DEVELOPMENT OF CPANEL, AN UNSTRUCTURED PANEL CODE, USING A MODIFIED TLS VELOCITY FORMULATION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Christopher R. Satterwhite

August 2015

COMMITTEE MEMBERSHIP

TITLE:	Development of CPanel, An Unstructured Panel Code, Using a Modified TLS Velocity Formulation
AUTHOR:	Christopher R. Satterwhite
DATE SUBMITTED:	August 2015
COMMITTEE CHAIR:	David D. Marshall, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Robert A. McDonald, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Kim Shollenberger, Ph.D. Professor of Mechanical Engineering
COMMITTEE MEMBER:	Nicholas Brake, MS Stork Aerospace

ABSTRACT

Development of CPanel, an Unstructured Panel Code, Using a Modified TLS Velocity Formulation

Christopher R. Satterwhite

The use of panel codes in the aerospace industry dates back many decades. Recent advances in computer capability have allowed them to evolve, both in speed and complexity, to provide very quick solutions to complex flow fields. By only requiring surface discretization, panel codes offer a faster alternative to volume based methods, delivering a solution in minutes, as opposed to hours or days. Despite their utility, the availability of these codes is very limited due to either cost, or rights restrictions.

This work incorporates modern software development practices, such as unit level testing and version control, into the development of an unstructured panel code, CPanel, with an object-oriented approach in C++. CPanel utilizes constant source and doublet panels to define the geometry and a vortex sheet wake representation. An octree data structure is employed to enhance the speed of geometrical queries and lay a framework for the application of a fast tree method. The challenge of accurately calculating surface velocities on an unstructured discretization is addressed with a constrained Hermite Taylor least-squares velocity formulation. Future enhancement was anticipated throughout development, leaving a strong framework from which to perform research on methods to more accurately predict the physical flow field with a tool based in potential flow theory.

Program results are verified using the analytical solution for flow around an ellipsoid, vortex lattice method solutions for simple planforms, as well an anchored panel code, CBAERO. CPanel solutions show strong agreement with these methods and programs. Additionally, aerodynamic coefficients calculated via surface integration are consistent with those calculated from a Trefftz plane analysis in CPanel. This consistency is not demonstrated in solutions from CBAERO, suggesting the CHTLS velocity formulation is more accurate than more commonly used vortex core methods.

ACKNOWLEDGMENTS

Throughout my time at Cal Poly, I have faced many new experiences and challenges, both intellectual and personal. Successful navigation of those challenges could not have happened without the support and guidance of friends, family, and faculty.

My advisor, Dr. Marshall, sparked my interest in fluid dynamics in undergraduate coursework, and added to that an intrigue in numerical methods through my graduate coursework and research. He always ensured I understand the process by which a solution is reached, and not just the solution itself. This teaching style resonated with my desire for a comprehensive understanding, rather than just the end product. The time spent in his office, both for this thesis and coursework, always left me with enhanced knowledge both of fluid dynamics and a new genre of music.

A special thanks goes to Dr. McDonald, for both his help with this thesis and passion for helping students enrolled in his courses. Although I'm sure there were times I made him regret keeping his office open to students all day, he always was willing to drop what he was doing and help out in any way he could. His knowledge and past experience with panel codes proved invaluable to the success of this research.

Lastly, but surely not least, I owe an immense amount of gratitude to my friends and family. My parents, Brian and Julie, continue to help me navigate through these formative years by thinking objectively, with a long term outlook. Their unwavering support is greatly appreciated. My girlfriend, Becca, has continually motivated me to keep pushing towards completion, despite the occasional setbacks and frustration. To my roommates during my time in San Luis Obispo, thanks for keeping balance in my life, always being open to getting out and doing something on the weekends when a break was needed. And lastly, a thanks to my dog, Posey, for walking the fine line between being a nice break and an unwanted distraction. To all of those that helped me get here, including those not mentioned, your contributions and support did not go unnoticed, and for that I thank you.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Acronyms

BEM	Boundary Element Method
BIE	Boundary Integral Equation
CFD	Computation Fluid Dynamics
CHTLS	Constrained Hermite Taylor Least-Squares
HTLS	Hermite Taylor Least-Squares
TLS	Taylor Least-Squares
VLM	Vortex Lattice Method

English Symbols

$\mathbf{A}$	Doublet Influence Coefficient Matrix
a	Doublet Panel Influence Coefficient
A	Area of Panel
$\mathbf{B}$	Source Influence Coefficient Matrix
b	Source Panel Influence Coefficient
b_{ref}	Reference Span
C	Wake Panel Influence Coefficient
$\mathbf{C_F}$	Vector of Force Coefficients

$\mathbf{C_M}$	Vector of Moment Coefficients
c_{ref}	Reference Chord
C_{D_i}	Induced Drag Coefficient
C_L	Lift Coefficient
C_p	Pressure Coefficient
$\mathbf{d}$	Vector from Streamline Point to Edge of Panel
$\mathbf{F}$	Force Vector
$\mathbf{h}$	Streamline Step Vector
$\mathbf{l}$	Length Vector of Vortex Filament
N	Number of Panels in Geometry
$\hat{n}$	Panel Unit Normal Vector
$\mathbf{P}$	Position Vector
p	Pressure
$\mathbf{r}$	Relative Position Vector
r	Relative Distance
RHS	Right Hand Side Vector ($= \mathbf{B}\boldsymbol{\sigma}$)
S_a	Surface of Object in Potential Flow
S_∞	Far Field Boundary of Domain
S_{ref}	Reference Area
S_w	Wake Surface
$\mathbf{V}$	Velocity Vector

V	Volume
$\mathbf{V}_{ind}$	Vortex Induced Velocity
$\mathbf{x}_{cg}$	Center of Gravity
$\bar{Y}$	Interpolation Weight

Greek Symbols

α	Angle of Attack
β	Angle of Sideslip
$\boldsymbol{\nabla}$	Del Operator
Γ	Vortex Strength
μ	Doublet Strength
$\boldsymbol{\mu}$	Vector of Panel Doublet Strengths
Φ	Velocity Potential
$\boldsymbol{\Phi}$	Velocity Potential at all Collocation Points
ρ	Density
σ	Source Strength
$\boldsymbol{\sigma}$	Vector of Panel Source Strengths
$\boldsymbol{\omega}$	Vorticity

Subscripts

cp	Quantity at Collocation Point

i	Counter for Influenced Panels
j	Counter for Influencing Panels
l	Lower Trailing Edge Panel
p	Point Along Streamline
u	Upper Trailing Edge Panel
w	Counter for Wake Panels
∞	Freestream Quantity

Superscripts

$*$	Perturbed Quantity

Chapter 1

Introduction

1.1 Motivation

Since the 1960s, panel methods have been a continually evolving analysis method in the field of fluid dynamics. Initially, due to computing resources, panel methods stood as the only practical way of obtaining the flow solution for arbitrary configurations. Under the larger umbrella of Boundary Element Methods (BEMs), panel methods reduce the dimensionality of a problem by one, allowing a three-dimensional problem to be solved using a two dimensional surface mesh. As technology has evolved, computers have grown much more powerful, making numerical methods such as the Finite Element Method (FEM) and Finite Difference Method (FDM) much more feasible. While these methods offer higher fidelity, their dimensionality is one order higher than BEMs, causing computation time to be much higher. In addition to computation time, discretizing the entire volume can be an arduous task, as the cell quality can have a significant impact on the solution. The development of unstructured meshing algorithms has made the generation of

surface meshes much simpler, giving panel codes an advantage in their ease of use. The ease of use, combined with their speed, make panel codes an ideal tool for conceptual design, or any part of the design process in which rapid design iterations take place. With the transition to unstructured algorithms, however, the widely used method of calculating the surface velocities via finite difference approximations is no longer possible. Most unstructured panel codes currently calculate the velocity via influence functions governed by the Biot-Savart law and a viscous core model to avoid issues with the singularity near panel edges. This research applies an alternative method, in the form of an enhanced Taylor series approach, taking advantage of the zero normal flow boundary condition on the surface.

While much of the research in the field of Computational Fluid Dynamics concerns the volume schemes such as the FEM and FDM, panel codes have seen their own growth as well, expanding both their capability and speed. The assumptions that govern a potential flow drive researchers to apply separate methods to the potential flow solution to model more complex aspects of the flow solution (i.e. compressibility or viscous effects). Additionally, while panel codes already boast faster computation time than solvers requiring a volume grid, acceleration algorithms such as the Fast Multipole Method have been an active area of research to further reduce the computation time, namely for large cases. All of these potential enhancements make a panel code a useful platform from which to perform research into one of many areas. While a number of panel codes have

already been developed, the existing codes generally carry one drawback or another that suggest the development of an in-house panel code at California Polytechnic State University - San Luis Obispo, would be highly beneficial.

1.2 Approach

This research seeks to take advantage of an opportunity for Cal Poly, providing both a useful tool for undergraduate students, and a platform on which graduate students can perform research. Due to their discretized nature, a panel code naturally lends itself to an object oriented language. For that reason, CPanel is written in ANSI C++, using an object oriented approach, and also utilizes modern software development practices, such as version control and unit level testing. Maintaining and developing CPanel with the source code in a version control system provides future developers with a history of the program as well as the opportunity for parallel development among multiple graduate students, simplifying the integration of new branches of research as they are performed. Along similar lines, unit testing will provide stability in the development process as well. By testing the software at the most basic level continuously through development, newly integrated software that unintentionally changes the behavior of the system can be detected and dealt with accordingly.[5]

1.3 Document Structure

The document is laid out in a manner that gives the reader a background in potential flow in Chapter 2, followed by a step by step description of the major aspects of a panel code in Chapter 3. Methods that are typically applied in these aspects are also addressed in Chapter 3. Chapter 4 dives into the specifics of CPanel's implementation, focusing on areas that differ significantly from other panel codes. Important aspects of the program design are communicated through UML diagrams, in an effort to make future navigation and modification of the program easier. Lastly, the results of CPanel are presented in Chapter 5, and verified using results from other potential flow based programs.

Chapter 2

Theory and General Numerical Implementation

The following chapter will outline potential flow theory and how the boundary element method can be applied to create a panel code. Boundary conditions will be addressed in presenting the general procedure by which a numerical solution can be computed.

2.1 Laplace's Equation

In order to arrive at the governing equation for a potential flow, one may start with the most basic of governing equations for fluid flow, the continuity equation.

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot \rho \mathbf{V} = 0 \tag{2.1}$$

Continuity states that the net flow of mass into the control volume is equal to the time rate of change of mass in the control volume. For steady flows, the time rate of change of mass is zero, leaving

$$\boldsymbol{\nabla} \cdot \rho \mathbf{V} = 0 \tag{2.2}$$

The dot product here can be expanded, separating Equation 2.2 into two terms.

$$\mathbf{V} \cdot \boldsymbol{\nabla}\rho + \rho \cdot \boldsymbol{\nabla}\mathbf{V} = 0 \tag{2.3}$$

Assuming that the fluid is incompressible, the first term disappears and the continuity equation is simplified to

$$\boldsymbol{\nabla} \cdot \mathbf{V} = 0 \tag{2.4}$$

At this stage, a scalar function called the velocity potential, Φ is introduced and defined as follows.

$$\mathbf{V} = \boldsymbol{\nabla}\Phi \tag{2.5}$$

By defining the velocity potential such that the velocity is the gradient of the velocity potential, another assumption is implied. The vorticity in a flow is defined as the curl of the velocity vector, $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{V}$. Substituting the gradient of the velocity potential for the velocity, the vorticity becomes

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{\nabla}\Phi = \mathbf{0} \tag{2.6}$$

The curl of a gradient is a vector calculus identity and is always zero. Therefore, the restriction that the flow is irrotational is necessary for a potential flow to exist. This implies that the flow must be inviscid, as viscous effects introduce vorticity into the

flow.

Equation 2.5 can now be used in conjunction with Equation 2.4 to arrive at Laplace's Equation, the only governing equation needed to model a steady, incompressible, and inviscid flow.

$$\boldsymbol{\nabla}^2\Phi = 0 \tag{2.7}$$

It is important to note that because Laplace's Equation is linear, the principle of superposition applies, meaning that the combination of a number of individual solutions is also a solution. The next section will summarize the general solution to the potential flow problem, concluding with the two fundamental solutions that lay the foundation for a panel code.

2.2 Derivation of the Boundary Integral Equation

Panel codes' ability to compute the flow solution using only a surface discretization is actually a specific case of a larger group of computational techniques for solving PDEs, called Boundary Element Methods. The BEM is used in many different engineering disciplines such as fluid dynamics, electrostatics, heat transfer and structural problems.[6] Boundary Element Methods can be split into two categories, direct methods, and indirect methods. In direct methods, the formulation of the Boundary Integral Equation (BIE) involves integrating the potential itself over the

surface. Indirect methods integrate discrete singularity elements over the surface, using influence functions for each singularity to set the singularity strengths such that they satisfy the applied boundary conditions on the surface. The following summarizes the indirect formulation of the BIE described in Katz and Plotkin.[2] For a more detailed explanation, one can consult the text itself.

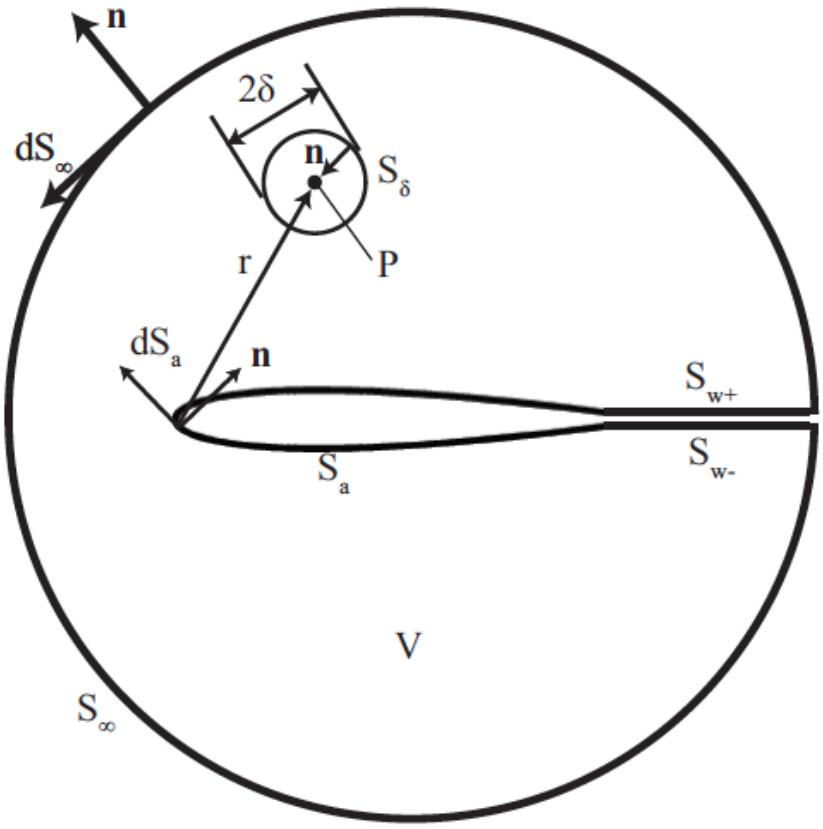


Figure 2.1: Boundaries used in BIE formulation[1]

Important to the derivation is a basic knowledge of Green's theorems that will provide a means of transforming volume integrals into surface integrals. Kellogg provides a more detailed description of the identities that will be presented in the

following paragraphs.[7] The boundaries referred to in the formulation are depicted in Figure 2.1. Although the wake boundary, S_w, may not be a physical boundary, it can be used to model a discontinuity in the velocity potential. This will be important in modeling lifting flows and the enforcement of the Kutta condition.

In order to arrive at Green's first identity, two scalar functions of position, U_1 and U_2 are defined in the volume of interest, V. U_1 and U_2 shall both be continuous in V and have continuous second derivatives as well. If these constraints hold, the divergence theorem can be written as

$$\int_V U_2 \boldsymbol{\nabla}^2 U_1 dV + \int_V (\boldsymbol{\nabla} U_1 \cdot \boldsymbol{\nabla} U_2) dV = \int_S U_2 \frac{\partial U_1}{\partial \hat{n}} dS \tag{2.8}$$

Green's theorem states that if U is harmonic and continuously differentiable, the integral of the normal derivative over the boundary of a closed region is zero. This yields

$$\int_S U_2 \frac{\partial U_1}{\partial \hat{n}} dS = 0 \tag{2.9}$$

And conversely, if U_1 and U_2 are reversed

$$\int_S U_1 \frac{\partial U_2}{\partial \hat{n}} dS = 0 \tag{2.10}$$

Subtracting Equation 2.9 from Equation 2.10 leads to Greens second identity.

$$\int_V (U_1 \boldsymbol{\nabla}^2 U_2 - U_2 \boldsymbol{\nabla}^2 U_1) dV = \int_S (U_1 \frac{\partial U_2}{\partial \hat{n}} - U_2 \frac{\partial U_1}{\partial \hat{n}}) dS = 0 \tag{2.11}$$

U_1 and U_2, are now defined as follows. Both represent solutions to Laplace's Equation in the volume, V, shown in Figure 2.1.

$$\begin{aligned} U_1 &= \frac{1}{r} \\ U_2 &= \Phi \end{aligned} \tag{2.12}$$

Due to the singularity created by U_1, a small sphere of radius, δ, is included in the surface integration and the boundary integral equation becomes,

$$\int_{S+S_\delta} \left[\frac{1}{r} \frac{\partial \Phi}{\partial \hat{n}} - \Phi \frac{\partial}{\partial \hat{n}} \left(\frac{1}{r} \right) \right] dS = 0 \tag{2.13}$$

where $S = S_\infty + S_a + S_w$. In order to formulate an expression for the potential at any point, $\Phi(\mathbf{P})$, each surface's integral will be addressed individually before arriving at the final boundary integral equation.

A spherical coordinate system is used to perform the integration along S_δ, yielding

$$\int_{S_\delta} \left[\frac{1}{r} \frac{\partial \Phi}{\partial \hat{n}} - \Phi \frac{\partial}{\partial \hat{n}} \left(\frac{1}{r} \right) \right] dS = -4\pi \Phi(\mathbf{P}) \tag{2.14}$$

Rearranging Equation 2.13 now gives the potential at the arbitrary point $\mathbf{P}$. The

coefficient, $\frac{1}{4\pi}$, is the Green's function for a three dimensional unbounded flow.[8]

$$\Phi(\mathbf{P}) = \frac{1}{4\pi}\int_S \left[\frac{1}{r}\frac{\partial\Phi}{\partial\hat{n}} - \Phi\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)\right] dS \tag{2.15}$$

In order to integrate across the surface, S_a, an inner potential, Φ_i, is defined for an interior point, $\mathbf{P}_i$. As $\mathbf{P}_i$ is outside the volume of interest, the potential at that point is zero and the normal is pointed in the opposite direction.

$$0 = -\frac{1}{4\pi}\int_S \left[\frac{1}{r}\frac{\partial\Phi_i}{\partial\hat{n}} - \Phi_i\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)\right] dS \tag{2.16}$$

The influence of the interior point can now be included by adding Equation 2.16 to Equation 2.15, resulting in an expression for the potential due to integrating across the surface S_a.

$$\Phi_a(\mathbf{P}) = \frac{1}{4\pi}\int_{S_a} \left[\frac{1}{r}\left(\frac{\partial\Phi}{\partial\hat{n}} - \frac{\partial\Phi_i}{\partial\hat{n}}\right) - (\Phi - \Phi_i)\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)\right] dS \tag{2.17}$$

The influence of the far field boundary is purely a function of the global position, $\mathbf{P}$, of the point of interest. This influence will be defined as

$$\Phi_\infty(\mathbf{P}) = \mathbf{V}_\infty \cdot \mathbf{P} \tag{2.18}$$

The only surface that still needs to be addressed is the wake. As stated earlier,

the wake can be used to model the discontinuity in the velocity potential that occurs in a lifting case. While the potential is discontinuous across the wake, the velocity is continuous, implying the that wake cannot bear any fluid dynamic loads. This makes the first terms in the surface integral go to zero, leaving

$$\Phi_w(\mathbf{P}) = -\frac{1}{4\pi}\int_{S_w} \Phi \frac{\partial}{\partial \hat{n}}\left(\frac{1}{r}\right) dS \tag{2.19}$$

With the influence of each individual surface, the total potential an arbitrary point can now be determined by summing the influence of each surface.

$$\begin{aligned}\Phi(\mathbf{P}) = \frac{1}{4\pi}\int_{S_a}\left[\frac{1}{r}\left(\frac{\partial\Phi}{\partial\hat{n}} - \frac{\partial\Phi_i}{\partial n}\right) - (\Phi - \Phi_i)\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)\right] dS \\ -\frac{1}{4\pi}\int_{S_w}\Phi\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right) dS + \Phi_\infty(\mathbf{P})\end{aligned} \tag{2.20}$$

2.3 Singularity Elements

The final boundary integral equation (2.20) reveals that the potential at any point in the volume of interest is a function of the potential and the normal derivative of the potential at the boundaries. If these values can be found using the necessary boundary conditions, then the flow solution throughout the domain can be computed. Herein lies the value in the Boundary Element Method. Because these two values are going to be recurring as the problem is constructed, they can be used

to define the strength of two fundamental elements, the source, σ and the doublet, μ.

$$\mu = \Phi_i - \Phi \tag{2.21}$$

$$\sigma = \frac{\partial \Phi_i}{\partial \hat{n}} - \frac{\partial \Phi}{\partial \hat{n}} \tag{2.22}$$

Using these definitions, Equation 2.20 can be rewritten to include these two singularity elements.

$$\Phi(\mathbf{P}) = \frac{1}{4\pi} \int_{S_a} \left[-\frac{1}{r}\sigma + \mu \frac{\partial}{\partial \hat{n}} \left(\frac{1}{r} \right) \right] dS + \frac{1}{4\pi} \int_{S_w} \mu \frac{\partial}{\partial \hat{n}} \left(\frac{1}{r} \right) dS + \Phi_\infty(\mathbf{P}) \tag{2.23}$$

This integral equation can also be written as the summation of N_s discrete point source elements and N_d point doublet elements.

$$\Phi(\mathbf{P}) = \frac{1}{4\pi} \left[\sum_{i=1}^{N_d} \mu_i \frac{\partial}{\partial \hat{n}} \left(\frac{1}{r_i} \right) - \sum_{j=1}^{N_s} \frac{\sigma_j}{r_j} \right] + \Phi_\infty(\mathbf{P}) \tag{2.24}$$

Looking at this form makes it easier to extract the influence of one individual singularity. The most easily defined is the source element.

$$\Phi_{source} = -\frac{\sigma}{4\pi r} \tag{2.25}$$

The doublet element requires more work in evaluating the partial derivative.

