



THE HONG KONG
POLYTECHNIC UNIVERSITY

香港理工大學

Pao Yue-kong Library

包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

DISCRETE MINIMAL SURFACES

by

WONG, Chi Kin

**A Thesis for
the Degree of Master of Philosophy**

**Department of Applied Mathematics
The Hong Kong Polytechnic University
Hong Kong**

2000



**Pao Yue-Kong Library
PolyU • Hong Kong**

ACKNOWLEDGEMENTS

First of all, I have to thank my supervisor, Dr. Cheung Leung Fu. He has guided and helped me a lot throughout the project. He has spent a lot of time to improve the contents in this thesis and gave me helpful suggestions. In fact, I have really benefited from him during these three years. I also wish to thank Miss Christine Yeung and all those who have helped me during my research.

CONTENTS

DECLARATION

ACKNOWLEDGEMENTS

ABSTRACT

1.	INTRODUCTION	P-1.1~1.3
2.	DEFINITION OF DISCRETE MINIMAL SURFACES	P-2.1~2.3
3.	FORMULATION	P-3.1~3.4
4.	METHODOLOGY	P-4.1
	4.1 Minimization Algorithm	P-4.2
	4.2 Triangulation	P-4.3
	4.3 Minimization Procedure	P-4.4~4.5
	4.4 Reflection Principles	P-4.6~4.7
	4.5 Refining Technique	P-4.8~4.9
5.	PROOF OF CONVERGENCE	P-5.1~5.10
6.	MATHEMATICA PACKAGE	P-6.1~6.13
7.	GRAPHICS	P-7.1~7.10
8.	CONCLUSION	P-8.1~8.2

REFERENCES

DECLARATION

I hereby declare that the thesis entitled "Discrete Minimal Surfaces" is original and has not been submitted for a degree, or other qualifications in this University or any other institutes. It does not contain any material, partly or wholly, published or written previous by others, except those references quoted in the text.

WONG, Chi Kin

**Abstract of thesis entitled “Discrete Minimal Surfaces”
submitted by WONG, Chi Kin
for the degree of Master of Philosophy
at The Hong Kong Polytechnic University in 2000**

In this thesis, I propose a new numerical procedure to obtain discrete minimal surfaces with fixed or partially free boundaries. Using this procedure, I recover most of the minimal surfaces obtained by other mathematicians such as Hildebrandt, Karcher as presented in the monograph “Minimal Surfaces” by Dierkes et al (1992) [1]. The same procedure also gives rise to new graphics of partially free boundary minimal surfaces as depicted in the popular scientific account “The parsimonious universe : shape and form in the natural world” by Hildebrandt and Tromba (1996) [3]. The origin of the minimization algorithm comes from the paper of Pinkall and Polthier [4] published in the journal *Experimental Mathematics* in 1993. My contributions consists of :

- (i) improving the algorithm of Pinkall and Polthier so that one point at a time needs be minimized, as a consequence of which the computer code is greatly simplified.
- (ii) writing down the codes in the language of Mathematica and implementing it,
- (iii) proving the convergence of my algorithm for the fixed boundary case,
- (iv) using this algorithm to produce graphics of most of the famous minimal surfaces in the book on minimal surfaces by Dierkes et al [1] as well as some additional new minimal surfaces by pasting and refinement techniques starting from some simple fundamental pieces, and
- (v) writing down the Mathematica codes to handle the partially free boundary problem case.

Discrete Minimal Surfaces

1. INTRODUCTION

Minimal surface, the mathematical synonym for soap film, is a challenging branch of mathematics that began to arise since the historic experimental investigation on liquid films performed by the Belgian physicist J. Plateau [5]. It was in the mid-nineteenth century when Plateau asked the famous question as to whether each closed non-self-intersecting curve in the space spans a soap film. In today's language, this can be reformulated as the question asking for the existence of a minimal surface of disk-type bounding each given Jordan curve in R^3 . Nowadays, as many mathematicians have demonstrated, there are numerous examples of more complex soap films spanning a Jordan curve¹.

Despite the simple appearance of Plateau's question, its solution seemed to be very difficult, and evaded many courageous attempts of some generations of mathematicians. The first mathematically rigorous solution to this problem was given in 1930 when J. Douglas [2] introduced a new idea to replace the area functional¹¹ by the energy functional and to minimize not the area functional directly but the energy integral. This energy integral is the so-called Dirichlet integral

$$(1.1) \quad D(f) = \iint_{\Omega} |\nabla f|^2 \, dx \, dy$$

A soap film is a minimal surface because a soap film naturally minimizes surface tension, whereas surface tension is proportional to surface area.

To find minimal surfaces, we are indeed looking for a surface which gives the minimum of area.

whereas the conventional area function is defined by

$$(1.2) \quad A(f) = \iint_{\Omega} |f_x \times f_y| \, dx \, dy$$

where $\Omega \subset \mathbb{R}^2$ and $f : \Omega \rightarrow \mathbb{R}^3$ satisfies $f(\partial \Omega) = \Gamma$ for a given Jordan curve $\Gamma \subset \mathbb{R}^3$.

Since the solution of Plateau's problem by J. Douglas in the late thirties, variants of Plateau's question were investigated, such as the so-called partially free boundary value problem and the thread problem. In the partially free boundary value problem, one has Jordan arcs whose ends sit on a surface. The aim is to look for a minimal surface with part of its boundary sitting on the arcs while the rest of it sitting on the surface.

In this paper, we study both cases of fixed boundary and partially free boundary problems. Part of our approach is based on the algorithm proposed by Pinkall and Polthier [4], where we apply a variant of their iteration technique in the minimization procedure to find the minimal surface numerically.

The plan of this dissertation is as follows: In the next section, we formulate definitions concerning discrete minimal surfaces and describe basic consequences of these definitions in our setting. After that, the procedure of our methodology to work with a discrete surface under our minimization algorithm is presented. Subsequently, we prove the convergence of our algorithm, and finally we describe the implementation of our algorithm using the software Mathematica. In fact, we set up a package using Mathematica to generate all the graphics outputs in this work. The detail of the package is mentioned in Section 6.

In Section 7 of this paper, all the computer graphics constructed will be shown and the conclusion will be given in the final section.

2. DEFINITION OF DISCRETE MINIMAL SURFACES

We first define the notion of a discrete boundary curve.

Definition 1

(D1) A discrete boundary curve Γ_D in R^3 is formed by a set of straight lines Γ_i with endpoints at P_i and P_{i+1} ($i = 0, 1, 2, \dots, n; P_{n+1} = P_0$), which are points on Γ_D . Furthermore, the curve Γ_D is non-self-intersecting and not knotted. (See Figure 2.1).

Note that since $P_{n+1} = P_0$,

$$(2.1) \quad \Gamma_n = P_n P_{n+1} = P_n P_0.$$

Hence, we may write (D1) as

$$(2.2) \quad \Gamma_D = \bigcup \{ \Gamma_0, \Gamma_1, \dots, \Gamma_n \}$$

where $\Gamma_i = P_i P_{i+1}$ and $P_i \in \Gamma_i$ ($i = 0, 1, 2, \dots, n; P_{n+1} = P_0$).

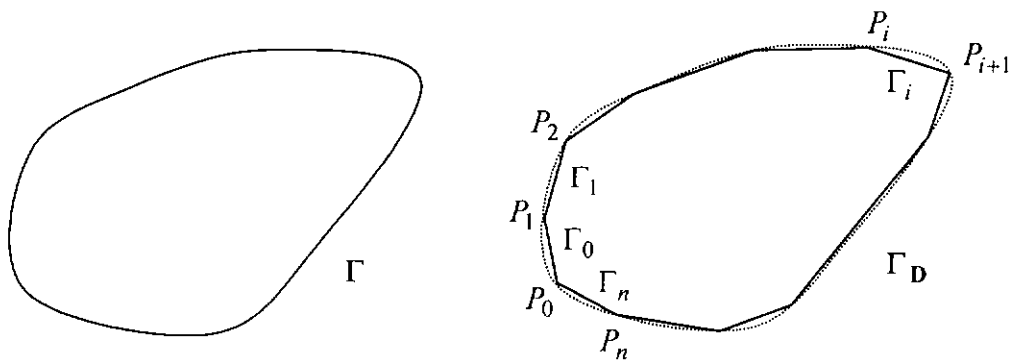


Figure 2.1 A smooth boundary curve Γ and a discrete boundary curve Γ_D approximating it

Definition 2

(D2) A discrete surface in R^3 with boundary Γ_D in R^3 is a simplicial complex consisting of triangles such that its boundary is Γ_D .

Definition 3

Let Ω be a domain in R^2 and consider Δ , a triangulation of Ω , i.e. a polygonal domain satisfying the following properties:

- (D3.a) The vertices of $\partial\Delta$, the boundary of Δ , is a subset of $\partial\Omega$.
- (D3.b) $\bar{\Delta}$ is the union of finitely many triangles $\{\Delta_j\}$ whose interiors are pairwise disjoint and such that each of their edges is either an edge of another triangle or an edge of the polygon $\partial\Delta$.
- (D3.c) A discrete parametric surface (or a "discrete surface" in short) with parameter domain Ω is a piecewise linear mapping f from Δ to R^3 such that $f(\partial\Delta)$ is a discrete boundary curve Γ_D . (For simplicity, Γ_D will be denoted by Γ in the sequel).
- (D3.d) A discrete minimal surface with fixed boundary is a discrete surface with interior vertices p_1, p_2, \dots, p_k such that its area is the least when compared to nearby surface in the following sense:

$$\text{Area of surface with interior vertices } p_1, \dots, p_k \leq \text{Area of surface with interior vertices } \tilde{p}_1, \dots, \tilde{p}_k$$

if $\|\tilde{p}_1 - p_1\| < \delta, \dots, \|\tilde{p}_k - p_k\| < \delta$; for $\exists \delta$ a small positive number.

Let Δ and M denote the triangulated domain and the discrete surface respectively. For a piecewise linear map, $f: \Delta \rightarrow f(\Delta) = M \subset R^3$, the discrete minimal surface, say M , is the one whose total surface area is the least.

$$(2.3) \quad \text{Area}(M) = \inf \{ \text{Area}(f: \Delta \rightarrow R^3), \forall \text{ piecewise linear } f \text{ satisfying} \\ \text{the boundary condition } f(\partial\Delta) = \Gamma \}$$

where $\text{Area}(X)$ is the area function of any discrete surface X .

3. FORMULATION

Our problem can now be formulated as follows. For simplicity, let Γ be a simple closed curve and M a discrete surface with boundary Γ , and let $f : \Delta \rightarrow R^3$ be a parametrization of a surface over a two dimensional domain $\Delta \subset R^2$, where Δ may be a fixed "disk-like" domain or a fixed rectangular domain, i.e.

$$\Delta = \{(x, y) \in R^2 : x^2 + y^2 \leq 1\} \text{ or } \Delta = \{(x, y) \in R^2 : 0 \leq x, y \leq 1\}.$$

To begin with, we note that for any two vectors $v, w \in R^3$, we have

$$\left(|v|^2 - |w|^2 \right)^2 \geq 0$$

which implies $\left(|v|^2 + |w|^2 \right)^2 \geq 4|v|^2|w|^2$ and hence

$$\frac{1}{4} \left(|v|^2 + |w|^2 \right)^2 \geq |v|^2|w|^2 \geq |v|^2|w|^2 - \langle v, w \rangle^2,$$

where $|v|^2|w|^2 - \langle v, w \rangle^2 = |v \times w|^2$. By letting $v = f_x$, $w = f_y$ and taking square root, we have

$$\frac{1}{2} \left(|f_x|^2 + |f_y|^2 \right) = \frac{1}{2} |\nabla f|^2 \geq |f_x \times f_y|.$$

After integration, it follows immediately that

$$(3.1) \quad E_D(f) = \frac{1}{2} D(f) \geq A(f)$$

where $E_D(f) = \frac{1}{2} \iint_{\Delta} |\nabla f|^2 dx dy$ is called the *Dirichlet energy* of f and

equality holds if and only if f is a conformal map.

Following the paper of Pinkall and Polthier [4], we recall the following lemmas and theorems which allow us to represent the Dirichlet energy of a piecewise linear map between two triangulated surfaces using geometric data of the discrete surfaces.

Lemma 1 *Let f be a linear map between two triangles Δ_A and Δ_B in two vector spaces with constant metrics A and B . Then the Dirichlet energy of f is*

$$(3.2) \quad E_D(f) = \frac{1}{4} \left(\cot \alpha |e_1|_B^2 + \cot \beta |e_2|_B^2 + \cot \gamma |e_3|_B^2 \right)$$

where α, β, γ are the angles of Δ_A with respect to the metric A , and $|e_1|_B, |e_2|_B, |e_3|_B$ are the lengths of their corresponding sides of Δ_B with respect to the metric B .

