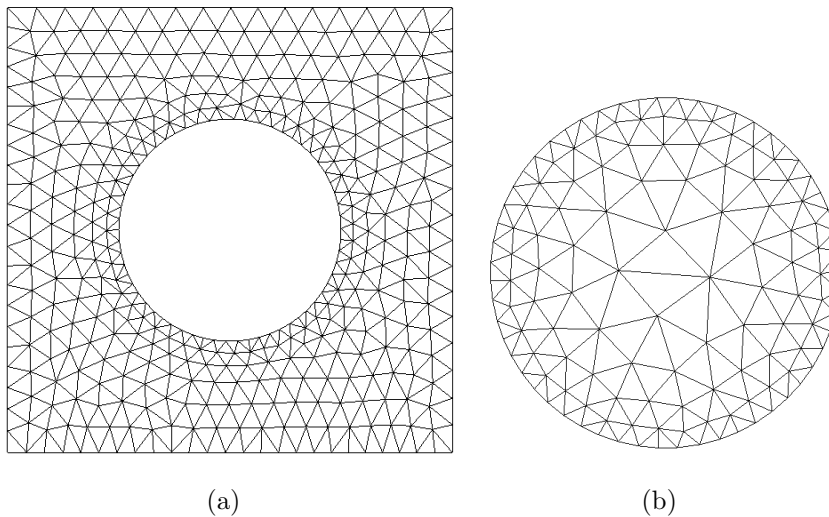


Figure 4: Mreža sa gradacijom.

Uvođenjem unutarnje funkcije udaljenost dI_{Ω} i pritom primjenjujući pripadajući algoritam, omogućava se usklađenost domene sa unutarnjim granicama.

Sada je moguće automatsko generiranje dvodimenzijske jednične ćelije sa 6 vlakana konstantnog polumjera i volumnim udjelom vlakana od 19%. Vlakna su posložena u dva reda i u svakom od njih su jednako razmaknuta.

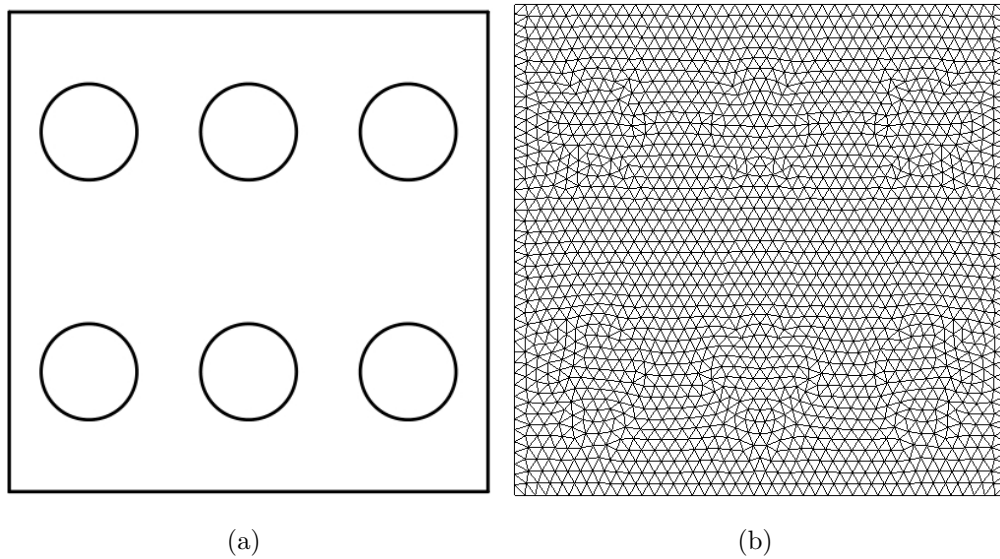


Figure 5: Geometrija (a) i diskretizacija (b) jedinice ćelije sa 6 vlakana.

Koristeći isti princip, omogućena je triangulacija 20 slučajno raspodjeljenih vlakana, različitih polumjera koji zauzimaju 15% domene.

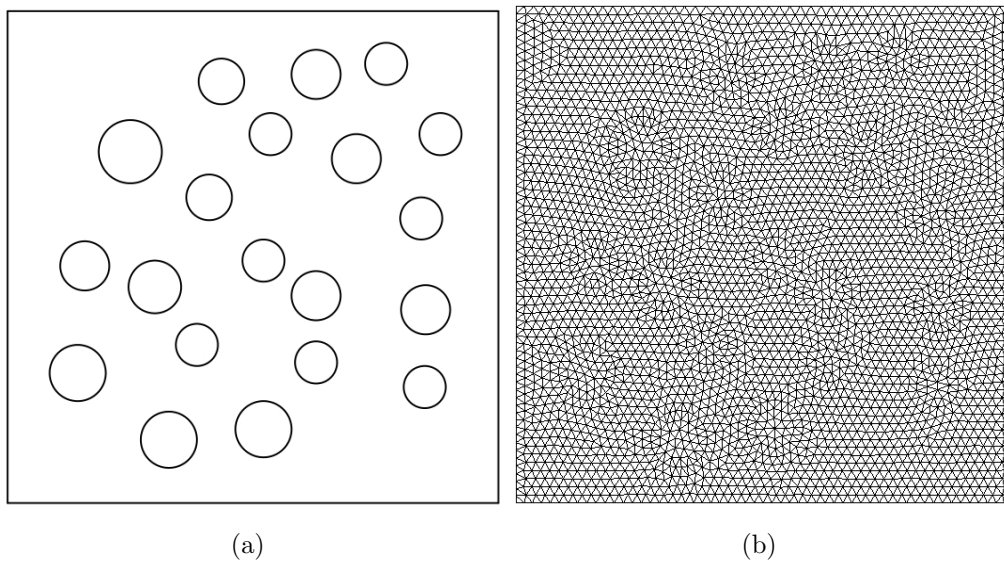
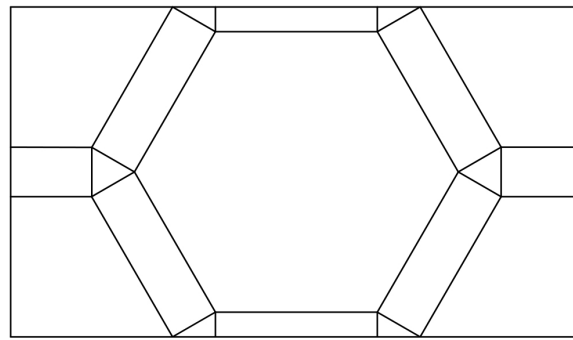
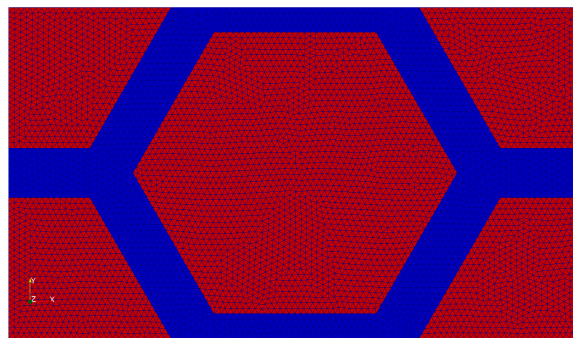


Figure 6: Geometrija (a) i diskretizacija (b) ćelije sa 20 slučajno raspoređenih vlakana.

Razvijena metoda automatskog generiranja numeričke mreže primjenjena je na problem analize naprezanja kod materijala sa mikrostrukturuom.



(a)



(b)

Figure 7: Geometrija (a) i diskretizacija (b) sintetičke mikrostrukture. Domena sa slike 7. diskretizirana je po principu "poligon po poligon". Takav princip podrazumijeva primjenu razvijenog algoritma za konveksne poligone i kontrolu kontinuiranosti raspodjele točaka na unutarnjim granicama.

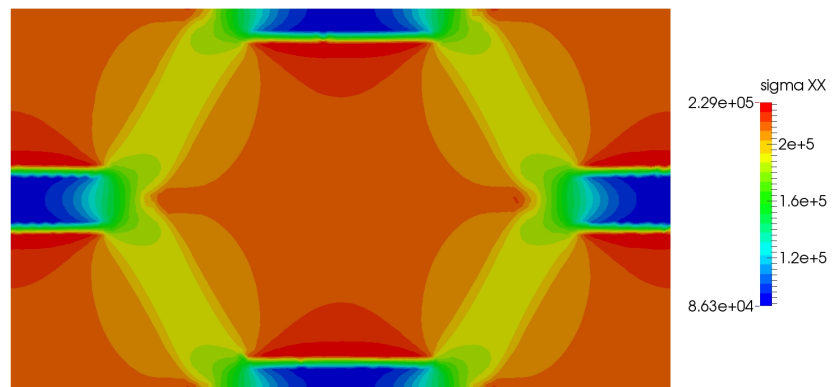


Figure 8: Raspodjela normalnog naprezanja σ_{xx} u materijalu sa mikrostrukturom.

Chapter 1

Introduction

1.1 Motivation

The mesh is the discrete structure on which the discrete representation of the governing partial differential equations is made. This discrete representation may be finite element, finite volume, or finite differences – in any case there is an underlying grid discretizing the field into a collection of finite cells defined by associated grid points (Thompson & Hamann, 1997). Within the finite volume method, the points are arranged so that they can be grouped into a set of volumes and the partial differential equations can be solved by equating various flux terms through the faces of the finite volumes (Shaw, 1992). Creating the computational mesh is very time consuming task. As reported by Versteeg and Malalasekera (Versteeg & Malalasekera, 2007) over 50% of the time spent in industry on a computational fluid dynamics project is devoted to the definition of the domain geometry and grid generation. If a domain is consisted of arbitrary shaped internal structures, an effort to discretize a domain is even more pronounced.

Therefore, the aim of this thesis is to develop an automatic mesh generator for discretization of domains with an explicitly defined two-phases (material and its internal structure). The main directions in the field of automatic mesh generation are given in the next Section.

1.2 Automatic Mesh Generation

As outlined by (Zienkiewicz *et al.*, 2005) the attempt to create a fully automatic mesh generator started from the early 1970s with the work of Zienkiewicz and Phillips (Zienkiewicz & Phillips, 1971). According to George (George, 1996) a method is said automatic if it requires no intervention from the user who only needs to provide the necessary data. A large number of automatic unstructured mesh generation algorithms have been proposed

in the literature, but the most widely used algorithms are based on one or some kind of combination of the three fundamentally distinctive methods, which are (Zienkiewicz *et al.*, 2005)

- the Delaunay triangulation method,
- the advancing front method and
- tree methods (the finite quadtree method in two dimensions and the finite octree method in three dimensions).

