A HIGHER-ORDER METHOD IMPLEMENTED IN AN UNSTRUCTURED PANEL CODE TO MODEL LINEARIZED SUPERSONIC FLOWS

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

Jake Daniel Davis

February 2019

COMMITTEE MEMBERSHIP

TITLE:	A Higher-Order Method Implemented in an Unstructured Panel Code to Model Linearized Supersonic Flows
AUTHOR:	Jake Daniel Davis
DATE SUBMITTED:	February 2019
COMMITTEE CHAIR:	David D. Marshall, Ph.D. Professor of Aerospace Engineering
COMMITTEE MEMBER:	Paulo Iscold, Ph.D. Assistant Professor of Aerospace Engineering
COMMITTEE MEMBER:	Colleen M. Kirk, Ph.D. Professor of Mathematics
COMMITTEE MEMBER:	Robert A. McDonald, Ph.D. Head of Vehicle Engineering, Uber Elevate, Uber
COMMITTEE MEMBER:	Graham A. Doig, Ph.D. Senior Aerodynamics Engineer, Evelozcity

ABSTRACT

A Higher-Order Method Implemented in an Unstructured Panel Code to Model Linearized Supersonic Flows

Jake Daniel Davis

Since their conception in the 1960s, panel codes have remained a critical tool in the design and development of air vehicles. With continued advancement in computational technologies, today's codes are able to solve flow fields around arbitrary bodies more quickly and with higher fidelity than those that preceded them. Panel codes prove most useful during the conceptual design phase of an air vehicle, allowing engineers to iterate designs, and generate full solutions of the flow field around a vehicle in a matter of seconds to minutes instead of hours to days using traditional CFD methods. There have been relatively few panel codes with the capacity to solve supersonic flow fields, and there has been little recently published work done to improve upon them.

This work implements supersonic potential flow methods into Cal Poly's open source panel code, CPanel. CPanel was originally developed to solve steady, subsonic flows utilizing constant strength source and doublet panels to define the geometry, and an unstructured geometry discretization; it was later extended to include viscous vortex particle wakes and transient modeling. In this thesis, a higher-order method is implemented in CPanel for use in solving linearized supersonic flows, where a higher-order method is one that utilizes at least one singularity element whose order is higher than constant. CPanel results are verified against analytical solutions, such as the Taylor-Maccoll solution for supersonic conical flows and 2D shock-expansion theory, and the PANAIR and MARCAP supersonic panel codes. Results correlate well with the analytical solutions, and show strong agreement with the other codes.

ACKNOWLEDGMENTS

My tenure at Cal Poly as both an undergraduate and graduate student has been a formative one, and I would not have managed to overcome the various trials and tribulations I encountered throughout this time without those who have supported me along the way.

I must first thank my thesis advisor, Dr. Marshall, for originally sparking my interest in numerics and taking me on as a graduate student as well as for his continued guidance throughout this project, which was instrumental to the success I found with this thesis. I am greatly appreciative of the many hours he gave in assisting me work through problems and setting me on a course for success. I would also like to thank Dr. Doig who had a large role in setting me on the path I am on now. Through his undergraduate and graduate courses and his passion for education and hands-on learning, I was able to realize my own passions both academically and professionally. And thank you to my committee for your guidance and time throughout this project.

Chris Satterwhite and Connor Sousa created and maintained an easy to read and use code base providing me a springboard from which to start my thesis, and for that I would like to thank them both.

None of my accomplishments would have been possible without the support of my family and friends. Thank you to my parents and siblings for their relentless support and guidance through all of my endeavors. To my friends and roommates during my time at Cal Poly, I thank you for good company and memorable experiences both in and outside the classroom. And to my girlfriend, Sam, who has stood by my side through it all and continually pushed me forward.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

NOMENCLATURE

Acronyms

BIE	Boundary Integral Equation
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CHTLS	Constrained Hermite Taylor Least-Squares
CSYS	Coordinate System
DOD	Domain of Dependence
DOI	Domain of Influence
PDE	Partial Differential Equation
VLM	Vortex Lattice Method

English Symbols

A	Transformation matrix
$\mathbf{A}$	Doublet influence coefficient matrix
a	Speed of sound; Doublet influence coefficient
B	Alternative form of Prandtl-Glauert factor
$\mathbf{B}$	Source influence coefficient matrix
b	Source influence coefficient
C_p	Pressure coefficient
c	Wing chord
c_r	Wing root chord
H	Subsonic fundamental integral
I	Subsonic doublet influence fundamental integral solutions
J	Subsonic source velocity fundamental integral solutions
$\mathbf{J}$	Vector of subsonic source velocity fundamental integral solutions

J_a	Supersonic source influence coefficient scaling factor
K	Supersonic doublet influence fundamental integral solutions
k	Control point offset factor
l_{avg}	Control point edge length average
M	Mach number
m	Edge slope
N_d	Number of doublets
N_n	Number of nodal points
N_p	Number of panels
N_s	Number of sources
$\hat{n}$	Unit normal vector
$\bar{n}_c$	Conormal vector
$\mathbf{P}$	Position vector, influenced point
$\mathbf{P}_0$	Position vector, influencing point
$\mathbf{P}_n$	Position vector, nodal point
p	Pressure
Q_I	Supersonic fundamental integral component
R	Generic hyperbolic distance; Gas constant
R_B	Hyperbolic distance for supersonic freestream Mach number
R_β	Hyperbolic distance for subsonic freestream Mach number
$\mathbf{r}$	Relative position vector
S	Generic surface of integration
S_a	Surface of object in potential flow
s	Wing span; Entropy
t	Time
U	Freestream velocity magnitude; Prandtl-Glauert equation solutions
$\mathbf{U}$	Freestream velocity vector

u, v, w	Perturbation velocity vector components
V	Total velocity magnitude
$\mathbf{V}$	Total velocity vector
$\forall$	Generic volume of integration
$\mathbf{v}$	Perturbation velocity vector
$\mathbf{v}_{offset}$	Control point offset vector
w_0	Supersonic fundamental integral component
$\mathbf{w}$	Perturbation mass flux vector
x_m, y_m, z_m	Supersonic edge oblique coordinate system
$\hat{x}_m, \hat{y}_m, \hat{z}_m$	Subsonic edge oblique coordinate system

Greek Symbols

α	Angle of attack
α_{ij}	Higher-order doublet panel influence coefficient expression
$\boldsymbol{\alpha}_{ij}$	Vector of higher-order doublet panel influence coefficients
β	Prandtl-Glauert factor
γ	Ratio of specific heats
θ	Subsonic edge inclination; Geometry half-angle
λ	Reciprocal of edge slope
μ	Doublet strength; Mach angle
$\boldsymbol{\mu}$	Vector of doublet strengths
ξ, η, ζ	Panel local coordinate system
Φ	Total velocity potential
$\boldsymbol{\Phi}$	Vector of total velocity potentials
ϕ	Perturbation velocity potential; Cone azimuth angle
$\boldsymbol{\phi}$	Vector of node-based influence coefficients
ρ	Density
σ	Source strength

$\boldsymbol{\sigma}$	Vector of source strengths

Subscripts

A	Average
cp	Quantity at control point
D	Difference; Doublet
i	Quantity interior of a body; Counter for influenced control points
j	Counter for influencing panels
k	Counter for influencing nodes
L	Lower
l	Quantity in local panel coordinate system
S	Source
U	Upper
∞	Freestream quantity

Superscripts

$*$	Finite part

Chapter 1

INTRODUCTION

1.1 Motivation

With today's advancements in computational technologies, simulation tools have become a staple in the engineering design process. The demands of the aerospace industry continue to diversify, requiring designers to create and use new tools in order to work quickly and efficiently. Rapid iteration during the conceptual design process of air vehicles necessitates the utilization of quick evaluation methods. This often demands the use of potential-based panel methods as opposed to the more common Navier-Stokes based Computational Fluid Dynamics (CFD) methods.

Panel methods are favorable over traditional CFD when computational speed is necessary. They have the ability to assess the flow field around a three-dimensional body using surface discretization as opposed to volumetric, resulting in run times on the order of seconds to minutes instead of hours to days. Most panel methods use a structured geometry discretization requiring substantial work up front by the user; however, the implementation of unstructured paneling, similar to that seen in most commercial CFD software packages, drastically cuts down the work required to set up a simulation. Furthermore, with growing interest in supersonic flight in the defense industry and commercial market, such tools could be utilized in the evaluation of new aircraft configurations designed specifically for the flight regime.

Most potential-based panel codes have been built for the evaluation of incompressible flow fields only, the first of which were developed in the early 1960s. Shortly after these methods were developed and published, Woodward and Carmichael created the first panel method to successfully model supersonic flow, referred to as the Woodward-Carmichael method [3, 4]. Several methods followed featuring different types and applications of singularity elements all using lower-order methods; however, each of them experienced numerical instabilities when modeling steady supersonic flows. It was not until the development of PANAIR in the 1980s that a supersonic panel method successfully modeled steady supersonic flow, while avoiding the instability problems of its predecessors. The difference in PANAIR was the enforcement of continuous doublet strength over the entire surface of a body, ensured by the use of quadratically varying doublets superimposed with linearly varying sources onto a curved surface [5]. The use of higher-order methods to enforce doublet strength continuity was found to be key in modeling supersonic flow using panel methods, and was implemented by programs following PANAIR such as HISSS [6] and MARCAP [7]. One other key common characteristic between all the aforementioned methods is the use of a structured surface discretization, which places a large burden on the user to correctly prepare geometry for evaluation. Vortex Lattice Methods (VLM), such as VORLAX [8] and VSPAERO [9], have also been utilized to solve supersonic flows, though these methods are lower fidelity than panel methods. There has yet to be a publicly available panel code that utilizes the accuracy and stability of higher-order methods for supersonic simulations paired with the convenience of unstructured paneling.

The marriage of unstructured paneling with a higher-order panel method would allow the designer to move seamlessly from a Computer Aided Design (CAD) model to a complete flow profile of an arbitrary geometry in subsonic or supersonic flight regimes. This capability would allow for rapid iteration and development of a diverse range of aircraft configurations by streamlining the conceptual design process. The development of such a capability is the subject of this paper.

1.2 Approach

CPanel is an unstructured panel code originally developed as v1.0 by Chris Satterwhite [10] and developed further as v2.0 by Connor Sousa [11] at Cal Poly, and is the platform from which this work is building from. It is an open source program developed in ANSI C++ with an object oriented approach and uses a Dirichlet boundary condition with constant strength sources and doublets representing body panels, and a vortex sheet for wake modeling. The key aspect of CPanel v1.0 is the choice of velocity formulation to handle unstructured paneling: a Constrained Hermite Taylor Least-Squares (CHTLS) method as opposed to a viscous core model [10]. With v2.0, a viscous vortex particle wake handling option was added to CPanel to allow for more accurate modeling of wake shapes, which brought with it the ability to run unsteady simulations [11]. The features implemented in v2.0 will not have bearing on this work. The reader is directed to Chapter 2 of the thesis by Sousa for a high-level overview of the original CPanel implementation, and to the thesis by Satterwhite for the detailed theory and implementation [11, 10].

This thesis will further expand the capabilities of CPanel by including the ability to model steady, supersonic flows around arbitrary bodies. Various modifications will be made to CPanel to accomplish this. To solve compressible flows using potential flow theory, the Prandtl-Glauert equation must be solved as opposed to the Laplace equation. This solution yields a different integral equation than the one currently used in CPanel; for the supersonic regime specifically, it introduces further difficulties to the solution process that do not arise in the case of subsonic flow. Flow disturbance propagation must be restricted to the Mach cone, whereas in subsonic flow, any disturbance propagates throughout the entire flow field. Furthermore, to avoid the numerical instability problems seen by early supersonic codes, a higher-order method is implemented.

1.3 Document Structure

This document mimics the structure of the original CPanel documentation in an effort to emphasize the differences in implementation required for the presented supersonic panel method. Chapter 2 covers the theory of supersonic potential flows. Chapter 3 shows how the aspects of a typical panel code, and specifically CPanel, must change to accommodate supersonic flows, and discusses the methods used by other codes. The specifics of how these changes were implemented into CPanel are discussed in Chapter 4. Lastly, the results of CPanel v3.0 are presented and discussed in Chapter 5 along with verification from exact theory and other supersonic panel codes. It should be noted that when CPanel is referred to throughout this document, the version including the implementation presented here is being referenced, unless otherwise specified.

Since many of the methods used throughout this work were developed by others for use in other codes, this paper focuses on how these methods were implemented in CPanel rather than the in depth theory and derivations behind them. Though, what has been determined to be essential background information in regard to these topics is discussed. Furthermore, with the future development and expansion of CPanel in mind, the content is structured and presented with the aim of providing the reader an understanding of how various aspects of the code function, and of the physical meaning behind some of the mathematics of these methods.

Chapter 2

THEORY AND GENERAL NUMERICAL IMPLEMENTATION

This chapter discusses the theory behind supersonic potential flow methods, then moves into the development of the equations from which most supersonic panel codes have been built, including the implementation of CPanel presented in this thesis. The different boundary conditions that can be used to ensure a unique solution are presented. Throughout the chapter, the theory and derivations are compared and contrasted with their subsonic counterparts.

2.1 Prandtl-Glauert Equation for Linearized Compressible Flow

For further details of the following derivation, the reader is directed to Katz and Plotkin [12] and Cummings et al. [13], as well as Ehlers et al. [2] for an alternate approach.

To arrive at the linearized potential equation for compressible flow, commonly known as the Prandtl-Glauert equation, one can begin with the Euler equations for steady, inviscid, and irrotational flow.

$$\boldsymbol{\nabla} \times \mathbf{V} = \mathbf{0} \tag{2.1}$$

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

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

Equation 2.2 can then be expanded and rewritten as

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

and because this derivation must account for compressibility, the first term in this equation is not zero as it would be for the incompressible case. Continuing this derivation with the removal of all density terms in the Euler equations would result in Laplace's equation, as shown by Satterwhite [10].

Equation 2.3 can be expanded to write the momentum equation in the x-, y-, and z-directions. Then multiplying each by the differential in their respective directions, substituting in the conditions for irrotationality, and adding them together yields

$$dp = -\frac{\rho}{2}d(V^2) = -(\rho V)dV \tag{2.5}$$

With the condition of irrotational flow, a scalar function called the total velocity potential, Φ, is defined such that

$$\mathbf{V} = \boldsymbol{\nabla}\Phi \tag{2.6}$$

and can be substituted into Equation 2.5 giving

$$dp = -\frac{\rho}{2}d(V^2) = -\frac{\rho}{2}d(|\boldsymbol{\nabla}\Phi|^2) \tag{2.7}$$

The pressure differential on the left hand side of the above equation can be replaced with a density differential using the definition of speed of sound, a, in an isentropic fluid,

$$a^2 = \frac{dp}{d\rho} \tag{2.8}$$

Rewriting Equation 2.7 as

$$d\rho = -\frac{\rho}{2a^2}d(|\boldsymbol{\nabla}\Phi|^2) \tag{2.9}$$

and taking the partial derivative of the density with respect to the x-, y-, and z-directions, combining them back into Equation 2.2 and collecting terms yields

$$\begin{gathered}\left(1-\frac{\Phi_x^2}{a^2}\right)\Phi_{xx}+\left(1-\frac{\Phi_y^2}{a^2}\right)\Phi_{yy}+\left(1-\frac{\Phi_z^2}{a^2}\right)\Phi_{zz}\\ -\left(\frac{2\Phi_x\Phi_y}{a^2}\right)\Phi_{xy}-\left(\frac{2\Phi_x\Phi_z}{a^2}\right)\Phi_{xz}-\left(\frac{2\Phi_y\Phi_z}{a^2}\right)\Phi_{yz}=0\end{gathered} \tag{2.10}$$

where

$$a^2 = a_\infty^2 - \frac{\gamma-1}{2}(\Phi_x^2+\Phi_y^2+\Phi_z^2) \tag{2.11}$$

This equation is the full potential equation. It can be used to describe any inviscid, adiabatic, and irrotational flow, but because it is non-linear in its current form, it is difficult to solve and would likely require a numerical solution. Instead, this equation can be linearized, and in its linearized form, an analytic solution can be obtained. To do this, perturbation velocity and perturbation velocity potential are introduced and defined such that

$$\boldsymbol{\nabla}\Phi = U + \boldsymbol{\nabla}\phi \tag{2.12}$$

$$\boldsymbol{\nabla}\phi = \mathbf{v} = (u, v, w) \tag{2.13}$$

where U is freestream velocity magnitude, the vector of which is assumed to be only in the x-direction. Equation 2.10 can now be rewritten in terms of perturbation velocity potential, multiplied through by a in terms of the perturbation velocity potentials, and terms collected to find

$$\begin{aligned}
(1-&M_\infty^2)\phi_{xx} + \phi_{yy} + \phi_{zz} = \\
&M_\infty^2\left[(\gamma+1)\frac{\phi_x}{U} + \left(\frac{\gamma+1}{2}\right)\frac{\phi_x^2}{U^2} + \left(\frac{\gamma-1}{2}\right)\left(\frac{\phi_y^2+\phi_z^2}{U^2}\right)\right]\phi_{xx} \\
&+M_\infty^2\left[(\gamma-1)\frac{\phi_x}{U} + \left(\frac{\gamma+1}{2}\right)\frac{\phi_y^2}{U^2} + \left(\frac{\gamma-1}{2}\right)\left(\frac{\phi_x^2+\phi_z^2}{U^2}\right)\right]\phi_{yy} \\
&+M_\infty^2\left[(\gamma-1)\frac{\phi_x}{U} + \left(\frac{\gamma+1}{2}\right)\frac{\phi_z^2}{U^2} + \left(\frac{\gamma-1}{2}\right)\left(\frac{\phi_x^2+\phi_y^2}{U^2}\right)\right]\phi_{zz} \\
&+2M_\infty^2\left[\frac{\phi_y}{U}\left(1+\frac{\phi_x}{U}\right)\phi_{xy} + \frac{\phi_z}{U}\left(1+\frac{\phi_x}{U}\right)\phi_{xz} + \frac{\phi_y\phi_z}{U^2}\phi_{yz}\right]
\end{aligned} \tag{2.14}$$

Assuming the perturbations are small relative to the freestream and that the Mach number of the freestream flow is not near unity, i.e. the flow field is entirely subsonic or supersonic, each term on the right hand side of the equation is much less than one and can be neglected. This leaves the Prandtl-Glauert equation for linearized compressible flow,

$$(1 - M_\infty^2)\phi_{xx} + \phi_{yy} + \phi_{zz} = 0 \tag{2.15}$$

For supersonic flows specifically, the negative of this equation is often used,

$$(M_\infty^2 - 1)\phi_{xx} - \phi_{yy} - \phi_{zz} = 0 \tag{2.16}$$

As discussed by Cummings et al. [13], Equation 2.15 takes the form of the wave equation in a supersonic flow field and requires that the onset flow is in the x-direction. This is a Partial Differential Equation (PDE) of hyperbolic type where the characteristic surface is the Mach cone which defines the solution space, as illustrated by Figure 2.1(a); this will be discussed further in the following chapters. For subsonic flow regimes, Equation 2.15 is a PDE of elliptic type, thus the solution space resembles Figure 2.1(b).

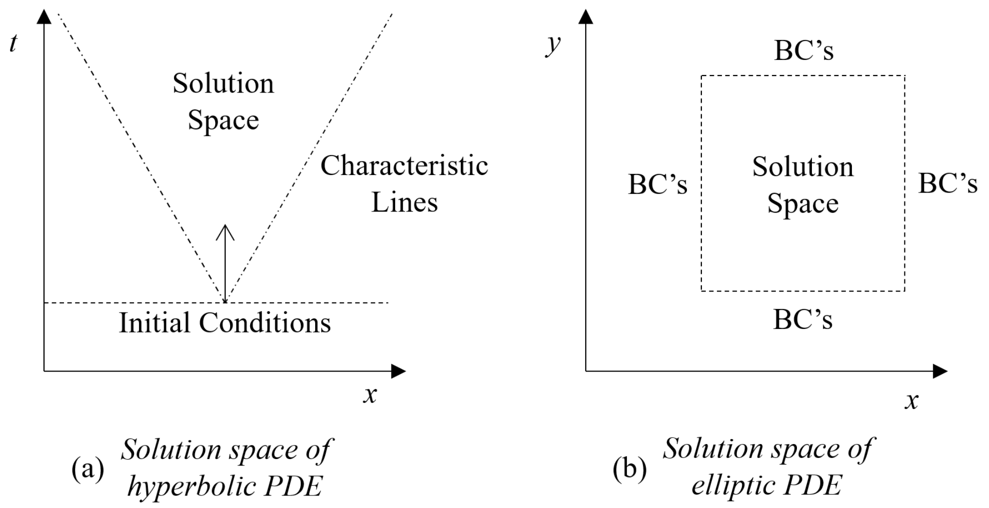


Figure 2.1: Solution spaces of different types of PDEs

2.1.1 Limitations

The key to the Prandtl-Glauert equation is that it is linearized, allowing the use of the principle of superposition to arrive at solutions. However eliminating higher order terms via the small perturbation assumption in the derivation of Equation 2.15 of course introduces limitations to its use.

All compression shocks present in a supersonic flow field must be weak such that the entropy change across any shock is small; thus, a solution to this equation with a high freestream Mach number will be in error, as well as that for a configuration with a high angle of attack or a thick body [1]. The same applies for expansion waves. The freestream Mach number also cannot be near unity, thus there can be no transonic flow, as this introduces a singularity into the solution of the Prandtl-Glauert equation that is not handled.

2.2 Boundary Integral Equation for Supersonic Flow

To compute the flow field around an arbitrary body in steady supersonic flow, the Prandtl-Glauert equation derived in the previous section, Equation 2.15, is solved analytically using an indirect formulation of the Boundary Integral Equation (BIE). This requires the integration of discrete singularity elements over a discretized surface. Each singularity element is given a strength using influence functions to satisfy boundary conditions applied on the surface.

This BIE, for a constant subsonic Mach number, can be obtained two primary ways as explained by Erickson [1]. For compressible flows, one can find the solution of the more general Prandtl-Glauert equation derived earlier using a mass-flux boundary condition applied to the true geometry. For incompressible flows, the Prandtl-Glauert transformation is used to convert Equation 2.15 to Laplaces's equation which is then solved using a velocity boundary condition [1]. The solution of the Prandtl-Glauert equation in the form of Laplace's equation, as well as its implementation into CPanel, is shown by Satterwhite [10]. To arrive at the BIE for a constant supersonic Mach number, the method of finding a solution for the general Prandtl-Glauert equation must be used.

This process for a constant supersonic Mach number is much more involved than that for its subsonic counterpart. There are fundamental differences in the characteristics of subsonic and supersonic flow fields that must be addressed, as well as complications in the mathematics of the supersonic case that do not arise in the subsonic case. The following sections go through this process, which is adapted from Ehlers et al. [2]; for a more thorough derivation and discussion, the reader is directed to their presentation of the material.

2.2.1 Preliminaries

The derivation of the boundary integral equation for supersonic flow starts with the more general case for that of compressible flows, but begins similarly to the BIE derivation for incompressible flow, which is shown by Satterwhite [10].

Two scalar functions, U_1 and U_2, and two vector functions, $\mathbf{w}_1$ and $\mathbf{w}_2$ are defined where

$$\mathbf{w} = (\beta^2 u, v, w) \tag{2.17}$$

$$\beta^2 = 1 - M^2 \tag{2.18}$$

and the subscripts 1 and 2 indicate two different solutions to Equation 2.15, which can be written in terms of $\mathbf{w}$, the linear perturbation mass flux vector, as

$$\boldsymbol{\nabla} \cdot \mathbf{w} = 0 \tag{2.19}$$

Note that when $M = 0$ in Equation 2.18, then $\mathbf{w} = \mathbf{v} = \boldsymbol{\nabla}\phi$, and so $\boldsymbol{\nabla}\cdot\mathbf{w} = \boldsymbol{\nabla}^2\phi = 0$, which is the governing equation to model incompressible flow. Both U and $\mathbf{w}$ are defined to be continuous and twice continuously differentiable within the volume of interest, $\cancel{V}$. Thus, the divergence theorem can be written in terms of these functions as

$$\int_{\cancel{V}} (U_1 \boldsymbol{\nabla} \cdot \mathbf{w}_2 + U_2 \boldsymbol{\nabla} \cdot \mathbf{w}_1) d\cancel{V} = \int_S (U_1 \mathbf{w}_2 \cdot \hat{n}) dS \tag{2.20}$$

where S is the boundary of $\forall$. According to Green's theorem, the right hand side of Equation 2.20 is equal to zero. The same holds true if the subscripts of U and $\mathbf{w}$, respectively, are switched, leading to Green's second identity,

$$\int_{\forall}(U_1\boldsymbol{\nabla}\cdot\mathbf{w}_2 - U_2\boldsymbol{\nabla}\cdot\mathbf{w}_1)d\forall = \int_S \big[(U_1\mathbf{w}_2 - U_2\mathbf{w}_1)\cdot\hat{n}\big]dS = 0 \tag{2.21}$$

The integral on the right hand side of Equation 2.21 is what needs to be solved, where U_1 and U_2 are both solutions to the Prandtl-Glauert equation. U_1 and U_2 are then defined as

$$\begin{aligned} U_1 &= \frac{1}{R} \\ U_2 &= \phi \end{aligned} \tag{2.22}$$

which are both solutions to the Prandtl-Glauert equation. Note that these functions are defined as they are by Satterwhite [10], which is the opposite of the definitions given by Ehlers et. al. [2].

To this point, this derivation applies to both subsonic and supersonic flows. Moving forward, the difference in solutions comes with the definition of R, more specifically, the sign of the Prandtl-Glauert factor squared, β^2. For subsonic flow, β^2 is positive which yields the first term of Equation 2.15 to be positive and the equation itself to be a PDE of elliptic type. For U_1 to be a solution of this PDE, R must be

$$R_\beta = \sqrt{(x-\xi_0)^2 + \beta^2\big[(y-\eta_0)^2 + (z-\zeta_0)^2\big]} \tag{2.23}$$

where

$$\mathbf{P} = (x, y, z) \tag{2.24}$$

$$\mathbf{P}_0 = (\xi_0, \eta_0, \zeta_0) \tag{2.25}$$

Here and for the following discussions, $\mathbf{P}$ is defined as the influenced point and $\mathbf{P}_0$ is the influencing point. For supersonic flow, β^2 is negative. This yields the first term of Equation 2.15 to be negative and the equation itself to be a PDE of hyperbolic type. For U_1 to be a solution in this case, R must now be

$$R_B = \sqrt{(x-\xi_0)^2 - B^2\left[(y-\eta_0)^2 + (z-\zeta_0)^2\right]} \tag{2.26}$$

where

$$B = \sqrt{M^2 - 1} \tag{2.27}$$

The key difference between the quantities R_β and R_B is the singularities they each introduce when substituted back into Equation 2.21, and integration is performed. In the subsonic case with R_β, just one singularity occurs at $\mathbf{P}_0$, which is easily handled in the integration process. For the supersonic case with R_B, there remains the singularity at $\mathbf{P}_0$, but there is now also a singularity along the surface defined by

$$x - \xi_0 = \pm Br \tag{2.28}$$

where

$$r = \sqrt{(y-\eta_0)^2 + (z-\zeta_0)^2} \tag{2.29}$$

This extra singularity creates difficulties in carrying out the integrals of Equation 2.21, the details of which will be discussed shortly.

2.2.2 Boundaries

Before continuing to develop the BIE for supersonic flow, the boundaries over which the integrals are to be evaluated must be specified. There are of course fundamental differences in the phenomena that occur in subsonic and supersonic flow fields, which have direct bearing on the formulation of the BIEs.

Figure 2.2 shows how a flow disturbance propagates in flow fields of increasing Mach number. In the case of a subsonic freestream velocity as illustrated by Figures 2.2(a) and 2.2(b), when a flow disturbance occurs at time $t = 0$, the wave front generated by this disturbance propagates throughout the flow field at the speed of

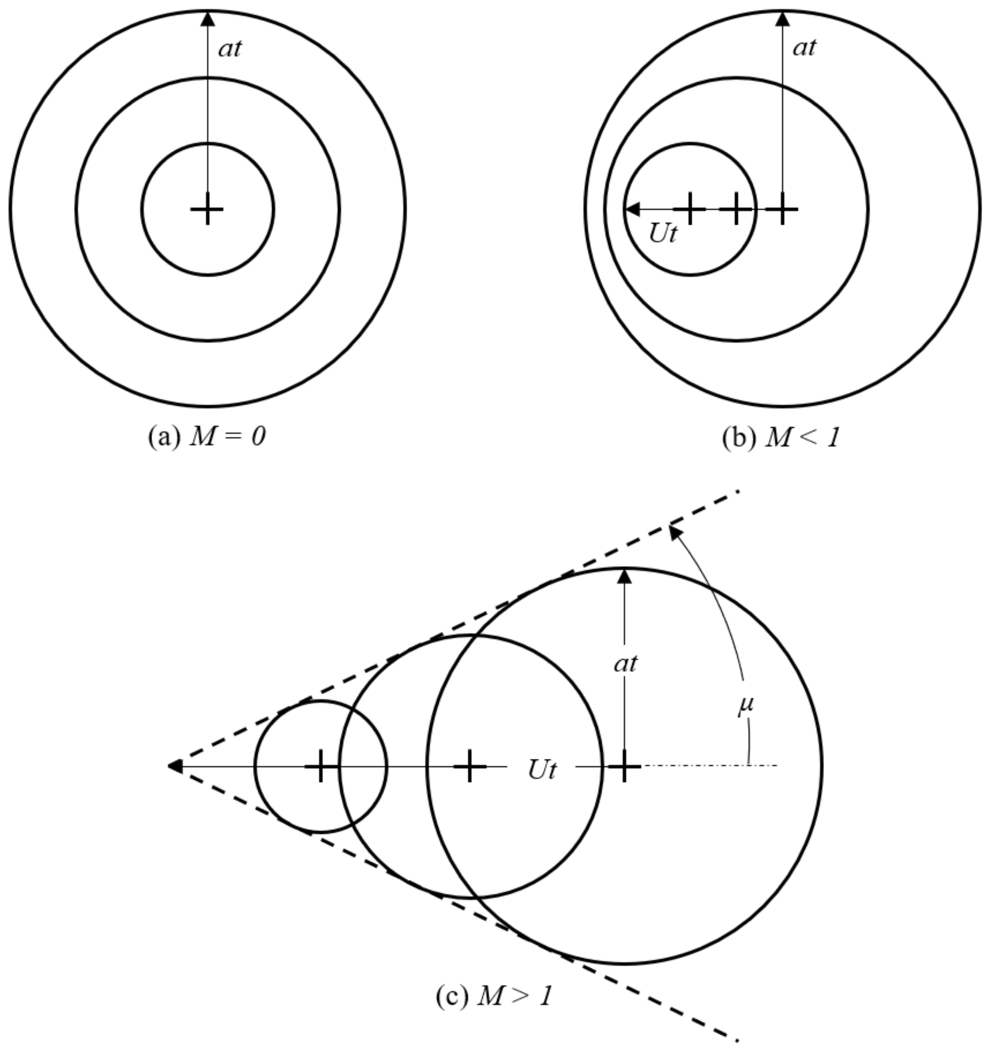


Figure 2.2: Flow disturbance propagation with varying Mach number

sound, a; the disturbance is also convected by the freestream flow. As time approaches infinity, this disturbance will fill the entirety of the flow domain. In the case of a supersonic freestream velocity as depicted by Figure 2.2(c), the flow disturbance still propagates throughout the flow field at the speed of sound, but it is now also convected at a speed greater than the speed of sound. Thus, at a time $t > 0$, the disturbance will have traveled further due to the freestream than it will have traveled due to its propagation at the speed of sound. What results is a conical region that defines the envelope in which the flow disturbance propagates, called the Mach cone.

The sine of the cone half angle, μ, yields the ratio of speed of sound to freestream velocity, which is the reciprocal of Mach number.

There is both an upstream Mach cone and a downstream Mach cone for a given point $\mathbf{P}_0$, as illustrated in Figure 2.3, and the surfaces of which are defined by the two forms of Equation 2.28. The upstream Mach cone, also called the Domain of Dependence (DOD), is the region that influences the point $\mathbf{P}_0$. Similarly, the downstream Mach cone, called the Domain of Influence (DOI), is the region that is influenced by the point $\mathbf{P}_0$. Any flow disturbance within the domain of dependence will influence this point, and the state of the flow at this point has influence on the volume enclosed by the domain of influence.