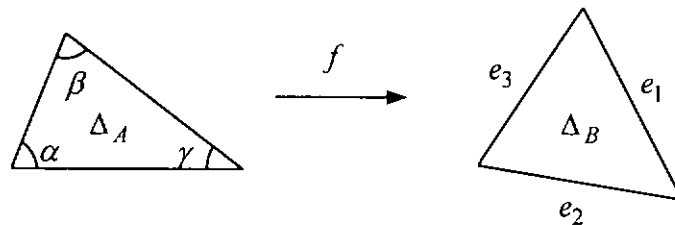


Figure 3.1 Linear map between two triangles

Theorem 1 *The Dirichlet energy of a piecewise linear map*

$f : \Delta \rightarrow M$ *of Δ bijectively onto M is given by*

$$(3.3) \quad E_D(f) = \sum_{\text{triangle } i} E_D(f_i) = \frac{1}{4} \sum_{\text{edge } j} (\cot \alpha_j + \cot \beta_j) |e_j|^2$$

where α_j and β_j are the corresponding angles opposite to the edge

e_j in the two adjacent triangles. (See Figure 3.2)

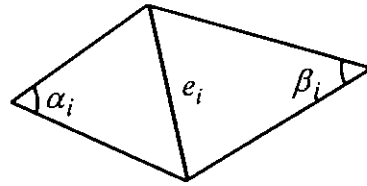


Figure 3.2 Notation for angles and edges

For edges along the boundary of M where one of the angles is missing, we take the value of this corresponding term to be zero. As an illustration, a neighborhood of a vertex, say p , on a discrete surface is shown in Figure 3.3 .

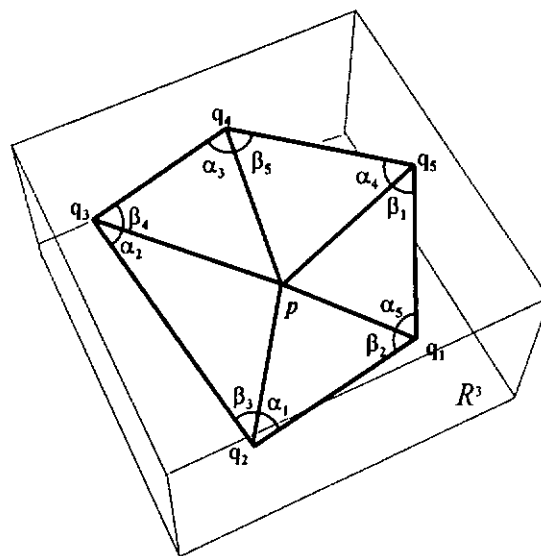


Figure 3.3 Neighborhood of a vertex p

Definition 4 (Pinkall-Polthier [4])

Let $f : \Delta \rightarrow M$ be a piecewise linear map in R^3 . Then f is called harmonicⁱⁱⁱ if and only if at every interior vertex $p \in M$

$$(3.4) \quad \frac{\partial}{\partial p} E_p(f) = 0$$

holds, where $\frac{\partial}{\partial p} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)^T$

ⁱⁱⁱ A discrete harmonic map is a critical point for the Dirichlet energy functional with respect to variations of interior surface vertices in R^3 .

For a discrete surface with energy defined in (3.2), the minimality condition translates to

$$(3.5) \quad \frac{1}{2} \sum_{\substack{\text{neighbouring} \\ \text{vertices } q_j \\ \text{of } p}} (\cot \alpha_j + \cot \beta_j)(p - q_j) = 0$$

where p is an interior vertex. Equivalently

$$(3.6) \quad p = \frac{\sum_j (\cot \alpha_j + \cot \beta_j) q_j}{\sum_j (\cot \alpha_j + \cot \beta_j)}$$

If all interior vertices p satisfy this condition, then f is a critical point for the discrete energy functional.

For the image points along the boundary, their movement is subject to the relevant boundary conditions:

- If the problem involves symmetries and partially free boundaries, the image boundary points may vary either along lines or in the image plane.
- The points of the image boundary are not allowed to move for all other cases.

In the next section, we make a closer look at the numerical procedure to compute a discrete minimal surface following the ideas discussed above.

4. METHODOLOGY

In section 3 we presented a method to find a discrete harmonic map for a given boundary Γ in R^3 and a given triangulated domain Δ in R^2 . Note that Γ is a set of Jordan curves and each may be a fixed curve, a symmetry line or a planar symmetry arc. Now, our problem is

Problem 3.1 *Let Γ be a given discrete boundary configuration, find a discrete minimal surface with boundary on Γ so that it can be extended across symmetry lines and arcs of ∂M if necessary.*

Section 4.1 contains a simplified description of the algorithm, in which we allow reflections of a fundamental piece of the discrete minimal surface and changes of the triangulation in the domain during iterations (*See Section 4.4 & 4.5 for more details*). Our minimization algorithm can solve both fixed and partially free boundary cases. Before we begin with the discussion of the minimization procedure, we consider the triangulation of the domain (*Section 4.2*). Then, in Section 4.3 we explain briefly the minimization procedure of how to vary points on the surface such that a discrete minimal surface with fixed boundary or partially free boundary can be obtained. The reflecting and refining techniques applied as a useful tool for speeding up the process of approximating surfaces that converge to a minimum are described in Section 4.4 and 4.5 respectively.

4.1 Minimization Algorithm

Step 1 : Let $\Delta^{(1)}$ and $M_0^{(1)}$ be the initial triangulated domain and an arbitrary initial surface with boundary $\partial M_0^{(1)} = \Gamma$ respectively; set $n = 1$ and $i = 0$.

Step 2 : Compute the next surface $M_{i+1}^{(n)}$ as the minimizer of

$$E_D(f_i : M_i^{(n)} \rightarrow M_{i+1}^{(n)}) = \inf_{\substack{M \subset R^3 \\ \partial M = \Gamma_D}} E_D(f_i : M_i^{(n)} \rightarrow M)$$

where M denotes an arbitrary discrete surface satisfying the boundary condition and the condition for the minimum given by (3.6).

Step 3 : For any tolerance $\varepsilon > 0$, while

$$(4.1) \quad \left| \text{Area}(M_i^{(n)}) - \text{Area}(M_{i+1}^{(n)}) \right| > \varepsilon,$$

we set i to $i+1$ and repeat **Step 2**.

Step 4 : Complete the iteration steps, say m , until one arrives at error below ε , and output the limiting minimum surface $M_m^{(1)}$.

Step 5 : Increase the number of grids and construct the next triangulated domain $\Delta^{(n+1)}$; set n to $n+1$ and $i = 0$.

Step 6 : Repeat **Step 2** until the size of grids is sufficiently small.

Step 7 : Reflect the fundamental piece across the symmetry planes if the minimized surface satisfies certain symmetry properties.

4.2 Triangulation

In general, most of the discrete surfaces in our project are mappings

$$f_i : M_i^{(n)} \rightarrow R^3$$

from a domain $M_i^{(n)}$ into R^3 , where $\Delta^{(1)} \subset R^2$ and $M_i^{(n)} \subset R^3$ for $i = 1, 2, \dots$. To construct an initial surface in the Euclidean space, we may set a triangulated rectangular domain Δ as

$$\Delta = \{ (x,y) \in R^2 : a \leq x \leq b, c \leq y \leq d \} .$$

In the case when the surface is not bounded by four sides a disk-type domain is more appropriate. Hence in our algorithm we most of the time set

$$\Delta = \{ (x,y) \in R^2 : x^2 + y^2 \leq r^2 \} .$$

For instance, a k -sided surface is parameterized by a disk-type domain divided into k sectors. A sector of the general disk-type domain and the triangulation setting are shown in Figure 4.2 .

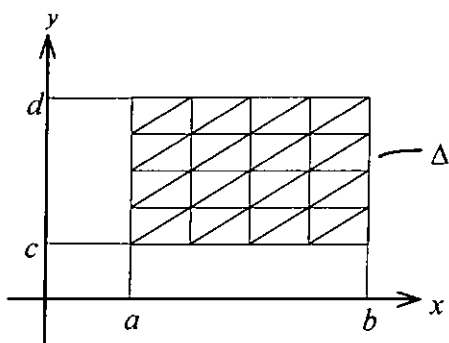


Figure 4.1 A rectangular domain

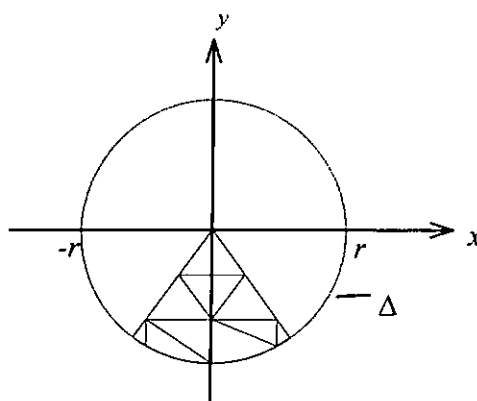


Figure 4.2 A "disk-type" domain

4.3 Minimization Procedure

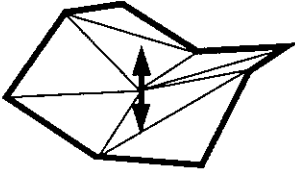
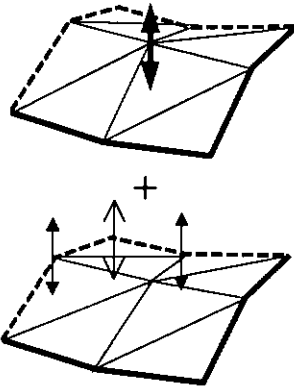
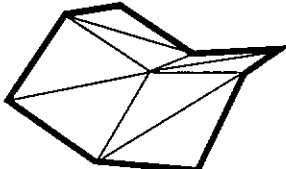
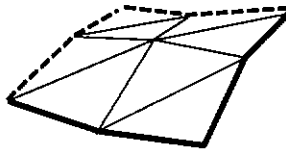
In the following, we denote $M_i^{(n)}$ simply by M_i . As mentioned in the algorithm in Section 4.1, for each iteration step, we have to repeat step 2 so as to find out a sequence of approximating surfaces M_i which converge to the minimum M . Here, we would describe briefly how the points on M_i vary in the image space so as to minimize the surface area.

For the fixed boundary case, we only need to consider the movement of the interior points on the surface during each iteration. However, for the partially free boundary case, we first regard the points on the free boundaries to be fixed while we are minimizing the interior region. After each interior minimization, we vary the free boundary points with respect to the boundary conditions. These two steps are repeatedly carried out until the stopping criterion is satisfied. In short, the minimization procedure consists mainly of two steps, which are







- interior minimization - to vary only the interior points one at a time such that each of them satisfies the condition (3.6).
- Boundary minimization - to move those points on the partially free boundary along lines or in planes subject to the boundary condition.

The minimization procedures of the fixed boundary and partially free boundary cases are illustrated and summarized in Table 4.1.

Table 4.1 Summary of the minimization procedure

Boundaries	Fixed	Partially free
Repeating Process		
Output $M.$		

Remark:

-  moving w.r.t condition (3.6)
-  moving in plane
-  moving along line
-  fixed boundary
-  free boundary
-  edge

4.4 Reflection Principles

In some of the problems, the symmetry properties can speed up the minimization process, because we only need to consider a specific part of the discrete surface which is bounded partially by symmetry lines or planes and then paste them together. This part of the surface is called the fundamental domain. From [4], we have

Theorem 4.1

- (i) *Every straight line contained in a minimal surface is an axis of symmetry of the surface.*
- (ii) *If a minimal surface intersects some plane Π perpendicularly, then Π is a plane of symmetry of the surface.*

Let us consider a simple example, the case of a catenoid. We construct a four-sided fundamental domain, three of whose sides are planar symmetry curves and the fourth side is fixed. (See Figure 4.3)



Figure 4.3 *The fundamental domain*

After going through the process of minimization of a fundamental piece, the reflections subject to the symmetry lines or planes is then used to generate a complete surface. Referring to the last example, a catenoid can be constructed by 8 copies of the minimized fundamental domain by repeated reflections about the symmetry planes. (See Figure 4.4)

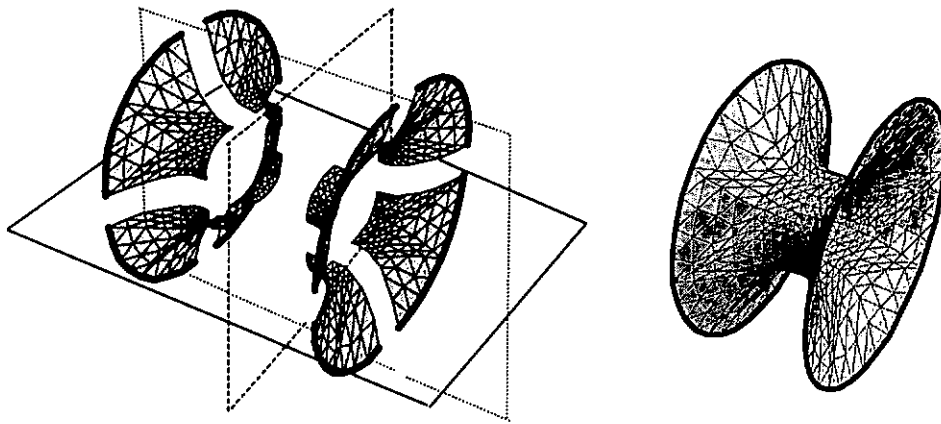


Figure 4.4 *A catenoid is formed by reflecting the fundamental piece.*

4.5 Refining Technique

During iteration, the triangulation is allowed to change by refining grids. The advantage of doing this is that at the beginning less grids are required to obtain the preliminary shape of the minimal surface, and then we use this as the initial surface in the next iteration after the number of grids has been increased by refinement. The iteration process is carried on repeatedly until the number of grids is sufficiently large. To describe our technique, we consider the example:

Let $p_1^{(n)}, p_2^{(n)}, p_3^{(n)}$ be the vertices of a triangle in the image space at the n th iteration step. Then

$$p_i^{(n+1)} = p_i^{(n)} \quad , \text{ for } i = 1, 2, 3$$

$$\text{and } q_{ij}^{(n+1)} = \frac{p_i^{(n)} + p_j^{(n)}}{2} \quad , \text{ for } i, j = 1, 2, 3 \ (i < j),$$

where $q_{ij}^{(n+1)}$ are the new points added in the $(n+1)$ iteration step.

