The Mathematica® Journal

Visualizing Minimal Surfaces

Rendering Solid Models with the Aid of 3D Printers

O. Michael Melko

3D printers are a potentially useful tool for geometric visualization in mathematical research and education. In this article, we describe the mathematics of minimal surfaces in some detail, and then we present a *Mathematica* package for generating solid model data of such surfaces. In particular, we show how the package was used to generate a 3D model of Costa's surface.

Introduction

In [1], Palais describes a program for the visualization of mathematics through the use of computer graphics. He notes that visualization has been instrumental in some important mathematical discoveries and is also useful for educational purposes. With this in mind, he proposes the creation of an online interactive gallery of mathematical visualization, which he calls a "mathematical exploratorium." As helpful as computer graphics are for visualization, it can be argued that there is an additional benefit to be had in viewing and handling an actual physical object. Indeed, there has been a long-standing tradition at German universities of producing plaster models of interesting geometric objects. Nowadays, physical models can be easily created by means of stereolithography, or "3D printing technology."

3D printers produce solid objects from appropriate input data. They were originally created for rapid prototyping of new product designs but are increasingly being used for other purposes, such as highly customized manufacturing and scientific visualization. Their applications will continue to grow as the underlying technology improves and decreases in cost. One type of 3D printer uses a powder-binder technology to create objects via a layering technique: a thin layer of powder is spread across a planar surface, and then a print head applies a binder within the cross-sectional area of the object being created. This process is repeated, adding layer upon layer, until the object is complete.

As input, 3D printers require data specifying the vertices, polygons, and normals of the object to be rendered. If the object is to be colored, the colors of the polygons must also be provided. Various file formats may be used to store this data, including the *Polygon File Format* (or *PLY*), which is also known as the *Stanford Triangle Format*. This is the file format used to render Costa's surface and is described later.

We generate point (or vertex) data for Costa's surface by means of its Weierstrass representation. Loosely speaking, the Weierstrass representation provides a recipe for creating a parametrization of a minimal surface in $\mathbb{R}^3$ from two meromorphic functions defined on a Riemann surface. In particular, when these functions are $\wp$ and $C/\wp'$, where $\wp$ is the Weierstrass $\wp$ function and C is a certain constant, the underlying Riemann surface is a torus of the form $\mathbb{C}/\Gamma$, where Γ is a discrete lattice in the complex plane $\mathbb{C}$. The resulting minimal surface has often been rendered in 3D graphics images. To produce the coordinate data for this surface, we must integrate certain rational functions of $\wp$ and $C/\wp'$. This results in coordinate functions that are expressed in terms of the Weierstrass $\wp$ and ζ functions. Since these functions are built into *Mathematica*, it is easy to generate the required data.

To generate a solid model, we must produce vertex and face data for a polyhedron that bounds a volume in $\mathbb{R}^3$. To achieve this, we first choose a proper subregion of a fundamental domain of the functions $\wp$ and ζ that contains no poles. This ensures that the corresponding piece of Costa's surface is of finite extent. We then generate two surfaces by means of normal displacement and "glue" the resulting boundaries together. This data is then exported to a PLY file, which is used to print the model.

In what follows, we first provide some background from minimal surface theory. This includes a review of classical surface theory, a description of the Weierstrass representation, a summary of pertinent facts about elliptic functions, and a description of the parametrization of Costa's surface that we use to generate model data. This is followed by a description of the *Minimal Surfaces* package developed to render the model data. Then we illustrate how to use the functionality provided by this package to create both graphics objects and PLY data. Finally, discuss some ways in which the work in this paper might be extended, including ideas for mathematical experimentation and enhancements to the *Minimal Surfaces* package.

■ A Thumbnail Sketch of Minimal Surface Theory

Intuitively, a smooth surface $\mathcal{S}$ in Euclidean space $\mathbb{R}^3$ is *locally area minimizing* if any small deformation of $\mathcal{S}$ results in a surface of larger area. (The precise mathematical definition of minimal surface requires the introduction of some technical preliminaries and is given later.) Soap films spanning a curve in $\mathbb{R}^3$, for example, satisfy this property. In general, $\mathcal{S}$ may have self-intersections, in which case the surface is said to be *immersed*, otherwise, we say that it is *embedded*. The purpose of this section is to describe the *Weierstrass representation for minimal surfaces,* which provides a way of explicitly constructing a large class of such surfaces via complex function theory. As an example, we shall discuss Costa's minimal surface in some detail. We begin by summarizing essential facts from classical surface theory; further details may be found in [2].

□ Essential Facts from Classical Surface Theory

We *parametrize* a smooth surface $\mathcal{S}$ in $\mathbb{R}^3$ by means of a map $X : \Omega \to \mathcal{S}$, where Ω is an open connected subset (or *region*) in $\mathbb{R}^2$, and we assume that $X(\Omega) = \mathcal{S}$. We refer to $\mathcal{S}$ as the *trace* of X. Furthermore, we use (u, v) to denote a system of coordinates on Ω, and we assume that X is *smooth*, i.e., differentiable to arbitrary order with respect to u and v. We also assume that X is *regular*, i.e., that the tangent vectors $\partial X / \partial u$ and $\partial X / \partial v$ are linearly independent for every point $(u, v) \in \Omega$.

Let $\langle\, , \rangle$ denote the standard inner product on $\mathbb{R}^3$, and fix an orientation. Then the *metric*, or *first fundamental form,* on $\mathcal{S}$ with respect to X is given by the symmetric tensor

$$\mathbf{I} := ds^2 = E\, du^2 + 2\, F\, du\, dv + G\, dv^2,$$

where the coefficients $E, F,$ and G are functions on Ω given by

$$E := \left\langle \frac{\partial X}{\partial u}, \frac{\partial X}{\partial u} \right\rangle, F := \left\langle \frac{\partial X}{\partial u}, \frac{\partial X}{\partial v} \right\rangle, G := \left\langle \frac{\partial X}{\partial v}, \frac{\partial X}{\partial v} \right\rangle.$$

Let S^2 denote the sphere of unit radius in $\mathbb{R}^3$ centered at the origin, and let $Z(u, v)$ be the unit normal vector at the point $X(u, v)$ in $\mathcal{S}$ that is consistent with the chosen orientation of $\mathbb{R}^3$. We use $\hat{Z}(u, v)$ to denote the unique element of S^2 that is parallel to $Z(u, v)$. The map $\hat{Z} : \Omega \subseteq \mathbb{R}^2 \to S^2 \subseteq \mathbb{R}^3$ is called the *Gauss map*—it provides a measure of how the surface bends in its ambient space. The Gauss map is used to define the *second fundamental form,*

$$\mathbf{II} := L\, du^2 + 2\, M\, du\, dv + N\, dv^2,$$

where the coefficients $L, M,$ and N are again functions on Ω given by

$$L := -\left\langle \frac{\partial \hat{Z}}{\partial u}, \frac{\partial X}{\partial u} \right\rangle,\ M := -\left\langle \frac{\partial \hat{Z}}{\partial u}, \frac{\partial X}{\partial v} \right\rangle,\ N := -\left\langle \frac{\partial \hat{Z}}{\partial v}, \frac{\partial X}{\partial v} \right\rangle.$$

We identify the first and second fundamental forms with the 2×2 symmetric matrices that define them:

$$\mathbf{I} = \begin{pmatrix} E & F \\ F & G \end{pmatrix}, \mathbf{II} = \begin{pmatrix} L & M \\ M & N \end{pmatrix}.$$

The forms **I** and **II** encode *intrinsic* and *extrinsic* geometric properties of the surface $\mathcal{S}$. Intrinsic properties are those that are derived from the presence of a distance measure on $\mathcal{S}$ and do not change under *isometric* (or distance-preserving) deformations in the ambient space $\mathbb{R}^3$. Extrinsic properties are those that depend on how the surface is immersed into $\mathbb{R}^3$. (As a motivating example, consider wrapping a geographical map into a tube: distances between points within the map do not change, but the way they lie in space does.) A fundamental fact from classical surface theory is that, up to rigid motion, the forms **I** and **II** completely determine the geometry of $\mathcal{S}$ and how it lies in $\mathbb{R}^3$.

The *mean curvature H* and the *Gauss curvature K* are defined by

$$\begin{aligned} H &:= \frac{1}{2}\operatorname{tr}\left(\mathbf{II}\cdot\mathbf{I}^{-1}\right) = \frac{1}{2}\,\frac{L\,G - 2\,M\,F + N\,E}{E\,G - F^2}, \\ K &:= \det\left(\mathbf{II}\cdot\mathbf{I}^{-1}\right) = \frac{L\,N - M^2}{E\,G - F^2}. \end{aligned} \tag{1}$$

Both H and K are independent of the choice of coordinates on Ω, and it turns out that H is an extrinsic property of $\mathcal{S}$, while K is an intrinsic property. The latter fact is the well-known *Theorem Egregrium* of Gauss.