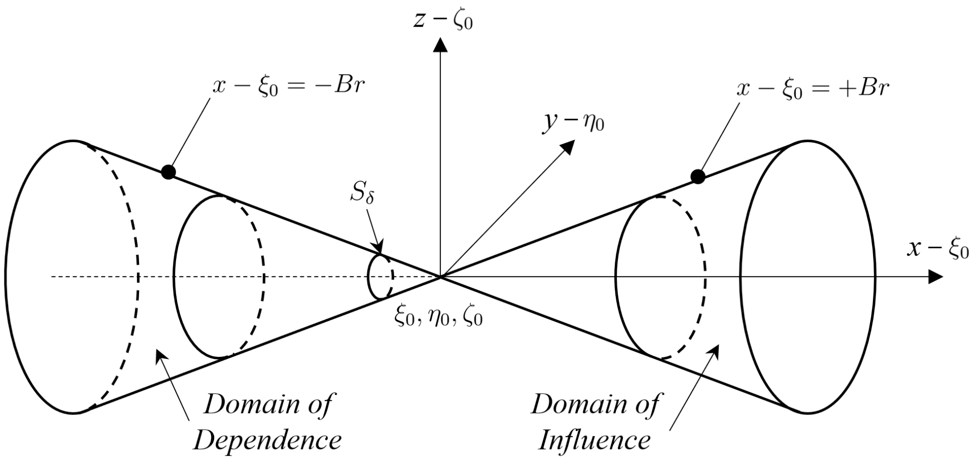


Figure 2.3: Definition of Domain of Dependence and Domain of Influence

Going back to equation 2.21 and the derivation of the supersonic BIE, it can now be seen from Figure 2.3 that the surface over which integration is to be performed to solve for the flow state at $\mathbf{P}$ as influenced by $\mathbf{P}_0$ is the surface of the DOD, any surfaces within this region that may cause perturbations in the flow field, and the surface S_δ as it approaches the point $\mathbf{P}_0$.

2.2.3 The Supersonic BIE

With the BIE defined in the form of Green's second identity and the boundaries defined, integration can be performed to solve the BIE. First, the singularity that occurs on the Mach cone boundaries must be handled. Simply put, this is done by taking the finite part of the divergent integral that arises in integrating U_1 over the Mach cone boundary. As explained by Ehlers et al. [2], this was first shown by Hadamard [14] and subsequently used by many others. The details of this formulation are not discussed here; the reader is directed to Ehlers et al. [2] for discussion and derivation of this concept.

Now noting that ϕ is constant along any characteristic surface and that the finite part of the divergent integral cancels over the upstream Mach cone boundary, any integration that is performed over the Mach cone boundary will cancel [2]. This leaves the surface S_δ, and any surfaces that may be within the Mach cone boundaries which cause flow perturbations, to be defined here as S_a where $\mathbf{P}_0$ is some point on S_a.

Substituting Equation 2.22 into the right hand side of Equation 2.21 and using R_B in place of R, the BIE can now be written as

$$\int^{*}_{S_a+S_\delta} \left[\left(\frac{1}{R_B} \mathbf{w}_2(\mathbf{P}_0) - \phi(\mathbf{P}_0)\mathbf{w}_1 \right) \cdot \hat{n} \right] dS = 0 \tag{2.30}$$

where the asterisk denotes the finite part of the integral. First, the integration over S_δ alone is addressed. Taking the limit as the surface S_δ approaches zero where the unit normal of this surface points upstream away from $\mathbf{P}_0$, it is found that

$$\lim_{S_\delta \to 0} \int_{S_\delta} \left[\left(\frac{1}{R_B} \mathbf{w}_2(\mathbf{P}_0) - \phi(\mathbf{P}_0)\mathbf{w}_1 \right) \cdot \hat{n} \right] dS = 2\pi(x - \xi_0)\mathbf{w}_1 \cdot \hat{n}/B - 2\pi\phi(\mathbf{P}) \tag{2.31}$$

This is a convergent integral so the asterisk is removed here. Noting that the term $(x - \xi_0)$ goes to zero as S_δ approaches $\mathbf{P}_0$ in Figure 2.3, substituting Equation 2.31

back into Equation 2.30 and rearranging yields

$$\phi(\mathbf{P}) = \frac{1}{2\pi}\int_{S_a}^{*}\left[\Big(\frac{1}{R_B}\mathbf{w}_2(\mathbf{P}_0) - \phi(\mathbf{P}_0)\mathbf{w}_1\Big)\cdot\hat{n}\right]dS \tag{2.32}$$

It is known that

$$\mathbf{w}_1\cdot\hat{n} = \hat{n}\cdot\bigg(-B^2\frac{\partial}{\partial\xi_0},\frac{\partial}{\partial\eta_0},\frac{\partial}{\partial\zeta_0}\bigg)\bigg(\frac{1}{R_B}\bigg) = \frac{B^2(\mathbf{P}_0-\mathbf{P})\cdot\hat{n}}{(R_B)^3} \tag{2.33}$$

and redefining $\mathbf{w}_2$ as

$$\mathbf{w}_2 = \mathbf{w} \tag{2.34}$$

Equation 2.32 becomes

$$\phi(\mathbf{P}) = \frac{1}{2\pi}\int_{S_a}\frac{\mathbf{w}(\mathbf{P}_0)\cdot\hat{n}}{R_B}dS - \frac{B^2}{2\pi}\int_{S_a}^{*}\phi(\mathbf{P_0})\frac{(\mathbf{P_0}-\mathbf{P})\cdot\hat{n}}{(R_B)^3}dS \tag{2.35}$$

The right hand side of this equation can be rewritten in terms of the conormal giving

$$\phi(\mathbf{P}) = \frac{1}{2\pi}\int_{S_a}\frac{\mathbf{w}(\mathbf{P}_0)\cdot\hat{n}}{R_B}dS - \frac{1}{2\pi}\int_{S_a}^{*}\phi(\mathbf{P}_0)\frac{\partial}{\partial\bar{n}_c}\left(\frac{1}{R_B}\right)dS \tag{2.36}$$

where $\hat{n}$ is the unit normal pointing away from the surface into the flow and $\bar{n}_c$ is the conormal vector defined as, respectively,

$$\begin{aligned}\hat{n} &= (n_x, n_y, n_z)\\ \bar{n}_c &= (-B^2 n_x, n_y, n_z)\end{aligned} \tag{2.37}$$

Note the similarities between Equation 2.36 and its subsonic counterpart given by Satterwhite [10], primarily in the general form of the equation and a coefficient that is a factor of $\frac{1}{c\pi}$ where c is some positive integer.

At this point in the derivation of the BIE for supersonic flow, a distinction must be made between what are referred to as superinclined and subinclined surfaces. A subinclined surface is that which is inclined at an angle less than the Mach angle, and a superinclined surface is that which is inclined at an angle greater than the Mach angle. Examples of superinclined surfaces are shown in Figure 2.4.

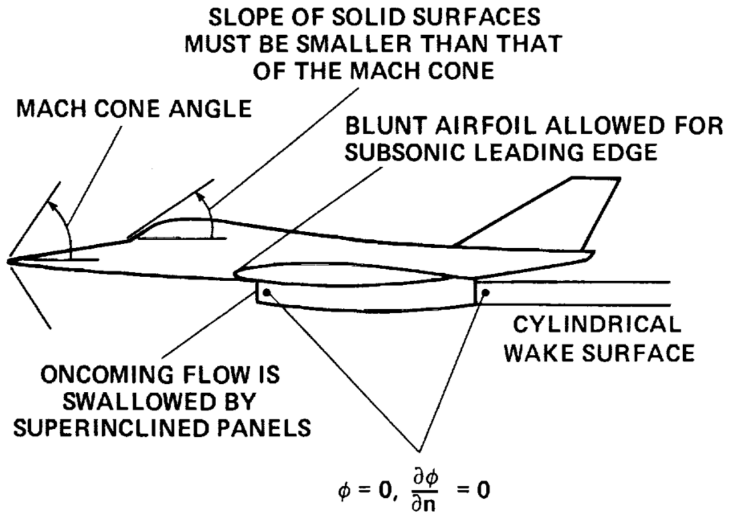

SLOPE OF SOLID SURFACES MUST BE SMALLER THAN THAT OF THE MACH CONE
MACH CONE ANGLE
BLUNT AIRFOIL ALLOWED FOR SUBSONIC LEADING EDGE
CYLINDRICAL WAKE SURFACE
ONCOMING FLOW IS SWALLOWED BY SUPERINCLINED PANELS
$\phi = 0, \frac{\partial \phi}{\partial n} = 0$

Figure 2.4: Examples of superinclined surfaces [1]

Superinclined surfaces can be used to model various flows such as that through a nacelle, or the exhaust from an engine by applying specific boundary conditions to the downstream side of superinclined surfaces, and they can only be applied to the downstream side. Boundary conditions for subinclined surfaces, however, must be applied to both sides of the surface to give a well-posed problem. This difference in boundary condition application is why a distinction in the BIE derivation is needed. Superinclined surfaces behave as subinclined in the special case of a subsonic leading edge, meaning it is swept such that the entire edge is within the Mach cone emanating from the root of the leading edge, as identified in Figure 2.4. The ability to apply boundary conditions to the downstream side of superinclined surfaces as mentioned above is not implemented with this work, so the remainder of the BIE derivation is specifically for subinclined panels.

As Equation 2.35 is currently written, it only accounts for integration over the upper surface of S_a, or the side that is submerged in the flow; the lower surface, the surface which is not exposed to the flow, must also be accounted for in the integration over a subinclined surface. Similar to what is done by Satterwhite [10] in deriving the subsonic implementation of CPanel, an inner potential, ϕ_i, is now defined at an interior point, $\mathbf{P}_i$. Since this point is outside of the flow domain, the potential here is zero, and the normal is pointing in the opposite direction. Writing Equation 2.32 for this point gives

$$0 = -\frac{1}{2\pi}\int_{S_a}\frac{\mathbf{w}_i(\mathbf{P}_0)\cdot\hat{n}}{R_B}dS - \frac{1}{2\pi}\int_{S_a}^{*}\phi_i(\mathbf{P}_0)\frac{\partial}{\partial\bar{n}_c}\left(\frac{1}{R_B}\right)dS \tag{2.38}$$

Finally, adding this to Equation 2.35 yields the complete BIE,

$$\phi(\mathbf{P}) = \frac{1}{2\pi}\int_{S_a}\frac{[\mathbf{w}(\mathbf{P}_0)-\mathbf{w}_i(\mathbf{P}_0)]\cdot\hat{n}}{R_B}dS - \frac{1}{2\pi}\int_{S_a}^{*}[\phi(\mathbf{P}_0)-\phi_i(\mathbf{P}_0)]\frac{\partial}{\partial\bar{n}_c}\left(\frac{1}{R_B}\right)dS \tag{2.39}$$

2.3 Singularity Elements

With the BIE for supersonic flow now defined, it must be written in terms of fundamental singularity elements. The perturbation mass flux vectors and the perturbation velocity potentials of Equation 2.39 can be expressed in terms of two useful fundamental singularity elements. As explained by Ehlers et. al., it can be seen from Equation 2.39 that the discontinuity in perturbation mass flux with a continuous perturbation potential across the surface produces sources of strength

$$\sigma = \mathbf{w}(\mathbf{P}_0)\cdot\hat{n} - \mathbf{w}_i(\mathbf{P}_0)\cdot\hat{n} \tag{2.40}$$

Furthermore, a discontinuity in velocity potential with continuous normal mass flux across a surface produces doublets of strength

$$\mu = \phi(\mathbf{P}_0) - \phi_i(\mathbf{P}_0) \tag{2.41}$$

Substituting Equations 2.40 and 2.41 into Equation 2.39 then gives the final form of the BIE for supersonic flow in terms of source and doublet strengths as

$$\phi(\mathbf{P}) = \frac{1}{2\pi}\int_{S_a}\frac{\sigma}{R_B}dS - \frac{1}{2\pi}\int_{S_a}^{*}\mu\frac{\partial}{\partial\bar{n}_c}\left(\frac{1}{R_B}\right)dS \tag{2.42}$$

The two integrals of this equation each represent the potential influence of a distribution of sources over a surface and the distribution of doublets over a surface, respectively, which can then be summed to find the perturbation potential at a point in the flow field. Before doing this, however, a few more items must be handled.

First is setting the order of the singularity strengths and determining the order of the discretized elements over which integration is to be performed. CPanel utilizes an unstructured discretization of flat triangular panels, so the order of the elements used in integration is predetermined for this implementation. The original implementation of CPanel uses a distribution of constant sources and constant doublets. This decision was made in part because of the use of unstructured paneling, and the ability to quickly generate high fidelity surface meshes while maintaining computational efficiency through the use of constant singularity elements. As discussed in the previous chapter, for supersonic modeling via panel methods, the order of the doublet elements must be at least one order higher than constant to enable the enforcement of doublet strength continuity across the body surface. The order of the doublet distribution that was eventually chosen and implemented will be discussed in Chapter 4.

Independent of the order of the singularity elements and discretized elements, boundary conditions must be applied to the surface to ensure a unique solution can be obtained.

2.4 Boundary Conditions

As discussed by Satterwhite [10], there are three different boundary conditions that must be considered: far field boundary conditions, boundary conditions on the body in the flow field, and the Kutta condition to ensure the smooth transition of flow over sharp trailing edges. Since in supersonic flow fields, the boundary of the flow field is defined by the Mach cone, the far field condition was already handled in the BIE derivation. For boundary conditions of the body, there are two different approaches that may be used: application of the Neumann or Dirichlet boundary conditions.

The Neumann approach involves the direct implementation of boundary conditions, as conditions are applied directly to the surface of the body. This type of condition is also sometimes referred to as a velocity boundary condition. The other approach is to use a Dirichlet type condition, which is applied in an indirect manner and involves the specification of the function value itself on the surface [10]. This is the type that is currently used in CPanel and will also be utilized for this implementation. It was specified earlier in the development of the BIE that the potential internal to the body and outside the flow domain, ϕ_i, is zero. Generally, this potential simply needs to be constant such that it has no contribution to computed perturbation velocities. However for supersonic flow, the internal potential must be zero to prevent spurious vortices from propagating down Mach lines and introducing non-physical perturbations on the upper surface of the body [2]. Applying this condition to Equation 2.42 gives

$$\frac{1}{2\pi}\int_{S_a}\frac{\sigma}{R_B}dS - \frac{1}{2\pi}\int_{S_a}^{*}\mu\frac{\partial}{\partial n_c}\left(\frac{1}{R_B}\right)dS = 0 \tag{2.43}$$

which can be written numerically as

$$\frac{1}{2\pi}\left[\sum_{j=1}^{N_s}\frac{1}{R_B}\sigma - \sum_{i=1}^{N_d}\mu\frac{\partial}{\partial n_c}\left(\frac{1}{R_B}\right)\right] = 0 \tag{2.44}$$

At this point, Equation 2.44 represents a linear system of equations with at least $2N$ unknowns and N equations, where the unknowns are the source and doublet strengths. Thus the condition that no mass flux may cross the boundary of the body in the flow field is applied.

$$\mathbf{w}\cdot\hat{n} = -\mathbf{U}\cdot\hat{n} \tag{2.45}$$

This in turn sets the strengths of the sources distributed over the surface as

$$\sigma = -\mathbf{U}\cdot\hat{n} \tag{2.46}$$

leaving only the doublet strengths distributed over the body as the unknowns, and ensuring a unique solution can be obtained. This type of condition is also often referred to as a mass flux boundary condition or Morino's scheme [15].

The final condition to be addressed is the Kutta condition. In subsonic lifting flows, this must always be applied to ensure the smooth transition of flow over sharp trailing edges, and that the rear stagnation point remains at the trailing edge. With supersonic lifting flows, the application of the Kutta condition is needed only in the case of so-called subsonic trailing edges–where the trailing edge is swept such that the entire edge is within the Mach cone emanating from the root of the edge. When a lifting surface in supersonic flow has a so-called supersonic trailing edge, no part of the surface will be within the downstream Mach cone of the wake, so the wake has no influence on the body and the Kutta condition does not need to be applied. This concept is illustrated in Figure 2.5. For this thesis, the ability to model wakes in supersonic flow was not implemented, so only wings with supersonic trailing edges can be modeled accurately and the Kutta condition does not need to be applied.

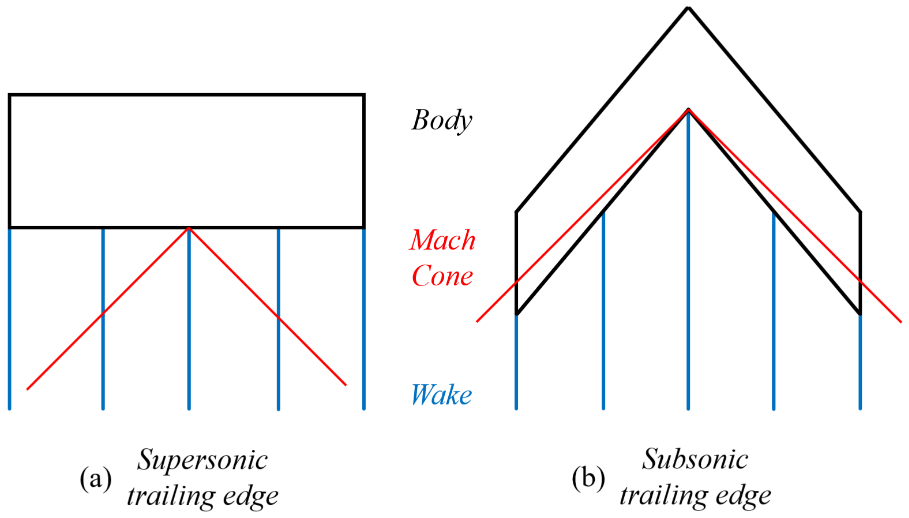


Figure 2.5: Influence of wake on body in supersonic flow

Chapter 3

GENERAL NUMERICAL METHODS

The following chapter outlines the methods by which generic BIEs are solved, as well as the one specific to this implementation. The different methods and approaches to computing influence coefficients and velocities in other supersonic panel codes are discussed. Lastly, the various formulations of the pressure coefficient for compressible flows is presented. Parallels and differences between the original subsonic CPanel implementation and this implementation will again be noted throughout the chapter.

3.1 Linear System of Equations

The majority of potential-based panel codes, regardless of the order of singularity elements or the order of the surface discretization, utilize the same general process of building and solving a linear system of equations. The crux of this process, and of panel codes themselves, is the construction of the influence coefficient matrices where each coefficient is found by computing the potential influence of a unit strength element onto a point in the solution domain. A source influence coefficient matrix and a doublet influence coefficient matrix are both built by solving the two integrals of a BIE with unit strength singularities. In this implementation, the expression of the supersonic BIE given in Equation 2.44 can be written as a linear system of equations as

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{B}\boldsymbol{\sigma} \tag{3.1}$$

In the subsonic implementation of CPanel, there remained a far field potential term and an internal potential in this expression. As discussed in Chapter 2, the far field term does not arise here since the influence domain in which integration is performed is restricted to the Mach cone, and the internal potential is set to zero, which leaves only the terms in the expression above.

Since the source strengths are prescribed from Equation 2.46, and influence coefficient matrices are determined from geometry and Mach number, the right hand side of Equation 3.1 is known and **A** is known, just leaving the system

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{RHS} \tag{3.2}$$

to be solved for the vector of doublet strengths, $\boldsymbol{\mu}$. Though this generic method of solving for the unknown doublet strengths does not depend on the order of the method employed, the construction and composition of the influence coefficient matrices does in fact depend on method order.

3.2 Influence Coefficients

In CPanel v1.0, constant doublet and constant source distributions are implemented. This same methodology is adopted in v2.0. For subsonic schemes and with modern computational speeds, this is arguably the best approach since what constant singularity methods used to lack in accuracy can now be made up by using finely discretized, automatically generated meshes [10]. Using constant doublets and constant sources results in greatly simplified solutions to the BIEs which in turn simplifies the construction of the influence coefficient matrices. The same control points can be used for both singularities, and the dimensions of the two matrices, which will both be square, will always be the same as one another. However if used with an unstructured mesh as is done in CPanel, one can not simply use finite differencing to compute velocity from velocity potential, and must use a more involved method such as CHTLS [16].

Clearly, this method cannot be used in implementing a supersonic scheme into

CPanel as doublet continuity cannot be enforced with a constant doublet distribution. Since PANAIR and its pilot code were first published in the late 1970s and early 1980s [5, 2], the majority of potential-based supersonic panel codes have adopted the same methodology, which was first applied to subsonic flows by Johnson and Rubbert [17]. This technique involves using a quadratic approximation for curved panels, quadratically varying doublets, and linearly varying sources. The curved panels are then broken up into eight flat, triangular subpanels like that shown in Figure 3.1. A splining method is used to describe the singularity distributions over each panel, then the subpanels are integrated over and summed to find the influence coefficients for a given panel [17]. This method first applied to modeling supersonic flow by Ehlers et al. [2], when it was then discovered that all panel edges must be contiguous and that a more strict splining method was needed to enforce doublet strength continuity, which then led to the development of PANAIR.

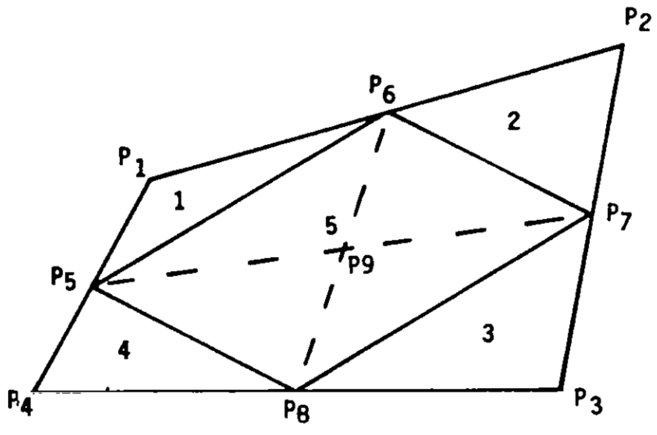


Figure 3.1: Subpanel definition used in other supersonic panel codes [2]

As noted above, most potential-based supersonic panel methods have adopted this methodology. One exception is MARCAP [7], which uses linearly varying doublets and constant sources on flat, triangular panels. The advantages and disadvantages of these different approaches will be discussed in the next chapter. Regardless of the order of singularities used in modeling supersonic potential flows, there remains the method by which the influence coefficients themselves are computed.

As discussed by Epton and Magnus [18], no matter the influence coefficient calculation method, all will yield the same results in computing the influence of a singularity element on a point, given the same problem. The approach adopted by PANAIR [18] is to establish a cylindrical coordinate system to be used for subsonic flow problems or with superinclined panels in supersonic flow, and a hyperbolic coordinate system for use with subinclined panels in supersonic flow. In working in these disparate coordinate systems, the developer can manipulate the solution to the BIE specific to the problem at hand, and simplify the implementation as well as the computations themselves. This was adopted by most supersonic panel codes following PANAIR. Another approach implemented by Ehlers et al. [2], based on the methods of Johnson [19], is to develop a small set of fundamental integrals which only need to be computed once per influence calculation; then the remaining computations are all performed in terms of these fundamental integrals. This is done in rectilinear coordinates.

Most of the intricacies of supersonic panel methods relative to subsonic methods lie in the influence coefficient calculations, as can be seen in the variation and complexities of the methods mentioned above. Thus the majority of the work for this thesis revolved around the understanding, development, and testing of the influence coefficient computations, as well as how to best implement the required changes and additions to the existing CPanel architecture. Chapter 4 focuses on these specific implementations, so the detailed discussion on the topic is reserved for said chapter.

3.3 Post Processing

The power of potential-based panel methods lies in the definition of velocity potential itself stated by Equation 2.6. Once velocity potential all over a body is known, which is easily calculated directly from the defined source strengths and computed doublet strengths, one simply needs to take the gradient of the potential to find velocity. And once velocity is known, the pressure coefficients all over the body can be found, which then leads to the ability to compute force and moment coefficients, giving a complete aerodynamic profile for the body of interest.

Because all supersonic panel codes, since PANAIR at least, are higher order methods, and whether or not they use a structured or unstructured surface discretization, the gradient of the potential can be taken without the need for more advanced methods such as using vortex filaments or methods like CHTLS [10]. This is a relatively trivial process and essentially the same approach has been used by all published codes.

3.3.1 Velocity

In a general panel method regardless of the flow regime or boundary conditions that are used, the velocity on the upper surface of a body (the side that is exposed to the flow field) and the lower surface of a body (the side that is not in the flow field, internal to the body), can be computed. This then yields itself to a definition of average velocity written as

$$\mathbf{v}_A = (\mathbf{v}_U + \mathbf{v}_L)/2 \tag{3.3}$$

and a difference velocity written as

$$\mathbf{v}_D = \mathbf{v}_U - \mathbf{v}_L \tag{3.4}$$

Now the upper and lower velocities, respectively, can be written in terms of the average and difference velocities as

$$\mathbf{v}_U = \mathbf{v}_A + \mathbf{v}_D/2 \tag{3.5}$$

$$\mathbf{v}_L = \mathbf{v}_A - \mathbf{v}_D/2 \tag{3.6}$$

The average and difference velocities can be split into tangential and normal components, but a distinction must first be made for the calculation of these quantities for use in subsonic or supersonic flow regimes. In the subsonic case as shown by Johnson [19], $\mathbf{v}_A$ and $\mathbf{v}_D$ are found to be, respectively,

$$\mathbf{v}_A = \boldsymbol{\nabla}\phi_A + \sigma\hat{n} \tag{3.7}$$

$$\mathbf{v}_D = \boldsymbol{\nabla}\mu + \sigma\hat{n} \tag{3.8}$$

For the supersonic case, both of these quantities are also functions of the conormal defined in Equation 2.37, and the normal component of $\mathbf{v}_A$ is found via the average perturbation mass flux rather than from source strength. As expressed by Epton and Magnus [18], they are then written as

$$\mathbf{v}_A = \boldsymbol{\nabla}\phi_A + \frac{\mathbf{w}_A \cdot \hat{n}}{\hat{n} \cdot \bar{n}_c}\hat{n} \tag{3.9}$$

$$\mathbf{v}_D = \boldsymbol{\nabla}\mu + \frac{\sigma}{\hat{n} \cdot \bar{n}_c}\hat{n} \tag{3.10}$$

As shown in Equations 3.8 and 3.10, the difference velocity is purely a function of the singularity strengths, and so can be computed as is without the need for any more information. Prior to computing the average velocity however, the applied boundary conditions must be considered since it is a function of potential, as well as perturbation mass flux in the supersonic case. If the boundary conditions of this implementation are applied, that is

$$\begin{aligned} \phi_L &= 0 \\ \sigma &= -U \cdot \hat{n} \end{aligned} \tag{3.11}$$

then it can be shown that for both the subsonic and supersonic cases,

$$\phi_A = \mu/2 \tag{3.12}$$

and for the supersonic case specifically,

$$\mathbf{w}_A \cdot \hat{n} = \sigma/2 \tag{3.13}$$

Thus, like the difference velocity, the average velocity can be computed directly from the singularity strengths. The same can be done if the normal perturbation mass flux is specified as some non-zero quantity.

If a different type of boundary condition is applied or the internal potential is non-zero, then $\mathbf{v}_A$ cannot be reduced to singularity strengths, and it must be computed using influence coefficients similar to how the influence coefficient matrix is constructed. Rather than computing the coefficients from the solution to the BIE however, they are computed from the gradient of the BIE which can be found analytically.

3.3.2 Pressure Coefficient

With velocities computed, the pressure coefficients across the surface of the body in the flow field can be found, from which other aerodynamic coefficients of interest easily follow. The pressure coefficient for compressible flow can be expressed in various ways depending on the applied simplifying assumptions. The following equations are adapted from Epton and Magnus [18].

The theoretically exact pressure coefficient expression, called the isentropic pressure coefficient formula, is written as

$$C_p = \frac{2}{\gamma M_\infty^2}\left[1 + \frac{\gamma - 1}{2}\left(1 - \frac{|\mathbf{V}|^2}{|\mathbf{U}|^2}\right)M_\infty^2\right]^{\frac{\gamma}{\gamma-1}} - 1 \tag{3.14}$$

This equation contains terms that are multiple orders higher than the linearized equations used to compute the velocities discussed earlier. Simplifying assumptions can be applied to arrive at expressions for C_p that are more appropriate for the circumstances.

Neglecting all third-order and higher terms gives the second-order pressure coefficient formula.

$$C_p = -\frac{2u}{|\mathbf{U}|} - \frac{(1 - M_\infty^2)u^2 + v^2 + w^2}{|\mathbf{U}|^2} \tag{3.15}$$

It is worth noting that as M_∞ approaches zero in Equations 3.14 and 3.15, the expression for C_p becomes that for incompressible flow, which is more easily seen by rewriting Equation 3.15 as

$$C_p = 1 - \frac{|\mathbf{V}|^2}{|\mathbf{U}|^2} + M_\infty^2 \frac{u^2}{|\mathbf{U}|^2} \tag{3.16}$$

To simplify the expression for C_p even further, the slender body assumption may be made which eliminates the quadratic expression of u yielding

$$C_p = -\frac{2u}{|\mathbf{U}|} - \frac{v^2 + w^2}{|\mathbf{U}|^2} \tag{3.17}$$

If all terms of a higher order than linear are neglected, this then leaves the linear pressure coefficient formula.

$$C_p = -\frac{2u}{|\mathbf{U}|} \tag{3.18}$$

These different expressions of C_p vary in correctness depending on the conditions of the simulation; this will be discussed further in Chapter 5. It should be noted here that these three methods of computing the pressure coefficient will be referred to as the 2^{nd} order, Slender, and Linear approximations in future discussions. Once the pressure coefficient has been found, any manner of methods can be used to compute other aerodynamic coefficients and quantities of interest.

Chapter 4

CPANEL IMPLEMENTATION

This work was implemented in two distinct stages. First, after developing a higher-order doublet scheme to enforce doublet strength continuity, a higher-order subsonic method was implemented with the goal of validating the proper functionality of the new doublet scheme; this had the added benefit of providing another method by which subsonic simulations may be carried out in CPanel. Once the higher-order subsonic method was validated, the supersonic implementation began using the same higher-order doublet scheme. Throughout the process of both implementation stages, extensive unit testing was performed to ensure each individual component was working properly prior to bringing them together for final implementation and testing. This and the more general development methodology of this work is discussed in Appendix D.

Chapter 4 presents the new CPanel functional flow diagram reflecting the new options the user has in using CPanel. A brief overview of the subsonic implementation is given, with a focus on the aspects relevant to both implementations. The supersonic implementation is then presented in detail.

4.1 Functional Flow Diagram

With the implementation of the higher-order subsonic scheme and the supersonic scheme, the user now has an expanded range of options to choose from in running CPanel. Similar to how the vortex particle implementation of CPanel was done [11], these new schemes were implemented in such a way that preserved the original code architecture, flow, and functionality wherever possible; it was also ensured that the inherent expandability of CPanel designed into its original development was preserved [10]. The new functional flow diagram of CPanel is depicted in Figure 4.1. It should be noted that the vortex particle wake option is not included in this diagram for brevity, and because this scheme cannot be used in conjunction with the new higher-order schemes as of this work, though this is something that could be pursued in the future.

Just as extra options were added to the CPanel input file format for the vortex particle wake implementation, an extra option has been added to control whether the original or the new higher-order scheme is to be used for subsonic simulations. The option for running a subsonic or supersonic simulation is simply controlled through the definition of Mach number. Further specifics of the new input file format are given in Appendix E.

Walking through the diagram of Figure 4.1, the first fork in the code occurs in determining whether or not a higher-order method is to be used. If not, then the original lower-order method is used, in which case a vortex particle wake and other associated options may be utilized. If one of the two higher-order methods is chosen, then new control points must be computed. The details and rationale behind this are discussed in the following sections. Next, the input freestream Mach number is checked and depending on whether it is subsonic or supersonic, the corresponding path is taken. The specifics of these two paths are the primary topic of this chapter.

4.2 Higher-Order Doublet Scheme

As discussed previously, it was found in early codes that to achieve a stable solution for supersonic flow fields using panel methods, enforcement of doublet strength continuity across the surface of the body must be implemented. This requires a higher-order doublet representation, where higher-order refers to any order higher than constant[1].