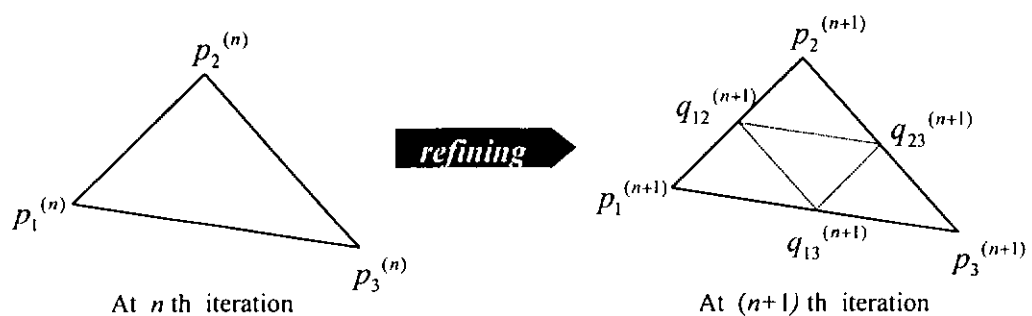


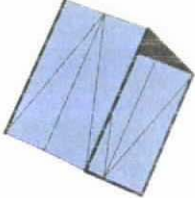

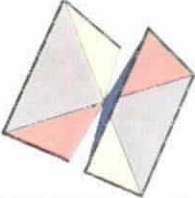
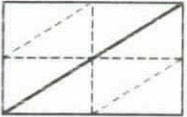

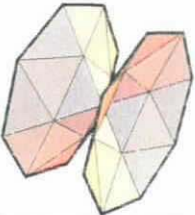

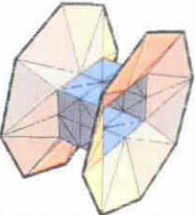


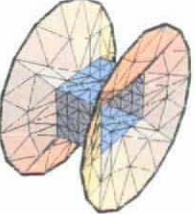

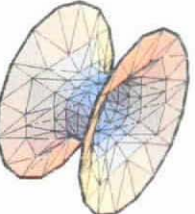

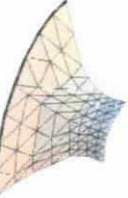
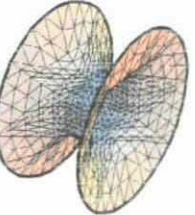
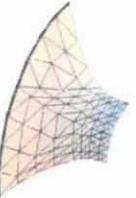
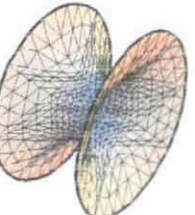


Figure 4.5 A triangle is refined into four smaller triangles

After refining, we repeat the minimization procedure with all the image points, including the original points and newly added points. In the next page, an example is given to describe how a catenoid is generated using this technique.

n	Domain $\Delta^{(n)}$	Before Minimization	After Minimization
1		<p>Image </p> <p>Surface </p>	<p>Image </p> <p>Surface </p>
2		<p>Image </p> <p>Surface </p>	<p>Image </p> <p>Surface </p>
3		<p>Image </p> <p>Surface </p>	<p>Image </p> <p>Surface </p>
4		<p>Image </p> <p>Surface </p>	<p>Image </p> <p>Surface </p>

5. PROOF OF CONVERGENCE

In this section, we will

- (i) show that the functional $E_D(f)$ stated in (3.2) is a positive quadratic function on the set of non-degenerate triangles and hence has a unique minimum,
- (ii) show that the decreasing sequence of discrete minimal surfaces in [4] contains a convergent subsequence,
- (iii) prove that the limiting discrete surface is minimal.

For simplicity, the analogous convergence of the algorithm for the partially free boundary value problem will not be discussed.

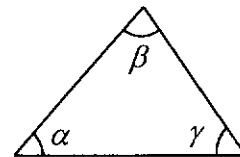
Following Pinkall and Polthier [4], the following non-degeneracy will be assumed throughout :

Definition 4 *A set of triangles is called non-degenerate if each*

element in it satisfies

$$0 < m_1 \leq \alpha, \beta, \gamma \leq m_2 < \pi$$

for some $m_1, m_2 \in R$



Now, let f be a linear map between two triangulated surfaces. The following lemma gives an explicit expression of $E_D(f)$ in terms of the corresponding angles and the positive definiteness of this functional.

Lemma 2 *The Dirichlet energy $E_D(f)$ is a positive quadratic functional on the set of non-degenerate triangles, where*

$$E_D(f) = \frac{1}{4} \left(\cot \alpha |\mathbf{a}|^2 + \cot \beta |\mathbf{b}|^2 + \cot \gamma |\mathbf{c}|^2 \right).$$

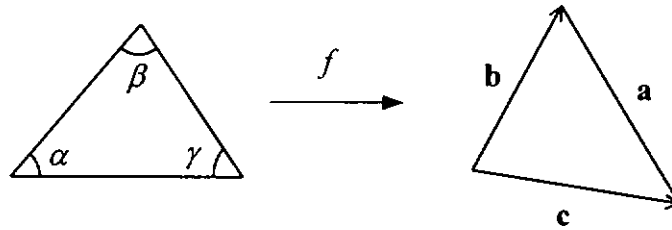


Figure 5.1 Mapping between two triangles

Proof

First note that this formula is just a reformulation of (3.2) of Lemma 1.

Putting $\mathbf{a} = \mathbf{c} - \mathbf{b}$ then (3.2) can be rewritten as

$$\begin{aligned} E_D(f) &= \frac{1}{4} \left(\cot \alpha |\mathbf{b} - \mathbf{c}|^2 + \cot \beta |\mathbf{b}|^2 + \cot \gamma |\mathbf{c}|^2 \right) \\ &= \frac{1}{4} \left[\cot \alpha \left(|\mathbf{b}|^2 + |\mathbf{c}|^2 - 2 \langle \mathbf{b}, \mathbf{c} \rangle \right) + \cot \beta |\mathbf{b}|^2 + \cot \gamma |\mathbf{c}|^2 \right] \\ &= \frac{1}{4} \left[(\cot \alpha + \cot \beta) |\mathbf{b}|^2 + (\cot \alpha + \cot \gamma) |\mathbf{c}|^2 - 2 \cot \alpha \langle \mathbf{b}, \mathbf{c} \rangle \right] \\ &= \frac{1}{4} (\mathbf{b}, \mathbf{c}) \begin{pmatrix} \cot \alpha + \cot \beta & -\cot \alpha \\ -\cot \alpha & \cot \alpha + \cot \gamma \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \end{aligned}$$

Here the usual multiplication is to be replaced by dot product.

For simplicity, we let $Q = \begin{pmatrix} \cot \alpha + \cot \beta & -\cot \alpha \\ -\cot \alpha & \cot \alpha + \cot \gamma \end{pmatrix}$ and let us check

the signs of $\cot \alpha + \cot \beta$ and $\cot \alpha + \cot \gamma$ first.

$$\cot \alpha + \cot \beta = \frac{\cos \alpha \sin \beta + \sin \alpha \cos \beta}{\sin \alpha \sin \beta} = \frac{\sin(\alpha + \beta)}{\sin \alpha \sin \beta}$$

$$\text{and } \cot \alpha + \cot \gamma = \frac{\cos \alpha \sin \gamma + \sin \alpha \cos \gamma}{\sin \alpha \sin \gamma} = \frac{\sin(\alpha + \gamma)}{\sin \alpha \sin \gamma}$$

For $\alpha + \beta + \gamma = \pi$ and $0 < \alpha, \beta, \gamma < \pi$, we have

$$\frac{\sin(\alpha + \beta)}{\sin \alpha \sin \beta} > 0 \quad \text{and} \quad \frac{\sin(\alpha + \gamma)}{\sin \alpha \sin \gamma} > 0 .$$

Hence $\cot \alpha + \cot \beta$ and $\cot \alpha + \cot \gamma$ must be positive. Next, we have to check whether $\det Q$ is greater than zero.

$$\begin{aligned} \det Q &= \begin{vmatrix} \cot \alpha + \cot \beta & -\cot \alpha \\ -\cot \alpha & \cot \alpha + \cot \gamma \end{vmatrix} \\ &= \cot \alpha \cot \beta + \cot \alpha \cot \gamma + \cot \gamma \cot \beta \end{aligned}$$

Replacing γ by $\pi - \alpha - \beta$, we can then have

$$\begin{aligned} \det Q &= \cot \alpha \cot \beta + \cot(\pi - \alpha - \beta)(\cot \alpha + \cot \beta) \\ &= \cot \alpha \cot \beta - \cot(\alpha + \beta)(\cot \alpha + \cot \beta) \\ &= \frac{1}{\tan \alpha \tan \beta} - \frac{1 - \tan \alpha \tan \beta}{\tan \alpha + \tan \beta} \left(\frac{1}{\tan \alpha} + \frac{1}{\tan \beta} \right) \\ &= 1 \\ &> 0 \end{aligned}$$

As a result, the matrix Q is positive definite. \square

Since $E_D(f)$ is a positive quadratic function of the vertices, it has a unique minimum. Next we construct a minimizing sequence and show that the sequence converges to a limiting minimal surface. For convenience, we consider the case where only one interior vertex exists.

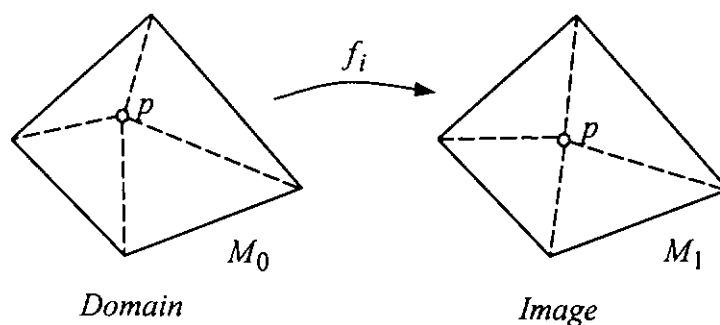


Figure 5.2 Surface M_1 is generated from an initial surface M_0

Starting with an initial surface M_0 , we get M_1 by solving the system of linear equations obtained by differentiating E_D with respect to the interior vertex,

$$\text{i.e. } \frac{\partial}{\partial p} E_D(f_i) = \frac{1}{2} \sum (\cot \alpha'_j + \cot \beta'_j)(p - q'_j) = 0$$

$$\text{thereby obtaining } p = \frac{\sum (\cot \alpha'_j + \cot \beta'_j) q'_j}{\sum (\cot \alpha'_j + \cot \beta'_j)}$$

(p depends on i)

In the same way, one gets M_{i+1} from M_i by solving a similar equation. Next we show the sequence of the areas of these discrete surfaces M_i are monotonically decreasing.

$$\begin{aligned}
A(f(M_i)) &= E_D(\text{id}: M_i \rightarrow \mathbb{R}^3) \\
&\geq E_D(f_i) \\
&= A(f(M_{i+1})) + E_C(f) \\
&\geq A(f(M_{i+1}))
\end{aligned}$$

where $E_C(f) = E_D(f) - A(f(M))$ is called the conformal energy of the mapping f .