An important example of an intrinsic property of $\mathcal{S}$ is its *area*, which we denote by $A(\mathcal{S})$. This area is expressed in terms of the first fundamental form as follows:

$$A\,(\mathcal{S}) := \int_\Omega dA = \int_\Omega \sqrt{\det\mathbf{I}}\;du\,dv = \int_\Omega \sqrt{E\,G\, - F^2}\;du\,dv, \tag{2}$$

where $dA = \sqrt{\det\mathbf{I}}\;du\,dv$ is the infinitesimal element of area on $\mathcal{S}$ with respect to the parametrization X. Note that the integral in equation (2) is independent of the choice of parametrization, so that it only depends on the trace $\mathcal{S}$ of X.

Suppose now that $\rho : \Omega \to \mathbb{R}$ is a smooth function. Then we can use ρ to define a *normal variation* X_t of M as follows:

$$X_t\,(u,\,v) := X\,(u,\,v) + t\,\rho\,(u,\,v)\,N\,(u,\,v).$$

For small values of t, the image $X_t(\Omega)$ is a smooth surface near $\mathcal{S} = X(\Omega)$. If $A(t)$ denotes the area of $X_t(\Omega)$, then a straightforward calculation shows that

$$A'\,(0) = -2\int_\Omega \rho\,H\,dA,$$

where H and dA denote the mean curvature and element of area of $\mathcal{S}$. If a surface is locally area minimizing, we expect the derivative $A'(0)$ to vanish for all choices of ρ. This can only happen if H vanishes identically. Hence, we have the following:

Theorem. *If the surface $\mathcal{S} \subseteq \mathbb{R}^3$ is locally area minimizing, then the mean curvature H of $\mathcal{S}$ vanishes identically ($H \equiv 0$).*

The standard definition of minimal surface is motivated by this fact:

Definition: *The surface $\mathcal{S} \subseteq \mathbb{R}^3$ is minimal if its mean curvature H vanishes identically.*

Thus $H \equiv 0$ is a necessary, but not sufficient, condition for $\mathcal{S}$ to be locally area minimizing. Determining whether a minimal surface is actually locally area minimizing would entail calculating the *second variation* of the area functional on $\mathcal{S}$. The implication would follow if the second variation were always positive.

The *total curvature* $K_{\mathcal{S}}$ of $\mathcal{S}$ is defined to be

$$K_{\mathcal{S}} := \int_{\Omega} |K| \, dA.$$

We say that $\mathcal{S}$ has *finite total curvature* if $K_{\mathcal{S}} < \infty$. Furthermore, we say that $\mathcal{S}$ is *complete* if all its geodesics can be extended indefinitely. A surface is said to be of *finite topological type* if it can be smoothly deformed into a compact surface of finite genus, possibly with several holes. It was long conjectured that the only complete, embedded minimal surfaces in $\mathbb{R}^3$ of finite topological type are the plane, the catenoid, and the helicoid. Costa's minimal surface was the first counterexample to this conjecture to have been found (see [3] for details).

□ The Weierstrass Representation for Minimal Surfaces

It turns out that the geometry of minimal surfaces is intimately related to complex function theory. This connection leads to a simple recipe for constructing minimal surfaces, which we describe here. We only state the necessary results; further details may be found in [2].

We identify $\mathbb{R}^2$ with the complex plane $\mathbb{C}$ by means of the usual correspondence $(u, v) \leftrightarrow u + i\,v$. Suppose that Ω is a *simply connected region* in $\mathbb{C}$, that is, a region in $\mathbb{C}$ in which all closed curves can be contracted to a point. A complex-valued function f on Ω is said to be *holomorphic* if its complex derivative $f'(z)$ exists for all $z \in \Omega$.

Theorem. *Suppose that $f, g : \Omega \to \mathbb{C}$ are two holomorphic functions on a simply connected region Ω, and define $\psi : \Omega \to \mathbb{C}^3$ to be the holomorphic curve with components*

$$\begin{aligned} \psi_1(z) &:= \frac{f(z)}{4} \left(1 - g(z)^2\right), \\ \psi_2(z) &:= i \, \frac{f(z)}{4} \left(1 + g(z)^2\right), \\ \psi_3(z) &:= \frac{1}{2} \, f(z) \, g(z). \end{aligned} \tag{3}$$

Then we have the following:

(i) Componentwise integration

$$\gamma(z) := \int_{z_0}^{z} \psi(\zeta) \, d\zeta \tag{4}$$

yields a holomorphic curve $\gamma : \Omega \to \mathbb{C}^3$.

(ii) For each $t \in \mathbb{R}$, the trace of the map

$$X_t(u, v) := \operatorname{Re}\left[e^{i\,t} \, \gamma(u + i\,v)\right] \tag{5}$$

is a minimal surface. The collection of all such maps is called the associate family of γ.

(iii) For any $t \in \mathbb{R}$*, g is the stereographic projection into* $\mathbb{C}$ *of the Gauss map of* X_t.

Since f and g are assumed to be holomorphic on Ω, and Ω is assumed to be simply connected, it follows from basic complex function theory that the integration in equation (4) is path independent.

We refer to a triple (f, g, Ω) satisfying the above conditions as the *Weierstrass data* for the corresponding associate family $X_t(\Omega)$, and we refer to X_0 as the *Weierstrass representation* of the minimal surface $\mathcal{S} := X_0(\Omega)$. Note that X_t is an *isothermal parametrization* for each $t \in \mathbb{R}$, that is, the coefficients of the first fundamental form satisfy $E = G$ and $F = 0$. In fact, it can be shown that

$$ds^2 = \frac{1}{4} \| f(z) \|^2 \left(1 + \| g(z) \|^2\right)^2 \| dz \|^2, \tag{6}$$

where $z = u + i\,v$. Also, we have

$$K = \frac{-16 \| g'(z) \|^2}{\| f(z) \|^2 \left(1 + \| g(z) \|^2\right)^4}. \tag{7}$$

Here, $\| z \|$ denotes the complex norm of z and $\| dz \|^2 = du^2 + dv^2$.

□ Meromorphic Functions on Complex Tori

Our goal in this subsection is to introduce the Weierstrass data used to obtain a parametrization of Costa's surface. Before doing so, we provide a little background in elliptic function theory. Details may be found in [4].

Suppose that ω_1 and ω_2 are two complex numbers such that $\text{Im}[\omega_1 / \omega_2] \neq 0$. Then the lattice of points $\Gamma := \{k_1\,\omega_1 + k_2\,\omega_2 \mid k_1, k_2 \in \mathbb{Z}\}$ is a subgroup of the group of translations on $\mathbb{C}$ and Γ is isomorphic to the additive group of Gaussian integers $\mathbb{Z}[i]$. Thus, $\omega \in \Gamma$ acts on $\mathbb{C}$ by the rule $z \to z + \omega$, and the quotient space $\mathbb{C} / \Gamma$ is topologically a torus, which inherits a complex structure from $\mathbb{C}$. Let $\pi : \mathbb{C} \to \mathbb{C} / \Gamma$ denote the corresponding projection map. We refer to ω_1 and ω_2 as basic periods of Γ and sometimes write T for $\mathbb{C} / \Gamma$.

Any function $\hat{f}$ on $\mathbb{C} / \Gamma$ lifts to a function f on $\mathbb{C}$. Such a function satisfies $f(z + \omega) = f(z)$ for all $\omega \in \Gamma$ and is said to be Γ *periodic*. It is not possible for a complex function to be both Γ periodic and holomorphic in all of $\mathbb{C}$, but there is a rich theory of functions that are Γ periodic and meromorphic. A function f is *meromorphic* on Ω if it is holomorphic on $\Omega \backslash A$, where A is a discrete set without accumulation points in Ω, and if, for any $a \in A$, f has a *pole* at a, that is, f has a power series expansion in a neighborhood of a of the form

$$f(z) = \frac{c_{-k}}{(z - a)^k} + \cdots + \frac{c_{-1}}{z - a} + c_0 + c_1 (z - a) + c_2 (z - a)^2 + \cdots.$$

The positive integer k is called the *order* of the pole at a. A pole is said to be *simple* if it is of order one. Functions that are both meromorphic and periodic with respect to some lattice Γ are referred to as *elliptic functions*.

Our objective in what follows is to describe a particular solution of the period problem. In the context of complex tori, it may stated as follows:

The Period Problem: *Find meromorphic functions f and g, periodic with respect to some lattice Γ in $\mathbb{C}$, such that* $\operatorname{Re}[\gamma]$ *is also Γ periodic, where γ is given by equation (4). Here, the Weierstrass data is $(f, g, \Omega \backslash A)$, where Ω is a fundamental domain of Γ in $\mathbb{C}$, and A is the set of poles of f and g in Ω.*

Note that, by making appropriate cuts in $\Omega \backslash A$, we may consider it to be simply connected. Any solution to this problem will topologically be a torus (possibly with several holes) immersed in $\mathbb{R}^3$.

