IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)
e-ISSN: 2278-1684,p-ISSN: 2320-334X, Volume 11, Issue 6 Ver. V (Nov- Dec. 2014), PP 67-72
www.iosrjournals.org

Three-dimensional mesh generation using Delaunay triangulation

[1]Essaid El Kennassi and [2]Lahbib Bousshine
Laboratoire des Technologies de Construction et Systèmes Industriels, Mechanics Department, ENSEM, Université Hassan II Ain Chock, Casablanca, Morocco.

Abstract: *Data structure is the entity which represents the shape geometry according to mesh topology. In this paper, we represent the geometry of some mesh shapes by using Matlab tools. This used data structure can be implemented to solve engineering physical problems. We create a Matlab code based on the CGAL Delaunay triangulation code. This code gives coordinates nodes and a shape of the mesh elements according to a given shape coordinates. Despite of the high number of commercial mesh generation software, their methodology is not specified. While our work gives a methodology by representing an explicit use of mesh generation.*
Keywords: *CGAL, Matlab, Mesh Generation, Methodology*

I. Introduction

The mesh generation is the first step to solve many engineering problems [1-9-11]. Due to the complexity of technical objects and the high cost of creating and testing prototypes, modeling [10] and simulation are the solutions for solving these problems and also for optimizing time. The data structures [2-4] provide the backbone for physical analysis of the finite element method. This latter is based on a geometrical modeling [10]. On the basis of the definition given by wikipedia, a mesh also called an unstructured grid is a tessellation of a part of the Euclidean plane or Euclidean space by simple shapes, such as 2D-triangles [5] or 3D-tetrahedra [6] in an irregular pattern. Grids of this type may be used in finite element analysis when the analysed input has an irregular shape. The mainly assumption of this work is that on the basis of simple shapes we can mesh more complex shapes.

II. Algorithm

A. Technics

The use of an iterative algorithm to triangulate different shapes is the technical method developed here and it is based on the work of Persson [7]. The Matlab command delaunayn gives the Delaunay Tessellation [8], such that a set of simplices is obtained. Simplices are triangles in 2D and tetrahedra in 3D.
We report in the following the Persson's flowchart [7],

```
dim=size(box,2);
ptol=.001; ttol=.1; L0mult=1+.4/2^(dim-1); deltat=.1;
geps=1e-1*h; deps=sqrt(eps)*h;
% 1. Create initial distribution in bounding box
if dim==1
p=(box(1):h:box(2))';
else
cbox=cell(1,dim);
for ii=1:dim
cbox{ii}=box(1,ii):h:box(2,ii);
end
pp=cell(1,dim);
[pp{:}]=ndgrid(cbox{:});
p=zeros(prod(size(pp{1})),dim);
for ii=1:dim
p(:,ii)=pp{ii}(:);
end
end
% 2. Remove points outside the region, apply the rejection method
p=p(feval(fdist,p,varargin{:})<geps,:);
r0=feval(fh,p);
p=[fix; p(rand(size(p,1),1)<min(r0)^dim./r0.^dim,:)];
N=size(p,1);
count=0;
p0=inf;
while 1
% 3. Retriangulation by Delaunay
if max(sqrt(sum((p-p0).^2,2)))>ttol*h
p0=p;
% 5. Graphical output of the current mesh
if dim==2
trimesh(t,p(:,1),p(:,2),zeros(N,1))
view(2),axis equal,axis off,drawnow
elseif dim==3
if mod(count,5)==0
simpplot(p,t,'p(:,2)>0');
title(['Retriangulation #',int2str(count)])
drawnow
end
else
disp(sprintf('Retriangulation #%d',count))
end
count=count+1;
end
% 6. Move mesh points based on edge lengths L and forces F
bars=p(pair(:,1),:)-p(pair(:,2),:);
L=sqrt(sum(bars.^2,2));
L0=feval(fh,(p(pair(:,1),:)+p(pair(:,2),:))/2);
L0=L0*L0mult*(sum(L.^dim)/sum(L0.^dim))^(1/dim);
F=max(L0-L,0);
Fbar=[bars,-bars].*repmat(F./L,1,2*dim);
dp=full(sparse(pair(:,[ones(1,dim),2*ones(1,dim)]), ...
ones(size(pair,1),1)*[1:dim,1:dim], ...
Fbar,N,dim));
dp(1:size(fix,1),:)=0;
p=p+deltat*dp;
% 7. Bring outside points back to the boundary
d=feval(fdist,p,varargin{:}); ix=d>0;
```

```
t=delaunayn(p);
pmid=zeros(size(t,1),dim)
for ii=1:dim+1
pmid=pmid+p(t(:,ii),:)/(dim+1);
end
t=t(feval(fdist,pmid,varargin{:})<-geps,:);
% 4. Describe each edge by a unique pair of nodes
pair=zeros(0,2);
localpairs=nchoosek(1:dim+1,2);
for ii=1:size(localpairs,1)
pair=[pair;t(:,localpairs(ii,:))];
end
pair=unique(sort(pair,2),'rows');
```

```
gradd=zeros(sum(ix),dim);
for ii=1:dim
a=zeros(1,dim);
a(ii)=deps;
d1x=feval(fdist,p(ix,:)+ones(sum(ix),1)*a,varargin{:});
gradd(:,ii)=(d1x-d(ix))/deps;
end
p(ix,:)=p(ix,:)-d(ix)*ones(1,dim).*gradd;
% 8. Termination criterion
maxdp=max(deltat*sqrt(sum(dp(d<-geps,:).^2,2)));
if maxdp<ptol*h, break; end
end
```