From the above, we see that the sequence of numbers $\{A(f(M_i))\}$ is monotonically decreasing. In the following, we argue that a subsequence of $\{M_i\}$ converges to a discrete minimal surface. To this end, we first show

Lemma 3

The set of vertices of the sequence of surfaces $\{M_i\}$ are bounded.

Proof

Suppose that there are N layers of triangles forming the surface M_i , and denote all the vertices of M_i by $p_{kj}^{(i)}$ for $j = 1, 2, \dots, n_k$ and $k = 0, 1, 2, \dots, N$, where n_k represents the number of vertices along the outer side of the k -th outermost layer, the interior vertices and boundary vertices can then be expressed as $\{p_{k1}^{(i)}, \dots, p_{kn_k}^{(i)}\}$ where $k = 1, 2, \dots, N$ and $\{p_{01}^{(i)}, \dots, p_{0n_0}^{(i)}\}$ respectively. (See Figure 5.3)

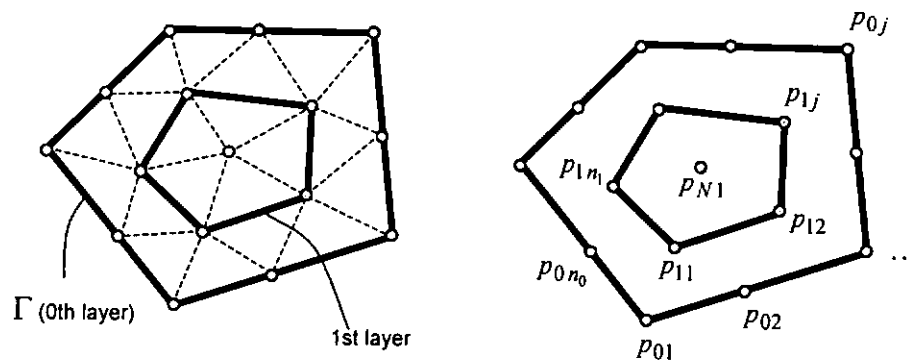


Figure 5.3 The notation of vertices in the N layers

For the boundary value problem with fixed boundary Γ , we have to show that the vertices are within a fixed distance. Let us consider the outermost layer first and look at one such triangle $p_{01}p_{02}p_{12}$:

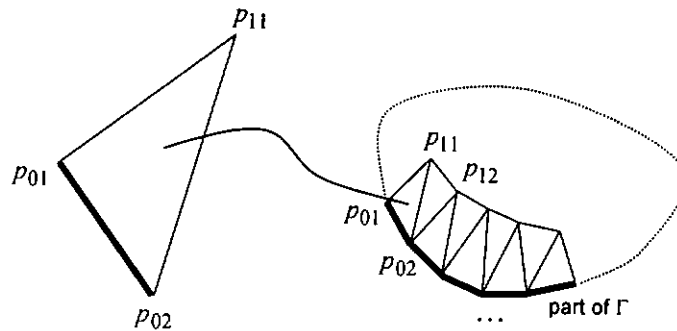


Figure 5.4 A particular triangle on the outermost layer

Each of the lengths of $p_{0j}p_{0j+1}$ on the fixed boundary must be finite.

Hence

$$(5.1) \quad 0 < L(p_{01}p_{02}) \leq \text{Max} \{ L(p_{01}p_{02}), L(p_{02}p_{03}), \dots, L(p_{0n_0}p_{01}) \} \\ \leq M \quad \text{for some real no. } M$$

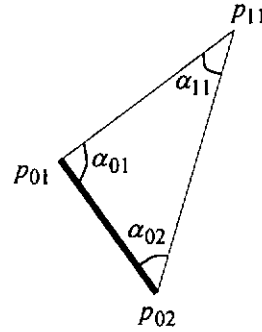
where $L(p_A p_B)$ is the length of between points p_A and p_B .

For convenience, we call the corresponding angles inside $\Delta p_{01}p_{02}p_{11}$ α_{01} , α_{02} and α_{11} respectively.

Then, by sine formula, we can have

$$\frac{L(p_{01}p_{11})}{\sin \alpha_{02}} = \frac{L(p_{01}p_{02})}{\sin \alpha_{11}}$$

$$L(p_{01}p_{11}) = \frac{\sin \alpha_{02}}{\sin \alpha_{11}} L(p_{01}p_{02})$$



Under the assumption of non-degenerated triangles,

$$0 < a \leq \alpha_{01}, \alpha_{02}, \alpha_{11} \leq A < \pi$$

and (5.1), we have

$$L(p_{01}p_{11}) \leq \frac{1}{\sin a} M$$

Inductively, we conclude that the vertices of each of the triangles in the sequence $\{M_j\}$ are at a bounded distance from the origin. \square

From Lemma 3 one concludes that after suitable ordering, each sequence of vertices of the simplicial complexes $\{M_j\}$ is a bounded sequence and therefore contains a convergent subsequence. Using a diagonal process, one can select a convergent subsequence of discrete surfaces again denoted by $\{M_j\}$ by abuse of language. More precisely, this means each vertex of M_i converges to a limiting vertex which together form the limiting surface M_∞ .

In the next step, we shall show

Lemma 4 M_∞ is a discrete minimal surface.

Proof

First we define the Dirichlet energy function $F_i = E_D(f_i)$ and compute the

Hessian of F at a vertex p . Since

$$F_i = \frac{1}{2} \sum_j (\cot \alpha_j^i + \cot \beta_j^i) |p - q_j^i|^2$$

Differentiating (5.1) with respect to x, y and z , we can then have

$$(5.2a) \quad \frac{\partial F_i}{\partial x} = \sum_j (\cot \alpha_j^i + \cot \beta_j^i) \langle e_1, p - q_j^i \rangle$$

$$(5.2b) \quad \frac{\partial F_i}{\partial y} = \sum_j (\cot \alpha_j^i + \cot \beta_j^i) \langle e_2, p - q_j^i \rangle$$

$$(5.2c) \quad \frac{\partial F_i}{\partial z} = \sum_j (\cot \alpha_j^i + \cot \beta_j^i) \langle e_3, p - q_j^i \rangle$$

where e_1, e_2, e_3 are unit vectors in the x, y and z directions. If all the equations in (5.2) are differentiated with respect to x, y and z again, it is not difficult to find that

$$(5.3a) \quad \frac{\partial^2 F_i}{\partial x^2} = \frac{\partial^2 F_i}{\partial y^2} = \frac{\partial^2 F_i}{\partial z^2} = \sum_j (\cot \alpha_j^i + \cot \beta_j^i)$$

$$(5.3b) \quad \frac{\partial^2 F_i}{\partial x \partial y} = \frac{\partial^2 F_i}{\partial y \partial x} = \frac{\partial^2 F_i}{\partial x \partial z} = \frac{\partial^2 F_i}{\partial z \partial x} = \frac{\partial^2 F_i}{\partial y \partial z} = \frac{\partial^2 F_i}{\partial z \partial y} = 0$$

Recall $\nabla F_i|_{p_{i+1}} = 0$, which means that ∇F_i vanishes at p_{i+1} . Then,

$$(5.4) \quad \frac{\partial F_i}{\partial x} \Big|_{p_i} = \frac{\partial F_i}{\partial x} \Big|_{p_i} - \frac{\partial F_i}{\partial x} \Big|_{p_{i+1}}$$

Similarly, the forms of $\frac{\partial F_i}{\partial y} \Big|_{p_i}$ and $\frac{\partial F_i}{\partial z} \Big|_{p_i}$ can be easily observed.

Now, consider a function with a parameter t which is defined as

$$(5.5) \quad \varphi(t) = \frac{\partial F_i}{\partial x} \left((1-t)p_i + t p_{i+1} \right)$$

Note that $\varphi(0) = \frac{\partial F_i}{\partial x} \Big|_{p_i}$ and $\varphi(1) = \frac{\partial F_i}{\partial x} \Big|_{p_{i+1}}$.

Using the Mean Value Theorem, we obtain

$$\begin{aligned} \frac{\partial F_i}{\partial x} \Big|_{p_i} - \frac{\partial F_i}{\partial x} \Big|_{p_{i+1}} &= \varphi(0) - \varphi(1) \\ &= \frac{d\varphi}{dt} \Big|_{t=\xi_i} \quad \text{for some } \xi_i \in (0, 1) \\ &= \left[\sum_{k=1}^3 \frac{\partial^2 F_i}{\partial x_k \partial x_k} \frac{d \left((1-t)p_i^{(k)} + t p_{i+1}^{(k)} \right)}{dt} \right] \Big|_{t=\xi_i} \end{aligned}$$

where $x_k = x, y, z$ for $k = 1, 2, 3$ and $p_i^{(k)}$ is the k -th component of the vector p_i . After simplifying, the equation in (5.4) can be rewritten as

$$(5.6) \quad \frac{\partial F_i}{\partial x_j} \Big|_{p_i} = \sum_{k=1}^3 \left[\frac{\partial^2 F_i}{\partial x_k \partial x_j} \Big|_{\xi_i} \left(p_i^{(k)} - p_{i+1}^{(k)} \right) \right]$$

By using (5.3), we can simplify (5.6) as

$$(5.7) \quad \left. \frac{\partial F_i}{\partial x_j} \right|_{p_i} = \left. \frac{\partial^2 F_i}{\partial x_j^2} \right|_{\xi_i} \left(p_i^{(j)} - p_{i+1}^{(j)} \right)$$

Computing the norm of the vector on both sides of this equality, we have

$$\left\| \left. \frac{\partial F_i}{\partial x_j} \right|_{p_i} \right\| \leq \left\| \left. \frac{\partial^2 F_i}{\partial x_j^2} \right|_{\xi_i} \left(p_i^{(j)} - p_{i+1}^{(j)} \right) \right\| \leq \left\| \left. \frac{\partial^2 F_i}{\partial x_j^2} \right|_{\xi_i} \right\| \left\| p_i^{(j)} - p_{i+1}^{(j)} \right\|$$

for some $\xi_i \in (0,1)$. Summing up, we get

$$\left\| \nabla F_i \right|_{p_i} \left\| \leq \left\| \nabla^2 F_i \right|_{\eta_i} \left\| \left\| p_i - p_{i+1} \right\| \right.$$

for some $\eta_i \in (0,1)$, where $\nabla^2 F_i$ denotes the Hessian matrix of F_i .

In the last inequality, we can estimate $\left\| \nabla^2 F_i \right|_{\eta_i} \left\|$ from above by the maximum

of the entries of the matrix. Let s_{\max} denote this value, then we have

$$\left\| \nabla F_i \right|_{p_i} \left\| \leq s_{\max} \left\| p_i - p_{i+1} \right\|$$

Finally, recall that $\lim_{i \rightarrow \infty} p_i = p_\infty$ hence the right-hand-side of this inequality

converges to zero and we can conclude that

$$\nabla F_\infty \Big|_{p_\infty} = \lim_{i \rightarrow \infty} \nabla F_i \Big|_{p_i} = 0$$

Therefore F_∞ is a discrete minimal surface as claimed. \square

6. MATHEMATICA PACKAGE

In this section we describe the package which I built into Mathematica to compute general problems of discrete minimal surfaces for any boundary conditions provided. One of the most important features of Mathematica is that it is an extensible system, and it is always possible and convenient to add more functionality. In our case, there are two main parts, the program body and the package body. They are separated but dependent. The program body can explicitly tell Mathematica to read in the package we defined so as to carry out the standard procedure of minimization, data processing as well as graphical output. To deal with different minimization problems concerning discrete surfaces, we merely need to change the input data in the program body subject to both the initial and the boundary conditions. Before discussing the program body, the structure of the package is described first in the following.

Package - 01

```
BeginPackage["Msurface"];
  Begin["`Package`"];
    SubProgram 1;
    SubProgram 2;
    ....
    SubProgram N;
  End[]
EndPackage[]
```

Program heading and the name of the package is defined as "Msurface".

Subprogram is the segment of a program that can perform a specific function.

Package ending

Mathematica uses the notion of `BeginPackage["package-name"]` and `EndPackage[]` to indicate the beginning and ending of a package. The general structure of the Mathematica package is shown in Package-01.

The package itself is decomposed into several subprograms. Each of them may call other small subprograms or functions to complete the task, such as drawing graphics, computing, processing data, etc. We now take a closer look at the main subprogram called “`minimize[]`” whose structure is shown below.

