An Arbitrarily High-Order, Unstructured, Free-Wake Panel Solver

by

John Pease Moore IV

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author .....
Department of Aeronautics and Astronautics
August 22, 2013

Certified by .....
Jaime Peraire
H.N. Slater Professor and Department Head
Thesis Supervisor

Accepted by .....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

An Arbitrarily High-Order, Unstructured, Free-Wake Panel Solver

by

John Pease Moore IV

Submitted to the Department of Aeronautics and Astronautics
on August 22, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

A high-order panel code capable of solving the potential flow equation about arbitrary curved geometries is presented. A new method for integrating curved, high-order panels using adaptive Gaussian quadrature is detailed. Furthermore, automated wake handling is addressed and a method to robustly solve for the steady-state free-wake rollup is proposed. Finally, a Fast Multipole Method with a complexity that scales as $O(N)$ is also presented so that large problems can be handled using only a linear mesh. Results are presented to demonstrate high order accuracy and agreement with other inviscid solvers for a variety of test cases.

Thesis Supervisor: Jaime Peraire
Title: H.N. Slater Professor and Department Head

Acknowledgments

The author would like to thank the numerous individuals that have made this work possible and have been with me along this journey. First and foremost, I'd like to start by thanking my wife for always being there for me, no matter what. You are so fun to be around, and are always able to put a smile on my face.

I would also like to thank all the students in the ACDL for their support and company. I have learned a lot from you guys. I would like to thank David Moro for his numerous fruitful conversations about aerodynamics and life in general. Many thanks to Xevi Roca for always being there to answer all my coding inquiries (and, of course, teaching me to surf). I would also like to thank Hemant Chaurasia for always being there to provide advice and encouragement. I would like to thank Ferran Vidal for putting up with me in the office, and for always being there to talk about projects and have a good laugh.

Additionally, I would like to thank Professor Mark Drela for his numerous advice on boundary element methods and potential flow. It would have been impossible to get to this point without your expertise.

Last, but not least, I would like to thank my advisor Professor Jaime Peraire for his patience, expertise, and wisdom. Even though you are busy with being Department Head, you always make time for your students, and remind us that it is just as important to relax and have a good time as it is to do cool research. All your support and advice is greatly appreciated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Panel methods are currently capable of rapidly solving the potential flow equation on rather complex geometries using only a workstation. These methods, which originated in the early 1960's and are a special case of the Boundary Element Method or BEM, continue to be attractive since the governing equations only need to be solved at the boundary [13, 17, 15, 8]. This eliminates the need for a volume mesh, as is needed when using Finite Difference, Finite Volume, or Finite Element methods, and results in a system which is of comparatively lower dimension. However, there are still two primary open issues in aerodynamic panel methods that this work seeks to address.

The first is that panel methods result in a system of equations that is dense, and memory requirements and matrix assembly complexity grows as the square of the number of degrees of freedom. Additionally, the quantity of interest is usually pressure, which for panel methods is generally obtained by differentiating a piecewise-linear potential field, resulting in only a first-order convergence in pressure compared to the second-order convergence usually obtained with standard CFD solvers. For example, a well-resolved Euler mesh in three dimensions will typically consist of over 100,000 *surface* elements. If this same mesh and approximation order was used in a potential flow solver the required memory would be at least 40GB and the pressure would converge at a rate one order lower than the corresponding Euler solution.

The second open issue is treatment of the wake. Vorticity must be shed into the wake in order to produce circulation (and hence, lift) about a lifting body. In practice,

this is usually accomplished by shedding a planar wake extending to infinity from the trailing edge of all lifting surfaces. However, this approach is not physically accurate since the wake should be force-free and convect with the local velocity. Approaches to address these two obstacles are discussed below.

1.1 Operation Count and Memory Requirements

Two methods are proposed for dealing with the dense system of equations that results from the BEM discretization of the potential flow equation. The first is to use high-order curved elements to represent the geometry of interest [25, 26, 29, 6, 16], and the second is to implement the Fast Multipole Method (FMM) and therefore never store the system in memory [12]. These two methods can be used together if desired.