The detailed derivation can be seen in Katz and Plotkin,[2] with the final result being

$$\Phi_{doublet} = \frac{\mu}{4\pi}\frac{\partial}{\partial \hat{n}}\left(\frac{1}{r}\right) = -\frac{\boldsymbol{\mu}\cdot\boldsymbol{r}}{4\pi r^3} \tag{2.26}$$

These two influences can be integrated over any geometry, a line, surface, or volume, to generate influence functions for any type of discretization. The order of the singularity strength over the element, as well as the order of the discretized element are choices that must be made by the developer, based on the intended application. Reasons for choosing lower or higher order methods will be discussed in the following chapter.

2.4 Boundary Conditions

In order to solve for the strength of singularities distributed across the surface, a decision must be made with regard to the type of boundary condition applied. The far field boundary requires that

$$\lim_{r\to\infty} \Phi^* = 0 \tag{2.27}$$

where Φ^* is the perturbed velocity potential. This condition was satisfied in the derivation of the singularity elements and therefore does not need to be addressed in the numerical evaluation of the external potential flow problem. Boundary conditions on the object submerged in a potential flow, however, need specification in one of two forms.

2.4.1 Neumann Problem

The most intuitive means of imposing a wall boundary condition would be to mathematically state that the velocity normal to the surface is zero.

$$\mathbf{V} \cdot \hat{n} = 0 \tag{2.28}$$

In terms of the velocity potential, this can be written as

$$\boldsymbol{\nabla}\Phi \cdot \hat{n} = \frac{\partial \Phi}{\partial \hat{n}} = 0 \tag{2.29}$$

This boundary condition could also be used to specify a non-zero velocity, as is the case with a mass flow boundary condition that could be used to model a jet or surface transpiration. Specification of the function derivative at the surface is referred to as a Neumann boundary condition, and in the case of panel codes, may be called the direct implementation of the boundary condition. The enforcement of a Neumann boundary condition requires the gradient of the influence functions in Equations 2.25 and 2.26 to get the velocity induced at the point the boundary condition is applied to, due to all the other singularity elements in the domain. The

boundary integral being solved for this type of problem becomes

$$\begin{aligned}\boldsymbol{\nabla}\Phi(\mathbf{P})\cdot\hat{n} = \frac{1}{4\pi}\int_{S_a}\left[-\frac{\partial}{\partial\hat{n}}\frac{1}{r}\sigma + \mu\frac{\partial^2}{\partial\hat{n}^2}\left(\frac{1}{r}\right)\right]dS \\ +\frac{1}{4\pi}\int_{S_w}\mu\frac{\partial^2}{\partial\hat{n}^2}\left(\frac{1}{r}\right)dS + \frac{\partial}{\partial\hat{n}}\Phi_\infty(\mathbf{P}) = 0\end{aligned} \tag{2.30}$$

2.4.2 Dirichlet Problem

The same physical condition can be applied in an indirect manner, using a Dirichlet boundary condition. A Dirichlet boundary condition involves the specification of the function value itself at the boundary. For any point enclosed by S_a in Figure 2.1, the velocity is zero. This implies that the internal potential, Φ_i, must be constant, as proven in Lamb.[9] While this constant could be selected arbitrarily, stating that the the internal potential must be equal to the free stream velocity potential, Φ_∞, leads to a simplification of the governing boundary integral equation in Equation 2.23.

$$\frac{1}{4\pi}\int_{S_a}\left[-\frac{1}{r}\sigma + \mu\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)\right]dS + \frac{1}{4\pi}\int_{S_w}\mu\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r}\right)dS = 0 \tag{2.31}$$

Equation 2.31 can be implemented numerically by satisfying

$$\frac{1}{4\pi}\left[\sum_{i=1}^{N_d}\mu_i\frac{\partial}{\partial\hat{n}}\left(\frac{1}{r_i}\right) - \sum_{j=1}^{N_s}\frac{\sigma_j}{r_j}\right] = 0 \tag{2.32}$$

at control points placed just interior to the surface at each influencing singularity element.

Equation 2.32 does not yield a unique solution when using both source and doublet elements, as there are at least $2N$ unknowns and only N equations, with more than 2N unknowns coming from the wake panels in a lifting flow. Typically, the source strengths are prescribed based on the free stream velocity,

$$\sigma = \hat{n} \cdot \mathbf{V}_\infty \tag{2.33}$$

reducing the problem to determining the doublet strengths distributed on the body. In the case of a lifting problem, the wake surface contains additional doublet singularities, as shown in Equation 2.19. As the wake is not a solid boundary, the boundary conditions are not applied on this surface, leaving more unknowns than control points. This can be addressed by writing the wake doublet strengths in terms of the elements that shed them.

2.4.3 Kutta Condition

In a flow around a body with sharp trailing edges, the fluid traveling over the upper and lower edges will merge at the trailing edges. This phenomenon is better known as the Kutta condition, which states that "a body with a sharp trailing edge in motion through a fluid creates about itself a circulation of sufficient strength to hold the rear stagnation point at the trailing edge".[10] If the circulation, Γ, is not specified around the body, the velocity across the sharp trailing edge becomes infinite due to

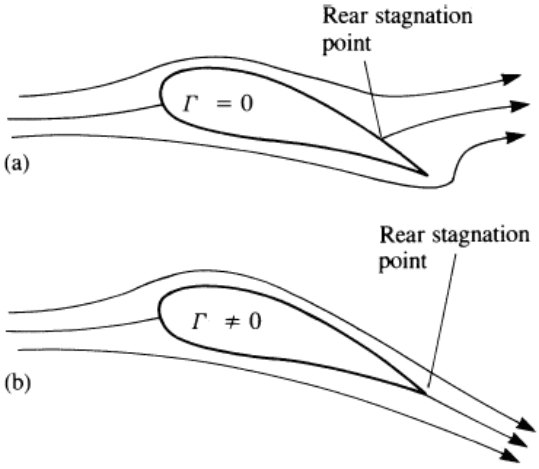


Figure 2.2: Visual Depiction of the Kutta Condition[2]

a discontinuous jump in pressure, as shown in Fig. 2.2a. If the circulation is fixed so that the stagnation point is at the trailing edge, one can see in Fig. 2.2b that the streamlines transition smoothly, as would be expected in an attached, lifting flow.

There are a number of methods used to implement this condition numerically in a BEM. The most commonly implemented method is to model the wake using a vortex sheet that is shed from the trailing edge. The difference in vorticity between

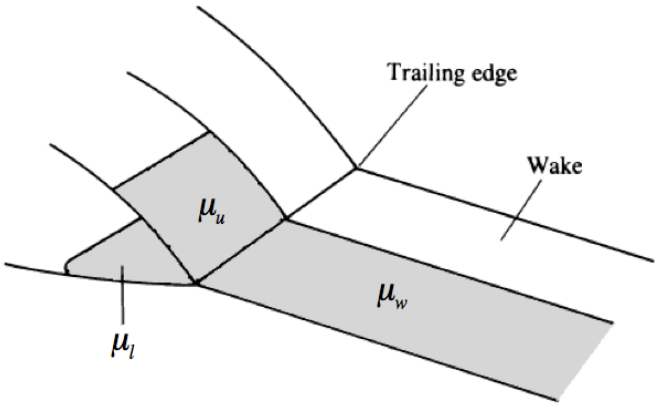


Figure 2.3: Wake Vortex Sheet for Kutta Condition Enforcement[2]

the upper and lower trailing edge panels, μ_u and μ_l respectively, is transferred into the wake panel such that

$$\mu_w = \mu_u - \mu_l \tag{2.34}$$

This ensures that the vorticity along the trailing edge segment, shared by all three panels, is zero, satisfying the Kutta condition. An illustration of how this is implemented is shown in Figure 2.3. The vortex sheet method also allows for a number of enhancements to the wake model that will be discussed in the following chapter.

Chapter 3

General Numerical Methods

This chapter will outline the numerical approach to solving the Boundary Integral Equation discussed in Chapter 2. Functional decomposition will be used in order to convey the solution process. By starting with a high level view of a typical panel code structure, each general area can be broken down into more detailed components, providing clarity to the program development process and design decisions that the developer faces. Functional decomposition also is important in the development process, as it inherently draws attention to areas in which the same function is performed, limiting the amount of redundant code and making the software easier to maintain and debug.[5] In each decomposition, methods implemented in existing panel codes will be presented, giving insight into the history of panel code development and advances being made in current research. The top level decomposition of a general panel code can be seen in Figure 3.1.

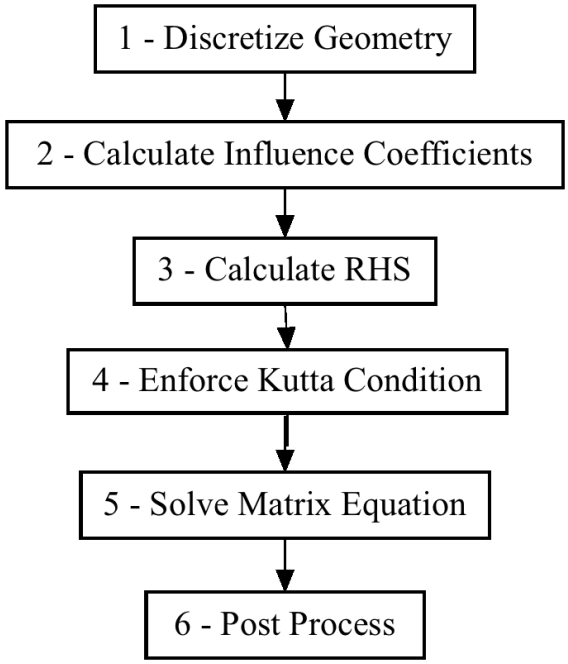


Figure 3.1: Functional Diagram for Panel Code Implementation

3.1 Geometry Discretization

In the early stages of panel code research and development, the discretization of the surface was included as a part of the panel code. The early codes required all information necessary for a simulation to be formatted input, making the discretization process complex and time consuming. This restricted solutions to those of relatively simple geometries, such as that of Figure 3.2. More recently, research into automating the discretization has resulted in software packages made specifically for that purpose.

While advances have been made in automating both unstructured and structured meshing processes, the term automation can take on a different meaning

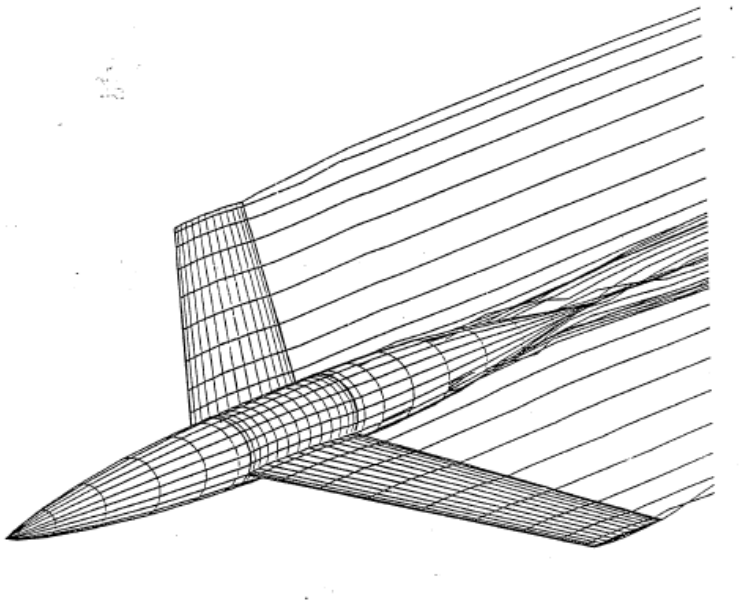


Figure 3.2: Wing Body Configuration Generated by PMARC[3]

for each type of topology. Structured meshes no longer have to be generated using formatted input, but still require the definition of hexahedral blocks that resemble the domain to control the number of elements at the intersection of surfaces. For complex geometries, the blocking process can require a significant amount of time and experience. Due to the fact that a given node can be shared by any number of cells in an unstructured mesh, the blocking process is bypassed and a mesh can be generated with less experience and time.[11] Structured meshes do offer advantages over unstructured meshes. The alignment of the cells in the direction of the vector field introduces less numerical error into the solution for the same number of cells. For example, a structured mesh can have a high density of cells in the chord wise direction for flow over a wing without also increasing the density in the spanwise

direction where gradients are not as large. Additionally, taking derivatives in a structured mesh can be done using finite differences, while an unstructured mesh calls for more complex methods. The value of unstructured meshes lies in the ease of their generation. The speed in generating them can be traded against the added resources required by the higher cell count required. By reducing the dimensionality of the problem by one, panel codes reduce the impact on computing resources that is associated with choosing an unstructured over structured mesh. This, combined with their simplicity in generation, makes unstructured meshes an ideal candidate for panel codes where turn around time is a priority. Due to the extensive research in this area, panel codes now tend to focus on the solver itself and the post processing capabilities, leaving the discretization to software developed solely for that purpose.

A number of commercial packages are available that can be used to generate an unstructured surface mesh from a CAD model. For the work of this thesis, OpenVSP was used due to it's open framework, rapid geometric modeling and automated mesh generation. The software can be downloaded from www.openvsp.org and information regarding the meshing capability can be found on the wiki accessible from the home page.

3.2 Linear System of Equations

As discussed in Chapter 2, the governing Boundary Integral Equation can be written is summation form for a system of discrete singularity elements. No matter what form the singularity element takes on, an analytical function can be derived for the influence of a unit strength element at any point in the domain. These are referred to as influence coefficients. Satisfying the given boundary condition at every collocation point in the domain results in a linear system of equations,

$$\mathbf{A}\boldsymbol{\mu} - \mathbf{B}\boldsymbol{\sigma} + \boldsymbol{\Phi}_{\infty} = \boldsymbol{\Phi}_{cp} \tag{3.1}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N_B} \\ a_{21} & a_{22} & \dots & a_{2N_B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N_B1} & a_{N_B2} & \dots & a_{N_BN_B} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N_B} \\ b_{21} & b_{22} & \dots & b_{2N_B} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N_B1} & b_{N_B2} & \dots & b_{N_BN_B} \end{bmatrix},$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{N_B} \end{bmatrix}, \quad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{N_B} \end{bmatrix},$$

$$\boldsymbol{\Phi}_\infty = \begin{bmatrix} \mathbf{V}_\infty \cdot \mathbf{p}_1 \\ \mathbf{V}_\infty \cdot \mathbf{p}_2 \\ \vdots \\ \mathbf{V}_\infty \cdot \mathbf{p}_{N_B} \end{bmatrix} \; and \; \boldsymbol{\Phi}_{cp} = \begin{bmatrix} \Phi_{cp_1} \\ \Phi_{cp_2} \\ \vdots \\ \Phi_{cp_{N_B}} \end{bmatrix}$$

The components, a_{ij} and b_{ij}, of the matrices $\mathbf{A}$ and $\mathbf{B}$ refer to the velocity potential influence at the i^{th} collocation point due to a unit strength source and doublet, respectively, located at the j^{th} singularity element. A visual representation of this is presented in Figure 3.3. Once the influence coefficient formulation is derived, the matrices can be constructed in a double loop through the N_b body

elements.

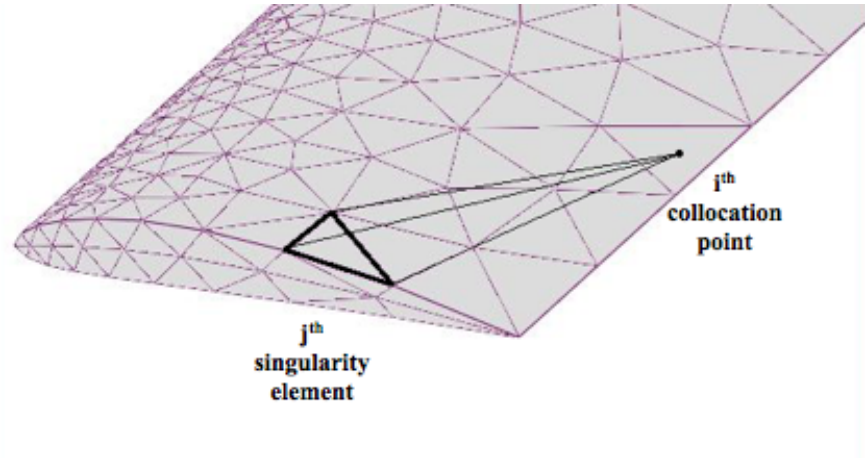


Figure 3.3: Illustration of Influence Coefficient Notation

As discussed earlier, the potential at the collocation points, $\mathbf{\Phi}_{cp}$, just interior to the surface can be specified as equal to the free stream potential, $\mathbf{\Phi}_{\infty}$. This simplifies Equation 3.1 to become

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{B}\boldsymbol{\sigma} \tag{3.2}$$

Physically, every line of Equation 3.2 is satisfying a zero normal flow boundary condition in Dirichlet form at each of the collocation points. At this stage, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ must be solved for, leaving an underdetermined system with $2N_B$ unknowns and only N_B equations. The source strengths can be set according to Equation 2.33,

accounting for the oncoming free stream flow.

$$\mathbf{RHS} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N_B} \\ b_{21} & b_{22} & \dots & b_{2N_B} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N_B1} & b_{N_B2} & \dots & b_{N_BN_B} \end{bmatrix} \begin{bmatrix} \mathbf{V}_\infty \cdot \hat{n}_{cp_1} \\ \mathbf{V}_\infty \cdot \hat{n}_{cp_2} \\ \vdots \\ \mathbf{V}_\infty \cdot \hat{n}_{cp_{N_B}} \end{bmatrix} \tag{3.3}$$

And the linear system,

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{RHS} \tag{3.4}$$

has a unique solution and can be solved using the developer's choice of linear algebra techniques. In constructing the matrices of influence coefficients, **A** and **B**, a couple of choices must be made by the developer in deriving the influence functions.

3.2.1 Influence Coefficients

The velocity potential at an arbitrary point due to a singularity element can be derived by integrating a point singularity over the surface of the panel. The generic form of this calculation is

$$a_{ij} = -\int_S \left(\frac{\hat{n}_j \cdot \mathbf{r}_{ij}}{4\pi \left|\mathbf{r}_{ij}\right|^3} \right) dS \tag{3.5}$$

$$b_{ij} = -\int_S \left(\frac{1}{4\pi \left|\mathbf{r}_{ij}\right|} \right) dS \tag{3.6}$$

for a doublet and source element, respectively, based on the results of Equations 2.25 and 2.26. The integration can be done either analytically or numerically, depending on the complexity. The influence coefficient derivation is dependent upon two choices that must be made by the developer: the order of the discretization and the order of the singularity distribution across the discretized surface.

The earliest panel codes in the 1960s, such as the Hess Code, began with flat panels and constant strength singularities. Gradually, higher order singularity distributions were adopted, before a transition back to lower order methods. In the early 1980s, the second version of the Hess Code, Hess II, used second order singularity distributions over parabolic panels. The advantages in this method lie in the ability to more accurately capture the flow physics with a coarser discretization, especially over regions of high curvature. Given that the discretization was done as formatted input, this provided a significant advantage. Early research also found that a continuous singularity distribution is required to reduce numerical instabilities that occur in a supersonic potential flow.[12] More recently, however, faster independent methods, such as those applied in CBAERO, have been developed to handle supersonic flow regions.[13] The higher order methods do come with a penalty, however, in the large increase in the number of equations being solved, as more boundary conditions must be applied at the panel edges in order to ensure the singularity strength is continuous between neighboring panels. Additionally, the derivation of the influence coefficient for that method is much

more involved.

Due to the increasingly automated discretization process, the benefits of using higher order methods are diminished, as increased accuracy can be obtained simply by generating a finer surface mesh. For this reason, CPanel applies a constant strength singularity distribution for both sources and doublets. The derivation for such a singularity was first done by Hess and Smith in the 1960s.[14,15] This early formulation, and the reformulation in Katz and Plotkin, requires that each influence coefficient calculation is performed in a local panel reference frame, and then transformed back into a global reference frame. Maskew developed a formulation that utilizes vector products to avoid the coordinate transformation and simplify the calculation.[16] The solution also utilizes the fact that a constant strength doublet element is equivalent to a vortex ring.[2] This allows the calculation to be reduced to a summation of the contribution from each of the edges of any polygon. The details of the formulation can be found in Maskew.

3.2.2 Kutta Condition Enforcement

For lifting flows, the matrix of influence coefficients must be modified such that the Kutta condition of Section 2.4.3 is satisfied. In a most basic sense, any method by which this is done specifies the circulation around the body such that the stagnation point is located at the trailing edge. This can be done in a number of ways with varying complexity and accuracy. At the root of each of these methods is the need

to model the discontinuous potential jump through the wake boundary. As a doublet represents a change in the velocity potential, these methods use some form of the doublet singularity element to model the wake.

The simplest form of a wake model consists of semi-infinite doublet panels, also known as horseshoe vortices, extending infinitely downstream from the trailing edge. The influence of such panels is well understood, as they stem from another potential flow analysis method, lifting line theory. This type of a wake model is pictured in Figure 2.3.

The velocity potential influence of each vortex sheet, just as the panels, can be written as the product of the influence coefficient and the strength of the element. The wake, however, is not a solid surface and a wall boundary condition cannot be applied on the wake panels. This adds the unknown wake strength to the system of equations without adding an equation. Equation 2.34 can be applied to eliminate the unknown wake strength and yield

$$\Phi_{iw} = C_{iw}\mu_w = C_{iw}\left(\mu_u - \mu_l\right) \tag{3.7}$$

where w is a counter for all of the wake vortex sheets, and i is counter for all of the collocation points where either a Neumann or Dirichlet boundary condition is enforced. This requires the influence of the upper and lower panels to be modified in

the matrix of influence coefficients of Equation 3.2.

$$a_{ij} = a_{ij} \pm C_{iw} \tag{3.8}$$

The influence of the wake panel is either added or subtracted from the influence of the surface panel, depending on whether the surface panel is above or below the wake panel, respectively.

While results from horseshoe vortex wake models demonstrate sufficient accuracy for most steady cases, the physical requirement that the wake is a force-free shear layer is not satisfied by this method. Enhancements to the wake model aim to satisfy this condition either by iteratively changing the shape of the wake, or allowing the wake to develop over a finite number of discrete time steps.