Costa's surface arises from what is arguably the most basic of elliptic functions: the Weierstrass $\wp$ function. It is defined by the series expansion

$$\wp(z; \Gamma) := \frac{1}{z^2} + \sum_{\omega \in \Gamma_*} \left\{ \frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right\}, \tag{8}$$

where $\Gamma_* := \Gamma \backslash \{0\}$ denotes the set of nonzero elements in Γ. This function has poles of order two at each of the lattice points in Γ and is holomorphic everywhere else in $\mathbb{C}$. It therefore projects to a meromorphic function with exactly one pole of order two in $\mathbb{C} / \Gamma$. It is known that any meromorphic function on $\mathbb{C} / \Gamma$ may be expressed as a rational function of $\wp(z; \Gamma)$ and its complex derivative $\wp'(z; \Gamma)$. In fact, these functions are related by the fundamental equation

$$\wp'(z; \Gamma)^2 = 4\,(\wp(z; \Gamma) - e_1)\,(\wp(z; \Gamma) - e_2)\,(\wp(z; \Gamma) - e_3), \tag{9}$$

where

$$e_1 := \wp\left(\frac{\omega_1}{2}\right),\ e_2 := \wp\left(\frac{\omega_2}{2}\right),\ e_3 := \wp\left(\frac{\omega_1 + \omega_2}{2}\right). \tag{10}$$

A related function is the Weierstrass ζ function, which is defined by

$$\zeta(z; \Gamma) := \frac{1}{z} + \sum_{\omega \in \Gamma_*} \left\{ \frac{1}{z-\omega} + \frac{1}{\omega} + \frac{z}{\omega^2} \right\}. \tag{11}$$

The ζ function is not Γ periodic and hence not elliptic. It is holomorphic on $\mathbb{C} \backslash \Gamma$, however, and has simple poles at the points of Γ. Furthermore, it is related to the Weierstrass $\wp$ function by the rule

$$\zeta'(z; \Gamma) = -\wp(z; \Gamma). \tag{12}$$

This function arises naturally when calculating the integral in equation (4) for the Weierstrass data of Costa's surface. Although the ζ function is not Γ periodic, it does satisfy the following period relations:

$$\zeta(z+\omega_k;\,\Gamma) = \zeta(z;\,\Gamma) + 2\,\zeta\left(\frac{\omega_k}{2};\,\Gamma\right),\; k = 1, 2. \tag{13}$$

Furthermore, a theorem due to Legendre states that

$$\omega_2\,\zeta\left(\frac{\omega_1}{2};\,\Gamma\right) - \omega_1\,\zeta\left(\frac{\omega_2}{2};\,\Gamma\right) = \pi\,i. \tag{14}$$

These facts are used in the next subsection.

We refer to special points in $\mathbb{C}$ or $\mathbb{C}/\Gamma$ as *marked points*. These are points at which singularities, such as poles, occur. Open disks centered at marked points are referred to as *marked disks*. Let $D(z, \varepsilon)$ denote the disk of radius ε with center at the point $z \in \mathbb{C}$, and define $D_\Gamma(z, \varepsilon)$ by

$$D_\Gamma(z, \varepsilon) := D(z, \varepsilon) + \Gamma = \{x + \omega \mid x \in D(z, \varepsilon),\, \omega \in \Gamma\}.$$

The set $D_\Gamma(z, \varepsilon)$ can be viewed as the collection of all points in $\mathbb{C}$ that are mapped to a marked disk in $\mathbb{C}/\Gamma$ by the projection π.

We now specialize to the case where $\omega_1 = 1$ and $\omega_2 = i$. In this case, Γ is the standard square lattice of Gaussian integers in $\mathbb{C}$, and we simply write $\wp(z)$ for $\wp(z;\,\Gamma)$ and $\zeta(z)$ for $\zeta(z;\,\Gamma)$. The numbers in equation (10) now satisfy

$$e_1 = -e_2,\; e_3 = 0. \tag{15}$$

The set $U := \{u + i\,v \mid 0 \leqslant u, v \leqslant 1\}$ defines a fundamental domain of the covering $\pi : \mathbb{C} \to \mathbb{C}/\Gamma$. Let

$$B = D_\Gamma(0, \varepsilon_1) \bigcup D_\Gamma(1/2, \varepsilon_2) \bigcup D_\Gamma(i/2, \varepsilon_3),$$

where ε_1, ε_2, and ε_3 are small positive numbers, and define $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ to be $U\backslash B$. Note that $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ is a unit square in $\mathbb{C}$ with four quarter-disks of radius ε_1 removed from the corners of U, two half-disks of radius ε_2 removed from the midpoints of the horizontal edges of U, and two half-disks of radius ε_3 removed from the midpoints of the vertical edges of U. The projection $\pi(U\backslash B)$ is a torus with three marked disks removed. These disks are centered at $\pi(0)$, $\pi(1/2)$, and $\pi(i/2)$.

With these preliminaries, we are ready to specify the Weierstrass data for Costa's surface. We take our domain to be $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$, and set

$$f(z) = \wp(z),\; g(z) = \frac{\sqrt{8\,\pi\,e_1}}{\wp'(z)}. \tag{16}$$

Both of these functions are holomorphic on $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$, and since $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ is simply connected, the integration in equation (4) is path independent.

□ Costa's Minimal Surface

In general, one might have to resort to numerical integration in equation (4) to obtain the Weierstrass representation for a surface. However, the integration can be carried out explicitly for the functions in equation (16), when restricted to $\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)$. This calculation was first performed by Alfred Gray and is given in [2]. The result is as follows:

Theorem. *Let* $\Gamma[1/2] := \{k_1\,\omega_1 + k_2\,\omega_2 \mid 2\,k_1, 2\,k_2 \in \mathbb{Z}\}$ *denote the lattice of Gaussian half-integers. Then the Weierstrass data in equation (16), when substituted into equation (3) and integrated, yields the holomorphic curve* $\gamma : \mathbb{C}\backslash\Gamma[1/2] \to \mathbb{C}^3$*, whose components are given by*

$$\begin{aligned}
\gamma_1(z) &:= \frac{1}{2}\left\{-\zeta(z) + \pi(z - i) + \frac{\pi^2(1+i)}{4\,e_1}\right\} + \frac{\pi}{4\,e_1}\left\{\zeta\left(z - \frac{1}{2}\right) - \zeta\left(z - \frac{i}{2}\right)\right\},\\
\gamma_2(z) &:= \frac{i}{2}\left\{-\zeta(z) - \pi\,(z - 1) - \frac{\pi^2(1+i)}{4\,e_1}\right\} - \frac{i\,\pi}{4\,e_1}\left\{\zeta\left(z - \frac{1}{2}\right) - \zeta\left(z - \frac{i}{2}\right)\right\},\\
\gamma_3(z) &:= \frac{\sqrt{2\,\pi}}{4}\left\{\log\left(\frac{\wp(z) - e_1}{\wp(z) + e_1}\right) - i\,\pi\right\}.
\end{aligned} \tag{17}$$

The corresponding trace of $X_0 = \mathrm{Re}[\gamma]$ *is Costa's minimal surface.*

We now use the properties of $\wp$ and ζ discussed earlier to demonstrate that Costa's surface is topologically a torus with three points removed. Define three points in $\mathcal{T} := \mathbb{C}/\Gamma$ by $\alpha_1 := \pi(0)$, $\alpha_2 := \pi(1/2)$, and $\alpha_3 := \pi(i/2)$. We may think of these points as the projection to $\mathcal{T}$ of $\Gamma[1/2]$ in $\mathbb{C}$. From the form of γ in equation (17), it is clear that γ is holomorphic on $\mathbb{C}\backslash\Gamma[1/2]$.

Note that, for the basic periods $(1, i)$, we have $i\,\Gamma = \Gamma$ as sets. In this case, it is clear from the definition of the ζ function in equation (11) that

$$\zeta(i\,z; \Gamma) = \zeta(i\,z; i\,\Gamma) = \frac{1}{i}\,\zeta(z; \Gamma).$$

This fact, together with equation (14), implies that

$$\zeta\left(\frac{1}{2}\right) = \frac{\pi}{2},\ \zeta\left(\frac{i}{2}\right) = -\frac{\pi\,i}{2}. \tag{18}$$

Equations (13, 17, 18) then allow us to conclude that

$$\gamma\,(z+1) = \gamma\,(z) + i\,\pi\,(1, 0, 0),\ \gamma\,(z+i) = \gamma\,(z+i) + i\,\pi\,(0, -1, 0).$$

This clearly implies that $\mathrm{Re}[\gamma(z+\omega)] = \mathrm{Re}[\gamma(z)]$ for all $\omega \in \Gamma$, that is, that X_0 is Γ periodic. Thus, we have demonstrated the following:

Proposition: *The map* $X_0 = \mathrm{Re}[\gamma]$*, where* γ *is given by equation (17), solves the period problem. Hence* X_0 *projects to a real-analytic map* $\hat{X}_0 : \mathcal{T}\backslash\{\alpha_1, \alpha_2, \alpha_3\} \to \mathbb{R}^3$.