1.1.1 Mitigation Strategy 1: High-Order Curved Elements

Various methods have been proposed to integrate single and double-layer potentials over curved elements including polynomial fitting [26, 29], element subdivision [6], and integrand desingularization through Taylor series expansions [16]. Polynomial fitting approaches are expensive and have only been demonstrated up to second order for doublet distributions [26]. Element subdivision approaches have not been capable of accurately computing self-term influences [6]. The integrand desingularization approach relies on a B-Spline surface parametrization while also using adaptive quadrature for near-field influences [16]. The integral desingularization approach presented in [16] is difficult to implement and is not extendible to the more general case of NURBS. These high order curved element approaches are particularly attractive for the simulation of rigid geometries since the influence matrix can be computed once and stored in memory for the remainder of the computation.

1.1.2 Mitigation Strategy 2: The Fast Multipole Method

The FMM was developed by Greengard et al. [12] and reduced the computational complexity of performing matrix-vector products in particle simulations from $O(N^2)$ to $O(N)$, with a memory requirement that scales as $O(N \log N)$. In the early 1990's, the FMM was coupled with a Boundary Element Method in order to rapidly simulate circuits [18]. In the late 1990's, this same approach was applied to aerodynamic cases with a computational complexity of $O(N \log N)$ [2, 23, 3]. The FMM is attractive for aeroelastic problems where the geometry is deformable since computing a single matrix-vector product with FMM is significantly faster than assembling a new influence matrix at each iteration. The FMM is also useful for large meshes where the system of equations resulting from the BEM discretization is too large to be stored in memory.

1.2 Approaches to Automated Wake Treatment

One of the primary advantages of the BEM is that it has the potential to require minimal pre-processing by the user, in contrast to volumetric methods which may require an experienced user to generate an acceptable volume mesh. However, it is not clear how automatic wake handling for lifting bodies should be treated. There are four popular approaches: (1) fixed-wake, (2) free-wake with panels, (3) free-wake with particles, and (4) vorticity transport. The fixed-wake approach is the simplest and least expensive since it does not involve time marching, but also the least accurate since the fixed wake geometry is somewhat arbitrarily chosen by the user and will violate the force-free wake requirement. The second and third approaches are free-wake, meaning that the wake elements or particles travel with the local velocity, satisfying the force-free wake requirement. They are computationally expensive since the only way to robustly obtain the wake position is through time marching. The primary issue with using panels in the wake is that they are singular at their surface. This is problematic if the wake is self-intersecting or intersects the geometry, although an approach using discontinuous basis functions has been proposed to remedy this [27].

Alternatively, vorticity in the wake can be represented with de-singularized vortons at the cost of decreased accuracy. The last approach involves converting the shed vorticity sheet into volumetric vorticity and solving the vorticity transport equation using any of the standard volumetric PDE approaches [4]. However, this requires a well-resolved geometry-conforming volume grid and may be nearly as expensive and require the same amount of pre-processing as solving the full incompressible Euler equations.

1.3 Thesis Contributions

This thesis proposes three approaches to deal with to the aforementioned aerodynamic panel method obstacles:

1. A new algorithm for the integration of arbitrarily high-order single and double-layer potentials on curved elements using an adaptive quadrature scheme.

2. A new method for solving for the steady-state potential solution and wake roll-up about lifting bodies by casting the BEM and wake evolution equations into a fully coupled nonlinear system.

3. A Fast Multipole-accelerated BEM with a cost that scales as $O(N)$ compared to previous potential flow solvers using a similar technique which scaled as $O(N \log N)$ [3, 2] .

The remainder of this thesis is organized as follows. Chapter two will introduce the equations governing incompressible potential flow and the required boundary conditions. Chapter three will detail the discretization in both space and time of the potential flow equation using the BEM. Chapter four will then introduce a new method for integrating the Green's function for Laplace's equation over high-order curved elements. This will be followed by a description of the nonlinear steady-state free-wake solver. Results will then be presented to demonstrate high order convergence, steady-state wake treatment, and application of the solver to complex

geometries. Lastly, conclusions will be drawn and suggestions for further research will be proposed.

Chapter 2

Incompressible Potential Flow

This chapter details the equations governing incompressible potential flow and the necessary boundary conditions. The pressure-velocity relationship governed by Bernoulli's equation and the Kutta condition will also be introduced.