In the first of these methods, a number of wake grid planes are used perpendicular to the free stream, at varying distances downstream, to divide the wake into quadrilateral panels. When the solution is computed, the corners of the wake panels are convected in the direction of the perturbation velocity in the plane perpendicular to the free stream velocity. The solution is then recomputed and the process repeats itself until the movement of the wake nodes reaches some tolerance. The setup for this method can be seen in Figure 3.4. VSAERO, as well as other panel codes, employs this method.[16]

The other method of generating a force free wake follows a similar principle,

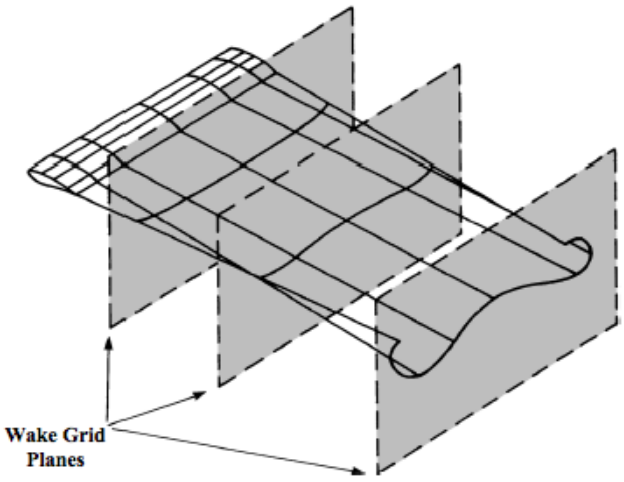
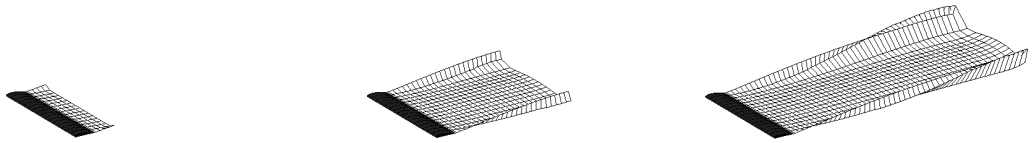


Figure 3.4: Relaxation Method for Force-Free Wake[2]

but the wake panels are instead generated by convecting the trailing edge nodes with the local velocity, not restricted to any plane, at each discrete time step. For each time step, a new row of wake panels is generated and the Kutta condition is enforced for that row. This method, illustrated in Figure 3.5, is advantageous compared to the relaxation method for two reasons. By creating wake panels purely based on the local velocity, fewer computations are needed than starting with all the wake panels already established.[17] The time stepping approach also expands the panel codes capability to model unsteady flows.

While both these methods do generate a more accurate wake shape, they both add significantly to the time needed to compute the solution. Additionally, as the wake nodes are moved in discrete time steps, a non-physical situation can occur if the wake panels intersect themselves or a solid body downstream.

Vortex particle wakes have emerged more recently as a method of generating a



(a) $t = 5\Delta t$ (b) $t = 25\Delta t$ (c) $t = 50\Delta t$

Figure 3.5: Time Stepping Wake Approach used in PMARC

force free wake without the concern of tracking the connectivity of the wake nodes. The solution process follows one similar to the time stepping approach, and therefore is more computationally expensive for steady cases than a fixed wake. Acceleration methods being applied to the standard solver are also being implemented with this wake model to make the method more feasible. The vortex partical method also has been used to capture propeller airframe interactions without requiring detailed information regarding the propeller.[18]

3.3 Post Processing

With known source and doublet strengths on the surface, the potential at each collocation point is known and the post processing can begin. Panel codes are a continually evolving analysis method and a number of independent theories and methods have been applied to the basic panel code to further enhance the post processing capability. This section will cover the basic post processing steps. The opportunity for additional enhancements will be discussed later as possible future

work.

3.3.1 Velocity Calculation

Once the singularity strengths are known, the velocity tangent to the surface at each collocation point is computed by computing the gradient of the potential. This stems directly from the definition given in Equation 2.5. Although simple in theory, the practical calculation of the gradient becomes more challenging, specifically on an unstructured surface discretization.

On a structured discretization, as the early panel codes tend to be, finite differences could be applied in the streamwise and spanwise directions. This method is described in detail in Maskew, outlining the process in general, as well as addressing special cases when the panel is on a sharp edge or at the edge of a patch.[16] Off-body velocity calculations can be done by taking the sum of the velocity influences from each panel at the desired point. The velocity influence can be calculated directly by taking the gradient of the potential influence functions in Equations 2.25 and 2.26 for a point singularity, or the influence function that results from integrating over a panel for a panel singularity. As an example, a constant strength doublet panel can be represented by a vortex ring and the velocity influence can be calculated as the sum of the influence of the N sides of the panel.[2]

The velocity induced by the side of a panel, or vortex filament, follows the

Biot-Savart Law.

$$\mathbf{V}_{ind}(x,y,z) = \frac{-\mu}{4\pi}\int_l \frac{\mathbf{r}\times d\mathbf{l}}{|\mathbf{r}|^3} \tag{3.9}$$

For a straight line segment, the integration yields

$$\mathbf{V}_{ind}(x,y,z) = \frac{\mu}{4\pi}\frac{(|\mathbf{r}_1|+|\mathbf{r}_2|)(\mathbf{r}_1\times\mathbf{r}_2)}{|\mathbf{r}_1||\mathbf{r}_2|(|\mathbf{r}_1||\mathbf{r}_2|+\mathbf{r}_1\cdot\mathbf{r}_2)} \tag{3.10}$$

The variables used in Equation 3.10 are illustrated in Figure 3.6.

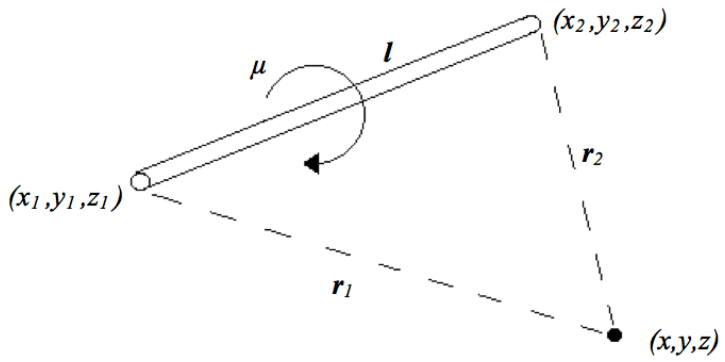


Figure 3.6: Induced Velocity from Vortex Filament

Typically, if the finite difference approach cannot be used, as is the case with an unstructured discretization, the surface velocity is computed by the same means as the off-body velocities are computed. Issues with this method arise from the singularity that occurs near a vortex filament, as is shown in Figure 3.7. Various core methods have been applied to create a more physical representation of the influencing vortex. The simplest of these is a Rankine vortex, in which the motion within the core radius is simulated by a rigid body and the induced velocity is

proportional to the distance from the vortex. Other methods, referred to as viscous core methods, allow the velocity to decay smoothly to zero as the radius decreases. The viscous core regularization shown in Figure 3.7 is drawn from the work of van Garrel.[19] These viscous core methods remain an active area of research. For a more

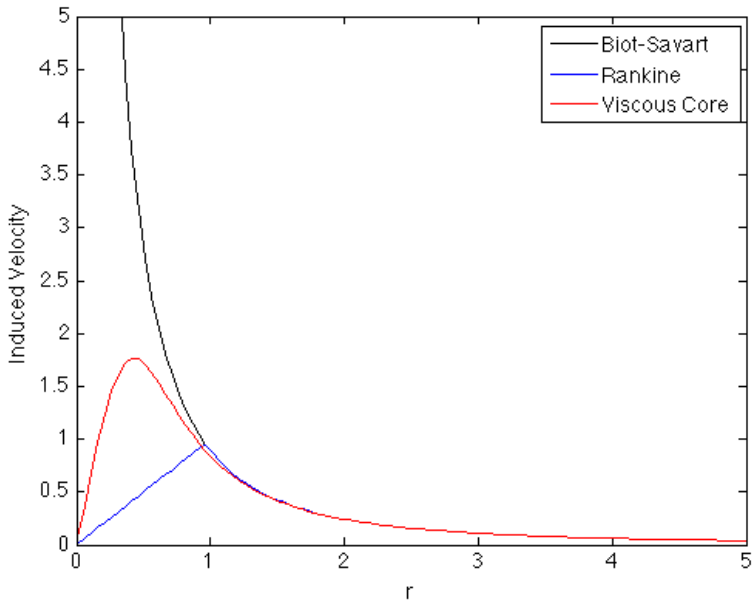


Figure 3.7: Velocity Induced by Vortex with Various Core Models

detailed explanation of various regularization techniques, one can look at the work of Winckelmans.[20] Early codes, such as PMARC, that employed vortex core models typically depended upon a user input of the core radius. Although they are not well documented, modern unstructured codes, such as CBAERO and FastAero, have managed to apply core methods that do not require user input.[13,21] These methods appear to be more robust and accurate, but still can cause a nonphysical velocity influence and adversely affect the accuracy of the solution. This will be shown and

discussed in the results section of Chapter 5.

The work of this thesis aims to avoid the issues outlined above by employing a constrained Hermite Taylor series least squares (CHTLS) method of computing the surface derivatives. Applying a least squares approach to the Taylor series representation of scattered function values is a well understood method of obtaining partial derivatives in unstructured and meshless solution methods. With only a surface discretization, however, the problem is ill posed without additional information. Including known directional derivatives solves this problem with very accurate results, as shown by McDonald and Ramos.[22] The method's specific implementation in CPanel will be further discussed in the following chapter.

3.3.2 Force and Moment Calculation

With the known local velocity at each collocation point, the pressure coefficient can be calculated for the associated panel. The pressure coefficient at the i^{th} panel is defined as

$$C_{p_i} = \frac{p_i - p_\infty}{\frac{1}{2}\rho_\infty |\mathbf{V}_\infty|^2} \tag{3.11}$$

and can be simplified for an incompressible flow to be

$$C_{p_i} = 1 - \left(\frac{|\mathbf{V}_i|}{|\mathbf{V}_\infty|} \right)^2 \tag{3.12}$$

Each panel's contribution to the force coefficients can be calculated using the

following equation.

$$\mathbf{C_{F_i}} = \frac{-C_{p_i} A_i \hat{n}_i}{S_{ref}} \tag{3.13}$$

Similarly, the panel's contribution to the moments about the center of gravity, $\mathbf{x}_{cg}$ can be calculated.

$$\mathbf{C_{M_i}} = \frac{(\mathbf{x}_i - \mathbf{x}_{cg}) \times \mathbf{C_{F_i}}}{l_{ref}} \tag{3.14}$$

The reference length, l_{ref} is either the reference chord for the pitching moment, or the reference span for the yawing and rolling moments. The total force and moment coefficients for the entire body can then be found by taking the sum of the individual contributions. For lift and drag coefficients, a coordinate transformation is required from body to wind axes. This method of calculating the lift and drag, however, can yield very inaccurate results, specifically in the induced drag coefficient. The combination of high curvature at the leading edge and large gradients in pressure combine to demand very high resolution of the surface near the leading edge in order to obtain satisfactory results.[23] While there are methods to obtain accurate lift and drag coefficients without high resolution near the leading edge, the moment calculations remain dependent on this method and therefore require a fine discretization for acceptable results.

3.3.3 Trefftz Plane Analysis

An alternative to the surface integration method for obtaining lift and drag coefficients involves the application of the integral form of the momentum equation, applied to a control volume surrounding the body. For a steady potential flow, this reduces to

$$\int_S \rho_\infty \mathbf{V} \left(\mathbf{V} \cdot \hat{n} \right) dS = \mathbf{F} - \int_S p \hat{n} dS \tag{3.15}$$

If the control volume is large, the perturbations on the boundaries, other than the one intersecting the wake, disappear as this was one of the conditions in the derivation of the boundary integral equation in Section 2.2. The divergence theorem can be used to convert the remaining surface integral into a line integral along the intersection of the wake and Trefftz plane, as shown in Figure 3.8. The resulting integrals for the lift and drag coefficients are

$$C_L = \frac{2}{|\mathbf{V}_\infty| S_{ref}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \Gamma \left(y \right) dy \tag{3.16}$$

$$C_{D_i} = \frac{1}{|\mathbf{V}_\infty|^2 S_{ref}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \Gamma \left(y \right) w \left(y \right) dy \tag{3.17}$$

The detailed derivation of the lift and drag expressions can be found in Katz and Plotkin.[2]

As for the numerical implementation of a Trefftz plane analysis, there a couple options that are very different from each other, each holding advantages and

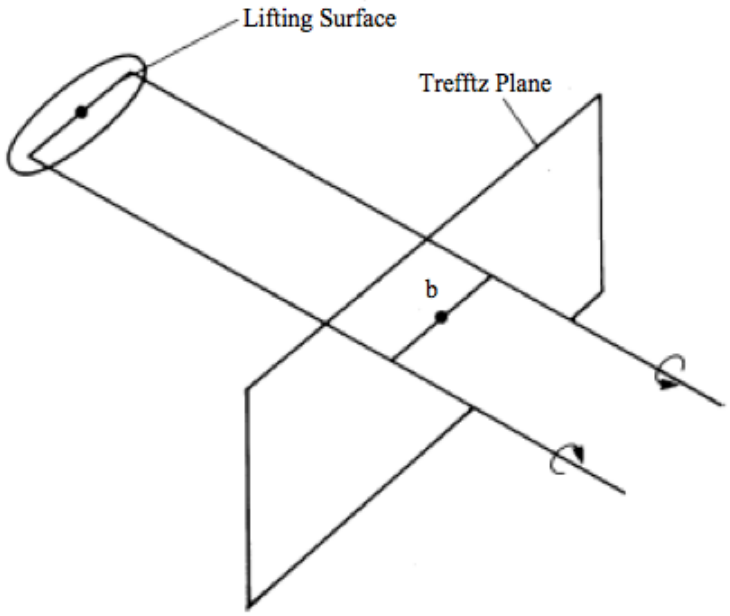


Figure 3.8: Intersection of the Wake and the Trefftz Plane[2]

disadvantages. The first method, described by Lundry, involves a series representation of the lift distribution, fit through points of known circulation. The terms of the series are then used in the induced drag calculation.[24] The advantages of this method lie in the speed of the computation, as well as the independence of the wake shape downstream. When previously discussed methods of generating a force-free wake, such as time stepping and relaxation, are employed, the difficulty of evaluating the integrals in Equations 3.16 and 3.17 remains unchanged. Additionally, the elimination of the induced downwash term, w, that can be seen in Equation 3.17, eliminates issues with the singular velocity induced by nearby wake vortex filaments. The primary issue with this method, however, is the requirement of knowledge of the local chord length at each span location where the circulation is

specified. From a numerical implementation perspective, this would either limit the application to simple planforms, or require additional computational work to find the local chord length from the given geometry.

An alternative is to prescribe a certain number of spanwise points at the intersection of the wake and Trefftz plane. The circulation, Γ, at these points can be interpolated from the known circulation of the nearest wake panels, and the induced downwash can be calculated either from a velocity influence routine employing a vortex core method, or numerical derivative of the potential at points extending perpendicular to the wake. The latter is employed in CPanel, with a cloud of points generated just above the wake in the Trefftz plane. A taylor series least squares method is then used to calculate the induced downwash from the known velocity potential at the cloud of points.

Chapter 4

CPanel Implementation

With the general numerical methods applied in panel methods discussed, the specifics of CPanel will be covered in the following sections. Only methods that differ significantly from those covered in Chapter 3 will be addressed, with results specific to those portions of the program verified in the next chapter.

4.1 Program Structure

As is the case with other numerical methods in engineering analysis, Boundary Element Methods naturally lend themselves to an object oriented approach in their development. The decomposition of the geometry into physical subcomponents provides a framework from which class definitions can be abstracted. Once the classes are defined, the data they contain and the methods that operate on that data are locked and hidden from future development. This allows for future expansion of program capabilities without having to learn all the intricacies of the program and its data structures. The abstraction of classes also restricts the

potential ripple effect of future changes made in the implementation, provided that the class interface remains the same.[25]

Written in ANSI C++, CPanel utilizes an object oriented approach in the design of the program. The abstraction of objects was primarily made from physical components of a discretized geometry. Figure 4.1 illustrates the relationship of the primary components of the program. Subclasses are shown by a solid arrow pointing from the base class to the derived, as is the case with the *panel* and *wakePanel* classes. Connections without arrows indicate a relationship, but not inheritance.

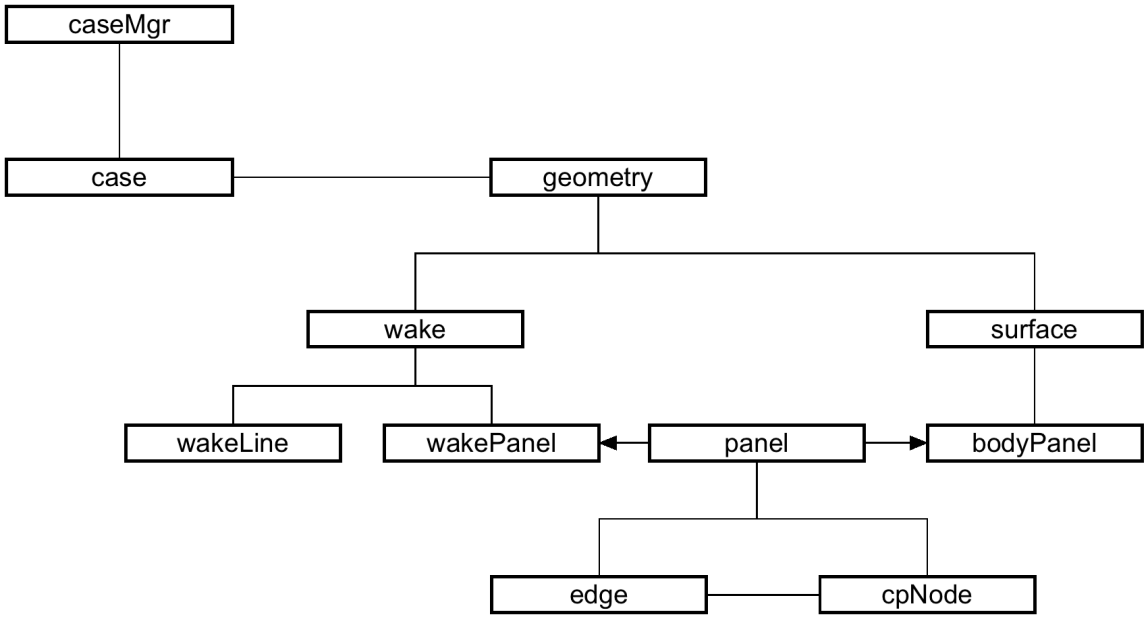


Figure 4.1: CPanel General Class Structure

The separation of the actual cases from the geometry is representative of how the solution process is divided. Figure 4.2 shows the need to create the geometry and calculate all the influence coefficients only once for all of the flow conditions

specified in the input file. The influence coefficients are only dependent on their spatial relationship to each other, and the change in flow conditions is reflected in the setting of source strengths using Equation 2.33. This separation allows the user to run a number of different solutions, while performing the most computationally expensive portion of the program only once. Additionally, if the option is turned on, CPanel will write the influence coefficient matrices to a text file for future use of the same geometry.

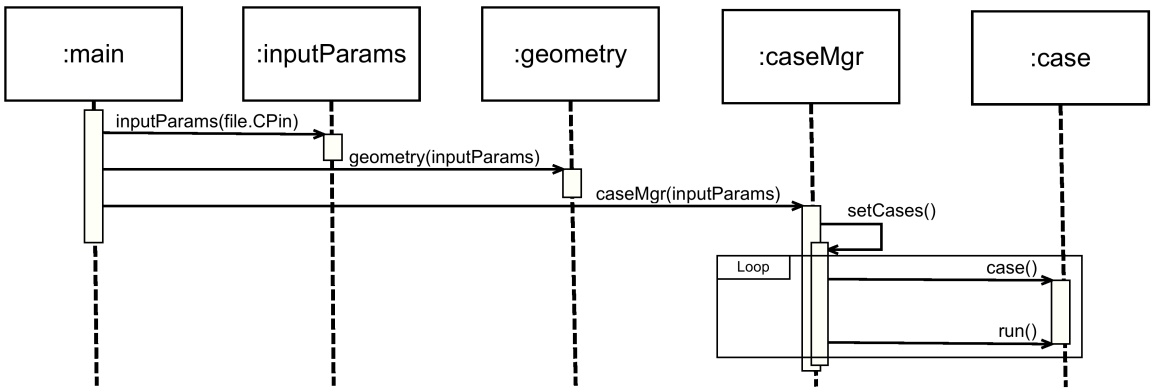


Figure 4.2: High Level Sequence Diagram of CPanel

To minimize memory requirements, while maintaining the ability of various classes to operate on commonly shared data, pointers are used heavily. The ownership of all instances under the *geometry* class is given to *geometry*, and the further abstracted objects contain pointers to those objects below it. For instance, *geometry* owns all of the *edge* instances, but *panel* contains pointers to the associated *edge* instances, and *surface* contains pointers to all the *bodyPanel* instances related to it.

Differentiation between panels that make up a physical boundary, *bodyPanels*, and those that model the discontinuous wake, *wakePanels*, stems from the distribution of both source and doublet singularities over the physical boundary, and just doublet singularities over the wake surface. Methods that are performed in the same way for both types of panels are defined in the parent class, *panel*, and those that are specific to one type of panel are defined in the corresponding derived class. The complete separation of the surfaces from the wakes immediately shows the advantages of object oriented development. With modification of the Kutta condition enforcement method being a planned enhancement to CPanel, the developer of that portion of the code only requires knowledge of the wake's interface with geometry in order to implement the new method, without worrying about adverse effects throughout the rest of the software.

In order to ensure that the interface of the classes remains the same through future development, a practice called unit level testing is employed. Unit level testing is a practice that helps in the maintenance and debugging of software throughout the entire development process. Unit tests are written without requiring the knowledge of implementation details within a class or a method, and are therefore sometimes referred to as "black box" testing methods. Maintenance of the software is made easier by the use of these tests to ensure that the class interfaces remain unchanged amidst changes to the internal implementation, especially when multiple developers may be contributing.[26] While they do not guarantee the proper

operation of the system as a whole, they direct the developer to small portions of code where an error is occurring, and reduce the time required to debug following changes to a unit's implementation.

4.1.1 Octree Data Structure

In addition to the structure discussed above, connecting the physical components to higher level components to which they are related, an octree data structure is employed to relate the physical location of individual panels. The primary motivation for this structure is looking forward to future enhancement of CPanel with the implementation of a fast tree method. Before discussion of that opportunity, the basics of an octree and how it is coded in CPanel are covered.

Octrees, or more generally tree-based data structures, are a type of spatial data structure that stores data in a hierarchal manner, allowing for faster queries regarding spatial relationships. These data structures are used in a number of different fields, including computer graphics, image processing, and even database management. They have also proven very useful in various CFD applications, such as cartesian grid generation, multi-grid solution methods, and fast tree methods in modern panel codes. The formulation of the tree is based on the recursive division of a node into child nodes. This division results in each node containing four children in 2D, and eight children in 3D, hence the name, octree. To make the concept more clear, a quadtree is shown if Figure 4.3, with the left side showing the

actual decomposition of the domain, and the right side showing the tree structure that results from the decomposition.