Substituting equation (16) into equation (6), we see, with the help of equations (9, 15), that the metric on $\mathcal{T}\backslash\{\alpha_1, \alpha_2, \alpha_3\}$ is given by

$$ds^2 = \frac{1}{4}\left(\|\wp(z)\| + \frac{2\pi\|e_1\|}{\|\wp(z)^2 - e_1{}^2\|}\right)^2 \|dz\|^2 . \tag{19}$$

From this formula, it is not difficult to show that the function $ds/\|dz\|$ has poles of order two at α_1, α_2, and α_3. Thus, the metric diverges at the *ends* of Costa's surface, which we define to be the image under $\hat{X}_0$ of the punctured disks $\pi(D'(0,\varepsilon))$, $\pi(D'(1/2,\varepsilon))$, and $\pi(D'(i/2,\varepsilon))$ in $\mathcal{T}$. Here, $D'(z,\varepsilon)$ denotes the disk $D(z,\varepsilon)$ in $\mathbb{C}$, excluding its center. We shall see later that $\hat{X}_0$ diverges at different rates at α_1 than at α_2 and α_3. This is due to the fact that the principal part of the Laurent expansion of $ds/\|dz\|$ at $z = 0$ has a larger leading coefficient than that at $z = 1/2$ or $z = i/2$, even though all of the poles are of the same order. As can be seen in Figure 3, the end corresponding to the punctured disk about $\alpha_1 = \pi(0)$ is asymptotically planar, while the other two ends are asymptotically catenoidal in form.

In the next section, we describe the contents of the *Minimal Surfaces* package. After that, we show how to use the package to generate polyhedral data representing the trace of the parametrization

$$\hat{X}_0 : \hat{\Omega}(\varepsilon_1, \varepsilon_2, \varepsilon_3) \to \mathbb{R}^3, \tag{20}$$

where $\hat{\Omega}(\varepsilon_1, \varepsilon_2, \varepsilon_3) := \pi[\Omega(\varepsilon_1, \varepsilon_2, \varepsilon_3)]$.

■ The Minimal Surfaces Package

We now describe the public functions in the *Minimal Surfaces* package. Note that there are a number of utility functions with private context that are not described here. Further documentation may be found within the package source file.

□ Special Data Types

`Arc[{x, y}, r, {θ1, θ2}, orientation]` is a two-dimensional graphics primitive specifying a circular arc with center $\{x, y\}$, radius r, and initial and terminal angles $\{\theta_1, \theta_2\}$, which are assumed to satisfy $-\pi \le \theta_1, \theta_2 \le \pi$. The *orientation* may be either `Clockwise` or `CounterClockwise`.

□ Surface-Generating Functions

`CostaSurface[u, v]` returns the image in $\mathbb{C}^3$ corresponding to the point in $\mathbb{R}^2 \setminus \mathbb{Z}^2$ with coordinates (u, v).

`ParallelSurface[X, p, d]` returns the normal displacement at distance d from $X(p)$ along the normal of X at $X(p)$, where X is a regular map on some domain $\Omega \subseteq \mathbb{R}^2$ into $\mathbb{R}^3$.

◻ Vertex Creation and Triangulation

`CreateVertexData[curve, options]` produces a collection of planar points within the boundary specified by the closed curve *curve* = $\{c_1, c_2, \ldots\}$, which consists of a list of line segments and circular arcs and is assumed to be traversed counterclockwise.

The output is a list of the form {*interior*, *boundary*}. The *interior* list consists of planar points properly inside the boundary specified by *curve*, while *boundary* $= \{b_1, b_2, \ldots\}$ is a list of sublists, each member of which is a list of planar points lying on the corresponding line segment or arc specified by *curve*.

Currently, the only supported option is `MeshSize` → $\{n_1, n_2\}$, which specifies the number of sample points to use in the x and y directions. The defaults are $n_1 = n_2 = 10$.

`GlueComponents[top, bottom]` glues together a pair of polyhedral surfaces specified by the lists *top* and *bottom*. These lists are assumed to have the same form {*vertices*, *faces*, *boundary*} as output from the `Triangulate` function. It is also assumed that the boundary components of *top* and *bottom* have the same shape.

The output has the form {*vertices*, *faces*}, where *vertices* is the join of the vertex sublists of *top* and *bottom*, and *faces* is a list of sublists of the form {*top-faces*, *bottom-faces*, *boundary-faces*}.

`Triangulate[vertices, curve, options]` produces a triangulation of a planar region. The list *vertices* is of the same form as the output of `CreateVertexData`, and *curve*, which consists of a list of line segments and circular arcs, defines the boundary of the region to be triangulated.

The output is a list of the form {*vertices*, *faces*, *boundary*}. The elements of *vertices* are planar points of the form $\{x, y\}$. The elements of *faces* are ordered triples of positive integers $\{a, b, c\}$. The elements of such triples refer to positions in the list of vertices and thereby define triangles. The list *boundary* is of the form $\{p_1, p_2, \ldots\}$, where each sublist p_j contains positive integers pointing to the position in the list *vertices* of points that lie on the j^{th} connected component of the boundary of the triangulated region.

This function has one option, which is of the form *Identifications* → l. The default value for l is $l = \{\}$, in which case no identifications occur. The other possibility is $l = \{l_1, l_2\}$, where each element of l_1 specifies the positions of a pair of edges in *curve* that are to be identified, and each element of l_2 specifies the positions of a cycle of circular arcs whose endpoints are to be identified.

Remark: The result after identification is assumed to be a Riemann surface, possibly with several marked disks removed.

◻ Graphics Functions

`CreatePolyhedron[{vertices, faces}]` returns a list of 3D graphics directives and primitives describing a polyhedron. The sublist *vertices* consists of points that are the vertices of a polyhedron, and *faces* is a list of sublists of the form $\{x_1, x_2, \ldots\}$, where x_j has the form $\{g_j, f_j\}$. Each g_j is a list of graphics directives to be applied to the face list f_j.

Remark: This function could easily be modified to allow the application of graphics directives to edges and vertices.

◻ Import/Export Functions

`ExportGraphics["file", data, "format"]` writes geometric data contained in the list *data* to an ASCII file named *file*. The list *data* is assumed to be in the same form as input data to `CreatePolyhedron`, except that *Mathematica* graphics directives are replaced with equivalent directives that are compliant with the export format. The form of the output is specified by *format*. Currently the only supported format is *PLY,* which stands for the polygon file format

Remark: A desirable enhancement to this function would be to include a *POVRAY* export format. This would produce object data appropriate for creating scenes with the Persistence of Vision Raytracer program.

◻ The Polygon File Format

We now give a brief summary of the PLY file format, limiting our discussion to those aspects used in the current version of the `ExportGraphics` function. Further details may be found in [5].

A PLY file has three main parts: a header, a list of vertices, and a list of faces. Listing 1 is an example of a PLY file specifying a cube with faces of various colors. The header contains comments and declarations of data types and their properties. Each line of a comment starts with the token *comment*, and data types are specified with the token *element*. The elements necessary for our purposes are *vertex* and *face*. The declaration of an element must include the number of occurrences of elements of that type in the file. The element's properties are declared immediately after its declaration. For example, the group of statements

```
element vertex 8
property float32 x
property float32 y
property float32 z
```

specifies that the file in question contains data for eight vertices and that each vertex has three properties (x, y, z), which are floating-point numbers representing the coordinates of the vertex.

Each line of the vertex list specifies the (x, y, z) coordinates of a vertex. A list of data specifying faces follows the vertex data. Each line of this list gives, in order, the number of vertices in a face and the position in the vertex list corresponding to a vertex of the face. Color attributes of the face can then be given in the form of RGB-color intensities, which are specified by integers from 0 to 255. The orientation of a face can be determined from the order in which its vertices are presented. Note that indexing of the vertices begins at 0. Thus, when 0 occurs in the vertex list of a face in Listing 1, for example, it points to the first vertex, which has coordinates $(0, 0, 0)$.

```
ply
format ASCII 1.0
comment a cube                            a comment
element vertex 8                          8 vertices in file
property float32 x                        x coordinate of vertex
property float32 y                        y coordinate of vertex
property float32 z                        z coordinate of vertex
element face 6                            6 faces in file
property list uchar int vertex_indices    vertex incidence list
property uchar red
property uchar green
property uchar blue
end header
0 0 0                                     start of vertex list
0 0 1
0 1 1
0 1 0
1 0 0
1 0 1
1 1 1
1 1 0
4 0 1 2 3 128   0 128                     start of face list
4 7 6 5 4 192 128   0
4 0 4 5 1 128 192   0
4 1 5 6 2 256   0   0
4 2 6 7 3   0 256   0
4 3 7 4 0   0   0 256
```

▲ **Listing 1.** A simple PLY file specifying a cube with faces of various colors. The comments in italics to the right are not part of the file.