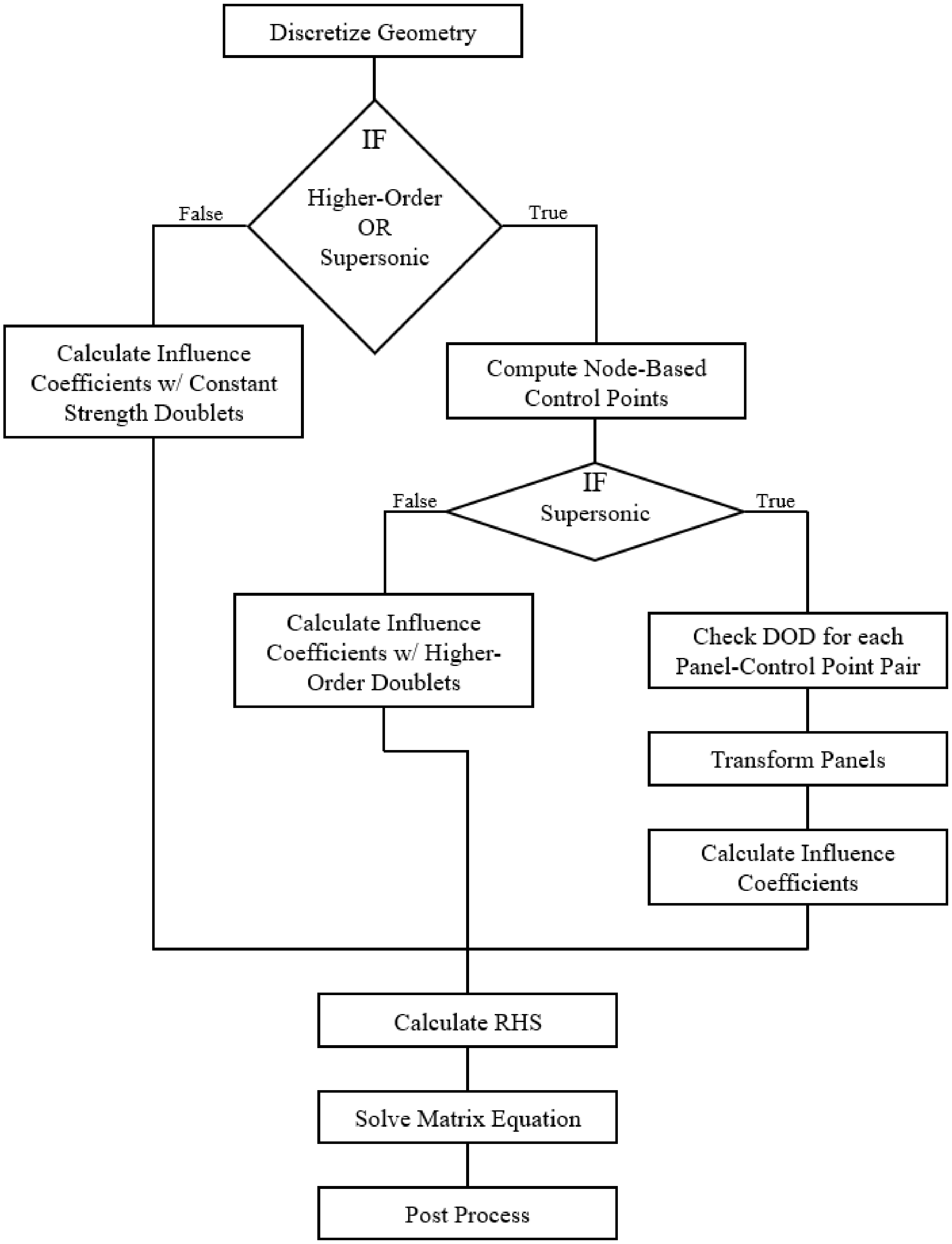

Discretize Geometry
IF
Higher-Order
OR
Supersonic
False
True
Calculate Influence Coefficients w/ Constant Strength Doublets
Compute Node-Based Control Points
IF
Supersonic
False
True
Calculate Influence Coefficients w/ Higher-Order Doublets
Check DOD for each Panel-Control Point Pair
Transform Panels
Calculate Influence Coefficients
Calculate RHS
Solve Matrix Equation
Post Process

Figure 4.1: CPanel v3.0 functional flow diagram

Of the supersonic codes that enforce doublet strength continuity, most have used quadratically varying doublets with curved panels, all for various reasons.

In general, there are a few clear advantages to doing this. First, it provides a more accurate representation of the body of interest with less panels. As a result, a relatively coarse geometry can be used while still achieving amply accurate results. Furthermore, it allows for the implementation of boundary layer models since the solved velocities vary linearly across the body [6]. The primary downside of using quadratically varying doublets is the complexity introduced in the solution to the BIEs, which in turn results in substantially longer computation times relative to using a lower-order doublet distribution.

If applied to the same surface discretization, a linear doublet implementation will generally give less accurate results than a quadratic doublet implementation, though the linear doublet implementation will have a quicker computation time. However using an unstructured discretization provides the same advantage as a quadratic doublet implementation in giving an accurate representation of the body, but also with the benefit of minimal work required by the user. Furthermore if linearly varying doublets are implemented with an unstructured surface discretization as is used in CPanel, and a finer mesh is utilized than would be with quadratic doublets, then accurate results can be obtained with both little work up front by the user, and with reduced computation time. For these reasons, a linearly varying doublet distribution was chosen for this work.

In regard to the order of the source distribution, it can been shown that there should only be a one order difference between the doublet and source distributions for codes that utilize the superposition of both, such as CPanel. This is due to the dependence of the velocity discontinuity between the upper and lower surfaces of a body on the doublet gradient and source strength [18], the expressions for which were shown in Chapter 3. So a constant strength source distribution was chosen for this implementation.

As discussed by Satterwhite [10], one of the primary downsides of using unstructured meshes is the inability to control the orientation of the panels relative to the flow direction. Panels that are aligned with the local velocity vector fields will yield less numerical error than randomly generated panels. Specifically in the case of modeling supersonic flows, one could use a structured mesh to deliberately align panels with the anticipated shocks in the flow field which would result in a more accurate solution in regions near these shocks. However as noted, performing such an exercise requires substantial work up front by the user. Utilizing modern computational speed in conjunction with a properly fine unstructured mesh is shown to be a satisfactory alternative.

To enforce doublet continuity, a method similar to that used by MARCAP [7] is implemented in CPanel. Control points are placed just below nodal points, as illustrated in Figure 4.2. Nodal points are the points that make up the vertices of the triangular panels, and control points are the points at which boundary conditions are applied. If the doublet strengths at the panel nodal points are solved for, as opposed to at the center of the panel as is most commonly done, then it is ensured that there will be continuity in doublet strength across the body surface. Since adjacent panels share edges, nodes, and control points, they also share the same linear variation in

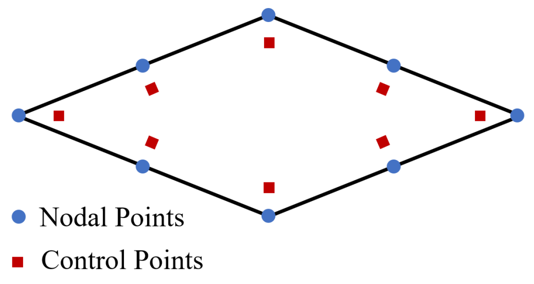


Figure 4.2: Example of nodal and control points

doublet strength along their edges, ensuring doublet strength continuity across the entire body.

The following demonstrates the method by which doublet strength continuity is enforced. The panel local coordinate system is defined in (ξ, η, ζ) coordinates with the origin at the center of a given panel, and where the panel is in the $\zeta = 0$ plane. Thus the equation for the linear doublet on a panel is

$$\mu(\xi, \eta) = \mu_0 + \mu_\xi \xi + \mu_\eta \eta \tag{4.1}$$

The potential influence of a panel with a linear doublet distribution on a point, using a form of the equation presented by Johnson [19], is found by computing

$$\phi(\mathbf{P}) = \mu(x, y) I_0 + \mu_x(x, y) I_1 + \mu_y(x, y) I_2 \tag{4.2}$$

where

$$\begin{aligned} \mu(x, y) &= \mu_0 + \mu_x x + \mu_y y \\ \mu_x(x, y) &= \mu_\xi \\ \mu_y(x, y) &= \mu_\eta \end{aligned} \tag{4.3}$$

The I terms of Equation 4.2 are the three coefficients found by solving the BIE with a unit strength linear doublet and unit strength constant source, which can be done with the BIE for any flow regime. Thus the I's are functions of panel geometry, the location of the point $\mathbf{P}$ relative to the panel, and Mach number in the case of compressible flows. The processes by which these terms are found for their use in CPanel are given in Appendix A. Substituting Equation 4.3 into Equation 4.2 yields

$$\phi(\mathbf{P}) = (\mu_0 + \mu_\xi x + \mu_\eta y) I_0 + \mu_\xi I_1 + \mu_\eta I_2 \tag{4.4}$$

Collecting terms,

$$\phi(\mathbf{P}) = \mu_0 I_0 + \mu_\xi (x I_0 + I_1) + \mu_\eta (y I_0 + I_2) \tag{4.5}$$

and writing in vector form,

$$\phi(\mathbf{P}) = [I_0, (xI_0 + I_1), (yI_0 + I_2)] \begin{bmatrix} \mu_0 \\ \mu_\xi \\ \mu_\eta \end{bmatrix} \tag{4.6}$$

Now doublet strengths at the three nodes of a panel are defined as

$$\begin{aligned} \mu_1 &= \mu_0 + \mu_\xi \xi_1 + \mu_\eta \eta_1 \\ \mu_2 &= \mu_0 + \mu_\xi \xi_2 + \mu_\eta \eta_2 \\ \mu_3 &= \mu_0 + \mu_\xi \xi_3 + \mu_\eta \eta_3 \end{aligned} \tag{4.7}$$

In matrix form,

$$\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_\xi \\ \mu_\eta \end{bmatrix} \tag{4.8}$$

Thus μ_0, μ_ξ, and μ_η for a panel can be expressed in terms of the doublet strengths at the panel nodes as,

$$\begin{bmatrix} \mu_0 \\ \mu_\xi \\ \mu_\eta \end{bmatrix} = \begin{bmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} \tag{4.9}$$

Substituting Equation 4.9 back into Equation 4.6, rewriting the vector of I's as a column vector, and dropping the vector of node-based doublet strengths since these are of unit strength in computing influence coefficients, now gives

$$\boldsymbol{\phi}(\mathbf{P}) = \begin{bmatrix} \phi_1(\mathbf{P}) \\ \phi_2(\mathbf{P}) \\ \phi_3(\mathbf{P}) \end{bmatrix} = \begin{bmatrix} I_0 \\ xI_0 + I_1 \\ yI_0 + I_2 \end{bmatrix} \begin{bmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{bmatrix}^{-1} \tag{4.10}$$

where the elements of $\boldsymbol{\phi}(\mathbf{P})$ are the influence coefficients for each node-based doublet on the point $\mathbf{P}$.

This method works independent of the I terms found by solving the BIE, so it can be used for subsonic and supersonic flows. The way this is used to construct the doublet influence coefficient matrix, $\mathbf{A}$, is discussed in the next section.

4.3 Subsonic Linear Doublet Implementation

Though this work set out with the ultimate goal of developing a higher-order supersonic panel method, a higher-order subsonic method was first developed and implemented in CPanel. This was done for a variety of reasons. First, it allowed an avenue through which familiarity with the code base could be gained, while avoiding the complexities brought about in developing a supersonic panel method. Next, in developing the higher-order doublet scheme discussed in the previous section, it was found that this new method would require substantial architectural updates in the CPanel code base that could be used by both subsonic and supersonic methods. Thus these changes could be developed generally, then applied and validated with a subsonic method first prior to tackling the more complex supersonic method. This guaranteed a relatively advanced minimum level of functionality when the supersonic implementation began, allowing focus to be given to the intricacies of the supersonic method. Lastly, the higher-order subsonic scheme gives the CPanel user the option to use a higher fidelity method for subsonic simulations if desired.

Since the subsonic implementation was only a stepping stone toward the final goal of this thesis, only the aspects relevant to the supersonic implementation are discussed in detail in the following sections, while other aspects are briefly noted.

4.3.1 Control Points

Looking back at the new CPanel functional flow diagram in Figure 4.1 and following the path that is taken if either of the two higher-order methods is chosen, the first process to be encountered is the computation of node-based control points. The control points, where boundary conditions are applied, are placed just beneath the nodal points, as illustrated in Figure 4.2. They are computed based on the geometries of the panels around them, and a predetermined offset factor.

A given node in the geometry discretization could be shared among any number of panels, so the calculation of a control point based off its node takes all of the panels that the node is connected to into account. This process, which is illustrated in Figure 4.3, starts with obtaining the location of the node that the control point will be based off of. Once this is found, all of the panels and edges that are connected to the node are identified. Then the direction that the control point will be offset relative to the node is computed by taking the average of the normals of all the panels the node is connected to. This gives a unit vector pointing from the body into the flow, but since the control points need to be defined inside the body, the negative of this vector is taken. Lastly, the offset of the control point relative to the nodal point is computed from the average length of all the edges the node is connected to; this quantity is then scaled by a predetermined factor, k. A control point is thus computed as

$$\mathbf{P}_{cp} = \mathbf{P}_n + (kl_{avg})\mathbf{v}_{offset} \tag{4.11}$$

The quantity, k, is predetermined and defined within the code itself. This factor is the primary control over how far offset a control point is from its corresponding nodal point. A study was performed to find at what value of k CPanel solutions began to converge. Using the higher-order subsonic method and running a simple ellipsoid shape, for which the geometry is shown in Figure 4.4, doublet strength was assessed with changing k values. These cases were run with $M_\infty \approx 0$, and no angle of attack or side slip. Doublet strength was chosen as the benchmark value because it is the first representation of the results that is computed; all other quantities of interest are computed downstream of doublet strength and will likely have been impacted by

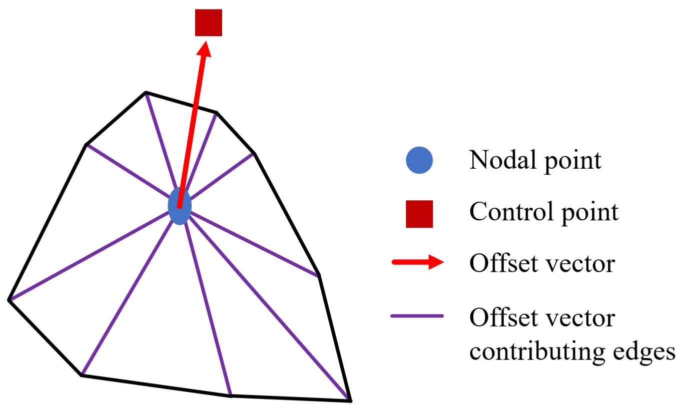


Figure 4.3: Control point definition

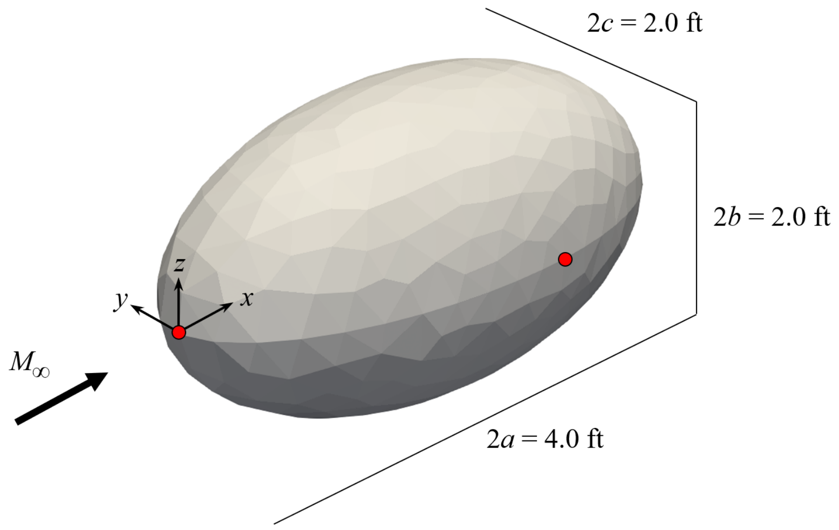


Figure 4.4: Ellipsoid geometry for control point offset convergence study

inevitable computational errors. Even though these errors are essentially negligible at the macro level, they can have an impact on a study such as this, but assessing doublet strength minimizes this impact.

The results of this study are shown in Table 4.1. The data presented here was taken at the $z = 0$ plane of the ellipsoid shown in Figure 4.4, at the forward and aft-most points (the stagnation points), and at the center-plane, which are marked in red. Only one side was taken since the results are symmetric. Higher fidelity position data relative to that presented was assessed, and the same trend was seen everywhere.

Table 4.1: Control point offset convergence study, doublet strengths

	k Values						
x-Position (ft)	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
0.0	diverges	7.469	7.749	7.749	7.729	7.729	7.729
2.0	diverges	-0.145	-0.513	-0.513	-0.611	-0.611	-0.611
4.0	diverges	-7.541	-8.157	-8.157	-8.223	-8.223	-8.223

The value of k in Equation 4.11 was varied from 10^0 to 10^{-6} as shown in the table. What was found is that for values of $k \geq 10^0$, the solution diverges. Furthermore, doublet strengths converged near $k = 10^{-4}$.

The convergence of quantities that are functions of doublet strength were also assessed, such as velocity and coefficient of pressure, and it was found that the variations in doublet strength with $k \leq 10^{-2}$ shown in Table 4.1 have a negligible effect on these downstream quantities. Maruyama et al. [7] arrived at a similar finding, leading to the conclusion that as long as k is not unreasonably large, an accurate solution can be found.

Regardless of this conclusion, it would seem that setting k to be as small as possible would be the best course of action in order to obtain the best accuracy, however this is not the case. As the control point approaches the node from which it is based, singularities in the influence coefficient calculations will occur from two different sources. One occurs when the control point is in the plane of the panel, and another when it is on the line extending from an edge of the panel. So if k is too small, then these singularities will occur and must be handled, and they can be handled as shown by Johnson [19]. Though if k is kept large enough, then these singularities can be avoided completely and an accurate solution can be obtained. Ultimately, k was set to be 10^{-5}.

4.3.2 Constructing the Linear System of Equations

The linear system of equations that is built to solve for the unknown doublet strengths was reviewed in Chapter 3, and the influence coefficient matrices were introduced. Now that the order of the method being used here has finally been determined, that is a linearly varying doublet distribution and a constant strength source distribution, and the control points defined, the construction process and composition of the influence coefficient matrices is presented.

Since this is for the case of subsonic flow, the far field potential and the interior potential, which is non-zero in the subsonic scheme, need to be accounted for in the linear system of equations. This system is shown by Satterwhite [10] to be

$$\mathbf{A}\boldsymbol{\mu} - \mathbf{B}\boldsymbol{\sigma} + \boldsymbol{\Phi}_\infty = \boldsymbol{\Phi}_{cp} \tag{4.12}$$

$$\boldsymbol{\Phi}_\infty = \begin{bmatrix} \mathbf{U}\cdot\mathbf{P}_1 \\ \mathbf{U}\cdot\mathbf{P}_2 \\ \vdots \\ \mathbf{U}\cdot\mathbf{P}_{N_n} \end{bmatrix} \tag{4.13}$$

$$\boldsymbol{\Phi}_{cp} = \begin{bmatrix} \Phi_{cp_1} \\ \Phi_{cp_2} \\ \vdots \\ \Phi_{cp_{N_n}} \end{bmatrix} \tag{4.14}$$

The vector of control point potentials, $\boldsymbol{\Phi}_{cp}$, is then set equal to $\boldsymbol{\Phi}_\infty$ yielding the system

$$\mathbf{A}\boldsymbol{\mu} = \mathbf{B}\boldsymbol{\sigma} \tag{4.15}$$

which is the same as the system given in Equation 3.1.

The doublets, for which the strengths need to be solved, are defined at panel nodal points. The sources, for which the strengths are assigned by Equation 2.46, are defined at each panel. So it is known that there are N_n doublets and N_p sources, and $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ can be written as, respectively,

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{N_n} \end{bmatrix} \qquad (4.16) \qquad \boldsymbol{\sigma} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{N_p} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \cdot \hat{n}_1 \\ \mathbf{U} \cdot \hat{n}_2 \\ \vdots \\ \mathbf{U} \cdot \hat{n}_{N_p} \end{bmatrix} \qquad (4.17)$$

The influence coefficient matrices are constructed by computing the potential influence of a panel on a control point, with unit strength singularity elements prescribed on the panel, for all the panel-control point pairings in the geometry. Even though the doublet strengths are defined at the nodal points of the geometry, the doublet influence coefficients still must be computed by finding the influence of a panel on a control point. The scheme presented in Section 4.2 is then used to convert the panel's linear doublet influence to node-based influences. The source influence coefficients are computed in the exact same manner as in the lower-order subsonic scheme since constant strength sources are still used. The influence coefficient matrices are written as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N_n} \\ a_{21} & a_{22} & \cdots & a_{2N_n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N_n1} & a_{N_n2} & \cdots & a_{N_nN_n} \end{bmatrix} \qquad (4.18) \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1N_p} \\ b_{21} & b_{22} & \cdots & b_{2N_p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N_n1} & b_{N_n2} & \cdots & b_{N_nN_p} \end{bmatrix} \qquad (4.19)$$

Note that these matrices are different dimensions: **A** is N_n x N_n and **B** is N_n x N_p. Each coefficient of **B**, b_{ij}, represents the unit potential influence of the j^{th} source panel on the i^{th} control point, and is found by solving

$$b_{ij} = -\int_S \sigma \left(\frac{1}{4\pi|\mathbf{r}_{ij}|} \right) dS \qquad (4.20)$$

This is the same scheme as is used in the original lower-order method of CPanel, except that the control points are located just under nodal points rather than just under panel centroids. Each coefficient of $\mathbf{A}$, a_{ik}, represents the unit potential influence of the k^{th} doublet point on the i^{th} control point. In determining the doublet influence coefficients for each node-control point pair, the influence of all the panels that share the given node must first be found. The process to compute a_{ik} from the influence of each of these panels is shown in Figure 4.5 and discussed below.

Starting at just one of the panels that contains the k^{th} control point and denoting it the j^{th} panel, the influence of the j^{th} linear doublet panel on the i^{th} control point is computed. This scenario is illustrated in Figure 4.5(a). The equation that is solved to compute this influence is

$$\alpha_{ij} = -\int_S \mu \left(\frac{\hat{n}_j \cdot \mathbf{r}_{ij}}{4\pi |\mathbf{r}_{ij}|^3} \right) dS \tag{4.21}$$

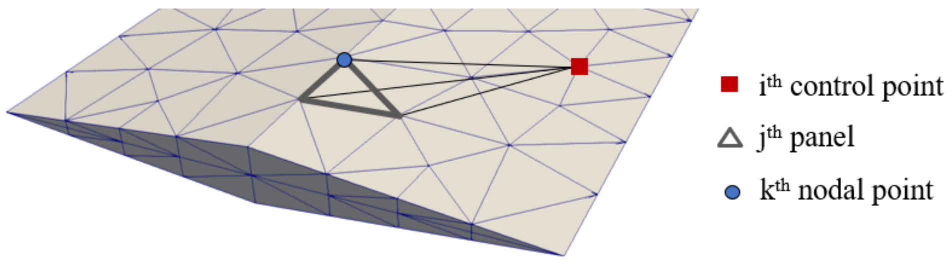
which is the same as that used for the lower-order method, except that the doublet strength term, μ, is of the form of Equation 4.1 rather than a constant. When solved, Equation 4.21 can be written as

$$\alpha_{ij} = \mu(x,y) I_0 + \mu_x(x,y) I_1 + \mu_y(x,y) I_2 \tag{4.22}$$

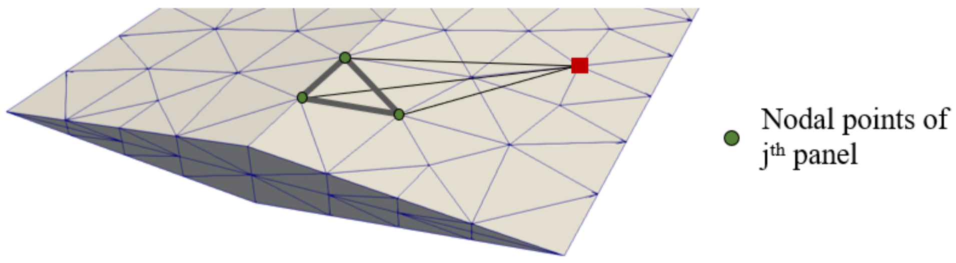
Equation 4.22 is the same form as Equation 4.2, thus the process laid out in Equations 4.4-4.10 to convert this panel-based doublet influence coefficient expression to node-based doublet influence coefficients can be used here. As illustrated in Figure 4.5(b), this gives the influence coefficients for the three nodes of the j^{th} panel on the i^{th} control point, based on only the j^{th} panel, which can be expressed as

$$\boldsymbol{\alpha}_{ij} = \begin{bmatrix} \alpha_{ij_1} \\ \alpha_{ij_2} \\ \alpha_{ij_3} \end{bmatrix} \tag{4.23}$$

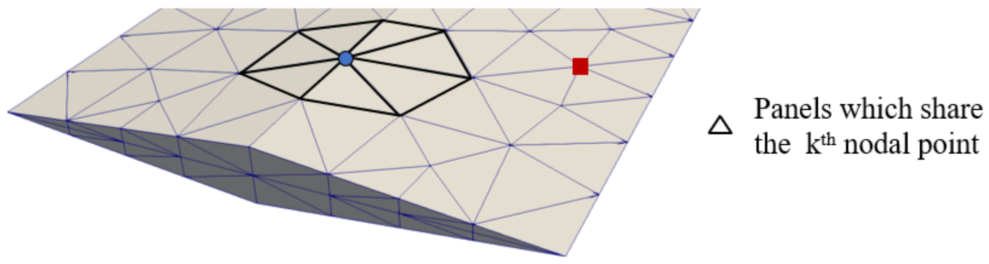
The same process is then followed for all the panels that share the k^{th} node, as identified in Figure 4.5(c), and $\boldsymbol{\alpha}_{ij}$ is found for each of these panels. With the $\boldsymbol{\alpha}_{ij}$



(a) *Influence of jth panel on ith control point*



(b) *Node based influences of jth panel on ith control point*



(c) *Panels whose influences contribute to the influence coefficient of the kth nodal point*

Figure 4.5: Computing the node-based influence coefficient via its surrounding panels

vector from each identified panel in Figure 4.5(c) found, the element from each vector which corresponds to the k^{th} nodal point is summed. This summation then finally gives the influence coefficient for the unit influence of the k^{th} nodal point on the i^{th} control point, a_{ik}.

The full solutions to the integrals of Equations 4.21 and 4.20 are given in Appendix A; the expressions for the various I terms of Equation 4.22 are also given. The vector of doublet strengths can then be found by solving the linear system of equations.

4.3.3 Post Processing

All other aerodynamic quantities of interest can be found once the doublet strengths have been solved for. The method that is used to compute velocity for this higher-order subsonic scheme is now presented; the computation of other aerodynamic quantities is not discussed here since once velocity is found, these quantities are computed using the same methods as given in the original CPanel documentation [10].

The first quantity to be found once the nodal doublet strengths are known is the total velocity potential, Φ, at the nodal points. The method presented in Section 3.3 could be used here, however an even simpler approach is used, which is the same approach used to compute total velocity potential in the lower-order subsonic method.

Similar to how a difference velocity was defined in Equation 3.4, a difference potential can be written in terms of the upper and lower surface potentials as

$$\Phi_D = \Phi_U - \Phi_L \tag{4.24}$$

The upper potential, Φ_U, is the potential that is being solved for at the nodal points; the lower potential, Φ_L, is the potential at the control points just below the surface of the nodal points, which is the same as the potentials expressed in Equation 4.14. The lower potential was defined to be equal to the far field potential,

$$\Phi_{L_k} = \mathbf{U} \cdot \mathbf{P}_k \tag{4.25}$$

Furthermore, nodal doublet strength, μ_k, represents the potential difference between the upper and lower surfaces at the nodal points, so

$$\Phi_{D_k} = -\mu_k \tag{4.26}$$

Now the upper potential at a nodal point can be written in terms of the known lower and difference potentials as

$$\Phi_{U_k} = \mathbf{U} \cdot \mathbf{P}_k - \mu_k \tag{4.27}$$

This is the calculation used to compute potential for a panel in CPanel's lower-order subsonic method. However in this case, the potentials are defined at nodal points. To compute velocity from this, a potential equation must be written for the entire panel.

Using the formulation of Equation 4.27, for the j^{th} panel in a given surface discretization, the upper potential at the nodal points of this panel can be computed and written in the form of Equation 4.23 as

$$\mathbf{\Phi}_j = \begin{bmatrix} \Phi_{j_1} \\ \Phi_{j_2} \\ \Phi_{j_3} \end{bmatrix} \tag{4.28}$$

In the same way that the nodal doublets of a panel were each written as linearly varying functions in terms of the nodal local coordinates in Equation 4.7, the same can be done for the nodal potentials. Thus the linear variation of potential for the j^{th} panel can be written in the form of Equation 4.9 as

$$\begin{bmatrix} \Phi_{0_j} \\ \Phi_{\xi_j} \\ \Phi_{\eta_j} \end{bmatrix} = \begin{bmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{bmatrix}^{-1} \begin{bmatrix} \Phi_{j_1} \\ \Phi_{j_2} \\ \Phi_{j_3} \end{bmatrix} \tag{4.29}$$

With the coefficients on the left hand side of Equation 4.29 known, the potential at a point on the j^{th} panel can now be found by computing

$$\Phi_j(\xi, \eta) = \Phi_{0_j} + \Phi_{\xi_j}\xi + \Phi_{\eta_j}\eta \tag{4.30}$$

Taking the gradient of this equation in the panel local coordinate system then gives the local tangential velocity for the j^{th} panel,

$$\mathbf{V}_{l_j} = \left[(\Phi_{\xi_j})_\xi, (\Phi_{\eta_j})_\eta, 0\right] \tag{4.31}$$

Transforming this vector to the global coordinate system then gives the total velocity vector at the j^{th} panel, $\mathbf{V}_j$.

4.4 Supersonic Implementation Preprocessing

Looking back at the functional flow diagram of Figure 4.1 and following the path for the supersonic scheme, there are two processes that are encountered prior to calculating the influence coefficients. A domain of dependence check must be performed for each panel-control point pairing, and each panel is transformed in a way which greatly simplifies the integration process. These two processes are together categorized as the supersonic implementation's preprocessing operations since these are the two operations that read the geometry discretization and other input data, and generate the new data that is used to ultimately compute the influence coefficients, and construct the linear system of equations.

4.4.1 Domain of Dependence Check

In Chapter 2, the Mach cone was defined and the definitions of the domain of dependence and the domain of influence were given. Also discussed in this chapter is the fact that a given point in a supersonic flow field can only be influenced by flow perturbations that occur inside this point's domain of dependence. Given a surface discretization with a supersonic flow field, this concept can be directly applied to determining whether or not the j^{th} panel is inside the domain of dependence of the i^{th} control point.

Performing a domain of dependence check for each panel-control point pair prior to computing influence coefficients enables the program to completely skip the influence coefficient calculation for the given pair if the j^{th} panel is found to be outside the domain of dependence of the i^{th} control point. A panel outside the domain of influence of a point has no influence on the point. Calculating the influence coefficients is easily the most computationally intensive process shown in the diagram of Figure 4.1, so skipping the process entirely when it is known from this check that there is no influence greatly enhances the computational efficiency of the supersonic scheme.

The process for checking whether the j^{th} panel is inside the domain of dependence of the i^{th} control point is given in Algorithm 4.1. This algorithm's form and function was influenced by those of Ehlers et. al. and Epton and Magnus [2, 18]. The inputs are shown, and the output is a flag that is defined to be true if the panel is either completely inside the DOD, or if it intersects the DOD, and the flag is false if neither of these cases are met.

Quickly walking through this algorithm, Lines 1-4 compute and assign the various

Algorithm 4.1: Domain of dependence check

```
   Input: Panel data, Control point, Mach number, Freestream direction
   Output: DOD flag
1  DOD flag ← false;
2  $R_{B_{cntr}}$ ← hyperbolic distance from control point to panel center;
3  $d$ ← shortest distance from Mach cone to panel center;
4  $r_p$ ← radius of panel;
5  if control point is downstream from panel center then
6      if $d > r_p$ then
7          if $R_{B_{cntr}} \geq 0$ then
8              return DOD flag ← true;
9          end
10     else
11         for each panel nodal point do
12             $R_{B_n}$ ← hyperbolic distance from control point to panel nodal point;
13             if $R_{B_n} \geq 0$ and control point is downstream from nodal point then
14                 return DOD flag ← true;
15             end
16         end
17     end
18 end
19 if $d < r_p$ then
20     return DOD flag ← true;
21 end
22 return DOD flag;
```

entities needed for the subsequent checks. Then it is checked whether or not the control point is downstream from the panel center. If it is not, then either the panel has no influence on the control point, or the DOD intersects the panel without the panel center being inside the DOD, as depicted in Figure 4.6. Line 19 checks for the latter case. This condition could be true even if the DOD does not intersect the panel; however, it cannot be trivially determined at this point in the program whether or not this is the case. So, the flag is set as true and it is checked more closely later in the program, which will be discussed further in the chapter.

