A Dissertation by

Saman Sahraei

Master of Science, University of Shiraz, 2007
Bachelor of Science, University of Kerman, 2004

Submitted to the Department of Mathematics, Statistics and Physics
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

May 2018
The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Philosophy with a major in Applied Mathematics.

Thomas DeLillo, Committee Chair

Buma Fridman, Committee Member

Tianshi Lu, Committee Member

Daowei Ma, Committee Member

Scott Miller, Committee Member

Accepted for the College of Liberal Arts and Sciences

Ron Matson, Dean

Accepted for the Graduate School

Dennis Livesay, Dean
Acknowledgement

It is a genuine pleasure to express my appreciation and gratitude to my advisor Dr. Thomas DeLillo, who helped me extensively during my study at Wichita State University and particularly with my dissertation. I would also like to thank my committee members Dr. Buma Fridman, Dr. Daowei Ma, Dr. Tianshi Lu, and Dr. Scott Miller for their time and suggestions.

Finally, I should thank the faculty and staff of the Department of Mathematics, Statistics, and Physics for their support during my work at Wichita State University.
Dedication

I would like to dedicate this work to my parents, describing whose help and support is beyond the reach of my words.
Calculations of potential flow and pressure over multi-element airfoils are studied, using conformal maps. A composition of classical Karman-Trefftz maps is used to smooth out the trailing edge corners successively, from each of the $m$ airfoils, producing a multiply connected domain with nearly circular boundaries. A Fourier/Laurent series method is then used to map a domain exterior to nonoverlapping disks to the exterior of the domain with smooth boundaries. The computations of potential flow and pressure are done in the domain with circular boundaries and then carried over to the physical space, using the conformal maps. In practice, this technique can be used as a first approximation to the solution in more complicated airfoil configurations.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>5</td>
</tr>
<tr>
<td>1.4</td>
<td>5</td>
</tr>
<tr>
<td>1.5</td>
<td>6</td>
</tr>
<tr>
<td>1.6</td>
<td>7</td>
</tr>
<tr>
<td>1.7</td>
<td>8</td>
</tr>
<tr>
<td>1.8</td>
<td>10</td>
</tr>
<tr>
<td>1.9</td>
<td>10</td>
</tr>
<tr>
<td>1.9.1</td>
<td>11</td>
</tr>
<tr>
<td>1.9.2</td>
<td>13</td>
</tr>
<tr>
<td>1.9.3</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>2.1</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
</tr>
<tr>
<td>3.1</td>
<td>22</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Examples Of Flows</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1 Uniform Flow</td>
<td>22</td>
</tr>
<tr>
<td>3.2.2 Flow From A Doublet Or Dipole</td>
<td>23</td>
</tr>
<tr>
<td>3.2.3 Flow Around A Point Vortex</td>
<td>24</td>
</tr>
<tr>
<td>3.3 Special Cases</td>
<td>25</td>
</tr>
<tr>
<td>3.4 Convergence Results</td>
<td>29</td>
</tr>
<tr>
<td>3.4.1 Reflections In Circles</td>
<td>29</td>
</tr>
<tr>
<td>3.4.2 Convergence Of The Streaming Flow</td>
<td>31</td>
</tr>
<tr>
<td>3.4.3 Convergence Of The Circulating Flow</td>
<td>35</td>
</tr>
<tr>
<td>4 The Code and Numerical Results</td>
<td>39</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>39</td>
</tr>
<tr>
<td>4.2 The Code</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Numerical Results</td>
<td>43</td>
</tr>
<tr>
<td>4.3.1 One Airfoil</td>
<td>43</td>
</tr>
<tr>
<td>4.3.2 Two Airfoils</td>
<td>45</td>
</tr>
<tr>
<td>4.3.3 Three Airfoils</td>
<td>46</td>
</tr>
<tr>
<td>4.3.4 Four Airfoils</td>
<td>47</td>
</tr>
<tr>
<td>4.4 Conclusions and future work</td>
<td>49</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>55</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>59</td>
</tr>
<tr>
<td>A Code Excerpts</td>
<td>60</td>
</tr>
<tr>
<td>A.0.1 Cosine Airfoil</td>
<td>60</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.0.2 Joukowski Airfoil</td>
<td>63</td>
</tr>
<tr>
<td>A.0.3 Karmann-Trefftz Map</td>
<td>65</td>
</tr>
<tr>
<td>A.0.4 Address Of The Leaves</td>
<td>70</td>
</tr>
<tr>
<td>A.0.5 Velocity</td>
<td>74</td>
</tr>
<tr>
<td>A.0.6 Solution In The Circular Domain</td>
<td>79</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Convergence of successive Newton-iteration errors $|S_{i+1} - S^i|_\infty$ at the Fourier points for the Fourier series map to the near-circular region with corners in Figure 4.2.</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Convergence of successive Newton-iteration errors $|S_{i+1} - S^i|_\infty$ at the Fourier points for the Fourier series map to the near-circular regions for examples with corners in Figure 4.7.</td>
<td>50</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1.1</td>
<td>Flow over 3 cosine airfoils (left). The streamlines are conformally mapped from the circle domain (right) through the (smooth) middle domain to the exterior of the airfoils.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Non-zero trailing-edge angle</td>
<td>7</td>
</tr>
<tr>
<td>1.3</td>
<td>Removing corner point using Karmann-Trefftz map</td>
<td>12</td>
</tr>
<tr>
<td>1.4</td>
<td>Successive application of Karmann-Trefftz map</td>
<td>12</td>
</tr>
<tr>
<td>4.1</td>
<td>Growth of the terms of the circulating velocity for three circles</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>The three figures show the composition of the Fornberg map to the smooth spline curve with $N_s = 101$ and $N = 256$. Streamlines are plotted using the velocity potential and applying the Kutta condition to adjust the circulation so that there is a stagnation point at the trailing edge.</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>Error for Joukowski map and $k^{-1} \circ h$.</td>
<td>46</td>
</tr>
<tr>
<td>4.4</td>
<td>Pressure $C_p = 1 -</td>
<td>V</td>
</tr>
<tr>
<td>4.5</td>
<td>Flow in absence of circulation</td>
<td>47</td>
</tr>
<tr>
<td>4.6</td>
<td>Flow in presence of circulation</td>
<td>49</td>
</tr>
<tr>
<td>4.7</td>
<td>Flow over two Cosine airfoils</td>
<td>49</td>
</tr>
<tr>
<td>4.8</td>
<td>Flow over two Joukowski airfoils</td>
<td>51</td>
</tr>
<tr>
<td>4.9</td>
<td>Flow over three Cosine airfoils</td>
<td>52</td>
</tr>
<tr>
<td>4.10</td>
<td>Flow over three Joukowski airfoils</td>
<td>53</td>
</tr>
<tr>
<td>4.11</td>
<td>Flow over four Cosine airfoils</td>
<td>54</td>
</tr>
<tr>
<td>4.12</td>
<td>Exceptional case for flow over four Cosine airfoils</td>
<td>54</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

1.1 Prelude

Numerical methods for computing conformal maps between domains in the complex plane exterior to disks and domains exterior to airfoils were used frequently in the past to compute potential flow over single and multi-elements airfoils; see e.g., Halsey [21]. Here, we revisit these calculations using more recently developed methods for conformal mapping of multiply connected domains [6] with smooth boundaries. In this setup, the classical Karman-Trefftz transformation \( k_i \) is used to smooth the trailing edge corners successively from each of the \( m \) airfoils \( \partial \Omega_i, i = 1, \ldots, m \) producing a multiply connected domain with nearly circular, smooth boundaries. A Fourier/Laurent series method is then used to compute the map \( h \) from a domain exterior to nonoverlapping disks to the exterior of a domain with smooth boundaries. The Karman-Trefftz maps are then inverted to produce a composite map,

\[
f = k_1^{-1} \circ k_2^{-1} \circ \cdots \circ k_m^{-1} \circ h,
\]

from the circle domain to the airfoil domain. The flow is computed in the circle domain by a method of images, with the circulations calculated to satisfy the Kutta condition, and mapped back to the airfoil domain. The procedure is illustrated in Figure 1.1 for three airfoils given by cosine curves (left), where the boundary curves in the middle domain are the result of the application of three successive Karman-Trefftz maps to smooth the trailing edges and the method [6] is used to compute the map from the exterior of the circles (right) to the exterior of the smooth domain. (The conformally equivalent circles are found as part of the calculation of the map.)
To produce these maps, an extension [6] to exterior multiply connected domains of Fornberg’s original method [20] for mapping the unit disk to a simply connected domain bounded by a smooth (no corners) curve; see also [17]. The more recent methods are Newton-like methods and converge more rapidly than the linearly convergent methods, such as Theodorsen’s and Timman/James, used in [21, 22]. Halsey and others [21] used the simply connected exterior method of Timman/James [13] applied Koebe’s method to successively map domains to the exterior of a disk; see [22]. Most of the more recently developed conformal mapping methods for multiply connected domains [6, 26, 28, 29] produce the map directly to the domain.

In this setup, the airfoil domain is mapped to the circle domain where the potential flow problem can be solved efficiently. The velocity potential is harmonic and transplants under conformal mapping. We implement the procedure of [21, 27] which uses a method of images to compute streaming and circulating flow around the circles. Successive reflection of
singularities in circles leads to an infinite series and convergence is proven, if the circles are sufficiently well-separated. A calculation of streaming flow by a reflection method is given in [7] and is implemented here and extended to the calculation of circulating flow. Since the problems are boundary value problems for the harmonic velocity potentials, superposition can be used to solve the problems and adjust the circulations to satisfy the Kutta condition, i.e., that the velocity should be zero at the trailing edges of the airfoils.

In practical cases, this method can be used together with other numerical techniques or more importantly it can provide certain boundary conditions close to the airfoils. In close vicinity of the airfoils, viscous effects might be quite crucial; however, when we get far from the airfoil they will eventually become negligible. So, the technique elaborated on here can solve the problem a bit further away from the airfoils, provide new boundary conditions, and then the problem can be solved more accurately using computational fluid dynamics techniques adjacent to the airfoils.

See [22, 29] for introductions to numerical conformal mapping. See [3, 9] for general introductions to fluid mechanics with applications of complex variable and conformal mapping. See [25, 26, 31] for extensive applications of complex variables to flow in the plane.

The remainder of this chapter is a review of standard literature in aerodynamics, following [3] and [4], pertaining to the material needed in this research. The main problem of our research is then set forth and a brief overview of the solution is presented. Chapter two elaborates on generalization of Fornberg’s method to multiply connected domains. This chapter is merely mentioned for the matter of completeness of discussion and the existence theorem there is essentially the logic behind the development of the numerical conformal map, the Fornberg-like method, and the corresponding code for it. The material with full detail can be found in [6]. Chapter three has some convergence results, explaining the logic
behind the technique used in our code, for finding the streaming and circulating flows in the circular domain. Finally, chapter four explains the structure of our code briefly and has a report on the results of the code applied in some special cases. Moreover, some parts of the code are presented in the appendices.

1.2 Basic Definitions

Here we review some basic but fundamental concepts from aerodynamics. The purpose of this chapter is mainly to set up the necessary background for this research and is merely for completeness of the discussion. For more detailed discussion of fluids and aerodynamics the reader may consult references [3] and [4].

Our study is restricted to flow of air in subsets of $\mathbb{R}^2$, which is eventually going to be identified with $\mathbb{C}$. Throughout the whole research, our attempt is to find the velocity field of a steady, subsonic, inviscid, incompressible, and irrotational flow; and, the only fundamental equation that we need from aerodynamics is the continuity equation,

$$\nabla \cdot \mathbf{U} = 0.$$

where $\mathbf{U}$ is the velocity field of the flow and $\nabla$ is the gradient operator. This version of the continuity equation, which is a statement of the conservation of mass, is suitable for the kind of flow that is of our concern here, namely steady flow.

Here we are interested in the flow of air over multi-element airfoils, where an airfoil is defined as a cross section of a wing. Note that the region in which air flows is the domain in $\mathbb{R}^2$ exterior to $n$ airfoils, showed by $\Omega$; hence, the flow of our interest is two dimensional. Moreover, we restrict our study to subsonic, inviscid, incompressible, and irrotational flow. Let us go over some definitions first.
Steady Flow. A flow is said to be steady if its velocity field does not depend on time.

Subsonic Flow. A flow is said to be subsonic if the speed of the fluid is less than the speed of sound in the medium.

Inviscid Flow. A flow is said to be inviscid if viscosity is equal to zero.

Incompressible Flow. A flow is said to be incompressible when density is constant.

Irrotational Flow. An irrotational flow is the one for which vorticity is zero, i.e. \( \nabla \times \mathbf{U} = 0 \).

Stagnation Point. A stagnation point is a point at which velocity is zero.