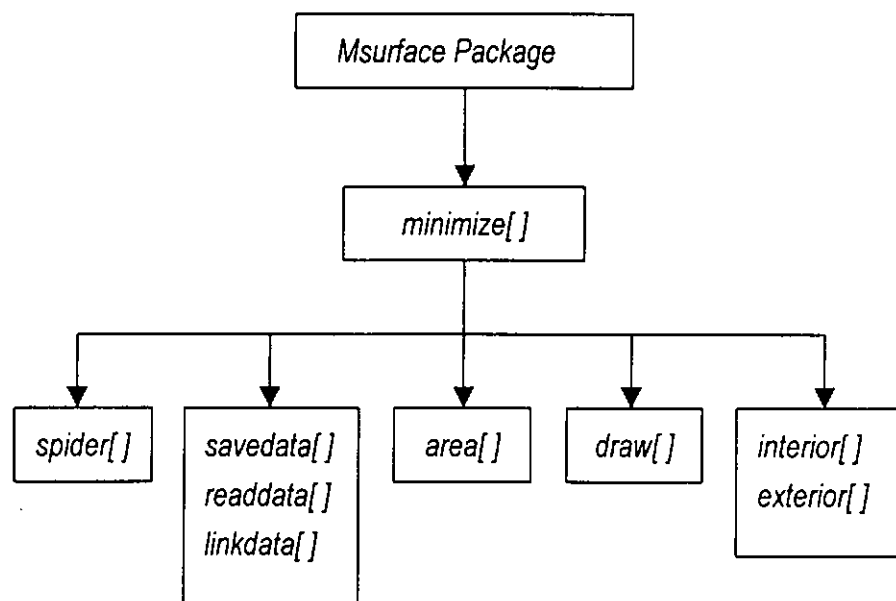


Figure 6.1 A top-down design

In `minimize[]`, user-defined functions and subprograms are contained to execute the standard process to minimize a discrete surface. The part of subprogram `minimize[]` is listed in Package-02, in which `minimize[]` creates an initial discrete surface, say M_0 , first by using another subprogram `spider[]` after a set of parameters (i.e. pts, guide, boundary boundpts and parameter) has been assigned.

Package = 02

```

minimize[ pts_, guide_, boundary_, boundpts_, parameter_ ]:=
Module[{ },
  { nr, tt, dt, opt }=parameter;
  { tol1,digit }={ 10^-4,10 };
  If [ tt==0, { t=tt, spider[ pts, guide ] } , {
    If [ TrueQ[ tt=="last" ],
      { { t }=ReadList[ "time.dat", Number ] , tt=t },
      t=tt ];
    linkdata[ p, readdata[ namefile[t] ] ]
  ]];
  olda=999;
  newa=area[p];
  draw[ Table[ Table[ Line[ { p[ v[m,nr,i] ],
    p[ v[m,nr,i+1] ] } ] , {i,0,w[nr]-1} ], {m,0,m-1} ] ];
  draw[ fdomain, opt ];
  Print[ "n", " ", "A(n)", " ", "A(n)-A(n-1)" ];
  Print[ "___", " ", "-----", " ", "-----" ];
  While[ TrueQ[ Abs[newa-olda] > tol ] , {
    Print[ t, " : ", newa, " ",
      If [ t==tt, " -", " ", Abs[newa-olda] ] ];
    interior[];
    exterior[];
    olda=newa;
    newa=area[p];
    t++;
    If [ Mod[ t, dt ]==0 , {
      savedata[t,namefile[t]];
      draw[fdomain,opt]; } ];
  ]];
];

```

minimize[] consists of different subprograms that can construct initial discrete surface, compute the area of each of the discrete surfaces, minimize the discrete surface area by moving points.

All the data about surfaces are saved in files called namefile[] and time.dat.

To identify each point p on the surface, we make use of a pointer $v[i,j,k]$ to state the coordinates of a point $p[v[i,j,k]]$. Hence we have to define the following functions.

Package :03

```

nr:=Length[ pts ];
w[ j_ ]:=If [ TrueQ[ j==nr ] , nr+1 , j ];
v[ m_,j_,i_ ]:=Module [ { mm=m , jj=j , ii=i } ,
  If [ TrueQ[ jj==0 ] , { mm=0 , ii=0 } , {
    While[ TrueQ[ ii<0 ] , { mm-=1 , ii+=w[ jj ] } ];
    While[ TrueQ[ ii>=w[ jj ] ] , { mm+=1 , ii-=w[ jj ] } ];
    While[ TrueQ[ mm<0 ] , mm+=m ];
    While[ TrueQ[ mm>=nr ] , mm-=m ];
  } ];
{ mm , jj , ii } ];

```

Rn is equal to the number of the boundary points given.

w[] determines how many points along each line of segment.

v[] denotes a pointer.

Next, during processing, we may have to store or recall data which can be done by calling the subprograms savedata[], readdata[] and linkdata[]. At each iteration step, we minimize the surface by moving the interior points first, and then the exterior points on the boundary. Such steps are performed in the subprograms of interior[] and exterior[]. The loop of minimization is executed repetitively until the termination criterion is fulfilled. In order to compare the difference of areas, we therefore define area[] to evaluate the total area of the discrete surface M_i . The figure of the minimized surface is finally implemented using the function of draw[]. All the subprograms in minimize[] mentioned above would be presented in more detail later.

To identify each point p on the surface, we make use of a pointer $v[i,j,k]$ to state the coordinates of a point $p[v[i,j,k]]$. Hence we have to define the following functions.

Package - 03.

```

n:=Length[ pts ];
w[ j_ ]:=If [ TrueQ[ j==nr ], nr+1 , j ];
v[ m_,j_,i_ ]:=Module [ { mm=m , jj=j , ii=i },
  If [ TrueQ[ jj==0 ], { mm=0 , ii=0 }, {
    While[ TrueQ[ ii<0 ], { mm-=1 , ii+=w[ jj ] } ];
    While[ TrueQ[ ii>=w[ jj ] ], { mm+=1 , ii-=w[ jj ] } ];
    While[ TrueQ[ mm<0 ], mm+=nr ];
    While[ TrueQ[ mm>=nr ], mm-=nr ];
  } ];
{ mm , jj , ii };

```

Rn is equal to the number of the boundary points given.

$w[j]$ determines how many points along each line of segment.

$v[]$ denotes a pointer.

Next, during processing, we may have to store or recall data which can be done by calling the subprograms `savedata[]`, `readdata[]` and `linkdata[]`. At each iteration step, we minimize the surface by moving the interior points first, and then the exterior points on the boundary. Such steps are performed in the subprograms of `interior[]` and `exterior[]`. The loop of minimization is executed repetitively until the termination criterion is fulfilled. In order to compare the difference of areas, we therefore define `area[]` to evaluate the total area of the discrete surface M_i . The figure of the minimized surface is finally implemented using the function of `draw[]`. All the subprograms in `minimize[]` mentioned above would be presented in more detail later.

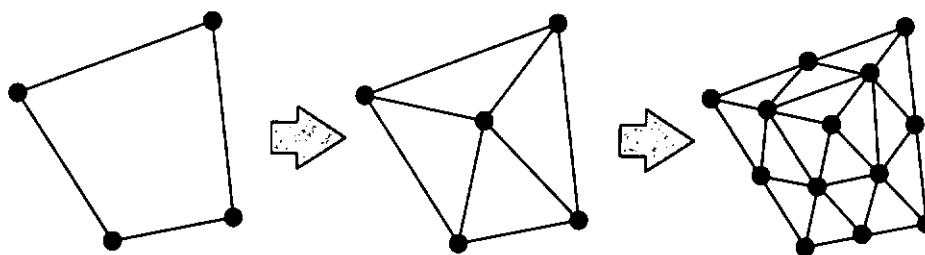


Figure 6.2 The working procedure in *spider*

Figure 6.2 shows a simple example how the triangulation of an initial domain is performed in the subprogram `spider[]`, in which pts contains p_0, p_1, \dots, p_N , the boundary points on the Jordan curve and $guide$ instructs how the boundary points connect together that may be either joining as a straight line or plotting according to some parametric mapping. Next, we determine a central point by averaging the weightings among the boundary points. After that, the function `ratio[]` is used to divide each line joined by two points.

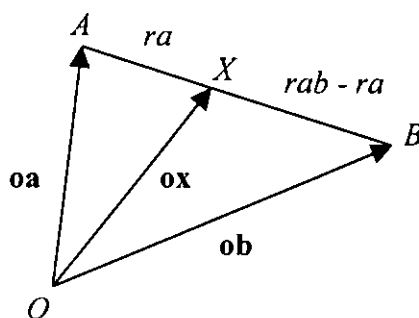


Figure 6.3 The vector ox is found by the function `ratio[]`

Package - 04

```

Spider[ pts_guide_ ]:= Module[ { },
  Do[ p[ v[m,nr,0] ] = N[ pts[[m+1]] ] , {m,0,nr-1} ];
  Do[
    If [ TrueQ[ guide[[m+1]] == "join" ] , {
      Do[ p[ v[m,nr,i] ] = ratio[ p[ v[m,nr,0] ] ,
        p[ v[m+1,nr,0] ] , i , w[nr] ] , {i,1,w[nr]-1} ];
    } , {
      ds=( #3 - #2 ) / w[nr] &@@ guide[[m+1,2]];
      Do[ p[ v[m,nr,i] ] = N[ #1/.#2[[1]] -> (#2[[2]]+i*ds) ]
        &@@ guide[[m+1]] , {i,1,w[nr]-1} ];
    } ];
  ,{m,0,nr-1});
  p[ v[0,0,0] ] = Sum[ p[ v[m,nr,i] ] , {m,0,nr-1}
    , {i,0,w[nr]-1} ] / (m*(nr+1));
  Do[ p[ v[m,j,0] ] = ratio[ p[ v[0,0,0] ] , p[ v[m,nr,0] ] , j
    , nr ] , {j,1,nr-1} , {m,0,nr-1} ];
  Do[ p[ v[m,j,i] ] = ratio[ p[ v[m,j,0] ] , p[ v[m+1,j,0] ] , i
    , w[j] ] , {j,2,nr-1} , {m,0,nr-1} , {i,0,w[j]-1} ];
];
ratio[ oa_,ob_,ra_,rab_ ]:=N[ ( (rab-ra)*oa + ra*ob ) / rab ];

```

guide indicates how the points pts join together by either a straight line or a parametric mapping.

p[v[0,0,0]] indicates the central point among the boundary points.

ratio[] is used to divided each line by ratio.

For convenience, it is better to save the data about the points on each discrete surface during minimization. As a result, we define the functions savedata[], readdata[] and linkdata[] so as to save data in files named by the function namefile[] and to read data from files if necessary.

Package 05

```

data[ x_ ]:=Insert[
    Flatten[
        Table[ x[v[m,j,i]] , {j,1,nr}, {m,0,m-1}, {i,0,w[j]-1} ]
    ,2]
    ,x[v[0,0,0]], 1 ];

```

```

Change[ t_ ]:= Module[ { tt, st },
    If[ t >= 1000, tt = Mod[t,1000], tt = t ];
    ft = ToString[ tt ];
    st = StringLength[ ft ];
    Do[ ft = "0"<>ft, {3-st} ];
    ft];

```

```

namefile[ t_ ]:= Module[ { ts },
    ts = ToStringQuotient[ t,1000 ];
    name = datafile<>ts<>". "<>Change[ t ];
    name];

```

```

savedata[ t_, file_ ]:= Module[ { },
    SetDirectory[ root ];
    WriteString[ "time.dat", t ];
    Close[ "time.dat" ];
    Do[ WriteString[ file,
        FortranForm[ SetPrecision[#1,digit] ], " ",
        FortranForm[ SetPrecision[#2,digit] ], " ",
        FortranForm[ SetPrecision[#3,digit] ], "\n"
    ]&@@@data[p][[m]]
    ], {m,1,Length[ data[p] ] };
    Close[file];
];

```

```

readdata[ file_ ]:= Module[ { },
    SetDirectory[ root ];
    outdata = Partition[ ReadList[ file, Real ], 3 ];
    outdata];

```

```

linkdata[ x_, indata_ ]:= Module[ { k=1 },
    x[v[0,0,0]] = indata[[k]];
    Do[ { k++;
        x[v[m,j,i]] = indata[[k]]
    }, {j,1,nr}, {m,0,m-1}, {i,0,w[j]-1} ];
];

```

data[] groups all pointers

Change[] converts the number t into a string as a part of the name of the data file.

namefile[] specify a name of file to store data at the t-th iteration.

savedata[] is to write the coordinates of all points in the specified file.

readdata[] is used to recall the saved data.

linkdata[] transforms the format of the data read.

Besides processing data, we also have to draw out the graphic of the output. Hence the function `draw[]` is defined to draw after a figure and options are provided. The triangulation of any discrete surfaces can be described in `polys[]` as a set of vertices of triangles, and the figure is represented in `picture[]` using `.` The program is listed below (see [Package-06] .)