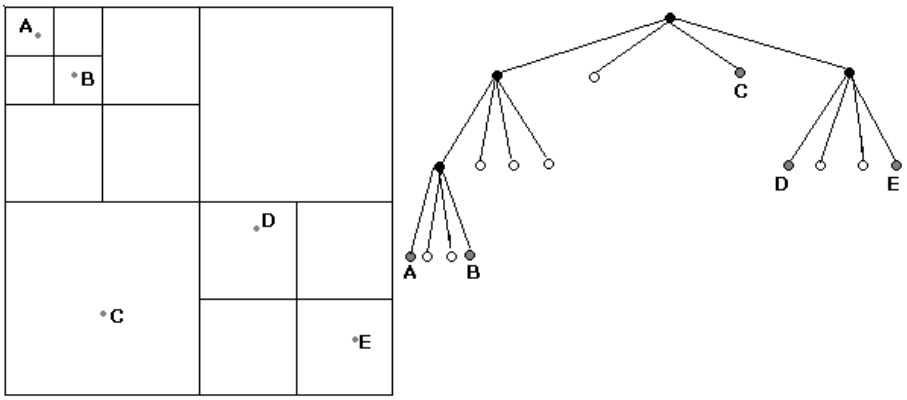


Figure 4.3: Generic Quadtree for Point Data[4]

Beginning with a bounding box containing all the data points, each node is divided into four if it contains more than one data point. This process is performed on each node until each child is a leaf node, meaning it has as many or fewer than the prescribed data points within its boundary. The process is exactly the same for an octree, simply with one added dimension. The process of building an octree, as it is done in CPanel, is shown in the sequence diagram in Figure 4.4. The implementation in CPanel is done in a templated base class, meaning that the octree can be used to store any object that the user specifies. The class cannot be used directly. A user specified method for determining the reference point of the object must be provided. In the case of panels, the centroid of the panel is used. This generalized implementation allows easy application of the octree structure to a vortex particle wake, as is used in FastAero.[27]

For readability, the creation of an octree of only three levels is shown in Figure 4.4, with the root node, one child, and a leaf node. In actuality, the number of children, or successive subdivisions of the domain, is dependent upon the size of the data and the maximum number of members allowed in a node.

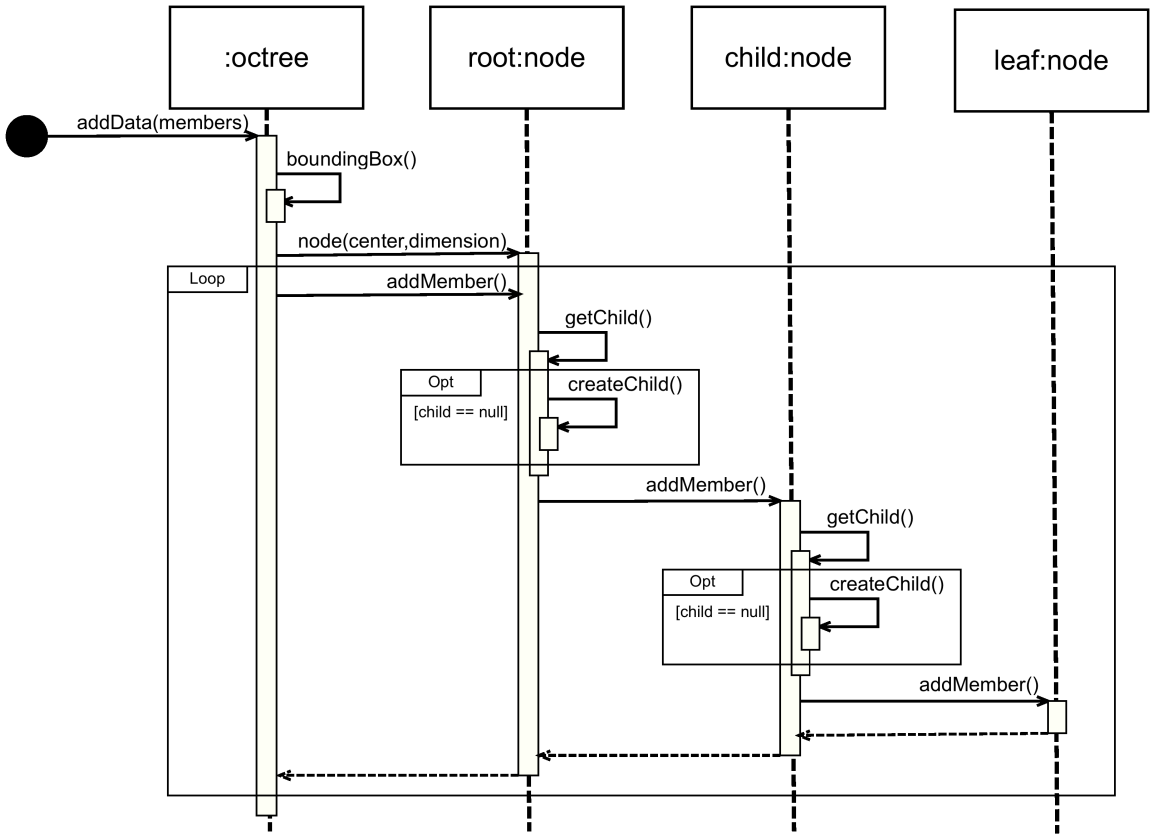
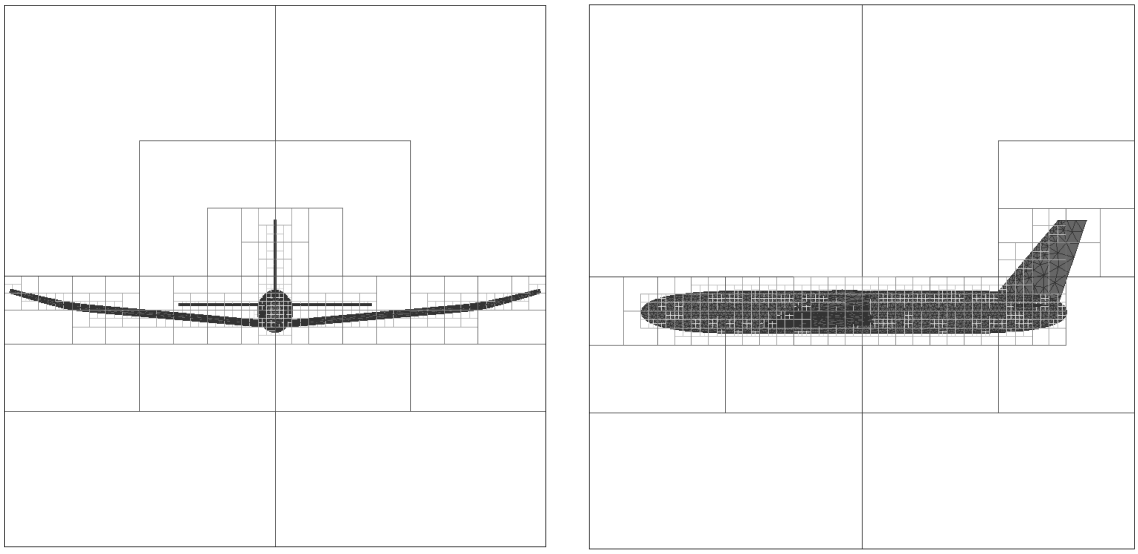


Figure 4.4: Sequence Diagram for Construction of Octree

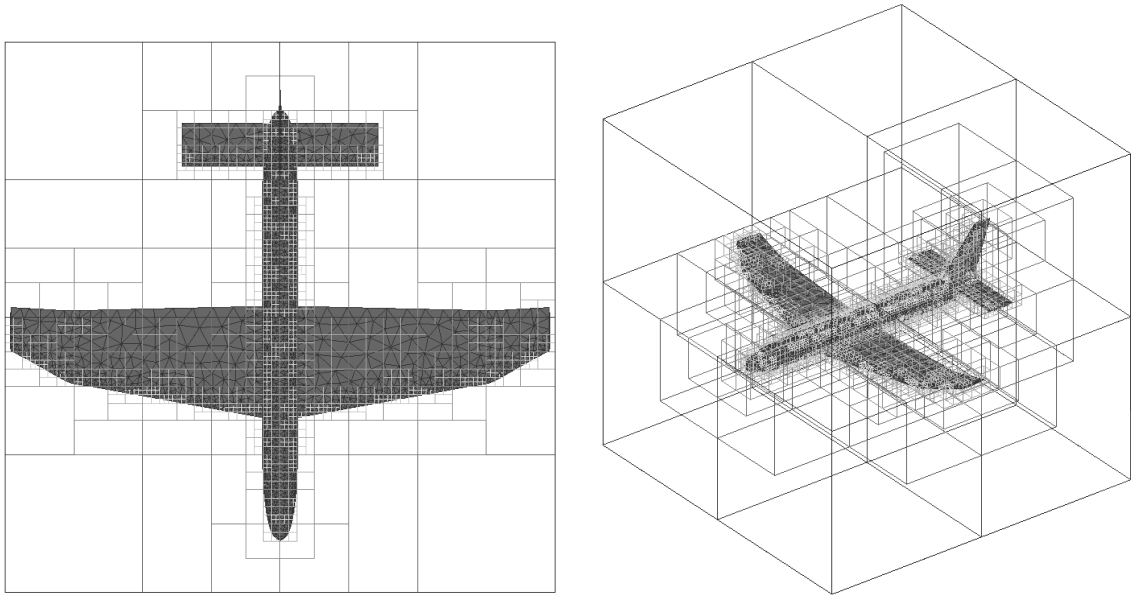
Figure 4.5 shows the octree generated around a generic aircraft geometry made up of just over 11,000 panels. For this case, the maximum panels per node is set to ten, and the result is an eight level octree.

In the early stages of development, prior to the implementation of a node-edge-panel structure, the octree data structure was used to find panel



(a) Front

(b) Side



(c) Top

(d) Isometric

Figure 4.5: Visual of Octree in CPanel

neighbors that would be used in the velocity calculation. Neighbors can be found by descending the tree to find the panel of interest, checking the other panels in the leaf node to see if they share an edge, and if necessary, checking the panels in the neighboring nodes as well. The method reduces the speed of the neighbor search from $O(N)$, for a linear search, to $O(3n + F)$, where N is the number of panels in the dataset, n is the levels of refinement in the octree, and F is the number of panels per node.[28] While the speed gained from the tree is significant, the adoption of the node-edge-panel structure offered significantly better performance.

Despite the modified neighbor searching algorithm, the octree still will prove useful in the future development of CPanel, specifically with the application of multipole methods and spatial queries outside of the neighbor search.

4.2 Kutta Condition Enforcement

After all edges are scanned to set each panel's neighbors, those edges that contain two surface panels and one wake panel can be flagged as trailing edges. From the midpoint of each trailing edge, a line of constant vorticity is created to be used in the calculation of each wake panel's strength. These lines are shown on a simple wing geometry in Figure 4.6. The strength of each wake line is that of the edge from which it is shed, corresponding to the difference in the strength of the upper and lower surface panels. This is consistent with Equation 2.34.

To enforce the Kutta condition, the influence of each wake panel must be

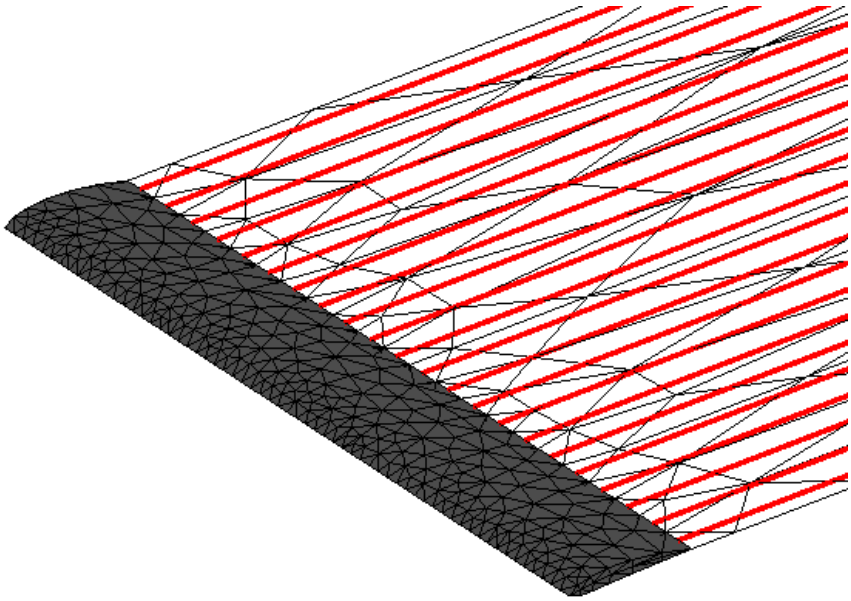


Figure 4.6: Wake Lines used in Kutta Condition Enforcement

incorporated into the influence coefficient matrix, **A**, of Equation 3.1. In order to avoid introducing additional unknowns to the system of equations, their strength must be represented as a function of the related surface panel strengths. Figure 4.7 defines the variables used in the following equations for the influence of the wake panel of interest. The wake strength for any panel can be calculated with a linear interpolation of the wake lines by which it's collocation point is bound

$$\mu_w = \mu_1 + (\mu_2 - \mu_1)\,\bar{Y} \tag{4.1}$$

where $\bar{Y}$ is the interpolation weight.

$$\bar{Y} = \frac{y_w - y_1}{y_2 - y_1} \tag{4.2}$$

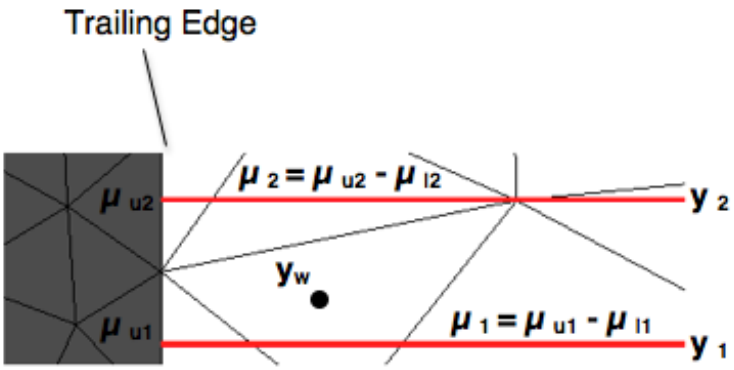


Figure 4.7: Variables used in Wake Strength Interpolation

The influence of the wake panel on the i^{th} surface panel is written as the product of the influence coefficient and the panel's strength.

$$\Phi_{iw} = C_{iw}\mu_w \tag{4.3}$$

Combining Equations 4.1 and 4.3 with the definitions of the wake line strengths shown in Figure 4.7, the influence of the wake panel on any panel, i, can be written as

$$\Phi_{iw} = C_{iw}\left[\left(1-\bar{Y}\right)\mu_{u1} + (\bar{Y}-1)\mu_{l1} + \bar{Y}\mu_{u2} - \bar{Y}\mu_{l2}\right] \tag{4.4}$$

The Kutta condition is satisfied by adding the coefficients to the column of A

that corresponds to the correct upper or lower surface panel. Once this is done, the linear system of equations is solved using a GMRES algorithm included in the Eigen library. The resulting doublet strengths at each panel give the perturbation potential across the surface and the post processing can begin.

4.3 Velocity Calculation

Arguably one of the more important aspects of a panel code is how the velocity on the surface is calculated once the singularity strengths are known. The importance stems from the rippling effect it has on the results, affecting the pressure distribution and therefore the integrated forces and moments. Despite the value it holds, it is an area that is not often covered in great detail in the program's documentation. Prior to the growth of automatic mesh generation and unstructured potential flow programs, the method was consistent, as the structured discretization lent itself to finite differences to approximate the gradient of the potential. Unstructured meshes drive the developer to explore different methods. As can be seen from Table 4.1, documentation of these methods is often nonexistent, or not very specific, but it is assumed that generally some type of viscous core method is utilized.

The list in Table 4.1 is by no means exhaustive, but it highlights some of the documented panel codes from the last few decades. A couple of the programs, such as VSAERO and APAME, were originally developed for a structured discretization, and therefore their outdated documentation cites a finite difference approach for the

Table 4.1: Surface Velocity Formulation in Existing Panel Codes

Program	Year	Discretization	Velocity Formulation
VSAERO[16]	1987	Structured	Finite Difference
PAN AIR[29]	1990	Structured (Subpanels)	Spline
PMARC[3]	1992	Structured	Finite Difference and Viscous Core
FPA[30]	1996	Unstructured	Viscous Core
FastAero[31]	2000	Unstructured	Higher Order Singularities
DWFS[32]	2003	Unstructured	Not Documented
CBAERO[13]	2004	Unstructured	Not Documented
APAME[33]	2008	Unstructured	Not Documented

velocity formulation. More recent versions of these codes demonstrate compatibility with unstructured meshes but new documentation is not available. PMARC employs a finite difference method on the physical boundaries, and a fixed viscous core method in the time stepping wake and off body streamlines. The core radius is set by the user, adding a potential additional source of error. The nonphysical velocity created, even with the viscous core, is made clear by an additional routine included in PMARC. When calculating off body streamlines, an algorithm to check for the streamline crossing through a physical boundary is needed when it approaches the surface.[3] FastAero uses higher order singularity distributions across the panels, allowing surface derivatives to be calculated directly, at the cost of increasing the number of equations being solved in the linear system.[31] CBAERO has demonstrated an ability to produce satisfactory results with constant strength singularities, although the method used is not documented and results show nonphysical dampening of the vortex influences. These results will be discussed in the following chapter.

CPanel differs greatly from other recent panel codes in its unique approach to the velocity formulation. The work of McDonald and Ramos offers a modified Taylor series approach, allowing the velocity to be calculated from the knowledge of the velocity potential at the collocation points, and the zero normal flow condition imposed at the boundary. The method incorporates knowledge of directional derivatives to create a Hermite Taylor Least Squares (HTLS) problem, and the exact specification of the directional derivative at the point of interest in a Constrained Hermite Taylor Least Squares (CHTLS) problem.[22] The Taylor Least Squares (TLS) method has already been used in CFD applications, primarily in volume based methods using unstructured meshes. However, with only data on the surface, the TLS problem is ill-posed. The inclusion of directional derivatives, which are readily available in a boundary element problem, solves this problem, providing an opportunity for unstructured panel codes that has yet to be employed.

4.3.1 CPanel Implementation of CHTLS Method

The inclusion of the CHTLS method into CPanel required special treatment for some special cases that will be discussed. Results of the method, and a method of increasing the accuracy of the approximation, will be discussed in the following chapter. Details of the method itself can be found in the work of McDonald and Ramos.[22]

The implementation of the method itself remained consistent with the

documentation, allowing for the approximation of derivatives of any order, r, of a function of any number of variables, N. The number of observations, t, required for the problem to be well posed is calculated as follows.

$$t = \frac{(r+N)!}{N!r!} - 1 \tag{4.5}$$

For a three dimensional problem ($N = 3$) carried out to a third order Taylor series ($r = 3$), as is typically the case in CPanel, this results in nineteen observations required. In the hermite type problem, a panel provides two observations, specifying both the function value, as well as the directional derivative at the collocation point. Special treatment is required, however, in areas near a discontinuity, or areas with a large number of coplanar panels.

If two collocation points align with each other through the point of interest, they do not provide a unique function observation. Figure 4.8 illustrates this scenario in a two dimensional sense. The two aligned points do not provide unique information for computing the derivative in the x direction, and therefore only provide one observation towards the required number, t, in the TLS problem. The result is a singular matrix that can be fixed by including more observations, or reduction in the order or dimensionality of the problem. The most likely scenario in a panel code in which this will arise is in a flat portion of a surface, where supporting panels are coplanar with the panel of interest. CPanel will first try to avoid the issue by including additional observations above the required amount. In

addition to increased robustness, this inclusion allows the least squares method to smooth possible errors resulting from the iterative solution to the linear system of equations.[22]

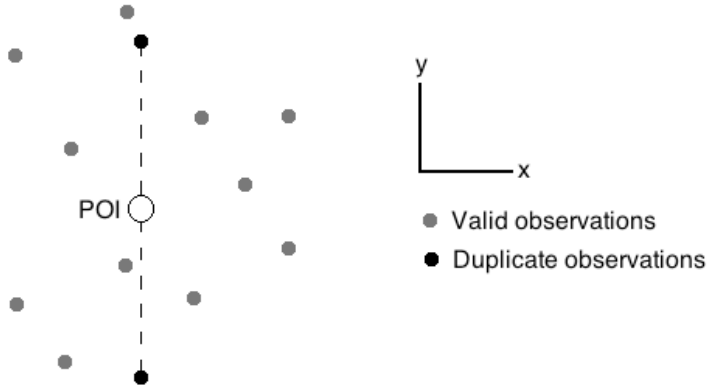


Figure 4.8: Duplicate Observations in TLS Derivative Approximation

If a singular matrix still arises, a two dimensional approximation of the derivatives is performed in the panel of interest's local reference frame. If the matrix remains singular, the velocity is approximated according to Equation 4.6, presented by Kinney.[13]

$$\mathbf{V} = (\hat{n} \times \mathbf{V}_\infty) \times \hat{n} \tag{4.6}$$

In regions near the wake, special care is taken in gathering the supporting data, as data on opposing sides of the wake will cause nonphysical velocities normal to the wake surface. This comes from the instantaneous jump in potential across the wake surface, as shown in Figure 4.9. One option to deal with the discontinuity, utilized by Ramos, is to use an adjusted potential value that accounts for the wake strength

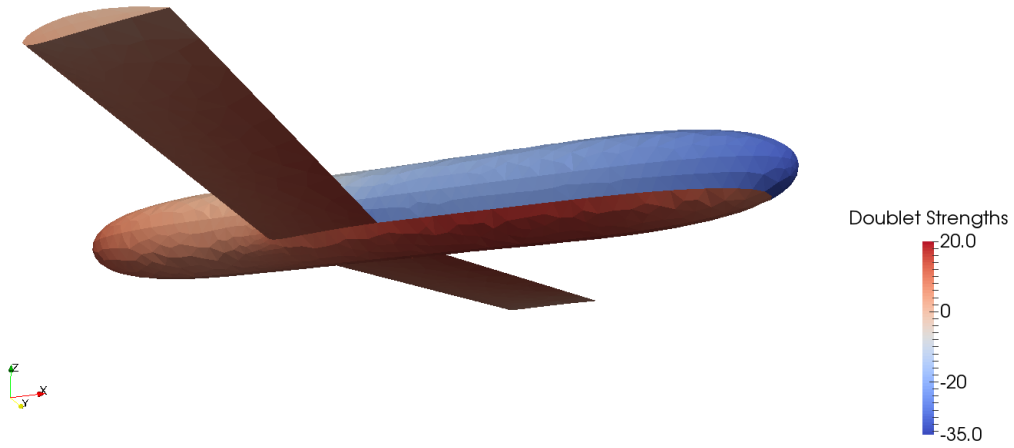
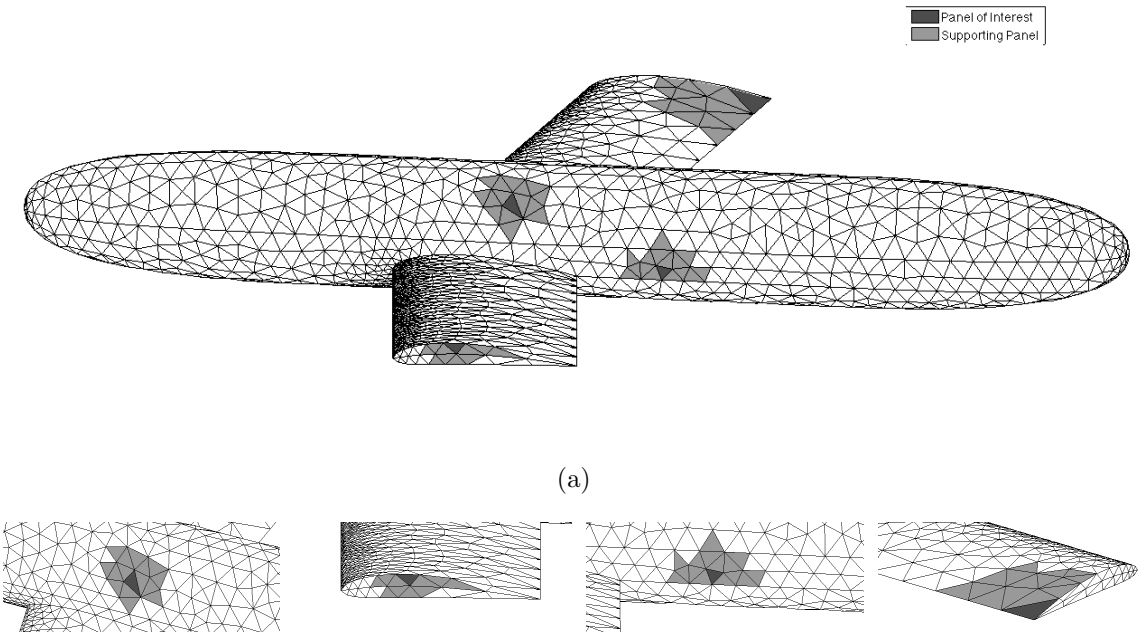


Figure 4.9: Perturbation Potential along Wake-Body Intersection

at that spanwise location.[1] CPanel addresses the issue simply by rejecting panels on the other side of the discontinuity. These cases arise near the trailing edge of the lifting surface, or on a downstream body intersecting the wake, such as the fuselage. Both of these are illustrated in Figure 4.10.