□ A Note on Performance

Of the functions listed above, the most time-consuming is `Triangulate`, which internally calls `DelaunayTriangulation`. It seemed convenient to use the latter because it is part of the `ComputationalGeometry` package, which is a standard *Mathematica* add-on package. Another attractive feature of Delaunay triangulation is that, by design, it produces the most regular triangulation of a planar point set. The most efficient algorithms for Delaunay triangulation have a time complexity of $O(n \log(n))$.

The model shown in Figure 4 was created by evaluating `Triangulate` on a vertex set containing about 60,000 points. This took more than 14 hours on a workstation with dual Opteron 244 processors. (*Mathematica* Version 5.2 only used one CPU, and the availability of RAM was not an issue.)

There are at least two ways the overall performance of the *Minimal Surfaces* package could be improved, both of which involve reducing the vertex data used as input to `Triangulate`. This is discussed further in the subsection on enhancements.

We note that an alternative approach might be to replace the current version of `Triangulate` with an "adaptive cell division algorithm." In this scenario, one would start with a sparse vertex set in the given domain Ω, which would be triangulated by some method. Then one would use some function, such as the metric in equation (6), to decide if the triangulation needs to be refined in a neighborhood of any given vertex. The best method of refinement is likely to be midpoint subdivision of any triangle incident to the vertex in question. This may prove to be faster than using Delaunay triangulation.

■ Generating the Model Data for Costa's Surface

We now show how the functions described above are used to create a model of Costa's surface. The essential steps are as follows:

1. Use the `Line` and `Arc` data types to define a curve bounding a suitable domain $\Omega \subseteq \mathbb{C}$.
2. Call `CreateVertexData` to obtain a collection of points $V \subseteq \Omega$.
3. Apply `Triangulate` to V in order to obtain its Delaunay triangulation; provide boundary identifications, if appropriate.
4. Apply a composition of `ParallelSurface` and `CostaSurface` to the data V generated in step 2 to obtain vertex data for two polyhedral surfaces in $\mathbb{R}^3$ that are normal displacements of points lying on Costa's surface. Note that the faces of the resulting polyhedral surfaces are defined by the same incidence relations resulting from step 3.
5. Use `GlueComponents` to create a single volume-bounding polyhedron from the two normally displaced polyhedral surfaces created in step 4.
6. Call `ExportGraphics` to produce the PLY file required for printing a solid model.

First, we load the *Minimal Surfaces* package and define some useful functions. It is available from www.mathematica-journal.com/data/uploads/2010/12/MinimalSurfaces.m. We assume the package is contained in the same directory as this article.

```
SetDirectory[NotebookDirectory[]];
<< MinimalSurfaces`
```

The following function defines the curve bounding the region $\Omega(\delta, \epsilon, \phi)$ described in the paragraph after equation (15). Recall that the input parameters (δ, ϵ, ϕ) represent the radii of the three marked disks with centers at $\pi(0)$, $\pi(1/2)$, and $\pi(i/2)$, where π, as before, denotes the projection $\pi : \mathbb{C} \to \mathbb{C}/\Gamma$. Note that this curve consists of 16 segments and is traversed counterclockwise, hence the circular arcs it contains are traversed clockwise.

```
boundary[δ_, ϵ_, ϕ_] := {
  Line[{{δ, 0}, {(1 / 2) - ϵ, 0}}],
  Arc[{1 / 2, 0}, ϵ, {π, 0}, Clockwise],
  Line[{{(1 / 2) + ϵ, 0}, {1 - δ, 0}}],
  Arc[{1, 0}, δ, {π, π / 2}, Clockwise],

  Line[{{1, δ}, {1, (1 / 2) - ϕ}}],
  Arc[{1, 1 / 2}, ϕ, {-π / 2, π / 2}, Clockwise],
  Line[{{1, (1 / 2) + ϕ}, {1, 1 - δ}}],
  Arc[{1, 1}, δ, {-π / 2, -π}, Clockwise],

  Line[{{1 - δ, 1}, {(1 / 2) + ϵ, 1}}],
  Arc[{1 / 2, 1}, ϵ, {0, -π}, Clockwise],
  Line[{{(1 / 2) - ϵ, 1}, {δ, 1}}],
  Arc[{0, 1}, δ, {0, -π / 2}, Clockwise],

  Line[{{0, 1 - δ}, {0, (1 / 2) + ϕ}}],
  Arc[{0, 1 / 2}, ϕ, {π / 2, -π / 2}, Clockwise],
  Line[{{0, (1 / 2) - ϕ}, {0, δ}}],
  Arc[{0, 0}, δ, {π / 2, 0}, Clockwise]
 }
```

The function `grid` generates a rectangular array of horizontal and vertical lines with offset $q = \{q_1, q_2\}$, step size $s = \{s_1, s_2\}$, and index range $n = \{\{n_{11}, n_{12}\}, \{n_{21}, n_{22}\}\}$; it is used to generate the background grid in Figure 2.

```
gridlines[q_, s_, n_] :=
  Join[
   Table[
    Line[{{q⟦1⟧ + k s⟦1⟧, q⟦2⟧ + n⟦2⟧⟦1⟧ s⟦2⟧},
      {q⟦1⟧ + k s⟦1⟧, q⟦2⟧ + n⟦2⟧⟦2⟧ s⟦2⟧}}],
    {k, n⟦1⟧⟦1⟧, n⟦1⟧⟦2⟧}
   ],
   Table[Line[{{q⟦1⟧ + n⟦1⟧ ⟦1⟧ s⟦1⟧, q⟦2⟧ + k s⟦2⟧},
      {q⟦1⟧ + n⟦1⟧ ⟦2⟧ s⟦1⟧, q⟦2⟧ + k s⟦2⟧}}],
    {k, n⟦2⟧⟦1⟧, n⟦2⟧⟦2⟧}]
  ];
```

□ Example Output from CreateVertexData and Triangulate

Before invoking the commands that produce the graphics object illustrated in Figure 3, we illustrate what the output of the `CreateVertexData` and `Triangulate` functions looks like for a small mesh size.

```
b = boundary[1 / 6, 1 / 4, 1 / 4];
v = CreateVertexData[b, MeshSize → {10, 10}];
```

The list `v` produced is of length 2. The first part of `v` consists of the interior points shown in green in Figure 1. Here is a short listing.

```
v⟦1⟧ // Short
```

$$\left\{\left\{\frac{1}{10}, \frac{1}{5}\right\}, \ll 43 \gg, \left\{\frac{9}{10}, \frac{4}{5}\right\}\right\}$$

The second part of `v` consists of 16 sublists, each of which contains vertices that lie on the corresponding segment of `b`. For example, the last segment in `b` corresponds to the quarter-circle in the lower-left corner of Figure 1. The vertices that lie in the segment `b⟦-1⟧` are the elements of the list `v⟦2, -1⟧`.

```
v⟦2, -1⟧
```

$$\left\{\left\{0, \frac{1}{6}\right\}, \left\{\frac{1}{10}, \frac{2}{15}\right\}, \left\{\frac{2}{15}, \frac{1}{10}\right\}, \left\{\frac{1}{6}, 0\right\}\right\}$$

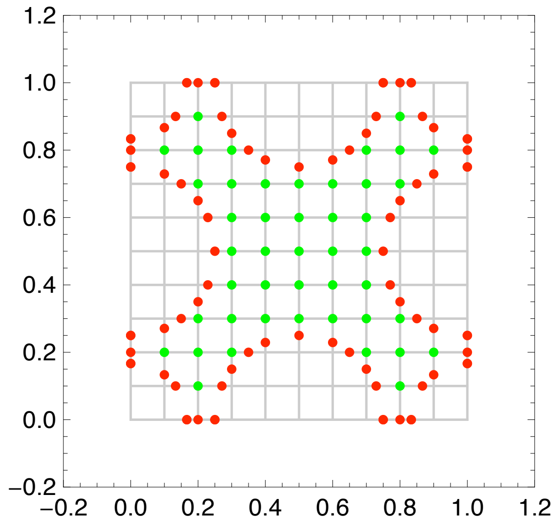
Here is the first part of `v⟦2⟧`, which contains vertices that lie on the leftmost horizontal line segment on the real axis. Note that the endpoints of incident segments overlap.

```
v⟦2, 1⟧
```

$\left\{\left\{\frac{1}{6}, 0\right\}, \left\{\frac{1}{5}, 0\right\}, \left\{\frac{1}{4}, 0\right\}\right\}$

Figure 1, below, graphically illustrates the structure of v. The green points are vertices that lie in the interior of the boundary curve b, and the red points are vertices lying in b itself. Note that all of the points in v lie in the gray rectangular grid.

```
Show[
 Graphics[
  {{GrayLevel[0.8`], gridlines[{0, 0}, {0.1`, 0.1`},
     {{0, 10}, {0, 10}}]},
   {Green, PointSize[0.02`], Point /@ v⟦1⟧},
   {Red, PointSize[0.02`], Point /@ Flatten[v⟦2⟧, 1]}}],
 Frame → True, PlotRange → {{-0.2`, 1.2`}, {-0.2`, 1.2`}},
 AspectRatio → Automatic]
```



▲ **Figure 1.** Vertex data generated using `CreateVertexData` with parameter values $\delta = 1/6$, $\epsilon = \phi = 1/4$, and `MeshSize → {10, 10}`.