These approaches are capable of automatically generating tetrahedral meshes for arbitrary domains. In these methods crucial steps are the local connectivity modifications (Roca Navarro, 2009). By observing the fact that a quadrilateral can be formed by two triangles which share a common edge, the above-mentioned methods can be extended to automatically generate unstructured quadrilateral meshes in two dimensions.

A brief information on these approaches is given in the remainder of the Section.

1.2.1 The Delaunay triangulation method

The Delaunay triangulation is a triangulation for which no circumcircle of a triangle contains points in its interior. This property guarantees that the Delaunay triangulation maximizes the minimum angle among all possible triangulations of the point set (Nguyen *et al.*, 2009).

One of the difficulties of the Delaunay approach is maintaining the integrity of the boundary (Henshaw, 1996). Care must be taken to prevent the formation of triangles whose edges cross the specified boundary. Another problem is that the Delaunay criteria is not appropriate for creating very thin triangles in a boundary layer, some other condition must be used (Henshaw, 1996).

The "empty circumcircle" geometrical criterion provides a mechanism for connecting points. The task of point generation must be considered independently. Hence, grid generation by Delaunay triangulation involves two distinct problems of point connection and point creation (Thompson *et al.*, 1999).

1.2.2 The Advancing Front Method

The advancing front method is a grid generation technique based on the simultaneous point generation and connection (Thompson *et al.*, 1999). The advancing front method starts from the boundaries and progressively adds triangles. The triangulated region grows into the interior, forming a propagating front (Henshaw, 1996). Since the procedure begins at the boundary, the triangles near the boundary can be constructed to be of high quality, this is an especially important feature for many PDEs (Henshaw, 1996).

1.2.3 Tree Methods – Quadtree (2D), Octree (3D)

The quadtree approach proceeds by dividing the region into four rectangles and then recursively subdividing some of those rectangles into four additional rectangles (Henshaw, 1996). The cell size is reduced to meet certain criteria and so that the boundary is represented to sufficient resolution. The cells intersecting the boundary are replaced by polygons that follow the boundary (Henshaw, 1996). Octree is the three dimensional analog of quadtree.

1.3 Outline of the Thesis

Among the algorithms presented in the last Section, here implemented method is closest to Delaunay triangulation algorithms. It is about DISTMESH algorithm proposed by Persson and Strang (Persson & Strang, 2004) which is described within the following Chapter 2. In the paper of Nguyen and collaborators (Nguyen *et al.*, 2009) DISTMESH is compared with MESHGEN and TRIANGLE. The last two are well established adaptive triangular mesh generators known for the high quality of the generated meshes (Fan *et al.*, 2011). The evaluation was done by considering the 12 quantitative quality measures and DISTMESH algorithm has shown better performance. This algorithm utilizes an implicit description of the domain through signed distance function. This is in contrary to the usual explicit form. A common explicit form for describing geometries are Non-Uniform Rational B-Splines (NURBS), which have become the standard in most popular Computer Aided Design (CAD) and Computer Aided Engineering (CAE) systems (Cui, 2013). Furthermore, within Chapter 2, some extensions of the base algorithm are proposed and explained. Chapter 3 presents generated meshes and shows some properties of the algorithm. The algorithms introduced in Chapter 2 are utilized and their reliability is shown. At the end, Chapter 4 summarizes the achievements of the work and offers recommendations regarding future developments.

Chapter 2

An Automatic Mesh Generator

2.1 Introduction

A mesh generator which automatically generates two-dimensional triangular meshes is described in this Chapter. The mesh generator is written in C++ and implemented within OpenFOAM®, an open-source C++ library for Computational Continuum Mechanics. The base algorithm is given in Section 2.2 and its contents are explained. Furthermore, an extension of the base MATLAB® code was necessary to C++ implementation and is discussed in Section 2.3. Since the application of this mesh generator is to discretize the domains of discontinuous material, handling of internal boundaries is introduced and described in Section 2.4. Important facts related to this Chapter are given in the last, Section 2.5.

2.2 The Algorithm

A mechanical analogy based mesh generation algorithm proposed by (Persson & Strang, 2004) is used as a base for the automatic mesh generator presented here. In order to discretize a domain Ω , the algorithm requires the user the definition of:

- A signed distance function d_Ω for the domain definition
- A mesh size function h_Ω to specify the relative mesh resolution

As mentioned above, the computational domain $\Omega \subseteq \mathbb{R}^2$ is defined by the signed distance function $d_\Omega : \mathbb{R}^2 \rightarrow \mathbb{R}$ where for an arbitrary point \mathbf{p}_i we have:

$$d_\Omega(\mathbf{p}_i) = \begin{cases} \min\{d(\mathbf{p}_i, \partial\Omega)\} & \text{if } \mathbf{p}_i \notin \Omega \\ 0 & \text{if } \mathbf{p}_i \in \partial\Omega \\ -\min\{d(\mathbf{p}_i, \partial\Omega)\} & \text{if } \mathbf{p}_i \in \Omega \end{cases} \quad (2.1)$$

where $\partial\Omega$ denotes domain's boundary. This means that the signed distance function gives the distance from any point in defined bounding box to the nearest boundary of the domain. The sign is negative for points inside the domain and positive for points outside the domain. Zero level set ($d_\Omega(\mathbf{p}_i) = 0$) defines the boundary of the domain. The size function $h_\Omega(\mathbf{p})$ specifies the mesh element sizes through the domain. For $h_\Omega(\mathbf{p}) = 1$ an uniform mesh is defined. Algorithm 1 shows the steps of the applied algorithm.

Algorithm 1 DISTMESH algorithm by Persson and Strang

Input: Desired mesh element size h_0 , bounding box $[x_1, y_1] \times [x_2, y_2] \subsetneq \Omega$, distance function d_Ω , size function h_Ω .

- 1: An initial set of points $p^n = p(0)$ is distributed over the defined bounding box region and a minimum spacing h_0 between them is ensured. Remove all points with $d_\Omega > 0$ as well as the points with a probability proportional to $1/h_\Omega^2$;
- 2: Triangulate p^n by means of Delaunay triangulation algorithm and compute all edge lengths l_e ;
- 3: Determine $\mathbf{F}_{int}(p^n)$ from ordinary linear springs model as the sum of forces applied by its neighbouring points;
- 4: Move the points by using the forward Euler method $p^{n+1} = p^n + \Delta t \mathbf{F}_{int}(p^n)$;
- 5: Find the points which now are outside the domain ($d_\Omega > 0$) and project them on the domain boundary applying first order approximate projection;
- 6: Break if every point has moved less than a predefined tolerance. Otherwise, set $n \mapsto n + 1$ and repeat from step 2.

Output: point array \mathbf{p} and triangles list \mathbf{m} .

Firstly (line 1), N initial points are distributed within the bounding box region and stored in an N -by-2 array \mathbf{p}

$$\mathbf{p} = [\mathbf{x} \ \mathbf{y}] \tag{2.2}$$

A distance between the points is h_0 in x - and $\sqrt{3}/2$ in y - direction. A minimum distance h_0 between the points is created by displacing every even row for $h_0/2$ in x -direction. The signed distance function is checked for all points and the points with $d_\Omega(\mathbf{p}_i) \leq 0$ are kept. Then, the size function $h_\Omega(\mathbf{p}_i)$ is evaluated for the retained points and the points with a probability proportional to $1/h_\Omega(\mathbf{p}_i)^2$ are discarded. This final set of points is now passed to a Delaunay triangulation routine (line 2) which connects those points. The Delaunay triangulation is a triangulation of a point set \mathbf{p} where the circumcircle of each triangle passes only through its points as shown on Figure 2.1.

Also, the Delaunay triangulation maximizes the smallest angle over all triangulation which leads to elements of high quality. Within the triangulation process, the centroids

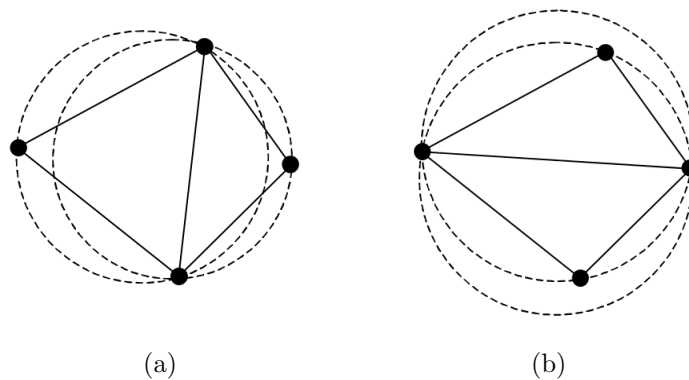


Figure 2.1: Satisfied (a) and not satisfied (b) Delaunay criterion (Owen, 1998).

of all elements are examined. If the element's centroid lies outside the geometry ($d_\Omega > 0$), the element is deleted.

This formation is now considered to be a fictive mechanical structure. The element's points corresponds to joints and the edges correspond to elastic springs. Forcing this mechanical structure to a static equilibrium gives the final positions of the points. In every mesh point \mathbf{p}_i the force field $\mathbf{F}(\mathbf{p}_i)$ is defined as a sum of all forces acting on that point (line 3):

$$\mathbf{F}(\mathbf{p}_i) = \mathbf{F}_{int}(\mathbf{p}_i) + \mathbf{F}_{ext}(\mathbf{p}_i) \quad (2.3)$$

where \mathbf{F}_{int} and \mathbf{F}_{ext} denote internal and external (reactions at boundaries) forces respectively. The internal forces acting on each point refer to the repulsive forces from all the edges connected to that point as depicted in the Figure 2.2 (a). The external forces are existing only for boundary points which go outside the domain (Figure 2.2 (b)). Since those points are going outside of the domain due to their moving induced by the internal forces, the external forces are discussed later in this Section (line 5). The total force acting at fixed points is equal to zero.