Despite being upstream of the discontinuous wake, velocity calculations on the wing tip patches were inconsistent when using supporting data from either the upper or lower surfaces, especially in aft panels near the trailing edge. CPanel therefore uses a flag to mark panels located on a tip patch, and performs a two dimensional CHTLS in the local reference frame with only supporting panels that also reside on the patch.



(a)

(b) No Constraints (c) Wing Tip (d) Near Wake (e) Trailing Edge

Figure 4.10: Constraints on Supporting Data in CHTLS Velocity Formulation

4.4 Streamlines

It is often helpful in the design process to visualize the flow field to see exactly what is happening in a region of interest. This is often done through the use of pathlines, streaklines, or streamlines, depending on the desired flow phenomenon. For a steady flow, all three of these lines coincide with each other. As CPanel is currently limited to steady cases, differentiation between these lines is not required and they will be referred to as streamlines in this document. Should CPanel be expanded to handle unsteady simulations, differentiation will be required and corresponding algorithms should be developed.

In addition to their utility in flow visualization, streamlines can be used in a panel code to provide a viscous approximation in an otherwise inviscid flow field. The algorithm presented lays the groundwork for further enhancement of CPanel to include viscous forces. In order to do so, a seeding algorithm will need to be developed that generates even coverage of the surface, and an integral boundary layer method can be applied along each of the streamlines.

Off body streamlines are also presented, although they encounter issues when they approach closely to a surface. Due to the singularity created by the Biot-Savart law in the velocity calculation, the flow field is not divergence free and it is possible for streamlines to cross a solid surface. This is an issue that must be addressed when a vortex particle wake is implemented, and therefore was deemed outside the scope of this work. The following section will cover the basic algorithm

used in generating streamlines. Examples of the aforementioned improvement opportunities will also be pointed out.

4.4.1 On-Body Streamlines

A streamline is created by inserting a massless particle at a specified location, and tracking the position as it convects through the flow field. Mathematically, the streamline can be found by integrating the following differential equation.

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p\left(\mathbf{x}_p, t\right) \tag{4.7}$$

The decisions the developer must make is where to start the streamlines, how to perform the integration, and what criteria dictates the end of streamline. Each of these three decisions will be discussed for the on-body case.

CPanel begins streamline tracing by locating all of the rear stagnation points on the geometry. The integration is then done in reverse until a forward stagnation point is reaches, following the method of Kinney.[13] This is done separately for each surface in the geometry, with lifting surfaces being treated differently from non lifting surfaces.

On lifting surfaces, the Kutta condition guarantees that the stagnation point is at the trailing edge. Therefore, CPanel starts a streamline on both the upper and lower surfaces at the midpoint of a sharp trailing edge. On a non lifting surface, the streamlines will converge at a single rear stagnation point. According the the

definition of the velocity potential, a stagnation point will occur where the gradient of the potential is zero, meaning there is a local maximum (rear stagnation point) or minimum (forward stagnation point) of the potential. However, due to the error that arises from the discretization, the collocation point of the panel with a maximum velocity potential is likely close to, but not the actual stagnation point. For this reason, CPanel starts a streamline at this collocation point and propagates forward until the stagnation point is reached. A cloud of points is then generated around the stagnation point, and projected on to the neighboring panels, creating the starting points for that surface.

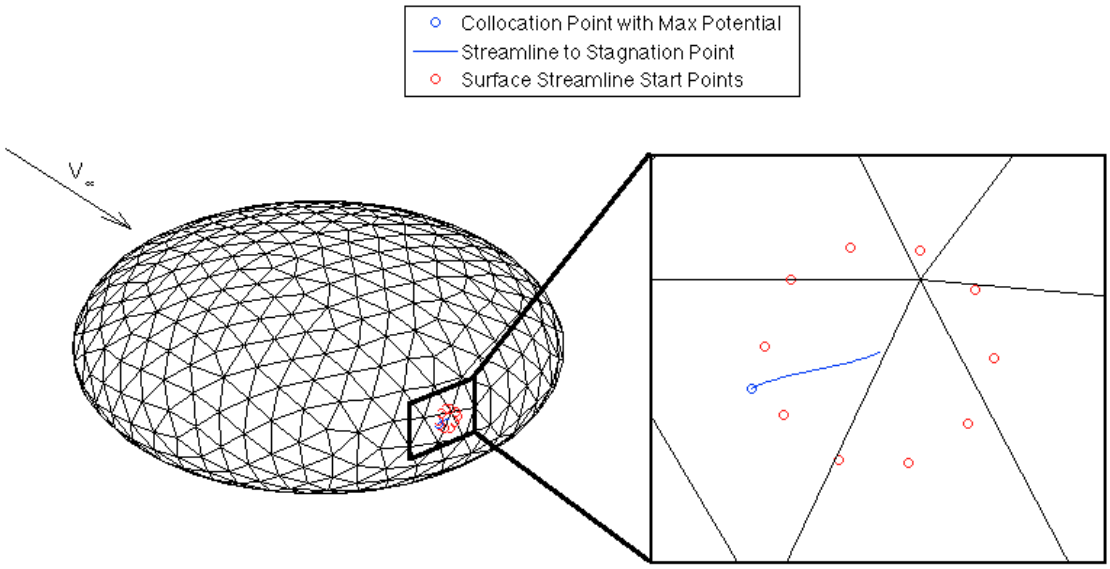


Figure 4.11: Streamline Starting Points on Surface without Sharp Trailing Edge

Once the starting points are found, the Euler method is used to propagate the

streamline backward until a forward stagnation point is reached. The logic behind the tracing of an on body streamline is shown in Figure 4.12.

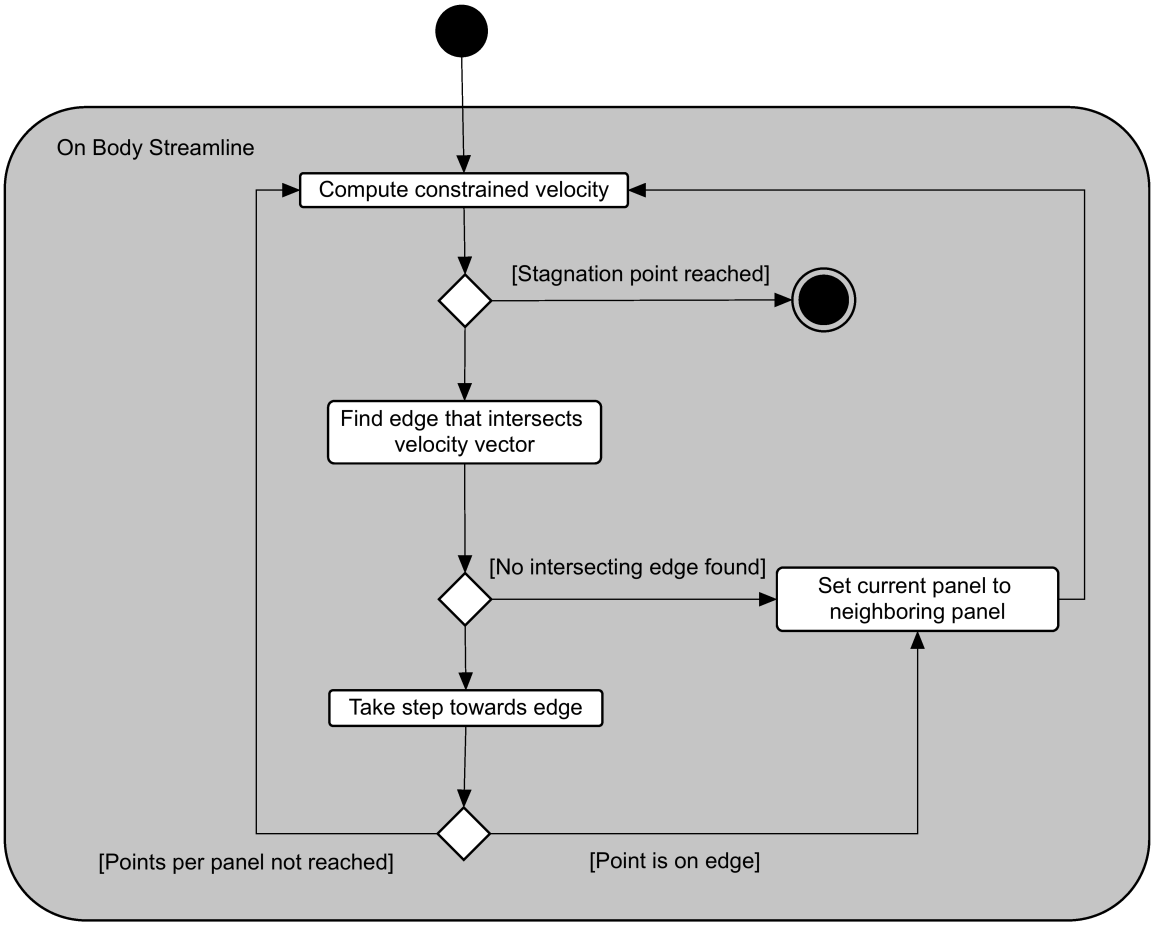


Figure 4.12: Activity Diagram for Creation of On Body Streamlines

The step size is set by a number of points per panel and the distance from the point to the next edge. The step vector is calculate with Equation 4.8

$$\mathbf{h} = \frac{\mathbf{d}}{N - i} \tag{4.8}$$

where $\mathbf{d}$ is the vector from the current point to the point where the velocity vector

and edge intersect, N is the points per panel, and i is the number of points already on the current panel.

With this methodology, as opposed to a fixed time step, keeping track of which panel the streamline is currently crossing is simple, allowing the use of the CHTLS method to calculate the velocity constrained to the surface. Additionally, assuming that a properly constructed mesh is more resolved in areas of large gradients of velocity, the streamlines will also be more resolved in those areas, providing smoother results.

The branch shown in Figure 4.12 that addresses the case when no intersecting edge can be found is in place to deal with premature termination of the streamline when the streamline is close to tangent to an edge. In this case, it is possible for a streamline to cross a panel to an edge, and then remain on that panel and traverse to another edge.

The algorithm currently in place is robust in the creation of individual streamlines. Due to the physics of the problem, however, streamlines will converge on each other in certain regions of the flow, leaving some parts of the surface bare, and others very densely covered. This does not affect the users ability to visualize the flow, but would produce poor results if the streamlines were used to calculate viscous forces. Prior to this capability being added to CPanel, the seeding and termination criteria need to be adapted to account for the proximity of nearby streamlines, as is done by Kinney.[34]

4.4.2 Off Body Streamlines

For off body analysis, a higher order Runge-Kutta method provides an advantage over lower order methods in that it will smooth regions with large gradients in the velocity. Additionally, methods such as the Runge-Kutta-Fehlberg (RKF) utilizes an adaptive step size to further smooth the integration. This smoothing becomes most important when an off body streamline is approaching a surface, limiting the chance that too large of a time step is taken and the boundary condition of zero normal flow on the surface is violated. In the interest of computation time, the velocity influence formulation is used for points not located on the body. If the TLS method were to be used, the potential at an entire cloud of points would need to be calculated in order to find the velocity at each point on the streamline. The resulting streamlines using the velocity influence formulation can be seen in Figure 4.13.

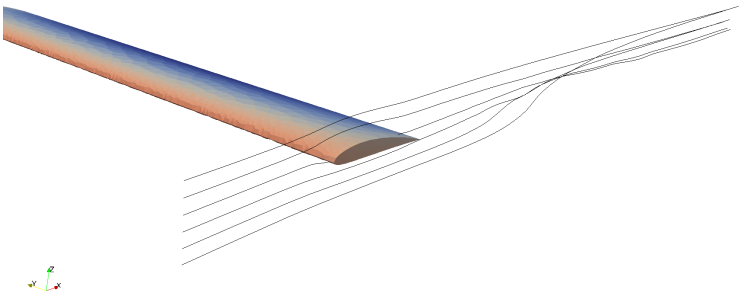


Figure 4.13: Off Body Streamlines over NACA 4412

As discussed previously, a divergence free velocity field is difficult to guarantee with vortex core models. The effect of the non physical velocity field is easily visualized by looking closer at the streamlines. Figure 4.14 shows the streamline

entering the body near the trailing edge on the lower surface. The solution to this problem will be similar to that used in the vortex particle wake implementation, as the guarantee of a divergence free field is the biggest challenge in using vortex particles.[18] It was therefore deemed outside the scope of this work, but the framework for the streamline creation is in place.

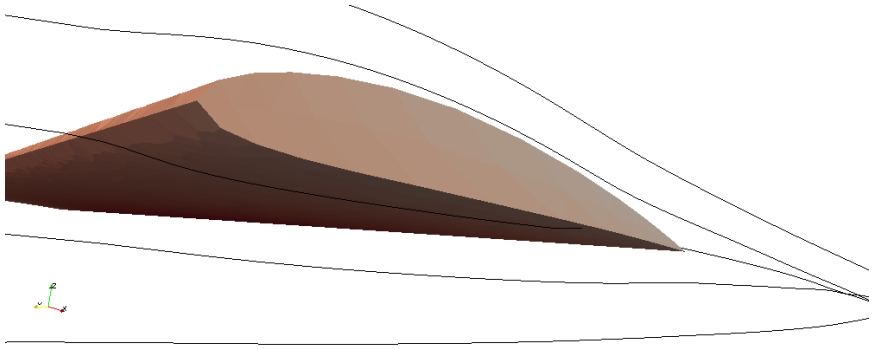


Figure 4.14: Streamline Penetration of Solid Surface

One suggested strategy, outside of a more complex viscous core model, would be to use the velocity influence method when the particle is a certain distance from the surface, and switch to a HTLS method when the particle is inside of that distance. The existing octree data structure can be used for those spatial queries as well.

4.5 Stability Derivatives

If the option is specified in the input file, CPanel will calculate the static stability derivatives for each case. This involves running two perturbed cases, one for angle of attack and one for angle of sideslip. Once the cases are run, the calculation of the derivatives is approximated with a simple first order difference. For example, the

derivative of the pitching moment with respect to angle of attack would be calculated as follows,

$$\frac{dC_m}{d\alpha} = C_{m_\alpha} = \frac{C_m^* - C_m}{\alpha^* - \alpha} \tag{4.9}$$

where C_m^* and α^* represent the perturbed value. CPanel perturbs each angle by half of a degree so as to avoid excessive influence of numerical error that could arise if a very small angle were used.

Dynamic stability derivatives are not currently included in CPanel. Due to the increase in cases that must be run for each stability derivative, addition of this feature would be most beneficial once the code is sped up with a fast tree method. The process by which those derivatives are calculate, however, is a modification of source strengths at each panel to simulate a free stream flow representative of the standard free stream velocity combined with a rolling motion about the prescribed axis. Once the source strengths are modified, the solution progresses in the same fashion as a standard case.

Chapter 5

Results and Verification

The software development process involves two processes that are intended to ensure the software meets requirements and the products satisfy their intended needs. The processes, verification and validation, typically are performed together and are more generally referred to V and V. Verification provides evidence for whether the software meets objectives as they relate to correctness and completeness. Validation provides evidence the whether the software is solving the right problem and correctly modeling physical laws.[35]

In the case of a panel code, their long history and use in the aerospace industry has taken them through extensive validation, and their strengths, as well as constraints, are well understood. Any new software, though, even if it is meeting a need that other programs have met, must undergo extensive verification to ensure that the results are reliable. For this reason, the results presented will be in the context of verification, with comparisons to existing potential flow programs that are widely accepted in the industry. The validation portion of the V and V,

consisting of comparisons to experimental flow data, can be found in a number of other reports, primarily from earlier panel codes' documentation.

This chapter will first introduce the tools used in the verification process, followed by results and the verification itself. The chronology will follow the natural progression of the program, beginning with non lifting flows, then lifting flows on a simple wing configuration, and finally geometries more representative of a complete aircraft. In addition to the verification, an interesting result of the CHTLS method is presented with recommendations for discretization software that can increase the accuracy of the surface gradients.

5.1 Verification Tools

The existence of a potential flow allows for the application of a number of different solution methods, depending on the geometry. A number of these solution methods are used in verifying the results provided by CPanel, each providing insights to the accuracy of a specific aspect of the software. These comparisons allowed for verification of the general solution process, as well as verification of implementation methods unique to CPanel, such as the Kutta condition enforcement and velocity formulation. Each solution method or program is listed below with a brief description.

5.1.1 Analytical Solution

While the need for panel codes and numerical techniques stems from the lack of an analytical solution for complex, arbitrary configurations, exact solutions exist for specific geometries. In three dimensions, the solution exists for an ellipsoid submersed in a potential flow. The work of Munk provides the solution in terms of the semi-principal axes of an ellipsoid at any orientation to the free stream flow.[36]

The solution allows for verification of the panel code solution via comparison of the velocity potential at the collocation points on a discretized ellipsoid. Additionally, as the solution for the potential exists in an analytical form, the analytical derivative provides the exact velocity at a given point. This provides the best way of verifying the CHTLS velocity formulation and investigating the sources of error that exist.

5.1.2 VLM Methods

Two different Vortex Lattice Methods are used in the verification process. For simple planforms, Athena Vortex Lattice is used. Developed by Mark Drela at MIT, AVL is used heavily in the design of planforms, especially when they are rapidly changing. The vortex lattice method provides good approximation for the characteristics of lifting surfaces. The main benefit is the very fast solution time, allowing for large design spaces to be explored, whether in a trade study or optimization problem. The deficiencies in the method lie in inability to capture the

thickness portion of the problem, which will have minimal effect on the lift and drag coefficients, but a significant effect on the pitching moment coefficient. In this verification process, AVL is used to ensure that the Kutta condition enforcement method is functioning properly in a simple lifting problem.

Another VLM became available more recently and is included in the OpenVSP distribution. Developed by Dave Kinney at NASA, VSPAERO is built off the same principles as AVL, but benefits from the use of a direct adaptation of the VSP model. Using the degenerate geometry, VSP creates a model of zero thickness surfaces representative of the actual model that can be used in a VLM. This allowed the SR22 configuration to be run through a VLM to compare to the panel code results from both CPanel and CBAERO.

5.1.3 CBAERO

CBAERO is an unstructured panel code also developed by Dave Kinney for NASA. The program has extensive capabilities in both subsonic and supersonic flows, providing viscous approximations, surface heating, and fluid-structural interaction as well. For the subsonic solver, the solution method utilizes a fast tree method based on first order singularities on flat panels. The singularity and discretization order are exactly the same as CPanel, providing nearly a one to one comparison. Differences arise in the tree method and the velocity formulation. Dave Kinney states that the velocity is calculated using analytical influence functions, and a hard

cutoff is implemented to avoid issues with the singularity. Additionally, empirical correlations between mach number and pressure coefficient are used to trim extreme pressure coefficient values to what is considered realistic (Dave Kinney, personal communication, August 21, 2015). As the approach of CBAERO is the closest of these methods to that of CPanel, it provides a means of comparing results on general configurations. Differences in the results are rather significant on the SR22 configuration and possible reasons for the discrepancy will be addressed.

5.2 Non Lifting Flow

Munk's solution for the velocity potential around an ellipsoid provides the first opportunity in the development of a panel code to check the solution process as a whole.

The velocity potential, valid for any point on the surface of an ellipsoid is given by

$$\Phi(x,y,z) = |\mathbf{V}| \left[\left(\frac{\alpha}{2-\alpha} + 1 \right) x + \left(\frac{\beta}{2-\beta} + 1 \right) y + \left(\frac{\gamma}{2-\gamma} + 1 \right) z \right] \tag{5.1}$$

where

$$\alpha = abc \int_0^\infty \frac{dt}{(a^2+t)\sqrt{(a^2+t)+(b^2+t)+(c^2+t)}}$$

$$\beta = abc \int_0^\infty \frac{dt}{(b^2+t)\sqrt{(a^2+t)+(b^2+t)+(c^2+t)}}$$

$$\gamma = abc \int_0^\infty \frac{dt}{(c^2+t)\sqrt{(a^2+t)+(b^2+t)+(c^2+t)}}$$

and a, b, and c represent the principle axis of the ellipsoid.

For the discretization, a MATLAB program called DistMesh is used due to VSP's inability to generate a perfect ellipsoid. DistMesh is developed by two faculty members at UC Berkeley and uses a signed distance function for the initial node generation, making it well suited for simple geometric shapes. For an ellipsoid, the signed distance function is

$$d(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 \tag{5.2}$$

The dimensions used in this validation are shown in Figure 5.1. The ellipsoid is set at an angle of attack and sideslip of 15° and 10°, respectively, so as to avoid an axis aligned velocity. A nominal mesh is shown with just over 5,000 panels. To verify the order of accuracy of the CPanel solution, a set of meshes ranging from 2,000 to 11,000 panels were used.

The resulting velocity potential on the surface is shown in Figure 5.2. The solution shown is computed on a surface made up of approximately 11,000 panels.

