MODULES FOR THE INVESTIGATION OF THE CENTRAL FAN QUESTION THROUGH NUMERICAL COMPUTATION

A Thesis by

Francis Le Nguyen

Bachelor of Science, Wichita State University, 2012

Submitted to the Department of Mathematics and Statistics
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

May 2015

MODULES FOR THE INVESTIGATION OF THE CENTRAL FAN QUESTION THROUGH NUMERICAL COMPUTATION

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Mathematics.

_____

Thomas K. DeLillo, Committee Chair

_____

Kirk Lancaster, Co-Chair

_____

Visvakumar Aravinthan, Committee Member

DEDICATION

To my brother Thomas Nguyen who was my first math teacher during my single digits. I appreciate the patience you had when I struggled to sit still and stay awake.

ACKNOWLEDGEMENTS

In completing this thesis, I couldn't have done this without the assistance of my colleagues, professors, the Mathematic's Department, LaWanda Holt-Fields, Shukura Bakari-Cozart, and the wonderful staff at McNair Scholar's Program here at Wichita State University.

I want to extend my gratitude to Dr. Thomas DeLillo, Dr. Kirk Lancaster, Dr. Alexander Bukhgeym, and Dr. Visvakumar Aravinthan for providing their expertise and time in assisting me in my research and the completion of my thesis.

Thank you Mrinal Nagrecha, Patric Mitchell, and Nathan Thompson for providing countless advice and encouragement. Finally, no words can describe my sincere gratitude towards William Ingle from his words of wisdom to his invaluable expertise throughout these past 5 years. We laughed, we argued, and we discussed, but I gained so much from this experience.

ABSTRACT

In this thesis, we continue the research of investigating the central fan question through numerical computation of minimal surfaces and the author develops a new conformal map in hopes of producing a more accurate and simplistic algorithm. The conformal mapping module adds greater power and accuracy to existing "toolboxes" designed for mathematical and technical problems that require those special transformations. In addition, the investigator numerically solves the Riemann-Hilbert problem and explores whether enough data for conjectures exist for central fans. With the use of MATLAB for numerical computations, we develop tools to analyze different capillary surfaces at reentrant corners to determine where a central fan does and does not exist. The author computes minimal surfaces from their Enneper-Weierstrass representation based on holomorphic functions, (f, g) , determined by their boundary conditions. Here f is determined by solving the Riemann-Hilbert problem defined by the geometry of the boundary, while we choose g to be the identity map applied to the conformal image of the boundary onto the unit circle. The understanding of capillary surfaces has applications outside the field of mathematics, especially in engineering. Computing the surfaces will allow us to consider the flow of fluids in zero gravity and other areas where surface tension plays an important role such as in instruments and components built in spacecraft. This study will enable engineers to test their designs by computing the capillary surfaces in many engineering applications such as DNA microarray processors, microthermal technologies, and fluid dynamics in zero gravity.

TABLE OF CONTENTS

Chapter 1

1 Introduction

1.1 Preliminary

There are many new manufacturing processes involving capillarity that have enabled us to produce modern materials, such as thin film applications in manufacturing and medicine. Yet we still do not completely understand the mathematics or physics behind the capillary phenomenon.

Capillary surfaces are non-linear and have interesting properties depending on the continuity or discontinuity of boundary data at isolated points. The behavior of fluids in space or other environments free of body forces such as gravity are often due to capillary effects. The study of capillarity saw a revival in the mid-twentieth century due to advances in engineering and surface chemistry, which led the way to increased safety in space exploration.

1.2 Past Research

In 1969, Paul Concus and Robert Finn [12] collaborated on the contact angle boundary problem in a container with a wedge corner involving two different fluid contact angles. Physically, the differing contact angles occur when the two walls are made of different materials. The results from this paper provided important details in the design of the fuel tanks that would be used for NASA spacecraft. Concus, who consulted with Lockheed, concluded that NASA's original fuel tank design wouldn't work correctly at zero gravity. This potentially saved NASA and Lockheed major expenses and possible catastrophe [14].

Later, in 1973, Michele Emmer [9] investigated a mathematical operator representing the energy of liquid in a capillary tube with two-dimensional cross section Ω, for any $h : \Omega \to \mathbb{R}$ of bounded variation L_v. Emmer published sufficient conditions for the existence and uniqueness for the minimum of the energy operator L_v of capillary surfaces for

domains with corners. This operator is given by

$$L_v(h) = \int_\Omega \sqrt{1 + |Dh|^2}dx + \int_\Omega h^2(x)dx + v\int_{\partial\Omega} h(x)dH_1 \tag{1.1}$$

where h is a function of bounded variation in Ω. As space exploration, thin film technology, and micro electronics expanded, mathematicians increasingly published research on capillary surfaces. Finn and Gerhardt [17] revisited the conditions Emmer stated for a non-convex corner and provided more general conditions for the existence of capillary surfaces. In 1980, Korevaar [10] proved that the solution of the capillary problem at a non-convex corner can be discontinuous even if the surface is bounded. Concus and Finn revisited capillary surfaces in wedges in 1996 (see [15]). Using the mean curvature equation with contact angle boundary condition

$$\operatorname{div}(Th) = \kappa h + \lambda \text{ in } \Omega, \quad Th \equiv \frac{\nabla h}{\sqrt{1 + |\nabla h|^2}} \tag{1.2}$$

$$\nu \cdot Th = \cos\gamma \text{ on } \partial\Omega \setminus \{A_1, ..., A_n\}. \tag{1.3}$$

these authors considered capillary surfaces in cylindrical containers for $\kappa = 0$ (zero gravity) to see if any solution could be found. One should note that the results and conjectures published by these authors were often in disagreement, and even highly respected mathematicians contradicted their own previous conclusions during the course of studying capillarity. To quote Concus and Finn [15] who revisited the capillary surface in a wedge problem from their 1993 paper [13]:

> In the interim, Keller, King, and Merchant studied again the question of a capillary surface $u(x, y)$ defined in a wedge, with (possibly) differing angles γ_1, γ_2 on the two sides. Statements given in that paper conflict basically with a previous result of ours [Concus and Finn] to which the authors refer and of which they assert a simplified proof. The new results of our present study disagree in turn

with those announced in for the corresponding general case. Our results are also to some extent at variance with the work by Vreeburg indicated above; these differences are discussed in our paper [15].

This illustrates the difficulty of the capillary problem as prominent mathematicians and scientists (e.g. Keller won the National Metal of Science in 1988) have incorrectly predicted the behavior of such surfaces.

Central fans were unknown until Kirk Lancaster [6] proved in 1985 that they exist for non-convex corners under certain conditions and that the interval of constant radial limit for a central fan will all exist. He also showed that outside of any fan of constant value the radial limit as a function of angle must be monotonic. With the collaboration of Alan Elcrat (see [1] and [2]), the two provided techniques to investigate the Dirichlet and contact angle boundary value problems for mean curvature equations.

In their 1996 publication, [7] Lancaster and Siegel proved that radial limits at the origin O of a bounded capillary surface do exist under weak conditions. Characterization of the radial limits behavior was also discussed. The paper demonstrates the presence of "fan domains", otherwise known as side and central fans that are attached to a corner where the surface height is constant at the contact angle. However, the paper lacked general sufficient conditions for the existence of central fans, which was noted by Finn [18].

Shi and Finn collaborated in 2004 [5] and in their article they examined a capillary surface provided as an example by Lancaster and Siegel (Example 2 pg. 184 [7]). They perturbed the boundary geometry by an arbitrarily small amount and proved that the central fan did not exist in the perturbed configuration. In mathematics one usually wants a situation in which a small change in the boundary condition results in a small change in the solution, or in this case, the minimal surface. Shi and Finn show that even the tiniest change in the boundary condition of this example resulted in the disappearance of the central fan altogether. While our understanding of central fans increases incrementally, this paper illustrated a major stumbling block in the search for sufficient conditions for the existence of central fans. That

is, any sufficient conditions for the existence of central fans should be "unstable" as illustrated by Shi and Finn [5].

The most recent publication concerning the existence of the central fan in capillary surfaces is a dissertation written by Ammar Khanfer under Kirk Lancaster's supervision [3]. Khanfer proved stability of the minimal surface solution of mean curvature equations (1.2) and (1.3) with respect to the contact angle. Khanfer provides examples to illustrate the existence and stability of the central fan phenomenon under conditions on the geometry of the system. In this paper, we strict our attention to the reentrant corners as in Khanfer's example 4.1 [3]. This gives us a reasonable expectation of designing a convergent scheme for computing the solution to (1.2) and (1.3); providing an investigative tool for examining changes in the capillary surface with respect to the contact angle for a given reentrant corner's geometry.

Chapter 2

2 The Central Fan Question

2.1 Why is this research important?

In order to design systems that operate in free fall or interplanetary space, one needs to understand how fluids act in containers under microgravity conditions. Such conditions occur in spacecraft, where surface tension surpasses the significant forces of gravity in systems operating at very small scales.

In 2008, Athanassenas and Lancaster [8] presented the central fan question to the mathematical community. They presented examples of capillary graphs, which are continuous or have central fans at reentrant corners. They further show that while continuity is a necessary consequence of the existence of a central fan under certain conditions, in general, sufficient conditions for the existence of a central fan are unknown. Athanassenas and Lancaster define the central fan question as follows:

(i) Is the solution h of (1.2) and (1.3) continuous at a corner?

(ii) Are sufficient conditions for the existence of central fans unstable, as illustrated in Shi and Finn's paper [5]?

(iii) What mathematical techniques will be employed or invented to prove sufficient conditions?

(iv) What are the geometric and analytic aspects of central fans?

(v) What insights will answering these problems lend to boundary value problems such as general quasilinear elliptic partial differential equations?

Lancaster considers these to be the most important open questions in the mathematical theory of capillarity. Robert Finn remarks [14] that this research was central to the correct

design of fuel tanks in the Gemini and Apollo spacecraft program. He goes on to explain that the solution of the central fan question will have growing importance in microfluidics (e.g. lab on a chip technology, microthermal technologies, DNA microarray processors, polymerase chain reaction and high throughput sequencing, electrowetting, etc.). It is in these small scale technologies that surface tension becomes the dominant force over gravity, electromagnetic force, and chemical bonding. We wish to predict the behavior of capillary surfaces using mathematical theory. However, the capillary equation is a highly nonlinear partial differential equation and the contact angle boundary condition is also nonlinear. Solving a capillary surface problem is a fundamentally difficult problem as described in the previous section, which has occupied mathematicians for centuries.

2.2 Present Research

To resolve the central fan question and promote better understanding of these technologies we need reliable computational tools for modeling capillary surfaces. Such tools provide a method to investigate a wide variety of surfaces in pursuit of answering the central fan question. These tools would further provide engineers with the ability to simulate designs free of the expense of prototype construction and extraterrestrial experimentation.