1.3 The Differential Equation

In this section we set up the differential equation that determines the velocity field, and in the subsequent sections I will discuss the boundary conditions that the solution should satisfy. Assume that \( \mathbf{U} = \nabla \phi \), for some function \( \phi \in \mathcal{C}^2(\Omega) \). In this case \( \phi \) is called the velocity potential and since \( \mathbf{U} \) satisfies the continuity equation, \( \nabla \cdot \mathbf{U} = 0 \), and \( \nabla \cdot \nabla = \Delta \), i.e. the Laplace operator, we have

\[ \Delta \phi = 0. \]

Therefore, if we can find such \( \phi \) on \( \Omega \), we can easily find \( \mathbf{U} \) by taking its gradient. In what follows we are going to try to find this velocity potential \( \phi \), satisfying certain boundary conditions.

1.4 Boundary Conditions

In order to determine the boundary conditions, we first note that \( \partial \Omega = \{ \gamma_1, \gamma_2, \ldots, \gamma_m \} \), where \( \gamma_i \) is the boundary of the \( i \)-th airfoil, \( i = 1, \ldots, m \). Here, we have implicitly identified \( \mathbb{R}^2 \) with \( \mathbb{C} \), and we are considering this as a subspace of \( \mathbb{C}^* \), the extended complex numbers.
The body of each airfoil is solid and the flow cannot penetrate the airfoil; thus, the velocity vector must be tangent to $\gamma_i$. This wall tangency condition gives us

$$\mathbf{U}\bigg|_{\gamma_i} \cdot \mathbf{n}_i = (\nabla \phi)\bigg|_{\gamma_i} \cdot \mathbf{n}_i = \frac{\partial \phi}{\partial n}\bigg|_{\gamma_i} = 0,$$

where $\mathbf{n}_i$ is the unit vector normal to $\gamma_i$. So we have Neumann boundary condition on the airfoils.

We want zero velocity perturbation at large distances from the airfoil elements; hence,

$$\lim_{(x,y) \to \infty} \nabla \phi = \text{constant vector}$$

This completes the determination of the boundary conditions. This boundary value problem, however, has infinitely many solutions; thus, in order to single out a unique solution, which corresponds to physically acceptable flow, we need an extra condition which is called the Kutta condition, but before presenting it we need to study circulation.

1.5 Circulation

The circulation of a flow with velocity field $\mathbf{U}$ around a simple close curve $C$ is defined as

$$\Gamma = -\int_C \mathbf{U}$$

where $C$ is traversed anti-clockwise. In the following chapters, when solving our Laplace equation, we try to start with known solutions, make some adjustments, and then superimpose the adjusted solutions. One of the adjustments is finding the correct circulation, which is given by the Kutta condition.
1.6 The Kutta Condition

According to experimental observations, when the flow has just started, and the flow pattern is just beginning to develop around the airfoil, it tries to curl around the sharp trailing edge from the bottom surface to the top surface. However, our considerations of inviscid, incompressible flow show that velocity becomes infinitely large at a sharp corner. Hence, this type of flow is not stable, and as time goes by, the stagnation point on the upper surface moves towards the trailing edge. Intuitively, for a steady flow over a given airfoil at a determined angle of attack, nature adopts the particular value of circulation which results in the flow leaving smoothly at the trailing edge, according to [4].
In order to state the Kutta condition accurately and analytically, we need to study the trailing edges more precisely. There are two types of trailing edges, one with non-zero trailing-edge angle and the other with zero trailing-edge angle. Since the airfoils of interest here are of the former type, the Kutta condition for this case is going to be mentioned; so, consider the trailing edge with non-zero angle as shown in figure 1.2.

Denote the velocities along the top surface and the bottom surface as $U_1$ and $U_2$, respectively. If the velocities were non-zero at the trailing edge, then we would have two velocities in different directions at the same point. However, this is not physically possible, and the only possible case is for both $U_1$ and $U_2$ to be zero at the trailing edge.

**Kutta Condition.** Now we can state the Kutta condition:

1. For a given airfoil at a given angle of attack, the value of $\Gamma$, circulation, around the airfoil is such that the flow leaves the trailing edge smoothly.
2. If the trialing-edge angle is non-zero, then the trailing edge is a stagnation point.

### 1.7 Stream Function and Stream Lines

In the process of solving our differential equation we will be in need of stream functions. A streamline, $\gamma = \gamma(t)$, is defined to be a curve whose tangent at any point is equal to the velocity vector at that point, i.e. an integral curve of the velocity vector field; thus,

$$\frac{d\gamma}{dt} = U(x, y).$$

The differential equation for a streamline in a steady flow is given by

$$\frac{dy}{dx} = \frac{v}{u}$$

where $U = \langle u, v \rangle$. Solving this equation will give us $\psi(x, y) = c$, where $c$ is an arbitrary value. The function $\psi$, is defined as the stream function, and one can obtain the equation
of a streamline by letting it be equal to some value $c$. Having the velocity potential, $\phi$, and stream function, $\psi$, we have

\[
\begin{cases}
\frac{\partial \phi}{\partial x} = u, \quad \frac{\partial \phi}{\partial y} = v \\
\frac{\partial \psi}{\partial x} = -v, \quad \frac{\partial \psi}{\partial y} = u.
\end{cases}
\]

This last set of equations tells us that $\phi$ and $\psi$ satisfy the Cauchy-Riemann equations; hence, we can define $w : \Omega \to \mathbb{C}$ as $w(\omega) = \phi(\omega) + i\psi(\omega)$, where $\omega = x + iy$. In this case, $w$ is called the complex velocity potential. Note that

\[
\frac{dw}{d\omega} = \frac{dw}{dx} = \phi_x + i\psi_x = u - iv
\]

hence, $u + iv = \overline{w'(\omega)}$. In the following chapters we are aiming for finding $w$, and having that we have $\phi$ and consequently $U$.

**Theorem 1.1.** The circulation $\Gamma$ of the flow around $C$ and the flux $F$ of the flow through $C$ are given by

\[
\Gamma + iF = \int_C w'(\omega) d\omega.
\]

**Proof.** The circulation is given by

\[
\Gamma = \int_C (\phi_x dx + \phi_y dy) = \int_C U \cdot \frac{dt}{ds} ds
\]

and the flux through $C$ is given by

\[
F = \int_C (\phi_x dy - \phi_y dx) = \int_C U \cdot \frac{dn}{ds} ds
\]

where $U = \langle u, v \rangle, dt/\,ds = dx/\,ds, dy/\,ds$ is the unit tangent to $C$, $dn/\,ds = (dy/\,ds, -dx/\,ds)$ is the unit outward normal to $C$, and $s$ is arclength along $C$. Note that

\[
w'(\omega)d\omega = (\phi_x - i\phi_y)(dx + idy) = \phi_x dx + \phi_y dy + i(\phi_x dy - \phi_y dx).
\]

\[\square\]
1.8 The Problem

In section 1.3 we saw that the velocity potential \( \phi \) satisfies the Laplace equation on the domain exterior to \( m \) airfoils, \( \Omega \). On the boundaries of the airfoils, \( \gamma_i, i = 1, \ldots, m \), we have Neumann condition, \( \frac{\partial \phi}{\partial n} \bigg|_{\gamma_i} = 0 \); moreover, we want zero velocity perturbation at large distances from the airfoil elements, and this would give us the boundary condition at infinity. Finally, the Kutta condition singles out the solution we are looking for, since without this auxiliary condition the boundary value problem has infinitely many solutions. Thus, the problem is

\[
\begin{align*}
\Delta \phi &= 0 \quad (x, y) \in \Omega \\
\frac{\partial \phi}{\partial n} \bigg|_{\gamma_i} &= 0, \quad i = 1, \ldots, m \\
\lim_{(x, y) \to \infty} \nabla \phi &= \text{constant vector} \\
\nabla \phi \bigg|_{T_i} &= 0, \quad i = i, \ldots, m
\end{align*}
\]

where \( T_i \) is the trailing edge of the \( i \)-th airfoil.

1.9 The Solution

The solution suggested here has three steps. The first step is mapping the problem itself to a domain with circular holes, instead of airfoils, one corresponding to each airfoil, using bijective conformal maps. The second step is solving the problem in the new space; and, finally, the third step is mapping the solution back to the original space.
Confomal maps have special properties that are quite crucial for solving our problem. Firstly, they preserve the Laplace operator, i.e., they map solutions of the Laplace equation to solutions of the Laplace equation. Secondly, they preserve angles. This property helps us with our boundary condition on the airfoils. Thirdly, conformal maps would map streamlines to streamlines and they preserve circulation. There is one extra condition that the conformal map we are using should satisfy. If we want to map uniform flow past some object into the same uniform flow past another object we must choose \( f(z) \) such that

\[
\lim_{z \to \infty} \left( \frac{df}{dz} \right) = 1.
\]

In the rest of this chapter we give an overview of the aforementioned steps and more details are left for to the next chapters.

### 1.9.1 Solution - Step One

This step consists of two parts. Part one is removing the corners, one at a time, by applying the Karmann-Trefftz maps successively to get a domain with smooth boundaries which are nearly circular; and, part two is using Fornberg-like method to map the problem to a domain exterior to \( m \) disks. Here we will study the Karmann-Trefftz map, and in the next chapter, an explanation of the Fornberg-like method and its numerical derivation, are presented in detail.

**Karmann-Trefftz Map**

The Karmann-Trefftz map is used to remove the corners from the airfoils in order to get a smooth curve. It is defined as follows

\[
\frac{\zeta - \zeta_1}{\zeta - \zeta_2} = \left( \frac{\omega - \omega_1}{\omega - \omega_2} \right)^\delta
\]
Figure 1.3: Removing corner point using Karmann-Trefftz map

\[ \delta = \frac{\pi}{2\pi - \lambda} \]

Figure 1.4: Successive application of Karmann-Trefftz map

where, \( \zeta \) shows a typical point in the target space and \( \lambda \) is the trailing-edge angle. Note that this map is a conformal map; so, if there are more than one airfoils, applying it once will
smooth out one of the trailing edges while not changing the rest of the trailing edge angles, hence this map is to be applied once corresponding to each corner point. The process of removing corners is shown in the figure 1.4.

1.9.2 Solution - Step Two

Now the problem is mapped to a more appropriate space and it is prepared to be solved. The idea here is to find the complex velocity potential, $w(z)$, which describes the physical flow in this space, and recall from the last section of chapter one that the relationship between the velocity field and the complex velocity potential is given by $U = \nabla \text{Re}(w)$. Before going any further, let us state the Milne-Thomson theorem, e.g. [9, Theorem 4.1], which is one of the main tools in our solution.

**Theorem 1.2. Milne-Thomson Theorem.** Suppose that we have a flow with complex velocity potential $f(z)$. If a circle, $C$, centered at $c$ with radius $r$ is introduced in the way of the flow, such that all the singularities of $f$ lie out of the disk $|z - c| \leq r$, then the new complex velocity potential is given by

$$w(z) = f(z) + f(c + \frac{r^2}{z-c})$$

**Proof.** By Schwarz reflection, $w(z)$ is clearly analytic at $z$, if $f(z)$ is analytic at $z$. Note that on the circle $C : z = c + re^{i\theta}$,

$$c + \frac{r^2}{\overline{c} + re^{-i\theta} - \overline{c}} = c + \frac{r^2}{re^{-i\theta}} = c + re^{i\theta} = z.$$  

Therefore, $w(z) = f(z) + \overline{f(z)} = 2\text{Re}f(z)$, so the stream function $\text{Im}f(z)$ is constant on $C$ and therefore $C$ is a streamline. □
The point \( c + \frac{r^2}{z - c} \) is the reflection of \( z \) with respect to the circle \( C \). This theorem is what causes the crucial difference between the flow analysis of single-element airfoil and multi-element airfoil. This will be studied extensively in chapter three.

1.9.3 Solution - Step Three

All the conformal maps that we used to map the problem to the circle space are invertible and their inverses are conformal as well. So, now by applying the Fornberg map and inverses of the Karmann-Trefftz maps, in order to put the corner points back, we can map the streamlines to the physical space and, more importantly, we can compute the pressure distribution over the airfoils.
CHAPTER 2
Fornberg-like Method

2.1 Introduction

In the previous chapter we saw that in order to map our problem to a problem on the plane with \( m \) disks removed from it, we need two maps, one Karmann-Trefftz and the other a Fornberg-like map. The former was studied in the previous chapter and the subject of this chapter is a brief study of the latter. In this chapter we will concentrate on a numerical method for conformally mapping unbounded one or multiply connected regions exterior to \( m \) non-overlapping disks to unbounded regions exterior to \( n \) non-overlapping Jordan curves where \( m \geq 1 \).