2.1 Scalar and Vector Potentials

Incompressible potential flow can be decomposed into two types of potential: A vector potential $\mathbf{\Psi}$ and a scalar potential ϕ [28]. The fluid velocity at any point in the domain can be represented as the superposition of the gradient of the scalar potential, curl of the vector potential, and the freestream velocity:

$$\mathbf{U} = \nabla\phi + \nabla \times \mathbf{\Psi} + \mathbf{V}_\infty \tag{2.1}$$

The continuity equation for an incompressible fluid is

$$\nabla \cdot \mathbf{U} = 0 \tag{2.2}$$

Equation 2.1 can be substituted into Equation 2.2 to yield Laplace's equation which governs inviscid, incompressible, and irrotational flow

$$\nabla^2\phi = 0 \tag{2.3}$$

since $\nabla \cdot V_\infty = 0$ and $\nabla \cdot (\nabla \times \Psi) = 0$. The vorticity field, $\boldsymbol{\omega}$, is simply the curl of the velocity field:

$$\boldsymbol{\omega} = \nabla \times \mathbf{U} \tag{2.4}$$

The evolution of vorticity in time can be derived from the incompressible Euler equation below:

$$\left(\frac{\partial}{\partial t} + \nabla \cdot \mathbf{U}\right) \mathbf{U} = -\frac{\nabla p}{\rho} \tag{2.5}$$

where p is the pressure and ρ is the fluid density. The vorticity equation is obtained by taking the curl of Equation 2.5 and rearranging:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{U} \cdot \nabla)\, \boldsymbol{\omega} = \frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\, \mathbf{U} \tag{2.6}$$

where $\frac{D()}{Dt}$ denotes the substantial derivative. The right hand side of Equation 2.6 represents vorticity stretching due to the presence of a velocity component in the vorticity direction. Note that this term will vanish in the 2D case since the vorticity vector and stream velocity will always be orthogonal.

2.2 Boundary Conditions

For the case of incompressible potential flow, the correct boundary condition is a no-penetration condition requiring that $\mathbf{U} \cdot \hat{n} = 0$ on the boundary

$$\mathbf{U} \cdot \hat{n} = (\nabla\phi + \nabla \times \boldsymbol{\Psi} + \mathbf{V}_\infty) \cdot \hat{n} = 0 \tag{2.7}$$

where $\hat{n}$ is the unit normal vector to the aerodynamic surface. The no-penetration condition can either be enforced explicitly with a Neumann boundary condition on the velocity or implicitly with a Dirichlet boundary condition on the potential inside the body, as will be discussed later.

2.3 Unsteady Bernoulli Equation

Pressure and velocity in an incompressible flow are related through the unsteady Bernoulli equation:

$$\frac{\partial \phi}{\partial t} + \frac{1}{2}||\mathbf{U}||^2 = \frac{\partial \phi}{\partial t} + \frac{1}{2}||\nabla\phi + \nabla \times \mathbf{\Psi} + \mathbf{V}_\infty||^2 = \frac{p}{\rho} \tag{2.8}$$

The coefficient of pressure, C_p, is defined as:

$$C_p = \frac{p - p_\infty}{p_\infty} \tag{2.9}$$

where $p_\infty = \frac{1}{2}V_\infty^2$ is the freestream dynamic pressure. The coefficient of pressure can be re-written in terms of Equation 2.8 as

$$C_p = \frac{\frac{\partial \phi}{\partial t} + \frac{1}{2}||\nabla\phi + \nabla \times \mathbf{\Psi} + \mathbf{V}_\infty||^2}{\frac{1}{2}||\mathbf{V}_\infty||^2} - 1 \tag{2.10}$$

2.4 Kutta Condition

Inviscid lifting flows require invoking the Kutta condition to enforce pressure continuity at the trailing edge [25]. This work imposes a linearised Kutta condition which requires that the strength of the wake sheet potential ϕ_w must equal the jump in potential from the upper trailing edge surface to the lower trailing edge surface [14]:

$$\phi_u - \phi_l = \phi_w \tag{2.11}$$

Note that a nonlinear Kutta condition could be imposed instead, where the pressure at the upper surface of the trailing edge is forced to equal the pressure at the lower surface using a Newton method. A nonlinear Kutta condition is necessary to capture highly unsteady effects, but since this work is primarily concerned with steady-state solutions, it has not yet been implemented.