The total internal force \mathbf{F}_{int} acting on point \mathbf{p}_i reads

$$\mathbf{F}_{int}(\mathbf{p}_i) = \sum_e f(l_e, l_0) \quad (2.4)$$

where e denotes all the edges connected with point \mathbf{p}_i , l_0 is the desired length of an element's edge and l_e is the length of the edge connecting the neighbouring nodes \mathbf{p}_i and \mathbf{p}_j which reads

$$l_e = \|\mathbf{p}_i - \mathbf{p}_j\| \quad (2.5)$$

The term under summation operator on the right hand side of equation (2.4) refers to a linear repulsive force attached at edge endpoints

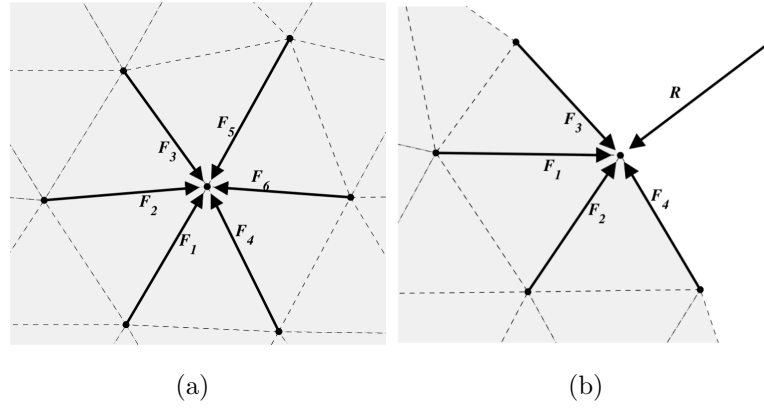


Figure 2.2: Forces: (a) internal forces and (b) external forces (Persson, 2006).

$$f(l_e, l_0) = f^{ij} = \begin{cases} (l_0 - l_e)n_{i,j} & \text{if } l_e < l_0 \\ 0 & \text{if } l_e \geq l_0 \end{cases} \quad (2.6)$$

where $n_{i,j}$ is the normal vector

$$n_{i,j} = \frac{\mathbf{p}_i - \mathbf{p}_j}{l_e} \quad (2.7)$$

The force f^{ij} acting on point \mathbf{p}_i is equal and of opposite sign to one acting on the neighbouring point \mathbf{p}_j (Figure 2.3)

$$f^{ij} = -f^{ji} \quad (2.8)$$

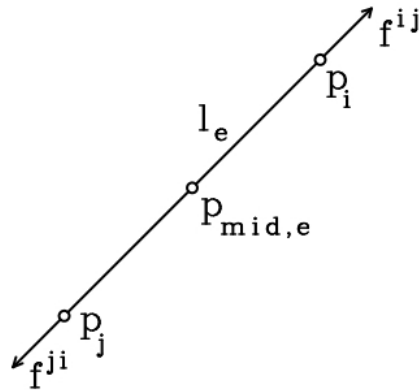


Figure 2.3: Repulsive forces acting on the edge endpoints.

The emphasis on the repulsive forces enables the distribution of points across the whole domain. This is also supported by choosing the value l_0 to be slightly larger than the desired length defined by the size function h_Ω

$$l_0 = h_\Omega(\mathbf{p}_{mid,e})F_{scale} \sqrt{\frac{\sum_e l_e^2}{\sum_e h_\Omega(\mathbf{p}_{mid,e})^2}} \quad (2.9)$$

where $\mathbf{p}_{mid,e}$ denotes an edge midpoint vector, and F_{scale} is a fixed factor. Persson and Strang (Persson & Strang, 2004) recommend the value of 1.2 for F_{scale} .

As outlined earlier, the locations of mesh points \mathbf{p} are computed by forcing a truss structure to its static equilibrium which reads

$$\mathbf{F}(\mathbf{p}) = 0 \quad (2.10)$$

To this end, a false transient is introduced as follows

$$\frac{d\mathbf{p}}{dt} = \mathbf{F}_{int}(\mathbf{p}), \quad t \geq 0 \quad (2.11)$$

The steady state solution of (2.11) satisfies (2.10) and provides well distributed points. The ODE (2.11) is solved using the Euler explicit algorithm (line 4)

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \Delta t \mathbf{F}_{int}(\mathbf{p}^n) \quad (2.12)$$

where \mathbf{p}^{n+1} are new and \mathbf{p}^n are known points positions. Pseudo time step is Δt and $\mathbf{F}_{int}(\mathbf{p}^n)$ is the total internal force.

Due to this update of points positions, some points can settle outside the domain Ω . Those points are than projected to the closest domain's boundary. The projection of an arbitrary point \mathbf{p}_i which crossed the boundary is defined as

$$\mathbf{p}_i = \mathbf{p}_i - \frac{d_\Omega(\mathbf{p}_i)}{|\nabla d_\Omega|^2} \nabla d_\Omega \quad (2.13)$$

and is repeated until $d_\Omega(\mathbf{p}_i) = 0$ is achieved.

The steps 2 – 5 of Algorithm 1 are repeated until every point has moved less than a predefined tolerance, i.e. the termination criterion is met.

2.3 The Level-set Function of a Convex Polygon

The base algorithm presented in former Section requires one user-defined function d_Ω to define the domain Ω . For many simple geometries, such as circles and ellipses the signed distance function can be expressed as a single function. On the other hand, in many real applications domains are convex polygons. Since the convex polygon cannot be expressed as a single function, an algorithm of quadratic time complexity $O(n^2)$ is proposed for computing the distance to a polygon given by its $N_{\mathbf{p}_{fix}}$ counter-clockwise defined fixed

points. Algorithm 2 presents proposed method to compute the signed distance function for a polygonal domain.

Algorithm 2 Signed distance function of a polygon

Input: point \mathbf{p}

```

1: Initialize  $\mathbf{n}, \mathbf{r}_p, \mathbf{e}_{\mathbf{r}_p}, \mathbf{d}_0, \mathbf{d}_N, d_N, L_e$ 
2: for  $c \leftarrow 0$  to  $N_{\mathbf{p}_{fix}} - 2$  do
3:   Compute  $\mathbf{n}, \mathbf{r}_p, \mathbf{e}_{\mathbf{r}_p}, \mathbf{d}_0, \mathbf{d}_N, d_N, L_e$  for all polygon's edges except the last one
4: end for
5: Compute  $\mathbf{n}, \mathbf{r}_p, \mathbf{e}_{\mathbf{r}_p}, \mathbf{d}_0, \mathbf{d}_N, d_N, L_e$  for the last polygon's edge
6: Initialize  $\mathbf{t}, \mathbf{L}_t, t, d_{out}(N_{\mathbf{P}_{fix}}), c_0, d_{outMin}, d_{dist}, d_{distMin}$ 
7: Check if the point is inside the polygon
8: if  $c_0 = N_{\mathbf{p}_{fix}}$  then
9:   for all  $d_N$  do
10:    if  $\|d_N(c_1)\| < \|d_{distMin}\|$  then
11:       $d_{dist} \leftarrow d_{distMin}$ 
12:    end if
13:  end for
14: end if
15: for all  $d_N$  do
16:   if  $d_N(c_2) > 0$  then
17:    Compute  $\mathbf{t}, \mathbf{L}_t, t$ 
18:    if  $t \geq 0$  and  $t \leq L_e(c_2)$  then
19:      Point is inside the edge zone – store  $d_{dist}$ 
20:      break
21:    else
22:      Point is outside the edge zone – store  $d_{dist}$ 
23:      break
24:    end if
25:   end if
26: end for
27:  $d_\Omega \leftarrow d_{dist}$ 

```

Output: signed distance d_Ω

Every point is passed once in Algorithm 2 and its distance to the nearest domain's boundary is returned. To determine the distance to the nearest boundary of the domain, for all domain's edges except the last one the following quantities are computed (lines 2–4, Figure 2.4):

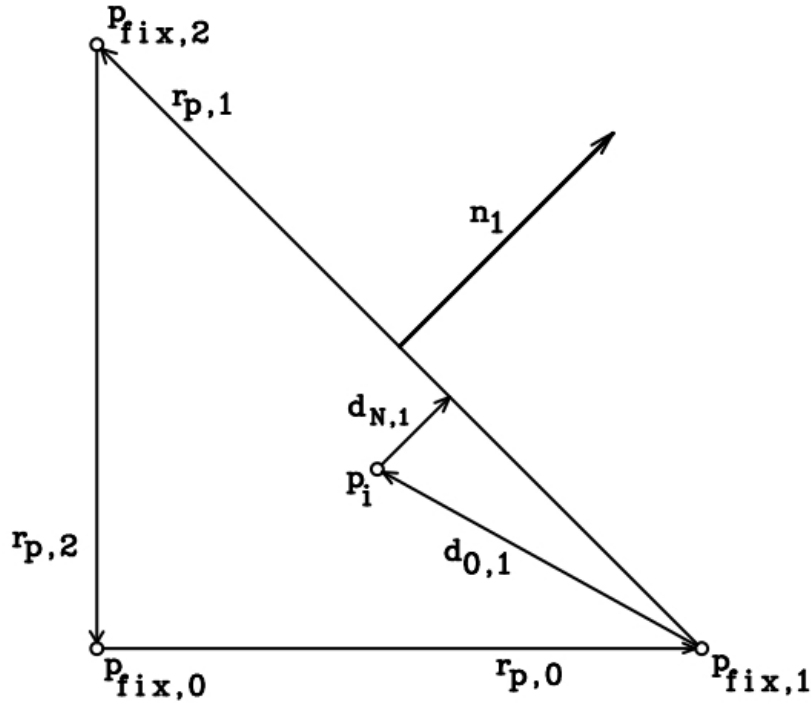


Figure 2.4: Computing the signed distance to a convex polygon.

- edge vector \mathbf{r}_p (c is the current value of the counter)

$$\mathbf{r}_p(c) = \mathbf{p}_{fix}(c+1) - \mathbf{p}_{fix}(c) \quad (2.14)$$

- edge length L_e

$$L_e(c) = \|\mathbf{r}_p(c)\| \quad (2.15)$$

- edge unit vector \mathbf{e}_{r_p}

$$\mathbf{e}_{r_p}(c) = \frac{\mathbf{r}_p(c)}{L_e(c)} \quad (2.16)$$

- edge normal \mathbf{n} ($\hat{\mathbf{k}}$ is the unit vector pointing in the positive z direction)

$$\mathbf{n}(c) = \mathbf{e}_{r_p}(c) \times \hat{\mathbf{k}} \quad (2.17)$$

- distance vector \mathbf{d}_0 from an arbitrary point \mathbf{p}_i to the edge's rightmost fixed point. The rightmost is defined with respect to outward pointing normal as shown on the Figure 2.4

$$\mathbf{d}_0(c) = \mathbf{p}_i - \mathbf{p}_{fix}(c) \quad (2.18)$$

- point to boundary distance vector \mathbf{d}_N

$$\mathbf{d}_N(c) = (\mathbf{d}_0(c) \cdot \mathbf{n}(c))\mathbf{n}(c) \quad (2.19)$$

- distance value d_N

$$d_N(c) = \mathbf{n}(c) \cdot \mathbf{d}_N(c) \quad (2.20)$$

The same procedure is applied on the last edge (line 5), only on the right hand side of Equation (2.14) instead of next fixed point is the first fixed point which yields

$$\mathbf{r}_p(c) = \mathbf{p}_{fix}(0) - \mathbf{p}_{fix}(c) \quad (2.21)$$

The distance between the point \mathbf{p}_i and all the polygon edges is calculated and stored in $N_{\mathbf{p}_{fix}} \times 1$ array d_N . If the point is inside the polygon, the distances to all edges are negative as one can see on the Figure 2.5