Due to the first order discretization and singularity strengths used in CPanel, the error, when plotted on a log scale, should show first order behavior on

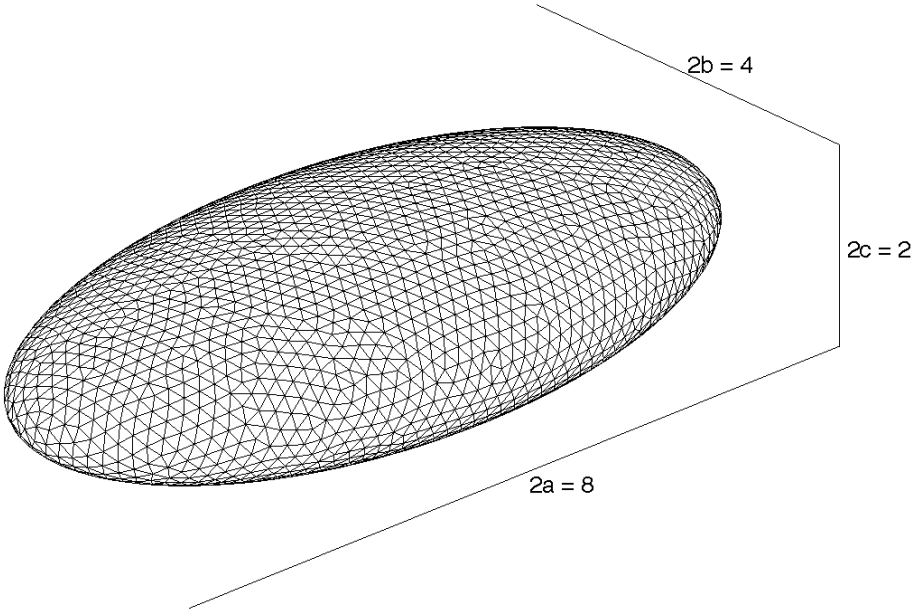
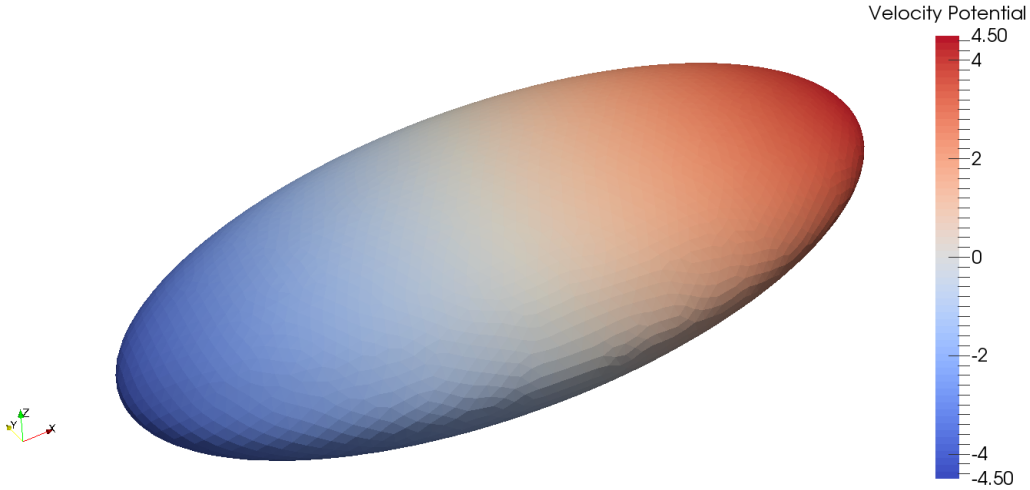


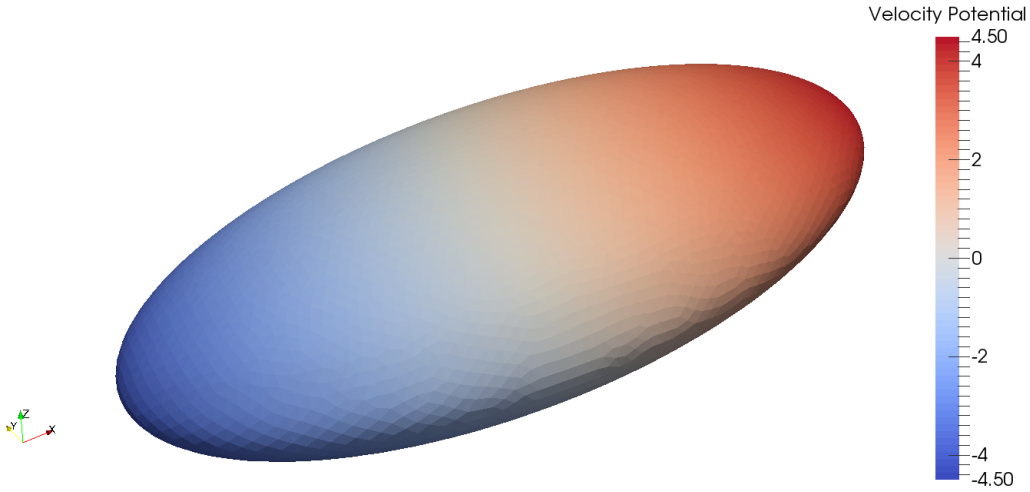
Figure 5.1: Ellipsoid Geometry and Dimensions

successively refined meshes, as is seen in Figure 5.3.

As mentioned previously, an exact solution for the potential also provides an exact solution for the velocity, and therefore the pressure coefficient. This provides an opportunity to look at the error in the CHTLS velocity formulation and investigate methods to mitigate that error. The most noticeable trend in the error is that the regions of high curvature show much more error in the pressure coefficient than the areas that are relatively flat. This can be attributed to the error between the normal vector based on the panel's geometry and the normal vector based on the underlying surface. The normal vector at any point on an ellipsoid is fairly straight forward to compute. Taking the gradient of the equation of an ellipsoid and



(a) CPanel Solution



(b) Exact Solution

Figure 5.2: Velocity Potential on Ellipsoid Surface ($\alpha = 15°, \beta = 10°$)

normalizing it yields the normal vector as a function of position.

$$\hat{n} = \frac{\boldsymbol{\nabla} F}{|\boldsymbol{\nabla} F|} = \frac{\left[\frac{2x}{a^2}, \frac{2y}{b^2}, \frac{2z}{c^2}\right]}{\sqrt{\left(\frac{2x}{a^2}\right)^2 + \left(\frac{2y}{b^2}\right)^2 + \left(\frac{2z}{c^2}\right)^2}} \tag{5.3}$$

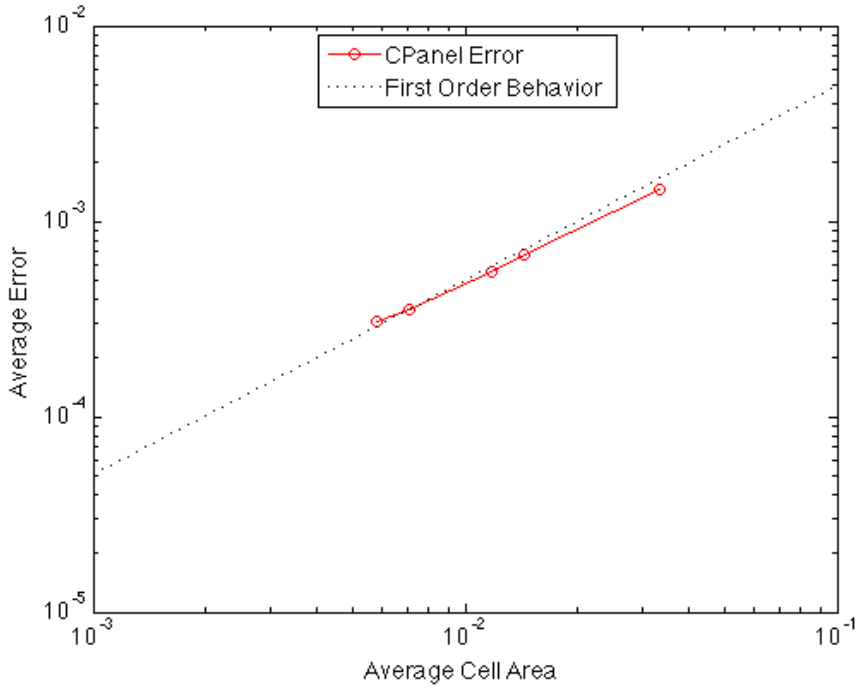
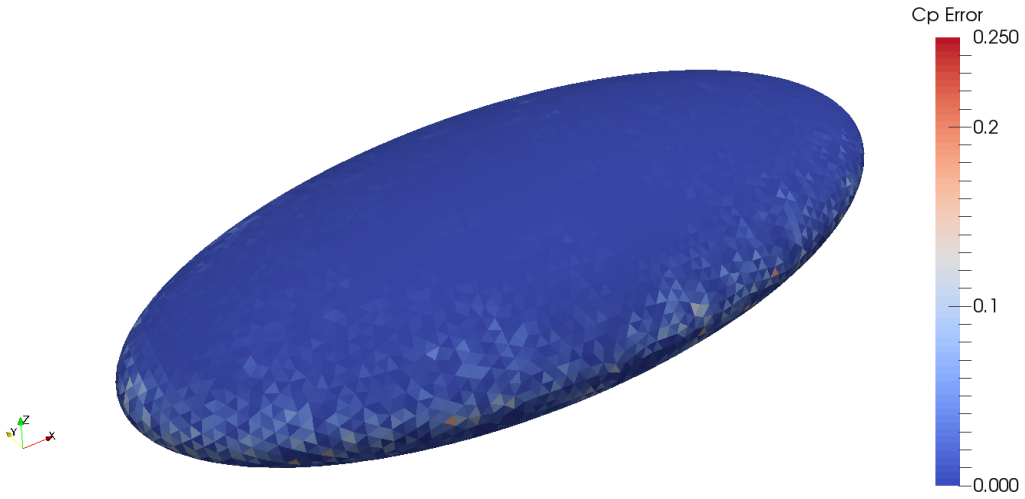


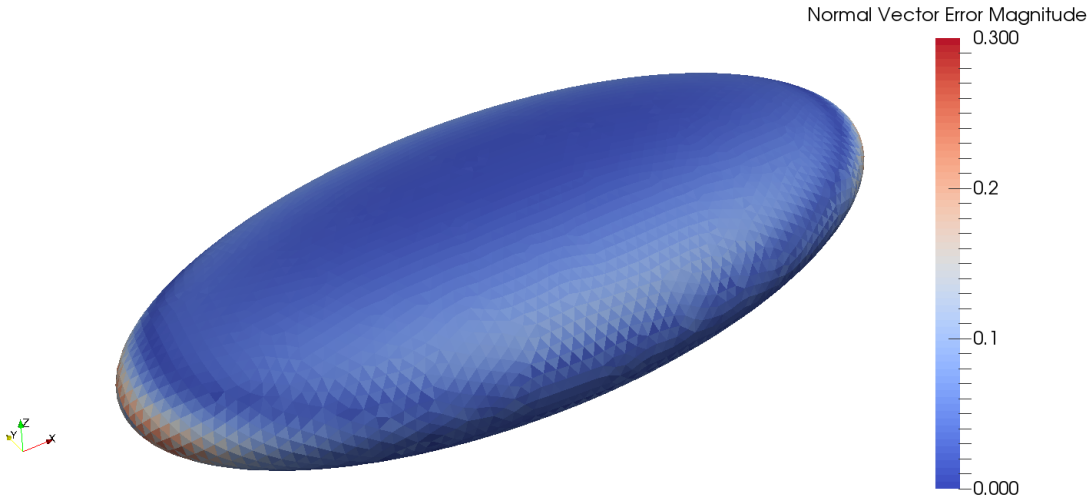
Figure 5.3: Average Error in Ellipsoid Solution on Successively Refined Discretizations

The magnitude of difference between the vector normal to the panel itself and the normal vector calculating using Equation 5.3 at the panel centroid is shown in Figure 5.2. The correlation is clear, with both the error in the pressure coefficient and the error in the normal vectors being largest along the highly curved portions of the ellipsoid.

To mitigate this source of error, CPanel was modified to accept a modified geometry file that includes normals calculated based on the bezier surface underlying the discretization. The file appends the normal vectors to the standard Cart3d format following the ID tags of the panels. Both the standard Cart3d (.tri) and the modified (.tricp) formats can be seen in Appendix A. While the use of the



(a) Error in Pressure Coefficient



(b) Magnitude of Error in Normal Vector

Figure 5.4: Correlation of Pressure Coefficient Error with Error in Normal Vector

surface based normal vectors reduces the error that arises in the CHTLS method, they cannot be used to reduce the error in the velocity potential. In fact, use of them in constructing the linear system of equation will result in a system that cannot be solved. This is due to the fact that the influence coefficients are a

function of the edge geometry on each panel. The normal vector used in setting the boundary condition must be normal to the surface made by those edges, and therefore cannot be the normal vector based on the bezier surface. The CHTLS method, however, is purely based on scattered sampling of the velocity potential at points on the surface, completely independent of the edges of each panel.

The benefit provided by the use of normals from the bezier surface is very significant, as can be seen in Figure 5.5. The error on a coarse discretization of 2,000 panels can be reduced to less than the error on a fine discretization of 11,000 panels just by using the surface based normal vectors. An interesting result of the CHTLS velocity formulation is seen in comparing the 3rd order and 2nd order formulations. In estimating first derivatives of the velocity potential, a 2nd order CHTLS should show first order accuracy, as is seen in Figure 5.5. The 3rd order CHTLS shows less than first order behavior, and an explanation for this could not be found. Because the CHTLS is based on velocity potential values with first order accuracy, an increase in the order of CHTLS should only be as accurate as the data it is based on. Therefore, it would be expected that the 3rd order CHTLS would show first order accuracy as well. Figure 5.5 shows that the 3rd order CHTLS appears to be converging to a non zero error, indicating an error. In an effort to find this bug, the output at each step from the CPanel implementation of the CHTLS method was compared to the code generated by McDonald in MATLAB. Quantities between the two implementations showed agreement, leaving the explanation for the

behavior requiring further investigation. Based on these results however, a 2nd order CHTLS is preferred. An added benefit of this finding is the requirement for fewer supporting panels to be found for use in the velocity calculation.

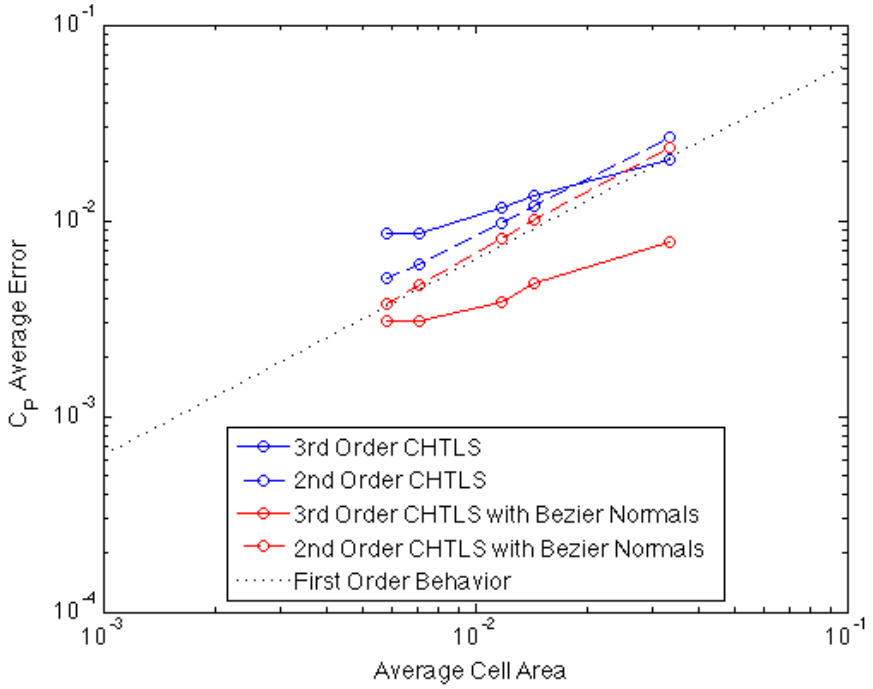


Figure 5.5: Normal Vector's Effect on Error in Pressure Coefficient

In order to take advantage of the increased accuracy when using Bezier based normal vectors, the software used for mesh generation must be capable of writing the normals from the underlying bezier surfaces to a file. The open nature of VSP makes this possible with a small amount of knowledge of the code. Bezier surfaces are navigated using u and w coordinates, similar to the way you can navigate along a line using a parameter, t. Therefore, the position on the surface in u and w coordinates that is closest to the center of the triangle, in x,y, and z coordinates,

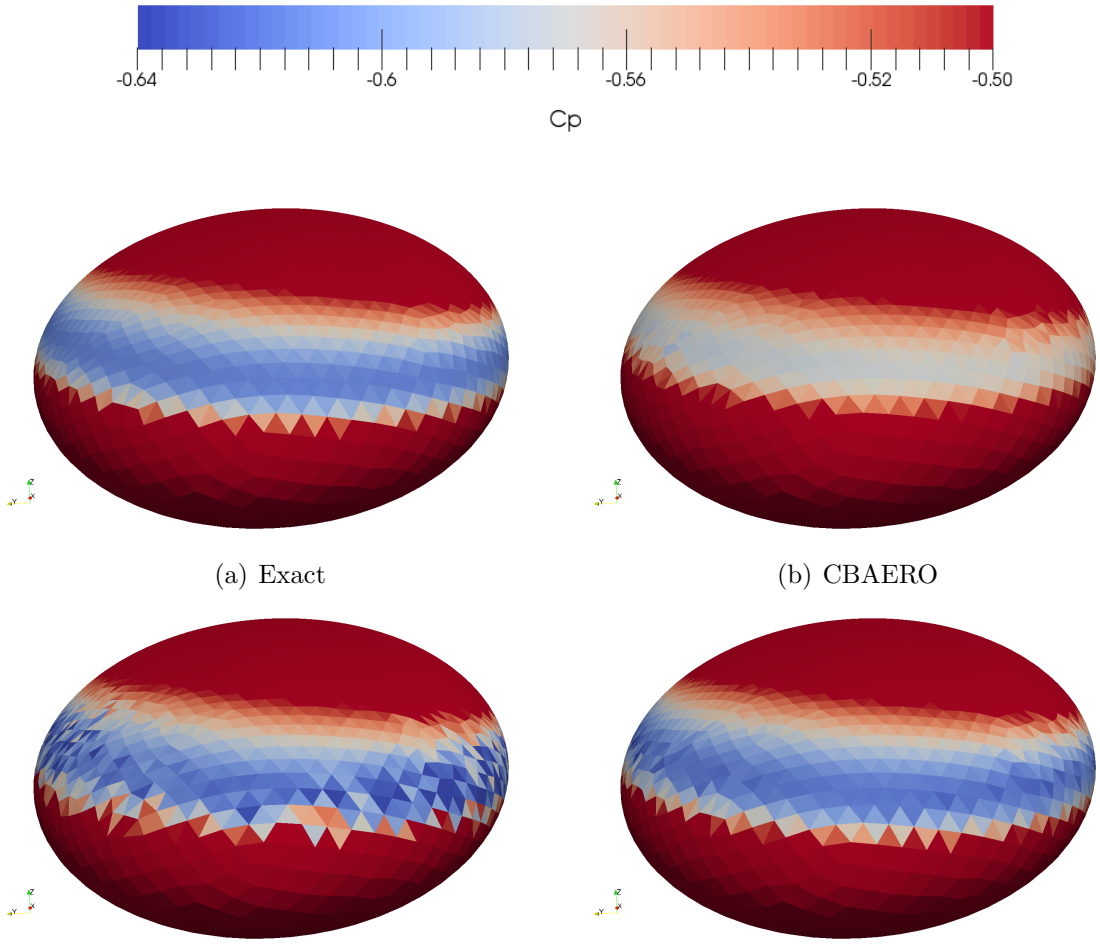
must be located. The bezier normal vector can then be calculated at that point. The center of a triangle is not uniquely defined, and it should be noted that the centroid,

$$\mathbf{P}_{centroid} = \frac{\mathbf{P}_1 + \mathbf{P}_2 + \mathbf{P}_3}{3} \tag{5.4}$$

is used in CPanel, where $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are the node locations. Once this is calculated for each panel, VSP already has routines to perform the remaining steps. For any user working with VSP and CPanel in conjunction, this is a recommended modification, and a possible modification for future releases of VSP itself.

In order to be consistent in the rest of the verification process, CBAERO was also run for the ellipsoid geometry to anchor the results against a known analytical solution. The surface pressures revealed that CBAERO predicts a weaker suction peak than the exact solution and the solution from CPanel. This is shown in Figure 5.6.

The exact solution predicts a minimum pressure coefficient of -0.62 so the limits were set to -0.5 and -0.65 to show more contrast. The weaker suction predicted by CBAERO could be attributed either to the cutoff method used in the velocity calculation, or due to the trimming that is done based on empirical correlations. The implications of this inaccuracy become more pronounced with the other geometries used in verification. Results show significant support for the CHTLS method when compared to other methods.



(a) Exact

(b) CBAERO

(c) CPanel

(d) CPanel with Bezier Normals

Figure 5.6: Variation in Suction Peak Pressure Coefficient Among Various Solution Methods

5.3 Lifting Flow

In order to validate the enforcement of the Kutta condition, a simple lifting geometry is used. This allows verification using three sources sources, CBAERO, AVL, and lifting line theory. The results also provide additional support for the CHTLS velocity formulation.

Differences between panel codes and VLMs in the predicted lift and induced drag coefficients are small, while differences in the pitching moment coefficient are more noticeable. This should be expected, and is explained with an analogy to a 2D case of the lifting problem.

The superposition principle can be used to show that this problem can be solved by solving three simpler problems: a thick airfoil at zero angle of attack, a flat plate at an angle of attack, and a cambered airfoil with zero thickness at zero angle of attack. This is illustrated in Figure 5.7. The only two problems that will induce vorticity into the flow are the latter two, both of which are captured in a VLM. The thickness portion of the problem will modify the pressure distribution, and therefore will affect the moment coefficients the most. With this in mind, the CPanel results for lift and drag coefficients should show a strong correlation with the AVL and lifting line theory.