Package 06

```
Polys[ x_ ]:= Partition[
  Partition[
    Flatten[ Join[
      Table[ { x[v[m,j,i]], x[v[m,j-1,i]], x[v[m,j-1,i-1]] }
        ,{j,2,nr-1}, {m,0,m-1}, {i,1,w[j]-1} ] ,
      Table[ { x[v[m,j,i]], x[v[m,j+1,i]], x[v[m,j+1,i+1]] }
        ,{j,0,nr-2}, {m,0,m-1}, {i,0,w[j]} ] ,
      Table[ { x[v[m,nr,i]], x[v[m,nr,i+1]], x[v[m,nr-1,i-1]] }
        ,{m,0,m-1}, {i,1,w[nr]-1} ] ,
      Table[ { x[v[m,nr-1,i]], x[v[m,nr-1,i+1]], x[v[m,nr,i+2]] }
        ,{m,0,m-1}, {i,0,w[nr-1]-1} ] ,
      Table[ { x[v[m,nr,0]], x[v[m,nr-1,0]], x[v[m,nr,1]] }
        ,{m,0,m-1} ]
    ] ]
  ,3,3]
,3];
```

```
picture[ x_ ]:= Polygon[ # ] &/@polys[ x ];
```

```
draw[ fig_ , opts_ ]:= Show[ Graphics3D[ fig ], opts ];
```

polys[] contains all the triangles formed on a surface.

picture[] formed a set of polygons or triangles if polys[] has found.

draw[] shows the graphical output under the opinions provided.

In general, it is necessary to measure the area and the angles of the triangles of a discrete surface in each iteration for the purpose of checking the termination criterion and performing the minimization process respectively. The functions `triangle[]` and `angle[]` can measure the area of $\triangle ABC$ and $\angle ABC$ respectively if the position vectors of OA , OB and OC are provided. And then the total area of triangles described in `poly[]` can be computed by the function `area[]`.

Package - 07

```
triangle[ oa_, ob_, oc_ ]:= Module[ { },
  { ab, ac } = { ob-oa, oc-oa };
  AppendTo[ ab, ab[[1]] ];
  AppendTo[ ac, ac[[1]] ];
  da = Sqrt[
    Sum[(ab[[m]]*ac[[m+1]]-ab[[m+1]]*ac[[m]])^2
      ,{m,1,3} ]
    ] / 2. ;
  da ];

area[ x_ ]:= Plus@@( triangle[ #[[1]], #[[2]], #[[3]] ] &/@polys[x] );

angle[ oa_, ob_, oc_ ]:= Module[ { },
  { ba, bc } = { oa-ob, oc-ob };
  babc = Sqrt[ (ba.ba)*(bc.bc) ];
  If[ TrueQ[babc<tol1] ,
    ang=0 ,
    ang=Re[ ArcCos[(ba.bc)/babc] ]
  ];
  ang ];
```

triangle[] computes the area of a triangle only.

area[] sums up the areas calculated by function triangle[].

angle[] can measure a angle in a triangle.

After having introduced some components in the package, the most important parts of the package will be discussed now. In order to minimize the area, the condition (3.6) for all interior points is used. For each interior point, say p_{ij} , its corresponding neighborhood (refer to Figure 3.3) is grouped and called `allv` in the program. Then the new location of a point p_{ij} is found in the subprogram `iteration[]` with respect to its neighborhood under condition (3.6).

Package 08:

```

allv:= { p[ v[m,j+1,i] ], p[ v[m,j+1,i+1] ],
         p[ v[m,j+1,i+2] ], p[ v[m,j,i+1] ],
         p[ v[m,j-1,i] ], p[ v[m,j-1,i-1] ],
         p[ v[m,j,i-1] ], p[ v[m,j+1,i-1] ]
};

iteration[ vp_, vlist_ ]:= Module [ { upp={0,0,0} , low=0 },
nv = Length[ vlist ];
vs = Insert[ Insert[ vlist, vlist[[1] ] , -1 ] , vlist[[nv] ] , 1 ];
Do[ {
    ang1 = angle[ vp, vs[[k-1] ] , vs[[k] ] ];
    If[ TrueQ[ ang1<tol1 || N[Abs[ang1-Pi]]<tol1 ] ,
        co1=0. ,
        co1 = N[ Cot[ ang1 ] ]
    ];
    ang2 = angle[ vp, vs[[k+1] ] , vs[[k] ] ];
    If[ TrueQ[ ang2<tol1 || N[Abs[ang2-Pi]]<tol1 ] ,
        co2=0. ,
        co2 = N [ Cot[ ang2 ] ]
    ];
    upp += (co1+co2)*vs[[k]];
    low += (co1+co2);
}, {k,2,nv+1} ];

If[ TrueQ[ low==0. ], outp = vp , outp = upp/low ];
outp];

```

the neighborhood of a point is defined as allv.

iteration[] can move the point vp to a new position such that the area of the discrete surface formed by its neighborhood vlist is reduced.

Note that the subprogram `iteration[]` moves only one point at each time. For all interior points on the discrete surface, `interior[]` is used to call `iteration[]` at each point until every interior point has moved to the position such that the area is minimized. (See Package-09)

Package-09

```
interior[]:= Module[ { },
  Do[ {
    If[ TrueQ[ j==nr-1 ], {
  If[ TrueQ[ i==0 ],
    vlist = Drop[ allv,{6} ],
    vlist = Drop[ Drop[ allv,{8} ], {1} ]
      ];
    },{
    If[ TrueQ[ i==0 ],
    vlist = Drop[ Drop[ allv,{6} ], {3} ],
    vlist = Drop[ Drop[ allv,{8} ], {3} ]
      ];
    }];
  p[v[m,j,i]] = iteration[ p[v[m,j,i]] , vlist ];
}, {j,nr-1,1,-1}, {m,0,m-1}, {i,0,w[j]-1} ];

vlist = Table[ p[v[m,1,0]] , {m,0,m-1} ];
p[v[0,0,0]] = iteration[ p[v[0,0,0]] , vlist ];
];
```

every interior point moves in interior[] by calling subprogram iteration[].

After interior minimization, we have to move those points on the boundary to minimize the area. If the boundary is fixed, no exterior points on that boundary move. For partially free boundary or free boundary, additional condition is needed to restrict the movement of the boundary points. The restriction may be that the points move either along a line or on a plane. To determine the moving exterior points, the functions `onsurface[]` and

atline[] are written to find out those points on the surface or along the line such that the areas of the triangles formed on the outermost layer are the least.

Package: 10

```

onsurface[ pt_, fct_ ]:= Module[ { },
Clear[ x, y, z, k ];
df = D[ fct, # ] &/@ { x, y, z };
lhs = Append[ 2*{ x, y, z }-pt, fct ];
rhs = Append[ k*df, 0 ];
soln = Solve[
      Table[ lhs[[m]] == rhs[[m]]
            ,{m,1,4}
            ,{x, y, z, k} ];
outp = Flatten[ {x, y, z}/.soln ];
outp];

atline[ pt_, fct_ ]:= Module[ { },
Clear[ x, y, z, k, s ];
df1 = D[ fct[[1]], # ] &/@ {x, y, z};
df2 = D[ fct[[2]], # ] &/@ {x, y, z};
lhs = Append[
      Append[ 2*{x,y,z}-pt, fct[[1]] ]
      , fct[[2]] ];
rhs = Append[
      Append[ k*df1+s*df2, 0 ]
      , 0 ];
soln = Solve[
      Table[ lhs[[m]] == rhs[[m]]
            ,{m,1,5} ]
      ,{x,y,z,k,s}];
outp = Flatten[ {x,y,z}/.soln ];
outp];

```

To vary all the free exterior points on planes or along lines, the subprogram exterior[] can call atline[] and onsurface[] subject to the boundary condition.

Package - 11:

```

exterior[]:=Module[{}
Do[If[TrueQ[boundary[[m+1]]=="fix"],,{
  Do[{p[v[m,nr,i]]=onsurface[
    p[v[m,nr-1,i-1]],boundary[[m+1]]]
    },{i,1,w[nr]-1}];
}];
If[TrueQ[boundpts[[m+1]]=="fix"],,{
  If[TrueQ[Length[boundpts[[m+1]]]==1],{
    p[v[m,nr,0]]=onsurface[
    p[v[m,nr-1,0]],boundpts[[m+1,1]]]
    },{p[v[m,nr,0]]=atline[p[v[m,nr-1,0]],boundpts[[m+1]]]
    }];
}];
},{m,0,m-1}];
];

```

Sometimes we can make use of reflections as well as rotations to construct a complete discrete minimal surface from a fundamental domain. In some cases, the subprograms `reflection[]` and `rotation[]` are applied by calling the functions `reflect[]` and `rotate[]` after minimizing a discrete fundamental piece.

Package - 12:

```

reflect[pt_,fct_]:=2*onsurface[pt,fct]-pt;
reflection[x_,fct_]:=Module[{}
x[v[0,0,0]]=reflect[x[v[0,0,0]],fct];
Do[ x[v[m,j,i]]=reflect[x[v[m,j,i]],fct]
,{j,1,nr},{m,0,m-1},{i,0,w[j]-1}];
];

rotate[pt_,ang_]:= { Cos[ang],-Sin[ang],0},
{ Sin[ang],Cos[ang],0},
{ 0,0,1 } .pt;

rotation[x_,ang_]:=Module[{}
x[v[0,0,0]]=rotate[x[v[0,0,0]],ang];
Do[ x[v[m,j,i]]=rotate[x[v[m,j,i]],ang]
,{j,1,nr},{m,0,m-1},{i,0,w[j]-1}];
];

```

7. GRAPHICS

In this section, some of the result of the discrete minimal surfaces minimized by the package described in Section 6 are shown. In which, there are both cases of surfaces with fixed boundary and partially free boundary. Also, the reflection of fundamental domain is used in some problems to construct a complete minimal surface. All the computer graphics in this section are drawn by *Mathematica*.

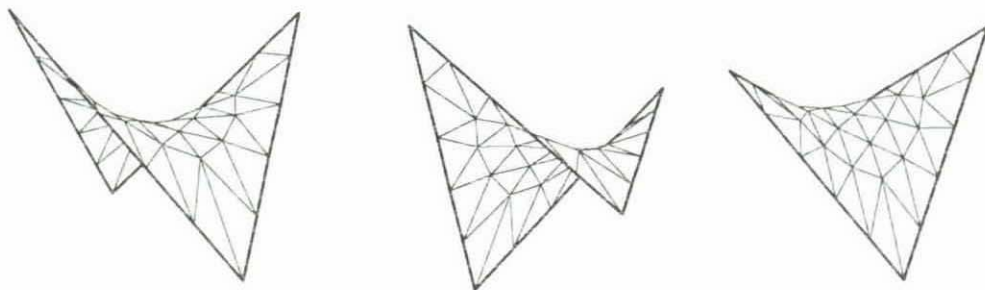


Figure 7.1 Schwarz's Surface from different point of view

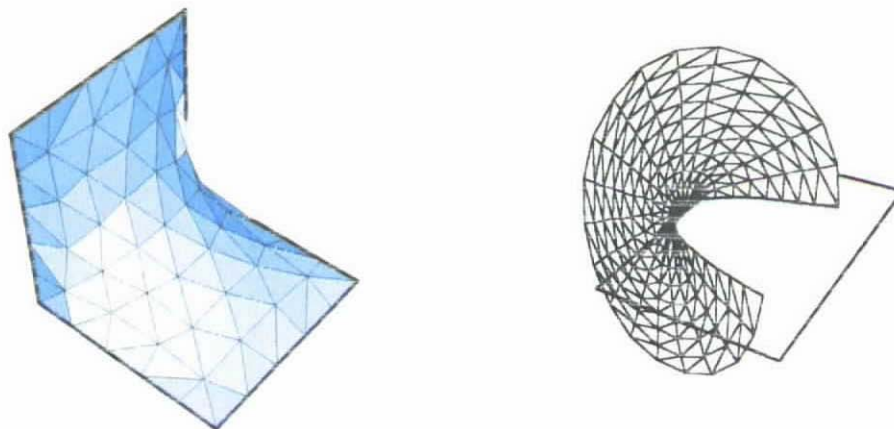


Figure 7.2 A case of fixed boundary problem (left) and a case of partially free boundary problem (right)

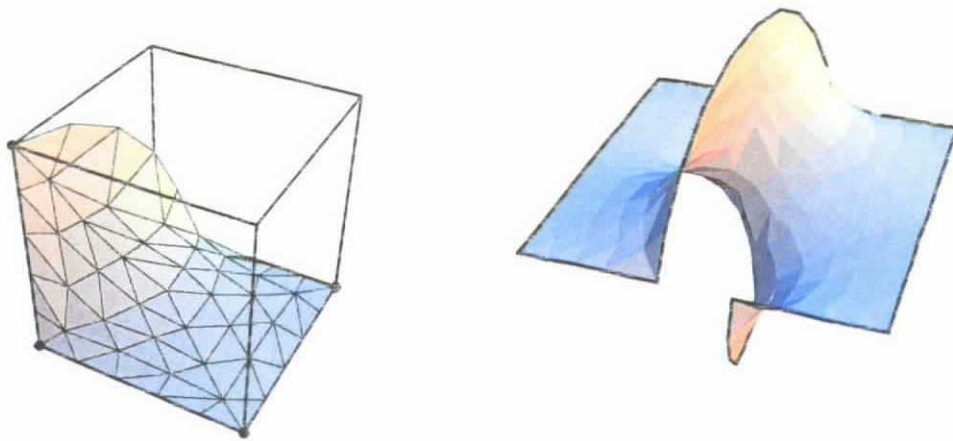


Figure 7.3 The fundamental domain (left) of the twisted rectangular strip surface (right). The points on the frame marked the initial position of the boundary points.