Chapter 3

Discretization

The potential flow equation is discretized in space using the Boundary Element Method and in time using an explicit Runge-Kutta method. The spatial and temporal discretization schemes implemented in this work are detailed below.

3.1 Boundary Integral Equations

Laplace's equation is discretized using the BEM based on the Morino formulation [17]. Morino's formulation ensures that the *perturbation* potential is equal to zero inside the aerodynamic body, forcing the surface to become a streamsurface. In practice, this is accomplished by imposing a Dirichlet boundary condition on the perturbation potential an infinitesimal distance inside the surface. A diagram of the surfaces, potentials, and vorton representation used in this work is shown below.

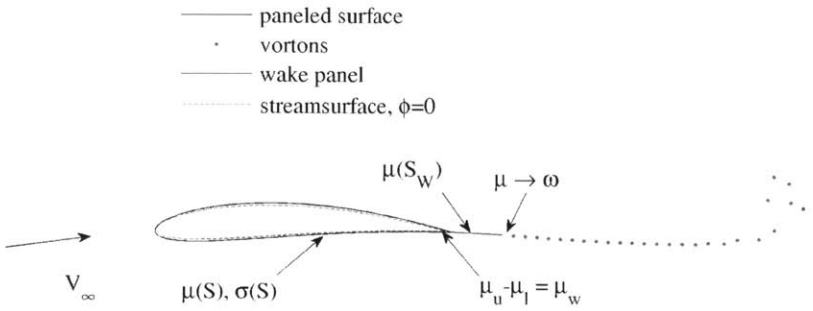

Figure 3-1: Potential shed into the wake in a direction that bisects the trailing edge with a strength equal to the jump in potential between the upper and lower trailing edge surfaces. The wake doublet potential is then converted into discrete vortons which are propagated in time.

The perturbation potential at some point $\mathbf{r}$ just inside the aerodynamic body, due to double-layer (doublet) densities distributed over the aerodynamic surface S and wake surface S_W, and the single-layer (source) densities distributed over the aerodynamic surface, is the superposition of the two potentials and must equal zero inside the surface.

$$\phi(\mathbf{r}) = \int_S \left(G_d(\mu, r) + G_s(\sigma_s(\mathbf{r}), r)\right) + \int_{S_W} G_d(\mu, r) = 0 \tag{3.1}$$

Here G_d is the Green's function for the double-layer potential satisfying Laplace's equation and G_s is the corresponding kernel for the single-layer potential. r is the scalar distance between $\mathbf{r}$ and a point $\mathbf{r}_s$ on surface S. The wake surface S_W is required to accurately enforce the Kutta condition. The Green's function for single-layer potential due to a source of strength σ is:

$$G_s(\sigma(\mathbf{r}), r) = \frac{1}{4\pi}\frac{\sigma(\mathbf{r})}{r} \tag{3.2}$$

The corresponding Green's function for double-layer potential with strength μ and

surface normal $\hat{n}_s$ is:

$$G_d(\mu, r) = \frac{1}{4\pi} \frac{\partial}{\partial \hat{n}_s} \frac{\mu}{r} \tag{3.3}$$

The source strength σ_s in Equation 3.1 is chosen to equal the component of the wake and freestream velocity influences in Equation 2.1 in the direction normal to the surface:

$$\sigma_s(\mathbf{r}) = (\mathbf{V}_\infty + \mathbf{V}_\Psi(\mathbf{r})) \cdot \hat{n}_s \tag{3.4}$$

where the second term is the vector potential velocity contribution from the wake vortons:

$$\mathbf{V}_\Psi(\mathbf{r}) = \int_D \nabla \times \mathbf{\Psi}(\boldsymbol{\omega}, r) \tag{3.5}$$