In his master's thesis, Mitchell [4] outlines a practical design for the necessary computational software. Mitchell's design required an understanding of how minimal surfaces are represented. Enneper and Weierstrass showed that, for a domain Ω in the complex plane, an arbitrary meromorphic function g in Ω and a holomorphic function f in Ω for which at each pole of g in Ω, f has a zero of order at least twice the order of the pole g, that the functions

$$\phi_1 = \frac{1}{2} f(1 - g^2), \phi_2 = \frac{i}{2} f(1 + g^2), \phi_3 = fg \tag{2.1}$$

satisfy

$$\phi_1^2 + \phi_2^2 + \phi_3^2 = 0. \tag{2.2}$$

Conversely, every single triple set of holomorphic functions in Ω that satisfies (2.2) may be represented in the form (2.1) for appropriate choices of f and g except $\phi_1 = i\phi_2$, $\phi_3 = 0$. Furthermore, every simply connected minimal surface in $\mathbb{R}^3$ can be represented in the form

$$x_k(z) = \Re \int_0^z \phi_k(s)\, ds + c_k,\ k = 1, 2, 3. \tag{2.3}$$

Here the ϕ_k are defined by (2.2) for choices f and g that satisfy the conditions for (2.1). This allows us to represent our capillary surfaces by choosing an appropriate meromorphic function g and a holomorphic function f.

A capillary surface as explained by Mitchell [4] is the interface separating two non-miscible fluids (one of the two fluids may be a gas). We consider a liquid fractionally filling a vertical cylinder with cross sectional profile Ω with a non-convex, or reentrant, corner. If, for, example the material constituting the cylinder walls of the non-convex corner consists of different materials, such that the contact angles of the fluid with the cylinder walls differ, then we would want to determine the nontrivial behavior of the surface at the corner. This problem becomes one of finding a function $h \in C^2(\Omega)$ that satisfies (1.2) and (1.3) where λ is a Lagrange parameter determined by the geometry of the cylinder; κ is a "capillary" constant, $\kappa = \frac{\rho g}{\sigma}$; ρ is the change in density across the surface, g is gravity, and σ is the surface tension; $\gamma = \gamma(s)$ is the contact angle of a liquid with a capillary wall on the boundary of Ω; and s is arc length. The surface $h = f(x, y)$ describes the steady state liquid-gas interface in a vertical cylinder with cross section Ω. Our current investigation concerns itself primarily with microgravity conditions (when $\kappa \approx 0$).

Without loss of generality we place the non-convex corner at the origin of the coordinate system and consider the boundary to be piecewise smooth. The tangent rays at the corner form an angle of magnitude 2α, and the equation of the tangent lines are given by $\theta = \pm\alpha$. A non-convex corner occurs when $\frac{\pi}{2} < \alpha < \pi$. We parameterize $\partial\Omega$ near the origin by arc length s, such that $(x(0), y(0)) = 0$, the origin, $y(s) > 0$ for $s > 0$ and $y(s) < 0$ for $s < 0$.

We assume that the contact angle $\gamma = \gamma(x, y)$ as a function of the boundary $\partial\Omega$ near the origin and the limits

$$\gamma_1 = \lim_{s\downarrow 0} \gamma(x(s), y(s)) \text{ and } \gamma_2 = \lim_{s\uparrow 0} \gamma(x(s), y(s)) \tag{2.4}$$

both exist. We denote the radial limits of a bounded solution h to (1.2) and (1.3) by

$$Rh(\theta) = \lim_{r\to 0^+} h(r\cos(\theta), r\sin(\theta)) \tag{2.5}$$

$$Rh(+\alpha) = \lim_{s\downarrow 0} h(x(s), y(s)) \tag{2.6}$$

and

$$Rh(-\alpha) = \lim_{s\uparrow 0} h(x(s), y(s)) \tag{2.7}$$

Thus $Rh(+\alpha)$ is defined by the limit taken along the boundary of Ω.

Lancaster and Siegel [7] proved the existence of the radial limits, $Rh(\theta)$, of a bounded solution h to (1.2) and (1.3) where $\partial\Omega$ is sufficiently smooth and the contact angle is bounded away from 0 and π. They also proved that Rh behaves in one of the following ways:

(i) There exist α_1, α_2 so that $-\alpha \le \alpha_1 < \alpha_2 \le \alpha$ and $Rh(\theta)$ is constant on $[-\alpha, \alpha_1]$ and $[\alpha_2, \alpha]$, and strictly monotonic on $[\alpha_1, \alpha_2]$.

(ii) There exist $\alpha_1, \alpha_L, \alpha_R, \alpha_2$ so that $-\alpha \le \alpha_1 < \alpha_L < \alpha_R < \alpha_2 \le \alpha$, $\alpha_R = \alpha_L + \pi$, Rh is constant on $[-\alpha, \alpha_1], [\alpha_L, \alpha_R]$, and $[\alpha_2, \alpha]$, and strictly monotonic on $[\alpha_1, \alpha_L]$ and strictly monotonic in the opposite direction on $[\alpha_R, \alpha_2]$.

Thus they proved that under the right conditions central fans exist over the interval $[\alpha_L, \alpha_R]$ and the interval is of length π. In addition Lancaster and Siegel [7] proved that in the case of a symmetric capillary profile Ω if h is discontinuous at the corner then (i) holds and there is a central fan. Thus for a central fan to exist, h must be continuous at the corner. Shi and Finn [5] constructed a nonsymmetric domain, changing the profile Ω well away from

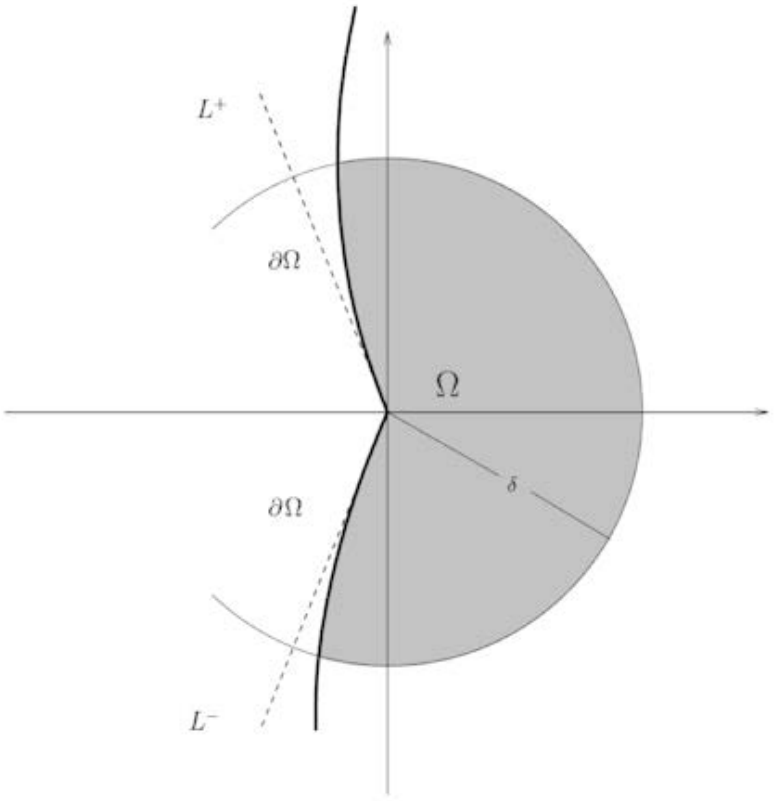

Figure 2.1: Radial Limits

the corner, and obtained a solution that was discontinuous at the corner. Therefore the symmetry condition was required for continuity and the geometry of the capillary surface affects the behavior of the surface at the corner. This complexity indicates that further research will be aided by the capacity to computationally construct capillary surfaces under a wide variety of conditions and geometries. Computing the surface $h(x, y)$ to determine the conditions for which central fans exist is the purpose of this investigation.

The computational task of constructing capillary surfaces is broken into three modules by Mitchell [4], and based on the mathematical framework specified by Lancaster and Athanassenas [8].

- Module 1: Composition of several conformal maps $\psi : E \to \mathbb{D}$ and $\psi^{-1} : \mathbb{D} \to E$.
- Module 2: Solving the Riemann-Hilbert boundary-value problems for continuous and discontinuous functions.
- Module 3: Plot solution using the Enneper-Weierstrass Representation based on functions, (f, g), determined by their boundary value conditions.

Once a pair of contact angles are chosen and a geometry selected (continuous or discontinuous capillary), one constructs the stereographic projection of the image of the Gauss map of the capillary surface. Although the solution surface to (1.2) and (1.3) is not yet known, choosing the contact angle pair and geometry allows one to compute the stereographic projection of a suitable capillary domain onto the upper left quadrant of the unit disk of the complex plane.

The required meromorphic function g turns out to be the conformal map of the projection onto the unit disk ([4], [8]). This is due to the necessity of determining the meromorphic function g and analytic function f of (2.1) in the Weierstrass representation of the capillary surface, such that the product fg^2 is holomorphic. This module has been developed for each function f, g and has been tested using well-known minimal surfaces, e.g. catenoid, helicoid, Scherk's surface, and Enneper's Surface. Once modules 1 and 2 are completed and refined for computing function f and letting g be the identity map, the third module will use these

functions to compute the minimal surface (see Appendices G):

$$x(w) = x_0 + \Re\frac{1}{2}\int_0^w f(t)(1 - g^2(t))dt \tag{2.8}$$

$$y(w) = y_0 + \Re\frac{i}{2}\int_0^w f(t)(1 + g^2(t))dt \tag{2.9}$$

$$z(w) = z_0 + \Re\int_0^w f(t)g(t))dt. \tag{2.10}$$

Chapter 3

3 Conformal Mapping Methods

3.1 Conformal Map Package Introduction

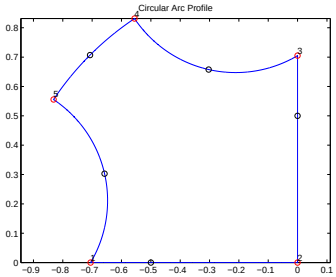

Figure 3.1: Domain E

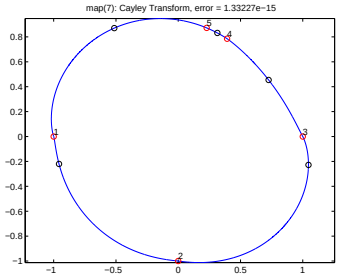

Figure 3.2: Domain J

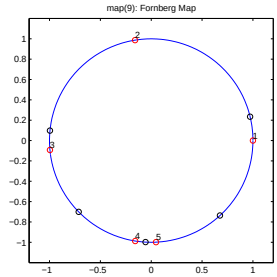

Figure 3.3: Unit Disk

We'll define the circular arc polygon and its interior E to be the image of the stereographic projection of the Gauss map of the minimal surface; and the simple closed Jordan curve and its interior as J. We construct a conformal map $C : E \to J$ such that $C(E) = J$ where $\partial J \in \mathrm{C}^2[0, 2\pi]$, $h : [0, 2\pi] \to \partial J$, and $C^{-1}(\mathrm{Int}(J)) = \mathrm{Int}(E)$. Take note that the Map C and its inverse are not guaranteed to be conformal on the exteriors of E or J. Map C is generally a composition of several conformal maps, being conformal on the interior and continuous up to the boundary, $C(\partial E) = \partial J$.