If the conditions of Lines 5 and 6 are true, then it is known that the panel is either completely inside or completely outside the DOD, which is then checked in Line 7. If the condition of Line 6 is not met, then each node is checked for whether it is inside the DOD or not, and returns true if any of them are since this indicates a panel-DOD intersection.

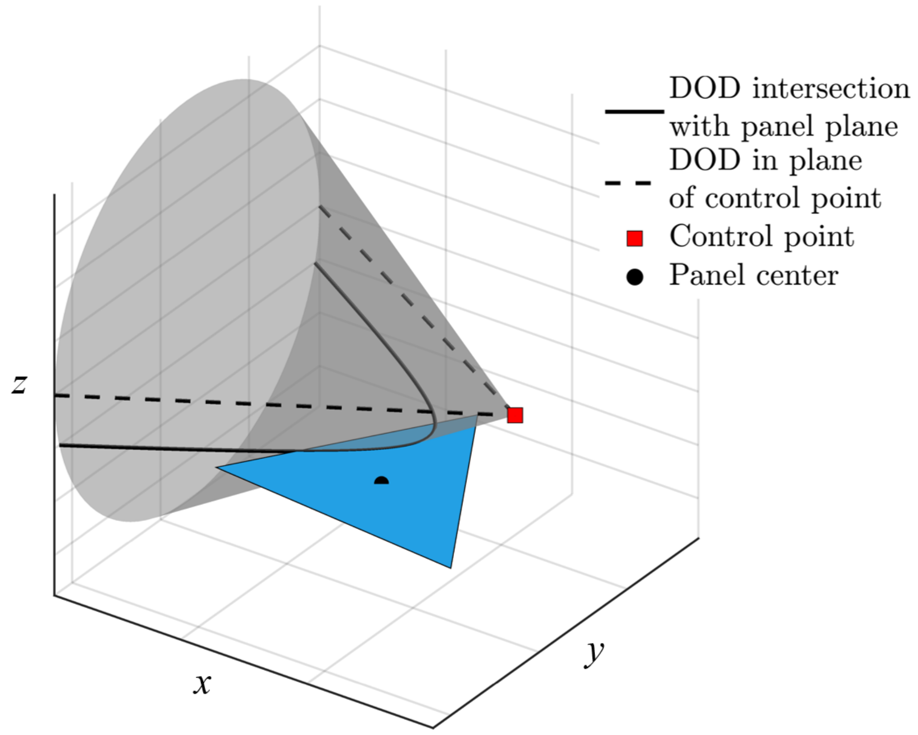

DOD intersection with panel plane
DOD in plane of control point
Control point
Panel center
z
x
y

Figure 4.6: Example panel-DOD intersection

4.4.2 Coordinate Transformation

All panel methods use some version of a coordinate transformation when computing influence coefficients, since the assessment is always how a given singularity element influences a given point. So it is easiest to isolate the problem and work in a coordinate system specific to the case at hand. This can be approached multiple different ways. Many panel codes have inherited what was first done by Hess and Smith [20] and transform the panel and control point into a panel local reference frame for the influence coefficients to be computed in. Another approach developed my Maskew [21] and implemented into CPanel's lower-order subsonic method [10] is to work in terms of vector products, thus avoiding the need to perform a direct coordinate transformation.

Supersonic panel codes often use a coordinate transformation for an additional benefit which is to simplify the BIE of Equation 2.43 thus also simplifying the solution to the BIE and its implementation. Along with transforming the panel and control point into a panel local reference frame, the coordinate system is also scaled such that the problem is reduced to the case of $M_\infty = \sqrt{2}$. Looking back at the definition of B in Equation 2.27, when $\sqrt{2}$ is substituted in for M here, then $B = 1$. This removes all instances of B in the BIE which greatly simplifies its evaluation. The panel is also transformed such that it is parallel to the freestream direction, and the origin of the system is at the panel center.

This method has been used by the PANAIR pilot code, PANAIR, and HISSS [2, 18, 6]. The former two present derivations of this transformation and go through the properties of the transformation matrix, which are both fairly involved and are not presented here. The form of the matrix used in this implementation is given by Epton et. al. [18](App. E).

Any given point can be transformed into this local, scaled coordinate system by computing

$$\mathbf{P}_l = [A](\mathbf{P} - \mathbf{P}_0) \tag{4.32}$$

where A is the transformation matrix. The physical effect of this transformation is shown in Figure 4.7 for a single panel and control point, with two different Mach numbers. Note that the same two DOD curves shown in Figure 4.6 are shown here, but from a viewing point perpendicular to the plane of the panel.

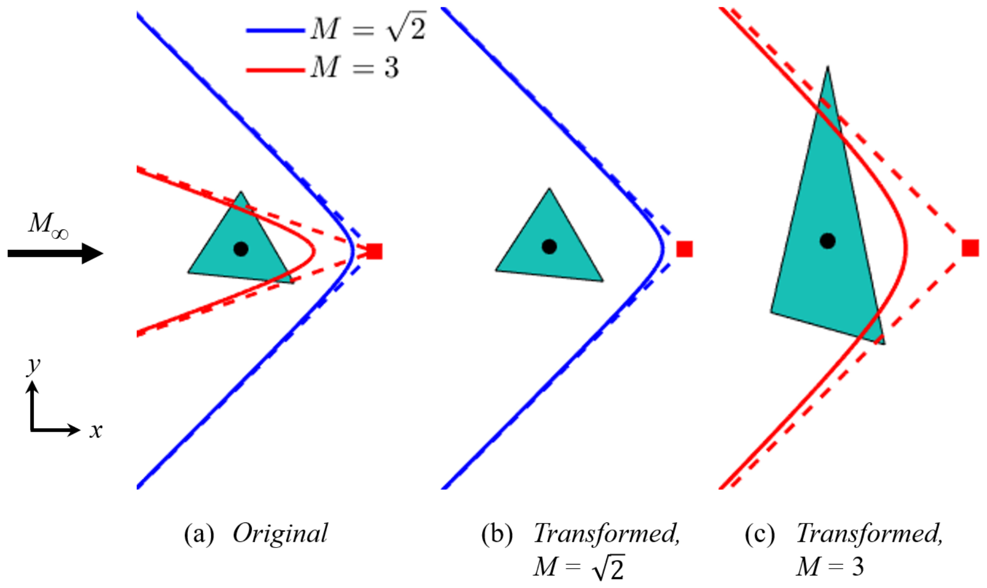


Figure 4.7: Effect of panel transformation for different Mach numbers

Figure 4.7(a) shows the original panel geometry before the panel and control point have been transformed, and it shows the upstream Mach cones for $M = \sqrt{2}$ and $M = 3$. Notice that the Mach cone intersection with the plane of the control point for the $M = \sqrt{2}$ case, represented by the blue dashed lines, forms a 90° degree angle–this is twice the Mach angle, as expected. For the $M = 3$ case, the Mach angle is clearly less. Figure 4.7(b) shows the panel and control point for the $M = \sqrt{2}$ case after they have been transformed using Equation 4.32. The geometry is completely unchanged since the case is already that for $M = \sqrt{2}$. In Figure 4.7(c) however, since $M = 3$ for this case, the panel geometry is scaled to reduce the problem to that for

$M = \sqrt{2}$. It can be validated that the panel has in fact been transformed correctly by observing that the half angle of the Mach cone is now 45°.

4.5 Supersonic Influence Coefficients Calculation Procedure

Even though the domain of dependence check is shown to occur prior to the panel transformation in Figure 4.1, this is not exactly the order in which these processes are executed. However it is shown this way in the functional flow diagram to emphasize that the domain of dependence check is performed on panel-control point pairs before they are transformed. The actual order of operations for this portion of the program is shown in Algorithm 4.2.

Every panel is guaranteed to be inside of or to intersect at least one control point's domain of dependence; this would not be the case if superinclined panels were used, however they are not allowed in this implementation. So the transformation matrix

Algorithm 4.2: Construction of the influence coefficient matrices

```
Result: Influence coefficient matrices
for each panel do
    compute and store transformation matrix;
    transform panel;
end
for $k^{th}$ node do
    get ith control point;
    for jth panel do
        DOD flag ← Algorithm 4.1;
        if DOD flag is true then
            $\mathbf{A}(i,k)$ ← doublet influence coefficient for node-control point pair;
            $\mathbf{B}(i,j)$ ← source influence coefficient for panel-control point pair;
        else
            $\mathbf{A}(i,k) \leftarrow 0$;
            $\mathbf{B}(i,j) \leftarrow 0$;
        end
    end
end
```

of each panel is computed and stored–since it will be used more than once throughout the program–and each panel is transformed; this is all done first in Algorithm 4.2. Since the domain of dependence check must be done for each panel-control point pair, the check is actually performed in a nested loop with the influence coefficient calculations.

The following sections of this chapter present the procedures which are performed as part of the influence coefficient calculation process, which all occur in Lines 10 and 11 of Algorithm 4.2. The methods of Ehlers et al. [2] are used for these calculations; since these methods are presented in great detail in their paper, these sections focus on the general characteristics of the procedures and how they are implemented in CPanel, rather than the equation formulations. The logical flow of how these methods are implemented in CPanel is also discussed since this aspect of the method may be unique to CPanel, and because its understanding is important to the future expandability of the program.

4.5.1 Build Edge-Based Coordinate System

In computing an influence coefficient, the integrals of Equation 2.43 are carried out over the panel geometry. However it is easiest to break down the integration over the panel into individual integrations over each edge, and to then sum the results. To further simplify these calculations, Ehlers et al. [2] developed an edge-based oblique coordinate system in which the quantities needed for the influence coefficient calculations are computed. Before discussing this oblique coordinate system, a distinction must be made between three different categories of panel edge orientations.

Edges are defined to be subsonically inclined, supersonically inclined, or sonically inclined, where the edge inclination is relative to freestream direction. When panels are transformed, part of the transformation is to change the orientation of the panel such that it is parallel to the freestream direction, which allows the direct comparison of panel edge inclination with the freestream direction since they are in the same plane. An example of each edge inclination type, as well as the downstream Mach cones emanating from the endpoints of each edge, is shown in Figure 4.8. A subsonic edge is that which is inclined at an angle less than the freestream Mach angle; a supersonic edge is at an inclination that is greater than the freestream Mach angle; and a sonic edge is at the same inclination as the freestream Mach angle.

The reason these distinctions must be made is that how an edge may influence a point depends on its orientation relative to the freestream direction. Notice in Figure

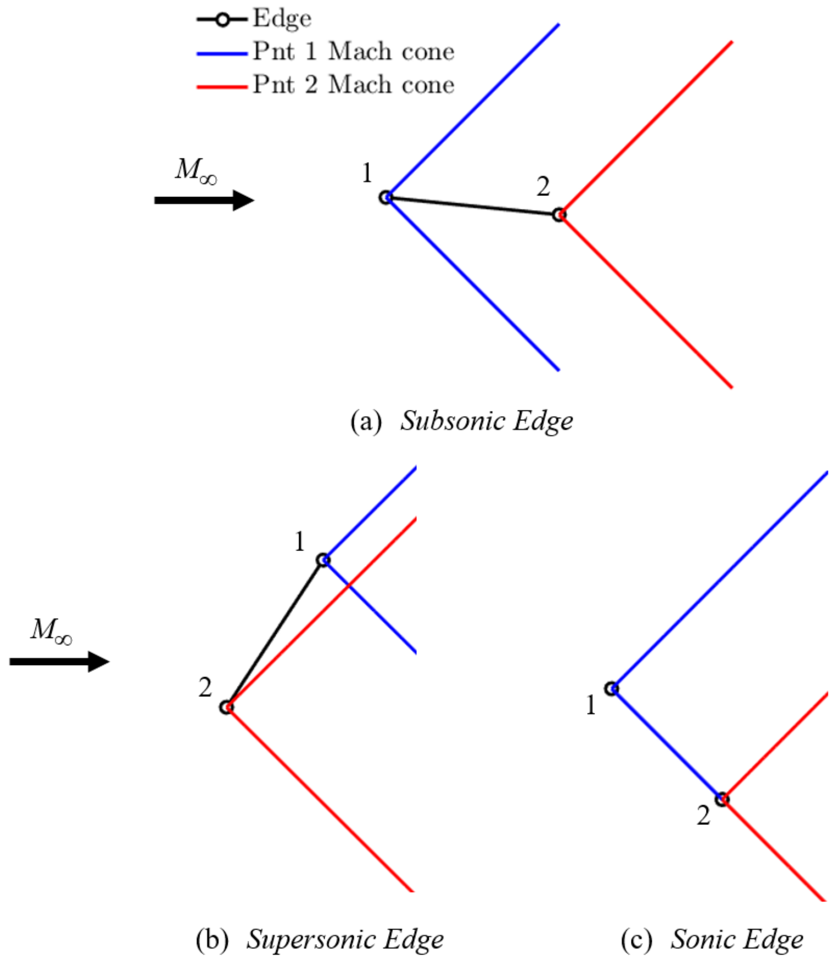
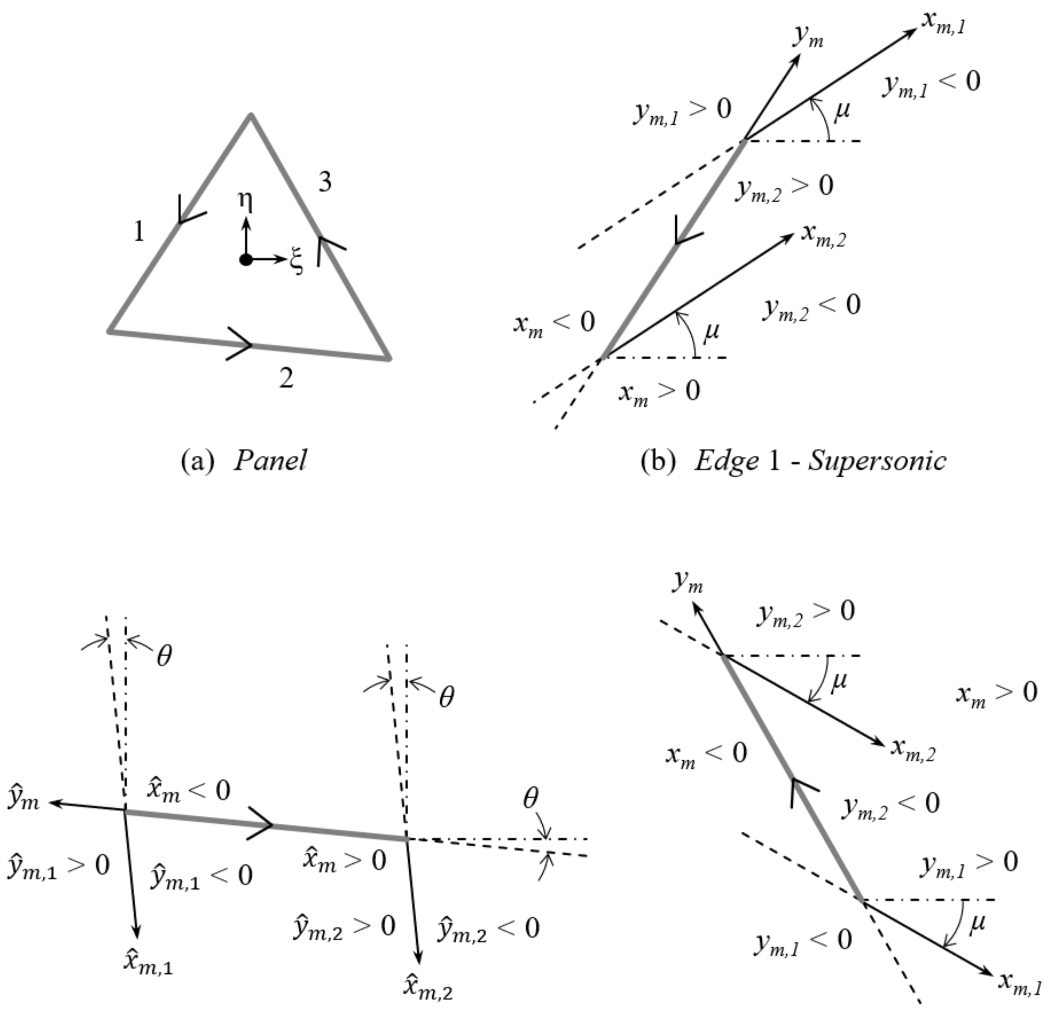


Figure 4.8: Edge inclination examples with downstream Mach cones

4.8(a) that for a subsonic edge, any point inside the domain of influence of point 2 is also inside the domain of influence of point 1. For the supersonic edge in Figure 4.8(b), this is not the case and there are regions where a point can be in one of the end point's domain of influence, but not the other. The sonic edge in Figure 4.8(c) is similar to the supersonic edge; however, because it is inclined at the Mach angle, a singularity occurs in the integration over this edge, which is handled in the process of computing the influence coefficients once the edge-based coordinate systems have been defined.

With the types of edge orientation defined, the oblique coordinate systems built from panel edges can now be discussed. Three different examples of the oblique edge-based coordinate systems are shown in Figures 4.9(b)-(d), and the panel these edges are a part of is shown in Figures 4.9(a); the counterclockwise direction of integration around the panel is also shown. For supersonic edges, the x-direction is denoted x_m, where there is an x_m direction based at both end points of an edge. The y-direction is denoted y_m. For subsonic edges, the nomenclature is essentially the same except $\hat{x}_m$ and $\hat{y}_m$ are instead used for the two coordinate directions. It should be noted that the $x_m - y_m$ system can also be applied to subsonic edges, but in actually computing influence coefficients for subsonic edges, the $\hat{x}_m - \hat{y}_m$ system is used.

For any edge, the y_m or $\hat{y}_m$ directions are always defined as the axis of the edge itself. The y_m direction is further defined to always point toward the positive η-direction of the panel coordinate system, and the $\hat{y}_m$ direction is defined to always point toward the negative ξ-direction of the panel coordinate system. It should be noted that the ξ-direction is always parallel to the freestream direction. As can be seen in comparing the y_m directions in Figures 4.9(b) and 4.9(d), this axis always points toward the positive η-direction regardless of the integration direction. The same is true of the $\hat{y}_m$-direction in pointing toward the negative ξ-direction, though it is not shown here.



(a) *Panel* (b) *Edge* 1 - *Supersonic*

(c) *Edge* 2 - *Subsonic* (d) *Edge* 3 - *Supersonic*

Figure 4.9: Example edge coordinate systems

What does change based on integration direction are the x_m and $\hat{x}_m$ axes. Again comparing the supersonic edges of Figures 4.9(b) and 4.9(d), notice that the x_m axes point opposite to the direction of integration across each respective edge. In either case, the x_m axes are furthermore defined as the axes of the downstream Mach cones emanating from each respective edge endpoint. Thus as is shown in the Edge 1 and 3 diagrams, the angle of each axis with respect to the freestream direction is the freestream Mach angle, μ, which is always true for any supersonic edge. For subsonic edges, the positive $\hat{x}_m$-direction points toward the direction of integration, contrary to supersonic edges. The axis itself is defined by a rotation of θ from the panel's η-direction about the panel's ζ axis which points out of the page; θ here is the angle of the edge relative to the freestream. For a sonically inclined edge, the two coordinate systems are identical.

With these new edge-based coordinate systems, the location of any point can be defined in terms of (x_m, y_m), or $(\hat{x}_m, \hat{y}_m)$. The third direction for these respective coordinate systems are defined similarly to the y-directions and are denoted z_m and $\hat{z}_m$; the exact definitions of these axes are not shown here. The benefit of this system is immediately apparent in the supersonic edge x_m-y_m system by observing that when a given point is written in both $(x_m, y_{m,1})$ and $(x_m, y_{m,2})$ coordinates, only the signs of these coordinates are needed to determine where the point is relative to the two endpoint downstream Mach cones. This is very useful and easily attainable information which is used in part to compute the influence coefficients. The details of this utility is discussed in the next section.

4.5.2 Compute Fundamental Integrals

The method used by Ehlers et al. [2] to compute influence coefficients uses the idea of fundamental integrals and is done in rectilinear coordinates, as was discussed in Chapter 3, and this is the method that is implemented here. The fundamental integrals are developed such that they can be computed once per panel edge-control point pair, then the remaining calculations can be performed in terms of the solutions of these integrals. For higher-order panel methods, this methodology is very advantageous since the BIE solutions can quickly become complex and computationally expensive, and doing this helps in alleviating some of the computational cost of higher-order methods.

Even though the methods of Ehlers et al. [2] were developed for use with a quadratically varying doublet distribution and a linearly varying source distribution, they can be easily modified for use with a linearly varying doublet distribution, and constant source distribution. Furthermore, the exact same fundamental integrals can be used.

Since the expression for the source potential influence of an edge on a point is written for a linearly varying source in Ehlers et al., all that needs to be done to arrive at the same expression for a constant source is to drop all higher-order terms, which leaves

$$\phi_S = \frac{\sigma}{2\pi}\big(x_m w_0 - zQ_I\big) \tag{4.33}$$

where w_0 and Q_I are two of a handful of fundamental integrals developed by Ehlers et al. A similar procedure can be followed to arrive at the expression for the doublet influence of an edge on a point for a linearly varying doublet from a quadratically varying doublet. Dropping all terms that are higher than first order from the original expression for a quadratic doublet from Ehlers et al. leaves

$$\phi_D = -\frac{1}{2\pi}\big(\mu Q_I - \mu_x z w_0 - \mu_y z w_0/m\big) \tag{4.34}$$

which uses the same two fundamental integrals as Equation 4.33, w_0 and Q_I. Thus these are the only two fundamental integrals which need to be computed in finding the influence coefficients.

Computing the solutions of these two integrals, w_0 and Q_I, is the core of the influence coefficient calculation process, and in turn, the core of this implementation. The way this process is implemented in CPanel is laid out in Algorithm 4.3. As noted earlier, the fundamental integrals are computed for each edge, so this algorithm is used for each edge of a panel, then summed at the end to find the fundamental integral solutions for the panel itself.

The first process of Algorithm 4.3, which is represented in Lines 7-18, is to compute the hyperbolic distances from the control point to each edge point, which is only carried out for each respective end point if the control point is downstream from the given end point. After the hyperbolic distances have been computed, it is checked whether or not they have imaginary parts. If one does, this indicates that the control point is outside the DOI of the given end point, in which case the hyperbolic distance is set to zero. If the control point is inside the DOI, then it is positive.

Once the two end point hyperbolic distances have been computed, it is checked if

Algorithm 4.3: Calculation of the fundamental integral solutions

```
Input: Transformed panel edge, Transformed control point
Result: Fundamental integral solutions
1  check edge orientation and compute corresponding edge-based control point
     coordinates;
2  $p_{cp} \leftarrow$ control point;
3  $p_1 \leftarrow$ edge point one;
4  $p_2 \leftarrow$ edge point two;
5  $R_1 \leftarrow 0$ ; // $p_{cp}$ to $p_1$ hyperbolic distance in panel local CSYS
6  $R_2 \leftarrow 0$ ; // $p_{cp}$ to $p_2$ hyperbolic distance in panel local CSYS
7  if $p_{cp}$ is downstream of $p_1$ then
8      compute $R_1$;
9      if $R_1$ is imaginary then
10         $R_1 \leftarrow 0$;
11     end
12 end
13 if $p_{cp}$ is downstream of $p_2$ then
14     compute $R_2$;
15     if $R_2$ is imaginary then
16         $R_2 \leftarrow 0$;
17     end
18 end
19 if $R_1 > 0$ or $R_2 > 0$ then
20     compute fundamental integrals based on edge orientation;
21 else if the edge is supersonic then
22     if $x_m > 0$ and sign($y_{m,1}$) != sign($y_{m,2}$) and $z_m > 0$ then
23         compute Mach wedge fundamental integrals;
24     end
25 else
26     the fundamental integral solutions are set to zero;
27 end
```

at least one of them is greater than zero. The logical 'or' is used in this scenario since the DOD only needs to intersect the edge for it to have an influence on the control point. If at least one of the end points is inside the control point DOD, which indicates the edge either intersects the DOD or is completely inside it, then the fundamental integral solutions are computed.

If neither of the conditions of Line 19 are met, this leaves two remaining options. The first option is that the edge has no influence on the control point. Remember that in the domain of dependence check of Algorithm 4.1, there was a scenario where the DOD flag could be set as true even if this could not be determined to be the case at the time. When this occurs in that algorithm, this is where it is finally determined, though each edge is checked individually instead of the panel itself. If Line 26 of Algorithm 4.3 is reached, then the edge has no influence on the control point.

The other scenario that could result in neither condition of Line 19 being met is that the control point is in the Mach wedge region of the edge. The Mach wedge is defined as the region between the Mach cones emanating from the end points of an edge, as illustrated by Ehlers et al. in Figure 4.10. When a control point is in this region, the two edge hyperbolic distances are imaginary and are thus set to zero in Algorithm 4.3.

When both edge hyperbolic distances are zero, to check whether the control point is inside the Mach wedge region or not, it is first checked if the edge is supersonic. Looking back at the 2D edges and Mach cones of Figure 4.8, it can be seen that only a supersonic edge can have a Mach wedge, which is the triangular region of Figure 4.8(b) enclosed by the edge and two edge Mach cones. If the edge is supersonic, then to check if the control point is inside the Mach wedge region of the edge, the edge-based (x_m, y_m, z_m) coordinate system is used. As alluded to at the end of the previous section, this is where the edge-based coordinate system proves most useful.

By checking just the signs of the edge-based coordinates for the control point, it can be determined whether or not the control point is inside the edge Mach wedge. If it is, then the fundamental integrals must be treated accordingly to find their solutions; if it is not then the edge has no influence on the point.

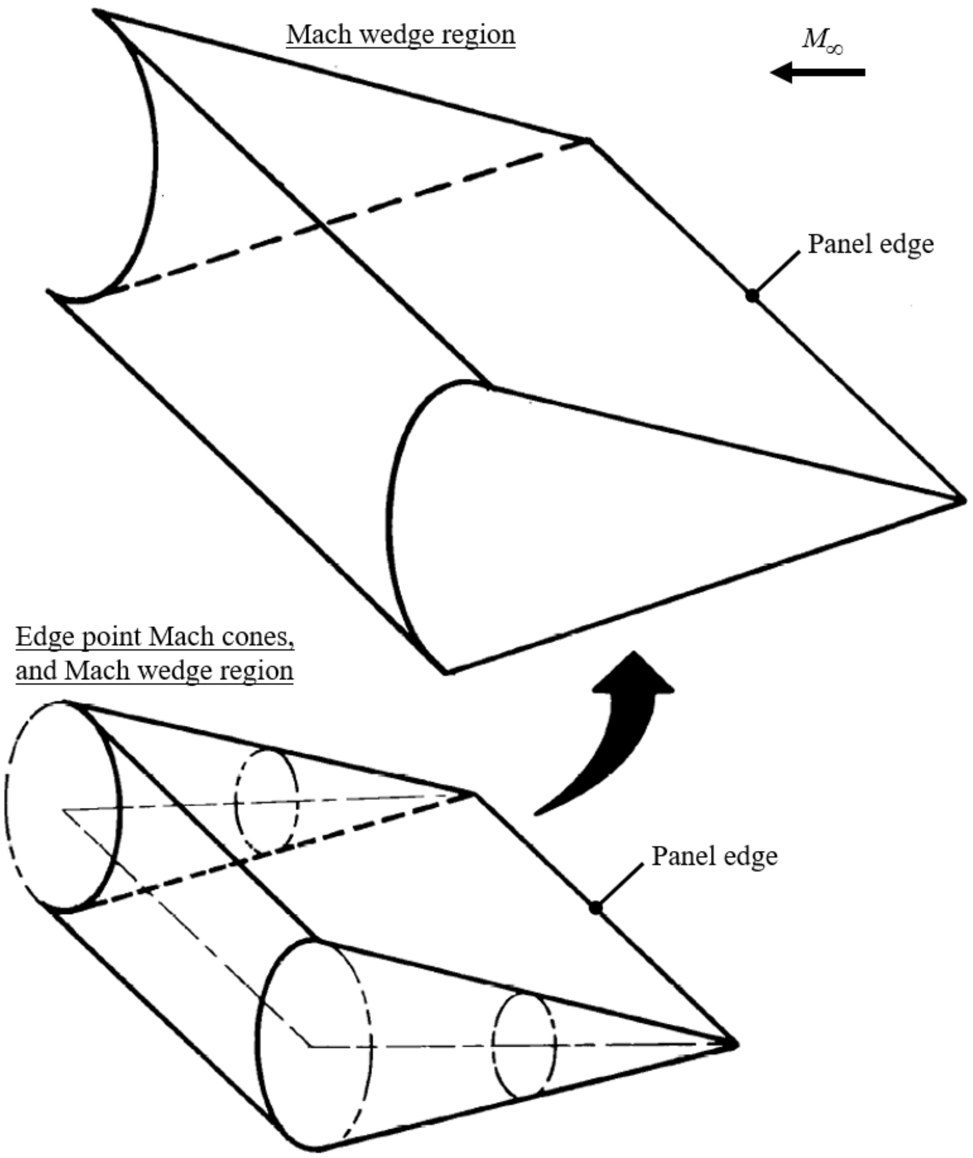


Figure 4.10: Example of the Mach wedge region [2]

4.5.3 Compute Influence Coefficients

After computing the fundamental integral solutions for a given panel-control point pair, w_0 and Q_I, the source and doublet influence coefficients for the panel-control point pair are computed using Equations 4.33 and 4.34, respectively. Unit strength singularities are used in these calculations. Before storing these influence coefficients in their respective influence coefficient matrices and moving on to the next panel-control point pair, one last operation is required for each computed coefficient.

As explained by Ehlers et al. [2], the influence coefficients were computed in the panel local-scaled coordinate system, but only the panel nodal points and control point were transformed, thus there remain parts of the integral equations that have yet to be transformed. Looking back at the integrals of Equation 2.43, R_B is the only term in this equation that has been transformed; the derivative of the conormal in the freestream direction and the elemental area terms also need to be transformed. The transformation of the elemental area, dS, manifests itself in multiplying the results of Equations 4.33 and 4.34 by a scaling factor, which is derived by Ehlers et al. to be

$$J_a = 1/\left(B\sqrt{1 - M^2 n_{x\infty}^2}\right) \tag{4.35}$$

$$\hat{n}_\infty = (n_{x\infty}, n_{y\infty}, n_{z\infty}) \tag{4.36}$$

where $\hat{n}_\infty$ is the panel normal in the wind frame coordinate system. So the final source influence coefficient is computed as

$$b_{ij} = J_a \phi_S \tag{4.37}$$

For the doublet influence coefficient, it is shown by Ehlers et al. that the scaling factor, J_a, actually does not need to be applied to Equation 4.34 if the conormal derivative in the left hand doublet integral of Equation 2.43 is replaced with the derivative of the panel local z-coordinate, ζ. This is in fact done in deriving Equation

4.34, though it is not shown. So Equation 4.34 gives the expression for the doublet influence coefficient of the j^{th} panel on the i^{th} control point,

$$\alpha_{ij} = \phi_D \tag{4.38}$$

which is a linearly varying equation.

Notice that Equation 4.34 has the same form as Equation 4.22, so it can be written as

$$\alpha_{ij} = \mu(x,y)K_0 - \mu_x(x,y)K_1 - \mu_y(x,y)K_2 \tag{4.39}$$

with

$$\begin{aligned} K_0 &= -\frac{1}{2\pi}Q_I \\ K_1 &= -\frac{1}{2\pi}zw_0 \\ K_2 &= -\frac{1}{2\pi}zw_0/m \end{aligned} \tag{4.40}$$

Furthermore, following the process of Section 4.2, the node-based influence coefficients for a given panel-control point pair are found as

$$\boldsymbol{\phi}_D = \boldsymbol{\alpha}_{ij} = \begin{bmatrix} \alpha_{ij_1} \\ \alpha_{ij_2} \\ \alpha_{ij_3} \end{bmatrix} = \begin{bmatrix} K_0 \\ xK_0 - K_1 \\ yK_0 - K_2 \end{bmatrix} \begin{bmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{bmatrix}^{-1} \tag{4.41}$$

Then, just as was done for the subsonic implementation described in Section 4.3, the node-based doublet influence coefficients are finally found using the process of Figure 4.5.