B. Reproductibility

The obtained result is based on the function code fd10 given below. This function determines intersections of six planes. Firstly, the intersection between the plane passing through x-axis at point 2 and the plane passing through z-axis at point -2. Secondly, the plane passing through z-axis at point 2 and the first one. Thirdly, the plane passing through x-axis at point -2 and the second one. Fourthly, the plane passing through y-axis at point 2 and the third one. Fifthly, the plane passing through y-axis at point -2 and the fourth one.

These five intersections give a parallelepiped shape as shown in (Fig. 1). The function code fd10 determines the difference between the parallelepiped distance function and the sphere distance function. The function code dsphere is given in [7] and it is assumed here that the coordinates of central position are xc=0, yc=0, and zc=0, with the radius of sphere equals to 1.75.

```
function d=fd10(p)
x=p(:,1);
y=p(:,2);
z=p(:,3);
d1=x-2;
d2=-x-2;
d3=y-2;
d4=-y-2;
d5=z-2;
d6=-z-2;
d=dintersect(dintersect(dintersect(dintersect(dintersect(d1,d6),d5),d2),d3),d4);
d=ddiff(d,dsphere(p,0,0,0,1.75));
end
```

III. Methodology

The geometry used is relatively simple because it shows how basic shapes are created and meshed. In this way, we can create more complex shapes. The methodology is based on the work of Lang [3] and Persson [7] where the initial grid is generated by the ndgrid Matlab command. This latter generate arrays by transforming the domain specified as bbox. The simulation data are presented in (Tab. 1), (Tab. 2.) and (Tab. 3.). The Delaunay tesselation is assumed by the delaunayn Matlab command, which computes a set of simplexes such as any p-data points, where p design positions, are contained in circumspheres of the simplexes (Tab. 7). The set of simplexes forms the Delaunay tessellation. p is an m-by-3 array representing m points (positions) in 3-dimensional space. t is a number of simplexes-by-(3+1) array where each row contains the indices into p of the vertices of the corresponding simplexes (Tab. 8).