The geometry used in this verification is shown in Figure 5.8. The airfoil is a NACA 4412 and the wing has a span of six and chord length of one. The mesh was created using OpenVSP and consists of 10,342 surface panels, with an additional 868 panels making up the wake surface that is not shown. Clustering of panels is concentrated near the leading edge, wing tips, and trailing edge in order to resolve the areas of high curvature and large gradients. The solution was computed using panel based normal vectors and does not reflect the improvement from using bezier surface based normal vectors. In comparing Trefftz and surface integrated

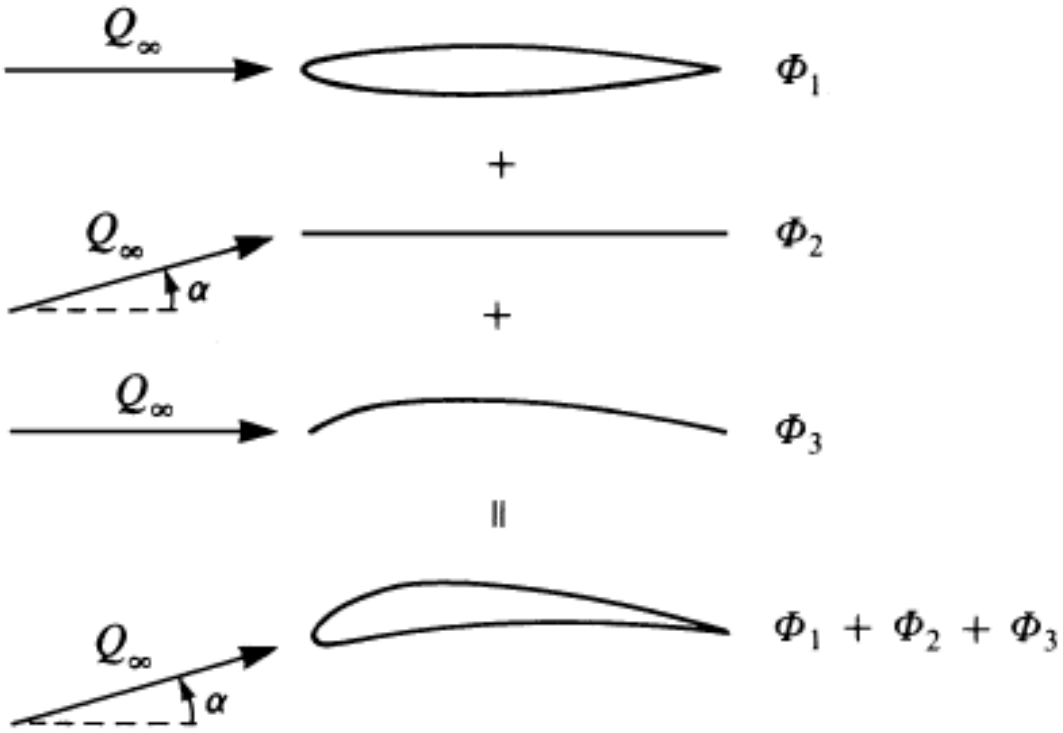


Figure 5.7: Solution to 2D Lifting Problem by Superposition[2]

coefficients, however, a solution using Bezier based normal vectors is shown for comparison.

The resulting lift, drag, and pitching moment coefficients are plotted in the figures below, alongside predictions from lifting line theory, AVL, and CBAERO. The lifting line theory results are based on fifteen spanwise vortices and are adjusted in order to reflect the zero lift angle of attack for a NACA 4412 airfoil of -4°.[37]

Beginning with the lift curve in Figure 5.9, the best agreement is shown between CPanel, AVL, and lifting line theory. CBAERO predicts consistently lower lift coefficient. The plots report the Trefftz plane lift coefficient and therefore

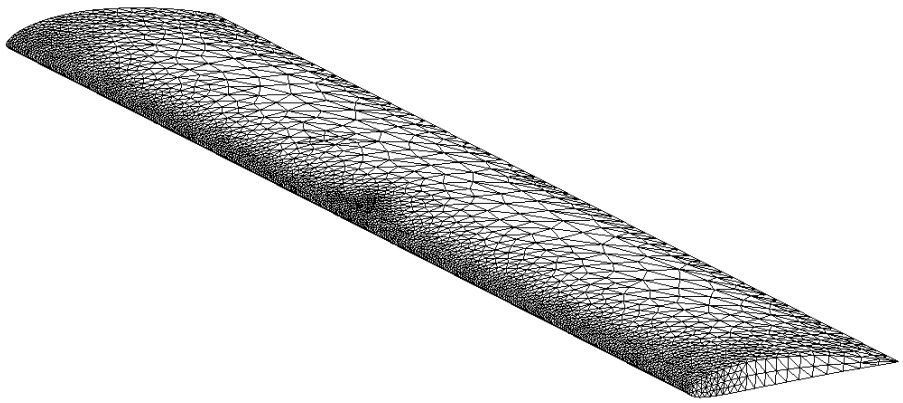


Figure 5.8: Discretized NACA 4412 Finite Wing Used in Lifting Flow Verification

differences cannot be attributed to the difference in velocity formulation between CBAERO and CPanel. The most likely cause, assuming that the fast tree method implemented in CBAERO showed agreement with the standard, fully dense, method, is the differences in implementation of the wake and enforcement of the Kutta condition. Kinney is not very specific in his documentation of how the Kutta condition is enforced, and therefore more detailed hypothesis for the source of the error cannot be explored. For this simple geometry, however, lifting line theory and the vortex lattice method employed in AVL provide very accurate predictions and the agreement between CPanel and those solution methods suggests that the Kutta condition enforcement in CPanel yields satisfactory results.

The induced drag shows a similar trend in Figure 5.10. The lower prediction of

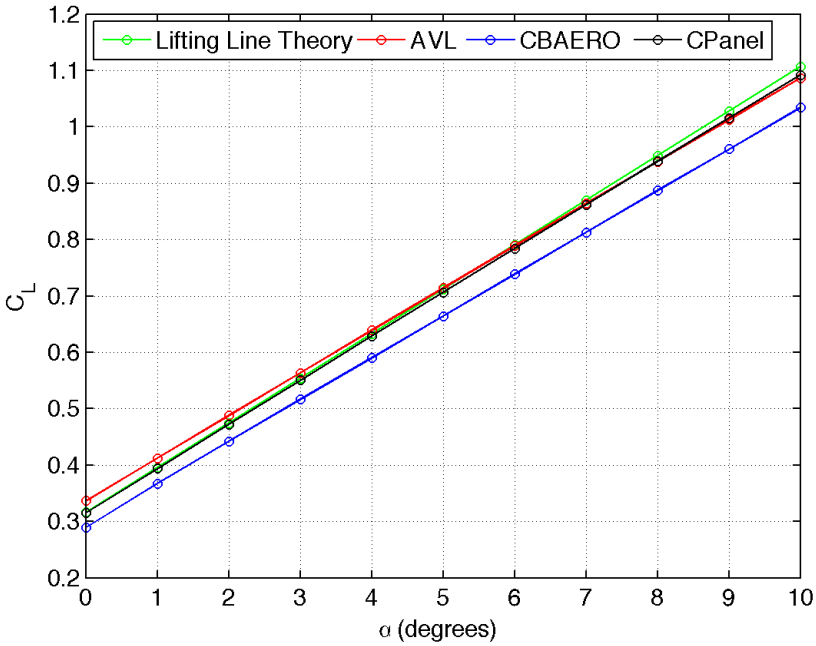


Figure 5.9: NACA 4412 C_L vs α

induced drag by CBAERO follows logically as the induced drag coefficient is proportional to the square of the lift coefficient. Another potential contributor to this discrepancy could be the manner in which the downwash is calculated. The numerical implementation of the Trefftz plane integration in CBAERO is not documented. However, assuming the implementation is similar to the CPanel method discussed in Section 3.3.3, the induced velocity normal to the wake, w, could be artificially decreased, resulting in a lower induced drag. This theory is supported later in Figures 5.13 and 5.14 as well.

The pitching moment, shown in Figure 5.11 is purely based on the integrated surface pressures, and therefore lends further insight into the differences in the velocity formulations. For this geometry, the center of gravity was computed in VSP

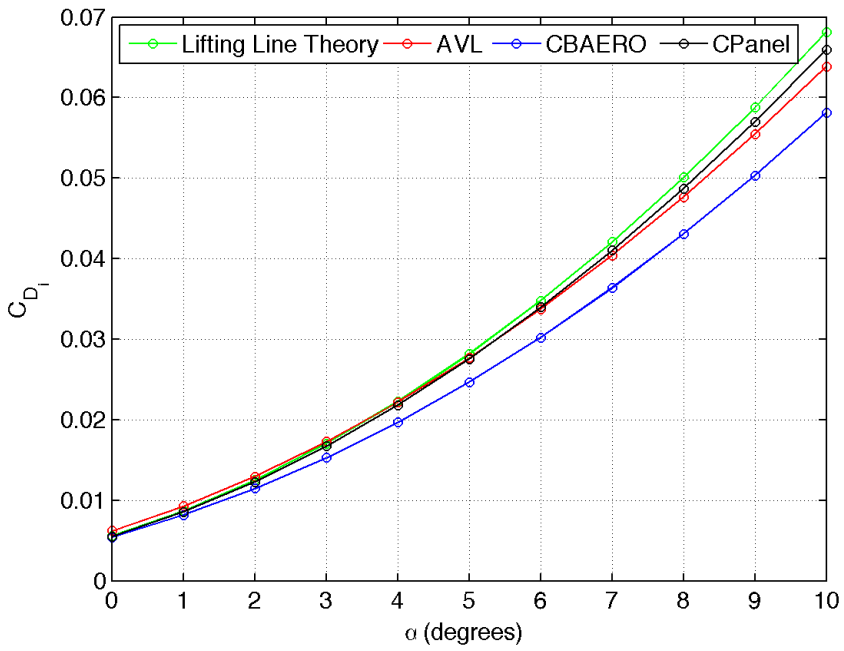


Figure 5.10: NACA 4412 C_{D_i} vs α

to be at a location of $[0.416, 0, 0.03]$. The largest contributor to the pitching moment is the suction peak that occurs in the region just aft of the leading edge. As was shown with the ellipsoid, artificial dampening of the vortex induced velocity would lower the velocity in this region and weaken the suction peak, thus decreasing the pitching moment. The impact would be most significant when the circulation is large, which occurs at higher angles of attack. This explains the difference in slope of the pitching moment curve between CBAERO and both the other solution from CPanel and AVL. The lower slope indicates dampening of the higher circulation at larger angles of attack in the CBAERO solution.

As a final check, the drag polar for the finite wing in Figure 5.12 shows very strong agreement between each of the methods.

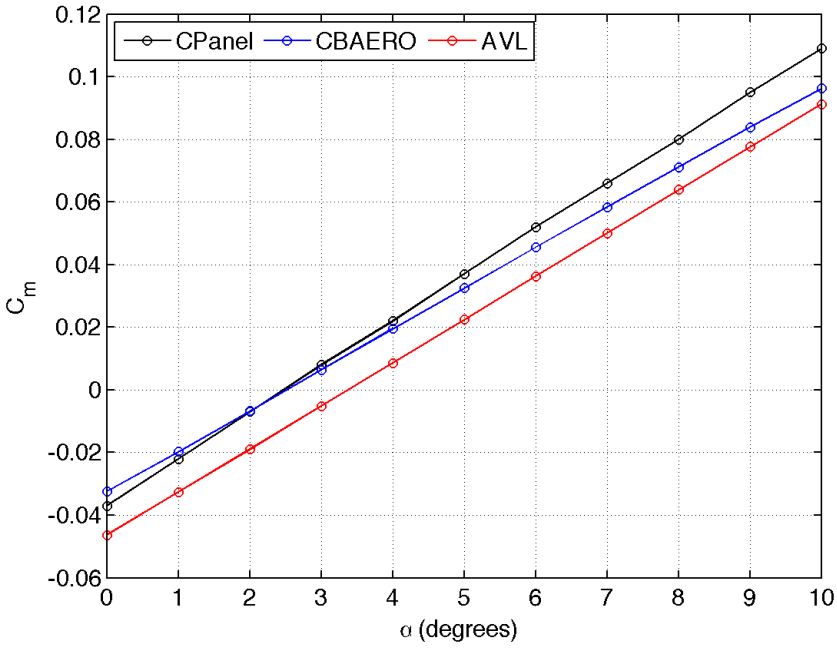


Figure 5.11: NACA 4412 C_m vs α

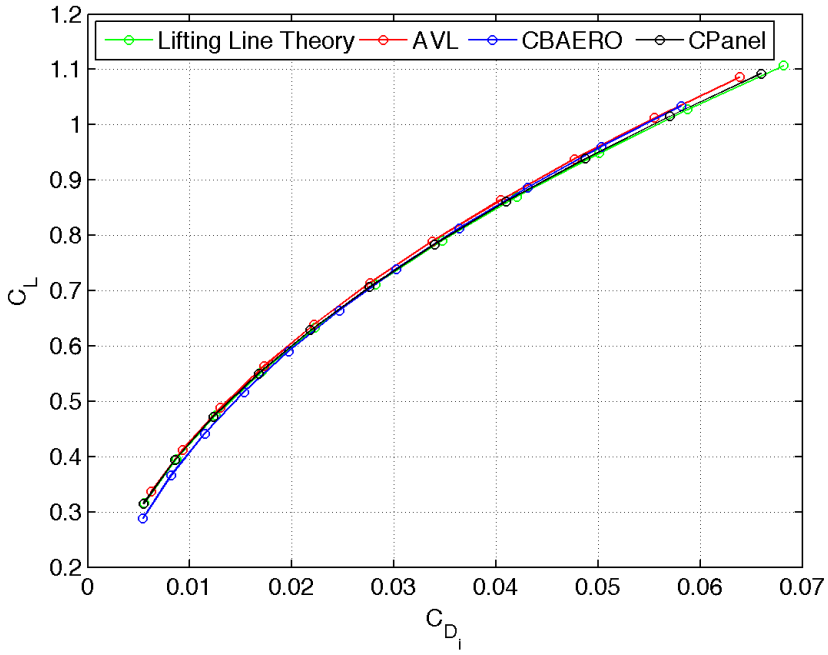


Figure 5.12: NACA 4412 Drag Polar

By comparing the predicted lift and induced drag coefficients using both surface and Trefftz plane integration, similar conclusions can be drawn regarding the effects of velocity survey methods. Figures 5.13 and 5.14 show both the methods of computing the coefficients for both CBAERO and CPanel. Theoretically, as the grid is refined to infinitely small panels, both methods should converge onto each other. CPanel demonstrates much stronger agreement between the two methods than CBAERO does, providing support for the CHTLS velocity formulation. The surface integrated lift coefficient from CBAERO is close, but slightly lower than the Trefftz plane solution. The artificial lowering of the surface velocities seen in the ellipsoid solution will result in higher pressures near the stagnation region on the leading edge, increasing the induced drag that is computed by surface integration. This is made most evident in the error of the induced drag calculation in CBAERO.

The strong agreement between the surface integrated coefficients and the Trefftz plane coefficients in CPanel demonstrates the increased accuracy obtained from the use of the CHTLS method, and that accuracy results in more accurate prediction of the moment coefficients and any variable requiring the use of the surface velocities. The method is enhanced using Bezier based normal vectors as well, as can be seen by the stronger agreement between the integrated and Trefftz plane coefficients using those exact normal vectors. In general, the agreement between CPanel and the other potential flow solution methods on a simple lifting geometry verifies the accuracy of the CPanel solution process, specifically the Kutta

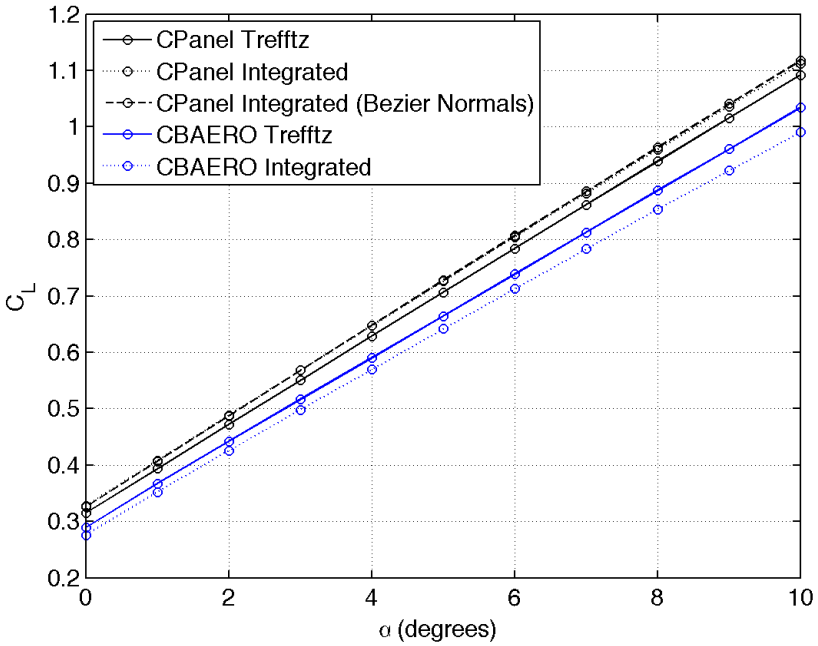


Figure 5.13: Comparison of Lift Coefficient Based on Trefftz Plane and Surface Integration

condition enforcement and velocity calculation.

5.4 SR22 Configuration

In order to ensure that CPanel operates properly with a generic aircraft configuration, results were generated for a Cirrus SR22 aircraft and compared to solutions from both CBAERO and VSPAERO. The SR22 model was obtained from the VSP Hangar and was created by Mark Moore. To minimize the mesh size, the landing gear struts, as well as the propeller have been removed from the model. The discretization, shown in Figure 5.15, consists of 22,500 panels. The mesh used in CPanel's analysis was drawn from a modified version of VSP and did include Bezier

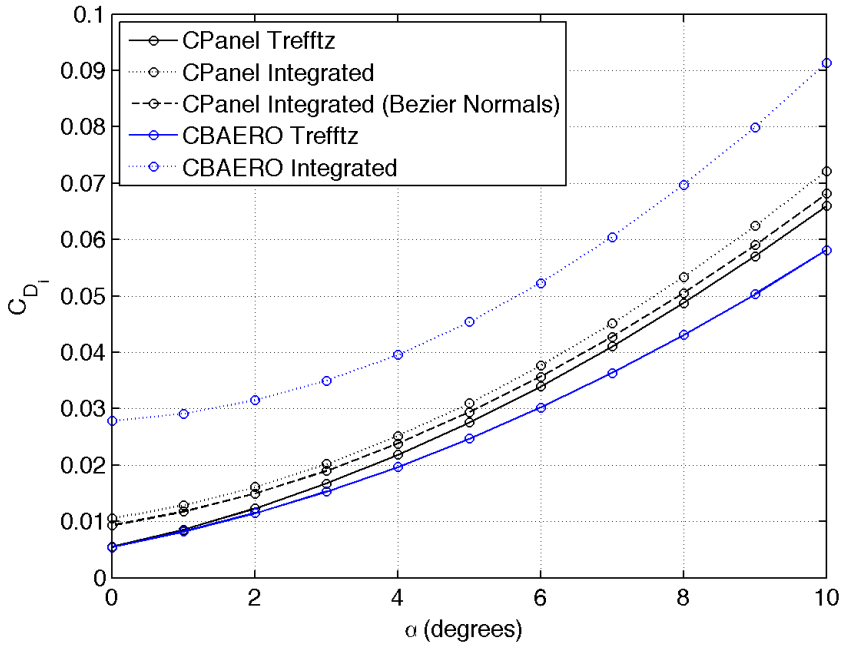


Figure 5.14: Comparison of Drag Coefficient Based on Trefftz Plane and Surface Integration

based normal vectors.

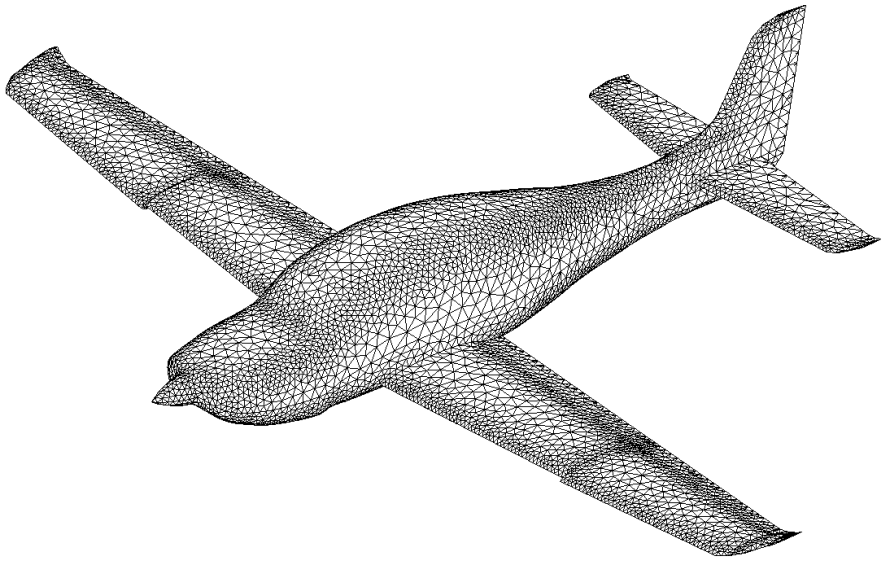


Figure 5.15: Discretized SR22

The resulting pressure distribution from CPanel is shown in Figure 5.16. A qualitative check shows that the pressure distribution is smooth, ensuring all the special cases for the CHTLS formulation are properly addressed. Additionally, the streamlines behave as expected and confirm the current algorithm is working properly with respect to seeding and termination of the streamlines.

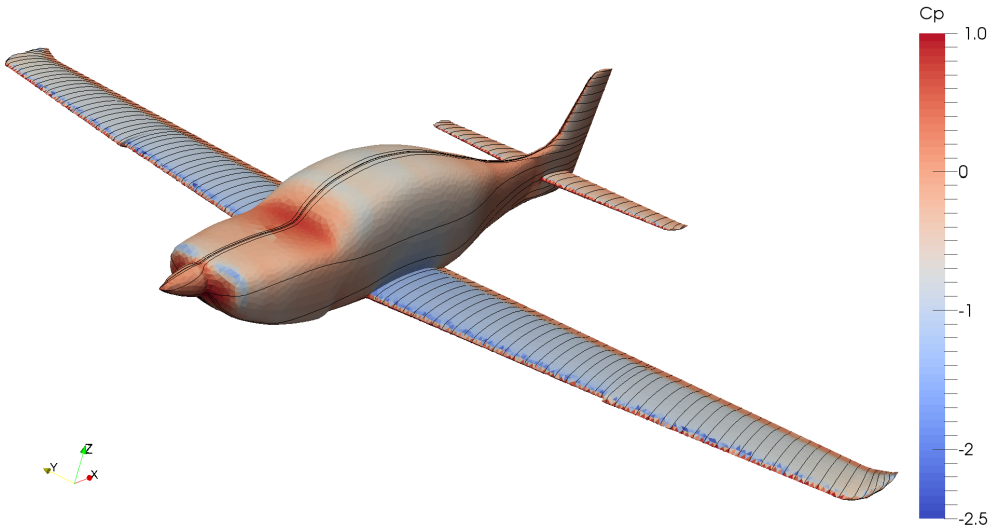


Figure 5.16: Pressure Distribution over SR22 ($\alpha = 5°, \beta = 0°$)

The lift and drag curves, as well as the drag polar are shown below in Figures 5.17-5.19. The differences between the CPanel and CBAERO results are similar, but more pronounced than those seen in the NACA 4412 analysis. CBAERO is consistently predicting lower lift and drag coefficients.