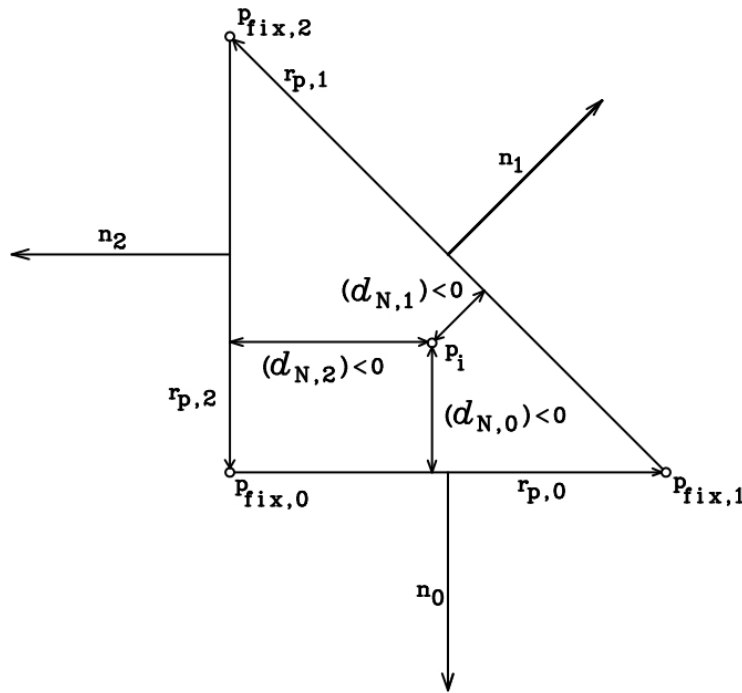


Figure 2.5: Only the point which has all distances negative is inside the polygon.

According to Equation (2.1) the smallest of those distances is determined (lines 8 – 14), and passed to Algorithm 1.

Otherwise, if a point has any positive distance, the point is outside the domain (line 16). Outside the domain, one can recognize two zones. One is an *edge zone* characterized by isolines of constant distance, and the other is a *point zone* where the isolines are circles with common center in the nearest fixed point. Figure 2.6 shows the point \mathbf{p}_i which lies inside the edge zone. The algorithm recognizes whether the point is inside or outside the edge zone by computing an indicator value t (line 17)

$$t = \mathbf{e}_{r_p}(c) \cdot \mathbf{L}_t \quad (2.22)$$

where \mathbf{L}_t denotes the distance vector from an intersection point \mathbf{t} to the rightmost edge's fixed point defined the same as in Equation (2.18). This vector reads

$$\mathbf{L}_t = \mathbf{t} - \mathbf{p}_{fix}(c) \quad (2.23)$$

where \mathbf{t} is the intersection point's vector defined as

$$\mathbf{t} = \mathbf{p}_i - \mathbf{d}_N(c) \quad (2.24)$$

If the indicator value t determined by Equation (2.22) is a value in range $[0,1]$ the point is inside the edge zone and distance $d_N(c)$ is returned (lines 18 – 20).

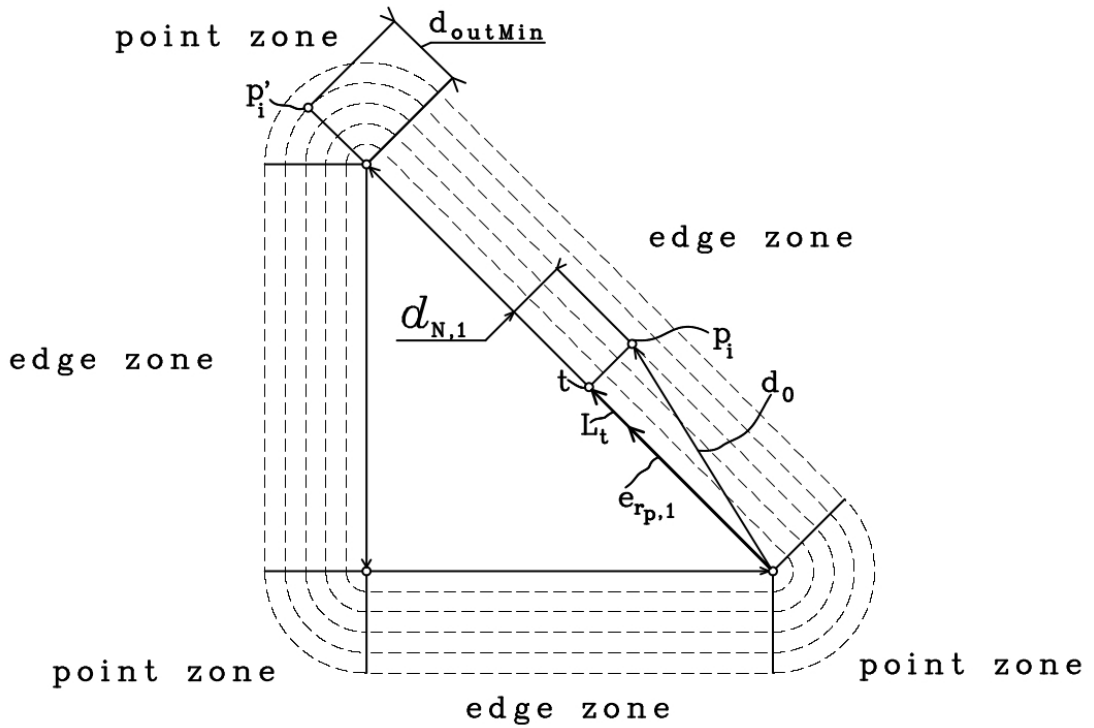


Figure 2.6: Points outside the polygon.

Let the point \mathbf{p}'_i lies in the point zone. Then, the distance is calculated as follows

$$d_{outMin} = \|\mathbf{p}'_i - \mathbf{p}_{fix}(c)\| \quad (2.25)$$

The distance to the nearest fixed point is determined (lines 21 – 23) and returned to mesh generation procedure given in Algorithm 1.

2.4 Internal Boundaries

In order to handle an internal boundary, one should extend Algorithm 1. Here it is done as proposed by Persson (Persson, 2005). The procedure presented as Algorithm 3 is added between the lines 4 and 5 in Algorithm 1. Therefore, an internal distance function dI_Ω is introduced. The internal distance function is checked for all the element endpoints in the domain (line 2 – 4). If for a two edge endpoints, dI_Ω has an opposite sign (line 5 or line 12), the edge is crossing the internal boundary. For the edge which is crossing the internal boundary, the distance magnitudes $\|dI_\Omega\|$ are computed and the lower distance magnitude $\|dI_\Omega\|$ is determined (line 6 or line 13). The edge endpoint with lower distance magnitude is then projected on internal boundary applying the same procedure as in line 5 of Algorithm 1.

Algorithm 3 Internal boundaries generation on an existing mesh

Input: Updated point set \mathbf{p}^{n+1} not aligned to internal boundaries

```

1: Initialize  $dI_0(N_{edges}), dI_1(N_{edges})$ 
2: for all edges do
3:    $dI_0(c) \leftarrow dI_\Omega(\mathbf{p}^{n+1}[edge[c][0]])$ 
4:    $dI_1(c) \leftarrow dI_\Omega(\mathbf{p}^{n+1}[edge[c][1]])$ 
5:   if  $dI_0(c) > 0$  and  $dI_1(c) < 0$  then
6:      $D_{min} \leftarrow \min(\|dI_0(c)\|, \|dI_1(c)\|)$ 
7:     if  $(\|dI_1(c)\| - D_{min}) < 1E-15$  then
8:       Project the second edge point on an internal boundary
9:     else
10:      Project the first edge point on an internal boundary
11:    end if
12:   else if  $dI_0(c) < 0$  and  $dI_1(c) > 0$  then
13:      $D_{min} \leftarrow \min(\|dI_0(c)\|, \|dI_1(c)\|)$ 
14:     if  $(\|dI_0(c)\| - D_{min}) < 1E-15$  then
15:       Project the first edge point on an internal boundary
16:     else
17:       Project the second edge point on an internal boundary
18:     end if
19:   end if
20: end for

```

Output: Updated point set \mathbf{p}^{n+1} aligned to internal boundaries

Figure 2.7 shows an edge which is crossing the internal boundary. Applied notation corresponds to one used in Algorithm 3.

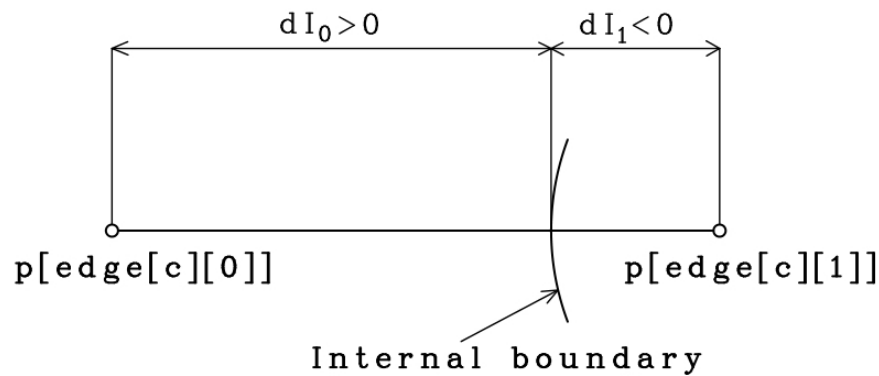


Figure 2.7: An edge which crosses the internal boundary. Right edge endpoint is closer to internal boundary and is thus projected.

2.5 Conclusion

Within this Chapter, the base algorithm is given and its main features are described. The algorithm spreads the points into a defined bounding box, utilizes the signed distance function to determine ones within domain and keep them. Those points are connected (triangulated) by Delaunay algorithm. Now those element's edges are considered to be elastic springs which are allowed only to produce repulsive forces (internal forces) and move the points. In such manner, the points fill the desired geometry. After the internal forces move the points, some boundary points will probably settle outside the domain. Those points are then projected (external forces) by using the distance function. The interplay between force-equilibrium approach and Delaunay triangulation is over when the termination criterion, based on points movement, is met. Since original MATLAB® implementation has "inpolygon" function to determine whether is point inside or outside the polygonal domain, to make polygonal domains possible in C++ an algorithm based on vector calculus is proposed. Finally, an algorithm which generates internal boundaries is added to base algorithm in order to handle discontinuous domains.