Here D denotes the entire domain. The vector potential is defined as

$$\mathbf{\Psi}(\boldsymbol{\omega}, r) = \frac{1}{4\pi} \frac{\boldsymbol{\omega}}{r + \epsilon} \tag{3.6}$$

where ϵ is a parameter that removes the vorton singularity and mimics a viscous core. The volume vorticity term $\boldsymbol{\omega}$ is a function of the shed doublet strength μ prescribed by the Kutta condition. Therefore, the only independent variable in Equation 3.1 is the doublet strength μ. The conversion of the wake vorticity sheet to volume vorticity is thoroughly detailed in Willis, et al. [7]. Finally, the velocity at any point $\mathbf{r}$ given in Equation 2.1 can be written in terms of Equations 3.1, and 3.5 as:

$$\mathbf{U}(\mathbf{r}) = \int_S \nabla \big(G_d(\mu, r) + G_s(\sigma_s(\mathbf{r}), r)\big) + \int_{S_W} \nabla G_d(\mu, r) + \int_D \nabla \times \mathbf{\Psi}(\boldsymbol{\omega}, r) + \mathbf{V}_\infty \tag{3.7}$$

One of the advantages of the Morino formulation is that the surface doublet density distribution is equal to potential distribution, and surface velocities can be obtained by simply differentiating the surface doublet density. Note that the Morino formulation allows the Kutta condition to simply be written as $\mu_u - \mu_l = \mu_w$.

3.2 Surface Discretization

A Galerkin method is implemented to discretize the Boundary Integral Equations in space, in a similar approach to the standard Finite Element Method. A Galerkin method was chosen over collocation due to increased accuracy [25]. Let Ω be a boundary in $\mathcal{R}^3$ composed of S and $\mathcal{T}_h$ be a collection of elements representing the triangulation of Ω. Furthermore, let Ω^- be the limit of Ω as approached from the the interior of the surface S. The Dirichlet boundary condition on the perturbation potential given in Equation 3.1 can now be written in terms of this new notation as:

$$\phi(\mu) = 0 \quad \text{in } \Omega^- \times (0, T] \tag{3.8}$$

The following approximation spaces are introduced, which reside in each element K in $\mathcal{T}_h$, and are used to represent the potential solution μ and the test function w:

$$W_h = \left\{ w \in C^0(\Omega^-) : w|_K \in P_k(K), \forall K \in \mathcal{T}_h \right\} \tag{3.9}$$

where $P_k(K)$ is the space of polynomials of degree k which reside in K and $C^0(\Omega^-)$ denotes the space of piecewise-continuous functions on the boundary Ω^-. In the current implementation, these polynomials are chosen to be the set of orthonormal Koornwinder polynomials in $\mathcal{R}^2$. The projection of Equation 3.1 onto a test function $w \in W$ ensures that the solution is orthogonal to the test function:

$$\langle \phi_K(\mu), w \rangle_K = 0, \quad \forall w \in W(K) \tag{3.10}$$

where, for compactness, the inner product is defined as $\langle a, b \rangle_K = \int_K ab$. The potential on each element K in Equation 3.10 is:

$$\phi_K(\mu) = \int_{S \cup S_W} \left(G_d^K(\mu, r) + G_s^K(\sigma_s(\mathbf{r}), r) \right) \tag{3.11}$$

where $\sigma_s(\mathbf{r})$ is determined from the freestream and wake velocities and also depends on the surface normal $\hat{n}_s$. Formally, the Galerkin method seeks to find $\mu_h \in W_h$ that

satisfies the projection of Equation 3.8:

$$\langle \phi_K(\mu_h), w \rangle_{\mathcal{T}_h} = 0, \quad \forall w \in W(\mathcal{T}_h) \tag{3.12}$$

which results a system of equations consisting of only the unknown doublet strengths μ_h.

3.3 Discretization in Time

One of the advantages of this particular boundary element formulation is that the governing equation (Equation 3.1) does not explicitly include a time-derivative term. This is a result of the fact that the flow is incompressible and that the only condition being enforced is the no-penetration condition on velocity [14]. However, there is an implicit dependence on time since the source strength $\sigma(\mathbf{r})$ depends on the wake position and vorticity strength which are permitted to evolve in time. The time-discretization of the governing equation then reduces to the time-discretization of the wake only.