We now triangulate the vertex data v shown in Figure 1 without edge identifications. The resulting triangulation differs from a Delaunay triangulation in that certain triangles incident to boundary points of $\Omega(\delta, \epsilon, \phi)$ but not lying within the domain itself are excluded.

```
p = Triangulate[v, b];
```

The output p of `Triangulate` has three parts. The first part p⟦1⟧ is a flat list of all the vertices (both interior and boundary) shown in Figure 1. Duplicate points have been removed.

```
p〚1〛 // Short
```

$$\left\{\left\{\frac{1}{10}, \frac{1}{5}\right\}, \ll 111 \gg, \left\{\frac{2}{15}, \frac{1}{10}\right\}\right\}$$

The second part of p contains incidence relations that define faces, which are always triangles. The positive integers in each sublist of `p〚2〛` are locations of vertices in `p〚1〛`.

```
p〚2〛 // Short
{{1, 109, 110}, <<154>>, {45, 77, 78}}
```

For example, the following command extracts the vertices in `p〚1〛` that correspond to the first face listed in `p〚2〛`.

```
Part[p〚1〛, p〚2, 1〛]
```

$$\left\{\left\{\frac{1}{10}, \frac{1}{5}\right\}, \left\{0, \frac{1}{4}\right\}, \left\{0, \frac{1}{5}\right\}\right\}$$

The third part of p is a list containing one sublist `p〚3, 1〛` that defines the boundary curve of the triangulation. When identifications are used, the list `p〚3〛` may contain multiple boundary components. In the next subsection, for example, `Triangulate` is called with identifications that produce three boundary components. Each component in that case corresponds to the boundary of a marked disk that has been removed from the torus $\mathcal{T}$ described in the paragraph after equation (17).

```
p〚3〛 // Short
{{46, 47, 48, 49, <<60>>, 110, 111, 112, 113}}
```

The following group of commands first maps the vertices of p into the horizontal plane of $\mathbb{R}^3$; the resulting output list is named q. Then some graphics directives are added to q, and the resulting list is used as input to `CreatePolyhedron`. The output list s is then flattened so that it can be used as input to `Graphics3D`.

```
id[u_, v_] := {u, v, 0}
q = {Apply[id, p〚1〛, {1}], p〚2〛, p〚3〛};
gr = {RGBColor[3/4, 3/4, 0], Specularity[GrayLevel[1], 5]};
qw = {q〚1〛, {{gr, q〚2〛}}};
s = CreatePolyhedron[qw];
ss = Flatten /@ s;
```

In general, the length of the output s of `CreatePolyhedron` is the same as the length of the second part of the input qw. Thus, in the present case, s has length 1. The first part of `s[[1]]` consists of graphics directives, and the second part consists of graphics primitives that are applied to the faces of the triangulation `q`. Part of the output from the current evaluation is shown below. To create the polyhedron shown in Figure 3, we will build a list similar to `qw`, above. The second part of the resulting list itself has five parts, each of which corresponds to one of the five parts of the surface shown in the figure, namely, the green (or inside) part, the yellow (or outside) part, and the three purple rims.

```
s[[1, 1]]
```

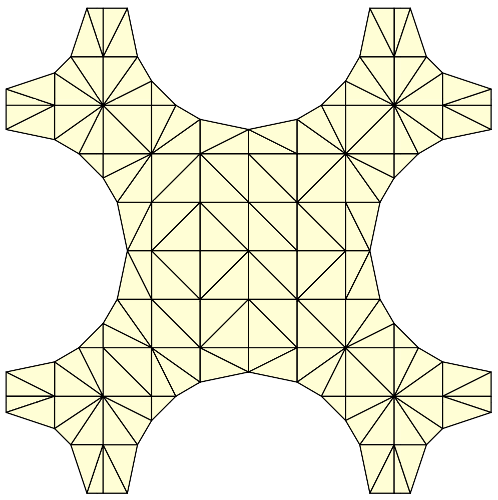
$$\left\{\text{RGBColor}\left[\frac{3}{4}, \frac{3}{4}, 0\right], \text{Specularity}[\text{GrayLevel}[1], 5]\right\}$$

```
s[[1, 2, 1]]
```

$$\text{Polygon}\left[\left\{\left\{\frac{1}{10}, \frac{1}{5}, 0\right\}, \left\{0, \frac{1}{4}, 0\right\}, \left\{0, \frac{1}{5}, 0\right\}\right\}\right]$$

Figure 2 shows what the triangulation of the small dataset looks like.

```
Show[Graphics3D[ss], AspectRatio → Automatic, Axes → False,
 Boxed → False, ViewPoint → {0, 0, 9}]
```



▲ **Figure 2.** Triangulation of the vertex data in Figure 1 using the `Triangulate` function.

□ Creation and Triangulation of Vertex Data for a Torus

In this subsection, we discuss how the `CreateVertexData` and `Triangulate` functions were used to help produce the polyhedron shown in Figure 3. The mesh size of 50×50 employed here is moderate in order to reduce computation time and the size of the resulting datasets. Note that our marked disks are given radii of $\delta = 1/8$ and $\epsilon = \phi = 1/24$. This compensates for the difference in the rate of divergence at the poles $\pi(0)$, $\pi(1/2)$, and $\pi(i/2)$ of the parametrization $\hat{X}_0$ defined in the proposition following equation (17). The ends of the resulting surface are roughly the same size.

```
b = boundary[1 / 8, 1 / 24, 1 / 24];
v = CreateVertexData[b, MeshSize → {50, 50}];
```

The list `edgeIds` specifies edges in `b` that are to be identified, and the list `vertexIds` specifies the endpoints of circular arcs in `b` that are to be identified. For example, the sublist {1, 11} in `edgeIds` indicates that the lower-left horizontal line segment in Figure 2 is to be identified with the upper-left horizontal line segment. Similarly, the sublist {2, 10} in `vertexIds` indicates that corresponding endpoints of the semicircular arcs with centers at $(1/2, 0)$ and $(1/2, i)$ are to be identified. The list `vertexIds` is used to identify which components of the boundary curve close up to form boundary curves. In this case, we get three such curves.

```
edgeIds = {{1, 11}, {3, 9}, {5, 15}, {7, 13}};
vertexIds = {{2, 10}, {4, 16, 12, 8}, {6, 14}};
```

When we pass these identifications to `Triangulate` as an option, output similar to that in the previous subsection is produced, except that some incidence relations in `p` are reassigned, and extra vertices are dropped.

```
p = Triangulate[v, b, Identifications → {edgeIds, vertexIds}];
```

□ Vertex Data for Normal Displacements of Costa's Surface

The `ParallelSurface` function is used to create a small normal offset of the vector-valued function it takes as an argument. Here, the function is `CostaSurface`, which can be used to generate the true Costa surface. Note that, since `CostaSurface` returns a vector in $\mathbb{C}^3$, `ParallelSurface` also returns a vector in $\mathbb{C}^3$.

```
Z[u_, v_, rho_] :=
 ParallelSurface[CostaSurface[x, y], {x, y}, rho] /.
  {x → u, y → v, d → rho}
```

To create the polyhedron in Figure 3, we choose $\rho = \pm 0.025$ as our offset and define two real vector-valued functions accordingly.

```
X1[u_, v_] := Re[Z[u, v, 0.025]]
X2[u_, v_] := Re[Z[u, v, -0.025]]
```

The following commands use X_1 and X_2 to map the vertex data of the two-dimensional triangulation defined by `p` into $\mathbb{R}^3$.

```
q1 = {Apply[X₁, p⟦1⟧, {1}], p⟦2⟧, p⟦3⟧};
q2 = {Apply[X₂, p⟦1⟧, {1}], p⟦2⟧, p⟦3⟧};
```

The function `GlueComponents` is now used to connect the boundary components of `q1` and `q2`. First, the vertex and face lists of `q1` and `q2` are concatenated, and then additional faces, which are incident to vertices in the boundary curves, are appended to the face list. The resulting polyhedron bounds a volume in $\mathbb{R}^3$.

```
zx = GlueComponents[q1, q2];
```

We now define the graphics directives we wish to apply to each component of our surface.

```
gr1 = {EdgeForm[], RGBColor[1/2, 3/4, 0],
   Specularity[GrayLevel[0.5`], 6]};
gr2 = {EdgeForm[], RGBColor[3/4, 1/2, 0],
   Specularity[GrayLevel[1], 9]};
gr3 = {EdgeForm[], RGBColor[1/2, 0, 1/2],
   Specularity[GrayLevel[1], 9]};
```