This method uses a Newton-like approach. Approximations of the boundary correspondences together with centers and radii of the circles are going to be updated in each Newton iteration. For Fornberg-like methods, in general, using discretisation by \( N \)-point trigonometric interpolation and FFTs leads to computational costs of \( O(N \log N) \) for the simply and doubly connected maps and \( O(N^2) \) for the multiply connected maps.

2.2 The Circle Map Problem

The cases where the circular domain is simply connected is slightly different from the case where it is multiply connected; thus, we need to consider these cases separately.
2.2.1 Simply Connected

We are looking for a function that maps the exterior of the unit disk conformally to the exterior of a smooth, closed curve, \( \partial \Omega \) parametrized by \( \gamma(S) \) such that \( \gamma'(S) \) is continuous. The form of the map is

\[
f(z) = a_1 z + a_0 + \sum_{k=1}^{\infty} a_{-k} z^{-k}.
\]

Note that \( f'(\infty) = a_1 \) in this case. A Newton-like method is used for finding the boundary correspondence \( S = S(\theta) \), such that

\[
f(e^{i\theta}) = \gamma(S(\theta)).
\]

If \( S^{(k)}(\theta) \) shows the approximation to \( S(\theta) \) at the \( k \)-th Newton step, a correction \( U^{(k)}(\theta) \) is computed, such that the boundary values of the linearisation,

\[
\gamma(S^{(k)}(\theta)) + U^{(k)}(\theta) \approx \gamma(S^{(k)}(\theta)) + \gamma'(S^{(k)}(\theta))U^{(k)}(\theta),
\]

are boundary values of a function analytic in the exterior of the disk with a simple pole at \( \infty \). The map is normalized by fixing a boundary point, \( f(1) = \gamma(S(0)) = \gamma_0 \). In order for a function, defined on the unit circle, to have an analytic extension, to the plane with interior of the unit disk removed, all its positive indexed Fourier coefficients, except \( a_1 \) should be zero, i.e., \( a_k = 0, k = 2, 3, \ldots \). This condition, and discretization by \( N \)-point trigonometric interpolation, leads to a symmetric positive (semi-)definite linear system for \( U^{(k)} \), which can be solved efficiently by the conjugate gradient method using the \texttt{fft}. The matrix-vector multiplications therefore cost \( O(N \log N) \). The Newton update is

\[
S^{(k+1)}(\theta) = S^{(k)}(\theta) + U^{(k)}(\theta),
\]

with \( U^{(k)}(0) = 0 \) to fix \( f(1) = \gamma_0 \). Details are given in [17] following [20].
2.2.2 Multiply connected

We are seeking a conformal map $f$ from the complement, $\Omega$, of $m$ closed non-intersecting disks, $D_k$, onto a region $Z$ which is exterior to $m$ non-intersecting smooth Jordan curves, $\gamma_k$, $1 \leq k \leq m$. The following theorem, establishes the existence and uniqueness of the circle map under suitable normalisation conditions. For detailed discussion, proof, and examples one can see [6].

**Theorem 2.1.** Let $Z$ be a region of connectivity $n \geq 2$ in the extended complex plane such that $\infty \in Z$. Then there exists a unique circular region $\Omega$ of connectivity $m$ and a unique one-to-one analytic function $f$ in $\Omega$ such that

$$f(z) = z + O(1/z) \quad \text{for } z \approx \infty,$$

such that $f(\Omega) = Z$.

The boundaries of the circular disks $D_k$ are the circles $C_k$ with centers $c_k$ and radii $r_k$ and are parametrized by $C_k(\theta) := c_k + r_k e^{i\theta}$. The boundary of $\Omega$ is, therefore, shown by $C = C_1 + \cdots + C_m$. The boundary of $Z$ is $\Lambda = \Lambda_1 + \cdots + \Lambda_m$, where the boundaries $\Lambda_k$ are parametrized by $s$, e.g. arclength, $\Lambda_k : \gamma_k = \gamma_k(s)$. For smooth $\Lambda_k$, $f$ extends smoothly to the boundary $f(C_k) = \Lambda_k$. The values of $f$ on $C$ will determine $f$ everywhere. The *circle map problem* is, thus, to find the boundary correspondences $S = S_k(\theta)$ and the conformal moduli $c_k, r_k$ such that

$$f(c_k + r_k e^{i\theta}) = \gamma(S_k(\theta)), \quad 1 \leq k \leq m,$$
where \( f \) is analytic and satisfies the normalisation condition \( f(z) = z + O(1/z), z \approx \infty \). The map \( f \) has the form

\[
f(z) = z + \sum_{k=1}^{m} \sum_{j=1}^{\infty} a_{k,j} \left( \frac{r_k}{z - c_k} \right)^j,
\]

where, for any integer \( j \), \( a_{k,j} := \frac{1}{2\pi i} \int_0^{2\pi} f(c_k + r_k e^{i\theta}) e^{-ij\theta} \, d\theta \) are the Fourier coefficients of the restriction of \( f \) to the boundary component \( C_k \).

### 2.3 Extension Theorems

Here the possibility of analytic extension of a function in \( \text{Lip}(C) \) is going to be studied. See [6] for complete proofs.

**Theorem 2.2.** A function \( h \in \text{Lip}(C) \), extends analytically to a function \( g \) in \( \Omega \cup \{\infty\} \) with \( g(\infty) = 0 \) if and only if for any \( z \in \bar{\Omega}^c \)

\[
\frac{1}{2\pi i} \int_{C} \frac{h(\zeta)}{\zeta - z} \, d\zeta = 0.
\]

We are now ready to state the main theorem of this chapter, which gives the necessary and sufficient conditions for analytic extension of the conformal map from the boundary into the domain in terms of the Fourier coefficients \( a_{k,j} \). This theorem is the logical foundation of the numerical technique whose discretisation is implemented in the code. We will need the binomial series

\[
\frac{1}{(1 - \Delta)^k} = \sum_{j=0}^{\infty} B_{k,j} \Delta^j, \ |\Delta| < 1,
\]
where the binomial coefficients $B_{k,j}$ are given by

$$B_{k,j} = \frac{k(k+1) \cdots (k+j-1)}{j!}, \quad k \geq 1, \quad j \geq 1, \quad B_{k,0} = 1.$$ 

**Theorem 2.3.** Let $\Omega$ be the region exterior to the circles $C_k : c_k + r_k e^{i\theta}, 1 \geq k \geq n$. Suppose $f \in \text{Lip}(C)$ has the Fourier series representation

$$f(c_k + r_k e^{i\theta}) = \sum_{j=-\infty}^{+\infty} a_{k,j} e^{ij\theta}, \quad 1 \geq k \geq n.$$ 

Then $f$ extends analytically into $\Omega$ with $f(z) = z + O(1/z)$ for $z \approx \infty$ if and only if

(i) for all centers $c_k$

$$\frac{1}{2\pi i} \int_{C_k} \frac{f(\zeta) - \zeta}{(\zeta - c_k)^j} d\zeta = 0, \quad j \geq 1;$$

(ii) or equivalently,

$$a_{k,j} - \left(\frac{r_k}{c_l - c_k}\right)^j \sum_{l \neq k} \sum_{\nu=0}^{\infty} B_{j+1,\nu} \left(\frac{r_l}{c_k - c_l}\right)^{\nu+1} a_{l,-\nu-1} = r_{k,j}, \quad j \geq 0,$$

where $r_{k,0} = c_k, r_{k,1} = r_k$ and $r_{k,j} = 0$ for $j \geq 2$.

### 2.4 Linearization and Discretization

Here we show how to linearize 2.2 with respect to $f$, $S_k(\theta)$, $c_k$, and $r_k$. We start with an initial guess for $S_k(\theta)$ and correct it with the real and $2\pi$-periodic function $U_k(\theta)$. In order to linearize $U_k(\theta)$ the following approximation is going to be used.

$$\gamma_k(S_k(\theta) + U_k(\theta)) \approx \xi_k(\theta) + \gamma'(S_k(\theta))U_k(\theta), \quad 1 \geq k \geq n, \quad (2.3)$$
where $\xi_k(\theta) := \gamma_k(S_k(\theta))$. The linearisation for the corrections $\delta c_k$ and $\delta r_k$ to $c_k$ and $r_k$ is

$$(f + \delta f)(c_k + \delta c_k + (r_k + \delta r_k)e^{i\theta}) \approx f(c_k + r_k e^{i\theta}) + f'(c_k + r_k e^{i\theta})(\delta c_k + \delta r_k e^{i\theta}). \quad (2.4)$$

Equating the linear terms in equations 2.3 and 2.4 yields

$$(f + \delta f)(c_k + r_k e^{i\theta}) = \xi_k(\theta) + \gamma'(S_k(\theta))U_k(\theta) + \xi_k(\theta)(\delta r_k + \delta z_k e^{-i\theta}), \quad (2.5)$$

where

$$\xi_k(\theta) := -f'(c_k + r_k e^{i\theta})e^{i\theta} = ir_k^{-1}\gamma'(S_k(\theta))S_k'(\theta). \quad (2.6)$$

The conditions in the theorem 2.3 are applied to require that 2.5 are the boundary values of a function analytic in $\Omega$. These conditions give linear equations for $U_k, \delta c_k, \delta r_k$. The discrete form of these equations is given next.

Requiring the functions to be boundary values of a function analytic in $D$ gives a linear system for the unknown Newton updates $U_k, \delta c_k, \delta r_k$ denoted by $U$, of the form,

$$AU = b,$$

where $A$ is a symmetric positive definite matrix of the form identity plus a low rank operator. The system can be solved efficiently by the conjugate gradient method, as in the simply connected case, except that the matrix-vector multiplication now costs $O((mN)^2)$ operations instead of $O(N \log N)$. 

20
The $i + 1$st Newton updates are

\begin{align*}
S_{k}^{i+1} &= S_{k}^{i} + U_{k}^{i} \\
c_{k}^{i+1} &= c_{k}^{i} + \delta c_{k}^{i} \\
r_{k}^{i+1} &= r_{k}^{i} + \delta r_{k}^{i}.
\end{align*}
3.1 Introduction

At this point our problem is mapped to the circle space and it is ready to be solved. All we need from the original physical problem is the image of the trailing edges on the circles.

Here we are trying to solve the problem stated in chapter two, but this time with circular boundaries rather than airfoils. The complex velocity potential is obtained from two elementary flows, i.e., the uniform flow and the circulating flow, by successive application of Milne-Thomson theorem, Theorem 1.2, depending on the number of the airfoils. In the next section we will study two special cases and then the general argument is going to be set forth.

3.2 Examples Of Flows

There are three types of flow, i.e. complex velocity potentials, study of which are helpful for solving our problem. They are uniform flow, dipole flow, and circulating flow. In particular, we are interested to see what happens if we apply Milne-Thomson theorem, Theorem 1.2, to these complex velocity potentials.

3.2.1 Uniform Flow

This flow is given by the complex velocity potential
\[ w(z) = e^{i\alpha}z, \]
on \mathbb{C}, where \( \alpha \) is the angle of attack. If a circle, centered at \( c \) with radius equal to \( r \) is introduced in its way, the new complex velocity potential is

\[ w(z) = e^{i\alpha}z + \frac{r^2 e^{-i\alpha}}{z-c} + e^{-i\alpha}c. \]

There are two points that are needed to be cleared. Firstly, note that calling the new complex velocity potential \( w(z) \) is an abuse of notation; meanwhile, here it prevents confusion rather than causing it, and we will be doing so throughout this chapter. Secondly, since we are interested in the derivative of the complex velocity potential we can ignore constants or even add them if necessary. So, ignoring the constant term above, the new complex velocity potential becomes

\[ w(z) = e^{i\alpha}z + \frac{r^2 e^{-i\alpha}}{z-c}. \]

It is interesting that the resulting flow has two parts. The first summand is the initial uniform flow we started with, and the second summand is a flow from a doublet or dipole with strength equal to \( r^2 \). We, also, need to see what happens to this flow, if a circle is introduced in its way.