IV. Results and Discussions

A. Parallelepiped With Spherical Cavity Mesh

The function used has the following form

[p,t]=x3ddiffparallelepiped(@fd10,@fh10,0.25,[-2,-2,-2 ;2,2,2],[2,-2,-2 ;2,2,-2 ;-2,2,-2 ;-2,-2,-2 ;2,-2,2 ;2,2,2 ;-2,2,2 ;-2,-2,2 ;1,1,1 ;-1,-1,1 ;-2,-2,-2]); where the initial length edge is equal to 0.25 with 12 fixed positions (nodes). The bbox computational domain [xmin, ymin, zmin ; xmax, ymax, zmax] is taken as [-2,-2,-2 ;2,2,2]. Indeed, this function integrates our fd10-functions and fh10-function elaborated by Persson [7]. Thus, the result related to a parallelepiped with a spherical cavity is presented in (Fig. 1).

Mesh													
Inputs	**Definitions and features**												
Shape	Parallelepiped with spherical cavity (hole)												
Boundaries xmin, xmax, ymin, ymax, zmin, zmax	-2	2	-2	2	-2	2							
Computational domain bbox xmin,ymin,zmin ; xmax,ymax,zmax	-2	2	-2	2	-2	2							
xc, yc, zc, r	0	0	0	1.75									
Initial length edge	0.25												
Fixed nodes	X Y Z	2 -2 -2	2 2 -2	-2 2 -2	-2 -2 -2	2 -2 2	2 2 2	-2 2 2	-2 -2 2	1 1 1	2 2 2	-1 -1 1	-2 -2 -2

Tab. 1. The simulation data for the parallelepiped with spherical cavity.

The result is based on two secondary results, namely a parallelepiped shape and a spherical shape (Fig. 2) and (Fig. 3) which we will detailed in the next subsections. Otherwise, the mesh obtained for the shape (Fig. 1) contains 15236 simplexes and 3567 nodes (Tab. 2) which is obtained after 105 iterations. The three-dimensional simplexes used are 3D-tetrahedra. The initial edge length is equal to 0.25 (Tab. 1).

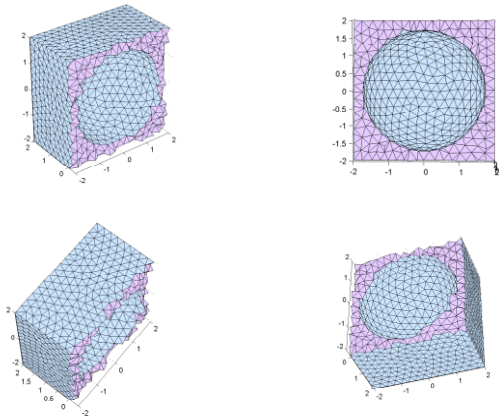

Fig. 1. Parallelepiped mesh with a spherical cavity shape.

The data structure obtained is in good agreement with finite element analysis [10]. The easily manipulation and the usefulness obtained data are the mainly result. Thus, it could be implemented in other software devoted to solve more physical problems.

Outputs	**Definitions and features**
Number of iterations	105
Number of nodes	3567
Tetrahedron simplexes number	15236

Tab. 2. Outputs of the parallelepiped with spherical cavity simulation.

B. Parallelepiped Mesh

For the parallelepiped mesh generation, a three-dimensional domain bbox=[-2,-2,-2 ;2,2,2] is used where the initial length edge equals to 0.9 with the same 12 fixed nodes as below (Tab. 3).

Mesh													
Inputs	**Definitions and features**												
Shape	Parallelepiped												
Boundaries xmin, xmax, ymin, ymax, zmin, zmax	-2	2	-2	2	-2	2							
Computational domain bbox xmin,ymin,zmin ; xmax,ymax,zmax	-2	-2	-2	2	2	2							
Initial length edge	0.9												
Fixed nodes	X Y Z	2 -2 -2	2 2 -2	-2 2 -2	-2 -2 -2	2 -2 2	2 2 2	-2 2 2	-2 -2 2	1 1 1	2 2 2	-1 -1 1	-2 -2 -2

Tab. 3. The simulation data for the parallelepiped.