The positions of the wake vortons at time t are denoted as $\mathbf{x}(t)$ and convect with the local velocity $\mathbf{U}(\mathbf{x}, t)$ according to the system of ODEs:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{U} \tag{3.13}$$

Similarly, the time-evolution of the vorticity governed by Equation 2.6 is

$$\frac{D\boldsymbol{\omega}}{Dt} = (\boldsymbol{\omega} \cdot \nabla)\mathbf{U} \tag{3.14}$$

The above two equations are integrated in time with a constant time step of Δt using an explicit Runge-Kutta method with s stages. All results presented in this work were obtained using the 1 stage explicit (Forward Euler) Runge-Kutta method, although any of the implicit or explicit Runge-Kutta methods could be used instead to obtain increased temporal accuracy and stability. The descretization of the wake evolution

using Forward Euler is:

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} = \mathbf{U}(\mathbf{x}^n) \tag{3.15}$$

where the superscript n denotes the time step. Similarly, the discrete vorticity evolution equation is:

$$\frac{\boldsymbol{\omega}^{n+1} - \boldsymbol{\omega}^n}{\Delta t} = (\boldsymbol{\omega}^n \cdot \nabla)\mathbf{U}(\mathbf{x}^n) \tag{3.16}$$

Chapter 4

High-Order Element Integration

Curved element integration is performed using adaptive Gaussian quadrature. The solution on each element is represented with basis functions which are chosen to be the set of orthonormal Koornwinder polynomials of degree k defined in reference space $\xi - \eta$. The geometry of each element is defined by the mapping $\mathbf{X}(\xi, \eta)$ from the reference triangle to the physical element. Two levels of integration are needed. At the innermost level the single and double-layer kernels must be evaluated on each of the elements for each Gauss point used to integrate the test functions. The outer level is the integration over the test functions.

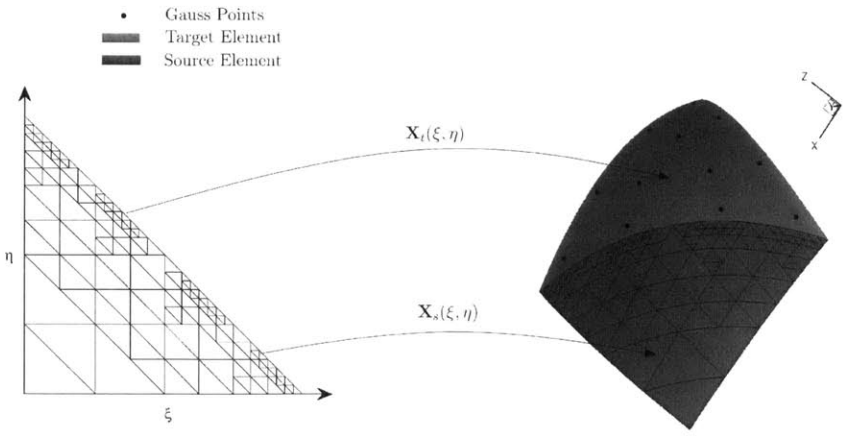

Figure 4-1: Subdivision of two neighboring elements of order $k = 2$ showing the subdivided master element (L) and the subdivided source and target elements (R).

Since the self-term integrands are singular and the near-field interactions are close to singular, standard numerical quadrature approaches will fail in these cases. It is easy to compute the rate of integrand decay for both single and double-layer potentials, and this rate can be used to create an efficient quadrature algorithm. The following adaptive quadrature scheme is proposed.

for $K_t \in \mathcal{T}_h$ **do**
  for $\mathbf{x}_t \in K_t$ **do**
    move target gauss points a small distance τ inside surface
    for $K_s \in \mathcal{T}_h$ **do**
      for $\mathbf{x}_s \in K_s$ **do**
        $r = ||\mathbf{x}_t - \mathbf{x}_s||$
        if $f_K(\mathbf{r}) < C$ **then**
          Evaluate $G_s(\sigma, r)$ and $G_d(\mu, r)$.
        else
          Divide K
          Compute trial functions on divided K
          go to [for $\mathbf{x}_s \in K_s$ **do]**
        end if
      end for
      Integrate trial functions over K_s
    end for
  end for
  Integrate test functions over K_t
end for