### 3.2.2 Flow From A Doublet Or Dipole

This flow is given by

\[ w(z) = \frac{A}{z-a} \]

where \(|A|\) is the strength of the dipole, and this flow is defined on the exterior of some circle centered at \( a \). If a new circle, centered at \( c \) with radius \( r \) is introduced in its way, in such a way that the new circle does not intersect the previous one, then the new flow is given by
\[ w(z) = \frac{A}{z - a} + \frac{\overline{A}}{\overline{c} + \frac{r^2}{z-c} - \overline{a}}. \]

By adding the constant \( \frac{\overline{A}}{\pi - \overline{c}} \) to this potential flow, we get

\[
\begin{align*}
  w(z) &= \frac{A}{z - a} + \frac{\overline{A}}{\overline{c} + \frac{r^2}{z-c} - \overline{a}} + \frac{\overline{A}}{\overline{a} - \overline{c}} \\
  &= \frac{A}{z - a} + \frac{\overline{A}}{\frac{r^2}{z-c} + \overline{c} - \overline{a}} + \frac{\overline{A}(z - c)}{(a - \overline{c})(z - c)} \\
  &= \frac{A}{z - a} + \frac{\overline{A}(\overline{a} - \overline{c})(z - c) + r^2\overline{A} + \overline{A}(\overline{c} - \overline{a})(z - c)}{-r^2(\overline{c} - \overline{a}) + (\overline{c} - \overline{a})^2(z - c)} \\
  &= \frac{A}{z - a} - \left( \frac{r}{\overline{c} - \overline{a}} \right)^2 \left( \frac{\overline{A}}{z - (c + \frac{r^2}{\pi - \overline{c}})} \right),
\end{align*}
\]

where \( c + \frac{r^2}{\overline{a} - \overline{c}} \) is the reflection of \( a \) through the new circle.

### 3.2.3 Flow Around A Point Vortex

This flow is given by

\[ w(z) = \frac{i}{2\pi} \log(z - a), \]

where \( a \) is the vortex. If a circle, centered at \( c \) with radius \( r \), is introduced in the flow and away from the vortex \( a \), the new flow would become
\[ w(z) = \frac{i}{2\pi} \log(z - a) + \frac{i}{2\pi} \log(\frac{r^2}{z - c} - a) \]

\[ = \frac{i}{2\pi} \log(z - a) - \frac{i}{2\pi} \log(\frac{r^2}{z - c} + \overline{c} - \overline{a}) \]

\[ = \frac{i}{2\pi} \log(z - a) - \frac{i}{2\pi} \left( \log(\frac{r^2 + (\overline{c} - \overline{a})(z - c)}{z - c}) \right) \]

\[ = \frac{i}{2\pi} \log(z - a) - \frac{i}{2\pi} \left[ \log(r^2 + (\overline{c} - \overline{a})(z - c)) - \log(z - c) \right] \]

\[ = \frac{i}{2\pi} \log(z - a) + \frac{i}{2\pi} \log(z - c) - \frac{i}{2\pi} \log((\overline{c} - \overline{a})(z - c + \frac{r^2}{\overline{c} - \overline{a}})) \]

\[ = \frac{i}{2\pi} \log(z - a) + \frac{i}{2\pi} \log(z - c) - \frac{i}{2\pi} \log(z - (c + \frac{r^2}{\overline{a} - \overline{c}})) - \frac{i}{2\pi} \log(\overline{c} - \overline{a}). \]

So, ignoring the last constant, we get

\[ w(z) = \frac{i}{2\pi} \log(z - a) + \frac{i}{2\pi} \log(z - c) - \frac{i}{2\pi} \log(z - \rho_c(a)), \]

where \( \rho_c(a) = c + \frac{r^2}{\overline{a} - \overline{c}} \) is the reflection of \( a \) through the new circle.

### 3.3 Special Cases

**Example 3.1.** In this example we try to find the velocity potential over one circle. Without loss of generality we may assume that this circle is the unit circle centered at the origin; moreover, assume that the trailing edge is at \((1, 0)\) and that the angle of attack is \(\alpha\). We start with the uniform flow \( w_s(z) = e^{i\alpha}z \). This is the flow in absence of the circle; thus, in presence of the circle the streaming complex velocity potential becomes
\[ w_s(z) = e^{i\alpha}z + \frac{e^{-i\alpha}}{z} \]

As mentioned earlier using \( w_s \) for the new flow is an abuse of notation, but we will be using it, since it makes notation less complicated.

This flow on its own is not enough, since it cannot necessarily keep the trailing edge stationary; that is, Kutta condition will not be satisfied necessarily. So, we need the circulating flow defined by \( w_c(z) = (i/2\pi)\log(z) \) with unit circulation around the origin. In order to find the correct circulation, \( \Gamma \), we solve the following equation for \( \Gamma \)

\[ w_s'(1) + \Gamma w_c'(1) = 0, \]

so, having

\[ w_s'(z) + \Gamma w_c'(z) = e^{i\alpha} - \frac{e^{-i\alpha}}{z^2} + \Gamma \frac{i}{2\pi} \frac{1}{z} \]

we can easily find the value of circulation as \( \Gamma = -4\pi \sin(\alpha) \), as in [3]. The correct complex velocity potential is then

\[ w(z) = w_s(z) + \Gamma w_c(z) = w_s(z) = e^{i\alpha}z + \frac{e^{-i\alpha}}{z} - 4\pi \sin(\alpha) \frac{i}{2\pi} \log(z) \]

This example illustrates the general pattern for solving the problem. We first find the correct streaming flow, then the flow with unit circulation around the airfoil, and finally we find the correct circulation by solving the equation which keeps the trailing edge stationary. This pattern can be generalized to several circles as well; however, there is a crucial difference when we try to find the proper streaming and circulating flows with more than one circle. This difference is illustrated in the following example.
Example 3.2. Suppose that two circles, $C_i$, centered at $c_i$, with radii $r_i$, and trailing edges at $T_i$, $i = 1, 2$, are given. We want to find the correct complex velocity potential with angle of attack $\alpha$.

In order to find the streaming flow, we start with $w_s(z) = e^{i\alpha}z$, similar to the previous example. Now, introducing the first circle in its way, we get

$$w_s(z) = w_1(z) = e^{i\alpha}z + \frac{r_1^2 e^{-i\alpha}}{z - c_1}.$$

When introducing the second circle in the stream of the resulting flow, the first summand will change as the uniform flow and the second summand will change as a dipole flow and we get

$$w_s(z) = w_2(z) = e^{i\alpha}z + \frac{r_2^2 e^{-i\alpha}}{z - c_1} + \frac{r_2^2 e^{-i\alpha}}{z - c_2} - \frac{\frac{r_1^2 r_2^2 e^{i\alpha}}{z - c_1}}{(c_2 - c_1)^2} - \frac{1}{z - c_21},$$

where $c_{21}$ is the reflection of $c_1$ through $C_2$. Note that $w_1$ has the circle $C_1$ as its streamline. When we compute $w_2$, it has the circle $C_2$ as its streamline; the circle $C_1$, however, is no longer a streamline. Because of this, we have to introduce $C_1$ as a new circle in the way of the tail of $w_2$, third and forth summands above, to get a flow that has $C_1$ as streamline, but this time $C_2$ is not going to be streamline. Thus, we need to keep applying the Milne-Thomson theorem and this ongoing reflection of terms would give us the complex velocity potential corresponding to the streaming flow as a series

$$w_s(z) = e^{i\alpha}z + \sum_{|\nu| = 1}^{\infty} \frac{\sum((-1)^{|\nu| + 1}) e^{i(-1)^{|\nu|}} A_{\nu}}{z - c_{\nu}} A_{\nu},$$

where $\nu$ is multi-index, i.e. a finite sequence of 1 and 2 with no two consecutive members equal, and $|\nu|$ is the length of the multi-index. Here $c_{\nu}$ is the reflection of the centers and it should be read from right to left. For example $c_{2121}$ is the reflection of $c_1$ in $C_2$, then in $C_1$, and finally in $C_2$. The coefficients $A_{\nu}$ are constants resulted from application of the
Milne-Thomson theorem, e.g. \( A_{21} = \frac{r_1^2 r_2^2}{(c_2 - c_1)^2} \) in above. The most important question is the convergence of this series and it will be studied later in this chapter.

This flow does not satisfy the Kutta condition; thus, we need circulating flow, as in the previous example. The difference here is that, though, we are trying to find two circulating flows. One with unit circulation around \( C_1 \) and zero circulation around \( C_2 \), and the other unit circulation around \( C_2 \) and zero circulation around \( C_1 \). We start with the former, the latter being similar to that. Let \( w_{c_1}(z) = (i/2\pi) \log(z - c_1) \). This flow has \( C_1 \) as streamline; however, \( C_2 \) is not a streamline. So, we need to apply the Milne-Thomson theorem, as above to get

\[
w_{c_1}(z) = \frac{i}{2\pi} \log(z - c_1) + \frac{i}{2\pi} \log(z - c_2) - \frac{i}{2\pi} \log(z - c_{21}),
\]

where \( c_{21} \) is as above. Now, this \( w_{c_1} \) has \( C_2 \) as streamline but not \( C_1 \); so, the Milne-Thomson theorem is to be applied to the tail, i.e. the last two summands, and this procedure is to be done consecutively to get

\[
w_{c_1}(z) = \frac{i}{2\pi} \sum_{|\nu|=1}^{\infty} (-1)^{|\nu|-1} \log(z - c_{\nu}),
\]

where \( \nu \) is as in the streaming case. We can find \( w_{c_2}(z) \) with a similar argument. Again, the proof of the convergence of these two series is left for a later section of this chapter.

In order to apply the Kutta condition, first let \( v_1 = \frac{dw_{c_1}}{dz} \), \( v_2 = \frac{dw_{c_2}}{dz} \), and \( v_s = \frac{dw_s}{dz} \). At the trailing edges velocity should be zero; thus, we are looking for \( \Gamma_1, \Gamma_2 \) such that

\[
\begin{align*}
v_s(T_1) + \Gamma_1 v_1(T_1) + \Gamma_2 v_2(T_1) &= 0 \\
v_s(T_2) + \Gamma_1 v_1(T_2) + \Gamma_2 v_2(T_2) &= 0
\end{align*}
\]

This in turn would give us

\[
\begin{bmatrix} v_1(T_1) & v_2(T_1) \\ v_1(T_2) & v_2(T_2) \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix} = -\begin{bmatrix} v_s(T_1) \\ v_s(T_2) \end{bmatrix}.
\]
The solution of this system of equations will give us the correct circulations we are looking for, and the correct complex velocity potential will be

\[ w(z) = w_s(z) + \Gamma_1 w_{c_1}(z) + \Gamma_2 w_{c_2}(z) \]

This was a rather long example; meanwhile, it shows us where the series are coming from, and moreover, it illustrates almost all of the steps used in the algorithm coded and the logic behind it.

### 3.4 Convergence Results

In the following we review the calculation of streaming and circulating flow in finitely connected domains in the plane exterior to \( m \) non-overlapping disks. Disks are introduced into a given flow and their influence on the complex velocity potential is given by the Milne-Thomson circle theorem as above. It is going to be shown that this process is equivalent to the method of images based on reflecting singularities repeatedly across the circular boundaries.

In the following \( \Omega \) is an \( m \)-connected, unbounded circular domain. That is \( \partial\Omega \) consists of \( m \) disjoint circles \( C_j = \{ z \in \mathbb{C} : |z - c_j| = r_j \} \), \( j = 1, \ldots, m \), and \( \Omega = \mathbb{C} \setminus \bigcup_{j=1}^{n} \overline{D}_j \), where \( D_j = \{ z \in \mathbb{C} : |z - c_j| < r_j \} \), are \( m \) disks with mutually disjoint closures, i.e. \( \overline{D}_i \cap \overline{D}_j = \emptyset \) when \( i \neq j \).

#### 3.4.1 Reflections In Circles

We define the reflection of \( z \) through a circle \( C \) centered at \( c \) with radius equal to \( r \) as
\[ z^* = \rho_C(z) := c + \frac{r^2}{z - c} \]

and we say \( z \) and \( z^* \) are symmetric points with respect to the circle \( C \). If \( C = C_\tau \) where \( \tau \) is an index of a circle, then \( \rho_{C_\tau} \) will be denoted by \( \rho_\tau \).

Fixing an index \( \nu_1 \in \{1, \ldots, m\} \), we reflect \( c_i \) through the circle \( C_j \) to form \( c_{ji} = \rho_j(c_i), i \neq j \).

In general, the reflections of the centers are labeled with multi-indices \( \nu = \nu_1\nu_2 \cdots \nu_n \) with \( \nu_j \in \{1, \ldots, m\} \), \( \nu_k \neq \nu_{k+1}, k = 1, \ldots, n-1 \), and we write \( |\nu| \) to denote the length of \( \nu \), i.e., \( |\nu| = n \).

**Definition 3.3.** The set of multi-indices of length \( n \) will be denoted 

\[ \sigma_n = \{\nu_1\nu_2 \cdots \nu_n : 1 \leq \nu_j \leq m, \nu_k \neq \nu_{k+1}, k = 1, \ldots, n-1\}, \quad n > 0, \]

and \( \sigma_0 = \phi \), in which case \( \nu i = i \). Also

\[ \sigma_n(i) = \{\nu \in \sigma_n : \nu_n \neq i\}, \]

denotes sequences in \( \sigma_n \) whose last factor never equals \( i \).