Chapter 5

RESULTS

Since this work was developed and implemented in two distinct phases as discussed in Chapter 4, it was also tested, validated, and verified as such. The higher-order subsonic method utilizing a linearly varying doublet distribution was validated against the original lower-order method of CPanel, which itself has been validated against other widely used codes [10]. This was done only for the case of non-lifting flows since the ability to use a wake with the linear doublet method was not implemented as a part of this work.

The supersonic implementation was extensively unit tested throughout its development to ensure the proper functionality of the implemented methods, and to characterize the behavior of the fundamental integrals and influence coefficients for various panel-control point relationships. The key results from this unit testing are discussed in this chapter. Geometries that resemble common supersonic aircraft shapes were then tested and results compared with theoretical and analytical solutions, solutions from the MARCAP and PANAIR codes, and wind tunnel test data.

5.1 Subsonic Higher-Order Method

As has been stated, the logic of the higher-order method that was developed and implemented for this thesis can be applied to both subsonic and supersonic flows, though the ultimate goal for this work was to implement it for use in supersonic modeling. However if applied to subsonic flow, the functionality of the higher-order method could be validated, while avoiding the complexities of modeling supersonic flow. Because the crux of the panel code process is the calculation of the influence coefficients, which is completely different between subsonic and supersonic applications, the subsonic results assessment presented here is more qualitative than quantitative since it was purely the successful enforcement of doublet strength continuity that was of interest.

To demonstrate the functionality of the higher-order method, results from the simple test case of non-lifting subsonic flow over an ellipsoid are presented below. The geometry used for this test is the same as that used for the control point offset study which is shown in Figure 4.4. The flow conditions are given in Table 5.1.

Table 5.1: Subsonic test case flow conditions

Mach	0.1
V_∞ (ft/s)	20
α (degrees)	15
β (degrees)	10

Before discussing the results of this test case, the methods by which data is output from CPanel and visualized should be noted. Data is written to .vtu files where it can be associated with the panels or nodes of the geometry. For example, doublet strength and velocity potentials are associated with panels for the lower-order method, while they are associated with nodes for the higher-order method. Other quantities like source strength or velocity are associated with panels for both methods. For this work, the program ParaView was used to read the output data and visualize the results. To visualize node-based data across the surface of a given geometry, ParaView performs linear interpolation of the data across panels, which can be used to demonstrate the linear variation of doublet strength or velocity potential.

Using this visualization method, the successful enforcement of doublet strength continuity is demonstrated by Figure 5.1(b) via velocity potential distributions. Figure 5.1(a) shows the distributions for the lower-order solution. These figures show the full range of the solved velocity potentials, where the magnitude of the minimum and maximum was found to be 48.4 using an exact ellipsoid solution generated by McDonald in MATLAB (Rob McDonald, personal communication, September 28, 2018). The continuity of velocity potential in Figure 5.1(b) is easily observed in comparison to Figure 5.1(a). The lower-order solution shows that velocity potential discontinuously changes from panel to panel; in supersonic flow, each one of these discontinuities would introduce error to the solution and possibly lead to divergence of the solution.

Figure 5.2 shows a narrower range of velocity potentials for the same solution using both subsonic methods. The difference in methods is more easily observed here. Figure 5.2(b) shows a near constant width band of velocity potential values about zero. It also shows a smooth transition into and out of this band on either side of zero. This band is not as distinguished in Figure 5.2(a). The location of zero

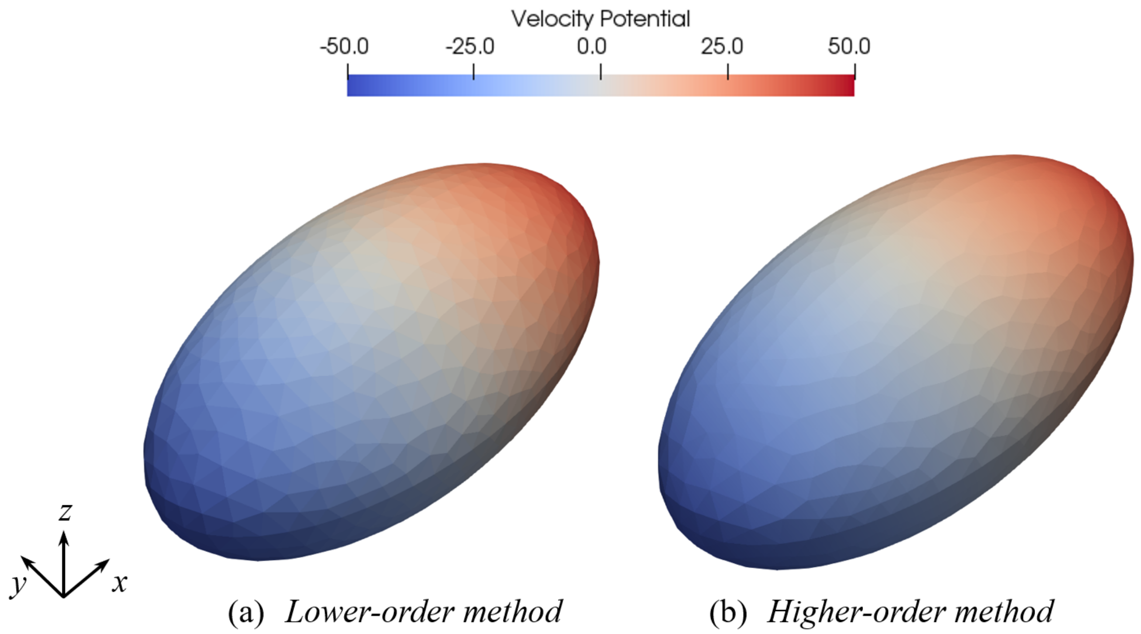


Figure 5.1: Full range of velocity potential distributions for subsonic ellipsoid test

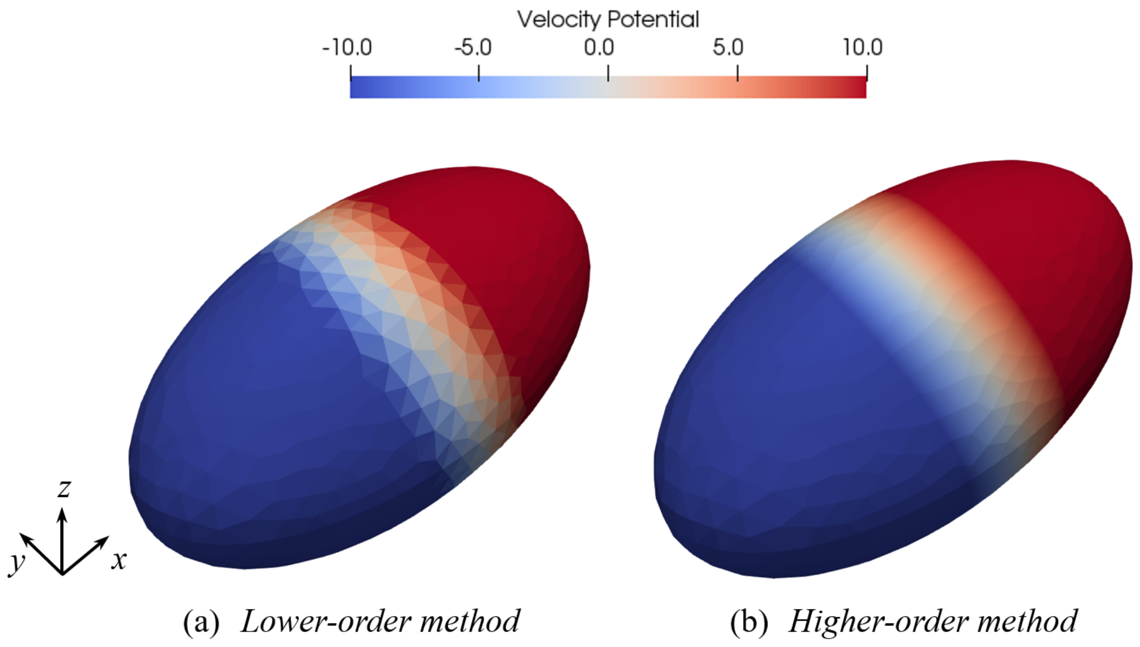


Figure 5.2: Narrowed range of velocity potential distributions for subsonic ellipsoid test

velocity potential waivers back and forth, and the transitions on either side of zero are ragged. These results show the necessity of doublet strength continuity enforcement in supersonic modeling, as well as that higher-order methods can more accurately model a geometry given the same surface discretization.

5.2 Supersonic Method

Besides the unit testing that was done, test cases for the supersonic scheme were chosen based on both the availability of solution data and the generality of the test cases; generality of test cases meaning that the tested geometries are generally representative of common supersonic aerodynamic shapes. Since MARCAP is the code that most resembles the methods implemented here, the solution data presented by Maruyama et al. [7] acted as the primary source of verification. A few of the generic geometries tested there–cones and rectangular wings–were generated and also tested in CPanel. Maruyama et al. compared their results with analytical solutions and solutions from PANAIR; this same data is also shown and discussed here.

Various delta wing geometries were also tested. CPanel results in the form of pressure, lift, and drag coefficient are compared against results from the PANAIR pilot code, theoretical solutions, and wind tunnel test data. Wing-body combinations were not tested since, as of this work, CPanel does not properly handle wing-body intersections. This causes spurious vortices to occur at these intersections. The affect of these vortices dies out in the subsonic solution, however they propagate downstream and are magnified in a supersonic flow field which can blow up the solution. However when this issue is resolved, such modeling will be possible with supersonic flows; this specific topic is discussed in more detail in Appendix C.

There is no dedicated discussion in relation to mesh fineness or mesh convergence since it was found that CPanel solutions for supersonic flows were accurate using relatively coarse meshes, so long as the mesh properly represented the geometry. Since the test cases presented in this chapter–as well as supersonic geometries in general–have either no or shallow curvature, relatively few panels are needed to correctly represent the geometries. Furthermore, because a higher-order method is used, velocity potential variation across geometries is accurately captured with few panels, and increasing the number of panels negligibly improves the solution. Similar conclusions were found by Ehlers et al. and Maruyama et al. [2, 7]. Brief discussions in regard to meshes are given for each geometry.

5.2.1 Unit Testing

Each of the algorithms presented in Chapter 4 were tested independently and together in a simplified routine built in MATLAB which was specifically developed for such testing. Of specific interest during these tests was how a panel singularity element influences a control point as the control point's domain of dependence intersects the panel in different ways. Assessing this behavior served multiple purposes. Along with acting as a means of validation for the influence coefficient calculation routines, it made clear the relationship between the computed influence coefficients and the geometry of a given panel-control point pair, as will be shown in the following figures. The results of this testing also motivated the eventual layout of the algorithms themselves primarily in regard to ensuring robustness. An example of the geometry that was used through out all unit testing is shown in Figure 5.3. As previously noted, the integration direction around the panel is counterclockwise. The edge numbering shown here will be referenced in the following discussions.

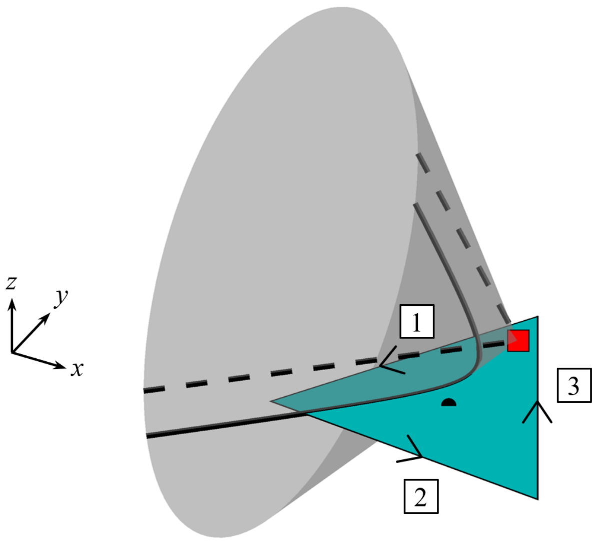
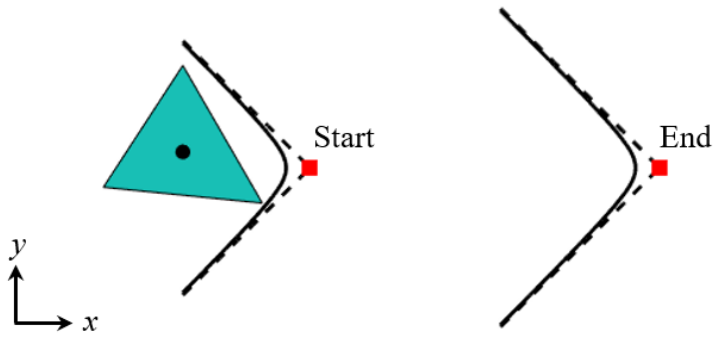
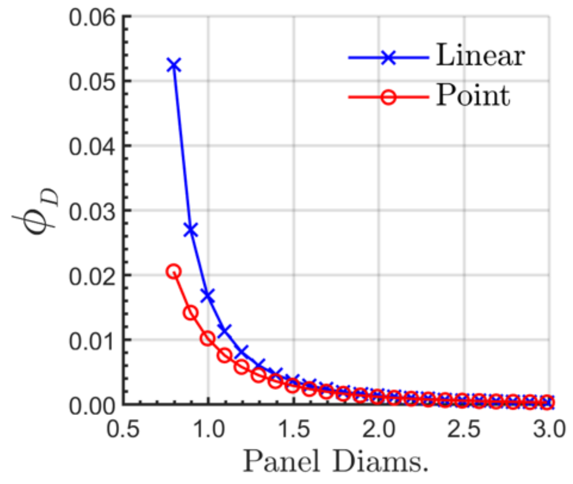


Figure 5.3: Example of geometry used during unit testing

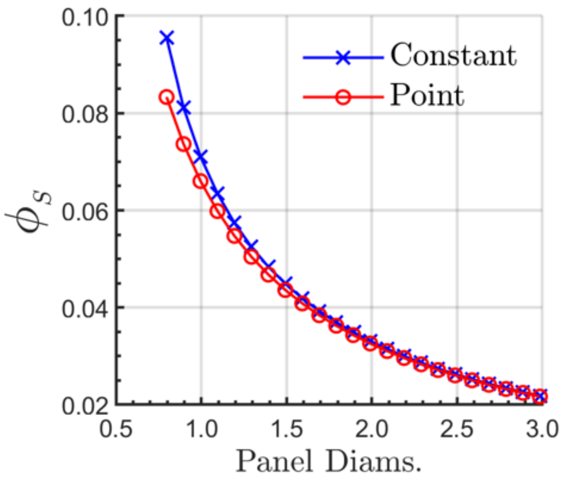
The first test was a simple comparison of panel singularity element influence against the equivalent point singularity element influence for both a source and doublet. In all scenarios, as illustrated in Figure 5.4(a), the control point was moved away from the panel in the x-direction and the potential influence was computed as the control point moved. For both the linear doublet panel and the constant source panel as shown in Figures 5.4(b) and 5.4(c), respectively, the panel and point singularity influences converge to each other and toward zero as the control point moves away,



(a) *Geometry*



(b) *Doublet Potential Influence*



(c) *Source Potential Influence*

Figure 5.4: Comparison of potentials against point singularity elements

which acts as validation for each respective calculation.

Notice that in Figure 5.4(a), the panel is inside the domain of dependence of the control point for the entire span of the test. This was done intentionally since the point singularity element calculation cannot be compared with the panel calculation if the domain of dependence was to intersect the panel. With the scenario for the panel entirely inside the DOD of the control point validated, cases for the DOD intersecting the panel were then tested three different ways. Each of these tests were performed in very similar manners. For each case, the control point starts at a point in space such that the panel is completely outside its DOD, and thus the panel has no influence on it. Then, similar to the test of Figure 5.4, the control point moves in a single coordinate direction with respect to the panel, and stops once the DOD is no longer intersecting the panel. This is done for each coordinate direction.

For the purposes of this unit testing study, different 'States' of panel-DOD intersections are defined where each state represents how the DOD is intersecting the panel. A different set of states is used for each of the three tests, which are defined in Tables B.1, B.2, and B.3, respectively. The states are defined because the behavior of a panel's influence on a control point directly depends on the state of the panel-DOD intersection, which is shown in the results presented in the following figures.

The results of the first two tests are shown in Figures 5.5 and 5.6, which involved moving the control point in the x- and y-directions, respectively. Both doublet and source potential influences are shown, with the axis for the former on the left of the figures, and the axis for the latter on the right. First assessing the doublet potential influence of Figure 5.5, there is a sudden jump in potential at the transition from State 1 to 2. A jump in potential occurs and not a gradual increase because the control point is immediately inside the Mach wedge of Edge 1 when the DOD first intersects the panel. This is in contrast to the transition from State 1 to 2 in Figure 5.6 where the doublet potential influence gradually and continuously increases as the control point moves through State 2. As the control point in Figure 5.5 moves into State 3, where the shared vertex of Edges 1 and 2 is now inside the DOD, a small change in the doublet influence curve is observed.

Moving from State 3 to 4 in Figure 5.5, there is again an instantaneous jump in doublet potential influence. This jump occurs because the third edge has entered the DOD of the control point, and since the direction of integration across this edge is the opposite of Edges 1 and 2 with respect to the y-axis, the sign of its influence is switched. This same phenomenon is observed in Figure 5.6 in the transition from

State 2 to 3. The remaining states of both Figures 5.5 and 5.6 are similar in that the rest of the changes simply involve the panel vertices going into and out of the control point DOD. The two tests do however end differently. The final state shown in Figure 5.6 is the same as the starting state, where the panel is completely outside the DOD, though this is not shown in Table B.2. For Figure 5.5, the panel never leaves the DOD and thus the doublet potential influence approaches zero as the control point moves away. Now assessing the source potential influence, its behavior in these two tests is very similar. In both cases, it gradually increases from zero as the panel enters the DOD, and gradually approaches zero as the control point moves away from the panel. It is also seen that the source potential influence peaks at the same state change in both instances, which is when Edge 3 enters the DOD. This makes intuitive sense

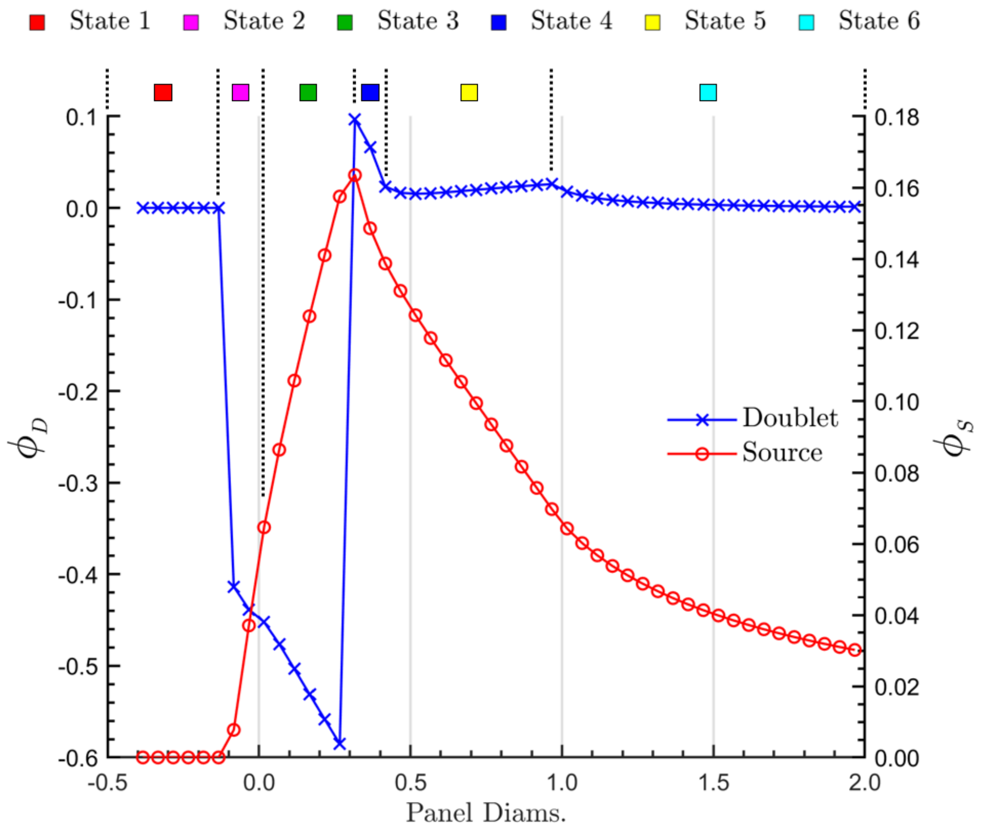


Figure 5.5: Doublet and source potential influences as the control point varies in the x-direction (State details in Table B.1)

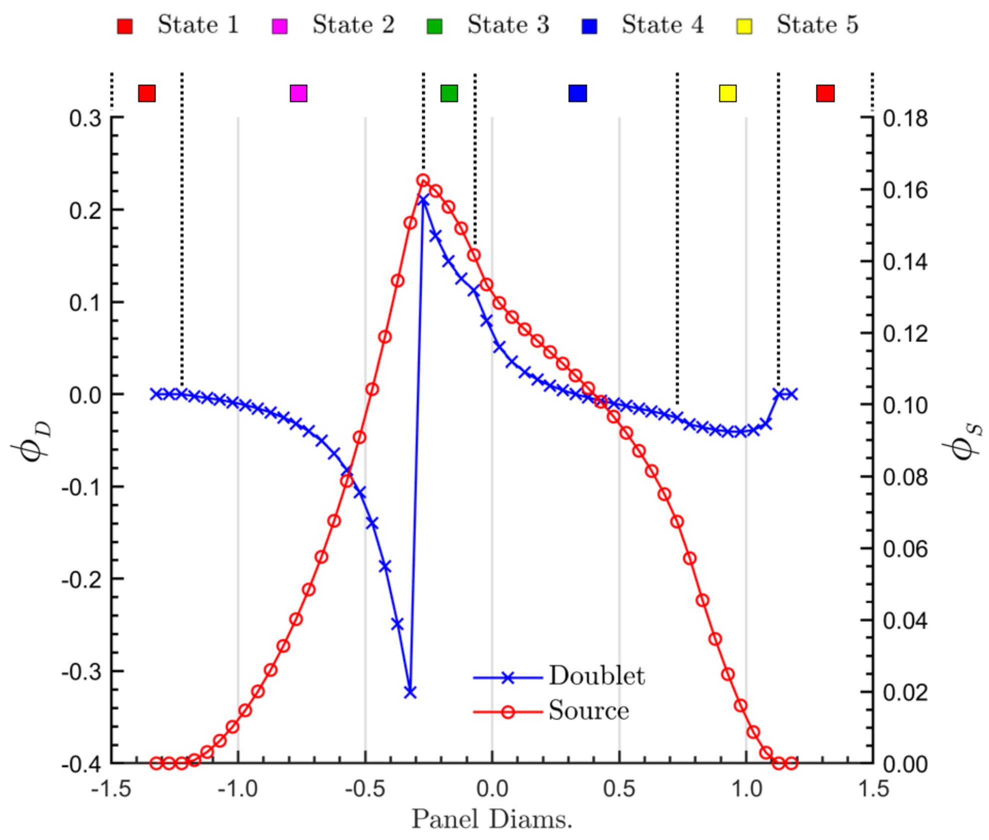


Figure 5.6: Doublet and source potential influences as the control point varies in the y-direction (State details in Table B.2)

because this is the instant that the balance of control point proximity to the panel and the area of the panel inside the DOD is at its maximum.

The final test shown here is the case of moving the control point purely in the z-direction relative to the panel, for which the states are shown in Table B.3 and the results in Figure 5.7. In this test, the control point started below the panel and moved in the positive z-direction. The doublet and source influence curves are seen to behave similarly to the prior two tests, except that there is now also symmetry in their behaviors as the control point moves. The doublet influence curve is antisymmetric across the y-axis of the plot, while the source influence curve is symmetric across the y-axis. This behavior is expected since the only difference in geometry between the left and right sides of the plot is the sign of the control point z-coordinate.

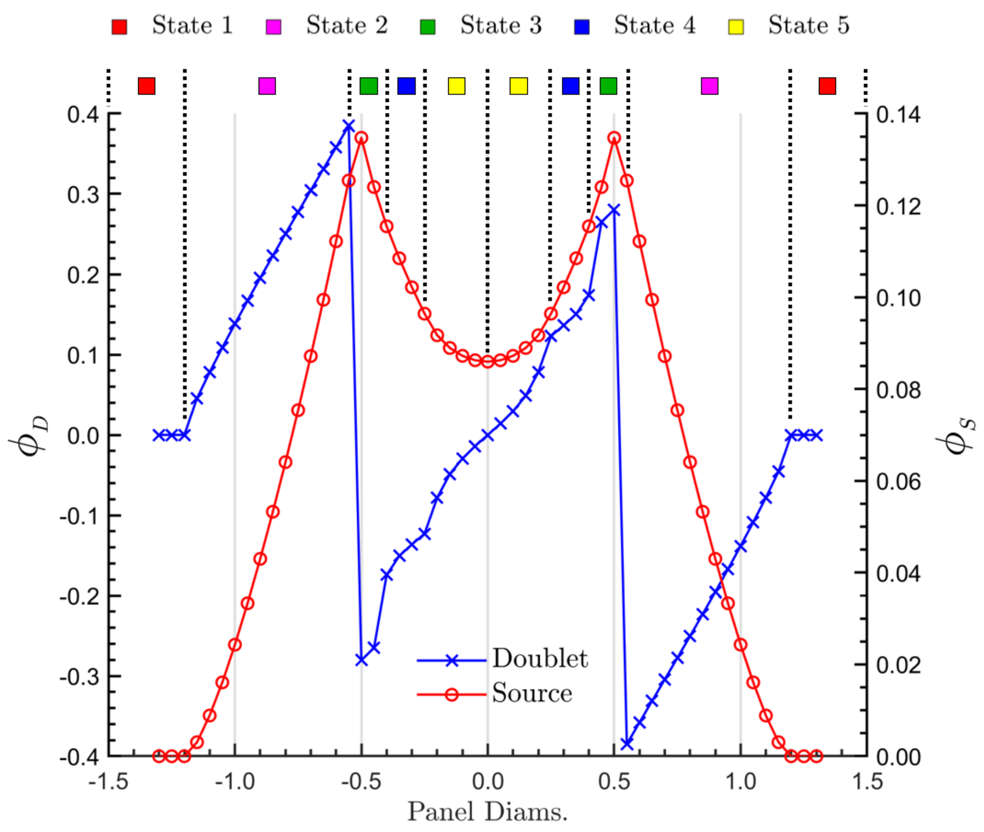


Figure 5.7: Doublet and source potential influences as the control point varies in the z-direction (State details in Table B.3)

5.2.2 Flows Around Cones

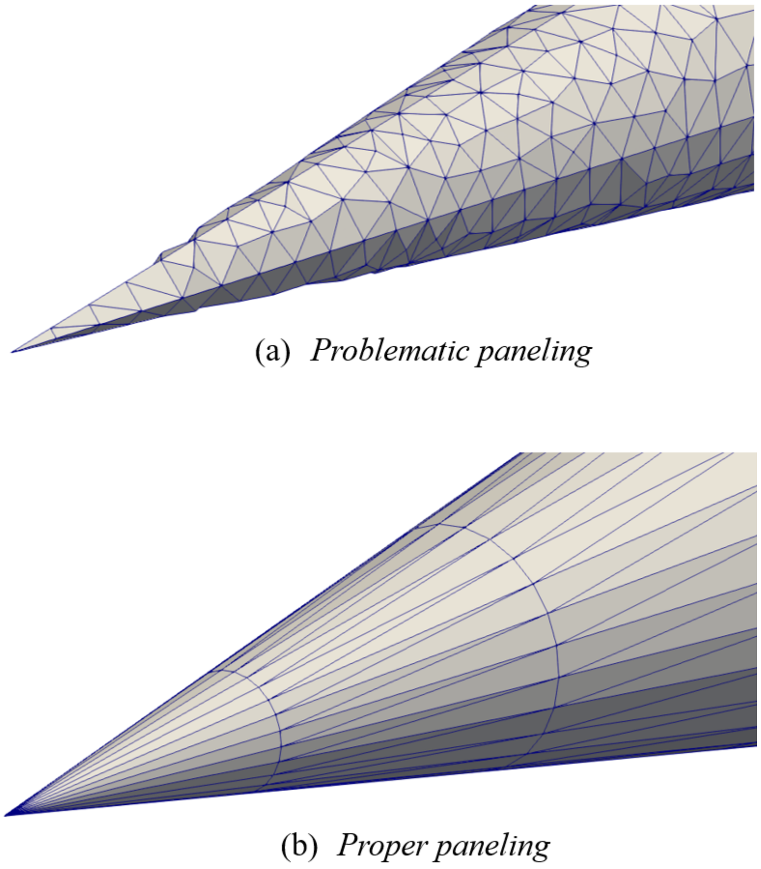
The ability to model the aerodynamic characteristics of a cone is a staple for any supersonic flow modeling code. The geometry of a cone resembles the shape of common supersonic vehicle fuselage designs, such as fighter aircraft, and is thus a simple benchmark test case for the code's ability to model a generic supersonic fuselage shape.

It was found in early supersonic tests of CPanel that high quality paneling is essential to solution accuracy. This is generally the case for all codes but for supersonic panel codes specifically, if the paneling of a model inaccurately represents the true geometry, then there is a risk that non-physical flow perturbations will occur and propagate downstream, causing errors in the results. Such errors were found to occur when the unstructured paneling methods conventionally used with CPanel were used to panel cones. The two methods of mesh generation used for the developments of CPanel v1.0 and v2.0 were through OpenVSP and a MATLAB program called DistMesh developed by two UC Berkeley faculty members [22]. Both of these programs were first used to generate paneling for cones to be tested here, however both were found to struggle in paneling the surface near the tip of the cone, an example of which is shown in Figure 5.8(a). Since this is the forward most point of the geometry, the resulting solutions were substantially in error.

This same issue is likely to be seen when meshing other pointed and slender bodies, so when utilizing unstructured mesh generating programs for such bodies in supersonic flows, the CPanel user must be cognizant of this. For more complex geometries where this may be an issue, it is recommended that more advanced meshing programs are used than the ones conventionally used with CPanel.

In order to construct a surface mesh of a cone of acceptable quality for the purposes of testing CPanel here, a MATLAB routine was created whose specific function is to generate high quality triangular paneling of cones. An example of the paneling generated by this program at the tip of a cone is shown in Figure 5.8(b). Given a cone length and half angle, the routine generates what could be considered a structured mesh based on the desired number of azimuthal slices and longitudinal slices input by the user. It then outputs the surface discretization in a .tri file format which is the standard file format used by CPanel.

Using this specialized cone paneling program to generate meshes, three different test cases were run involving cone geometries, all of which were also run by MARCAP and PANAIR. The first test is of a cone with a 10° half-angle at no angle of attack or sideslip, and at varying freestream Mach numbers. Figure 5.9 shows the results from this test in the form of C_p vs Mach number. The three different approximations of



(a) *Problematic paneling*

(b) *Proper paneling*

Figure 5.8: Examples of paneling at the tip of a cone

C_p presented in Section 3.3 are all plotted where the '2', 'S', and 'L' in the legend correspond to the 2^{nd} order, Slender, and Linear approximations, respectively. This nomenclature is used throughout the chapter. The same three C_p approximations presented in Maruyama et al. [7] for MARCAP are shown.

Both the 2^{nd} order and Slender approximations from CPanel trend closely with those of MARCAP. The Linear approximation is the only one that would be considered to not match the results of MARCAP. However, there appears to be an unexplained shift for the center portion of the Linear C_p data from MARCAP relative to the other MARCAP data points, and relative to the shape of the curve created by the CPanel data. A similar shift is seen in the last two points of the MARCAP 2^{nd} order data. The exact reason for this phenomenon in the MARCAP data is unknown;

it is hypothesized that it is mesh related. If this shift is disregarded for the Linear data, then the CPanel and MARCAP data would be considered to match one another with the CPanel results being slightly less accurate. The Slender C_p approximation is clearly the most accurate in this instance. The CPanel results nearly match and are actually slightly more accurate than the MARCAP results, which both trend almost exactly with the results of PANAIR. As expected, the results of all three programs begin to diverge away from the analytical solution as Mach number increases and the linearized assumption begins to become invalid. This occurs around Mach 2.5 in this test case.