Also, the map from J to the unit circle is the inverse composition of a Möbius transform, M, and conformal map $F : \mathbb{D} \to J$ using the Fornberg method, [19], $(F \circ M)^{-1} = M^{-1} \circ F^{-1}$. Whereas, the map from the unit circle to J is the composition of a Möbius transform with the Fornberg method. To compute F^{-1} we use a Newton's method called, "Halley's Method", on the complex plane $\mathbb{C}$ and $M^{-1} : \mathbb{D} \to \mathbb{D}$. Respectively, $M : \mathbb{D} \to \mathbb{D}$ and $F : \mathbb{D} \to \mathrm{J}$; $F(\mathrm{Int}(\mathbb{D})) \to \mathrm{Int}(J)$, is the interior of the Map J. However $F(\partial\mathbb{D}) \to \mathbb{C}$ such that $\partial\mathbb{D}$ maps to a neighborhood of ∂J because we determine a periodic cubic spline approximation of the boundary correspondence function h. The periodic cubic spline interpolant and its first three derivatives are used in the Fornberg algorithm [19].

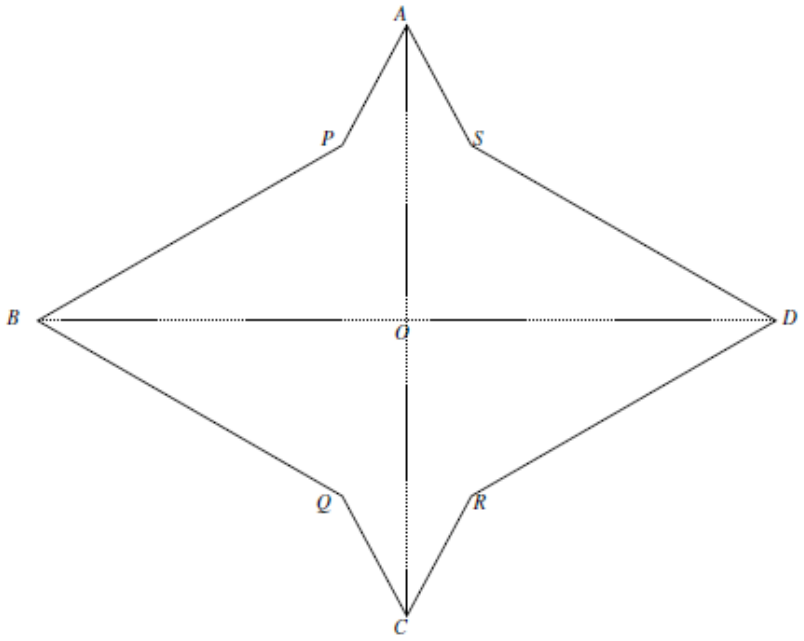

Figure 3.4: The Domain Ω

3.2 Module 1a: Stereographic Projection of the Gauss Map

The construction of the capillary graph, comes from the domain Ω being symmetric, centered at the origin, and starshaped as shown in Figure (3.4) as explained in Athanassenas and Lancaster's paper ([8], section 3). Notice that in each quadrant, $\partial\Omega$ has a reentrant corner. This is our main focus as we wish to investigate the behavior of the capillary surface at these corners. For the conformal mapping and by the recommendation of the authors ([8], section 3), we'll be focusing on the reentrant corner in the second quadrant. The authors note the following:

> We assume $\cos(\gamma(x, y))$ is an odd function of x and y...We note that the symmetry condition does not hold at P (nor at Q, R, and S) if $\gamma_1 \neq \gamma_2$; specifically, h will not be symmetric with respect to the line through P that bisects angle APB in Figure 3.4 if $\gamma_1 \neq \gamma_2$. We will restrict our attention to Ω_0, specifically the quarter of the domain in the second quadrant and require the Gauss map of the surface $z = h$ over Ω_0 to be injective. This requirement that the Gauss map be

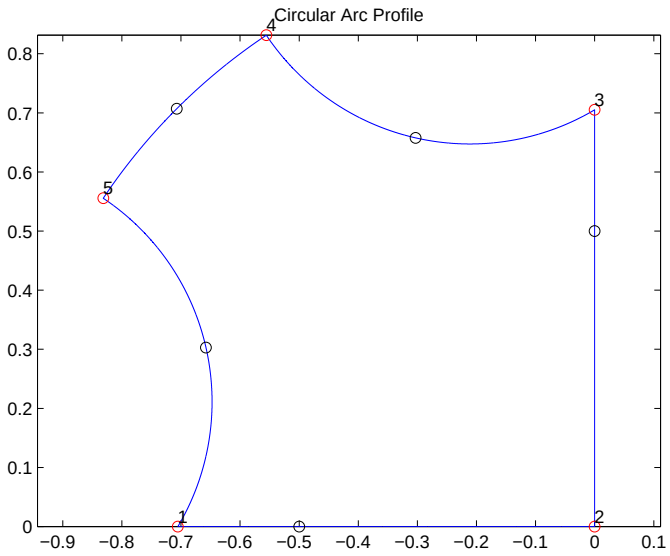

Figure 3.5: (Domain E) Circular Arc Polygon

injective is critical for our construction using the Weierstrass (f, g)-representation of minimal surfaces; however, minimal surfaces with injective Gauss maps over nonconvex domains are in some sense rare and this requirement will pick out (depending on γ) the particular domain Ω in which we will work.

To create the stereographic projection of the Gauss map in Figure (3.5), note this is the discontinuous case for capillary surfaces at reentrant corners and we'll use the conditions written in Athanassenas and Lancaster's paper ([8], section 4 and 6). Since this particular region is the discontinuous case, we are given $\gamma_1 \geq \delta$, $\gamma_2 \geq \delta$, and $\gamma_1 + \gamma_2 < \frac{\pi}{2} - 2\delta$. With these conditions we can substitute some values of γ_1, γ_2, and δ to create the circular arcs of the polygon w_1 and w_2. Also, we can find the vertices t_1, t_2, t_3, and t_4 by:

$$t_1 = e^{(\pi-\delta-\gamma_1)i} = w_1 + r_1 e^{(\frac{\pi}{2}-\delta-\gamma_1)i}$$

$$t_4 = e^{(\frac{\pi}{2}+\delta+\gamma_2)i} = w_2 + r_2 e^{(-\pi+\delta+\gamma_2)i}$$

$$t_2 = w_1 + r_1 e^{i\tau_{1B}} \text{ for some } \tau_{1B} \in [-\frac{\pi}{2}, -\delta)$$

$$t_3 = w_2 + r_2 e^{i\tau_{2B}} \text{ for some } \tau_{2B} \in (\delta - \frac{\pi}{2}, 0].$$

To determine our E, let $B_1 = \{w \in \mathbb{C} : |w| < 1\}$, $Q_1 = \{w \in B_1 : \mathrm{Re}(w) < 0,\ \mathrm{Im}(w) > 0\}$ and set

$$E_0 = \{w \in Q_1 : |w - w_1| > \tan(\gamma_1), |w - w_2| > \tan(\gamma_2)\}, \text{where}$$

Let $w_1 = u_1 + iv_1 = -\cos(\delta)\sec(\gamma_1) + i\sin(\delta)\sec(\gamma_1)$ and $c_1 = \{w : |w - w_1| = \tan(\gamma_1)\}$, where $r_1 = \tan(\gamma_1)$ and $t_2 = x + i \cdot 0$. Given $t_2 = w_1 + r_1 e^{i\tau_{1B}}$, t_2 is also equivalent to:

$$t_2 = w_1 + r_1 e^{i\tau_{1B}} = -\cos(\delta)\sec(\gamma_1) + i\sin(\delta)\sec(\gamma_1) + \tan(\gamma_1)\big(\cos(\tau_{1B}) + i\sin(\tau_{1B})\big)$$

$$= -\cos(\delta)\sec(\gamma_1) + \tan(\gamma_1)\cos(\tau_{1B}) + i\big(\sin(\delta)\sec(\gamma_1) + \tan(\gamma_1)\sin(\tau_{1B})\big).$$

Since $t_2 = x + i \cdot 0$, then we have

$$t_2 = x + i \cdot 0 = -\cos(\delta)\sec(\gamma_1) + \tan(\gamma_1)\cos(\tau_{1B}) + i \cdot 0.$$

Let us substitute $\cos(\tau_{1B})$ for $(1 - \sin^2 \tau_{1B})^{1/2}$, where

$$\sin(\tau_{1B}) = i \cdot 0 = -\frac{\sin(\delta)\sec(\gamma_1)}{\tan(\gamma_1)} = -\sin(\delta) \cdot \frac{1}{\cos(\gamma_1)} \cdot \frac{\cos(\gamma_1)}{\sin(\gamma_1)} = -\frac{\sin(\delta)}{\sin(\gamma_1)}.$$

By finding $\sin(\tau_{1B}) = -\frac{\sin(\delta)}{\sin(\gamma_1)}$, we have that

$$\cos(\tau_{1B}) = (1 - \sin^2 \tau_{1B})^{1/2} = (1 - \frac{\sin^2(\delta)}{\sin^2(\gamma_1)})^{1/2} = \left(\frac{\sin^2(\gamma_1) - \sin^2(\delta)}{\sin^2(\gamma_1)}\right)^{1/2}.$$

Now, we have

$$\begin{aligned} t_2 = x &= -\cos(\delta)\sec(\gamma_1) + \tan(\gamma_1)\left(\frac{\sin^2(\gamma_1)-\sin^2(\delta)}{\sin^2(\gamma_1)}\right)^{1/2} \\ &= -\cos(\delta)\sec(\gamma_1) + \frac{\sin(\gamma_1)}{\cos(\gamma_1)}\left(\frac{\sin^2(\gamma_1)-\sin^2(\delta)}{\sin^2(\gamma_1)}\right)^{1/2} \\ &= -\cos(\delta)\sec(\gamma_1) + \frac{1}{\cos(\gamma_1)}\sqrt{(\sin^2(\gamma_1)-\sin^2(\delta)} \\ &= -\cos(\delta)\sec(\gamma_1) + \sec(\gamma_1)\sqrt{(\sin^2(\gamma_1)-\sin^2(\delta)} \\ &= \sec(\gamma_1)\left(-\cos(\delta) + \sqrt{(\sin^2(\gamma_1)-\sin^2(\delta)}\right). \end{aligned}$$

Now to find t_3, we let $w_2 = u_2 + iv_2 = -\sin(\delta)\sec(\gamma_2) + i\cos(\delta)\sec(\gamma_2)$ and $c_2 = \{w : |w - w_2| = \tan(\gamma_2)\}$. Given $t_3 = 0 + iy$ and $r_2 = \tan(\gamma_2)$,

$$t_3 = w_2 + r_2 e^{i\tau_{2B}} = -\sin(\delta)\sec(\gamma_2) + i\cos(\delta)\sec(\gamma_2) + \tan(\gamma_2)(\cos(\tau_{2B}) + i\sin(\tau_{2B}))$$

$$= -\sin(\delta)\sec(\gamma_2) + \tan(\gamma_2)\cos(\tau_{2B}) + i(\cos(\delta)\sec(\gamma_2) + \tan(\gamma_2)\sin(\tau_{2B}))$$