Chapter 3

Examples

3.1 Introduction

In the automatic mesh generator developed here, the geometry is defined in an implicit form. Such approach enables representing a domain by a continuous function or set of fixed points on domain's boundaries. Here, the both modes are presented as well as some additional features and extensions of the base algorithm. At the end, an application in meshing of material with discontinuities is given.

3.2 Discussion

An unit circle is meshed with triangular elements as shown on Figure 3.1.

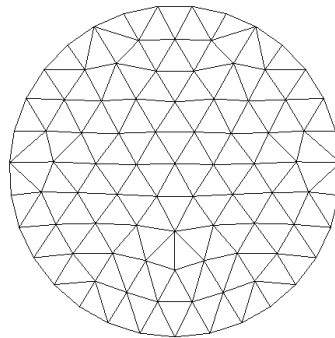


Figure 3.1: Discretization of the unit circle.

This domain is defined by an implicit function

$$d_{\Omega}(\mathbf{p}) = \|\mathbf{p}\| - 1 \quad (3.1)$$

Decreasing the initial spacing h_0 results in finer mesh (Figure 3.2).

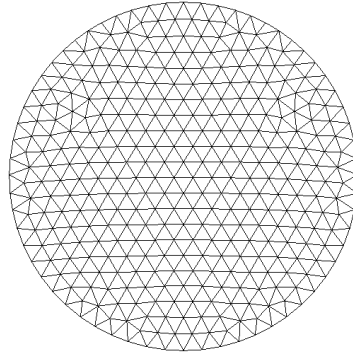


Figure 3.2: Refined discretization of the unit circle.

The parameters of both meshes are summarized in Table 3.1.

Table 3.1: Parameters of the unit circle meshes

Figure	$d_{\Omega}(\mathbf{p})$	$h_{\Omega}(\mathbf{p})$	h_0	BB	n_{it}	n_r	t_{ex} [s]
3.1	$\ \mathbf{p}\ - 1$	1	0,2	$[-2 -2] \times [2 2]$	126	28	0,8
3.2	$\ \mathbf{p}\ - 1$	1	0,1	$[-2 -2] \times [2 2]$	324	44	8,62

where BB denotes bounding box, n_{it} is number of iterations, while n_r and t_{ex} are number of retriangulations and execution time respectively. Also, an ellipse can be defined by a single function

$$d_{\Omega}(\mathbf{p}) = (\mathbf{p}_x^2/4 + \mathbf{p}_y^2/1)^{1/2} - 1 \quad (3.2)$$

and its triangular mesh is shown on Figure 3.3 while the characteristics are in Table 3.2.

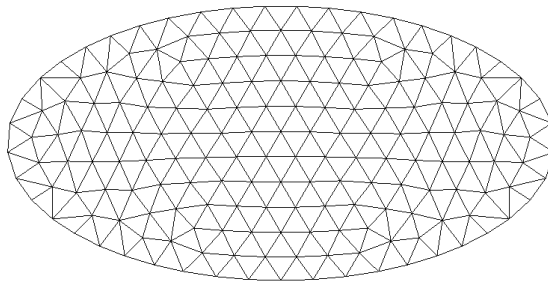


Figure 3.3: Discretized ellipse.

Furthermore, the algorithm allows domain definition via set operations such as *union*, *difference* and *intersection*. The union is defined as follows

$$d_{\Omega_1 \cup \Omega_2}(\mathbf{p}) = \min(d_{\Omega_1}(\mathbf{p}), d_{\Omega_2}(\mathbf{p})) \quad (3.3)$$

and is depicted on Figure 3.4.

Table 3.2: Parameters of the ellipse mesh

Figure	$d_{\Omega}(\mathbf{p})$	$h_{\Omega}(\mathbf{p})$	h_0	BB	n_{it}	n_r	t_{ex} [s]
3.3	Equation (3.2)	1	0,2	[-3 -2] x [3 2]	137	19	1,85

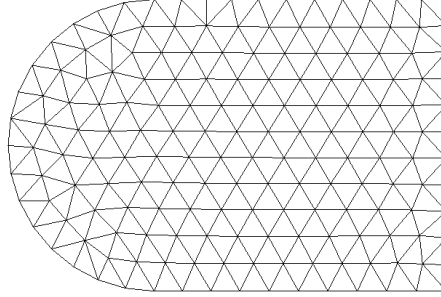


Figure 3.4: Domain defined using union scheme.

The polygonal part of the domain is described through its counter-clockwise defined fixed points. More on polygonal domains is given later in this Section. The level-set function

$$d_{\Omega_2} = ((\mathbf{p}_x - 1, 5)^2 + (\mathbf{p}_y - 1, 5)^2)^{1/2} - 0,5 \quad (3.4)$$

determines the circular part of the domain.

The difference reads

$$d_{\Omega_1 \setminus \Omega_2}(\mathbf{p}) = \max(d_{\Omega_1}(\mathbf{p}), -d_{\Omega_2}(\mathbf{p})) \quad (3.5)$$

and enables meshing surfaces with holes as shown on Figure 3.5.

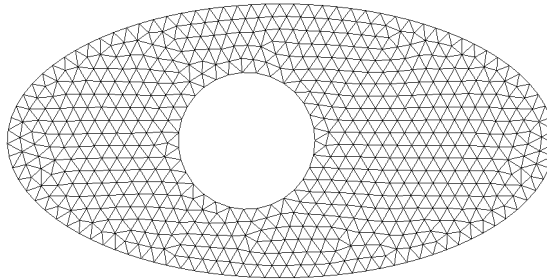


Figure 3.5: Discretized surface with hole.

Where the ellipse reads

$$d_{\Omega_1} = (\mathbf{p}_x^2/16 + \mathbf{p}_y^2/4)^{1/2} - 1 \quad (3.6)$$

and the subtracted circle is

$$d_{\Omega_2} = ((\mathbf{p}_x + 0, 5)^2 + \mathbf{p}_y^2)^{1/2} - 1 \quad (3.7)$$

In this case, Equation (3.4) turns the signed distance of interior circle's points to positive. In this manner inner points are deleted (line 1 of Algorithm 1).

The intersection is given by

$$d_{\Omega_1 \cap \Omega_2}(\mathbf{p}) = \max(d_{\Omega_1}(\mathbf{p}), d_{\Omega_2}(\mathbf{p})) \quad (3.8)$$

and gives

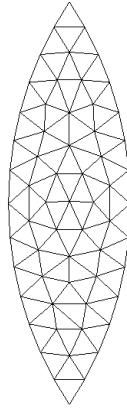


Figure 3.6: Domain created by intersecting two circles of constant radii.

where both domains are circles with constant radii

$$d_{\Omega_1} = ((\mathbf{p}_x - 3)^2 + (\mathbf{p}_y - 3)^2)^{1/2} - 1, 5 \quad (3.9)$$

$$d_{\Omega_2} = ((\mathbf{p}_x - 5, 5)^2 + (\mathbf{p}_y - 3)^2)^{1/2} - 1, 5 \quad (3.10)$$

The parameters of the meshes created by set operations are summarized in Table 3.3.

Table 3.3: Characteristics of the meshes defined by set operations

Figure	$d_{\Omega}(\mathbf{p})$	$h_{\Omega}(\mathbf{p})$	h_0	BB	n_{it}	n_r	t_{ex} [s]
3.4	d_{Ω_1} - Algorithm 2	1	0,1	[0 0] x [3,5 3]	1000	80	10,28
	d_{Ω_2} - Equation (3.4)						
3.5	d_{Ω_1} - Equation (3.6)	1	0,2	[-5 -3] x [5 3]	435	38	19,97
	d_{Ω_2} - Equation (3.7)						
3.6	d_{Ω_1} - Equation (3.9)	1	0,1	[0 0] x [8,5 6]	293	46	1,16
	d_{Ω_2} - Equation (3.10)						

When a polygonal domain is considered, the user specifies its fixed points. To determine the signed distance, Algorithm 2 utilizes vector calculus on the defined set of fixed points.

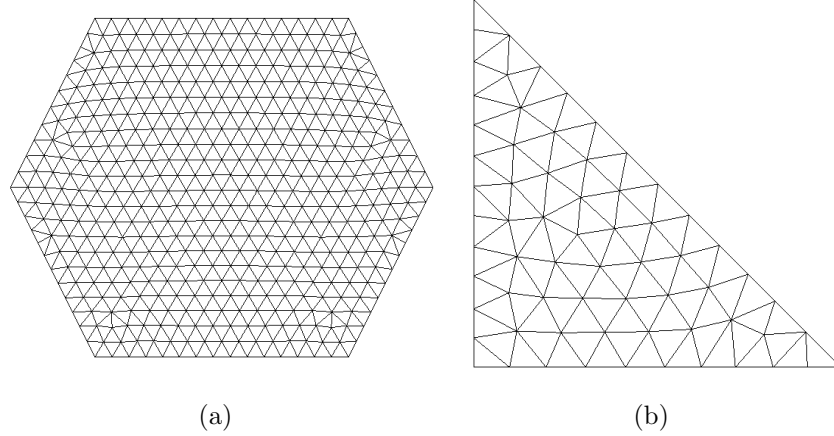


Figure 3.7: Polygonal domains.

Combining polygonal and circular domain by means of previously introduced subtraction operation (Equation 3.5), one can obtain a square plate with a circular hole (Figure 3.8) which can, for example, serve in stress analysis.

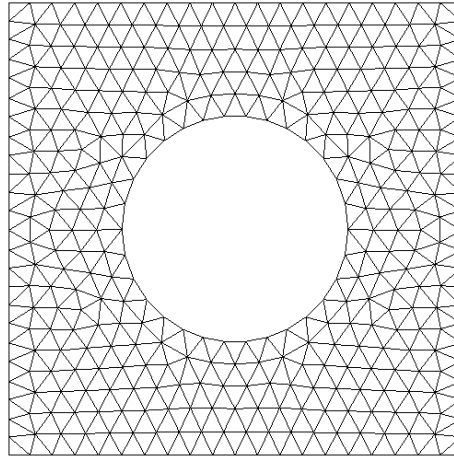


Figure 3.8: Square plate with a circular hole.

Note that all the meshes presented so far were almost uniform, i.e. $h_{\Omega}(\mathbf{p}) = 1$ is applied. Since the size function h_{Ω} describes the relative spacing between the mesh points, refinement of specific zones is thus enabled.