One final observation to be made from Figure 5.9 is that the error bars for the first two CPanel data points are of substantial span. These error bars represent three standard deviations of the C_p data taken across the length of the cone. Since the cone is at no angle of attack or sideslip, C_p should be the same all over the surface

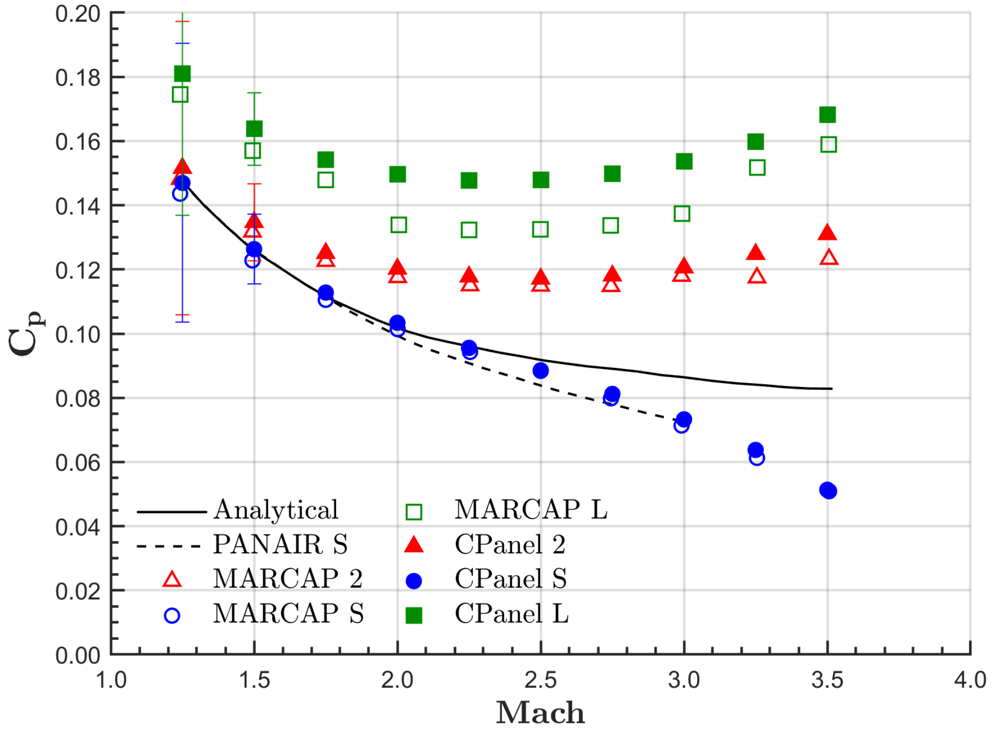


Figure 5.9: 10° **half-angle cone,** C_p **vs Mach**

of the cone, so any variation here is captured by the error bars. Appreciable error is observed only at the two lower Mach numbers of this test. This is because these Mach numbers are approaching the transonic flow regime which linear theory does not properly model, so the solution becomes unstable as Mach number approaches unity.

The second test case involving flows around cones presented here is similar to the first except now Mach number is held constant while cone half-angle is varied, represented as the variable, θ. The test is run at Mach $\sqrt{2}$ and again with no angle of attack or sideslip. Figure 5.10 shows the results of this test case as well as the results of MARCAP and CPanel just as was done in Figure 5.9. Similar conclusions can be drawn from this plot as were drawn before. The Slender C_p approximation is again clearly the most accurate, and the results of all three programs using this approximation trend together as θ increases. Furthermore, the Linear and 2^{nd} order approximations from CPanel and MARCAP also trend together and are again the less accurate C_p approximations.

Three standard deviations are again shown via error bars for each data point from CPanel. For reference, the standard deviation at $\theta = 10°$ here is approximately the same as that for the Mach 1.5 case of Figure 5.9. As θ increases, the standard deviation also increases as expected. At the higher θ values, the error bars span a substantial range of C_p values, even though the mean value that is plotted remains spot on with the MARCAP and PANAIR data. This indicates that if specific surface data of a case such as this is desired, it may be in error, while the averaged aerodynamic characteristics remain accurate.

The results of the two above test cases presented in Figures 5.9 and 5.10 provide the future CPanel user a handful of key insights in regard to modeling cone like geometries at supersonic speeds. As already discussed, the paneling must be of a

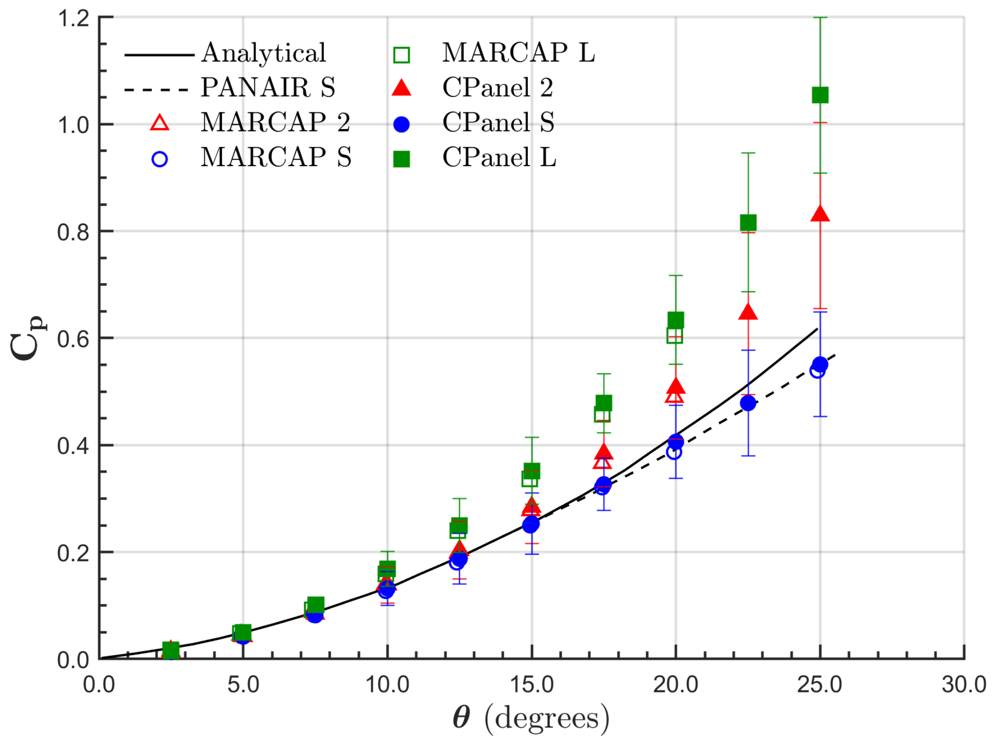


Figure 5.10: Cone at Mach $\sqrt{2}$, C_p vs θ

high quality, and specifically so in the forward-most regions of the geometry. The user must also be aware of the combination of body thickness and Mach number, since if either is too large, the linear assumptions of the method begin to break down yielding inaccurate results; however, the data has been shown to be inaccurate in a somewhat predictable manner so insight can still be gained from results at high Mach numbers or from modeling thick bodies. Lastly, the Slender C_p approximation is without a doubt the most accurate of the three approximations in these test cases, and should thus always be used in assessing flows involving conical shocks.

Modeling a cone at an angle of attack is the third and final test case to be presented in this section. The same geometry and paneling as was used for the first test case with varying Mach number is also used here. This test is run at Mach 1.5 and with the cone at an angle of attack of 5°. Figure 5.11 shows the results of this test in a plot of C_p vs ϕ, where ϕ here represents azimuth angle around the cone, as illustrated

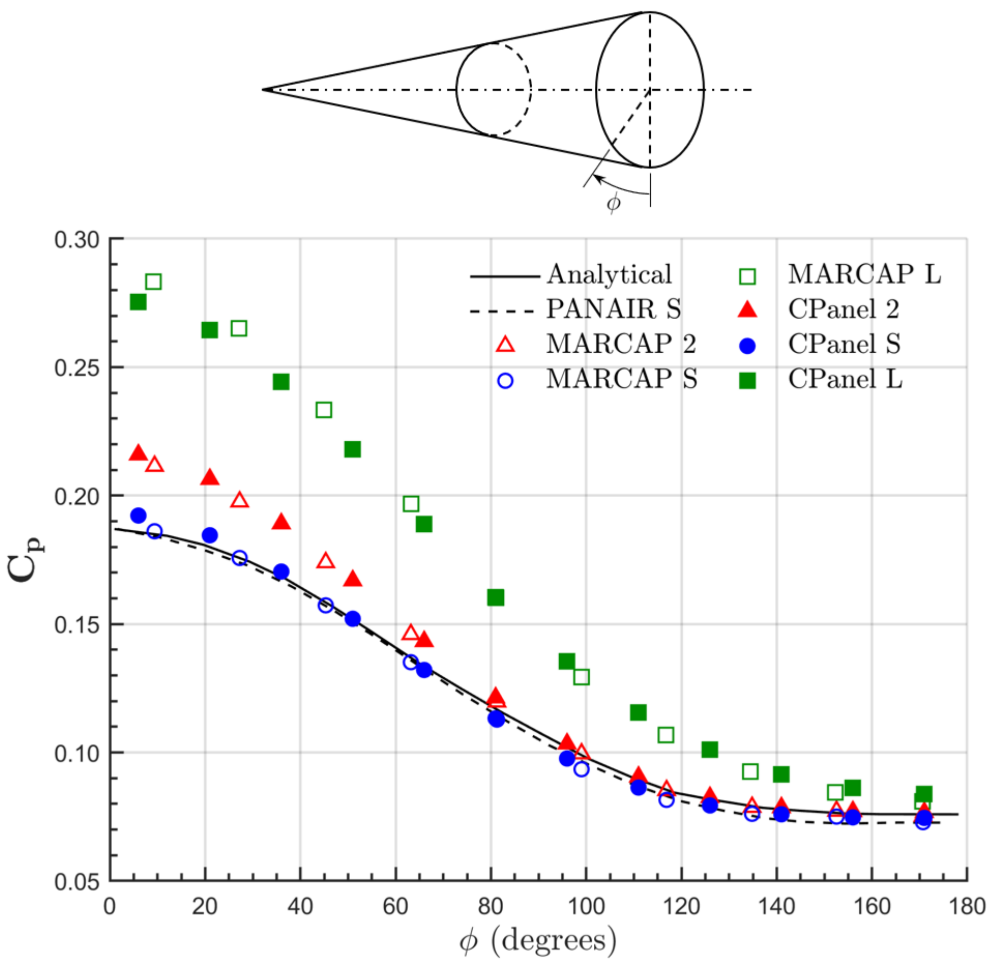


Figure 5.11: 10° **half-angle cone at Mach 1.5 and** $\alpha = 5^\circ$**,** C_p **vs** ϕ

at the top of Figure 5.11.

CPanel results for this test again match the results of MARCAP, and the Slender C_p approximation matches PANAIR and the analytic solution. Through these tests, CPanel has been shown to have the ability to properly model supersonic flows around cones, and can thus be extended for use on generic supersonic fuselage shapes. There are limitations in regard to paneling quality, geometry thickness, and freestream Mach number, but these limitations are expected, and CPanel behaves appropriately as these limits are approached.

Furthermore, it was found that the cone solutions retained the same level of accuracy, such as that shown in Figure 5.11, regardless of the fineness of the mesh so long as the discretization reasonably represented the cone. Accurate solutions were found with as few as 100 panels with nearly negligible gains in accuracy with finer meshes for the reasons discussed at the beginning of Section 5.2, which is in contrast to CPanel's lower-order method as shown by Satterwhite [10]. To obtain high resolution data however, a finer mesh is needed.

5.2.3 Diamond Airfoil Rectangular Wings

The next set of test cases used to verify the results of the supersonic implementation of CPanel was the modeling of flows over rectangular wings with diamond airfoils. This is a favorable geometry to test with for multiple reasons. Modeling a rectangular wing in supersonic flow guarantees that no part of the wing will be influenced by its wake, as was illustrated in Figure 2.5(a). Since wake modeling was not implemented as a part of this work, this is necessary in acquiring accurate results for these test cases. Furthermore, a diamond airfoil wing is representative of generic supersonic wing designs, which are often derivative of this basic shape. Lastly, modeling this geometry allows for the computed solution to be compared with 2D shock-expansion theory, since the flow over any region of the wing that is outside the influence of the wing tips will behave two-dimensionally.

The specific wing geometry used for the first test case in this section matches a geometry for which results are published by both MARCAP and PANAIR. As illustrated in Figure 5.12, this is a diamond airfoil with a half-angle of 6°. Also shown in the figure is the span location where data is taken, relative to the tip of the wing, which is outside the influence of the wing tip and can thus be treated as a 2D airfoil at this location. The wing was modeled at no angle of attack or sideslip for this test.

Results of CPanel are compared against the theoretical shock-expansion solution

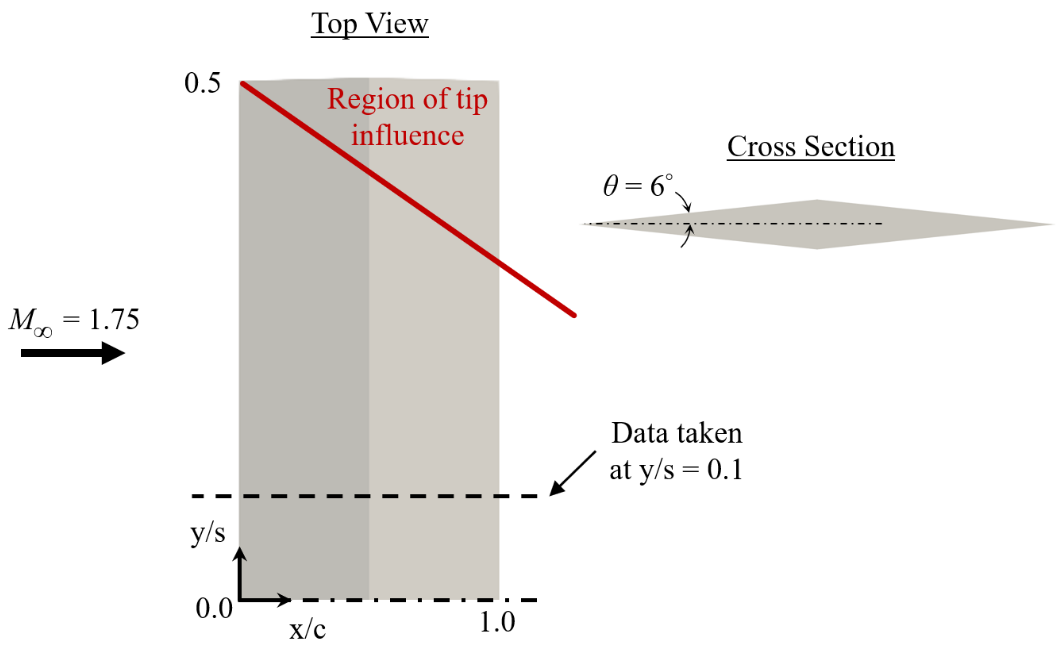


Figure 5.12: Geometry and conditions for diamond airfoil test case

for this airfoil in Table 5.2 and Figure 5.13. The 2^{nd} order and Linear C_p approximations yielded the same results for this test case, so only the results from the Linear approximation are shown and discussed here for clarity. All assessments of the Linear approximation also apply to the 2^{nd} order approximation. Figure 5.13 shows that there is no discernible variation in C_p across the forward or aft ramps of the diamond airfoil. This observation was true independent of the number of panels that spanned

Table 5.2: Percent error of CPanel C_p against theoretical C_p for a diamond airfoil

	C_p		% Error	
x/c	0.0-0.5	0.5-1.0	0.0-0.5	0.5-1.0
CPanel Slender Avg.	0.158	-0.136	4.9	5.6
CPanel Linear Avg.	0.173	-0.128	4.3	0.9
Theoretical	0.166	-0.129		

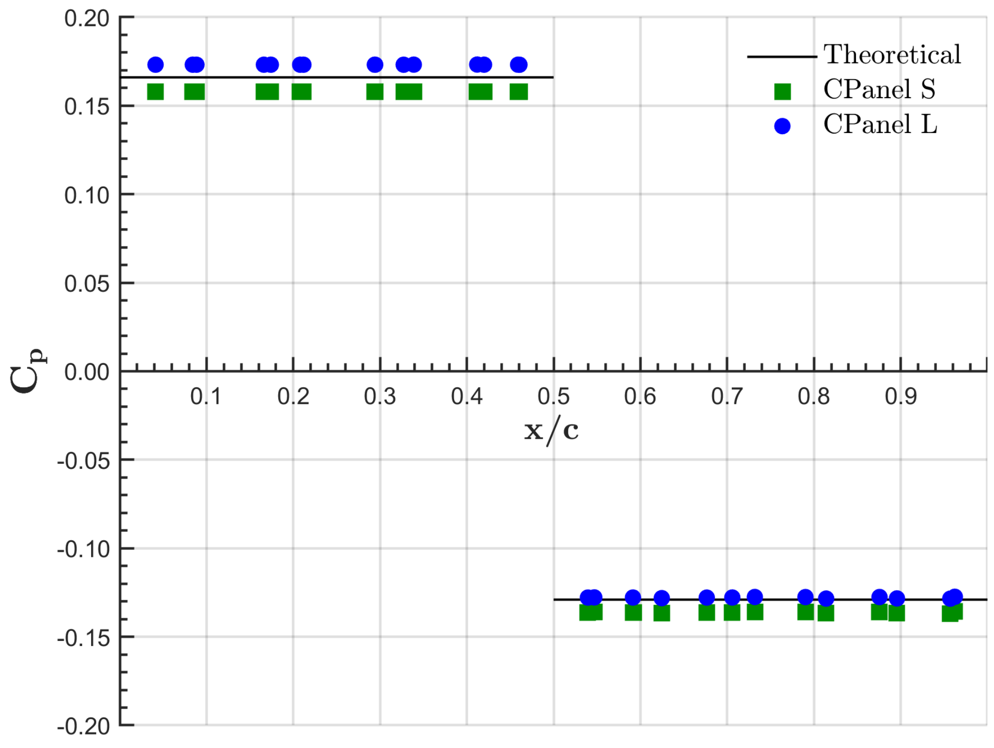


Figure 5.13: Diamond airfoil, C_p vs x/c – theoretical solution comparison

each respective ramp, as long as that number was at least three panels. For the case discussed here, five panels spanned each half-chord ramp of the airfoil.

Table 5.2 and Figure 5.13 also show that the Linear approximation of C_p is more accurate than the Slender approximation. Recall from Section 3.3 that the Slender C_p approximation gives the most weight to the v and w components of the perturbation velocity. Since these results are being compared with the 2D theoretical solution, if the v component of the perturbation velocity computed by CPanel is non-zero, then the results will be in error relative to the theoretical solution, which is what occurs here. Considering there will always be imperfections in any surface discretization and that CPanel uses unstructured paneling, it is expected that some non-physical y-component of velocity is found in a test case such as this.

The results of CPanel are compared against that of MARCAP and PANAIR in Table 5.3. The solutions from all codes show percent errors less than 6% relative to

Table 5.3: Comparison of CPanel error with MARCAP and PANAIR for a diamond airfoil

		C_p		% Error	
	x/c	0.0-0.5	0.5-1.0	0.0-0.5	0.5-1.0
Slender	CPanel	0.158	-0.136	4.9	5.6
	MARCAP	0.157	-0.134	5.5	3.8
	PANAIR	0.174	-0.125	4.6	2.8
Linear	CPanel	0.173	-0.128	4.3	0.9
	MARCAP	0.172	-0.125	3.6	3.3
	PANAIR	0.168	-0.123	1.3	4.5
	Theoretical	0.166	-0.129		

2D shock-expansion theory. This level of accuracy is as expected of panel codes, and CPanel is shown to give similar degrees of accuracy as the other two codes.

Similar to the studies that were done with the cone geometries, the effects of increasing Mach number and half-angle for a diamond wing airfoil were assessed. Even though no other codes have published the results for a study such as this, it is performed here to provide understanding of the limitations of CPanel in modeling generic supersonic wings. A diamond airfoil rectangular wing is again used, so the corresponding shock-expansion solutions can be computed and compared against; however, only the forward ramp of the airfoil is evaluated for these studies. Along with the pressure coefficients on the forward ramp, the theoretical entropy change, non-dimensionalized by the gas constant, across the compression shock at the leading edge of the airfoil is also plotted. The linearized equations from which panel methods are derived assume no entropy change throughout the flow field, so computing and plotting entropy change here enables a quantitative assessment of why the CPanel solutions become less accurate as Mach number and half-angle increase.

The first test is of a 5° half-angle diamond airfoil at increasing freestream Mach numbers and no angle of attack or sideslip, for which the results are shown in Figure 5.14. As was found earlier, the 2^{nd} order and Linear C_p approximations turn out to be the same, and it is now seen that this is regardless of Mach number. However they both begin to diverge away from the exact solution as Mach number increases, though they diverge slowly. The Slender approximation is less accurate than the other two for the lowest Mach numbers, but then consistently tracks closely with the exact solution through all the following test points. What this shows is that with a sufficiently thin airfoil, CPanel can give acceptably accurate results for a wide range of Mach numbers; the reason for this is shown by the plotted entropy change values, which are all small.

The results for the next test are plotted in Figure 5.15. This was run at Mach 2.0 and again with no angle of attack or sideslip. It is shown that in modeling wings,

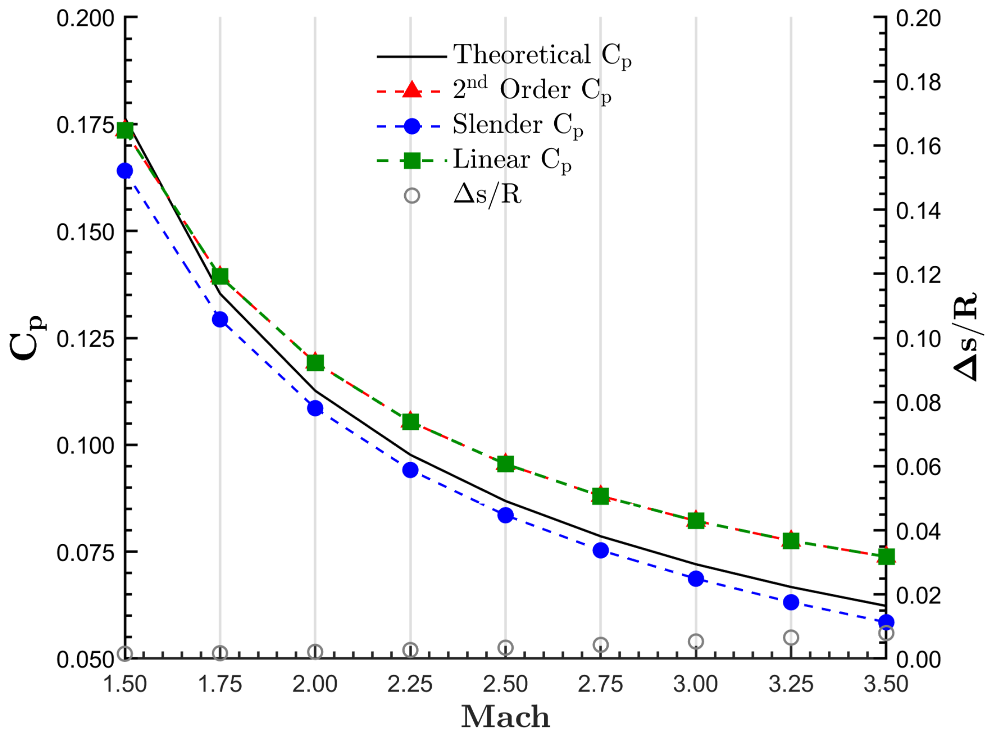


Figure 5.14: 5° half-angle wedge, C_p and $\Delta s/R$ vs Mach

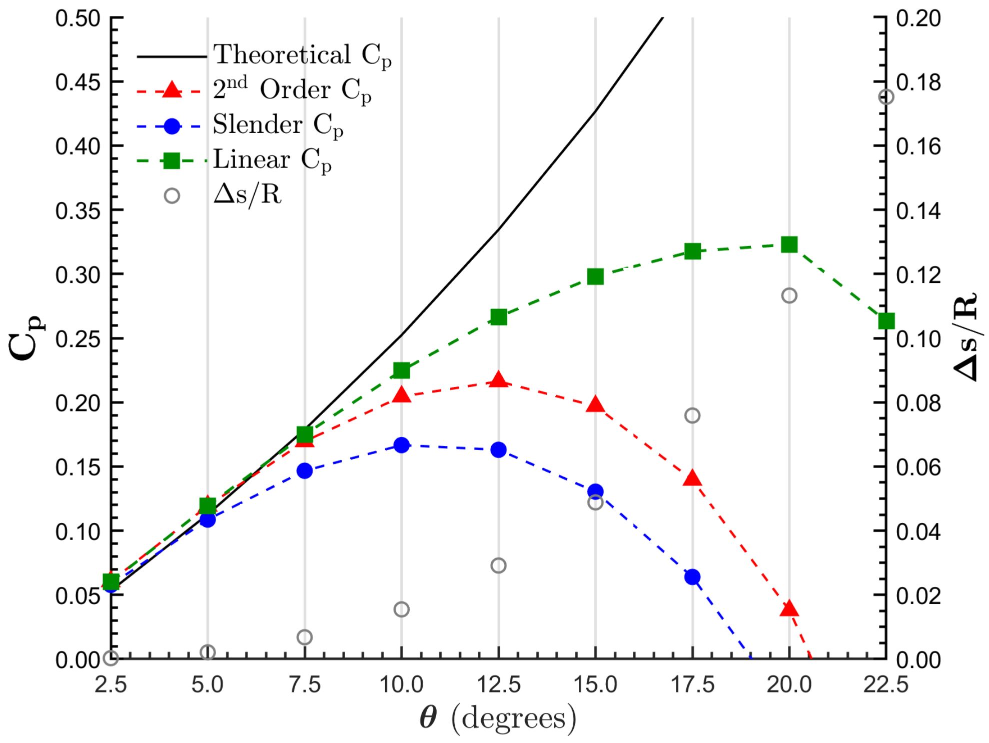


Figure 5.15: Wedge at Mach 2.0, C_p and $\Delta s/R$ vs θ

CPanel is much more sensitive to thickness than it is to Mach number when they are varied independently. The results here are accurate up until an airfoil half-angle of around 7.5°. Looking at the entropy change, it is already greater at $\theta = 10°$ than it was at Mach 3.5 in Figure 5.14, which explains the rapid divergence away from the theoretical solution. This same behavior would be expected from any linearized solution process.

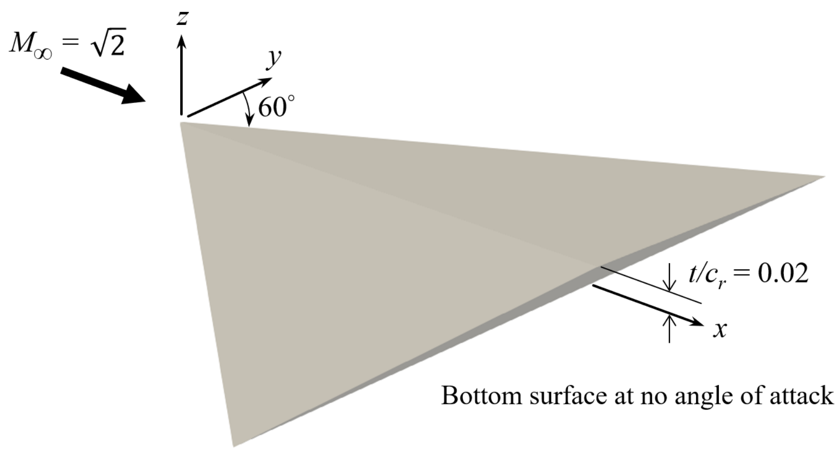
5.2.4 Delta Wings

The final set of test cases for this work was the modeling of various delta wing geometries. Delta wings were chosen because, like the diamond airfoil rectangular wing, they are representative of a wide range of supersonic wing designs; there are even entire aircraft designs that are delta wings or that are derived from them. Unlike rectangular wings however, the flow fields around delta wings are more complex in that there is three-dimensionality to them. Due to their ubiquitous use in supersonic aerodynamics, they have been studied and tested extensively, providing a plethora of resources to compare results against. Through prior tests, the Slender C_p approximation was found to be the most generally appropriate approximation, so this is what is used for the following tests.

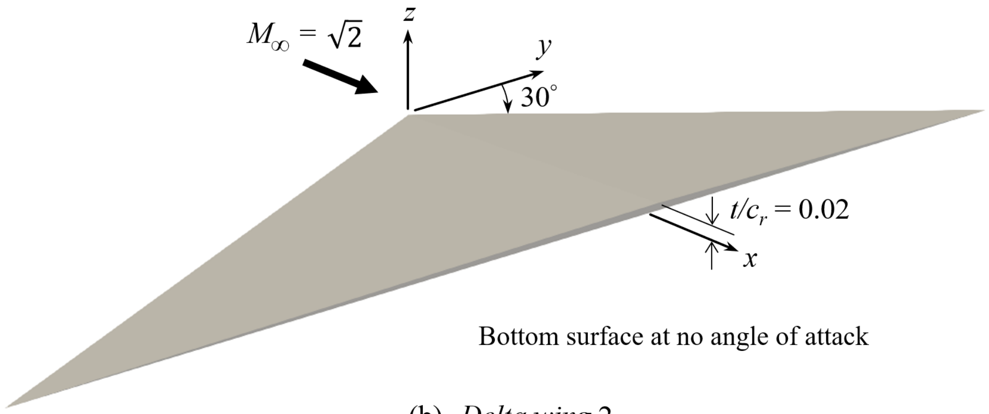
Figure 5.16 shows the delta wing geometries that were chosen to be tested in CPanel. Delta wings 1 and 2 shown in Figures 5.16(a) and 5.16(b), respectively, are geometries that were modeled by the PANAIR pilot code and results presented by Moran et. al. [23]. These wings were modeled at Mach $\sqrt{2}$, and with the bottom surfaces of the wings at no angle of attack. With a freestream Mach number of $\sqrt{2}$, the Mach angle is 45° for these two tests, which means Wing 1 has a subsonic leading edge and Wing 2 has a supersonic leading edge. Delta wing 3 in Figure 5.16(c) is modeled after a geometry that was used in a series of wind tunnel tests by Love [24].

One may notice that the aft ends of Wings 1 and 2 in Figures 5.16(a) and 5.16(b), respectively, are closed with a blunt trailing edge–in other words, the aft ends are closed with superinclined panels. It has been discussed that superinclined panels are not allowed as of this implementation of CPanel; however, an exception is made for special cases such as this. Because the trailing edge of both wings is a supersonic trailing edge and no part of the wing is downstream of the trailing edge in either case, the superinclined panels have no influence on the flow fields over either wing. Thus, the superinclined panels can be disregarded for the solutions of these wings, and the trailing edge treated as open. CPanel checks for such cases to allow the modeling of geometries like these; this is discussed further in Appendix E.