The mesh generation simulation is obtained after 32 iterations. The nodes number is 136, which is equivalent to the positions number, and we have 421 simplexes as shown in the following (Tab. 4).

Outputs	**Definitions and features**
Number of iterations	32
Number of nodes	136
Tetrahedron simplexes number	421

Tab. 4. Outputs of the parallelepiped simulation.

The mesh obtained is given in (Fig. 2). This result is relatively uniform despite of the use of huniform found by Persson [7].

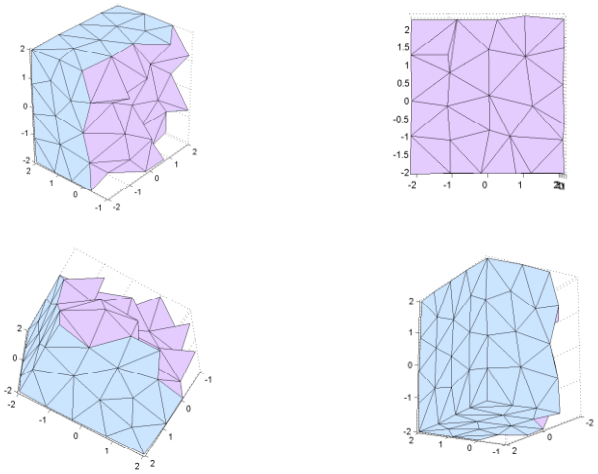

Fig. 2. Parallelepiped mesh shape.

C. Spherical Mesh

For the parallelepiped mesh generation, a three-dimensional domain bbox=[-1,-1,-1 ;1,1,1] is used where the initial length edge is equal to 0.25 without any fixed nodes (Tab. 5).

Mesh						
Inputs	**Definitions and features**					
Shape	Sphere					
Boundaries xmin, xmax, ymin, ymax, zmin, zmax	-1	1	-1	1	-1	1
Computational domain bbox xmin,ymin,zmin ; xmax,ymax,zmax	-1	-1	-1	1	1	1
xc, yc, zc, r	0	0	0	1	-	-
Initial length edge	0.25					

Tab. 5. The simulation data for the sphere.

The mesh generation simulation is obtained after 63 iterations. The nodes number is 257, which is equivalent to the positions number, and we have 1070 simplexes as shown in the following table (Tab. 6),

Outputs	**Definitions and features**
Number of iterations	63
Number of nodes	257
Tetrahedron simplexes number	1070

Tab. 6. Outputs for the sphere simulation.

The mesh obtained is given in the following where this result exhibits a good uniformity (Fig. 3).

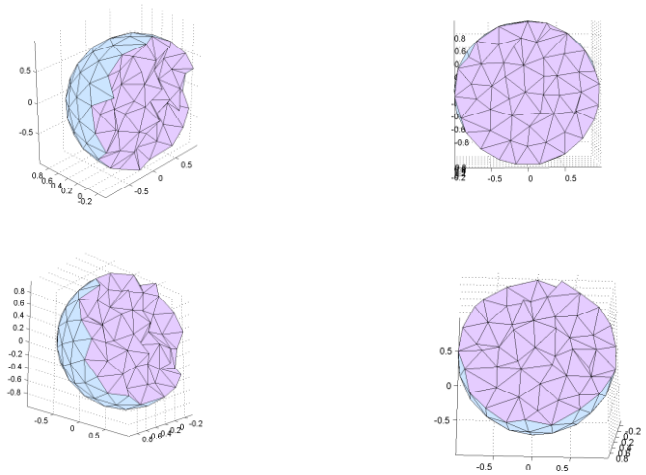

Fig. 3. Spherical mesh shape.