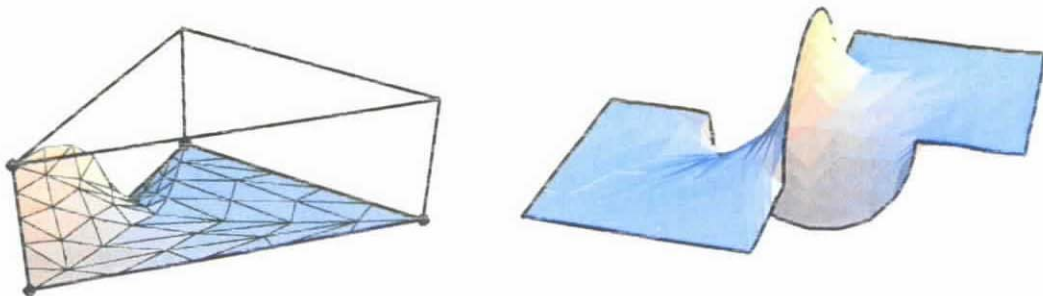


Figure 7.4 The fundamental domain (left) of double twisted rectangular strip surface (right).



Figure 7.5 The fundamental domain (left) of the Jorje Meeks surface (right).

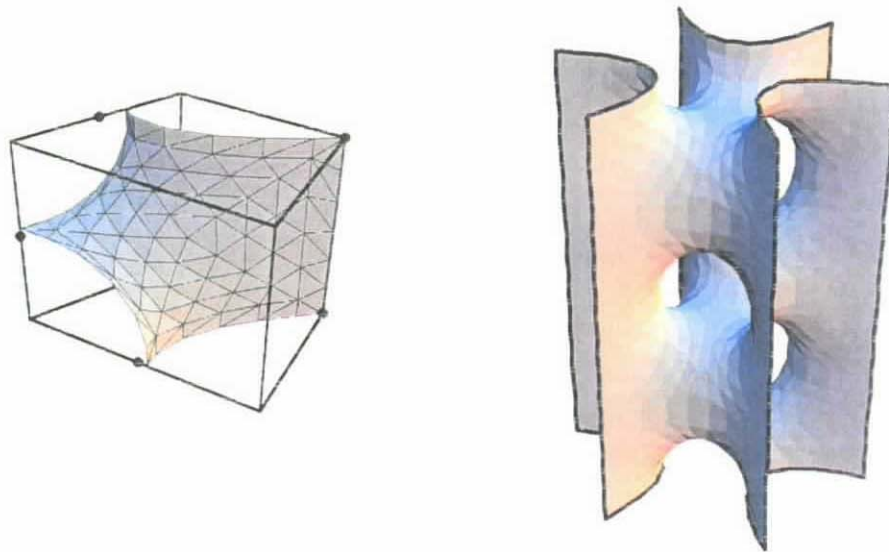


Figure 7.6 The fundamental domain (left) of the Scherk's tower (right).

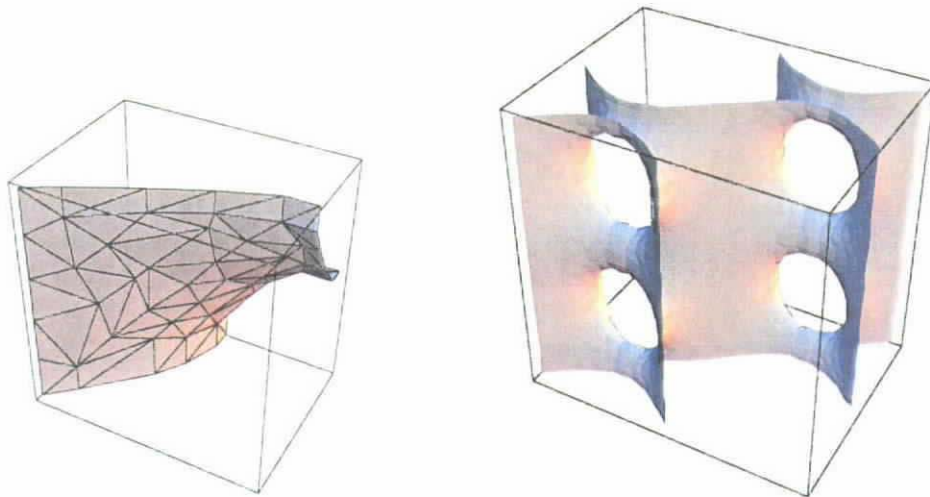


Figure 7.7 The fundamental domain (left) of the Schwarz's CLP-surface (right).

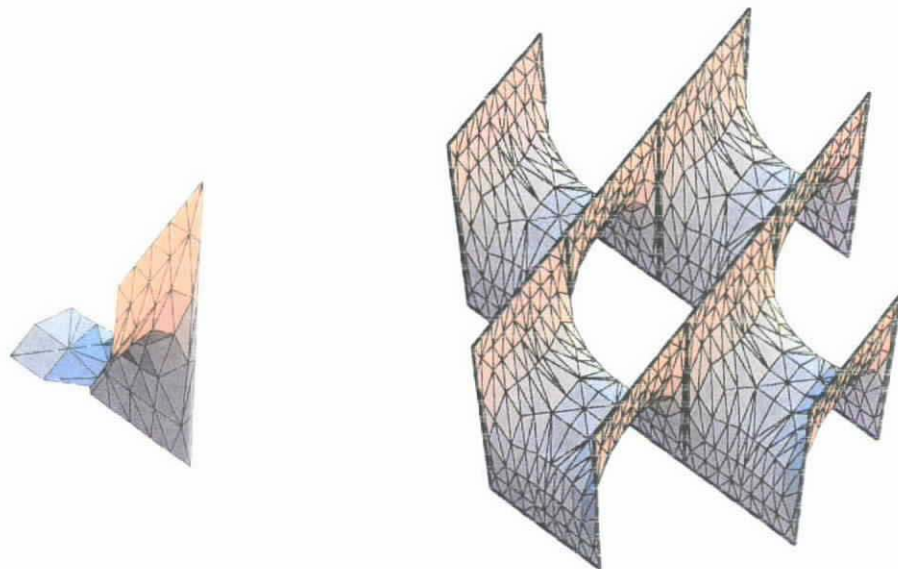


Figure 7.8 The fundamental domain (left) of a Scherk's first surface and a Scherk's surface can be built through repeated reflection in edges (right).

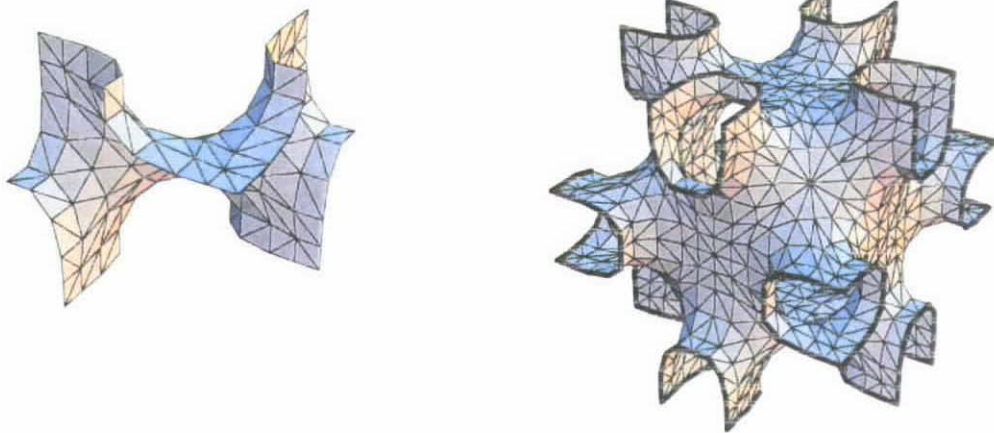


Figure 7.9 The fundamental domain (left) of the Neovius surface (right).

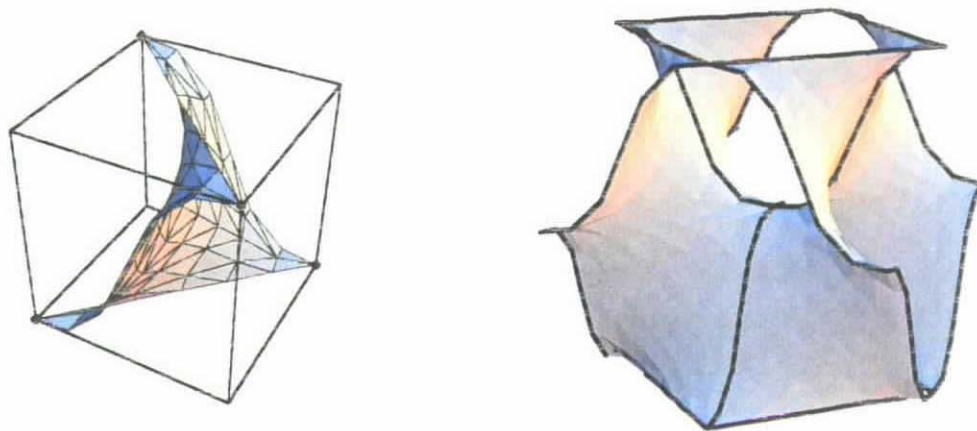


Figure 7.10 The Gergonne's surface (left) generates a periodic minimal surface (right).

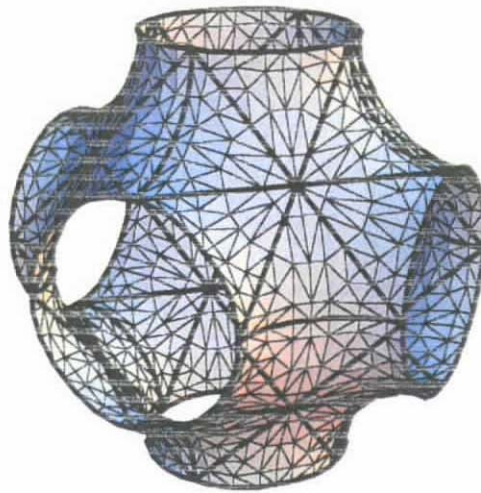


Figure 7.11a A part of Schwarz's P-surface is formed by 48 fundamental pieces after reflecting.

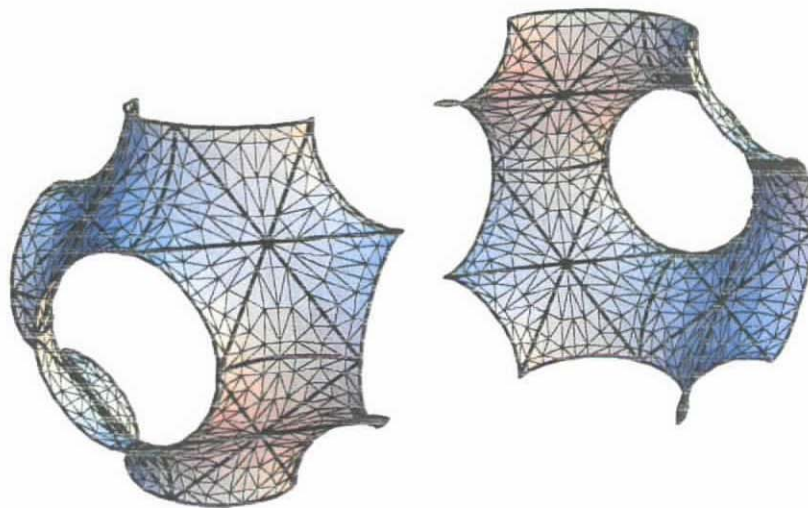


Figure 7.11b The cross section of Figure 7.11a

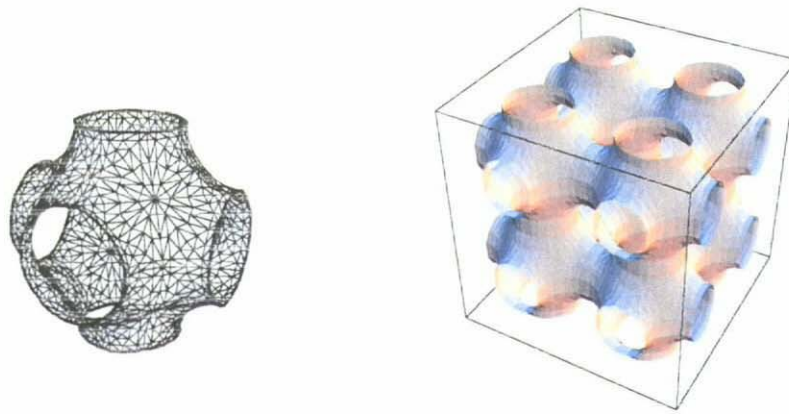


Figure 7.12 A fundamental cell (left) of the Schwarz P-surface (right).