**Lemma 3.4.** \( c_\nu = \rho_{\nu_1}(\rho_{\nu_2}(\cdots (\rho_{\nu_{n-1}}(c_{\nu_n}) \cdots ))) \) for \( \nu = \nu_1\nu_2 \cdots \nu_n \in \sigma_n \).

In order to state our convergence results, we need the following definitions. It is essential to have the disjointness (separation) of the \( m \) boundary circles \( C_j \) expressed analytically. We do this with the nonoverlap inequalities

\[ \mu^{sep}_{jk} := \frac{r_j + r_k}{|c_j - c_k|} < 1, \quad j \neq k, \quad 1 \leq j, k \leq m. \quad (3.1) \]

We now define the separation modulus of the region

\[ \Delta := \max_{i,j;i \neq j} \mu^{sep}_{ij} \quad (3.2) \]
for the m boundary circles $C_j$, c.f., [22, p. 501]. Let $\tilde{C}_j$ denote the circle with center $c_j$ and radius $r_j/\Delta$. Then, geometrically, $1/\Delta$ is the smallest magnification of the $m$ radii such that at least two $\tilde{C}_j$ just touch. We will use the following inequality from [22, p. 505].

**Lemma 3.5.**

$$\sum_{\nu \in \sigma_{n+1}} r_{\nu}^2 \leq \Delta^4 n \sum_{i=0}^{m} r_i^2. \quad (3.3)$$

### 3.4.2 Convergence Of The Streaming Flow

We start with the streaming flow and work with the complex velocity and give some convergence results. It is interesting that this case was treated in [7, 8]. There the velocity potential $w(z)$ for streaming flow around an assembly of disks $D_j$ in the plane with constant velocity $\overline{A}$ at $\infty$ and $\text{Im } w(z) = \text{constant}$ for $z \in C_j$ is given as

$$w(z) = Az + \sum_{\nu} \frac{A_{\nu}}{z - c_{\nu}}, \quad (3.4)$$

where the indices $\nu$ are as above and the coefficients are generated recursively from $A_j = r_j^2 \overline{A}, j = 1, \ldots, m$ by

$$A_{j\nu} = -\left(\frac{r_j}{c_{\nu} - c_j}\right)^2 \overline{A}_{\nu}. \quad (3.5)$$

Note that for convenience here and below, whenever we write an index like $j\nu$ we implicitly assuming that the indices do not repeat. That is, for $j\nu = j\nu_1 \nu_2 \ldots, j \neq \nu_1.$) This can be seen from the fact that for $z \in C_j$ the terms

$$\frac{A_{\nu}}{z - c_{\nu}} + \frac{A_{j\nu}}{z - c_{j\nu}} \quad (3.6)$$
in \( w(z) \) are the sum of two complex conjugates and therefore real. The reflections \( c_\nu \) of the circle centers are generated in the same way for \( w(z) \) and for \( S(z) \), the singularity function in [14].

We can prove convergence of the series for \( w(z) \) if the circles are separated well enough. The convergence result and accuracy of our numerical examples are given by the sum of the magnitudes of the coefficients \(|A_\nu|\) at each level \( N \) and how well the series truncated after \( N \) levels of reflection,

\[
 w_N(z) := A z + \sum_{|\nu|=1}^{N+1} \frac{A_\nu}{z - c_\nu} \tag{3.7}
\]
satisfies the boundary conditions \( w_N(z) = \text{constant} (= \text{average value on each circle} \neq 0) \) for \( z \in C_i \).

**Theorem 3.6.** For \( m \geq 2 \), \( w_N(z) \) converges uniformly for \( z \in \overline{\Omega} \) to a function \( w(z) \) by the following estimate

\[
 |w_N(z) - w_{N-1}(z)| = O((\Delta^2(m - 1))^{N+1}).
\]

for regions satisfying the separation condition

\[
 \Delta < \frac{1}{(m - 1)^{1/2}}.
\]

**Proof.** Suppose \( c_\nu \in D_i, i \neq j \). Note that \( |c_i - c_j| \leq |c_\nu - c_j| + r_i \). By the definition of \( \Delta < 1 \), we have

\[
 r_i + r_j \leq \Delta |c_i - c_j| \leq \Delta |c_\nu - c_j| + \Delta r_i < \Delta |c_\nu - c_j| + r_i.
\]

This gives

\[
 \frac{r_j}{|c_\nu - c_j|} < \Delta \quad \text{and} \quad |A_{j\nu}| < \Delta^2 |A_\nu|.
\]

For \( z \in \overline{\Omega}, 1/|z - c_\nu| \) is bounded away from 0. Therefore, the \( N \)th-level terms of \( w(z) \) are bounded by
\[
\sum_{|\nu| = N} \frac{|A_\nu|}{|z - c_\nu|} = O((\Delta^2(m - 1))^N).
\]

We can also prove a convergence result for the boundary condition for \( w(z) \) similar to the result for \( S(z) \), above.

**Theorem 3.7.** If \( \Delta < (m - 1)^{-1/2} \) then for \( z \in C_i, i = 1, \ldots, m \)

\[
\text{Im}\{w_N(z)\} = \text{Im}\{Ac_i\} + O((\Delta^2(m - 1))^N)
\]

and therefore

\[
\text{Im}\{w(z)\} = \text{constant on } C_i.
\]

**Proof.** Note first for \( z = c_i + r_ie^{i\theta}, (s_i = c_i) \) that, as in the Milne-Thomson circle theorem,

\[
Az + \frac{A_i}{z - c_i} = Ac_i + Ar_ie^{i\theta} + \overline{Ar_ie^{i\theta}} = Ac_i + 2\text{Re}\{Ar_ie^{i\theta}\}.
\]

Therefore,

\[
\text{Im}\{Az + \frac{A_i}{z - c_i}\} = \text{Im}\{Ac_i\} = \text{constant on } C_i.
\]

Recall from Lemma 3.4, that \( c_i\nu = \rho_i(c_\nu) \). We pair the remaining terms \( A_\nu/(z - c_\nu) \), where \( \nu_1 \neq 1 \), in \( w(z) \) with their reflections \( A_\nu/(z - c_\nu) \) as in the proof of the boundary condition for \( S(z) \) in [14]. The imaginary part of the sum of these two terms will be 0. The error for the truncated series \( w_N(z) \) can be estimated as in the \( S(z) \) case. Here is the pairing for \( z \in C_i, \)
\[ w_N(z) = Az + \sum_{|\nu|=1}^{N+1} \frac{A_\nu}{z - c_\nu} \]

\[ = Az + \frac{A_i}{z - c_i} + \sum_{|\nu|=1, \nu \neq i}^N \left( \frac{A_\nu}{z - c_\nu} + \frac{A_i\nu}{z - c_i\nu} \right) + \sum_{|\nu|=N+1, \nu \neq i} \frac{A_\nu}{z - c_\nu}. \]

The imaginary parts of the paired terms are constant. The last term is small due to the decay of the \( A_\nu \)'s, since for \( z \in C_i \) and \( \nu_1 \neq i \) (i.e., \( c_\nu \in D_{\nu_1} \neq D_i \)), there exists a \( \delta > 0 \), such that \( |z - c_\nu| > \delta \). Therefore,

\[ \frac{|A_\nu|}{|z - c_\nu|} < \frac{1}{\delta} |A| \left( \Delta^2 (m - 1) \right)^N \]

The circulation around the circles are easy to be evaluated using the complex velocity potential.

**Theorem 3.8.** The circulation \( \Gamma_{C_i} \) of the streaming flow \( w(z) \) around the circles \( C_i \) is 0.

**Proof.** Since

\[ w'(z) = A - \sum_\nu \frac{A_\nu}{(z - c_\nu)^2}, \]

the residues of \( w'(z) \) are zero in any closed curve. In particular,

\[ \Gamma_{C_i} + iF_{C_i} = \int_{C_i} w'(z)dz = 0, \]

so the circulation \( \Gamma_{C_i} = 0. \)
3.4.3 Convergence Of The Circulating Flow

We will show that $w_{\Gamma_i}$ has the form

$$-\frac{2\pi}{i\Gamma_i} w_{\Gamma_i}(z) = \ln(z - c_i) + \sum_{\nu \in \sigma_n(i), n=1}^{n=\infty} (-1)^{n+1} \left( \ln(z - c_{\nu}) - \ln(z - c_{\nu i}) \right).$$

It will be easier to show convergence of the series and satisfaction of the conditions above by working with the velocity,

$$i \frac{2\pi}{\Gamma_i} w'_{\Gamma_i}(z) = \frac{1}{z - c_i} + \sum_{\nu \in \sigma_n(i), n=1}^{n=\infty} (-1)^{n+1} \left( \frac{1}{z - c_{\nu}} - \frac{1}{z - c_{\nu i}} \right).$$

Then $w_{\Gamma_i} = \int w'_{\Gamma_i}(z) dz$. It is obvious that $w'_{\Gamma_i,N}(\infty) = w'_{\Gamma_i}(\infty) = 0$.

For the convergence proof and for the numerical implementation, we must work with a truncated version of the series at level $N$,

$$i \frac{2\pi}{\Gamma_i} w'_{\Gamma_i,N}(z) = \frac{1}{z - c_i} + \sum_{\nu \in \sigma_n(i), n=1}^{n=N} (-1)^{n+1} \left( \frac{1}{z - c_{\nu}} - \frac{1}{z - c_{\nu i}} \right).$$

**Theorem 3.9.** For $w_{\Gamma_i,N}(z)$, the circulation around $C_i$ is $\Gamma_i$ and the circulation around $C_j, j = 1, \ldots, m, j \neq i$ is $\Gamma_j = 0$.

**Proof.** This is obvious using the velocity, $w'_{\Gamma_i,N}(z)$, since, the residues at $c_{\nu}$ and $c_{\nu i}$ cancel in pairs in any circle, leaving only the residue at $c_i$ in $C_i$. \qed

To show that the $C_j$'s are streamlines, we must pair $c_{\nu}$'s with their reflections $c_{j\nu}$ in $C_j$, as we did in Theorem 3.7. The steps are close to those for establishing the infinite product formulas for conformal maps from the exterior of $m$ disks to the exterior of $m$ concentric circular arcs derived in [12]. We need the following results.

The boundary condition that the $C_k$ are streamlines, $\text{Im } w(z) = \text{constant on the } C_k$, are expressed by the following lemma.
Lemma 3.10. \( \text{Re}\{(z - c_k)w'_{\Gamma_i}(z)\} = 0, z \in C_k, \text{for all } i, k = 1, \ldots, m. \)

Proof. For \( z \in C_k \), we have \( z = s_k + r_ke^{i\theta} \) and since \( w_{\Gamma_i}(z) = \phi(z) + i\psi(z) \) and \( C_k \) is a streamline, \( \text{Im}\{w_{\Gamma_i}(z)\} = \psi(z) \) is constant. Therefore,

\[
0 = \frac{\partial}{\partial \theta}\text{Im}\{w_{\Gamma_i}(z)\} = \frac{\partial}{\partial \theta}\text{Im}\{w_{\Gamma_i}(c_k + r_ke^{i\theta})\}
= \text{Im}\{ir_ke^{i\theta}w'_{\Gamma_i}(c_k + r_ke^{i\theta})\} = \text{Re}\{r_ke^{i\theta}w'_{\Gamma_i}(c_k + r_ke^{i\theta})\}.
\]

First, we show that \( w'_{\Gamma_i,N} \) converges uniformly in \( \Omega \) if the circles are sufficiently well-separated. Then \( w_{\Gamma_i} \) can be defined. However, it is really the velocity \( w'(z) \) that is needed to plot the streamlines and compute the lift and pressure.

Theorem 3.11. If \( \Delta < (m - 1)^{1/4} \), \( w'_{\Gamma_i,N}(z) \) converges absolutely as \( N \to \infty \) for \( z \in \Omega \) exterior to the \( D_j \).