On Figure 3.9 (a) one can recognize a square with a hole which is refined at hole. Such refinement is achieved by introducing space-dependent size function

$$h_{\Omega}(\mathbf{p}) = \min(4\|\mathbf{p}\| - 1, 2) \quad (3.11)$$

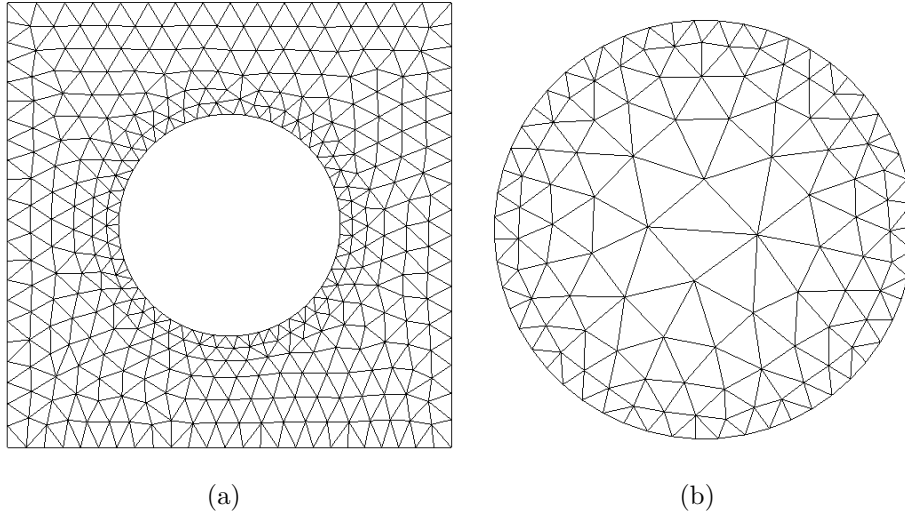


Figure 3.9: Graded mesh at specific zones.

Also, Figure 3.9 (b) represents a domain with size function defined as

$$h_{\Omega}(\mathbf{p}) = 1 - 1.5\|\mathbf{p}\| \quad (3.12)$$

Introducing the internal distance function dI_{Ω} and utilizing Algorithm 3, a mesh alignment with the internal boundaries is ensured. The basic mesh which contains internal boundaries given on Figure 3.10 (b). For easier recognizing the internal boundary, a domain without triangulation is depicted on Figure 3.10 (a).

Now one can automatically generate a mesh of a two-dimensional unit cell with 6 fibres and volume content of fibres 19%. On Figure 3.11 (a) the fibres in the unit cell are placed in two rows and equally spaced in each row. The problem discretization is given on Figure 3.11 (b). Furthermore, a triangulation of 20 randomly distributed fibres with different radii is created (Figure 3.12). The volume content of fibres is 15%.

The finite volume numerical stress analysis is carried out on the two-phase synthetic microstructure using OpenFOAM 3.1-ext®. A $3\mu\text{m} \times 1.7\mu\text{m}$ two dimensional periodic microstructure cell is considered (Figure 3.13). The domain is discretized in a polygon-by-polygon manner. Such approach utilizes Algorithm 2 and demands continuity control of the internal boundary points distribution. The Young's modulus and Poisson's ratio of both particulates and matrix are shown in Table 3.4.

Microstructure is loaded by applying fixed normal displacement ($0.001 \mu\text{m}$) at the right boundary. The symmetry boundary condition is used on the remaining boundaries. Plane strain linear elastic model is assumed and numerical solution is obtained using the finite volume solver described in (Tuković *et al.*, 2013). Figures 3.14 and 3.15 show distribution

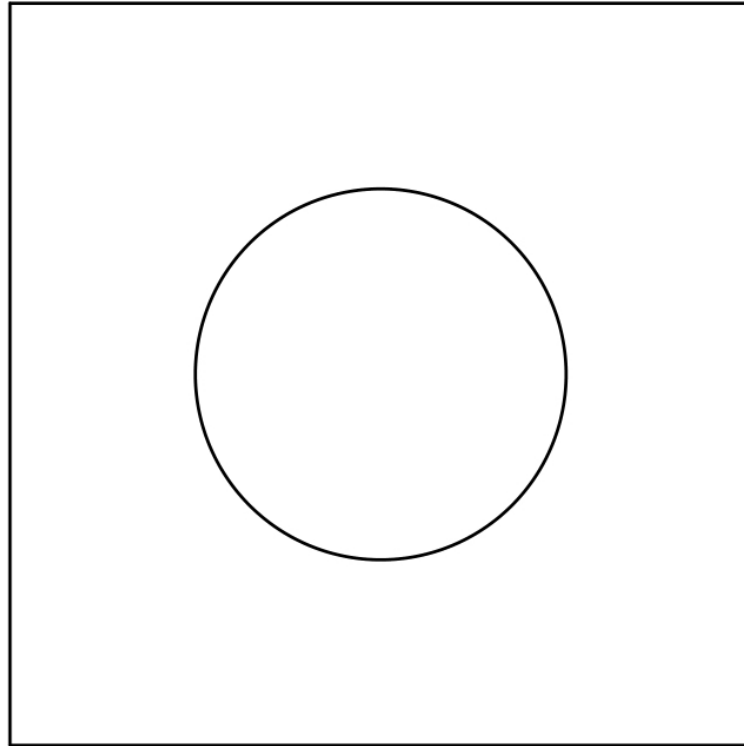
of normal stresses σ_{xx} , σ_{yy} , shear stress τ_{xy} and displacements along the microstructure domain.

Table 3.4: Material properties

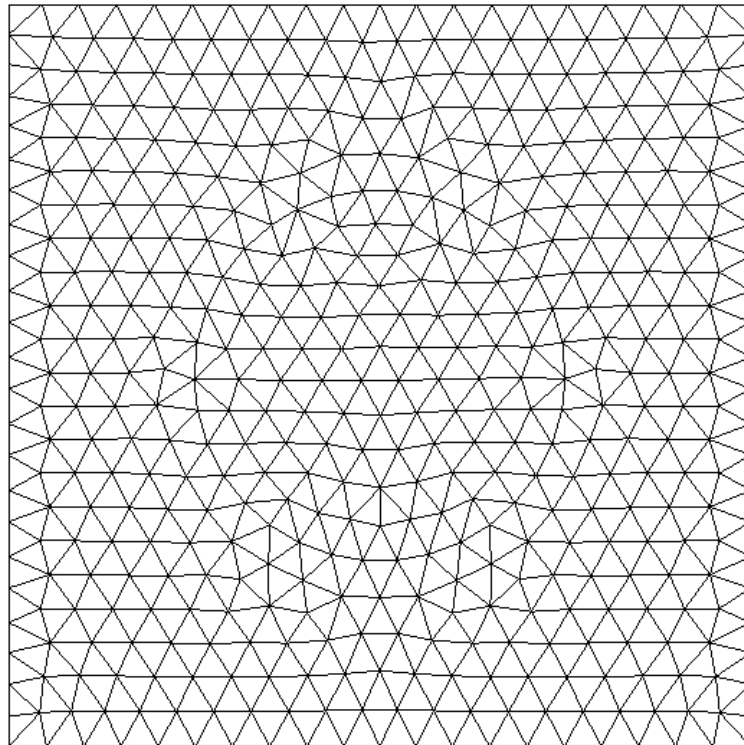
	E [GPa]	ν
Particulates (red contour)	800	0,1
Matrix (blue contour)	300	0,1

3.3 Conclusion

The automatically generated finite volume meshes are presented in this Chapter. Resulting meshes are well shaped which is due to conjunction between the force-balance analogy and Delaunay triangulation. The geometries are defined by a continuous function (circles and ellipses), fixed points on boundaries (polygons) or combination of those two (polygons with holes). An extension of the base algorithm is done with respect to internal boundaries. This leads to quality fitting the interface between the exterior and interior domain. Therefore, the algorithm is applied to discretize discontinuous materials. Finally, finite volume based stress analysis was performed on a synthetic microstructure. The synthetic microstructure is discretized with the mesh generator proposed in this thesis.

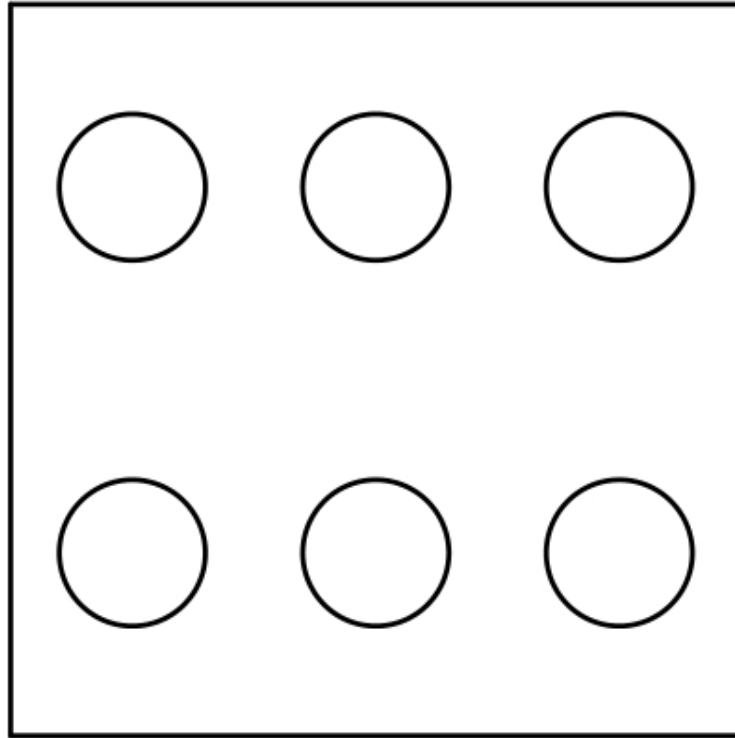


(a)

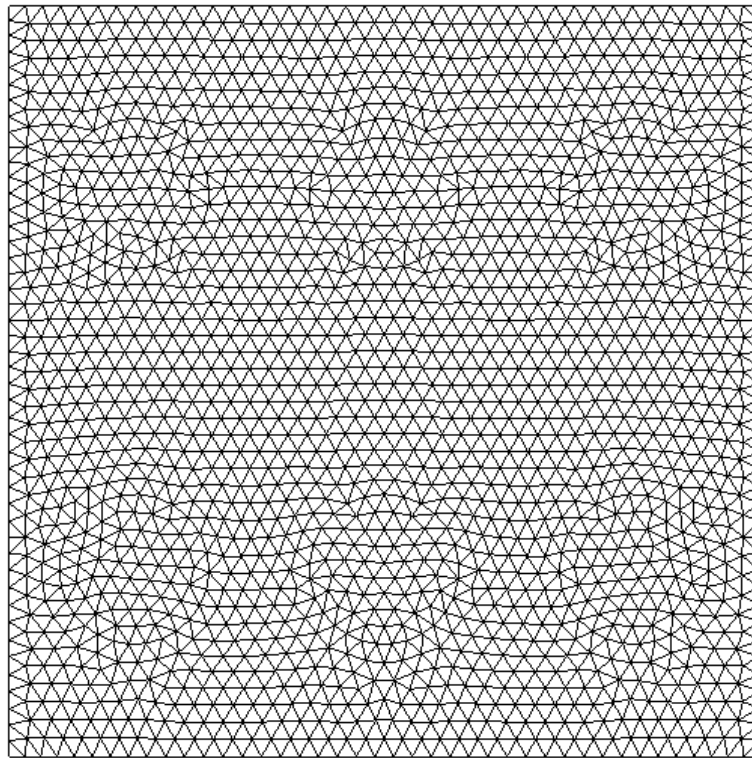


(b)

Figure 3.10: The geometry (a) and the discretization (b) of the basic case with internal interface.