Since $\sin(\tau_{2B}) = -(1-\cos^2(\tau_{2B}))^{1/2}$, we shall solve for $\cos(\tau_{2B})$

$$0 = -\sin(\delta)\sec(\gamma_2) + \tan(\gamma_2)\cos(\tau_{2B})$$

$$\frac{\sin(\delta)\sec(\gamma_2)}{\tan(\gamma_2)} = \cos(\tau_{2B}) = \frac{\sin(\delta)}{\sin(\gamma_2)}.$$

So,

$$\sin(\tau_{2B}) = -(1-\cos^2(\tau_{2B}))^{1/2} = -\left(1 - \frac{\sin^2(\delta)}{\sin^2(\gamma_2)}\right)^{1/2} \text{ or } -\left(\frac{\sin^2(\gamma_2)-\sin^2(\delta)}{\sin^2(\gamma_2)}\right)^{1/2}.$$

Then substituting $\tan(\gamma_2) = \frac{\sin(\gamma_2)}{\cos(\gamma_2)}$,

$$t_3 = iy = i\left(\cos(\delta)\sec(\gamma_2) - \frac{\sin(\gamma_2)}{\cos(\gamma_2)}\left(\frac{\sin^2(\gamma_2)-\sin^2(\delta)}{\sin^2(\gamma_2)}\right)^{1/2}\right)$$

$$= i\left(\cos(\delta)\sec(\gamma_2) - \sec(\gamma_2)\sqrt{\sin^2(\gamma_2) - \sin^2(\delta)}\right)$$

$$= i\sec(\gamma_2)\left(\cos(\delta) - \sqrt{\sin^2(\gamma_2) - \sin^2(\delta)}\right)$$

For our particular region E Figure (3.5), we choose γ_1 and γ_2 to be $\frac{\pi}{8}$, while δ is $\frac{\pi}{16}$.

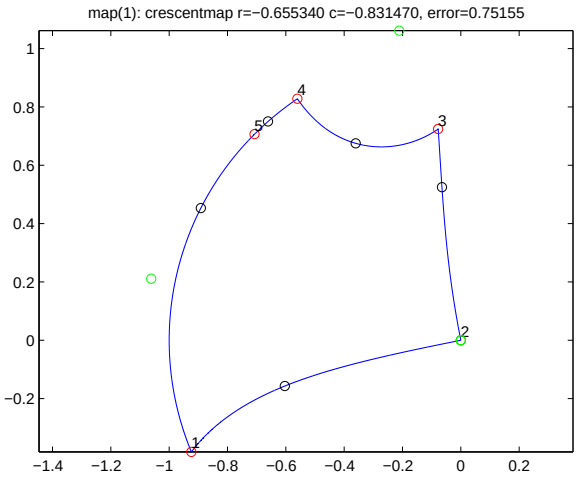

Figure 3.6: Cresent Map from Henrici vol 3

3.3 Module 1b: Forward Conformal Map $\psi : E \to \mathbb{D}$

In Figure (3.6) and (3.7), we'll be using Henrici corner removal method called, "The crescent map", ([11] page 331, chapter 16 example 4). Refer to the Matlab code in the Appendices B for where we define most of the variables for the crescent map class. This crescent map class comprises several conformal maps. First let K be a disk that intersects the boundary of E, the unit disk, and $D_1 \equiv \frac{E}{K}$. The crescent map takes any simply connected sub-domain of D_1 onto E. We define $c = e^{i\gamma}$ and $\overline{c} = e^{-i\gamma}$ as points that intersect K and E. Also, we define the point $-r$ and $0 < r < 1$ in E where K intersects the negative real axis.

1. We take a rotation of original unit vector m_r to the negative real axis. Henrici's crescent mapping method assumes that m_r lies on the negative real axis so that the rotation aligns the image of $-r$ to -1.

2. Next is the Moebius Transform:

   $z \mapsto z' = \frac{z-c}{z-\overline{c}} \cdot \frac{r+\overline{c}}{r+c}$ this maps D_1 onto the wedge $0 <$ arg $z' < \alpha$, where $c = -e^{-i\alpha}\frac{1+e^{i\alpha}}{1+e^{-i\alpha}}$ and $\alpha = \text{angle}(1')$. α is the image of 1 under the first Moebius transformation, such that $\alpha = \text{angle}(\text{moebius}(0))$.

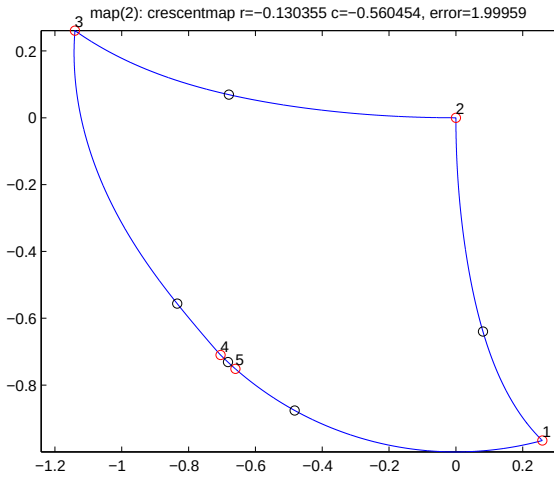

Figure 3.7: Cresent Map

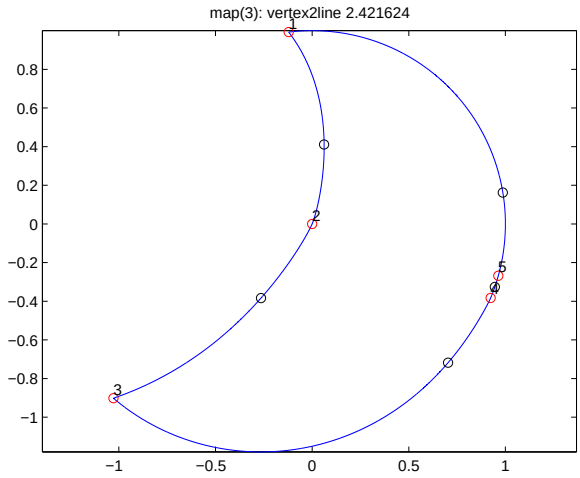

Figure 3.8: Power Map

3. The power map:

   $z' \mapsto z'' = (z')^{\pi/a}$ this maps the wedge to the upper-half plane.

4. Another Moebius Transform:

   $z'' \mapsto w := (z'' - e^(b))/(z'' - e^(- b))$, this maps the upper-half plane onto the unit disk where $b = \arg(0') * pi/a$

5. Lastly a rotation map $w \mapsto w = w * e^{-b}$ that maps $z = -r$ to $w = -1$ on the real axis.

After popping two of the circular arcs in our E map, we use a power map from complex analysis in our matlab routine vertex2line.m (see Appendices C). Here vertex2line.m encapsulates a map to open a vertex in the boundary of a simply connected set on the complex plane to remove the vertex from the boundary. Since our region is a simply connected region that does not contain 0 and $\beta \in \mathbb{C}$, we will define $z^{\frac{\pi}{\beta}} = (re^{i\beta})^{\frac{\pi}{\beta}} = w$ where $\log z$ can be calculated by passing a point $z_0 \in \Omega$. As result z^α is analytic on Ω.

Next we apply the Moebius Transformation to Figure (3.8), we get the Figure (3.9). We take vertices 1, 2, and 3 that are labeled in our exponential map E in Figure 3.5 and map those points to 0, 1, and ∞, respectively (see Appendices B).

Here we use our vertex2line.m class again from Figure (3.8) to open up the vertex angle at the origin to pi radians. If the vertex angle still have some curvature, then we continue to

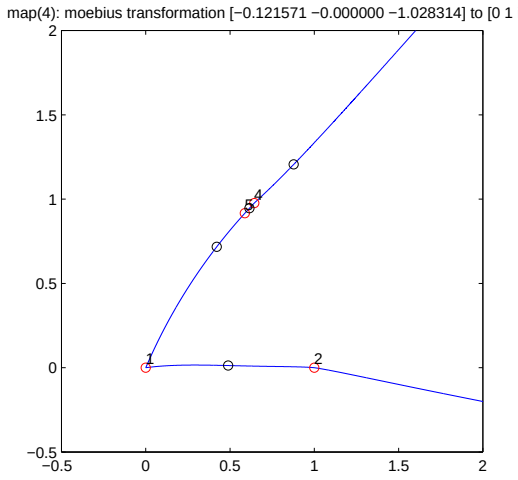

Figure 3.9: Moebius Transformation 1

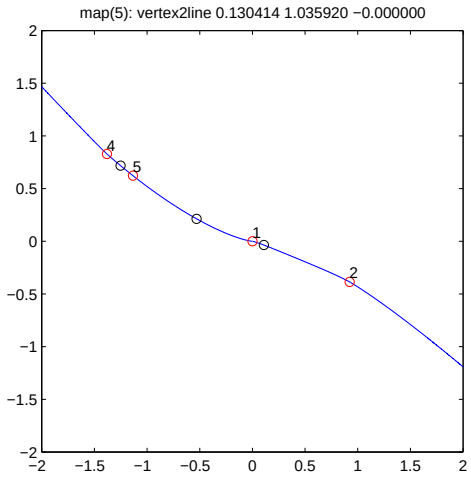

Figure 3.10: Power Map on vertex 1

open the vertex angle until the chosen vertex angle is as close to pi radians. Then we align the graph of the line along the x-axis and centered at the origin as shown in Figure (3.11).

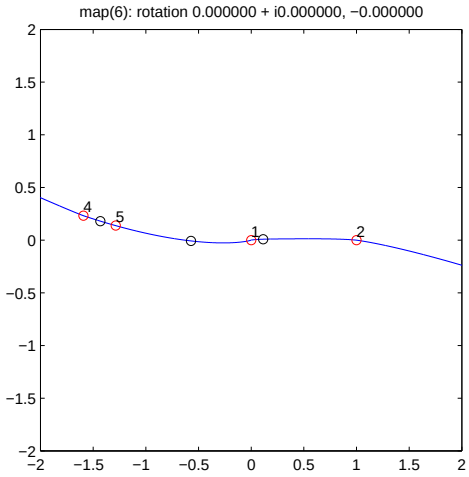

Figure 3.11: Rotation map

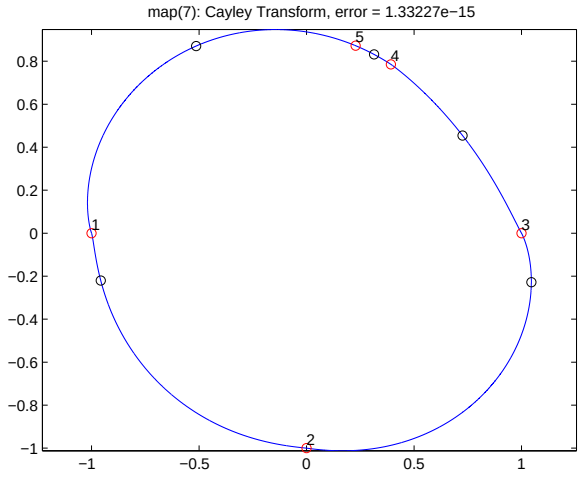

Figure 3.12: Cayley's Transform

We use the Cayley's transform to map our line through infinity to a bounded smooth curve by mapping the points 0, 1, and ∞ from Figure (3.11) to the points -1, $-i$, and 1 respectively on Figure (3.12). This map implements the transformation of the complex plane onto itself given by $w : z \to \frac{z-i}{z+i}$ (see Appendices E).