Proof. From above, we have that

\[
\left| \frac{2\pi}{\Gamma_i} w'_{\Gamma_i,N}(z) \right| \leq \left| \frac{1}{z - c_i} \right| + \sum_{\nu \in \sigma_{n(i)}, n=1}^{n=N} \left| \frac{1}{z - c_\nu} - \frac{1}{z - c_{\nu i}} \right|
\leq \left| \frac{1}{z - c_i} \right| + \sum_{\nu \in \sigma_{n(i)}, n=1}^{n=N} \left| \frac{c_\nu - c_{\nu i}}{(z - c_\nu)(z - c_{\nu i})} \right|.
\]

Since \( c_\nu, c_{\nu i} \in C_\nu \), where \( C_\nu = \rho_{\nu_1}\rho_{\nu_2} \cdots \rho_{\nu_{n-1}}(C_{\nu n}) \), the repeated reflection of the circle \( C_{\nu n} \), and for \( z \in \Omega \) there exists a \( \delta > 0 \) such that \( |z - c_\nu|, |z - c_{\nu i}| > \delta \), using \( |c_\nu - c_{\nu i}| \leq 2r_\nu \), the radius of \( C_\nu \), the Cauchy-Schwarz inequality gives
\[
\sum_{\nu \in \sigma_n(i)} \frac{|c_\nu - c_{\nu i}|}{|z - c_\nu||z - c_{\nu i}|} \leq \frac{2}{\delta^2} \sum_{\nu \in \sigma_n(i)} r_\nu \\
\leq \frac{2}{\delta^2} \left( \sum_{\nu \in \sigma_n(i)} r_\nu^2 \right)^{1/2} \left( \sum_{\nu \in \sigma_n(i)} 1 \right)^{1/2} \\
\leq \frac{2}{\delta^2} \left( \sum_{\nu \in \sigma_n(i)} r_\nu^2 \right)^{1/2} (m - 1)^{n/2} \\
\leq \frac{2}{\delta^2} \Delta^{2n} \left( \sum_{i=1}^m r_i^2 \right)^{1/2} (m - 1)^{n/2} \\
\leq C\Delta^{2n} (m - 1)^{n/2}
\]

by Lemma 3.5 where \( \delta = \delta_\Omega \). Therefore the series converges if \( \Delta^2 \sqrt{m - 1} < 1 \).

\[\square\]

Next, we show that the \( C_k \) are streamlines using Lemma 3.10 to show that the \( w_{\Gamma_i,N}(z) \to \) constant on \( C_k \) for circles satisfying the separation condition. We will use the formula

\[
\text{Im} \left\{ \frac{w}{w - 1} - \frac{w^*}{w^* - 1} \right\} = 0
\]

where \( w \) and \( w^* = 1/w \) are symmetric points with respect to the unit circle.

**Theorem 3.12.** If \( \Delta < (m - 1)^{-1/4} \) then

\[
\text{Re}\{(z - c_k)w'_{\Gamma_i,N}(z)\} = O((\Delta^2(m - 1))^N), z \in C_k,
\]

for all \( i, k = 1, \ldots, m \).

**Proof.** Note that

\[
i \frac{2\pi}{\Gamma_i} (z - c_k)w'_{\Gamma_i,N}(z) = \frac{z - c_k}{z - c_i} + \sum_{\nu \in \sigma_n(i), n=1}^{n=N} (-1)^{n+1} \left( \frac{z - c_\nu}{z - c_\nu} - \frac{z - c_k}{z - c_{\nu i}} \right) \\
= \frac{z - c_k}{z - c_k} + \sum_{|\nu| = 1, \nu \neq k}^{N-1} (-1)^{|\nu|+1} \left( \frac{z - c_k}{z - c_\nu} - \frac{z - c_k}{z - c_\nu_k} \right) \\
+ \sum_{|\nu| = N, \nu_i \neq k, \nu_N \neq i} (-1)^{N+1} \left( \frac{z - c_k}{z - c_\nu} - \frac{z - c_k}{z - c_{\nu i}} \right).
\]
and that we want to show that, as $N \to \infty$ for $z \in C_k$,

$$\text{Im}\left\{\frac{2\pi}{\Gamma_i}(z - c_k)w'_{\Gamma_i}N(z)\right\} = \frac{2\pi}{\Gamma_i}\text{Re}\{(z - c_k)w'_{\Gamma_i}N(z)\} \to 0.$$ 

Since $(z - c_k)/(z - c_k) = 1$, we only need to work with the two sums. For the terms in the first sum, we have that

$$\frac{z - c_k}{z - c_k} - \frac{z - c_k}{z - c_k} = \frac{(z - c_k)/(c_\nu - c_k)}{(z - c_k)/(c_\nu - c_k) - 1} - \frac{(z - c_k)/(c_\nu - c_k)}{(z - c_k)/(c_\nu - c_k) - 1}$$

$$= \frac{w}{w - 1} - \frac{w^*}{w^* - 1},$$

where $w = (z - c_k)/(c_\nu - c_k)$ and $w^* = 1/w = (z - c_k)/(c_{k\nu} - c_k)$ are reflections of each other through the unit circle. Therefore, by (3.8), the imaginary part of the first sum is 0. For the last sum, note that

$$\sum_{|\nu|=N,\nu_1 \neq k,\nu_N \neq i} \left| \frac{z - c_k}{z - c_\nu} - \frac{z - c_k}{z - c_{\nu i}} \right| = \sum_{|\nu|=N,\nu_1 \neq k,\nu_N \neq i} \frac{|(z - c_k)(c_\nu - c_{\nu i})|}{|z - c_\nu||z - c_{\nu i}|}$$

$$= O(\Delta^{2N}(m - 1)^{N/2}) \to 0,$$

as $N \to \infty$. 

\[ \square \]
CHAPTER 4
The Code and Numerical Results

4.1 Introduction

In this chapter the algorithm used in the code is going to be explained. The main emphasis is on the algorithm for the solution in the circular domain. Karmann-Trefftz map has an analytic representation; so, there is not much to explain there. The Fornberg-like map, on the other hand, uses a numerical algorithm, that was partly explained in chapter three.

4.2 The Code

The programming technique used here is object oriented programming. The classes used for solving the problem are as follows

- \textit{physicalproblem(aerfoils,angle\_attack,generation)},
- \textit{KarmanTrefftz(aerfoil\_collection)},
- \textit{Fornberg(KT,connectivity,generation)},
- \textit{circlespace(char)},
- \textit{leafnodes(connectivity,generation)},
- \textit{velocity(char)}.

There is one other class that is not mentioned above and that is \textit{characteristics}. This class is basically an auxiliary class that carries some characteristics of the flow between some of
the classes and is passed to the constructor of \textit{circlespace} and \textit{velocity} as \textit{char}. Moreover, \textit{KT} above is an instantiation of the class \textit{KarmanTrefftz}.

Here, instead of a comprehensive and exhaustive explanation of the list of methods, a general and brief overview of the logic of the program is going to be presented. The class \textit{physicalproblem(aerfoils,angle\_attack,generation)} is to be instantiated first to solve the problem. The input \textit{aerfoils} is a structure array of airfoils, which are themselves instantiations of certain airfoil classes. Moreover, \textit{angle\_attack} and \textit{generation} are the angle of attack of the flow and the generation of the reflection of the centers, where the latter is going to be explained later. In the constructor \textit{KT}, an object of \textit{KarmanTrefftz}, \textit{FB} an object of \textit{Fornberg}, and \textit{circlesolver} an object of \textit{circlespace} are instantiated. The object \textit{KT} will basically map the physical plane to a nearly circular domain using successive application of Karman-Trefftz maps. This object has the responsibility of keeping track of the trailing edges and their corresponding angles. It, also, has methods for taking the derivative of the corresponding maps for computation of the pressures. The object \textit{FB}, on the other hand, has the responsibility of finding the circular domain corresponding to our problem. This object contains a method which finds the centers and radii of the circles in the circular domain, a method which evaluates the Fornberg map, and finally one that computes the derivative of the Fornberg map.

The object \textit{circlesolver} is responsible for solving the problem in the circular domain. For the case with two circles the idea behind the algorithm for the solution in this domain is explained in example 3.2. In general, in order to solve the problem in this domain, we need to find two flows, which are, streaming flow and circulating flow. In order to find the streaming flow \( w_s(z) \), we know from the previous chapters that it looks as follows

\[
 w(z) = e^{\alpha_i z} + \sum_{\nu} \frac{A_{\nu}}{z - c_{\nu}},
\]
where $\alpha$ is the angle of attack, the subscripts $\nu$ are multi-indices as explained in the previous chapter, and the coefficients are generated recursively from $A_j = r_j^2 \alpha$, $j = 1, \ldots, m$ by

$$A_{j\nu} = -\left(\frac{r_j}{c_{\nu} - c_j}\right)^2 \alpha_{\nu}.$$  

This infinite series determines the streaming flow completely. In the code this series is truncated based on the level of reflection of the centers of the circles, and is given by the method $[\text{anu}, \text{snu}, \text{musep}] = \text{Fornberg.reflectasmi}(a, \text{cnu}, \text{rnu}, N)$, where the inputs are $a = e^{i\alpha}$, $\text{cnu}$, the centers in a vector, $\text{rnu}$, the radii in a vector, and $N$, the level of reflection. The outputs are $\text{anu}$, the coefficients $A_{\nu}$, and $\text{snu}$, the reflection of the centers $c_{\nu}$. The method $\text{velocity.streaming.velocity}(z)$ finds the streaming velocity.

At this point, we need $n$ flows, $n$ being the number of the circles, such that the $i$-th flow has circulation one around the $i$-th circle and circulation zero around the rest of the circles. Here the algorithm that is used in the code is going to be explained for the case with more than two airfoils. First, suppose that circle 1 is fixed. We are trying to find the flow that has circulation one around circle 1 and circulation zero around the rest of the circles. The terms of the series corresponding to this flow grow like a tree. The node of the tree is $w_c(z) = \frac{i}{2\pi} \log(z - c_1)$, and this has $m - 1$ circles ahead of it, $m \geq 2$; meanwhile, recall from chapter four that when a circulating flow has a circle ahead of it, two terms will be produced in the series corresponding to the resulting flow. Thus, introducing circle $k$, $k \neq 1$ in the way of the flow we get $\frac{i}{2\pi} \log(z - c_k) - \frac{i}{2\pi} \log(z - c_{k1})$. Each of the terms above looks like the initial flow we started with, but with different vorticity centers, and has $n - 1$ circles ahead of it, i.e. all the circles but the one in which vorticity center lies. Applying this pattern successively, it is quite clear that the terms of the series corresponding to this flow grow like a tree. This growth is illustrated in figure 4.1, for the case where $m = 3$.

As can be seen from the diagram, all the terms in the tree are similar; meanwhile, the difference is in the multi-indices $c_{\nu}$. The multi-indices are to be read from right to left, and
they always start, on the right, with the index of the circle we started with, in this case 1, and they end in the index of the circle that they have finally been reflected in. Instead of implementing the tree, we just need to have the collection of multi-indices, up to the \( N \)-th generation of the tree, which is the \textit{generation} passed to the constructor of \textit{physicalproblem}. Note that the class \textit{circlespace} instantiates an object of \textit{velocity}, which in turn the latter instantiates an object of \textit{leafnodes}. The class \textit{leafnodes(connectivity, generation)} will get the connectivity and the desired generation of the tree and stores the addresses of the leaves, collection of all multi-indices from the generation 2 up to the determined generation, in the field \textit{leavesadd}, by means of the static method \textit{leafnodes.leaves()}. Having the collection of multi-indices we can easily find the collection of \( m \) circulating velocities, which are the derivatives of the circulating flow, with circulation one around circle \( k \) and zero circulation.

Figure 4.1: Growth of the terms of the circulating velocity for three circles
around the rest of the circles. These circulating velocities are computed in the method `velocity.circ_velocity(z,label)`, where `label` is the label of the circle around which we want unit circulation. At this point, we need to find the correct circulations, which are the solutions of the following system of equations

\[
\begin{bmatrix}
v_1(T_1) & \cdots & v_m(T_1) \\
\vdots & & \vdots \\
v_1(T_m) & \cdots & v_m(T_m)
\end{bmatrix}
\begin{bmatrix}
\Gamma_1 \\
\vdots \\
\Gamma_m
\end{bmatrix}
= -
\begin{bmatrix}
v_s(T_1) \\
\vdots \\
v_s(T_m)
\end{bmatrix}.
\]

Having all the velocities necessary, we can easily find the correct circulations of the flow. This is done in the method `Gamma = velocity.circ_circulations()`, and then `Gamma` is used in the method `velocity.circ_velocity(z)` in order to compute the velocity in the circular domain.

Having the velocity in the circular domain, we can plot the streaming lines of the flow in the circular domain using the matlab ode solver `ode45()` for certain initial values. This is done in the method `circlespace.solution(y_lower,increment,y_upper,theta,Fornberg,style)` where `y_lower` and `y_upper` determine the bounds of the grid on the y-axis, `increment` determines the grid size in this direction, `theta` is a vector constituted of the angles corresponding to the trailing edges, and `Fornberg` is a Fornberg object which has been generated already.

### 4.3 Numerical Results

In this section we will go over some solutions of the problem with different connectivities. We start with the case of one airfoil.