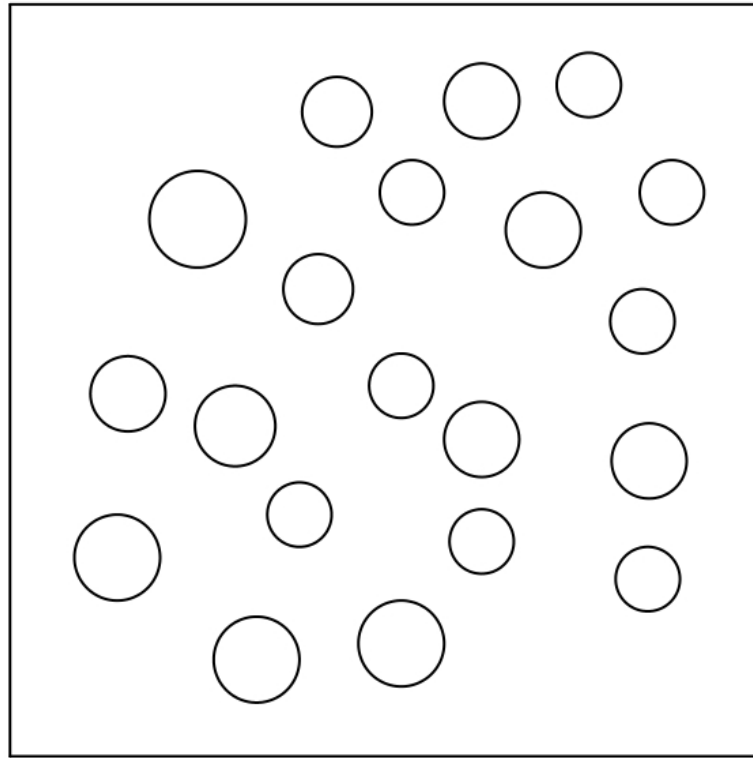


(a)

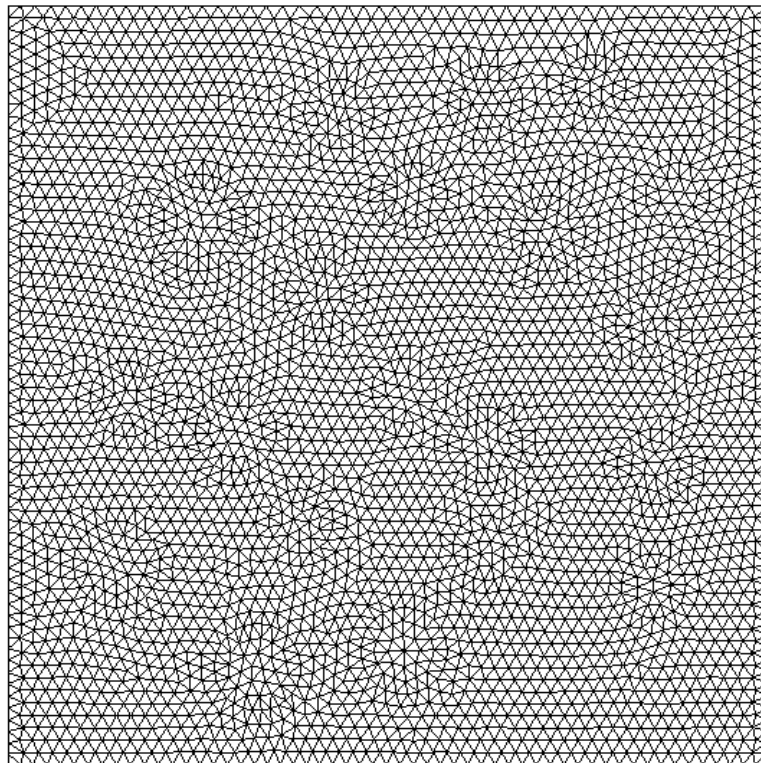


(b)

Figure 3.11: The geometry (a) and the discretization (b) of the unit square with 6 fibres.

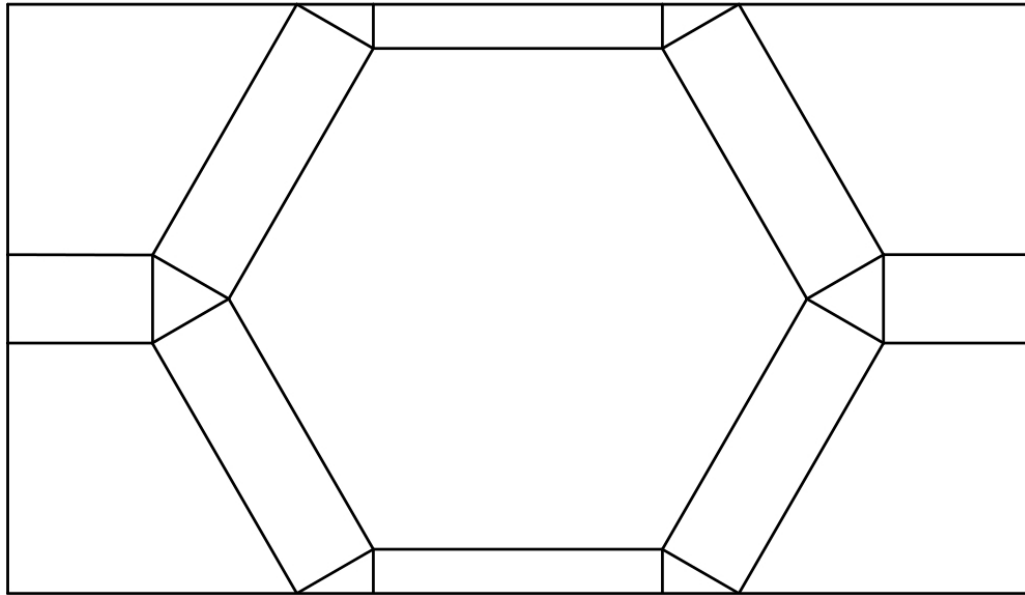


(a)

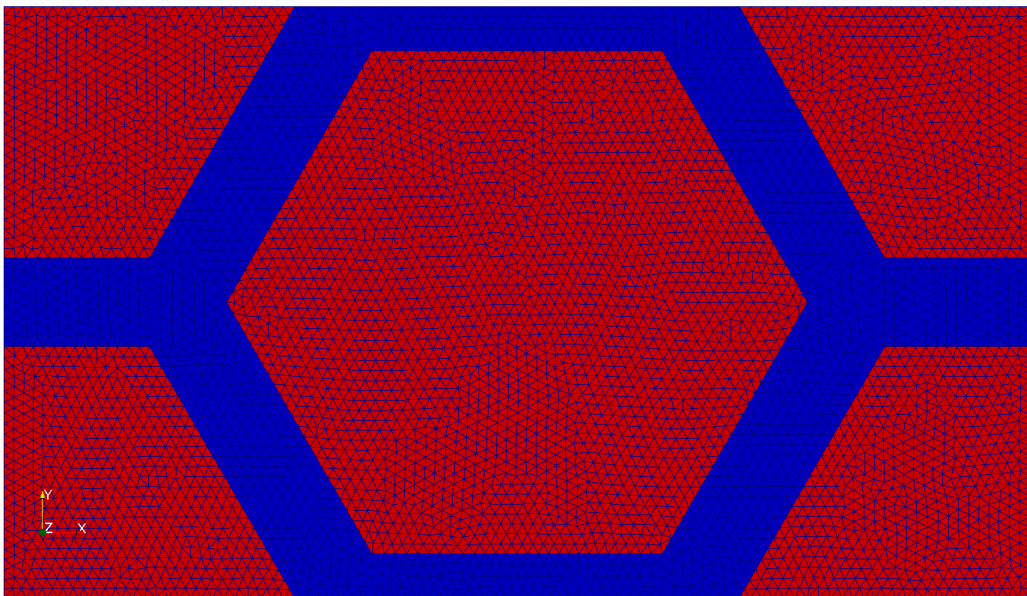


(b)

Figure 3.12: The geometry (a) and the discretization (b) of the cell with 20 randomly distributed fibres.

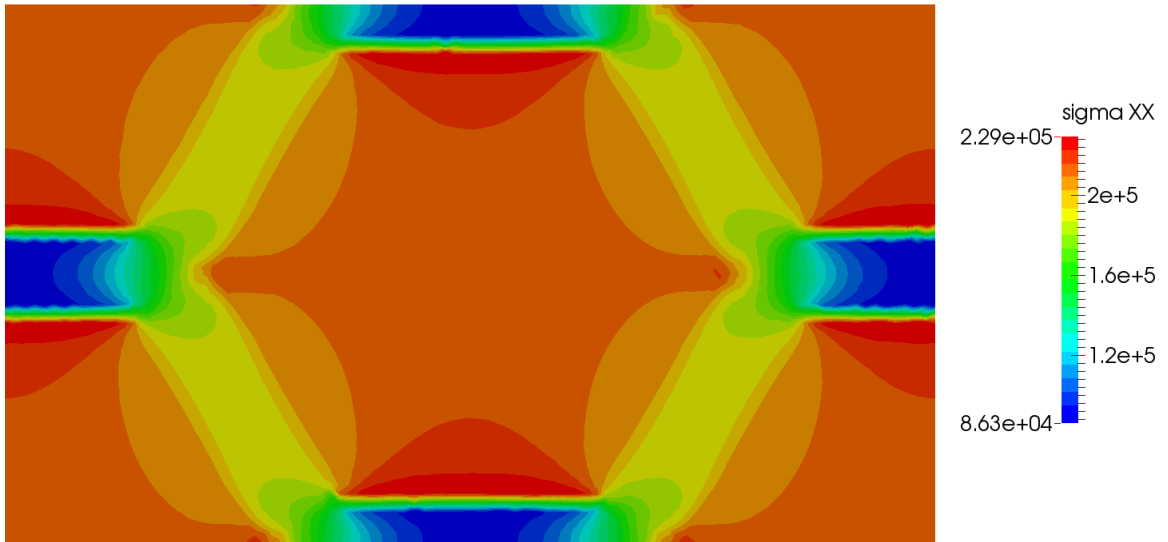


(a)