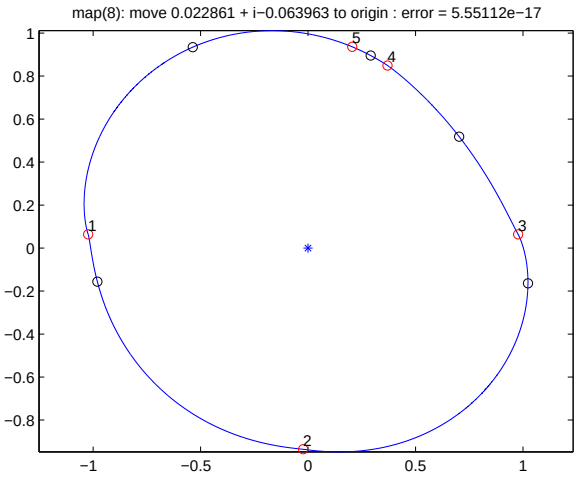

Figure 3.13: Center of the smooth curve

In Figure (3.13) we make sure our smooth curve is centered at the Origin to apply the next mapping algorithm called, "Halley's Method". The Halley's method is a root-finding algorithm. Suppose we needed to find a root of a function $f(x)$, such that x satisfies $f(x) = 0$. We give a guess of the root, x_n, then we want to iterate on our guess to approximate to the root as close as possible. With this method, we are able to graph our domain J to the unit circle (see Appendices F).

3.4 Module 1c: Inverse Map $\psi^{-1} : \mathbb{D} \to E$

With the unit circle, we use a conformal map called, "Fornberg". The Fornberg map is from Thomas DeLillo's Confft Package with collaboration with his student Lianju Wang see [19], which uses a spline fit on a simply connected smoothed curve J and maps an approximation of the points on the smooth curve J to a unit circle.

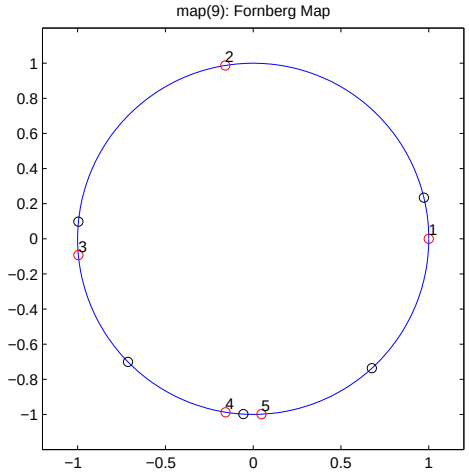

Figure 3.14: Fornberg Map

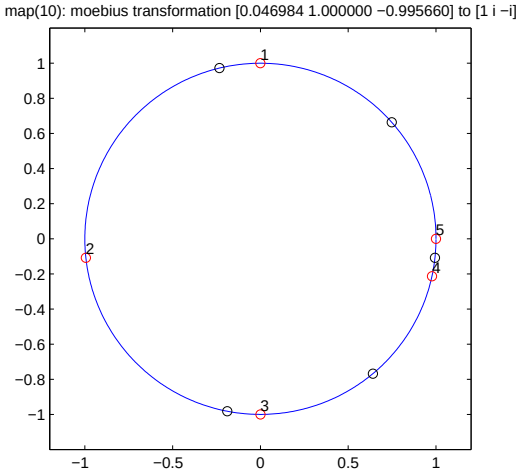

Figure 3.15: Moebius Transformation 2

In Figure (3.15) we apply the moebius transformation again, but this time targeting the vertices 1, 3, and 5 onto i, $-i$, and 1, respectively (see Appendices A). Once we have the map $C^{-1} : \mathbb{D} \to J$, then we have the Figure (3.17).

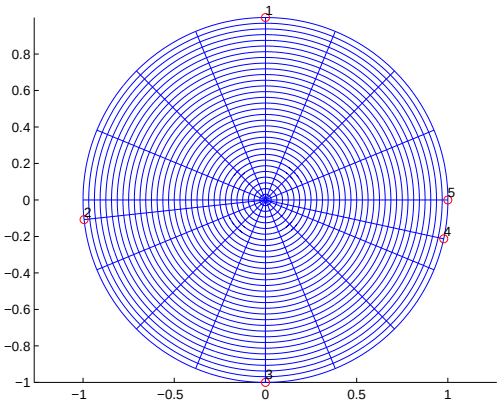

Figure 3.16: Radial Lines

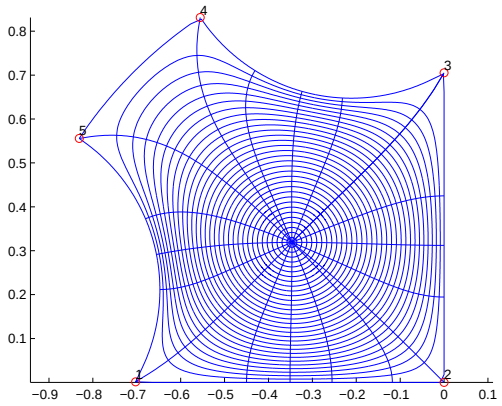

Figure 3.17: Inverse Map of E

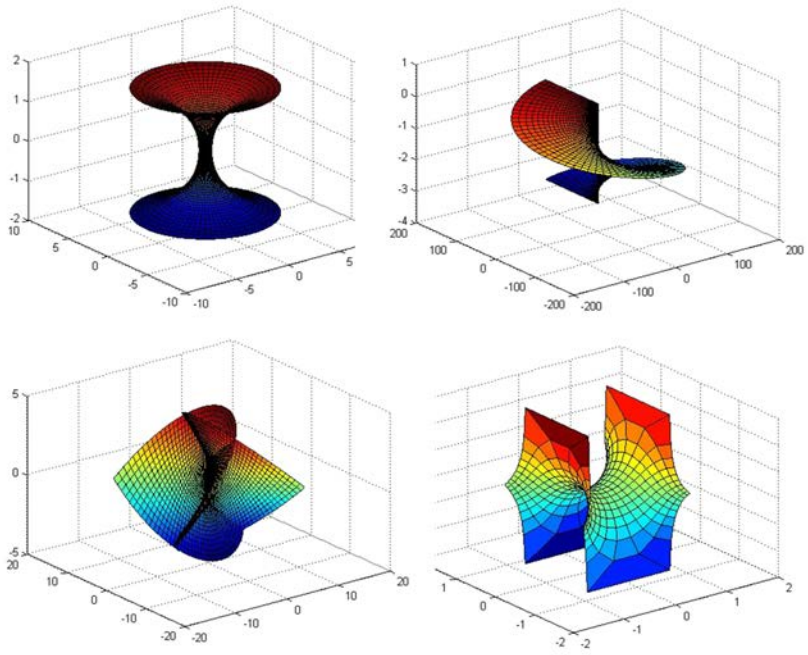

Figure 3.18: Examples of Minimal Surfaces

3.5 Module 3: Enneper-Weierstrass Representation

Once one has a bijective conformal map $\psi : E \to \mathbb{D}$ that is piecewise continuous on the boundary we may determine the Enneper-Weierstrass representation functions f and g. By construction, $g : \overline{E} \to \overline{E}$ is the identity map and we must find an analytic function f on the interior of E and satisfies the Riemann-Hilbert boundary condition

$$\mathrm{Re}(f(z)G(z)) = 0, \text{ on } \partial E. \tag{3.1}$$

Athanassenas and Lancaster showed that the boundary function G is a piecewise continuous function on ∂E determined by the boundary conditions of the minimal surface equations (2.8)-(2.10) and the geometry of E [21]. The boundary function $G : \partial E \to \mathbb{C}$ depends on the geometry of the reentrant corner, the contact angles, and the behavior of the fluid at the reentrant corner. Recall that two cases exist for the behavior of fluid when a central fan exists; the continuous case and the discontinuous case. The focus in this paper is the

discontinuous case (specifically case 5 in our Figure (3.5) from other cases listed in [8] page 225 -226). The geometry mentioned in Athanassenas and Lancaster is given by [8]:

$$G(w) = \begin{cases} a_1(u,v) + ib_1(u,v) = ie^{i\delta}(w - w_1)(e^{-2\delta i} - w^2) & w \in \sigma_1 \\ a_2(u,v) + ib_2(u,v) = e^{-i\delta}(w - w_2)(e^{2\delta i} + w^2) & w \in \sigma_2 \\ a_3(u,v) + ib_3(u,v) = -1 & w \in \sigma_3 \\ a_4(u,v) + ib_4(u,v) = -1 & w \in \sigma_4 \\ a_5(u,v) + ib_5(u,v) = (u + iv)^2 & w \in \sigma_5 \end{cases} \tag{3.2}$$

where $w = u + iv$.

For the Enneper-Weierstrass representation, we originally seek a function f where $g(z) = z$ on E. When taking the boundary conditions of domain E into consideration, we find that our function f solves the Riemann-Hilbert problem (3.1) where $G(w) = a_k(w) + ib_k(w), w \in \sigma_k$. The numerical computation for the geometry of domain E is difficult, so we define $f^* : \partial\mathbb{D} \to \mathbb{C}$, $G^* : \partial\mathbb{D} \to \mathbb{C}$ by $f^* = f \circ \psi^{-1}$ and $g^* = g \circ \psi^{-1} = \psi^{-1}$ to obtain the Enneper-Weierstrass representation functions f^* and g^*. Now we seek a solution to

$$\mathrm{Re}(G^*(\zeta)f^*(\zeta)) = 0, \text{ for } \tau \in [0,1]. \tag{3.3}$$

Hence $G^*(e^{2\pi i\tau}) = G(\psi^{-1}(e^{2\pi i\tau})) = a_k(\psi^{-1}(e^{2\pi i\tau}) + ib_k(\psi^{-1}(e^{2\pi i\tau})$ for $\tau \in [0,1]$. Using the method of Monahkov [20] we solve the Riemann-Hilbert problem on the disk with G^*.