#### 4.3.1 One Airfoil
Example 4.1. This is an important test case for the numerical method, since an analytical map from the airfoil to the circle, in the circular domain, exists; so, the results can be computed in two ways and then compared. The map from the circle to the Joukowski airfoil is given by the sequence of maps,

\[ z = (1 + h) \sqrt{1 + b^2} \exp(i(\theta + \theta_1)) - h + (1 + h)bi, \]
\[ Z = (z - 1)/(z + 1), \]
\[ W = Z^{1.9}, \]
\[ \omega = (2 + 2W)/(1 - W), \]

where \( b = 0.2, \ h = 0.3, \ \theta_1 = \tan^{-1}(-b). \) This sequence of functions maps the unit circle to the Joukowski airfoil and is shown in the first row of the following figure, which shows the streaming flow around the airfoil when the angle of attack is equal to \( \alpha = 5^\circ. \)

In figure 4.2, \( Ns = 101 \) points, equally spaces in \( \theta \) distributed around the airfoil and mapped with \( k(z) \) to the smooth domain. The errors between the exact Joukowski map and \( k^{-1} \circ h \) for increasing values of \( N \) and \( Ns \) are given in Figure 4.3. Note that the error can be made as small as we please, since the preimage of the one vertex at the trailing edge can always be made to be a Fourier point. This is not possible, in general, for connectivity greater than one.

Example 4.2. The figures 4.5 and 4.6 give us a comparison between the flow in absence and presence of circulation.
Figure 4.2: The three figures show the composition of the Fornberg map to the smooth spline curve with $N_s = 101$ and $N = 256$. Streamlines are plotted using the velocity potential and applying the Kutta condition to adjust the circulation so that there is a stagnation point at the trailing edge.

### 4.3.2 Two Airfoils

In the examples of this section number of Fourier points around the circles is equal to $N = 128$ and the angle of attack is equal to $\alpha = -5^\circ$; moreover, in order to compute circular velocity, reflection of the centers up to the forth generation are used.

**Example 4.3.** The flow over two cosine airfoils is shown in figure 4.7 and the table 4.2 represents the errors corresponding to the computations.

**Example 4.4.** In figure 4.8 we can see the flow over two Joukowski airfoils. All three steps of the solution are shown in this figures.
4.3.3 Three Airfoils

In the examples of this section, as in section 4.3.2, number of Fourier points around the circles is equal to $N = 128$ and the angle of attack is equal to $\alpha = -5^\circ$; moreover, in order to compute circular velocity, reflection of the centers up to the forth generation are used.

**Example 4.5.** The flow over three cosine airfoils is shown in figure 4.9.

**Example 4.6.** The flow over three cosine airfoils is shown in figure 4.10.
Figure 4.4: Pressure $C_p = 1 - |V|^2$ for exact Joukowski airfoil (left) and computed map (right).

Figure 4.5: Flow in absence of circulation

4.3.4 Four Airfoils

In the examples of this section, as in sections 4.3.2 and 4.3.3, number of Fourier points around
Table 4.1: Convergence of successive Newton-iteration errors $\|S^{i+1} - S^i\|_\infty$ at the Fourier points for the Fourier series map to the near-circular region with corners in Figure 4.2.

<table>
<thead>
<tr>
<th>Newton iterations</th>
<th>Joukowski airfoil</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 256, N_s = 101$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>2</td>
<td>3.0e-01</td>
</tr>
<tr>
<td>3</td>
<td>2.3e-02</td>
</tr>
<tr>
<td>4</td>
<td>1.2e-04</td>
</tr>
<tr>
<td>5</td>
<td>2.1e-09</td>
</tr>
<tr>
<td>6</td>
<td>5.4e-10</td>
</tr>
<tr>
<td>7</td>
<td>5.4e-10</td>
</tr>
<tr>
<td>8</td>
<td>5.4e-10</td>
</tr>
<tr>
<td>9</td>
<td>5.4e-10</td>
</tr>
<tr>
<td>10</td>
<td>5.4e-10</td>
</tr>
</tbody>
</table>

The circles is equal to $N = 128$ and the angle of attack is equal to $\alpha = -5^\circ$; moreover, in order to compute circular velocity, reflection of the centers up to the forth generation are used. Here, the difference between using up to the forth level of reflection and the third level of reflection makes a crucial difference.

**Example 4.7.** The flow over four cosine airfoils is shown in figure 4.11.

**Example 4.8.** Figure 4.12 illustrates an exceptional case in which sufficient conditions for convergence are violated. Here $\Delta = 0.7771 > (m - 1)^{-\frac{1}{2}} = 0.7598 > (m - 1)^{-\frac{1}{2}} = 0.5774$. 

48
4.4 Conclusions and future work

We have given convergence results for the method of reflection. The only such results in the aerodynamics literature are for the case $m = 2$ [30]. However, our sufficient conditions for convergence are far from necessary, in practice, as in the case for the Schwarz-Christoffel maps [15]. In any case, the reflection method should be replaced by a more efficient method,
Table 4.2: Convergence of successive Newton-iteration errors $\|S^{i+1} - S^i\|_{\infty}$ at the Fourier points for the Fourier series map to the the near-circular regions for examples with corners in Figure 4.7.

<table>
<thead>
<tr>
<th>Newton iterations</th>
<th>Two cosine airfoils</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 128, N_s = 101$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.4e+00</td>
</tr>
<tr>
<td>2</td>
<td>3.0e-01</td>
</tr>
<tr>
<td>3</td>
<td>3.0e-03</td>
</tr>
<tr>
<td>4</td>
<td>1.6e-05</td>
</tr>
<tr>
<td>5</td>
<td>3.4e-08</td>
</tr>
<tr>
<td>6</td>
<td>1.1e-10</td>
</tr>
<tr>
<td>7</td>
<td>3.3e-13</td>
</tr>
<tr>
<td>8</td>
<td>1.7e-15</td>
</tr>
<tr>
<td>9</td>
<td>4.1e-15</td>
</tr>
<tr>
<td>10</td>
<td>9.7e-15</td>
</tr>
</tbody>
</table>

such as Crowdy’s formulation based on the Schottky-Klein prime function [10, 11] or a series-based method for computing the potentials following, e.g., [12, 15, 16]. In future work, we also plan to apply our method to realistic airfoils, e.g. Joukowski or NACA airfoils; see, e.g., [1].

Fornberg’s method will also be replaced by the by Wegmann’s similar Newton-like method. Also, Wegmann’s [29] version of Prosnak’s projection method [26] for mapping directly to the airfoil domain will be tested. The projection method does not require smooth boundaries, but is only linearly convergent.

Moreover, one of our future attempts is going to be applying the techniques studied in this
research to more realistic examples. The main goal is to combine this technique with CFD techniques. This can be a two step solution, the first step of which is using conformal mapping for regions with negligible viscosity, as first approximation, and then obtaining boundary conditions for regions with viscosity and using CFD techniques to obtain more accurate solutions.

Figure 4.8: Flow over two Joukowski airfoils
Figure 4.9: Flow over three Cosine airfoils
Figure 4.10: Flow over three Joukowski airfoils
Figure 4.11: Flow over four Cosine airfoils

Figure 4.12: Exceptional case for flow over four Cosine airfoils
LIST OF REFERENCES


APPENDIX A

Code Excerpts

Here the implementations of the airfoils used in the examples are given. The two types of airfoils used are cosine airfoil and Joukowski airfoil whose implementations follow somewhat similar pattern. In both cases a standard type of airfoil is implemented first and then three fields shift, angle, and mag_factor would give us the option of altering the position, angle, and size of the airfoil. The field style would help us to keep track of the airfoils, in case there are more than two of them. The method plot() in both cases, would plot the airfoil.

A.0.1 Cosine Airfoil

Here is the implementation of the cosine airfoil.

```matlab
classdef Cosine
properties
    trailing_edge
    inner_point
    target_trailing_edge
    target_inner_point
    mag_factor
    shift
    angle
    k
    delta
```

APPENDIX A (continued)

aerfoil_basic

style
end

methods

% 0. Constructor

function obj = Cosine(shift,angle,mag_factor,style)
    k = 4;
    n = 1001;
    tau = linspace(-pi/(2*k),pi/(2*k), n);
    trailing_edge = 0;
    inner_point = -0.9;
    target_inner_point = -0.9;
    obj.aerfoil_basic = trailing_edge - cos(4*tau).*exp(1i*tau);
    obj.trailing_edge = mag_factor*exp(angle*1i)*obj.aerfoil_basic(1) + shift;
    obj.inner_point = mag_factor*exp(angle*1i)*inner_point + shift;
    obj.target_trailing_edge = mag_factor*exp(angle*1i)*obj.aerfoil_basic(1) + shift;
    obj.target_inner_point = mag_factor*exp(angle*1i)*target_inner_point + shift;
    obj.shift = shift;
    obj.angle = angle;
    obj.mag_factor = mag_factor;
    obj.k = 4;
    obj.delta = obj.k/(2*obj.k - 1);
APPENDIX A (continued)

    obj.style = style;
end

% 1.
function aerfoil = aerfoil_generator(obj)
    aerfoil = obj.mag_factor*exp(obj.angle*1i)*obj.aerfoil_basic + obj.shift;
end

% 2.
function [trailing_edge,inner_point,target_trailing_edge, ...
    target_inner_point,delta,shift,angle] = aerfoil_points(obj)
    trailing_edge = obj.trailing_edge;
    inner_point = obj.inner_point;
    target_trailing_edge = obj.target_trailing_edge;
    target_inner_point = obj.target_inner_point;
    shift = obj.shift;
    angle = obj.angle;
    delta = obj.delta;
end

% 3.
function plot(obj)
    plot(obj.aerfoil_generator(),obj.style);
    axis equal
    hold on
    plot(obj.trailing_edge,'x')
    hold on
end
A.0.2 Joukowski Airfoil

Here is the implementation of the Joukowski airfoil.

classdef Joukowski
    properties
        trailing_edge
        inner_point
        target_trailing_edge
        target_inner_point
        mag_factor
        shift
        angle
        aerfoil_basic
        delta
    end
    methods
        % 0. Constructor
        function object = Joukowski(shift,angle,mag_factor)
            %a = .2;
            n = 1001;
            b = .2;
        end
    end
end
APPENDIX A (continued)

\[ h = .2; \]
\[ \theta_1 = \text{atan}(-b); \]
\[ \theta = \text{linspace}(0, 2\pi,n); \]
\[ \text{circle}_b = (1 + h)\sqrt{1 + b^2}\exp(i(\theta + \theta_1)) - h + (1 + h)b i; \]
\[ Z_b = (\text{circle}_b - 1)/(\text{circle}_b + 1); \]
\[ W_b = Z_b^{1.925}; \]
object.aerfoil_basic = \(2 + 2W_b)/(1 - W_b);\]
object.trailing_edge = mag_factor*exp(angle*i)*object.aerfoil_basic(1) + shift;
object.inner_point = mag_factor*exp(angle*i)*(0.2590 + 0.7050i) + shift;
object.target_trailing_edge = mag_factor*exp(angle*i)*...
object.aerfoil_basic(1) + shift;
object.target_inner_point = mag_factor*exp(angle*i)*(1.4235 + 0.1576i) + shift;
object.shift = shift;
object.angle = angle;
object.mag_factor = mag_factor;
object.delta = 1/1.925;
end

\% 1.
function aerfoil = aerfoil_generator(obj)
    aerfoil = obj.mag_factor*exp(obj.angle*i)*obj.aerfoil_basic + obj.shift;
end

\% 2.
function [trailing_edge,inner_point,target_trailing_edge, ... 
    target_inner_point,delta,shift,angle] = aerfoil_points(obj)
    trailing_edge = obj.trailing_edge;
inner_point = obj.inner_point;
target_trailing_edge = obj.target_trailing_edge;
target_inner_point = obj.target_inner_point;
shift = obj.shift;
angle = obj.angle;
delta = obj.delta;
end

A.0.3 Karmann-Trefftz Map

Here the class implementing Karmann-Trefftz map is provided. This class also includes the
derivative of composition of several inerse of Karmann-Trefftz maps as well.

classdef KarmanTrefftz
APPENDIX A (continued)

properties
    aerfoils_collection
end

methods

% 0. Constructor
function obj = KarmanTrefftz(aerfoils_collection)
    obj.aerfoils_collection = aerfoils_collection;
end

% 1.
function aerfoils_matrix = KT_mc(obj)
    [aerfoils_matrix,z2,zeta,delta] = aerfoils(obj.aerfoils_collection);
    m = size(aerfoils_matrix,1);
    z1 = zeros(size(z2,1),1);
    for i=1:m
        z1(i) = aerfoils_matrix(i,end);
    end
    for jo=1:m
        for j=1:m
            aerfoils_matrix(j,:) = KT(aerfoils_matrix(j,:),z1(jo),z2(jo),...
                zeta(jo,1),zeta(jo,2),delta(j));
            hold on
        end
        for j=jo+1:m
            z1(j) = KT(z1(j),z1(jo),z2(jo),zeta(jo,1),zeta(jo,2),delta(j));
            z2(j) = KT(z2(j),z1(jo),z2(jo),zeta(jo,1),zeta(jo,2),delta(j));
        end
    end
%figure(2)
subplot(1,3,2)
for j=1:m
    plot(aerfoils_matrix(j,:))
    hold on
    plot(real(aerfoils_matrix(j,end)),imag(aerfoils_matrix(j,end)),'x')
    hold on
end
axis equal
obj.z1points(z1);
obj.z2points(z2);
obj.zeta1points(zeta(:,1));
obj.zeta2points(zeta(:,2));
obj.connectivity(m);
end