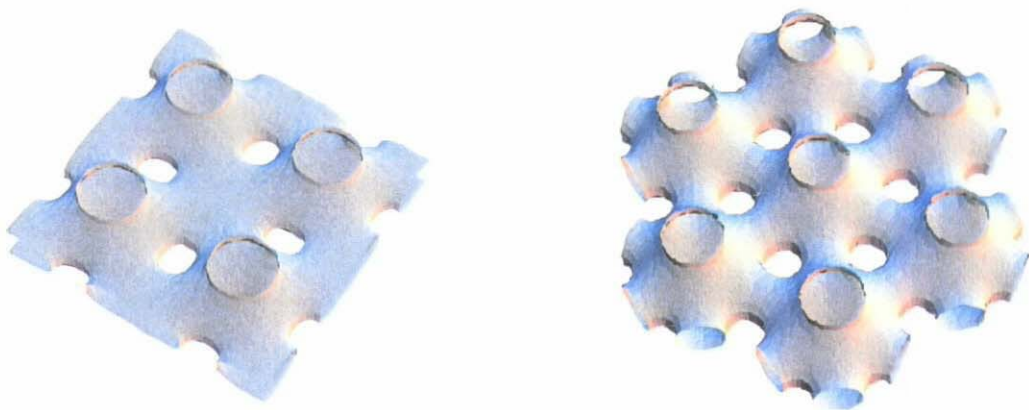


Figure 7.13 Schoen's S'-S''-surface (left) and one layer of dual lattice (right).

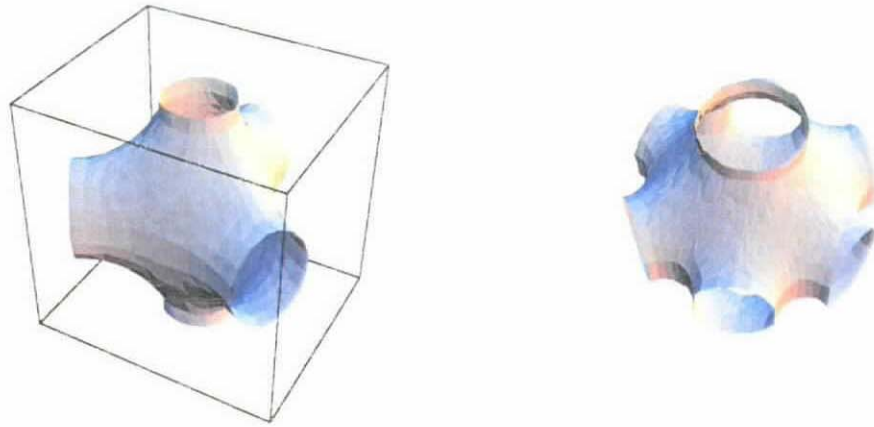


Figure 7.14 Trigonon fundamental cell (left) and hexagonal fundamental cell (right)

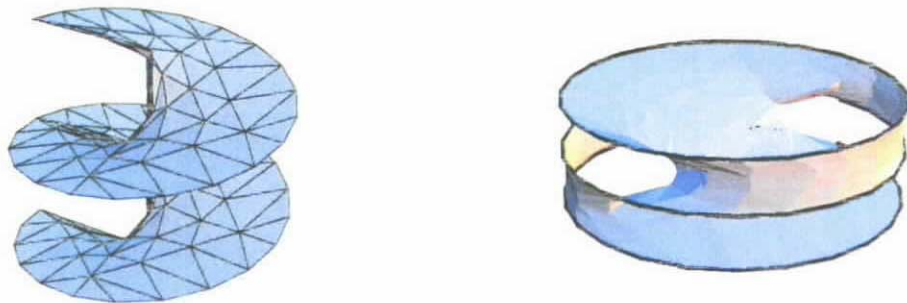


Figure 7.15 Part of a helicoid (left) and a rotationally symmetric configuration consisting of three coaxial circles (right).

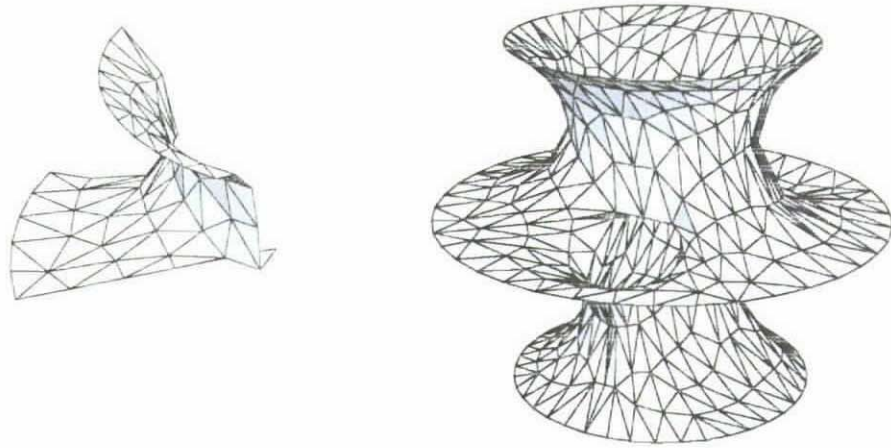


Figure 7.16 The fundamental domain (left) of the Costa surface (right).

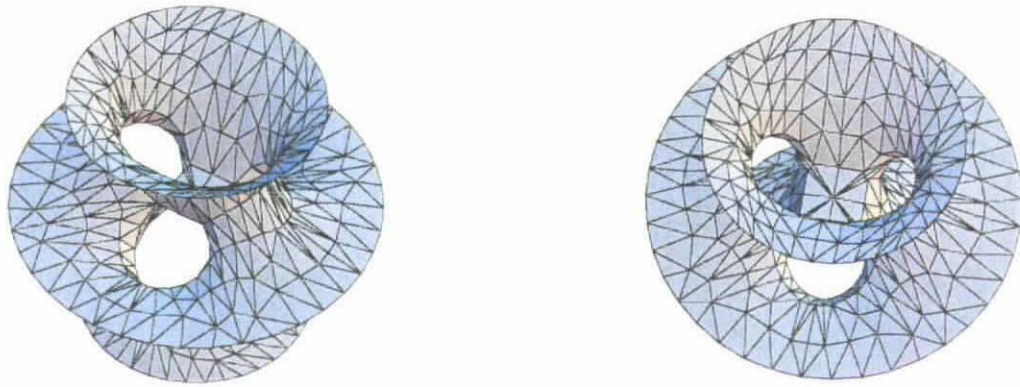


Figure 7.17 Different views of the costa surface.

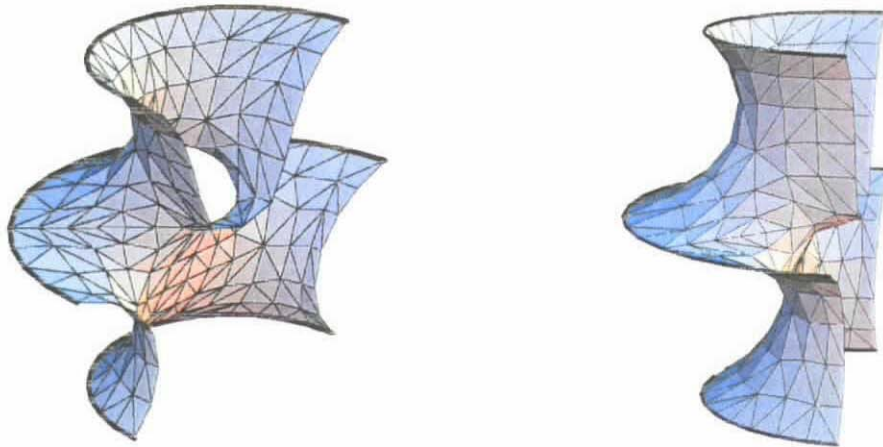


Figure 7.18 The left side of the costa surface.

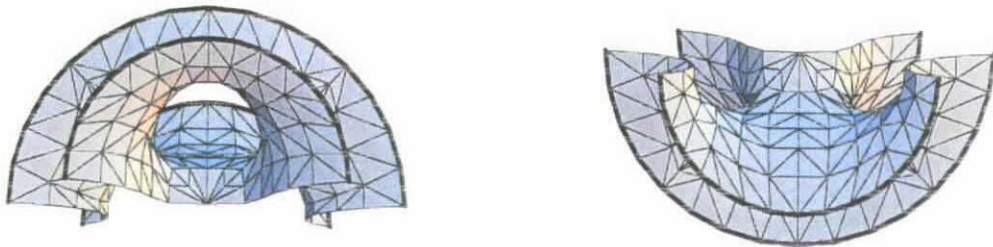


Figure 7.19 The top and bottom views of Figure 7.18.

8. CONCLUSION

In general, there are two approaches leading to the construction of minimal surfaces in the three-dimensional Euclidean space. The first approach is based on the Weierstrass representation formulas, as explained in Dierkes et al [1]. This approach leads to stationary minimal surfaces (i.e. they are stationary points of the area functional and may not be minimizing). The second approach relies on minimization of area in certain class of objects. Although the first approach is elegant, it is often a tedious if not impossible task to find complicated or periodic minimal surfaces this way, as it is not so easy to find the appropriate Weierstrass representation formulas or to computer them. In this paper, we studied a new method based on discrete techniques proposed by Pinkall and Polthier [4]. This is a direct minimization approach for which the objects in consideration are (non-smooth) surfaces. Indeed, they are all specific collections of triangles spanned by a set of vertices on a surface and have polygonal contours. In this thesis, I have made an in-depth study and improvements of their minimization algorithm. This includes convergence proof as well as programming and implementation of the algorithm in the Mathematica language. It also includes further refinements of the algorithm to include the case of minimal surfaces with partially free boundaries. From this study, I discovered the following interesting facts :

1. The algorithm can be simplified by minimizing one interior vertex each time,
2. The algorithm produces a unique minimal surface after the whole iteration procedure, because of the fact that a quadratic function is being minimized each time,

3. By suitable adaptations, the algorithm can be made to handle the case of partially free boundary problems.

In the partially free boundary case, there are two ways to model the minimal surfaces obtained.

- (A) Using the fact that minimal surfaces are perpendicular to the supporting surfaces, we can require the first layer of interior points (i.e. the layer immediately above the supporting surface) to be perpendicular to the supporting surface. More precisely, we minimize the interior points one after another. Subsequently, we minimize this layer of points successively requiring that they are perpendicular to the supporting surface.
- (B) We minimize the interior points successively, with the additional condition requiring the points on the layer immediately next to the supporting surface to satisfy the constraints.

For method (A), although the algorithm seems to model the phenomenon, the convergence proof is not clear. The major difficulty lies in the fact that in the discrete case, a minimizing surface may not be perpendicular to the supporting surface. For method (B), the convergence should follow from arguments along the same line as in the proof for the fixed boundary case. This is because the set of all interior points lie in a bounded set as before. To see this, one needs to observe that the proof of Lemma 3 in P-5.5 requires only one fixed line segment. Having such a segment, one can estimate the positions of all other line segments successively using the Sine law.

REFERENCES

- [1] U. Dierkes, S. Hildebrandt, A. Küster and O. Wohlrab, "Minimal Surfaces", Grundlehren der mathematischen Wissenschaften, 295-296, Springer Verlag, 1992.
- [2] J. Douglas, "Solution of the problem of Plateau", Trans. Amer. Math. Soc. 33, 263-321, (1931).
- [3] S. Hildebrandt, A. Tromba, "The parsimonious universe : shape and form in the natural world", New York, Copernicus, 1996.
- [4] U. Pinkall and Polthier, "Computing discrete minimal surfaces and their conjugates", Experimental Mathematics, Vol. 2, No.1, (1993).
- [5] J. Plateau, "Sur les figures d'équilibre d'une masse liquide sans pesanteur", Mém. Acad. Roy. Belgique, New Series, Vol. 23, (1849).