3.6 Module 2: Homogeneous Riemann-Hilbert Problem

This module computes the solution to the homogeneous Riemann-Hilbert problem on the disk (3.1) given the function G^* defined on the unit circle by (3.2) using Monakhov [20], chapter 1, section 4, 2°, "*The Hilbert problem with piecewise Hölder coefficients*". We denote arcs of the unit circle with the central angles of their endpoints in radians. Here G^* is continuous on arcs $[\varphi_k, \varphi_{k+1}]$ on the unit circle with finite discontinuities at the images of the vertices $t_1, ..., t_5$ of E under ψ. Keeping with the notation of [20] and [8] we define for $t \in \partial E$, $\zeta = \psi(t)$, so that $\zeta_k = \psi(t_k) = e^{\varphi_k}$ and φ is the argument of gz on the unit circle. As in [8], we constructed ψ so that

$$\begin{array}{lllllllll} \phi(t_1) & = & 1 & = & \zeta_1 & = & e^0 & = & e^{\varphi_1} \\ \phi(t_2) & = & i & = & \zeta_2 & = & e^{\pi/2} & = & e^{\varphi_2} \\ \phi(t_3) & = & -i & = & \zeta_3 & = & e^{3\pi/2} & = & e^{\varphi_3} \\ \phi(t_4) & = & \zeta_4 & = & e^{arg(\zeta_4)} & = & e^{\varphi_4}. & & \end{array}$$

Here the argument of ζ_4 is taken to be such that $\frac{3\pi}{2} < arg(\zeta_4) < 2\pi$. Note that $\zeta_5 = \psi(t_5) = \psi(0)$ is not used in our computations due to the fact that $G^* \equiv -1$, a constant, on $(\varphi_2, \varphi_5] \cup [\varphi_5, \varphi_3)$ and hence continuous at ζ_5. Keeping with the notation of [8], we parameterize the unit circle by $\zeta(\tau) = e^{2\pi i\tau}$, for $\tau \in [0, 1]$. Throughout the rest of the paper, we will refer G^* and f^* as functions of ζ and functions of $e^{2\pi i\tau}$ where the choice of notation clarifies the expression.

Monakhov constructs the function $G_1 : \partial\mathbb{D} \to \partial\mathbb{D}$ so that $G_1(\zeta) := -\frac{\overline{G^*(\zeta)}}{G^*(\zeta)}$, computes the index of this piecewise continuous function in the class $O_4(\zeta_1, ..., \zeta_4)$ and defines $\theta_k := \frac{1}{2\pi}\{\text{Arg}(G_1(\zeta_k - 0)) - \arg(G_1(\zeta_k + 0))\}$ where $\text{Arg}(G_1(\zeta_k - 0))$ is computed by continuity on the arc $(\varphi_{k-1}, \varphi_k)$, $\arg(G_1(\varphi_k + 0))$ is computed by continuity on the arc $(\varphi_k, \varphi_{k+1})$ such that the selected result of the many-valued function $\arg(G_1(\zeta_k + 0))$ is chosen so that $\text{Arg}(G_1(\zeta_k - 0))$ and $\arg(G_1(\zeta_k + 0))$ lie on a single-valued branch of $\text{Log}(G_1(\zeta_k - 0))$ ([20], pp. 18). Next we compute the index of G_1 in $O_4(\zeta_1, ..., \zeta_4)$. The values of these limits $\text{Arg}(G_1(\zeta_k - 0))$ and $\arg(G_1(\zeta_k - 0))$ are given by [8] for our geometry in the proof of theorem 3.2 (see [8], pp.

220). Likewise Athanassenas and Lancaster compute the discontinuous index for our case, $\kappa = 0$. Next we define G_0 as in [20] by

$$G_0(\zeta) := -\frac{1}{\zeta^{2\kappa+1}}\left(\prod_{k=1}^{n}\frac{(\frac{1}{\zeta}-\overline{\zeta_k})^{\alpha_k}}{(\zeta-\zeta_k)^{\alpha_k}}\right)\frac{\overline{G^*(\zeta)}}{G^*(\zeta)}. \tag{3.4}$$

Since on the unit circle, $\frac{1}{\zeta} = \overline{\zeta}$ we define our own function G_0^* on the unit circle by

$$G_0^*(\zeta) := \left(\prod_{k=1}^{4}(\zeta-\zeta_k)^{\alpha_k}\right)G^*(\zeta). \tag{3.5}$$

As $\kappa = 0$ and $z \mapsto \frac{\overline{z}}{z}$ maps the punctured plane to the unit disk we obtain

$$G_0(\zeta) = -\frac{1}{\zeta}\frac{\overline{G_0^*(\zeta)}}{G_0^*(\zeta)} \tag{3.6}$$

$$= -\frac{1}{\zeta}\,e^{2i\mathrm{Arg}(G_0^*(\zeta))}. \tag{3.7}$$

This is computationally superior to multiplication since

$$\mathrm{Arg}(G_0^*(\zeta)) = \mathrm{Arg}(G^*(\zeta)) + \sum_{k=1}^{4}\alpha_k \cdot \mathrm{Arg}(\zeta-\zeta_k). \tag{3.8}$$

We must at this time design a convergent scheme to compute the correct $\lim_{\zeta\to\zeta_k^+}\mathrm{Arg}(G_0^*(\zeta))$ and $\lim_{\zeta\to\zeta_k^-}\mathrm{Arg}(G_0^*(\zeta))$. Monakhov notes that G_0 should be a continuous function of index 0 on the unit circle.

We continue in Monakhov ([20], pp. 45 (a)) to compute

$$\Gamma_0(z) = \frac{1}{4\pi i} \int\limits_{|\zeta|=1} \frac{\ln G_0(\zeta)}{\zeta} \cdot \frac{\zeta + z}{\zeta - z}\, d\zeta. \tag{3.9}$$

As in [4] we compute (3.9) using the Hilbert Transform on the unit circle and the Fourier coefficients of

$$h(e^{2\pi i\varphi}) = \frac{\ln(G_0(e^{2\pi i\varphi}))}{e^{2\pi i\varphi}}, \text{ where } \zeta = e^{2\pi\varphi}. \tag{3.10}$$

Let us take a closer look at the computation in [4] concerning Γ_0:

$$\Gamma_0(e^{2i\pi\tau}) = \frac{1}{4\pi i} \int_{|t|=1} \frac{\ln\left(G_0(t)\right)}{t} \cdot \frac{t + e^{2i\pi\tau}}{t - e^{2i\pi\tau}}\, dt.$$

If we substitute $t = e^{2i\pi s}$ where $s \in [0,1]$and $dt = 2i\pi e^{2i\pi s} ds$, then

$$\Gamma_0(e^{2i\pi\tau}) = \frac{1}{4\pi i} \int_0^1 \frac{ln\left(G_0\left(e^{2i\pi s}\right)\right)}{e^{2i\pi s}} \cdot \frac{e^{2i\pi s} + e^{2i\pi\tau}}{e^{2i\pi s} - e^{2i\pi\tau}}\, 2i\pi e^{2i\pi s} ds.$$

We can rewrite $\frac{e^{2i\pi s}+e^{2i\pi\tau}}{e^{2i\pi s}-e^{2i\pi\tau}} = \frac{e^{2i\pi s}-e^{2i\pi\tau}+2e^{2i\pi\tau}}{e^{2i\pi s}-e^{2i\pi\tau}} = 1 + \frac{2e^{2i\pi\tau}}{e^{2i\pi s}-e^{2i\pi\tau}}$ to get

$$\begin{aligned}
&= \frac{1}{2} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right) \left(1 + \frac{2e^{2i\pi\tau}}{e^{2i\pi s} - e^{2i\pi\tau}}\right)\, ds \\
&= \frac{1}{2} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right)\, ds + \frac{1}{2} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right) \left(\frac{2e^{2i\pi\tau}}{e^{2i\pi s} - e^{2i\pi\tau}}\right)\, ds \\
&= \frac{1}{2} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right)\, ds + \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right) \left(\frac{e^{2i\pi\tau}}{e^{2i\pi s} - e^{2i\pi\tau}}\right)\, ds \\
&= \Gamma_0(0) + e^{2i\pi\tau} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right) \left(\frac{1}{e^{2i\pi s} - e^{2i\pi\tau}}\right)\, ds \\
&= \Gamma_0(0) + e^{2i\pi\tau} \int_0^1 \ln\left(G_0\left(e^{2i\pi s}\right)\right) \left(\frac{e^{-2i\pi s}}{1 - e^{2i\pi(\tau - s)}}\right)\, ds.
\end{aligned}$$

Now we'll use [11] (pp. 108-111) to compute the integral using the Fourier coefficients of $h(e^{2i\pi s}) = e^{-2i\pi s} \ln\left(G_0\left(e^{2i\pi s}\right)\right)$ or $h(t) = \frac{\ln(G_0(t))}{t}$. Then

$$\Gamma_0(z) \approx \Gamma_0(0) + z \begin{cases} \displaystyle\sum_{k=0}^{m-1} c_k\, z^k + \frac{c_m}{2}\, z^m, & |z| < 1 \\ \frac{c_0}{2} + \frac{1}{2} \displaystyle\sum_{k=1}^{m-1} \left(c_k\, z^k - c_{-k}\, z^{-k} \right) + \frac{c_m}{4}\, z^m - \frac{c_{-m}}{4}\, z^{-m}, & |z| = 1 \end{cases}$$

$$\approx \Gamma_0(0) + \begin{cases} \displaystyle\sum_{k=0}^{m-1} c_k\, z^{k+1} + \frac{c_m}{2}\, z^{m+1}, & |z| < 1 \\ \frac{c_0}{2}\, z + \frac{1}{2} \displaystyle\sum_{k=1}^{m-1} \left(c_k\, z^{k+1} - c_{-k}\, z^{-(k-1)} \right) + \frac{c_m}{4}\, z^{m+1} - \frac{c_{-m}}{4}\, z^{-(m-1)}, & |z| = 1. \end{cases}$$

As noted in [20] and [8], $e^{\Gamma_0(\zeta)}$ is a continuous, nonvanishing function on $\partial\mathbb{D}$ and we define

$$f_\mu^*(\zeta) = c(e^{i\mu}\zeta - e^{-i\mu})\Pi(\zeta)e^{\Gamma_0(\zeta)}$$

where $c, \mu \in \mathbb{R}$ and $\Pi := (\zeta - 1)^{\alpha_1}(\zeta - i)^{\alpha_2}(\zeta + i)^{\alpha_3}(\zeta - \zeta_4)^{\alpha_4}$. Monakhov shows that this function is a solution to the homogeneous Riemann-Hilbert problem we seek [20].We must choose appropriate constants $c, \mu \in \mathbb{R}$ to determine the particular solution that satisfies our geometrical considerations for our minimal surface. As in [8] we choose $\mu \in \left[\frac{\varphi_4}{2} - 2\pi, 0\right]$ so that $\phi^{-1}\left(e^{-2*1i*\mu}\right) \in \sigma_5$ on ∂E (see [8], p.221, figure 5). A method for computing μ from the geometry and/or boundary conditions is an open problem. For a given μ we obtain a value of $c \in \mathbb{R}$ from observing that $\mathrm{Re}\left(f_\mu^*(\zeta)\right) = 0$ when $\phi^{-1}(\zeta) \in (\sigma_3 \cup \sigma_4) \setminus \{t_2, t_3\}$. Hence we have

$$f_\mu(t) := c(e^{i\mu}\psi(t) - e^{-i\psi(t)})\Pi(\psi(t))e^{\Gamma_0(\psi(t))}$$

and we can compute

$$-1/c = \int_{t_2}^{0} (1 + u^2) i f_\mu(u)\, du \quad \text{(see [8])}.$$

When both constants c and μ are determined we compute the minimal surface using the Enneper-Weierstrass pair $(f^*,\, g^*)$. Recall that $g^* = \psi$ since g is the identity map on E.