% 2.
function curve = inverse_KT_mc(obj,curve,delta)
    z1 = obj.z1points;
    z2 = obj.z2points;
    zeta1 = obj.zeta1points;
    zeta2 = obj.zeta2points;
    m = obj.connectivity;
    for jo=m:-1:1
APPENDIX A (continued)

curve = invKT(curve,z1(jo),z2(jo),zeta1(jo),zeta2(jo),delta(jo));
end
end

% 3.
function der = KT_inv_prime(curve,z1,z2,zeta1,zeta2,delta)
    w = ((curve-zeta1)./curve - zeta2).^(1/delta);
    dinvKTdw = (z1-z2)./((1-w).^2);
    dwdzeta = (1/delta)*(((curve-zeta1)./(curve-zeta2)).^(1/delta-1));
    dwdzeta = ((zeta1-zeta2)./((curve-zeta2).^2)).*dwdzeta;
    der = dinvKTdw.*dwdzeta;
end

% 4.
function der = KT_inv_prime_mc(obj,curve)
    z1 = obj.z1points;
    z2 = obj.z2points;
    zeta1 = obj.zeta1points;
    zeta2 = obj.zeta2points;
    m = obj.connectivity;
    der = ones(size(curve));
    for i=m:-1:1
        der = der.*KT_inv_prime(curve,z1(i),z2(i),zeta1(i),zeta2(i),delta);
        curve = invKT(curve,z1(i),z2(i),zeta1(i),zeta2(i),delta);
    end
end
end
methods(Static)

1.
function out = z1points(x)
persistent z1;
if nargin
    z1 = x;
end
out = z1;
end

2.
function out = z2points(x)
persistent z2;
if nargin
    z2 = x;
end
out = z2;
end

3.
function out = zeta1points(x)
persistent zeta1;
if nargin
    zeta1 = x;
end
out = zeta1;
end
APPENDIX A (continued)

% 4.
function out = zeta2points(x)
    persistent zeta2;
    if nargin
        zeta2 = x;
    end
    out = zeta2;
end
% 5.
function out = connectivity(x)
    persistent connectivity;
    if nargin
        connectivity = x;
    end
    out = connectivity;
end
end
end

A.0.4 Address Of The Leaves

As it was mentioned in chapter four, the terms in the complex velocity potential grow like a tree. The following class is implemented in order to find the addresses of the leaves of this tree.

classdef leafnodes
APPENDIX A (continued)

properties
    connectivity
    generation
    leavesadd
end

methods
    % 0. Constructor
    function obj = leafnodes(connectivity, generation)
        obj.connectivity = connectivity;
        obj.generation = generation;
        obj.leavesadd = obj.leaves(obj.forest());
    end

    % 1.
    function modarr = eliminate(\', n, k)
        modarr = zeros(1, n-1)';
        if(k==1)
            modarr = (2:n)';
        end
        if(k==n)
            modarr = (1:n-1)';
        else
            modarr(1:(k-1), 1) = (1:(k-1))';
            modarr(k:(n-1), 1) = ((k+1):n)';
        end
    end
end
% 2.
function permut = fixedleft(obj,l,n,k)
    permut = zeros(n*(n-1)^l,1);
    if (k<=n)
        permut(:,1)=k;
    end
    for m = 0:(n-1)
        permut(m*(n-1)^l+1:(m+1)*(n-1)^l,1) = m+1;
    end
    for j=(l-1):-1:2;
        for m=0:n*((n-1)^(l-j)-1
            modarray = obj.eliminate(n,permut(m*(n-1)^(j-1)+1,j+1));
            for p=0:(n-2)
                permut(p*(n-1)^j+1+m*(n-1)^(j-1):(n-1)^j*(p+1)+m*(n-1)^j,1) = modarray(p+1);
            end
        end
    end
end
% 3.
function add = leavesaddress(obj,len,n,k)
    permut = obj.fixedleft(len+2,n,0);
    [p,q] = size(permut);
    l = zeros(p,q-1);
    for j=1:(q-1)
        l(:,j) = permut(:,j+1);
APPENDIX A (continued)

end
leaves = {}; for i=0:(n-1)
    ch = l(i*(n-1)^len+1:(i+1)*(n-1)^len,:);
    leaves{i+1} = ch;
end
add = leaves{k};
end

% 4.
function treeleaves = forest(obj)
treeleaves = {}; for i=1:obj.connectivity
    for j=1:obj.generation
        treeleaves{i}{j} = obj.leavesaddress(j,obj.connectivity,i);
    end
end
obj.leaves(treeleaves);
end
end

methods(Static)
% 0.
function out = leaves(x)
persistent leaf;
    if nargin
        leaf = x;
A.0.5 Velocity

Here the class that finds velocity in the circular domain will be presented. This class gets the necessary fields from the physical space and then finds the correct velocity in the circular domain, but first, the class characteristics is going to be presented, which is just an auxiliary class an object of which makes transferring certain data between other objects easier.

classdef characteristics
    properties
        connectivity
        angle_attack
        generation
        centre
        radius
        tedge
        snu
        anu
    end
    methods
        % 0. Constructor
end
function obj = characteristics(connectivity,angle_attack,generation,centre,radius,tedge,snu,anu)
    obj.connectivity = connectivity;
    obj.angle_attack = angle_attack;
    obj.generation = generation;
    obj.centre = centre;
    obj.radius = radius;
    obj.tedge = tedge;
    obj.snu = snu;
    obj.anu = anu;
end
end
end

Now, the class velocity

classdef velocity
    properties
        char
        leaves
        Gamma
        flow_velocity
    end
    methods
        % 0. Constructor
        function obj = velocity(char)
end
APPENDIX A (continued)

```matlab
obj.char = char;
obj.leaves = leafnodes(char.connectivity, char.generation).leavesadd;
obj.Gamma = obj.circulations(obj.circle_circulations());
obj.flow_vel();
obj.flow_velocity = obj.circlevelocity();
end

% 1.
function ref = reflection(obj, index)
    index_length = length(index);
    ref = obj.char.centre(index(index_length));
    for i = index_length-1:-1:1
        ref = obj.char.centre(index(i)) + (obj.char.radius(index(i))^2)/((ref - obj.char.centre(index(i)))');
    end
end

% 2.
function vel = circ_vel_atomic(obj, z, leaf)
    l = length(leaf);
    c1 = obj.reflection(leaf(1:(end-1)));
    c2 = obj.reflection(leaf);
    vel = ((-1)^l)*(1./(z-c1))+(-1)^(l+1)*(1./(z-c2));
end

% 3.
function vel = circ_velocity(obj, z, label)
    vel = 0;
    for l = 1:obj.char.generation
```

76
APPENDIX A (continued)

for i=1:size(obj.leaves{label}{1},1)
    vel = vel + obj.circ_vel_atomic(z,obj.leaves{label}{1}(i,:));
end
end
vel = (1i/(2*pi))*(1./(z-obj.char.centre(label)) + vel);
end

% 4.
function v_s = streaming_velocity(obj,z)
    [n,k] = size(obj.char.snu);
    v_s = exp(-obj.char.angle_attack*1i);
    for j=1:k
        for nu=1:n
            v_s = v_s - exp(((-1)^nu)*-obj.char.angle_attack*1i)*obj.char.anu(nu, j)./(z - obj.char.snu(nu, j)).^2;
        end
    end
end

% 5.
function Gamma = circle_circulations(obj)
    circular_velocity = zeros(obj.char.connectivity, obj.char.connectivity);
    for p=1:obj.char.connectivity
        for q=1:obj.char.connectivity
            circular_velocity(p,q) = obj.circ_velocity(obj.char.tedge(p),q);
        end
    end
    stream_velocity = zeros(obj.char.connectivity, 1);
for p = 1:obj.char.connectivity
    stream_velocity(p,1) = -obj.streaming_velocity(obj.char.tedge(p));
end
Gamma = circular_velocity\stream_velocity;
obj.circulations(Gamma);
end

% 6.
function vel = circle_velocity(obj,z)
    dwdz_circ = 0;
    for k=1:obj.char.connectivity
        dwdz_circ = dwdz_circ + obj.Gamma(k)*obj.circ_velocity(z,k);
    end
    % Streaming Term
    dwdz_streaming = obj.streaming_velocity(z);
    % Circulating Term
    vel = (dwdz_streaming+dwdz_circ);
end

% 7.
function flow_vel(obj)
    obj.circlevelocity(@(z)obj.circle_velocity(z));
end
end

methods(Static)

% 0.
function out = circulations(x)
persistent Gamma;
if nargin
    Gamma = x;
end
out = Gamma;
end

1. function out = circlevelocity(x)
    persistent vel;
    if nargin
        vel = x;
    end
    out = vel;
end
end
end

A.0.6 Solution In The Circular Domain

classdef circlespace
properties
    char
    vel
end
methods
    % 0. Constructor
    function obj = circlespace(char)
APPENDIX A (continued)

```matlab
obj.char = char;
obj.vel = velocity(char);
end
% 1.
function ydot = flowvelocity(obj,~,y)
    z = y(1)+1i*y(2);
    dwdz = obj.vel.flow_velocity(z);%circle_velocity(z);
    ydot = [real(dwdz);-imag(dwdz)];
end
% 2.
function plotcircles(obj,style)
    alpha = linspace(0,2*pi,2000);
    circle = zeros(obj.char.connectivity,length(alpha));
    for i=1:obj.char.connectivity
        circle(i,:) = obj.char.centre(i) + obj.char.radius(i)*cos(alpha) + 1i*(obj.char.radius(i)*sin(alpha));
    end
    subplot(1,3,3)
    for i=1:obj.char.connectivity
        plot(real(circle(i,:)),imag(circle(i,:)),style(i))
        axis equal
        hold on
        plot(real(obj.char.tedge(i)),imag(obj.char.tedge(i)),'x')
        hold on
    end
end
```
function solution(obj,y_lower,increment,y_upper,theta,Fornberg,style) % increment=.4

obj.plotcircles(style);

for ms = y_lower:increment:y_upper
    tspan = [0 30]; % time interval
    y0 =[-10;ms]; % initial position x_left-epsilon
    [~,yout] = ode45(@obj.flowvelocity,tspan,y0);
    hold on
    %figure(3)
    subplot(1,3,3)
    plot(yout(:,1),yout(:,2))
    hold on
    axis equal
    Fornberg.fornberg_map(yout(:,1)+1i*yout(:,2));
end

obj.solution_tedge(theta,Fornberg)
end

function solution_tedge(obj,theta,Fornberg)

    epsilon = zeros(1,obj.char.connectivity);
    for i=1:obj.char.connectivity
        epsilon(i) = 0.01;
    end

    for l=1:obj.char.connectivity
        tspan = [0 15];
APPENDIX A (continued)

\[ y_1 = \left[ \text{real}(\text{obj.char.centre}(l)+(1+\epsilon(l))\times\text{obj.char.radius}(l)\times\exp(\theta(l)\times1i)); \text{imag}(\text{obj.char.centre}(l)+(1+\epsilon(l))\times\text{obj.char.radius}(l)\times\exp(\theta(l)\times1i)) \right]; \]
\[ [-,yout] = \text{ode45}(@\text{obj.flowvelocity},\text{tspan},y1); \]
\[ \text{hold on} \]
\[ \text{subplot}(1,3,3) \]
\[ \text{plot}(\text{yout}(:,1),\text{yout}(:,2)) \]
\[ \text{hold on} \]
\[ \text{Fornberg.fornberg_map}(\text{yout}(:,1)+1i\times\text{yout}(:,2)); \]
\end
\]
\end
%
5.
\text{function vel = flow_vel_circle(obj,num_Fourier_points,label)}
\alpha = \text{linspace}(0,2\times\pi,\text{num_Fourier_points});
\text{circle} = \text{obj.char.centre}(\text{label}) + \text{obj.char.radius}(\text{label}).\times\exp(1i\times\alpha);
\text{vel} = \text{obj.vel.circle_velocity(circle)};
\end
%
6.
\text{function p = pressure_circle(obj,num_Fourier_points,label)}
\alpha = \text{linspace}(0,2\times\pi,\text{num_Fourier_points});
\text{circle} = \text{obj.char.centre}(\text{label}) + \text{obj.char.radius}(\text{label}).\times\exp(1i\times\alpha);
p = 1 - \text{abs}((\text{obj.vel.circle_velocity(circle)}).\times2);
\end
\end