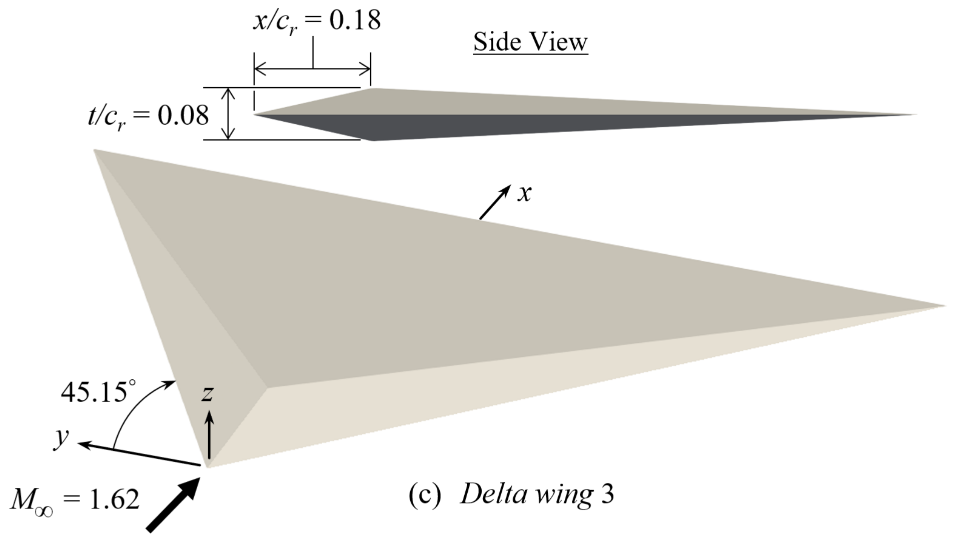
During preliminary delta wing tests, two different but related issues were found in regard to CPanel. Solutions were found to be in error when modeling geometries with sharp supersonic trailing edges at an angle of attack or with 3D flow phenomena–the



(a) *Delta wing* 1



(b) *Delta wing* 2



(c) *Delta wing* 3

Figure 5.16: Delta wing test cases

diamond airfoil rectangular wings discussed before do not encounter this error at no angle of attack for regions outside the wingtip Mach cones. Furthermore, the solutions of any wing with a subsonic leading edge were discovered to be in error. Both of these errors were found to stem from the improper enforcement of doublet strength continuity at these edges, where doublet strength is supposed to be discontinuous. The solution to these issues is a non-trivial one and was not implemented as a part of this thesis, though temporary workarounds were implemented to still allow for the geometries of Figure 5.16 to be assessed and results verified. These issues, their proposed future solutions, and the present temporary workarounds are all discussed in detail in Appendix C.

Since Wing 1 has a sharp subsonic leading edge, a temporary adjustment in the solution process was made such that the flow field over the top surface of the wing could still be solved. Refer to Appendix C for details of this modification. The pressure distribution for the top surface of Wing 1 solved in CPanel is shown in Figure 5.17. Pressure coefficients were taken at a cross-section of the top surface at $x = 0.9c$, and plotted in Figure 5.18 with results from the PANAIR pilot code and the theoretical solution, both obtained from Moran et. al. [23].

CPanel results track well with PANAIR results and the theoretical solution in Figure 5.18 until a y/y_{max} of approximately 0.75. The inaccuracy of CPanel relative to PANAIR near the leading edge is attributed to two sources. The first is the difference in method order between the codes where PANAIR is one order higher than CPanel which gives PANAIR an inherent accuracy advantage, and the second is CPanel's use of an unstructured surface discretization. As has been discussed, unstructured paneling will always introduce error that may not arise in a comparable structured paneling. An indication that error due to unstructured paneling is occurring in this solution is the wavering of the CPanel data points in Figure 5.18. The further downstream data is taken, the more prominent this error becomes which can be observed

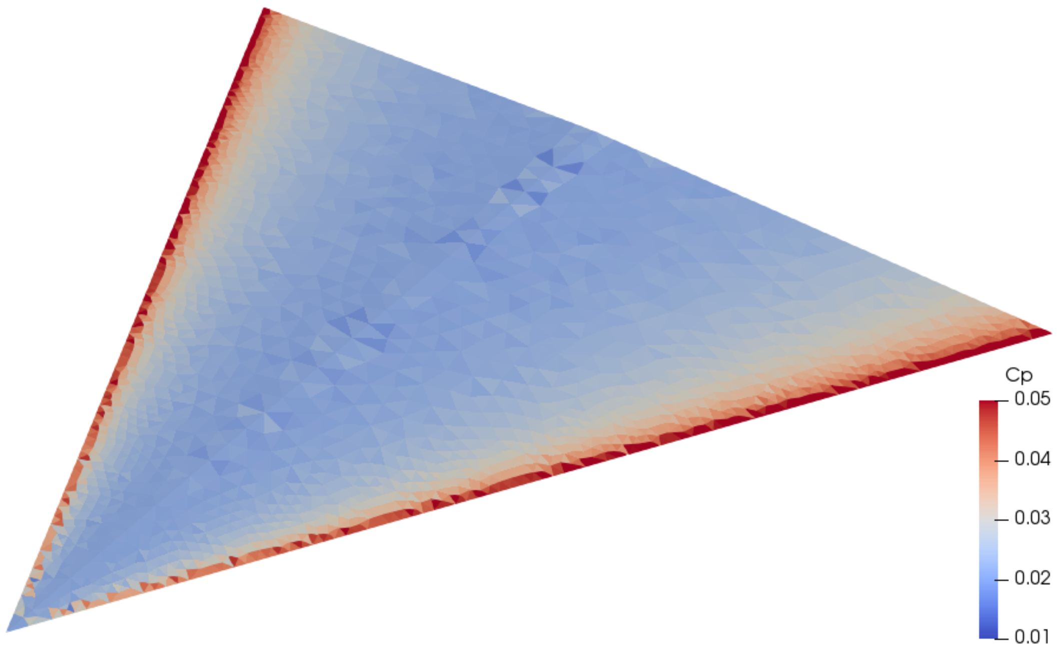


Figure 5.17: Delta wing 1 at Mach $\sqrt{2}$, C_p distribution

in Figure 5.17, specifically along the centerline of the geometry.

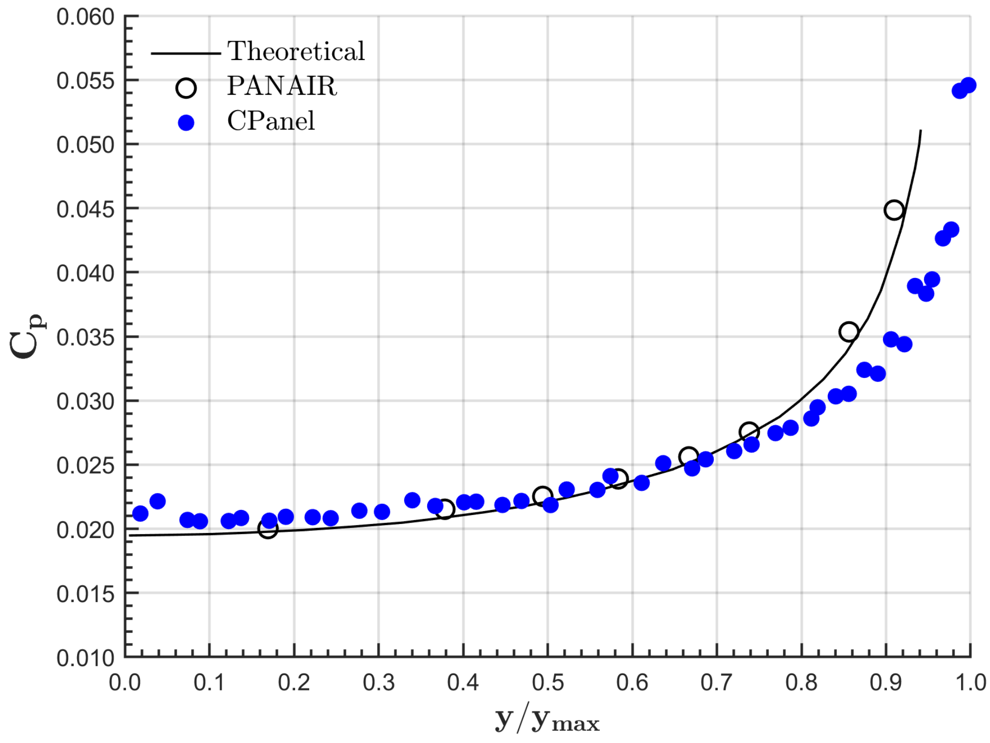


Figure 5.18: Delta wing 1 at Mach $\sqrt{2}$, C_p vs y/y_{max} at $x = 0.9c$

Though Wing 2 has a supersonic leading edge and a blunt supersonic trailing edge, there were still issues in modeling this geometry, however unrelated to the issues in modeling Wing 1. In the case of Wing 2, the solution is in error due to what seems to be an inability to accurately model the bottom surface of the wing, which is parallel to the freestream direction. The exact source of this error is unclear, though it has been determined that it is most likely due to either a mesh which does not accurately represent a perfectly flat bottom surface, or the unavoidable error introduced with unstructured paneling is of a large enough magnitude in this test to result in the entire solution being in error. Both of these possibilities are discussed in more detail in Appendix C.

For this paper, a small temporary adjustment was made to the code to give the proper solution for the top surface, similar to what was done with Wing 1. The pressure distribution for this surface is shown in Figure 5.19. This distribution distinctly shows the region of the surface which is within the Mach cone emanating from the tip of the wing. There are similarities between this distribution inside the tip Mach cone and the pressure distribution for Wing 1 in Figure 5.17, for which the entire wing is inside the tip Mach cone. The lowest pressure region for both wings is along the

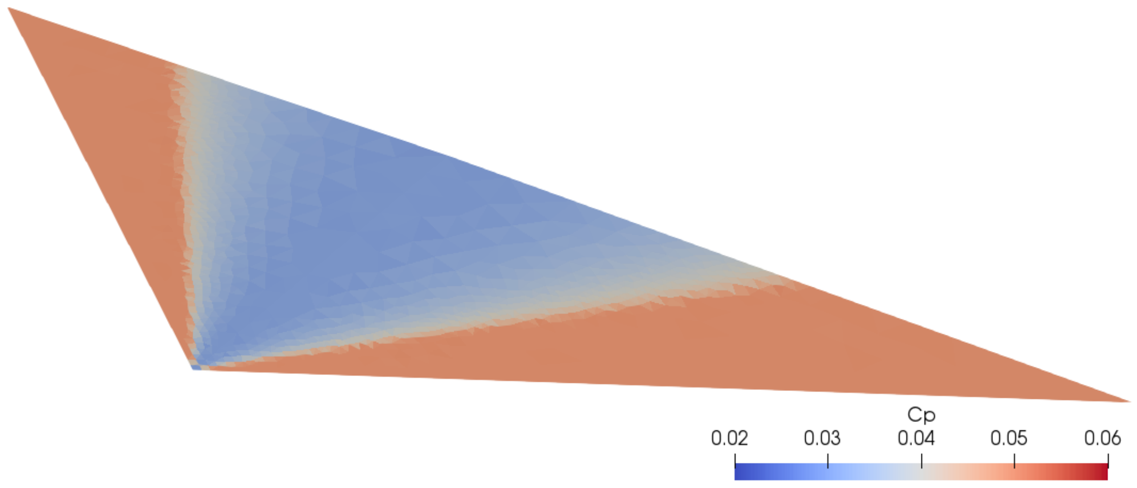


Figure 5.19: Delta wing 2 at Mach $\sqrt{2}$, C_p distribution

centerline, which then gradually increases moving outboard. However instead of the pressure approaching infinity like with Wing 1, it approaches the constant pressure of the surface that is outside the tip Mach cone.

As was done with Wing 1, pressure coefficients were taken at the $x = 0.9c$ cross-section, and plotted with the PANAIR pilot code results and theoretical solution, which is shown in Figure 5.20. CPanel results are shown to match both PANAIR and the theoretical solution. The similarities between the solutions of these wings discussed above is also observed in comparing the plots of Figure 5.18 and Figure 5.20.

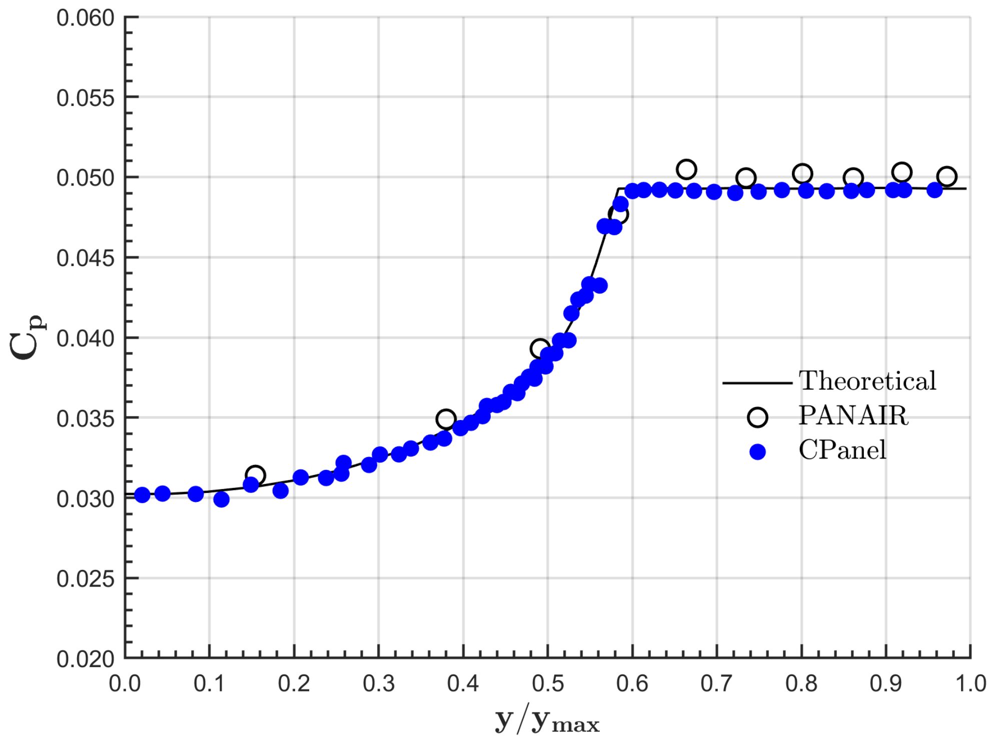


Figure 5.20: Delta wing 2 at Mach $\sqrt{2}$, C_p vs y/y_{max} at $x = 0.9c$

The final delta wing test case is that of Wing 3 shown in Figure 5.16(c). This geometry is one of many for which C_L and C_D data from wind tunnel tests are given by Love [24]. Of Love's tested wings, Wing 3 was chosen because it has a supersonic leading edge at the test Mach number of 1.62; and because it does not have any flat surfaces that are parallel to the freestream direction, neither of the modifications made for the previous two tests needed to be made here. A small modification in the post-processing at the trailing edge panels was needed however, since it has a sharp trailing edge in 3D flow and so CPanel incorrectly enforces doublet strength continuity here. The details of this modification are discussed in Appendix C.

Wing 3 was tested at Mach 1.62 and varying angles of attack, and C_L and C_D computed using CPanel's existing methods of summing each panel's contribution to the respective force coefficients. The pressure distribution of the top surface of Wing 3 at $\alpha = 0°$ is shown in Figure 5.21. This shows the Mach cone emanating from the tip of the wing, as well as the Mach cone emanating from the tip of the expansion turn at the maximum thickness location of the wing. Though it is not shown here, the pressure distribution on the bottom surface appears identical.

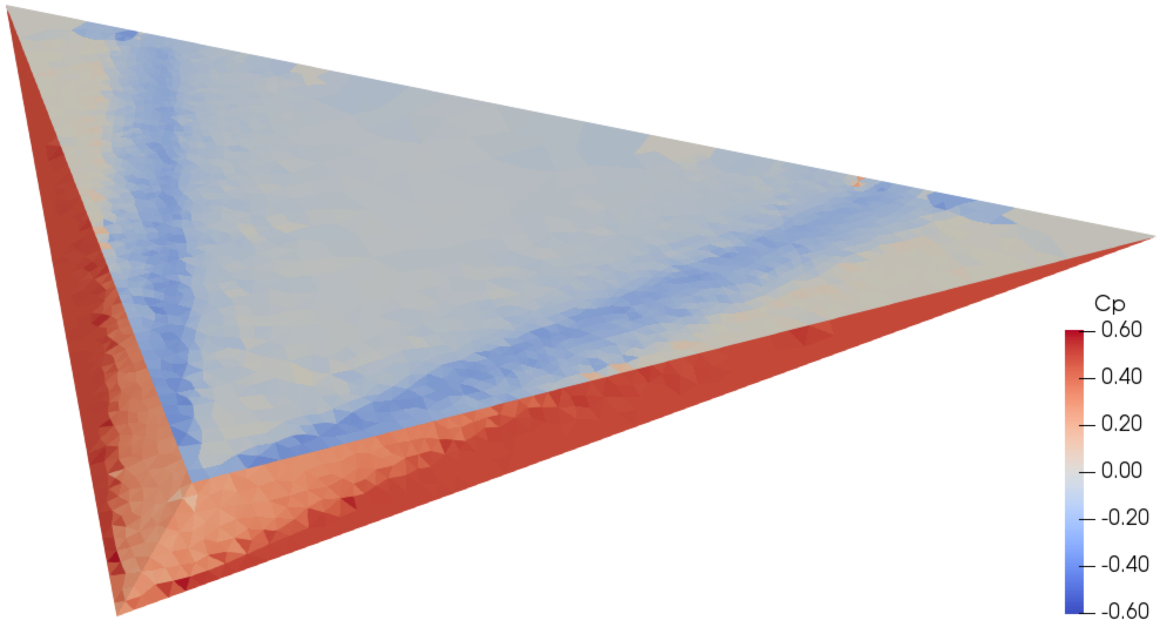


Figure 5.21: Delta wing 3 at Mach 1.62 and $\alpha = 0$, C_p distribution

C_L and C_D computed by CPanel are plotted against the results of Love in Figure 5.22. CPanel matches the wind tunnel test results of Love for the lower angles of attack in the range shown. At higher angles of attack, CPanel underpredicts C_L, which is expected of a linearized method. As the angle of attack increases, the bottom surface leading edge shock increases in strength. Since linearized methods assume no entropy change in the flow field, as this shock strength increases, CPanel becomes less accurate and is predicting the shock as weaker than it actually is. An underpredicted shock strength indicates an overpredicted velocity over the bottom surface, and thus underpredicted bottom surface pressures. This yields a decreased pressure differential between the top and bottom surfaces which is captured by the underpredicted C_L results in Figure 5.22. The drag coefficient is not as affected by this shock strength underprediction since the geometry is relatively slender, so the contribution of this error in the direction of the drag force is small.

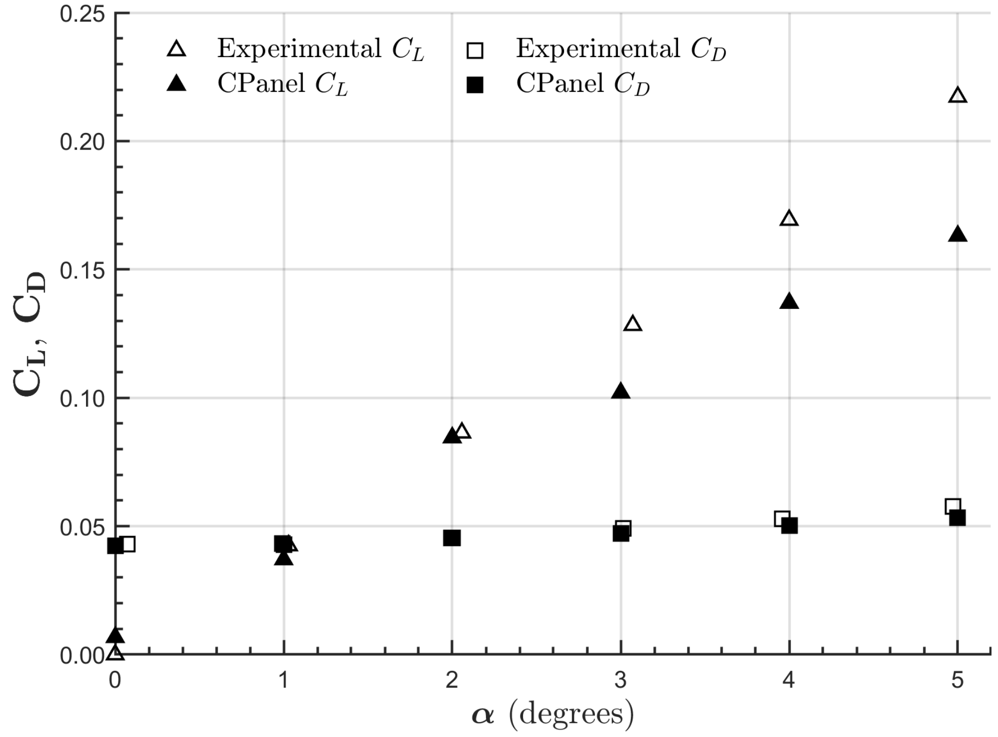


Figure 5.22: Delta wing 3 at Mach 1.62, C_L and C_D vs α

For the delta wing test cases presented here, CPanel gave accurate results so long as the regions across which shocks and expansions emanated were finely discretized; the mesh could remain relatively course away from these regions for the reasons discussed in Section 5.2. As mentioned in reference to the C_p distribution shown in Figure 5.17 and the plot of Figure 5.18, error due to the use of unstructured meshes was observed. Thus if an overly fine unstructured mesh is used, there will likely be higher error in the downstream-most regions of the geometry relative to a coarser mesh, assuming both meshes properly represent the geometry. This error may be of concern if high resolution and high accuracy pressure data, for example, is of interest; however, it has been shown that this error is generally low and does not take away from the utility that is most often needed of a panel code.

Chapter 6

CONCLUSION

6.1 Summary

A higher-order method was successfully implemented in CPanel for use in modeling supersonic flows. Application of the higher-order method to subsonic flows was also implemented. The doublet strength continuity enforcement scheme developed with this work has been shown to function properly and consistently with CPanel's existing code architecture designed for solving unstructured meshes. A strong base has been laid to allow for the future enhancement of CPanel's supersonic modeling capabilities, and of CPanel as a whole.

Results from CPanel were compared with those from other codes and theoretical solutions, showing strong correlation with both. Limits to solution accuracy were found in relation to higher Mach numbers or thicker geometries, as observed with other codes and as expected of linearized solution methods. As of this work, certain types of geometries can not be modeled accurately; however, partial solutions to these geometries were still shown to be accurate with small code modifications.

6.2 Future Work

CPanel is designed to be a continually developing tool, acting as a platform for students to perform research and to contribute to the field of computational fluid dynamics. With this in mind, as well as the unaddressed issues with the supersonic scheme discussed in this paper, the list below was put together as possible modifications and enhancements to be made to CPanel. The list is ordered in what was deemed highest priority to lowest priority in regard to the supersonic modeling abilities of CPanel, and each item will be briefly discussed.

- Fix issue of shared nodes at subsonic leading edges
- Fix issue of shared nodes at trailing edges
- Implement the ability to model wakes with the new higher-order methods
  - Incorporate vortex particle wake modeling
- Address spurious vortex formation at wing-body intersections
- Perform more rigorous testing with more complex geometries. A method of characteristics code can be used to validate
  - Detailed assessment of solution accuracy and stability as a function of paneling quality
- Include the ability to use superinclined panels to model inlet and exhaust flows
- Add off-body calculations for use in streamline tracing
- Implement wave drag calculations
- Allow use of velocity boundary conditions and compare solution accuracy with mass flux boundary conditions
- Extend supersonic solution for use at high subsonic Mach numbers, compare with simply using Prandtl-Glauert transformation on the current subsonic solution processes
- Implement transonic modeling to work in conjunction with the present subsonic and supersonic modeling schemes

- Implement a linear source/quadratic doublet solution
- Include the ability to use integral boundary layer methods to model viscous effects

In order for CPanel to be able to model any geometry, the issues discussed in Appendix C need to be addressed, which manifests itself in creating disparate upper and lower nodes on leading and trailing edges. The issue of shared nodes at these edges is also the primary source of the problem of spurious vortices forming at wing-body intersections. Seperate nodes are needed for the wing and body at the intersection, and extra boundary conditions applied, similar to the panel network scheme of PANAIR [1]. The first four items of this list are all related to the issue of shared nodes.

Once the above items have been completed, more testing can be performed to better understand CPanel's sensitivity to paneling, and to specifically find the issue in modeling flat surfaces that are parallel to the freestream flow. Adding superinclined panels, off-body calculations, wave drag calculations, and velocity boundary conditions are all non-essential additions, but would broaden the range of geometries and flows the CPanel user could model, as well as give more insightful results. Using the base architecture for higher-order methods that was implemented, the full Prandtl-Glauert subsonic solution could be implemented, to allow for accurate modeling at high subsonic Mach numbers. If this is done, CPanel could be further extended to modeling transonic flows by implementing a transonic modeling method to work in conjunction with CPanel's existing methods [25]. A linear source/quadratic doublet scheme could be implemented, and accuracy compared with the present method. Lastly, an integral boundary layer method could be included in CPanel.

BIBLIOGRAPHY

[1] Larry L. Erickson. Panel Methods – An Introduction. Technical Report NASA TR 2995, National Aeronautics and Space Administration, 1990.

[2] F. Edward Ehlers, Michael A. Epton, Forrester T. Johnson, Alfred E. Magnus, and Paul E. Rubbert. A Higher Order Panel Method for Linearized Supersonic Flow. Contractor Report NASA CR 3062, National Aeronautics and Space Administration, 1979.

[3] Ralph L. Carmichael and Frank A. Woodward. An Integrated Approach to the Analysis and Design of Wings and Wing-Body Combinations in Supersonic Flow. Technical Note NASA TN D-3685, National Aeronautics and Space Administration, 1966.

[4] Frank A. Woodward. Analysis and Design of Wing-Body Configurations at Subsonic and Supersonic Speeds. *Journal of Aircraft*, 5(6):528–534, 1968.

[5] Ralph L. Carmichael and Larry L. Erickson. PANAIR – A Higher Order Panel Method for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations. In *AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto, California*, June 1981.

[6] L. Fornasier. HISSS – A Higher-Order Subsonic/Supersonic Singularity Method for Calculating Linearized Potential Flow. In *AIAA 17th Fluid, Plasma Dynamics, and Lasers Conference, Snowmass, Colorado*, June 1984.

[7] Yuichi Maruyama, Sadao Akishita, and Akihito Nakamura. Numerical Simulations of Supersonic Flows about Arbitrary Configurations Using a New

Panel Method Program. In *Astrodynamics Conference, Williamsburg, Virginia*, 1986.

[8] Luis R. Miranda, Robert D. Elliot, and William M. Baker. A Generalized Vortex Lattice Method for Subsonic and Supersonic Flow Applications. Contractor Report NASA CR 2865, National Aeronautics and Space Administration, 1977.

[9] Brandon L. Litherland. Using VSPAERO. `http://openvsp.org/wiki/doku.php?id=vspaerotutorial`, July 2015. [cited February 2, 2019].

[10] Christopher R. Satterwhite. Development of CPanel, an Unstructured Panel Code, Using a Modified TLS Velocity Formulation. Master's thesis, California Polytechnic State University, San Luis Obispo, 2015.

[11] Connor Sousa. Unsteady Panel Code Utilizing a Vortex Particle Wake. Master's thesis, California Polytechnic State University, San Luis Obispo, 2016.

[12] Joseph Katz and Allen Plotkin. *Low-Speed Aerodynamics*, volume 13. Cambridge University Press, 2001.

[13] Russel M. Cummings, William H. Mason, Scott A. Morton, and David R. McDaniel. *Applied Computational Aerodynamics*. Cambridge University Press, 2015.

[14] J. Hadamard. *Lectures on Cauchy's Problem in Linear Partial Differential Equations*. Yale University Press, 1928.

[15] Luigi Morino, Lee-Tzong Chen, and Emil O. Sucio. Steady and Oscillatory Subsonic and Supersonic Aerodynamics around Complex Configurations. *AIAA Journal*, 13(3):368–374, 1975.

[16] Rob A. McDonald and Alejandro Ramos. Constrained Hermite Interpolation for Mesh-Free Derivitive Estimation Near and on Boundaries. *AIAA Journal*, 49(10), 2011.

[17] Forrester T. Johnson and Paul E. Rubbert. Advanced Panel-Type Influence Coefficient Methods Applied to Subsonic Flows. In *AIAA 13th Aerospace Sciences Meeting, Pasadena, California*, January 1975.

[18] Michael A. Epton and Alfred E. Magnus. PANAIR–A Computer Program for Predicting Subsonic or Supersonic Potential Flows About Arbitrary Configurations Using a Higher Order Panel Method. Contractor Report NASA CR 3251, National Aeronautics and Space Administration, 1990.

[19] Forrester T. Johnson. A General Panel Method for the Analysis and Design of Arbitrary Configurations in Incompressible Flows. Contractor Report NASA CR 3079, National Aeronautics and Space Administration, 1980.

[20] John L. Hess and A.M.O Smith. Calculation of Non-Lifting Potential Flow about Arbitrary Three-Dimensional Bodies. Technical report, Douglas Aircraft Division, 1962.

[21] Brian Maskew. Program VSAERO Theory Document. Contractor Report NASA CR 4023, National Aeronautics and Space Administration, 1987.

[22] Per-Olof Persson and Gilbert Strang. A Simple Mesh Generator in MATLAB. *SIAM Review*, 46(2):329–345, June 2004.

[23] Jack Moran, Edward N. Tinoco, and Forrester T. Johnson. User's manual: Subsonic/Supersonic Advanced Panel Pilot Code. Contractor Report NASA CR 152047, National Aeronautics and Space Administration, 1978.

[24] Eugene S. Love. Investigations at Supersonic Speeds of 22 Triangular Wings Representing Two Airfoil Sections for each of 11 Apex Angles. Research Memorandum NACA RM L9D07, National Advisory Committee for Aeronautics, 1949.

[25] Gino Moretti and Gary Bleich. Three-Dimensional Flow Around Blunt Bodies. In *AIAA 5th Aerospace Sciences Meeting, New York, New York*, January 1967.

[26] David D. Marshall and Eric A. Mehiel. Introduction of Software Development Practices into Aerospace Engineering Curriculum. In *46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada*, January 2008.

APPENDICES

Appendix A

INFLUENCE COEFFICIENTS

The influence coefficient formulations used for this thesis were primarily adopted from other published works. Nearly all of these works involved the implementation of a linearly varying source distribution and a quadratically varying doublet distribution, while CPanel's higher-order methods use a constant source distribution and a linearly varying doublet distribution. The following sections show how one arrives at the influence coefficient formulations implemented in CPanel from those presented by other codes.

A.1 Subsonic Implementation

Since constant strength sources were used with the higher-order subsonic implementation, no changes were needed in the source influence coefficient calculations, and the original CPanel scheme is used. A new scheme was however needed for the doublet influence coefficient calculations, which was adopted from Johnson [19].

The influence coefficient expression for a linearly varying doublet can be found two different ways from the expressions given by Johnson. The first is to start from the influence coefficient solution for a linearly varying source. This begins with the integral given in Equation 4.20, which is rewritten below.

$$\phi_S(\mathbf{P}) = -\int_S \sigma \left(\frac{1}{4\pi|\mathbf{r}_{ij}|} \right) dS \tag{A.1}$$

Differentiating and solving this equation gives the expression for source-induced perturbation velocity as

$$\mathbf{v}_S(\mathbf{P}) = \sigma(x,y)\mathbf{J}_0 + \sigma_x(x,y)\mathbf{J}_1 + \sigma_y(x,y)\mathbf{J}_2 \tag{A.2}$$

where

$$\begin{aligned} \sigma(x,y) &= \sigma_0 + \sigma_x x + \sigma_y y \\ \sigma_x(x,y) &= \sigma_\xi \\ \sigma_y(x,y) &= \sigma_\eta \end{aligned} \tag{A.3}$$

Note that this is the same form as Equations 4.2-4.3. In Equation A.2, each **J** represents different solutions to a single fundamental integral. This fundamental integral will be discussed in greater detail shortly. Each of the three fundamental integral solutions can be written in vector form as

$$\mathbf{J} = [J_x, J_y, J_z] \tag{A.4}$$

Katz and Plotkin [12] show that the expression for the doublet-induced potential is the same as that for the source-induced velocity in the z-direction, which is a function of the J_z fundamental integral solutions. The source-induced velocity in the z-direction is

$$\mathbf{v}_{S,z} = \sigma(x,y)J_{0,z} + \sigma_x(x,y)J_{1,z} + \sigma_y(x,y)J_{2,z} \tag{A.5}$$

thus the influence coefficient for a linearly varying doublet is

$$\phi_D = \mu(x,y)J_{0,z} + \mu_x(x,y)J_{1,z} + \mu_y(x,y)J_{2,z} \tag{A.6}$$

The fundamental J_z expression, using the nomenclature of Johnson, is written as

$$J_z(M,N) = -\frac{1}{4\pi}\Big[-zH(M,N,3) + \textit{higher order terms}\Big] \tag{A.7}$$

$$H(M,N,K) = \int_S \frac{(\xi - x)^{M-1}(\eta - y)^{N-1}}{\left[\sqrt{(\xi - x)^2 + (\eta - y)^2 + z^2}\right]^K} dS \tag{A.8}$$

In Equation A.7, the higher order terms are for curved paneling, and so are dropped for CPanel's implementation. The J terms of Equation A.6 can be rewritten in the form of A.7 as

$$\begin{aligned} J_{0,z} &= J_z(1,1) \\ J_{1,z} &= J_z(2,1) \\ J_{2,z} &= J_z(1,2) \end{aligned} \tag{A.9}$$

The J_x and J_y fundamental integral solutions used just for computing source-induced velocity are similar to Equation A.7, except they are functions of $H(M+1,N,3)$ and $H(M,N+1,3)$, respectively, instead of $H(M,N,3)$.