Chapter 4

4 Conclusion

4.1 Remarks

We provide two of the three major components for computationally analyzing central fans, specifically module 1 and 3. Both have uses beyond the purposes of this research and are implemented so that other researchers can easily incorporate them into their numerical investigations. While performing rigorous testing we uncovered the limits and boundaries of our components; that these modules were built around case (v) from [8]. New updates and further research and testing continue on both modules 1 and 3 for we are dedicated to providing robust implementations of these object classes as well as new tools for determining sufficient conditions for the existence of central fans. Module 1 is capable of creating a conformal map from $\psi : E \to \mathbb{D}$ and $\psi^{-1} : \mathbb{D} \to E$ to an accuracy of 10^{-12} to 10^{-15} for any reasonable circular arc polygon as long as it lacks sharp vertices such as those found in the continuous cases. The surface construction module is well-tested and robust. We are currently and still modifying the Riemann-Hilbert module. The goal here is to develop a package that accurately computes the capillary surface with a discontinuous and symmetric domain E. In particular we continue to develop a convergent algorithm for the Riemann-Hilbert module and the current version is available on request. With its completion the investigators and our colleagues can continue investigation into the central fan question.

REFERENCES

REFERENCES

[1] A.Elcrat and K.E. Lancaster, *On the behavior of a nonparametric minimal surface in a nonconvex quadrilateral*, Arch. Rational Mech. Anal. 94, **3** (1986), 209-226.

[2] A.Elcrat and K.E. Lancaster, *Boundary behavior of a nonparametric surface of prescribed mean curvature near a reentrant corner*, Trans. Amer. Math. Soc., **297** (1986), 645-650.

[3] A. Khanfer, *On the Existence of Central Fans of Capillary Surfaces*, Dissertation, Wichita State University, (2013).

[4] C.P. Mitchell, *An investigation of capillary surfaces at non-convex corners*, Master's Thesis [532], Wichita State University, 2009.

[5] Danzhu Shi and R. Finn, *On a theorem of Lancaster and Siegel*, Pacific J. Math. **213** (2004), 111119.

[6] K. E. Lancaster, *Boundary behavior of a non-parametric minimal surface in* $\mathbb{R}^3$ *at a non-convex point*, Analysis **5** (1985), 61–69. Corrigendum: Analysis **6** (1986), 413.

[7] K.E. Lancaster and D. Siegel, *Existence and behavior of the radial limits of bounded capillary surface at a corner*, Pacific J. Math. **176** (1996), 165-194. Correction (to figures) **179** (1997), 397-402.

[8] M. Athanassenas and K. E. Lancaster, *CMC Capillary Surfaces at Reentrant Corners*, Pacific J. Math. **Vol. 234**, No. 2 (2008), 201-228.

[9] M. Emmer, *Esistenza, unicit e regolarit nelle superfici de equilibrio nei capillari*, Ann. Univ. Ferrara Sez. VII (N.S.) **18** (1973), 7994.

[10] N.J. Korevaar, *Capillary surface convexity above convex domains*, Indiana Univ. Math. J. 32 , **1** (1983), 7381.

[11] P. Henrici, *Applied and Computational Complex Analysis*, Volume 3,Wiley-Interscience, New York, NY, 1993.

[12] P. Concus and R. Finn, *On the behavior of a capillary surface in a wedge*, Proc Natl Acad Sci USA **63** (1969), 292-299.

[13] P. Concus and R. Finn, (1993) Proceedings from International Symposium on Microgravity Science and Application, *Capillary Surfaces in a Wedge: Differing Contact Angles*, Beijing, China.

[14] P. Concus, R. Finn, and M. Weislogel, *Interface Configuration Experiment Preliminary Results*, Joint Science Review for USMLl and USUP1 with the Mcmgmvity Measurement Gmup, September 22-24, 1993, Huntsville, Alabama, USA.

[15] P. Concus and R. Finn, *Capillary wedges revisited*, SIAM J. Math. Anal. 27 , **1** (1996),5669.

[16] R. Finn and C. Gerhardt, *The internal sphere condition and the capillary problem*, Ann. Mat. Pura Appl. (4) 112 (1977), 1331.

[17] R. Finn and C. Gerhardt, *The internal sphere condition and the capillary problem*, Ann. Mat. Pura Appl. (4) 112 (1977), 1331.

[18] R. Finn, *Featured Review of Existence and behavior of the radial limits of a bounded capillary surface at a corner by Lancaster and Siegel*, Math Reviews MR1433987 (98g:58030a) and MR1452541 (98g:58030b).

[19] T.K. DeLillo and L. Wang, *A Matlab Toolbox (FFTCONP) for Computing Conformal Maps with Fourier Series Methods*, preprint.

[20] V.N. Monakhov, *Boundary-Value Problems with Free Boundaries for Elliptic Systems of Equations*, Translations of Mathematical Monographs, AMS, Providence, 1980.

[21] R. Osserman, *A Survey of Minimal Surfaces*, Second ed., Dover Publications, New York, 1986.

[22] T.A. Driscoll, *Schwarz-Christoffel Mapping*, 1 edition, Cambridge University Press, 2002, http://www.math.udel.edu/ driscoll/research/conformal.html.

APPENDICES

APPENDICES

A Profile Map Class

```
classdef profilemap < handle
    %PROFILEMAP Profile map object encapsulates the maps and parameters
    %   necessary to map the unit disk onto a circular arc polygon
    %   (or polygon) in the upper left quadrant of the complex plane
    %   (quad II). On the boundary we need to the profile map to satisfy
    %   the following:
    %       vertex 1 |---> i
    %       vertex 3 |---> -i
    %       vertex 5 |---> 1
    properties
        ap;         % arc polygon representing profile
        maps;       % collection of maps

        v;          % pre-images of vertices of circular arc polygon on the
                    % unit disk
        plotlevel;  % indicates completeness of plotting steps for testing
    end
    methods
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Constructor
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        function self = profilemap(m_ap, m_profile)

            self.plotlevel = m_profile.plotlevel;
            self.ap        = m_ap;
            self.maps      = {};
            delta          = m_profile.dh;
            mapcount       = 0;

            % need vertices and points to create some maps
            self.v = getvertex(m_ap);
            z_ = linspace(m_ap, delta);
            w = z_;

            if self.plotlevel >= 3
                s_ = m_ap.segment;
                self.doplot(z_, 3, self.v, s_);
                title('Circular Arc Profile')
            end
            % Map 1:
            % Conformal map from circular arc polygon to Jordan curve
            % and interior of circular arc polygon to interior of curve
            map = ap2jordan(m_ap, delta, z_, m_profile.plotlevel);
            mapcount = mapcount+1;
            self.maps{mapcount} = map;
            % need vertices and points to create some maps
            self.v = map.eval(self.v);
```

```
        z_ = map.eval(z_);

        if self.plotlevel > 3
            s_ = map.eval(s_);
            self.doplot(z_, 4, self.v, s_);
            strTitle = sprintf('map(%d): ap2jordan', mapcount);
            title(strTitle)
        end
        % Map 2:
        % Initialize confft map
        map = confft(z_, m_profile, self);
        mapcount = mapcount+1;
        self.maps{mapcount} = map;
        self.v = map.eval(self.v);
        % don't call eval for all of z because the inverse Fornberg
        % map is slow.
        if self.plotlevel > 3
            s_ = map.eval(s_);
            self.doplot([], 4, self.v, s_);
            axis([-1.2 1.2 -1.2 1.2]);
            strTitle = sprintf('map(%d): Fornberg Map', mapcount);
            title(strTitle)
            vt = self.evalbndry(self.v);
            st = self.evalbndry(s_);
            self.doplot(w, 4, vt, st);
            clear st
            clear w
        end
        % Map 3:
        % Use Moebius transformation to map self.v(5)->1 self.v(1)->i
        % self.v(3)->-i
        map = klein([self.v(5) self.v(1) self.v(3)], [1 1i -1i]);
        vt  = self.v;
        mapcount = mapcount+1;
        self.maps{mapcount} = map;
        % need vertices and points to create some maps
        self.v = map.eval(self.v);
        if self.plotlevel > 3
            s_ = map.eval(s_);
            self.doplot([], 4, self.v, s_);
            axis([-1.2 1.2 -1.2 1.2]);
            strTitle = sprintf('map(%d): moebius transformation ...
            [%f %f %f] to [1 i -i]', mapcount, vt(5), vt(1), vt(3));
            title(strTitle)
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Map from exponential map to unit disk
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function w = evalinv(obj, z)
        w = z;
        if isempty(z), return; end

        len = length(obj.maps);
```

```
    for i=1:len
        w = obj.maps{i}.eval(w);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Map from unit disk to exponential map
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = eval(obj, z)
    if isempty(z), w=[]; return; end
    % compute conformal map
    w = z;
    len = length(obj.maps);
    for i=len:-1:1
        w = obj.maps{i}.evalinv(w);
    end
end
function varargout = feval(varargin)
    %FEVAL   Equivalent to EVAL.

    if nargout
        varargout = cell(1,nargout);
        [varargout{:}] = eval(varargin{:});
    else
        varargout{1} = eval(varargin{:});
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Map from boundary of unit disk to boundary of exponential map
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = evalbndry(obj, z)
    if isempty(z), w=[]; return; end

    w = z;
    len = length(obj.maps);
    for i=len:-1:2
        w = obj.maps{i}.evalinv(w);
    end
    w = obj.maps{1}.evalbndryinv(w);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Map from Jordan curve to boundary of exponential map
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = evalJordan2Ap(obj, z, angledata)
    if isempty(z), w=[]; return; end

    w = obj.maps{1}.evalbndryinv(z, angledata);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get vertices of original exponential map
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function w = getvertex(obj)
    w = obj.ap.getvertex();
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        % Get vertices of original exponential map
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        function w = getvertexdisk(obj)
            w = obj.evalinv(obj.getvertex());
        end
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Plot stages of map construction
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        function plothandle = doplot(obj, z, level, v, s, varargin)
            if level <= obj.plotlevel
                isholdon = ~isempty(findobj('type','figure')) && ishold;
                if isholdon
                    plothandle = gcf;
                else
                    plothandle = figure;
                    hold on
                end

                if isempty(z)
                    % plot unit circle
                    theta = 0:.01:2*pi;
                    plot(cos(theta), sin(theta), varargin{:});
                else
                    plot(z, varargin{:});
                end
                axis equal

                if nargin > 3 && ~isempty(v)
                    plot(v,'or');

                    for j=1:length(v)
                        text(real(v(j)), imag(v(j)), ...
                             num2str(j), ...
                             'VerticalAlignment', 'bottom')
                    end
                end

                if nargin > 4 && ~isempty(s)
                    hold on
                    plot(s,'ok');
                end

                if ~isholdon,  hold off; end
            end
        end
    end
end
```