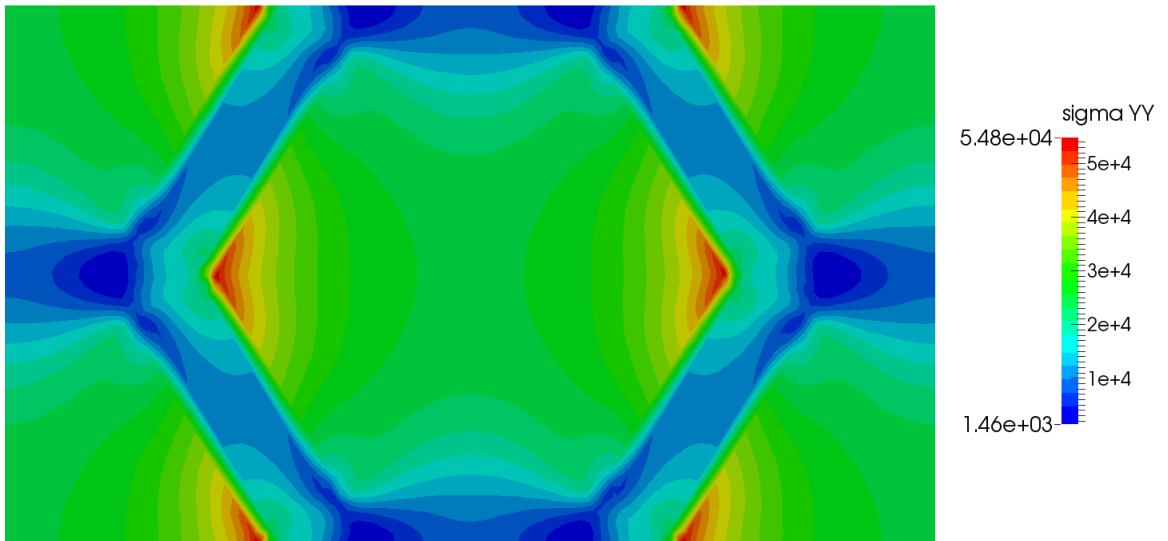


(b)

Figure 3.13: The geometry (a) and the discretization (b) of the two-phase synthetic microstructure.

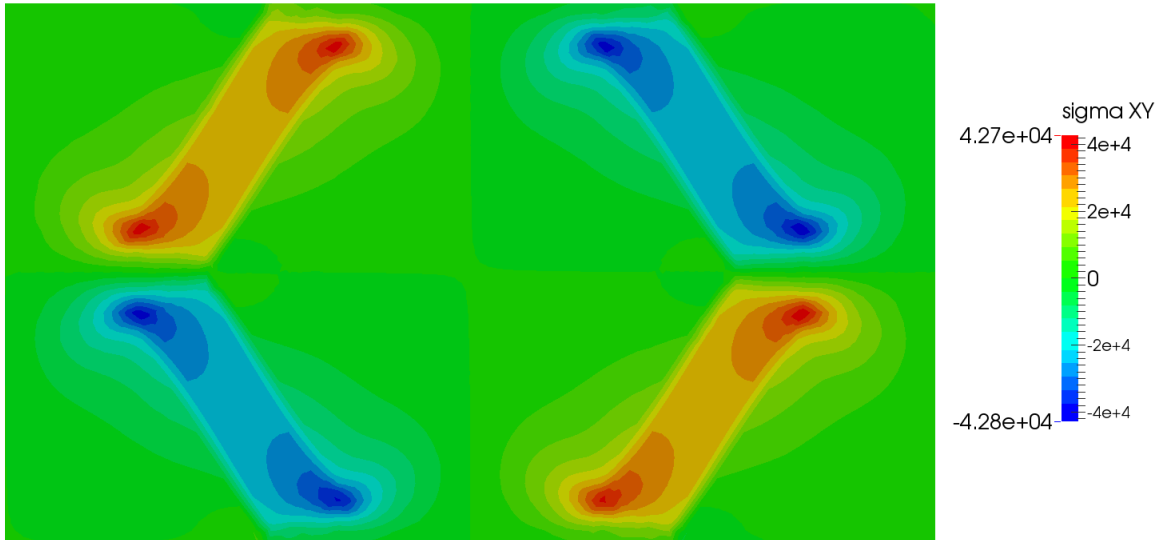


(a)

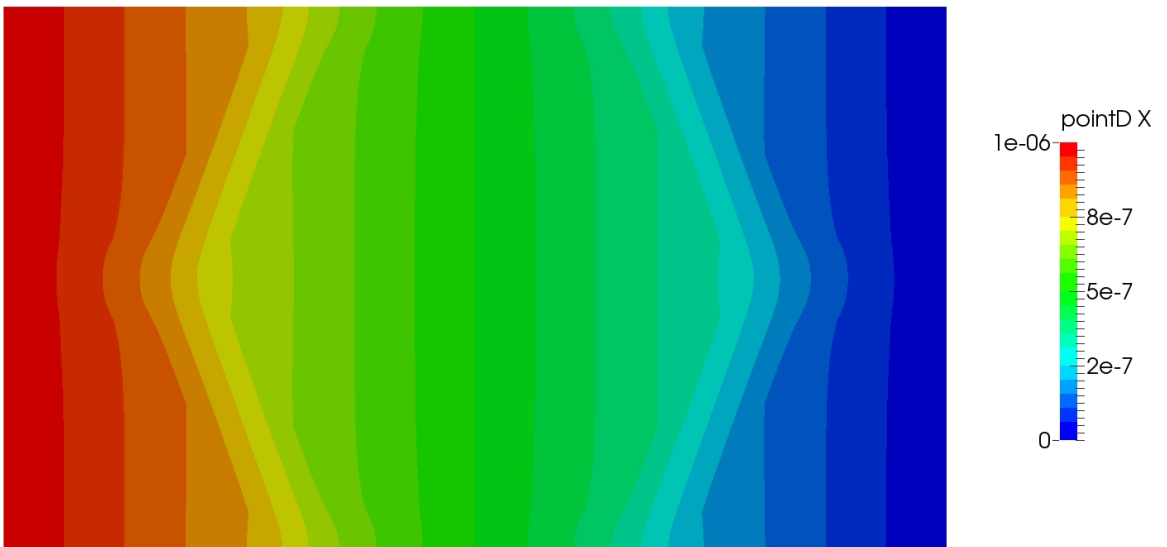


(b)

Figure 3.14: Normal stresses σ_{xx} (a) and σ_{yy} (b) distribution in the numerical microstructure.



(a)



(b)

Figure 3.15: Shear stress τ_{xy} (a) and displacement (b) distribution in the numerical microstructure.

Chapter 4

Conclusion and Perspectives

4.1 Conclusion

This thesis has presented an automatic finite volume mesh generator based on the conjunction between the Delaunay triangulation in a plane and mechanical analogy. This work actually implements the base algorithm described in Chapter 2 within the framework of OpenFOAM® an open-source C++ library for computational continuum mechanics. Notably, the issue of generating polygonal domains has been solved since the original MATLAB® version works with embedded "inpolygon" function for such purpose. Furthermore, the extension to internal interfaces handling has been done.

As expected, the Delaunay triangulation method gave the meshes of high quality. A mesh smoothing through internal and external forces, compensates the problem of the Delaunay method related to boundary cells. An application in discretizing the two-phase materials has shown very good results.

4.2 Perspectives

Whereas the base algorithm is very effective and simple, a wide variety of perspective is achievable.

The first is an extension to three dimension makes possible discretization of complex domains with internal boundaries such as sphere packing.

Another interesting perspective is to replace the Delaunay algorithm with its dual, Voronoi diagram, and generate polygonal cells which possess better quality than triangular cells. Also, on this way it is expected to have smaller amount of cells in the domain.

References

- CUI, J. (2013). *Body-fitting Meshes for the Discontinuous Galerkin Method*. Ph.D. thesis, Technische Universität Darmstadt.
- FAN, X., SUN, S., WEI, W. & JISHENG, K. (2011). Numerical Simulation of Pollutant Transport in Fractured Vuggy Porous Karstic Aquifers. *Journal of Applied Mathematics*.
- GEORGE, P. (1996). *Automatic Mesh Generation and Finite Element Computation*, vol. 4 of *Handbook of Numerical Analysis*. Elsevier.
- HENSHAW, W. (1996). Automatic Grid Generation. *Acta Numerica*, **5**, 121–148.
- NGUYEN, H., BURKARDT, J., GUNZBURGER, M., JU, L. & SAKA, Y. (2009). Constrained CVT meshes and a comparison of triangular mesh generators. *Computational Geometry*, **42**, 1–19.
- OWEN, S.J. (1998). A Survey of Unstructured Mesh Generation Technology. In *Proceedings of the 7th International Meshing Roundtable*, 239–267, Sandia National Laboratories.
- PERSSON, P.O. (2005). *Mesh Generation for Implicit Geometries*. Ph.D. thesis, Massachusetts Institute of Technology.
- PERSSON, P.O. (2006). Discontinuous Galerkin Methods for Fluid Flows and Implicit Mesh Generation.
- PERSSON, P.O. & STRANG, G. (2004). A Simple Mesh generator in MATLAB. *SIAM Review*, **46**, 329–345.
- ROCA NAVARRO, X. (2009). *Paving the Path Towards Automatic Hexahedral Mesh Generation*. Ph.D. thesis, Universitat Politècnica de Catalunya.
- SHAW, C. (1992). *Using Computational Fluid Dynamics*. Prentice Hall.

- THOMPSON, J., SONI, B. & WEATHERILL, N. (1999). *Handbook of Grid Generation*. CRC Press.
- THOMPSON, J.F. & HAMANN, B. (1997). A survey of grid generation techniques and systems with emphasis on recent developments. *Surveys on Mathematics for Industry*, **6**, 289–310.
- TUKOVIĆ, Z., IVANKOVIĆ, A. & KARAOČ, A. (2013). Finite-volume stress analysis in multi-material linear elastic body. *International Journal for Numerical Methods in Engineering*, **93**, 400–419.
- VERSTEEG, H. & MALALASEKERA, W. (2007). *An Introduction to Computational Fluid Dynamics*. Pearson Education Limited, 2nd edn.
- ZIENKIEWICZ, O. & PHILLIPS, D. (1971). An automatic mesh generation scheme for plane and curved surfaces by isoparametric co-ordinates. *International Journal for Numerical Methods in Engineering*, **3**, 519–528.
- ZIENKIEWICZ, O., TAYLOR, R. & J.Z., Z. (2005). *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 6th edn.