P positions (coordinates)						
x	**y**	**z**		**x**	**y**	**z**
2	-2	-2		…	…	…
2	2	-2		-0,112954710440458	2	2,01059228892260
-2	2	-2		0,144553927627283	2,00931137294874	2
-2	-2	-2	… from 1st	0,371665814462710	2	2,00956309429855
2	-2	2	position to	0,576635291365136	2	2,00926876269988
2	2	2	position	0,832875071008775	2,01229339406564	2
-2	2	2	3567 …	1,08504943688351	2,01140773837162	2
-2	-2	2		1,34546139599652	2	2,00852791430579
1,01036297196570	1,01036297196570	1,01036297196570		1,55927213927976	2,00955097084731	2
1,01036297196570	-1,01036297089585	-		1,79650731437846	2	2,01075774373259
…	…	1,01036297089585		2	2	2
		…				

Tab. 7. Example of data structures concerning positions of the parellelepiped with spherical cavity mesh shape.

t simplexes (the tetrahedra vertices)								
P1	**P2**	**P3**	**P4**		**P1**	**P2**	**P3**	**P4**
572	238	282	2		…	…	…	…
572	282	588	299		2022	2021	2039	2038
1971	1826	1817	1970		2022	1895	1877	2038
1971	1826	1827	1817	…from simplexe 1 to	2022	1895	1878	1877
318	4	302	301	simplexe 15236 …	2022	2039	1895	2038
318	29	302	4		1078	1269	1270	1287
571	572	588	823		1078	1077	1269	1094
571	588	572	282		1078	1269	1287	1094
571	588	282	299		1078	1095	1077	1094
1816	1817	1826	1970		1078	1287	1095	1094
…	…	…	…		1078	1077	1095	858

Tab. 8. Example of data structures concerning the tetrahedra vertices of the parellelepiped with spherical cavity mesh shape.

Engineers can focus on solving problems with high accuracy [3]. This can present a gain of effort and a converging solution to be applicable for optimizing funds, time, materials and gives to wide public better products with minimal cost.

V. Conclusion

The simulation of a parallelepiped with spherical cavity has achieved the three-dimensional mesh. Results analysis demonstrated that the tetrahedral elements give a better uniformity in the case of spherical shapes. Thus, the results obtained in this paper allow us to suggest that our technical method for meshing could be applied to any form. Consequently, several engineering areas could benefit from our results to mesh the forms, such as, finite elements analysis, mechanics, electromagnetism, diffusion, fluid dynamics, chemistry and so on.

Acknowledgements

One of us (E. E.) wants to thank Dr. A. Salhoumi, Master Y. Safi, Pr. M. Madiafi and Dr. F. El Kennassi for their pertinent remarks and carefully reading of the manuscript.

References

[1] P. L. George, Automatic mesh generation and finite element computation, in: P. G. Ciarlet and J. L. Lions (Eds.), Handbook of Numerical Analysis, Vol. IV, Elsevier Science B.V., 1996.
[2] T. J. Baker, Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation, Engineering with Computers, vol 5, 1989, 161-175.
[3] P. Laug, Contribution à la génération automatique de maillage de qualité pour la simulation numérique, habilitation Université Pierre et Marie Curie, mars 2006.
[4] A. V. Aho, J. E. Hopcroft, J. Ullman, Data Structures and Algorithms, Addison-Wesley, Reading, MA, 1983.
[5] P. L. George, Mailleur bidimensionnel du club Modulef, INRIA, 1985.
[6] P. L. George, , Construction et modification de maillage, INRIA, 1988.
[7] P. O. Persson, G. Strang, A simple mesh generator in MATLAB, Soc Individual App Math, SIAM, Rev 46, 2004.
[8] Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.
[9] F. Hecht, Outils et algorithms pour la methode des éléments finis, Université Pierre et Marie Curie, juin 1992.
[10] F. Gustrau, D. Manteuffel, EM Modeling of antennas and RF components for wireless communication systems, Berlin, Germany: Springer-Verlag, 2006.
[11] H. Edelsbrunner, Geometry and Topology for Mesh Generation, Cambridge University Press, Cambridge, United Kingdom, 2001.