B Crescent Map Class

```
classdef crescentmap < handle
    %CRESCENTMAP: Henrici vol 3, pp 331-332.
    %   Let K be a disk that intersects the boundary of E, the unit disk,
    %   and D_1 := E\K. The crescent map take any simply connected
    %   subdomain of D_1 onto E. Here c = exp(i*gamma) and
    %   conj(c) = exp(-i*gamma) are the points of intersection of K and E.
    %   Also -r is the point in E where the K intersects the negative real
    %   axis. The crescent map is a composition of conformal maps:
    %   0.  The rotation of original unit vector m_r to the negative real
    %           axis. Henrici assumes that m_r lies on the negative real
    %           axis and makes a rotation at the end to aline the image
    %           of -r to -1.
    %   1.  The Moebius transformation
    %           z -> z' := ((z-c)/(z-conj(c)))*((r + conj(c))/(r + c))
    %           This maps D_1 onto the wedge 0 < arg z' < a
    %           where c = -exp(-i*a) * (1 + exp(i*a)) / (1 + exp(-ia)
    %           that is, a = angle(1')
    %   2.  The power map
    %           z' -> z'' := (z')^(pi/a)
    %           This maps the wedge to the upper-half plane
    %   3.  The Moebius transformation and
    %           z'' -> w := (z'' - exp(b))/(z'' - exp(-b))
    %           This maps the upper-half plane onto the unit disk where
    %           b = angle(0')*pi/a (see Henrici(vol 3) p. 332)
    %   4.  The rotation
    %           w -> w := w * exp(-b)
    %           This maps z=-r to w=-1 on the real axis.

    properties
        r;      % (-r,0) is intersection of arc with negative real axis
        c;      % c, conj(c) are intersection of arc with unit circle
        alpha;  % angle of image of 1 under the 1st Moebius transformation
                %   alpha = angle(moebius(0))
        beta;   % multiple of the angle of the image of 0 under the 1st
                % Moebius transformation
                %   beta = 1i*pi*angle(moebius(0))/alpha
        map;    % list of maps whose composition constitues the crescent
                %  map
        mbi;   % inverses of two moebius transformations map{2}, map{3}
    end

    methods
        function self = crescentmap(m_r, m_c)
            if abs(m_r)-1 > 100*eps
                error('crescentmap:parameter', ...
                'the radius of circle K must lie in open unit disk');
            end
            if abs(abs(m_c)-1) > 100*eps
                error('crescentmap:parameter', ...
                'the intersection K of must lie on boundary of unit disk.');
```

```
        end

        a = angle(m_r);
        if a >= 0
            a = pi-a;
        else
            a = -(pi+a);
        end

        self.map{1} = rotation(0, a);
        self.r      = self.map{1}.eval(m_r);
        self.c      = self.map{1}.eval(m_c);

        if (abs(imag(self.r)) > 100*eps) || (self.r >= 0)
            error('crescentmap:rotation', ...
            'rotation failed to map m_r to negative real axis');
        end
        if imag(self.c) <= 0
            self.c = conj(self.c);
        end

        moeb        = moebius([self.c self.r conj(self.c)],[0 1 inf]);
        self.alpha  = angle(moeb(1));
        self.beta   = angle(moeb(0));
        self.map{2} = moeb;
        self.mbi{1} = inv(moeb);

        % The power map is too simple to place in a container
        % w -> z.^(pi/self.alpha)

        b = pi*self.beta/self.alpha;
        self.map{3} = moebius([exp(1i*b) inf exp(-1i*b)],[0 1 inf]);
        self.mbi{2} = inv(self.map{3});
        self.map{4} = rotation(0, -b);
    end

    function w = eval(obj, z)
        w = obj.map{1}.eval(z);
        w = obj.map{2}(w);
        w = w.^(pi/obj.alpha);
        w = obj.map{3}(w);
        w = obj.map{4}.eval(w);
    end

    function varargout = feval(varargin)
        %FEVAL   Equivalent to EVAL.

        if nargout
            varargout = cell(1,nargout);
            [varargout{:}] = eval(varargin{:});
        else
            varargout{1} = eval(varargin{:});
        end
    end
```

```
        function w = evalinv(obj,z)
            w = obj.map{4}.evalinv(z);
            w = obj.mbi{2}(w);
            w = w.^(obj.alpha/pi);
            w = obj.mbi{1}(w);
            w = obj.map{1}.evalinv(w);
        end
    end
end
```

C Power Map on Vertex

```
classdef vertex2line < handle
    %VERTEX2LINE encapsulates a map to open a vertex in the boundary of
    %   a simply connected set on the complex plane to remove the vertex
    %   from the boundary.
    properties
        alpha;
        beta;
        v;
    end
    methods
        function self = vertex2line(z, k)
            % constructor input is a vector of data points and the index
            % of the vertex to open. The data should proceed in counter-
            % clockwise order.
            if (length(z) > 3) && (length(z) > k)
                self.v    = zeros(3,1);
                self.v(2) = z(k);

                if k < length(z)
                    self.v(3) = z(k+1);
                else
                    self.v(3) = z(1);
                end

                if k > 1
                    self.v(1) = z(k-1);
                else
                    self.v(1) = z(end);
                end

                self.alpha = angle(self.v(3)-self.v(2));
                if self.alpha < 0
                    self.alpha = self.alpha + 2*pi;
                end

                self.beta  = angle(self.v(1)-self.v(2));
                if self.beta < 0
                    self.beta = self.beta + 2*pi;
                end

                if self.alpha > self.beta
                    temp       = self.alpha;
                    self.alpha = self.beta;
                    self.beta  = temp;
                end

                if self.beta - self.alpha > pi
                    % This causes a mapping problem...more code needed!
                    error('vertex2line:Unsupported', sprintf( ...
                        'Angle: %f ', self.beta-self.alpha));
```

```
                end

                self.beta = self.beta - self.alpha;

            elseif (length(z) == 3)
                self.v     = zeros(3,1);
                self.v(2)  = z(1);
                self.alpha = z(2);
                self.beta  = z(3);
            end
        end

        function w = evalinv(obj, z, varargin)
            % use power map to create vertex
            w = z.^(obj.beta/pi);
            % rotate line segment away from positive real axis
            w = w*exp(1i*obj.alpha);
            % move vertex back
            w = w + obj.v(2);
            % infinity is a fixed point
            w(isinf(z)) = inf;
        end

        function w = eval(obj, z, varargin)
            % move vertex to origin
            w = z - obj.v(2);
            % rotate line segment to positive real axis
            w = w*exp(-1i*obj.alpha);
            % use power map to remove vertex
            w = w.^(pi/obj.beta);
            % infinity is a fixed point
            w(isinf(z)) = inf;
        end
        function varargout = feval(varargin)
            %FEVAL   Equivalent to EVAL.
            if nargout
                varargout = cell(1,nargout);
                [varargout{:}] = eval(varargin{:});
            else
                varargout{1} = eval(varargin{:});
            end
        end
    end
end
```

D LINFRACT Class

```
classdef linfract < handle
    %LINFRACT encapsulates a map of an open vertex and maps certain
    %   vertices to a set position on the complex plane. This class
    %   is an extension from Toby Driscoll's Moebius Transformation
    %   function and Inverted Moebius Transformation function in SCTOOLBOX.
    %   See (Driscoll's 1999 and 2005 paper)
    properties
        moeb;
        minv;
    end
    methods
        function self = linfract(m_v, m_w)
            %Here we call Driscroll's Moebius Transformation, you will need
            %SCTOOLBOX or the moebius.m file from SCTOOLBOX.
            self.moeb = moebius(m_v, m_w);

            %Here we call Driscroll's Inverted Moebius Transformation, you
            %will need SCTOOLBOX or the inv.m file from SCTOOLBOX.
            self.minv = inv(self.moeb);
        end
        function w = eval(obj, z)
            w = obj.moeb(z);
        end
        function w = evalinv(obj, z)
            w = obj.minv(z);
        end
    end
end
% Map 4:
% Use Moebius transformation to map v(5)->0 v(1)->1 v(2)->inf
map  = linfract([self.v(1) self.v(2) self.v(3)], [0 1 inf]);
tmp  = self.v;
mapcount = mapcount+1;
self.maps{mapcount} = map;
% need vertices and points to create some maps
self.v = map.eval(self.v);
self.s = map.eval(self.s);
self.z = map.eval(self.z);

if m_plotlevel > 3
    self.doplot(self.z, 4, self.v, self.s);
    axis([-.5 2 -.5 2]);
    strTitle = sprintf('map(%d): moebius transformation [%f %f %f] to ...
                [0 1 inf]', mapcount, tmp(1), tmp(2), tmp(3));
    title(strTitle)
end
```

E Cayley's Transform

```
classdef CayleyTransform < handle
    %CAYLEYTRANSFORM implements the Cayley transform of the complex plane
    %   onto itself given by w = (z - i)/(z + i);

    properties
    end

    methods
        function self = CayleyTransform()
            return;
        end

        function w = eval(obj, z)
            w = (z - 1i)./(z + 1i);
            w(isinf(z)) = 1;
        end

        function w = evalinv(obj, z)
            w = 1i*(1 + z)./(1 - z);
        end
    end
end
```

F Halley's Transform

```
function [ x, tol, iter ] = halley( map, y, guess )
    f   = @(z) polyval(map.p, z);
    fp  = @(z) polyval(map.dp, z);
    fpp = @(z) polyval(map.d2p, z);

    x    = guess;
    tol  = zeros(size(y)) + inf;
    iter = zeros(size(y));
    N    = length(y);
    T    = 10^(-8);
    MaxIter = 25;

    if ( map.plotlevel >= 3 )
        fprintf('Iterations:\t\tError:\n');
    end

    for k = 1:N
        xt = x(k);
        yt = y(k);
        g  = @(z) (polyval(map.p, z)-yt);

        for i=1:MaxIter
            xt = xt - (2*g(xt)*fp(xt))/(2*fp(xt)^2 - g(xt)*fpp(xt));
            r  = norm(xt);
            if (r>1)
                xt= xt/r;
            end
            err = abs(yt-f(xt));
            if err < T
                break;
            end
        end

        x(k)    = xt;
        tol(k)  = err;
        iter(k) = i;
        if i >= MaxIter
            fprintf('Maximum(%d) number of iterations(%d) exceeded\n',...
                        MaxIter, i);
        end

        if ( map.plotlevel >= 3 )
            fprintf('%g:\t\t\t %g\n', iter(k), tol(k));
        end
    end
end
```

G Enneper-Weierstrass Representation

```
function [x,y,z] = Enneper(f, g, limits)
%    phi1 = @(z) f(z).*(1-g(z).^2)./2;
%    phi2 = @(z) 1i.*f(z).*(1+g(z).^2)./2;
%    phi3 = @(z) f(z).*g(z);
    x = linspace(limits(1), limits(2), ceil(limits(2)-limits(1))*20);
    y = linspace(limits(3), limits(4), ceil(limits(4)-limits(3))*20);
    [x,y] = meshgrid(x,y);
    w = complex(x,y);
    [m,n] = size(w);

    for j=1:m
        for k=1:n
           x(j,k) = real(lineint(@phi1,0,w(j,k)));
           y(j,k) = real(lineint(@phi2,0,w(j,k)));
           z(j,k) = real(lineint(@phi3,0,w(j,k)));
        end
    end

    function w = phi1(z)
        w = f(z).*(1-g(z).^2)/2;
    end
    function w = phi2(z)
        w = 1i.*f(z).*(1+g(z).^2)/2;
    end
    function w = phi3(z)
        w = f(z).*g(z);
    end
end
```