In the case of the SR22, the difference in the induced drag coefficient is rather significant. Based on the explanation that velocity calculation method could be causing a nonphysical dampening of the induced velocity from the panel edges, one

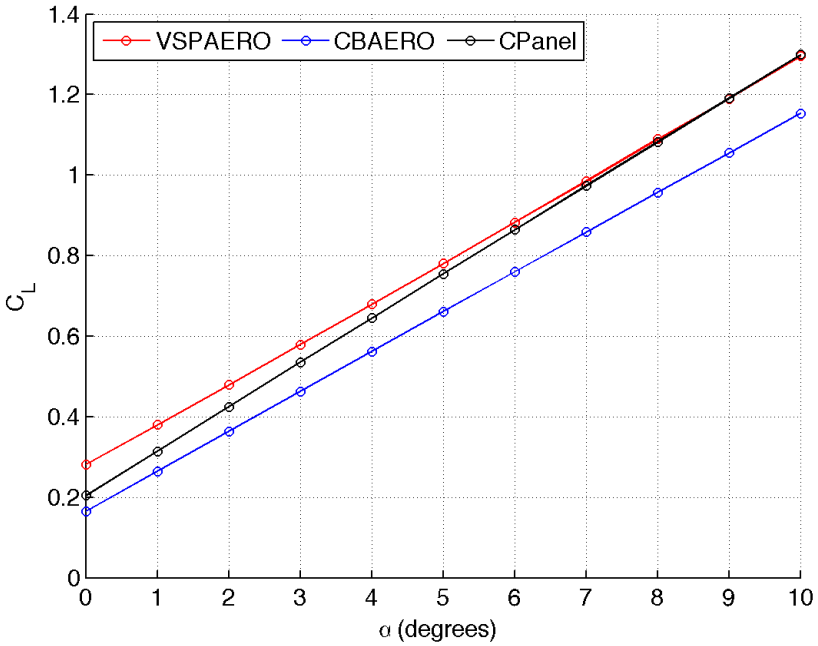


Figure 5.17: SR22 C_L vs α

would expect that with larger vortex strengths, the dampening would be more pronounced. This would explain the growing difference in induced drag with angle of attack, as higher angles of attack generate greater circulation and stronger vortices. The results of CPanel do, however, match more closely with VSPAERO results. As mentioned previously, this should be expected for the lift and drag coefficients.

Figures 5.20 and 5.21 show a comparison of the coefficients from integrating the surface pressures and from a Trefftz plane analysis. The results show a similar result to that of the NACA 4412, with CPanel demonstrating much stronger agreement between the two methods, especially in induced drag. This adds more support for the proposed explanation of the discrepancy in the results, and strengthens the case that the CHTLS method provides for more accurate surface velocities. In this case,

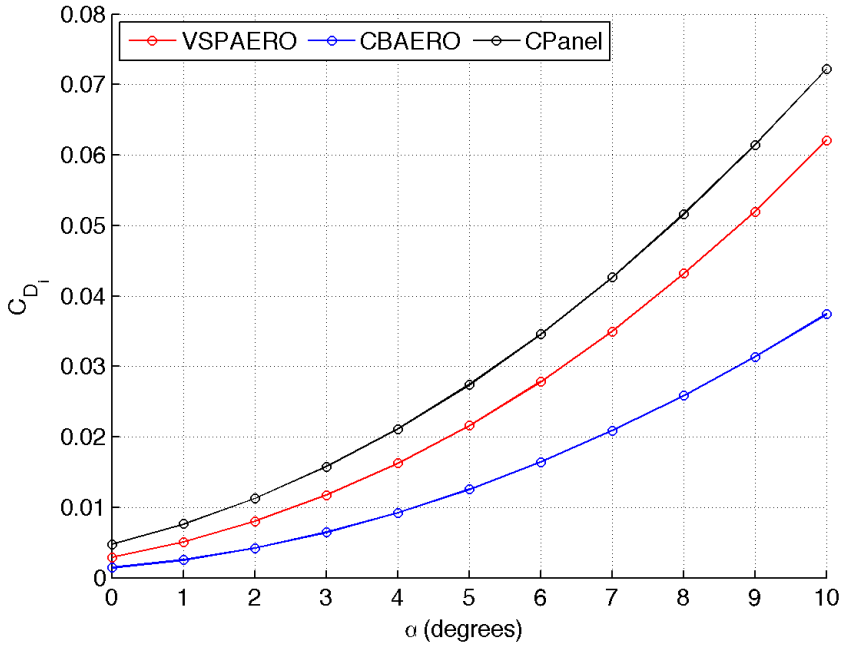


Figure 5.18: SR22 C_{D_i} vs α

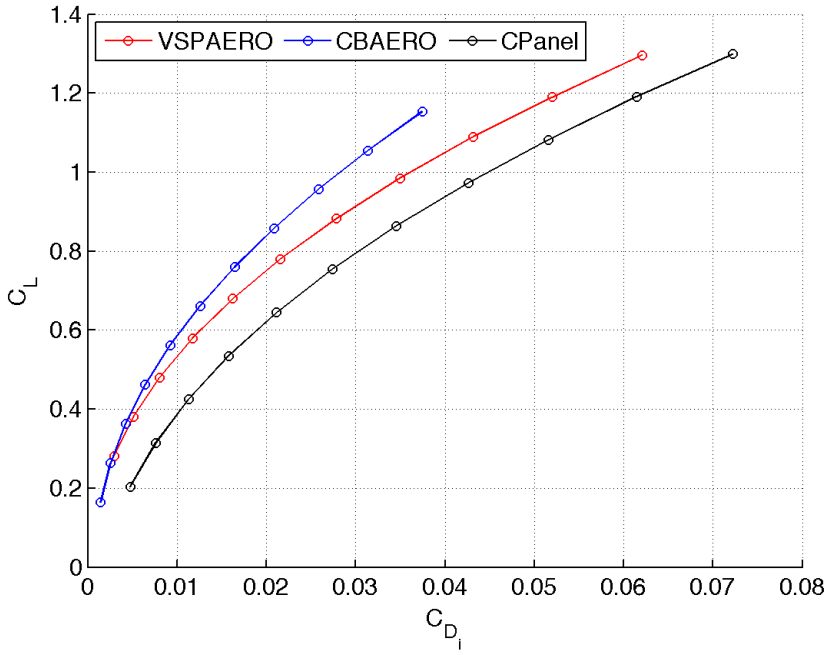


Figure 5.19: SR22 Drag Polar

the solution using Bezier based normal vectors is further from the Trefftz plane coefficients. One possible reason for this could be the lack of resolution in the leading and trailing edge due to restrictions on the mesh size. One would expect that with a better mesh, results would be more in line with what was seen on the NACA 4412 geometry.

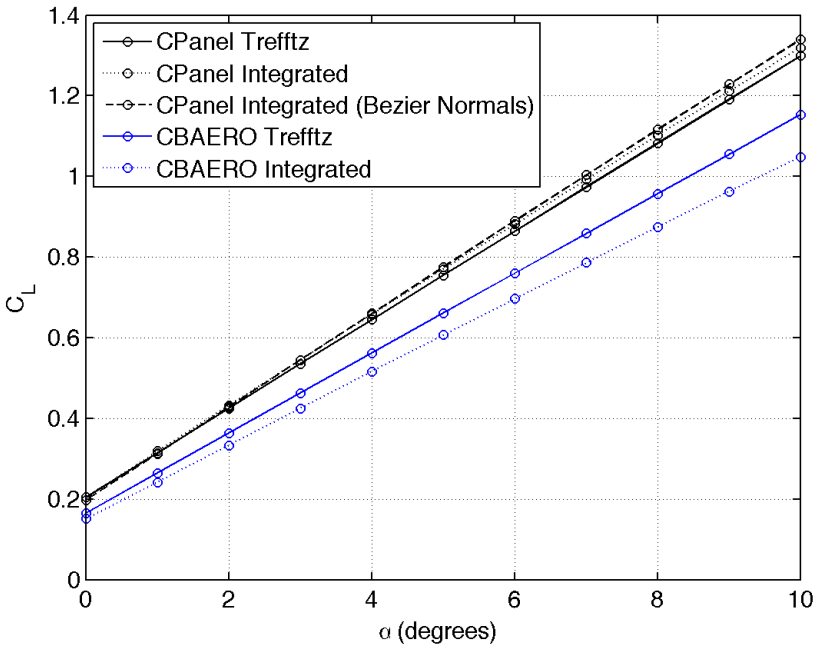


Figure 5.20: Comparison of Lift Coefficient Based on Trefftz Plane and Surface Integration for SR22

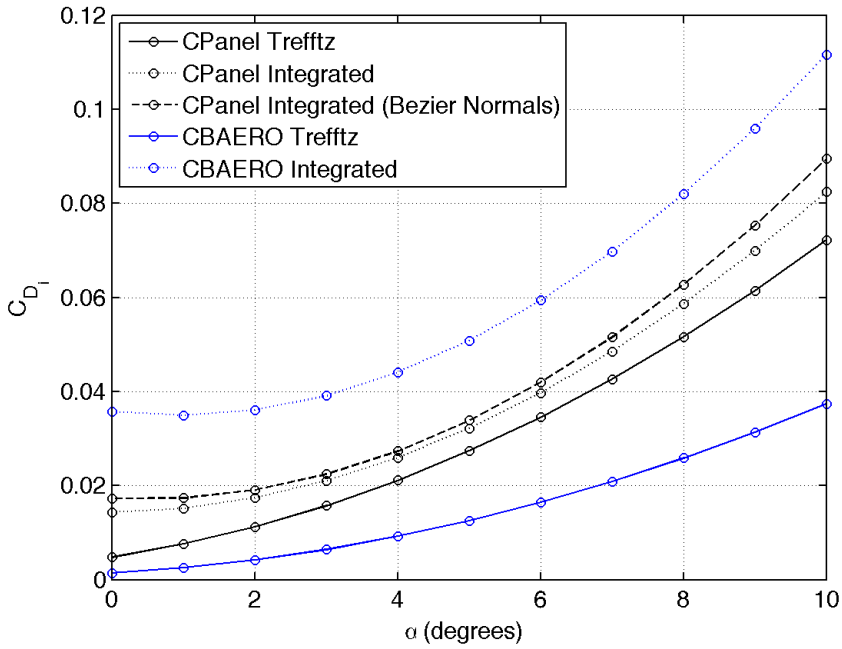


Figure 5.21: Comparison of Drag Coefficient Based on Trefftz Plane and Surface Integration for SR22

Chapter 6

Conclusion

6.1 Summary

A program was successfully developed that predicts the flow solution about arbitrary geometries in a potential flow. The program uses an unstructured surface discretization, saving the engineer significant time in the preprocessing phase of the analysis. The program was developed with ease of use and future development in mind, creating the potential for a longstanding project in the Cal Poly graduate program that will benefit undergraduate students with minimal required training to operate the program.

Both the open source nature of the project, as well as the application of a new velocity formulation separate CPanel from existing unstructured panel codes. Results compared to an exact solution showed an opportunity for improvement of the solution's accuracy without refinement of the discretization by use of normal vectors from the underlying bezier surface. This opportunity is purely a result of the CHTLS velocity formulation and would not be present in other unstructured panel

codes that use a viscous core model. The lifting flow results also reveal that the CHTLS velocity formulation generates surface forces that are more consistent with a Trefftz plane analysis. The agreement between the two solutions is greater than that seen in unstructured codes using a viscous core velocity influence formulation, such as CBAERO. This suggests that the results based purely on surface integration, such as moment coefficients and stability derivatives, are more accurate as well.

6.2 Future Work

While the theory of panel codes is well understood and documented, the nature of their restriction to potential flows drives developers to employ creative ways of obtaining a more accurate solution. This leaves the door wide open for future expansion of CPanel's capabilities. Currently, work is being done to employ a wake model made up of vortex particles, as is done in FastAero.[21] Additional enhancements include expansion to supersonic flows, transient simulations, and the development of a viscous solution method using an integral boundary layer along surface streamlines. In order for the viscous solution to be done properly, enhancement of the seeding method used in the current surface streamline algorithm will be required to ensure complete coverage of the surface.

In addition to capabilities, CPanel will benefit from some method of speeding up the generation of the linear system of equations. Most current panel codes use some fast tree method that represents entire portions of the domain, far from the

point of interest, as one element, resulting in a sparse matrix as opposed to the dense matrix that is currently constructed in CPanel. The result is reduction of both the memory requirements and solution time from $O(n^2)$ to $O(n \log n)$. An octree data structure is required for many of the speed up methods and therefore was employed early in the development of CPanel. Speed up would benefit all future developers working on CPanel and therefore should be placed near the top of the priority lists. As a final note, continued expansion of the program will likely call for a rigorous profiling of the code to identify areas where memory and computation time can be reduced.

BIBLIOGRAPHY

[1] Ramos, A., *Development of a Meshless Method to Solve Compressible Potential Flows*, Master's thesis, California Polytechnic State University, 2010.

[2] Katz, J. and Plotkin, A., *Low-Speed Aerodynamics*, Cambridge University Press, 2001.

[3] Ashby, D., Dudley, M., Iguchi, S., Browne, L., and Katz, J., *Potential Flow Theory and Operation Guide for the Panel Code PMARC 12*, NASA Ames Research Center, December 1992.

[4] Winder, R., "The Kinetic PR Quadtree," Tech. rep., University of Maryland, 2000.

[5] Marshall, D. and Mehiel, E., "Intoduction of Software Development Practices into Aerospace Engineering Curriculum," *46th AIAA Aerospace Sciences Meeting and Exhibit*, No. 2008-496, January, 2008.

[6] Katsikadelis, J., *Boundary Elements: Theory and Applications*, Elsevier Science Ltd, 2002.

[7] Kellogg, O. D., *Foundations of Potential Theory*, Dover, 1929.

[8] Winckelmans, G., Cocle, R., Dufresne, L., and Capart, R., "Vortex Methods and their Application to Trailing Wake Vortex Simulation," *Comptes Rendus Physique*, Vol. 6, 2005, pp. 467–486.

[9] Lamb, H., *Hydrodynamics*, Cambridge University Press, 1895.

[10] Xu, C., "Kutta Condition for sharp edge flows," *Mechanics Research Communications*, Vol. 25, No. 4, 1998, pp. 415–420.

[11] Malcevic, I., "Automated Blocking for Structured CFD Gridding with an Application to Turbomachinery Secondary Flows," *20th AIAA Computational Fluid Dynamics Conference*, No. 2011-3049, 2011.

[12] Erickson, L., "Panel Methods–An Introduction," Tech. Rep. 2995, NASA, 1990.

[13] Kinney, D., "Aero-Thermodynamics for Conceptual Design," *42nd AIAA Aerospace Sciences Meeting*, No. 2004-31, 2004.

[14] Hess, J. and Smith, A., "Calculation of Non-Lifting Potential Flow about Arbitrary Three-Dimensional Bodies," Tech. rep., Douglas Aircraft Division, 1962.

[15] Hess, J., "Calculation of Potential Flow about Arbitrary Three-Dimensional Lifting Bodies," Tech. rep., Department of the Navy, 1972.

[16] Maskew, B., "Program VSAERO Theory Document," Tech. Rep. 4023, NASA, 1987.

[17] Bramesfeld, G., *A Higher Order Vortex-Lattice Method with a Force-Free Wake*, Ph.D. thesis, Pennsylvania State University, 2006.

[18] Calabretta, J., *A Three Dimensional Vortex Partical-Panel Code for Modeling Propeller-Airframe Interaction*, Master's thesis, California Polytechnic State University, 2010.

[19] van Garrel, A., "Development of a Wind Turbine Aerodynamics Simulation Module," Tech. rep., Netherlands Agency for Energy and the Environment, 2003.

[20] Winckelmans, G., *Topics in Vortex Methods for the Computation of Three and Two Dimensional Incompressible Unsteady Flows*, Ph.D. thesis, California Institute of Technology, 1989.

[21] Moore, J., *An Arbitrarily High-Order, Unstructured, Free-Wake Panel Solver*, Master's thesis, Massachusetts Institute of Technology, 2013.

[22] McDonald, R. and Ramos, A., "Constrained Hermite Interpolation for Mesh-Free Derivative Estimation Near and on Boundaries," *AIAA Journal*, Vol. 49, No. 10, 2011.

[23] Smith, S., "A Computational and Experimental Study of Nonlinear Aspects of Induced Drag," Tech. Rep. 3598, NASA, 1996.

[24] Lundry, J., "Calculation of Lift and Induced Drag from Sparse Span Loading Data," *Journal of Aircraft*, Vol. 14, No. 3, 1977.

[25] Archer, G., *Object-Oriented Finite Element Analysis*, Ph.D. thesis, University of California at Berkeley, 1996.

[26] Baker, P., Dai, Z., Grabowski, J., Haugen, O., Schieferdecker, I., and Williams, C., *Model-Driven Testing*, Springer, 2008.

[27] Willis, D., Peraire, J., and White, J., "A combined pFFT-multipole tree code, unsteady panel method with vortex particle wakes," *International Journal for Numerical Methods in Fluids*, 2000.

[28] Samet, H., *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, 2006.

[29] Epton, M. and Magnus, A., "PAN AIR - A Computer Program for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configuration Using a Higher Order Panel Method," Tech. Rep. 3253, NASA, 1990.

[30] Boschitsch, A., Curbishley, T., Quackenbush, T., and Teske, M., "A Fast Panel Method for Potential Flows about Complex Geometries," Tech. rep., Conintuum Dynmaics, Inc., 1996.

[31] Willis, D., *An Unsteady, Accelerated, High Order Panel Method with Vortex Particle Wakes*, Ph.D. thesis, Massachusetts Institute of Technology, 2006.

[32] Eller, D. and Carlsson, M., "An efficient aerodynamic boundary element method for aeroelastic simulations and its experimental validation," *Aerospace Science and Technology*, Vol. 7, No. 7, 2003, pp. 532–539.

[33] Filkovic, D., *Graduate Work*, Master's thesis, University of Zagreb, 2008.

[34] Kinney, D., Garcia, J., and Huynh, L., "Predicted Convective and Radiative Aerothermodynamic Environments for Various Reentry Vehicles Using CBAERO," *44th AIAA Aerospace Sciences Meeting and Exhibit*, No. 2006-659, 2006.

[35] "1012-2012 - IEEE Standard for System and Software Verification and Validation," .

[36] Munk, M., "Remarks on the Pressure Distribution Over the Surface of an Ellipsoid, Moving Translationally through a Perfect Fluid," Tech. rep., National Advisory Committee for Aeronautics, 1924.

[37] W.H.Mason, "Simple Lifting Line Theory for Unswept Trapezoidal Wings," Tech. rep., Virginia Polytechnic Institute and State University, 1971.

APPENDICES

Appendix A

CPanel Input and Output Files

The following sections will give brief descriptions of the file formats used in CPanel. Usage of the program requires only the input file, with the command below issued from the command line.

```
$ CPanel InputFile.CPin
```

Prior to running CPanel, the location of the executable should be added to the $PATH variable with the following command

```
$ PATH=$PATH:/path/to/the/executable
```

A.1 Input File

An example of a CPanel input file is shown below. Upon running, the input file and all associated files will be placed in a new subfolder with the name of the geometry. The geometry file should be listed as a relative path to the input file, or an absolute path. At this stage, CPanel accepts two geometry file formats, .tri and

.tricp, which will be described in the following section. The names of each variable should appear exactly as they do here or they will not be read properly. Additionally, there must be a space in between the variable, the equal sign, and its value for the line to be parsed correctly. Comments can be added, preceded by a %, and will be skipped when reading the file. At this stage, CPanel is for only subsonic flows and therefore Mach numbers greater than one will cause the program to fail. Due to limitation of the Prandtl Glauert correction, it is not recommended to use Mach numbers over 0.6, as the linear approximation begins to break down.

```
%% CPanel Input File %%
% Reference Geometry %
   GeomFile = SR22.tricp
   S_ref = 144.889
   b_ref = 38.556
   c_ref = 3.94
   X_cg = 2.603
   Y_cg = 0.0
   Z_cg = 0.106
% Cases %
   Velocity (ft/s)
   1
   100.0
   Angle_of_Attack (degrees)
   11
   0.0
   1.0
   2.0
   3.0
   4.0
   5.0
   6.0
   7.0
   8.0
   9.0
```

```
    10.0
    Angle_of_Sideslip (degrees)
    3
    0.0
    2.0
    4.0
    Mach_Number
    2
    0.1
    0.3
% Solver Options (0 = OFF, 1 = ON) %
    Surface_Streamlines
    0
    Stability_Derivatives
    1
    Write_Influence_Coefficients
    0
```

It should be noted that turning any of the solver options on will increase computation time significantly. Surface streamlines can be run on a coarser mesh to save time, without losing too much accuracy in the visualization. In order to calculate the stability derivatives, two perturbed cases are run, one for both α and β. Writing the influence coefficient matrix to a file is recommended if the mesh size is under 10,000 cells and numerous additional runs are anticipated. The time to write and read the file becomes quite significant and the file size approaches 1 GB as the mesh size rises above 10,000 cells.

A.2 Mesh File Formats

CPanel currently accepts two file formats for the mesh, .tri and .tricp. the TRI file format is described in the Cart3D documentation. A brief description follows

below. More information can be found at

http://people.nas.nasa.gov/aftosmis/cart3d/cart3dTriangulations.html.

```
nVertsnTris
x_1  y_1  z_1
x_2  y_2  z_2
x_3  y_3  z_3
.
.
.
x_nVerts  y_nVerts  z_nVerts
v1_t1  v2_t1  v3_t1
v1_t2  v2_t2  v3_t2
v1_t3  v2_t3  v3_t3
.
.
.
v1_nTris  v2_nTris  v3_nTris
surfID_1
surfID_2
surfID_3
.
.
.
surfID_nTris
```

The .tricp format is simply a modified .tri format that is appended with bezier based normal vectors taken from the centroid of each panel. The appended normals appear after the surface IDs and are formatted as follows.

```
nx_1  ny_1  nz_1
nx_2  ny_2  nz_2
nx_3  ny_3  nz_3
.
.
.
nx_nTris  ny_nTris  nz_nTris
```

A.3 Output Files

Following completion of running CPanel, all output files will be placed in a subdirectory with the name of the geometry file. For example, a geometry file named Cessna.tri will result in file being placed in /currentDirectory/Cessna. The files printed by CPanel are a summary file and files containing the surface data for visualization.

A.3.1 .CPout Summary File

The summary file will echo the inputs for the simulation first, followed by the results for each combination of velocity, angle of attack, angle of sideslip and Mach number. For each case, the Trefftz plane lift and drag coefficients are printed. The integrated forces in both body and wind axes are printed as well. Moment coefficients follow the integrated forces, and if the stability coefficient option is turned on, these will be printed last.

A.3.2 Visualization Files

All data needed for visualization of the results is written to .vtu files, the VTK file format for data on an unstructured grid. This file type can be read by a number of third party visualization tools. For the work of this thesis, Paraview was used. The details of the VTU file can be found at
`http://www.vtk.org/wp-content/uploads/2015/04/file-formats.pdf`.