The second means of arriving at the influence coefficient expression for a linearly varying doublet is to simply start with the influence coefficient expression for a quadratically varying doublet which is

$$\phi_D = \mu(x,y)I_0 + \mu_x(x,y)I_1 + \mu_y(x,y)I_2 + \textit{higher order terms} \tag{A.10}$$

where

$$I(M,N) = \frac{1}{4\pi}\left[zH(M,N,3) + \textit{higher order terms}\right] \tag{A.11}$$

$$\begin{aligned} I_0 &= I(1,1) \\ I_1 &= I(2,1) \\ I_2 &= I(1,2) \end{aligned} \tag{A.12}$$

and drop the higher order terms, which are for the quadratically varying components of the doublet expression in A.10 and curved panels in A.11.

A.2 Supersonic Implementation

The influence coefficients used for the supersonic implementation were adopted from Ehlers et. al. [2]. Section 4.5 gave the constant source influence coefficient expression and linear doublet influence coefficient expression as, respectively

$$\phi_S = \frac{\sigma}{2\pi}\big(x_m w_0 - z Q_I\big) \tag{A.13}$$

$$\phi_D = -\frac{1}{2\pi}\big(\mu Q_I - \mu_x z w_0 - \mu_y z w_0/m\big) \tag{A.14}$$

which were derived from the original expressions of Ehlers et. al. by dropping higher order terms. Equations A.13 and A.14 give the influence coefficient for the edge of a panel on a control point, so they must be computed for each edge of a given panel and summed to find the influence coefficients for a given panel-control point pair. The fundamental integrals in these equations are w_0 and Q_I, for which there are different solutions depending on the inclination of the given edge: subsonic, supersonic or sonic. Their solutions also depend on which parts of the edge are inside the DOD of the control point.

A.2.1 Subsonic Edges

For a subsonic edge,

$$w_0 = \frac{m}{2\sqrt{1-m^2}}\ln\left[\frac{-\hat{y}_m + R\sqrt{1-m^2}}{-\hat{y}_m - R\sqrt{1-m^2}}\right]\Bigg|_1^2 \tag{A.15}$$

$$Q_I = \text{sign}(z)\tan^{-1}\left[\frac{\hat{x}_m R}{-|z|\hat{y}_m}\right]\Bigg|_1^2 \tag{A.16}$$

where 1 and 2 indicate the two end points of the edge being evaluated. In these equations and all following, m is edge slope. Taking the natural log and inverse tangent as shown above is quite computationally expensive, considering they must be taken for every edge endpoint for every panel-control point pair in the geometry. Some computational efficiency can be gained by combining the log and inverse tangent terms for each expression, giving

$$w_0 = \frac{m}{\sqrt{1-m^2}}\ln\left[\frac{-\hat{y}_{m,2} + R_2\sqrt{1-m^2}}{-\hat{y}_{m,1} - R_1\sqrt{1-m^2}}\right] \tag{A.17}$$

$$Q_I = \tan^{-1}\left[\frac{z\hat{x}_m\big[(-\hat{y}_{m,1})R_2 - (-\hat{y}_{m,2})R_1\big]}{z^2(-\hat{y}_{m,1})(-\hat{y}_{m,2}) + \hat{x}_m^2 R_1 R_2}\right] \tag{A.18}$$

Equations A.17 and A.18 are the most efficient means of computing the fundamental integrals for an edge that is completely inside the DOD. When an edge is intersected by the DOD and one of the endpoints is outside the DOD, the endpoint's corresponding R term will have an imaginary part and its real part will be zero. Only the real part of R is taken which is zero, so the fundamental integral solutions can be simplified even further. For an edge intersection, the fundamental integrals are computed as

$$w_0 = w_{0,2} - w_{0,1} \tag{A.19}$$

$$Q_I = Q_{I,2} - Q_{I,1} \tag{A.20}$$

where the terms corresponding to the endpoint outside the DOD are zero, and the other terms are computed from Equations A.15 and A.16.

For panel edges that are parallel to the panel local coordinate system x-direction, $m = 0$ so the expression

$$\hat{w}_0 = w_0/m \tag{A.21}$$

is instead used. This simplifies Equation A.15 to

$$w_0 = \frac{1}{2}\ln\left[\frac{-\hat{y}_m + R}{-\hat{y}_m - R}\right]\Bigg|_1^2 \tag{A.22}$$

A similar simplification can be made to Equation A.17. No special form of Q_I is needed when $m = 0$. Note that subsonic edges are always entirely inside the downstream Mach cone emanating from the leading endpoint of the edge, so there is never a Mach wedge for subsonic edges.

A.2.2 Supersonic Edges

For a supersonic edge,

$$w_0 = \frac{1}{\sqrt{1-\lambda^2}} \tan^{-1}\left[\frac{y_m}{R\sqrt{1-\lambda^2}}\right]\Bigg|_1^2 \tag{A.23}$$

$$Q_I = \tan^{-1}\left[\frac{z y_m}{x_m R}\right]\Bigg|_1^2 \tag{A.24}$$

where $\lambda = 1/m$, which is used instead of m to keep singularities from occurring in Equation A.23 when the edge is parallel to the panel local y-direction where $m = \infty$. Just like with the subsonic edges, terms can be combined to give more computationally efficient expressions.

$$w_0 = \frac{1}{\sqrt{1-\lambda^2}} \tan^{-1}\left[\frac{\sqrt{1-\lambda^2}\left[(-y_{m,1})R_2 - (-y_{m,2})R_1\right]}{(-y_{m,1})(-y_{m,2}) + (1-\lambda^2)R_1 R_2}\right] \tag{A.25}$$

$$Q_I = \tan^{-1}\left[\frac{z x_m\left[(-y_{m,1})R_2 - (-y_{m,2})R_1\right]}{x_m^2 R_1 R_2 + z^2(-y_{m,1})(-y_{m,2})}\right] \tag{A.26}$$

These equations can be further simplified when the edge is intersected by the DOD as with subsonic edges, though with a supersonic edge, it can be intersected by the DOD without either endpoint being inside the DOD. This means the control point is inside the Mach wedge of the edge, in which case these equations can be simplified even further. First addressing the case of an edge-DOD intersection with one endpoint inside the DOD, Equations A.19 and A.20 are used to compute the fundamental integrals where the terms for the endpoint outside the DOD simplify to

$$w_0 = \text{sign}(y_m)\frac{\pi}{2\sqrt{1-\lambda^2}} \tag{A.27}$$

$$Q_I = \text{sign}(z y_m)\frac{\pi}{2} \tag{A.28}$$

These same expressions are used for the Mach wedge calculations, except they are used for both endpoints.

A.2.3 Sonic Edges

When an edge is inclined at the Mach angle, or $|m| = 1$, it is a sonic edge, and a singularity occurs in any of the above expressions for $w0$. So it is expanded by powers of $\sqrt{1-\lambda^2}$ giving,

$$w_0 = z_r \left[1 - \frac{(1-\lambda^2) z_r^2}{3} + \frac{(1-\lambda^2)^2 z_r^4}{5} - \frac{(1-\lambda^2)^3 z_r^6}{7} + \cdots \right] \tag{A.29}$$

where

$$z_r = \frac{y_{m,1} R_2 - y_{m,2} R_1}{y_{m,1} y_{m,2} + (1-\lambda^2) R_1 R_2} \tag{A.30}$$

Either the subsonic or supersonic edge expression for Q_I can be used since $|m| = 1$ makes them identical, and no singularities are introduced.

Appendix B

UNIT TESTING STATE DESCRIPTIONS

The tables on the following pages give illustrations and descriptions of the states used with the supersonic unit testing results in Chapter 5. Note that the images shown for the geometries of each state are just snapshots, and are simply representative of each respective state. As shown in the results, a certain state can exist for various ranges of control point locations with respect to a panel.

Note that the possible state of two edge intersections and no vertices inside the DOD is not included in this unit testing study. This state behaves similarly to State 4 of Table B.1, so it was not included to improve the clarity of the results.

Table B.1: State descriptions for control point variation in the x-direction

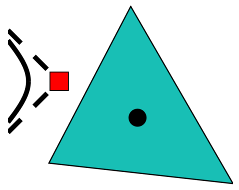
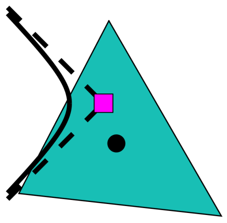
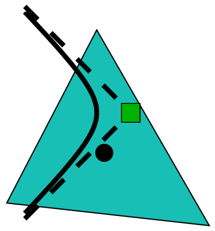
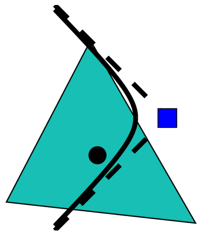
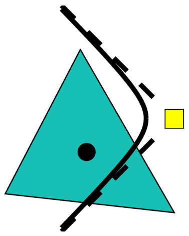
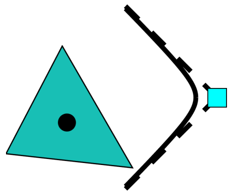
State	Geometry	Description
1		Panel completely outside DOD
2		One edge intersection, no vertices inside DOD
3		Two edge intersections, one vertex inside DOD
4		Three edge intersections, one vertex inside DOD
5		One edge completely inside DOD
6		Panel completely inside DOD

Table B.2: State descriptions for control point variation in the y-direction

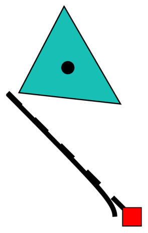
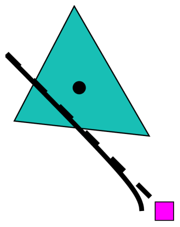
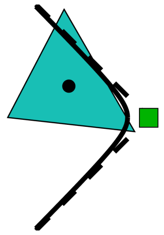
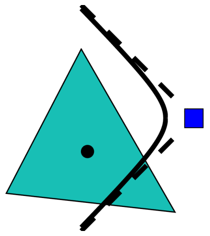
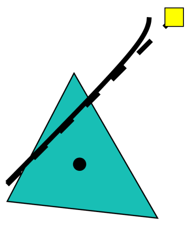
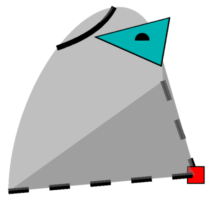
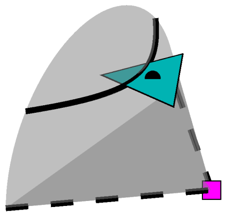
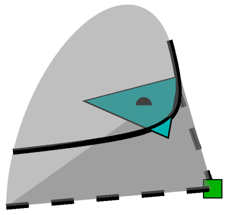
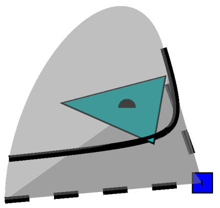
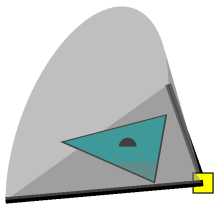
State	Geometry	Description
1		Panel completely outside DOD
2		Two edge intersections, one vertex inside DOD
3		Three edge intersections, one vertex inside DOD
4		One edge completely inside DOD
5		Two edge intersections, one vertex inside DOD

Table B.3: State descriptions for control point variation in the z-direction

State	Geometry	Description
1		Panel completely outside DOD
2		Two edge intersections, one vertex inside DOD
3		Three edge intersections, one vertex inside DOD
4		One edge completely inside DOD
5		Panel completely inside DOD

Appendix C

OUTSTANDING ISSUES

A single fundamental aspect of how CPanel processes input geometries is the source of the present inability to model geometries in supersonic flow with any one of the following characteristics:

- Sharp subsonic leading edges
- Sharp trailing edges
- Wing-body intersections
- Wake panels

CPanel uses the .tri file format for mesh files, the format for which is shown below:

```
nVerts nTris
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
```

When meshes are generated using this format, the triangular panels throughout the geometry are all connected via shared vertices, i.e. nodes. This means that with sharp leading or trailing edges, a single node on one of these edges will belong to both upper and lower panels that are connected to the edge. Figure 4.2 depicts this scenario, which is again shown below. The same situation occurs at wing-body intersections–a single node will belong to both body and wing panels that are at the intersection.

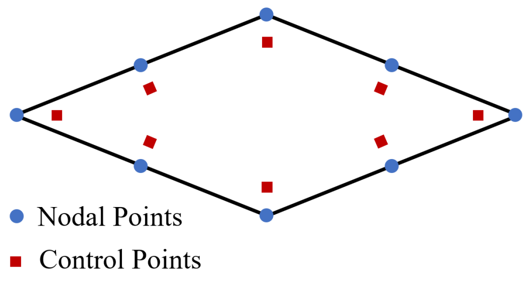
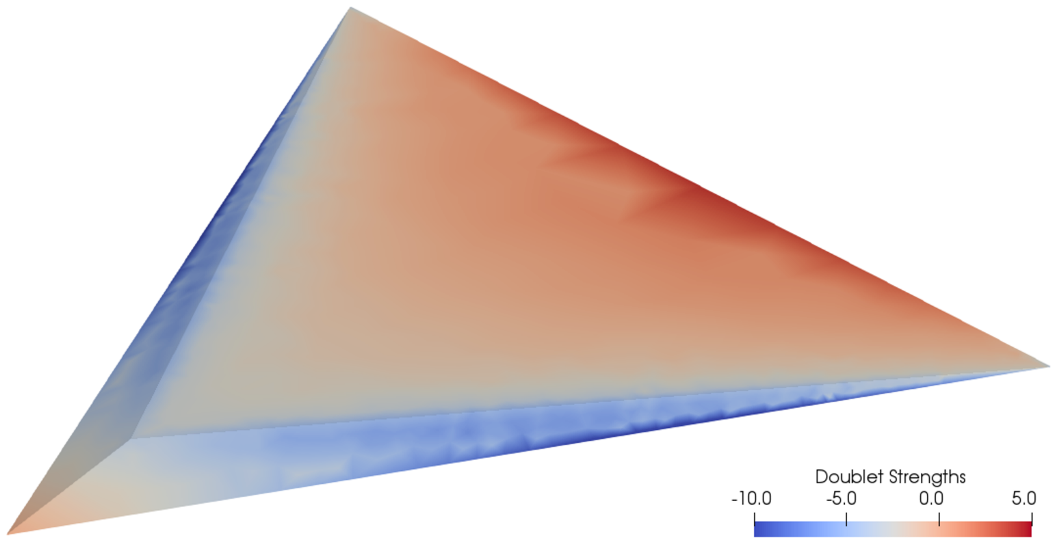


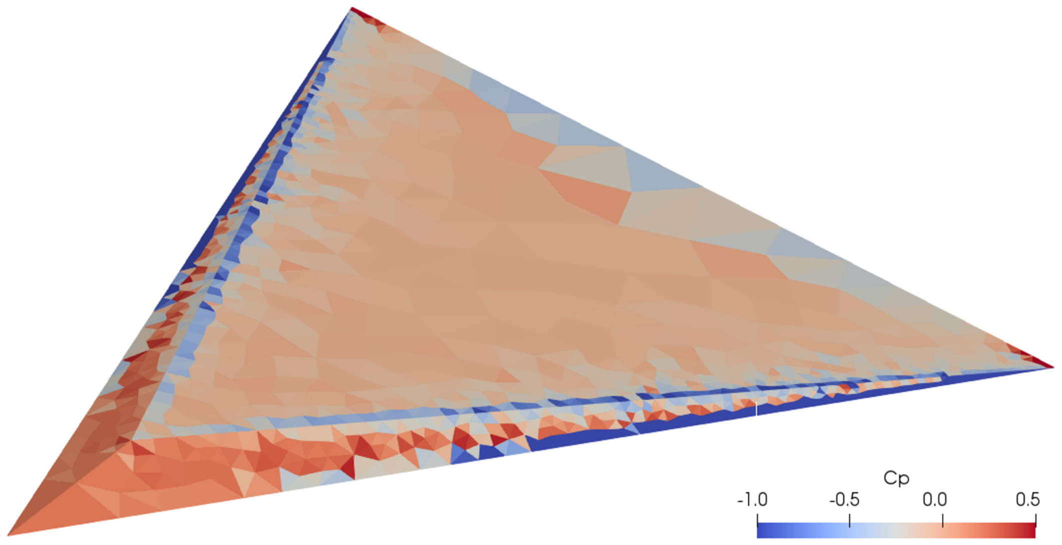
Figure C.1: Present nodal and control point definition

In these scenarios, there needs to be a doublet strength discontinuity. This means there must be a doublet strength at the panels' shared point that is associated with the panel on one side of the intersection, and a different doublet strength associated with the panel on the other side of the intersection. Since only one node is defined at these intersections in CPanel currently, and doublet strength is defined and solved at these nodes, only one doublet strength can be associated with the panels on either side of the intersection. So doublet strength is forced to be continuous, where it should be discontinuous. This is the source of the problem which prevents the accurate modeling of the geometries with the characteristics listed above. The ability to model wakes is inhibited by this issue because there also needs to be upper and lower wake panels, where each set is associated with the upper and lower sets of nodes on the trailing edge.

An example of this issue's effect on the leading and trailing edges is shown in Figure C.2. This wing has the same airfoil as Wing 3 in Figure 5.16(c), but with more sweep such that the leading edge is subsonic. It is clear in the doublet distribution of



(a) *Doublet strength distribution*

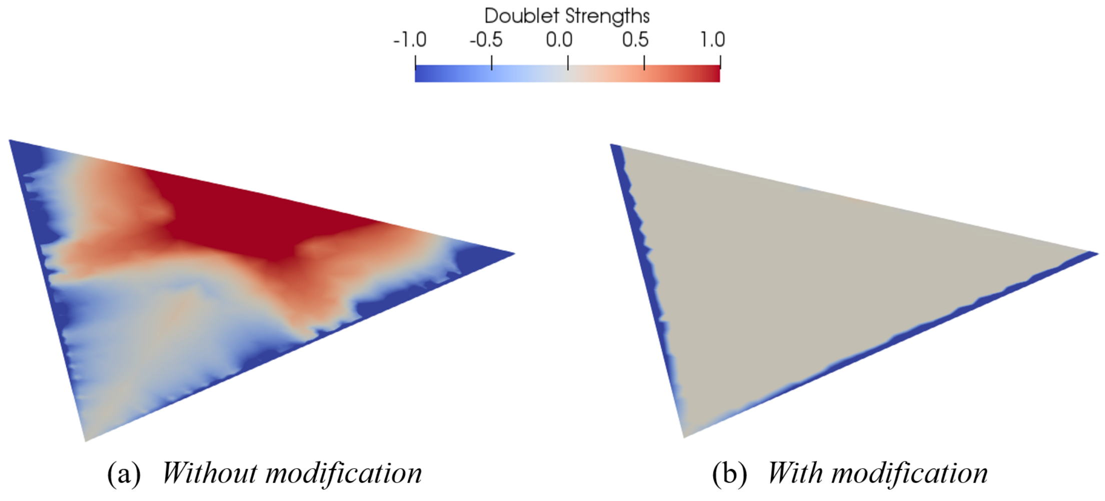


(b) *Pressure distribution*

Figure C.2: Example of leading and trailing edge errors

Figure C.2(a) and in the pressure distribution of Figure C.2(b) that there is error on the leading and trailing edges. Notice in Figure C.2(b) that only the panels that have nodes which are part of the respective edges are clearly in error. Though because there are errors at the leading edge, there will be error in all panels downstream of the leading edge also. This is not the case for the trailing edge since it is downstream of all other panels.

Despite these issues, accurate results for the top surfaces of Delta wings 1 and 2, and for the entirety of Delta wing 3, were given by CPanel. This was accomplished via small code modifications for each test case. Wing 1 of Figure 5.16(a) has a subsonic leading edge; it also has a flat bottom surface which is parallel to the freestream direction, so the doublet strength across the entirety of the bottom surface should be zero. To acquire the correct solution for the top surface of this wing, the top and bottom surfaces are essentially decoupled by forcing the doublet strengths on the bottom surface panels that are not on leading or trailing edges to zero. The bottom surface is forced to the correct solution thus allowing the top surface to be naturally solved and yield accurate results. Figure C.3 shows the doublet strength distribution of the bottom surface with and without this modification. Since there is only one set



(a) *Without modification* (b) *With modification*

Figure C.3: Wing 1 bottom surface solution with and without modification

of nodes on the leading edge, ParaView interpolates the doublet strengths there onto the leading lower surface panels as seen in Figure C.3(b), however this does not affect the solution of the top surface.

Delta wing 2 in Figure 5.16(b) has a supersonic leading edge, so doublet strength can actually be continuous here since it is zero at supersonic leading edges. There were however other issues in modeling this wing, which were traced to what seems to be CPanel's inability to model a flat surface which is parallel to the freestream direction in supersonic flow. The source of this issue was not exactly determined though two likely culprits were identified.

The first is that the bottom surface of the geometry is not actually being modeled as perfectly flat, thus introducing error into the solution. A modification in the code was made to force the z-coordinates of the bottom surface panels to zero which did show a slight improvement in the results, but the solution was still substantially in error. The second and most likely culprit is that the errors are simply due to the unavoidable errors introduced with unstructured paneling. Since this geometry is so thin and the scale at which the pressure coefficient is being measured is so small, even a small error in the forward region of the wing could cause the entire solution to be in error. Once the leading edge error has been resolved, Wing 1 can be tested again to see if the same errors occur as are presently occurring with Wing 2. An example of this error is shown in Figure C.4. To acquire the correct solution for the top surface of Wing 2, the same scheme of forcing the doublet strengths to zero on the bottom surface was used.

Delta wing 3 in Figure 5.16(c) has a supersonic leading edge, so no modification was needed in that regard. A small temporary post-processing step was added however to fix the incorrect trailing edge panels such as those shown in Figure C.2. All that was done was when computing pressure coefficients for each panel, in computing

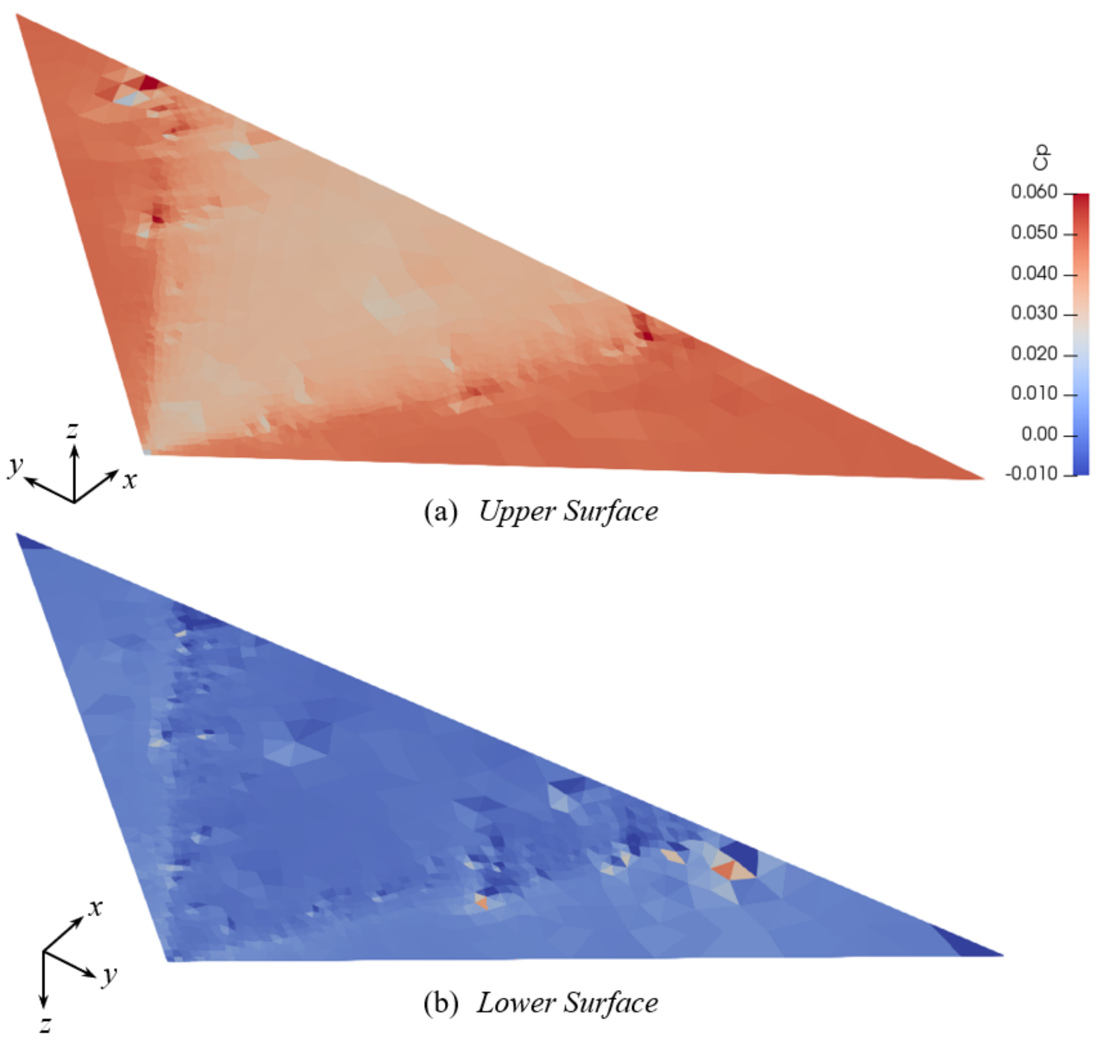


Figure C.4: Example of Wing 2 errors

this for trailing edge panels, the pressure found at the panels just upstream of the trailing edge was given to the trailing edge panels, which is shown to be an appropriate approximation. Using this method gave the results presented in Figures 5.21 and 5.22.

The permanent solution to these various issues could be approached different ways. One is to modify the methods of mesh generation to recognize sharp edges and wing-body intersections, and for them to then create disparate nodes associated with the panels on either side of the edges or intersections. This kind of approach is used by PANAIR [18] and similar codes; PANAIR requires the user to define paneling networks on the geometries for this purpose. Such an approach is not favorable for CPanel since one of its greatest strengths is its ease of use, specifically in regard to users being able to quickly generate unstructured meshes. Another approach is to apply line vortices at sharp edges and intersections, though this would not work with trailing edges that shed wakes.

The approach that is proposed as the best solution is to implement algorithms which identify leading edges, trailing edges, and wing-body intersections in CPanel. These identifications should be made purely from processing the .tri data, and before CPanel constructs and stores its own version of the geometry data for use throughout the program. This allows the raw .tri data to be read, new nodes created and triangles modified to include the new nodes, and an updated .tri file to be created with this new data. Then the same geometry processing schemes already used in CPanel can again be used here. There will likely need to be a user input controlling what angle between two panels is defined as a sharp edge. Lastly in regard to this proposed solution, for nodal points that exist at the same location but are different (as in one belongs to an upper surface and the other belongs to a lower surface), the control points cannot exist at the same location, so they will have to be moved away from each other, such as is depicted in Figure C.5.

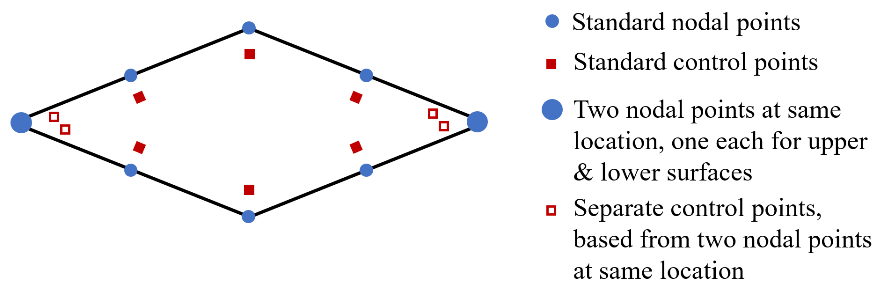

Standard nodal points
Standard control points
Two nodal points at same location, one each for upper & lower surfaces
Separate control points, based from two nodal points at same location

Figure C.5: Example of proposed new nodal and control point definition

Appendix D

DEVELOPMENT METHODOLOGY

Implementing an entirely new solution scheme within CPanel's existing code architecture brought with it the risk of introducing bugs to the previously implemented schemes. There was also the risk that parts of the original code base would introduce bugs to the newly implemented schemes, even when such bugs did not occur for the existing schemes. To mitigate these risks, modern software development practices were exercised throughout the work of this thesis, like was done with CPanel's original implementation [10].

A version control system was used to both preserve previous versions of CPanel, and to catalog the progress of the present work. The process of functional decomposition was utilized, specifically in the early development stages. This process of reducing complex tasks to smaller less complex tasks [26] proved instrumental in identifying and isolating algorithms unique to the other CPanel solution schemes, as well is in identifying which parts of these schemes could be used with the present implementation. The functional flow diagram in Figure 4.1 demonstrates a high level functional decomposition of CPanel. More specific processes, such as the algorithms shown in Chapter 4, were decomposed similarly to facilitate their development and implementation.

As discussed in Chapter 5, unit level testing was utilized extensively for this work. For both the subsonic and supersonic implementations, some of the more complex tasks, such as the influence coefficient calculations, were broken down and developed independently in MATLAB where they were tested prior to their implementation in CPanel. Specifically for the supersonic scheme, processes like the domain of dependence check and the coordinate transformation were first developed and tested in

MATLAB independently and together. Though it may seem repetitive and inefficient to write and test these schemes on two different platforms, this process proved critical to the success of this thesis. It enabled steady and incremental implementations to be performed, and allowed direct comparisons to be made between the MATLAB and CPanel software units, accelerating what easily could have been a tedious and painstaking validation process.

One of the many examples in which this development methodology facilitated the tracking down of a bug is when CPanel's supersonic solutions were in error due to CPanel's default direction of integration around a panel, which was the opposite of that needed for the supersonic computations. This is a case where a part of CPanel that is used for all solution schemes didn't introduce a bug in CPanel v1.0 and v2.0, but it did in the supersonic solution scheme. Partially automated testing was used to track down bugs such as these, where a case is run in CPanel which would then output sets of panel-control point pairs that could be run in MATLAB, and results compared. This process substantially accelerated debugging efforts such as this one, and was used during all stages of code development.

Appendix E

INPUT FILE FORMAT

In regard to the CPanel input file entries and options, they are nearly unchanged from the format given in the original CPanel documentation [10]. The only change is the addition of a solver option added to control whether the user would like to use the higher-order method for subsonic flow. An example input file is shown below with this additional solver option. As before, the variable names and spacing must be exactly as shown below, and comments can be added using a %.

There are now however additional restrictions that must be adhered to in running supersonic models. The control for modeling supersonic flow is simply via the Mach number input. A warning will be given if the input Mach number is between 0.6 and 1.3 since these approach the transonic regime which can not be presently modeled in CPanel. If the 'Subsonic_Higher_Order_Method' option is turned on, and the input Mach number is supersonic, the user will be warned that the subsonic option will be ignored and asked to continue in modeling supersonic flow, or exit. The user will also be warned if superinclined panels are found since the existence of such panel will cause errors in the solution if they are not downstream of all other panels in the geometry. Since wakes cannot be modeled as of this implementation, the user will also be warned if wake panels are included in the geometry and that they will be ignored.

Lastly, cases can not be used in supersonic modeling. In subsonic modeling, the influence coefficient matrices are constructed using purely geometric data, allowing the user to model different velocities, angles of attack, etc. in one run. However in modeling supersonic flow, the influence coefficient matrices are dependant on these quantities, and so the matrices change if the inputs change. If the user inputs multiple cases with a supersonic Mach number, they will be warned that only the first entry of each input will be used, and the remaining cases will be ignored.

```
%% CPanel Input File %%

% Reference Geometry (ft) %
GeomFile = diamondWing.tri
S_ref = 16.0
b_ref = 8.0
c_ref = 2.0
X_cg = 0.0
Y_cg = 0.0
Z_cg = 0.0

% Cases %
Velocity (ft/s)
1
1600.0
Angle_of_Attack (degrees)
1
10.0
Angle_of_Sideslip (degrees)
1
0.0
Mach_Number
1
1.5

% Solver Options (0 = OFF, 1 = ON) %
Subsonic_Higher_Order_Method
0
Surface_Streamlines
1
Stability_Derivatives
0
Write_Influence_Coefficients
0
```