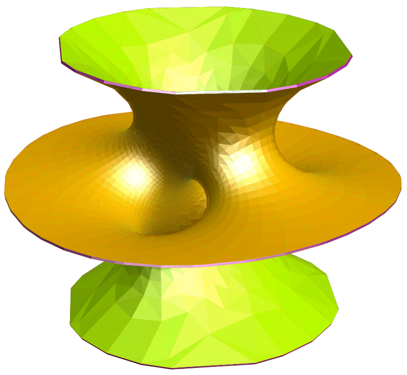
The following command produces input in the form required by `CreatePolyhedron`. That is, `qw` has the form `qw = {verts, {{g1, f1}, …, {gn, fn}}}`, where each f_j is a list of incidence relations in the vertex list *verts* defining faces and each g_j is a list of graphics directives to be applied to f_j.

```
qw = {zx⟦1⟧, Join[{{gr1, zx⟦2⟧⟦1⟧}, {gr2, zx⟦2⟧⟦2⟧}},
    Map[{gr3, #} &, zx⟦2⟧⟦3⟧]]};
```

Now we apply `CreatePolyhedron` to get a graphics object. As before, we need to flatten the list `s` to produce input acceptable to the `Graphics3D` function.

```
s = CreatePolyhedron[qw];
ss = Map[Flatten, s];
```

```
Graphics3D[ss, AspectRatio → Automatic, Axes → False,
 Boxed → False, Lighting → "Neutral",
 ViewPoint → {-2.417`, -1.984`, 1.294`}]
```



▲ **Figure 3.** Costa's surface with $\delta = 1/8$, $\epsilon = \phi = 1/24$, $\rho = \pm 0.1$, and `MeshSize → {50, 50}`.

□ Printing Technology and the Solid Model

The photograph shown in Figure 4 is a 3D model of Costa's surface that was printed using a ZCorp Model 402Z 3D printer. This device is no longer in production but is similar to the ZCorp Model 510. The principal difference between the two is that the Model 510 has a higher print resolution. See [6] for more details on the Model 510.

We first convert our choice of RGB values to integers in the range [0, 255].

```
rgb =
  Floor[256 {{1 / 2, 3 / 4, 0}, {3 / 4, 1 / 2, 0}, {1 / 2, 0, 1 / 2}}];
```

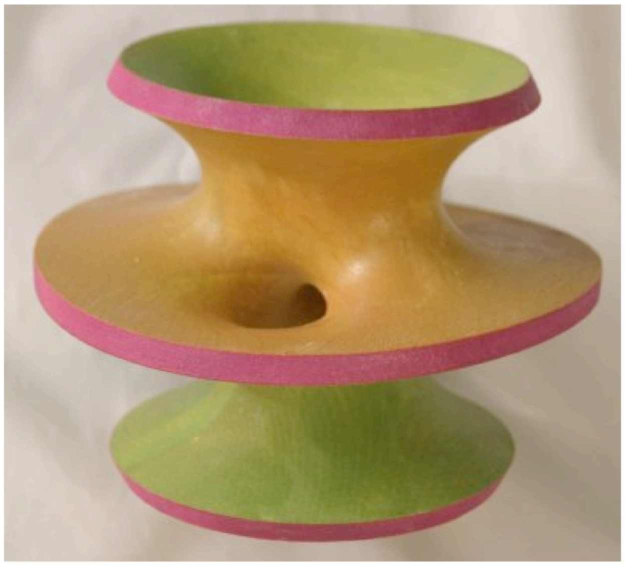
Now we produce a list `rw` in the correct format for input into the `ExportGraphics` function. Note its similarity in structure to `qw`.

```
rw = {zx⟦1⟧, Join[{{rgb⟦1⟧, zx⟦2⟧⟦1⟧}, {rgb⟦2⟧, zx⟦2⟧⟦2⟧}},
    Map[{rgb⟦3⟧, #} &, zx⟦2⟧⟦3⟧]]};
```

The PLY file corresponding to the surface pictured in Figure 3 is produced by invoking the `ExportGraphics` function as follows.

```
ExportGraphics["costa-8-24-24-50-50.ply", rw, "PLY"]
```

To obtain the solid model illustrated in Figure 4, a PLY file was created using the parameters $\delta = 1/8$, $\epsilon = \phi = 1/24$, $\rho = \pm 0.2$, and `MeshSize → {250, 250}`. The large mesh size ensured that the model would be smooth, and the rather large normal displacement $\rho = \pm 0.2$ ensured that it would be thick enough to avoid breakage during production. The resulting object is about eight inches in diameter.



▲ **Figure 4.** Photograph of a solid model of Costa's surface generated from a PLY file with parameters $\delta = 1/8$, $\epsilon = \phi = 1/24$, $\rho = \pm 0.2$, and `MeshSize → {250, 250}`.

■ Discussion

In this section, we put what was done here in perspective and outline some ways of generating Weierstrass data that may lead to new examples of minimal surfaces. These methods, together with other enhancements discussed in this section, may be incorporated in future versions of the *Minimal Surfaces* package.

□ Genus One Minimal Surfaces

First, it would be interesting to look for other examples of genus one minimal surfaces, possibly with more than three ends. The Weierstrass data for such a surface would be given by elliptic functions on $\mathbb{C}$. An obvious choice for Weierstrass data would be

$$f(z) = \wp_k(z;\,\Gamma),\;\; g(z) = \frac{C}{\wp_k'(z;\,\Gamma)}, \tag{21}$$

where C is a constant, and $\wp_k(z;\,\Gamma)$, for $k \geq 2$, is defined on $\mathbb{C}/\Gamma$ by the absolutely convergent series

$$\wp_k(z;\,\Gamma) := \frac{1}{z^k} + \sum_{\omega \in \Gamma_*} \left\{ \frac{1}{(z-\omega)^k} - \frac{1}{\omega^k} \right\}.$$

The case $k = 2$ is just that of Costa's surface, where $\Gamma = \mathbb{Z}[i]$ and $C = \sqrt{8\pi e_1}$. The convergence of this series is actually rather subtle for $k = 2$, and the difference in the summands cannot be separated. For $k \geq 3$, we may rewrite $\wp_k(z; \Gamma)$ as

$$\wp_k(z; \Gamma) = \frac{(-1)^{k-2}}{k!} \frac{d^{k-2}}{dz^{k-2}} \wp(z; \Gamma) + E_k(\Gamma),$$

where $E_k(\Gamma)$ is the *Eisenstein series of weight k*:

$$E_k(\Gamma) := \sum_{\omega \in \Gamma_*} \frac{1}{\omega^k}.$$

It was mentioned earlier that any elliptic function may be expressed as a rational function of $\wp(z; \Gamma)$ and its derivative $\wp'(z; \Gamma)$. We can achieve this for the Weierstrass data in equation (21) by repeatedly differentiating equation (9) and applying the above observations.

□ Higher Genus Minimal Surfaces

We now discuss two ways in which one may generate examples of minimal surfaces that have a topology other than that of a torus.

■ *Automorphic Functions on Hyperbolic Space*

Every compact, connected surface $\mathcal{S}$ has a *simply-connected universal covering space*, which we denote by $\mathcal{S}^*$. If $\mathcal{S}$ has no additional structure, we know that $\mathcal{S}^*$ must be topologically a sphere, which we take to be the extended complex plane $\mathbb{C} \bigcup \{\infty\}$, or an open subset of the plane, which we take to be $\mathbb{C}$. A surface $\mathcal{S}$ is said to have a *conformal structure* if, for every point $p \in \mathcal{S}$, one has a notion of the angle between any two tangent vectors based at p. If $\mathcal{S}$ is a simply connected region in $\mathbb{C}$, we have the following fundamental fact from complex analysis:

Riemann Mapping Theorem. *Any simply connected region in the complex plane $\mathbb{C}$ (other than $\mathbb{C}$ itself) is conformally equivalent to the unit disk $U := \{z \in \mathbb{C} \mid \|z\| < 1\}$.*

A *Riemann surface* is a surface endowed with a conformal structure. If $\mathcal{S}$ is a Riemann surface of genus at least two, the Riemann mapping theorem allows us to think of its universal covering space $\mathcal{S}^*$ as being the unit disk $U \subseteq \mathbb{C}$. In fact, following [7], a simply connected Riemann surface $\mathcal{S}^*$ is classified as being

- *elliptic* if it is conformally equivalent to the whole Riemann sphere $\mathbb{C} \bigcup \{\infty\}$
- *parabolic* if it is conformally equivalent to the finite plane $\mathbb{C}$
- *hyperbolic* if it is conformally equivalent to the unit disk U

Thus, any compact Riemann surface $\mathcal{S}$ is of the form $\mathcal{S} = \mathcal{S}^* / \Gamma$, where Γ is a discrete subgroup of a group $\mathcal{G}$ of conformal transformations on $\mathcal{S}^*$.

- If $\mathcal{S}^*$ is elliptic, then $\mathcal{G}$ is the full group $\mathcal{E}$ of linear fractional transformations $z \to (a z + b) / (c z + d)$ for complex constants a, b, c, and d with $a d - b c \neq 0$.
- If $\mathcal{S}^*$ is parabolic, then $\mathcal{G}$ is the group $\mathcal{P}$ of linear transformations $z \to a z + b$, for complex constants a and b.
- If $\mathcal{S}^*$ is hyperbolic, then $\mathcal{G}$ is the group $\mathcal{H}$ of linear fractional transformations of the form $z \to e^{i\theta}(z - a) \big/ (1 - \bar{a} z)$, where $\theta \in \mathbb{R}$, $a \in U$, and $\bar{a}$ is the complex conjugate of a. In this case, $\mathcal{H}$ is isomorphic to the matrix group $\mathrm{PSL}_2(\mathbb{R})$.

In accordance with the above, we say that $\mathcal{S}$ is elliptic if $\mathcal{S}^*$ is elliptic, $\mathcal{S}$ is parabolic if $\mathcal{S}^*$ is parabolic, etc. To simplify the discussion a bit, we sometimes use the term Weierstrass data to mean a pair of meromorphic functions $\left(\hat{f}, \hat{g}\right)$ on a compact Riemann surface $\mathcal{S}$. These functions lift naturally to a pair of Γ-invariant functions (f, g) on $\mathcal{S}^*$.

The catenoid is an example of a minimal surface that arises from Weierstrass data on an elliptic Riemann surface. The main subject of this paper has been Costa's surface, which arises from Weierstrass data on a parabolic Riemann surface (i.e., a torus). Minimal surfaces of higher topological type arise from Weierstrass data on Riemann surfaces of hyperbolic type. In order to find such data, we seek meromorphic functions (f, g) on the unit disk U that are invariant under a discrete subgroup $\Gamma \subseteq \mathcal{H}$. Such functions are referred to as *automorphic functions*. A fundamental domain of Γ acting on U can be specified by a subregion of U bounded by line segments and circular arcs. Thus, the rendering methodology used here should extend directly to the hyperbolic case. This leads us to pose the following:

Problem: *Use automorphic functions on the unit disk U as Weierstrass data for the construction of new examples of minimal surfaces.*

It should be noted that the author has not been able to find any references in the mathematical literature taking this approach to the construction of minimal surfaces. There is another approach, however, which is briefly discussed in the next section.

■ *Algebraic Curves in* $\mathbb{C}^2$

Riemann surfaces may also be realized as *affine algebraic curves* in $\mathbb{C}^2$, which are the solutions of polynomial equations in two complex variables of the form $P(z, w) = 0$. Such curves are not compact, but they can be compactified via an embedding into the complex projective plane $\mathbb{CP}^2$. From this perspective, Riemann surfaces can easily be identified with branched coverings of the complex plane $\mathbb{C}$ or the Riemann sphere $\mathbb{C} \bigcup \{\infty\}$. In equation (9), for example, we see that the Weierstrass data for Costa's surface essentially provides a complex parametrization of the elliptic curve

$$w^2 = 4 z^3 - g_2 z - g_3,$$

where $g_2 := -4 (e_1 e_2 + e_1 e_3 + e_2 e_3)$ and $g_3 := 4 e_1 e_2 e_3$.

In [8], Thayer describes a family of higher genus minimal surfaces, which he calls *Chen–*

Gackstatter–Karcher surfaces or *CGK surfaces*. In constructing these surfaces, he first identifies Riemann surfaces of the form

$$w^k - R(z) = 0,$$

where $R(z)$ is a rational function in z of a specific form. He then constructs minimal surfaces with the Weierstrass data $f(z) = 1$ and $g(z) = C\, w^{k-1} = C\, R(z)^{(k-1)/k}$, which are defined on a specific branch of the underlying Riemann surface. Here, the Gauss map g may be viewed as a multivalued function from $\mathbb{C}$ into the Riemann sphere $\mathbb{C} \bigcup \{\infty\}$. This leads us to pose the following:

Problem: *Is it possible to construct an explicit correspondence between automorphic functions on the unit disk U and affine or projective algebraic curves? In particular, can we find automorphic functions on U that are Weierstrass data for CGK surfaces?*

We note that Thayer's paper [8] contains numerous figures illustrating specific examples of CGK surfaces. The figures were generated using a program called MESH, which was written by James Hoffman. MESH is an adaptive mesh generation program that is specifically designed to generate vertex, edge, and face data for minimal surfaces. This data is generated via numerical integration of the component functions of the Weierstrass representation. MESH is a command-line application with a client-server architecture, the use of which requires some knowledge of C++ or FORTRAN programming.

□ Periodic Minimal Surfaces

Finally, we note that there are known examples of surfaces that are periodic in the sense that they are invariant under a discrete group of rigid motions in $\mathbb{R}^3$. A particularly interesting family of such surfaces is described in [10]. It would be interesting to search for new examples of such periodic minimal surfaces via the use of automorphic functions.

□ Enhancements to the *Minimal Surfaces* Package

We finish with a short list of possible enhancements to the *Minimal Surfaces* package.

- Improve performance by employing a method for selecting the density of vertex data on the basis of the surface metric given by equation (6). Alternatively, use the metric as a basis for an adaptive mesh generation algorithm.
- Improve performance by employing symmetries of surfaces to reduce the amount of computation.
- Include functionality for rendering other minimal surfaces, including the CGK surfaces described above.
- Incorporate functionality that permits the visualization of different coordinate systems on a minimal surface. Of particular interest would be principal curvature lines and asymptotic lines.

- Include functionality for rendering geodesics, including the ability to find and render closed geodesics.
- Include an export function that is suitable for use with the Persistence of Vision Ray Tracer (POVRAY) program.

Finally, we note that the *Minimal Surfaces* package was originally written using *Mathematica* Version 5.2. One reviewer of this paper indicated that the `DelaunayTri⋱angulation` routine in the Computational Geometry Package for *Mathematica* 5.2 is known to have poor complexity and has suggested the use of Martin Kraus's *Polygon Triangulation* package [11] instead. Martin Kraus's package seems to provide significantly better performance and may be incorporated into a future version of *Minimal Surfaces*. The same reviewer also pointed out that, in later releases of *Mathematica*, the built-in `Export` function supports both PLY and POVRAY file formats.

■ Acknowledgments

The idea for this paper grew out of a short course on computational topology the author attended from July 6–16, 2004. The course was taught by Herbert Edelsbruner (Duke University) and John Harer (Duke University), and held at the Institute for Mathematics and Its Applications (IMA) at the University of Minnesota. Thanks are due to Doug Arnold, the director of the IMA, for hosting the course, Herbert Edelsbruner for kindly providing access to his 3D printer, and Rachael Brady (Duke University) for her patient help in debugging the export function of the *Minimal Surfaces* package and successfully printing out the solid model shown in Figure 4. Thanks are also due to the reviewers of this paper, who provided many useful comments.

■ References

[1] R. S. Palais, "The Visualization of Mathematics: Towards a Mathematical Exploratorium," *Notices of the American Mathematical Society*, **46**(6), 1999 pp. 647–658. 3d-xplormath.org/DocumentationPages/VisOfMath.pdf.

[2] A. Gray, *Modern Differential Geometry of Curves and Surfaces with Mathematica,* 2nd ed., Boca Raton, FL: CRC Press, 1999.

[3] D. A. Hoffman and W. Meeks III, "A Complete Embedded Minimal Surface in $\mathbb{R}^3$ with Genus One and Three Ends," *Journal of Differential Geometry*, **21**, 1985 pp. 109–127.

[4] K. Chandrasekharan, *Elliptic Functions*, Grundlehren der Mathematischen Wissenschaften **281**, New York: Springer–Verlag, 1985.

[5] P. Bourke. "PLY-Polygon File Format." paulbourke.net/dataformats/ply.

[6] ZCorporation. www.zcorp.com.

[7] G. Springer, *Introduction to Riemann Surfaces,* 2nd. ed., New York: Chelsea, 1981.

[8] E. C. Thayer, “Higher-Genus Chen–Gackstatter Surfaces and the Weierstrass Representation for Surfaces of Infinite Genus,” *Experimental Mathematics*, **4**(1), 1995 pp. 19–39. www.emis.de/journals/EM/expmath/volumes/4/4.html.

[9] J. T. Hoffman, “MESH: A Program for Generating Parametric Surfaces Using an Adaptive Mesh,” 1996.

[10] V. R. Batista, “A Family of Triply Periodic Costa Surfaces,” *Pacific Journal of Mathematics*, **212**(2), 2003 pp. 347–370. citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.410.

[11] M. Kraus. “Polygon Triangulation.” library.wolfram.com/infocenter/MathSource/23/.

O. Michael Melko, “Visualizing Minimal Surfaces,” *The Mathematica Journal*, 2010.
dx.doi.org/doi:10.3888/tmj.12–6.

About the Author

Mike Melko is an assistant professor of mathematics at Khalifa University, Sharjah Campus. He received his Ph.D. from the University of California at Santa Cruz in 1989. His research interests are in the areas of differential geometry, mathematical physics, and computational mathematics.

O. Michael Melko
Khalifa University of Science, Technology and Research
Sharjah Campus
PO Box 573
Sharjah
UAE
Mike.Melko@kustar.ac.ae