where the subscripts s and t denote the "source" and "target" elements and $\mathbf{x}$ are the Gauss points belonging to each element K. $f_K(\mathbf{r})$ is a distance function which dictates the threshold for which a source element should be divided. In three dimensions the potential due to the single-layer kernel decays at a rate of $\frac{A_K}{r}$ while the double-layer kernel decays at a rate of $\frac{(\mathbf{r}\cdot\hat{n})A_K}{r^3}$ where A_K is the area of element K. A reasonable choice for a cut-off function should satisfy $\max\{\frac{(\mathbf{r}\cdot\hat{n})A_K}{r^3}, \frac{A_K}{r^r}\} \leq C$ where C is a constant

indicative of the quadrature error committed. In this work, the following threshold function is implemented:

$$f_K(\mathbf{r}) = \max\left\{\frac{(\mathbf{r}\cdot\hat{n})A_K}{r^3}, \frac{A_K}{r^r}\right\} \tag{4.1}$$

$C = 1$ and $\tau = 10^{-10}$ have proven to be sufficient parameters for the test cases considered in this work.

Chapter 5

Steady-State Solver

In this section, a method is developed to solve the nonlinear free-wake steady-state potential flow problem. The nonlinearity arises from the dependence of the doublet potential on the vorton positions and strengths. Currently, the nonlinear system if solved with the standard Newton method if FMM acceleration is not used. For cases where the FMM method is used to accelerate wake interactions, a Jacobian-free Newton method is used since the resulting analytical Jacobian would be cumbersome to implement.

5.1 The Residual Vector

There are three equations governing the potential flow model detailed above: (1) the condition that the perturbation potential vanishes just inside the body (Equation 3.1), (2) the ODE governing the wake evolution (Equation 3.13), and (3) the ODE governing the vorticity evolution (Equation 3.14). These equations can be written as a system consisting of the N_μ surface doublet distribution degrees of freedom, the three components of the N_V vorton spatial coordinates, and the three components of the N_V vorton vorticity vector. A residual vector written from the three governing equations mentioned above can be defined as:

$$\mathbf{F} = \begin{bmatrix} \Phi_1 & \Phi_2 & \dots & \Phi_{N_\mu} & \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_{N_V} & \mathbf{W}_1 & \mathbf{W}_2 & \dots & \mathbf{W}_{N_V} \end{bmatrix}^T = 0 \quad (5.1)$$

where Φ is the residual resulting from the Galerkin BEM in Equation 3.12, $\mathbf{X}$ is a residual from the Forward Euler discretization of Equation 3.13, and $\mathbf{W}$ is the residual from the forward Euler discretization of Equation 3.14. The vorton position residual at time step n is then

$$\mathbf{X}^n = \frac{\mathbf{x}^n - \mathbf{x}^{n-1}}{\Delta t} - \mathbf{U}(\mathbf{x}^{n-1}) = 0 \tag{5.2}$$

Assuming that the wake behavior is non-chaotic, the wake can be forced into lock-step, providing the following relationship:

$$\mathbf{x}_j^n = \mathbf{x}_{j-N_s}^{n-1}, \quad j \in \{N_s + 1, ..., N_V\} \tag{5.3}$$

where N_s is the number of vortons shed at each time step. Since the system is in lock-step, the superscripts can be eliminated from the Forward Euler discretization and the residual vector can be written in terms of the indices $\mathbf{x}_{j-N_s}$ and $\mathbf{x}_j$ instead. The wake evolution equation can then be re-writen as:

$$\mathbf{X}_j = \frac{\mathbf{x}_j - \mathbf{x}_{j-N_s}}{\Delta t} - \mathbf{U}(\mathbf{x}_{j-N_s}) = 0, \quad j \in \{N_s + 1, ..., N_V\} \tag{5.4}$$

The same principal can be used to write a residual vector for the vorticity evolution equation in lock-step:

$$\mathbf{W}_j = \frac{\boldsymbol{\omega}_j - \boldsymbol{\omega}_{j-N_s}}{\Delta t} - (\boldsymbol{\omega}_j \cdot \nabla)\mathbf{U}(\mathbf{x}_{j-N_s}) = 0, \quad j \in \{N_s + 1, ..., N_V\} \tag{5.5}$$

5.2 The Jacobian

The Jacobian of the residual vector with respect to the vector of unknowns $\mathbf{u}$ is

$$J = \frac{\partial \mathbf{F}}{\partial \mathbf{u}} \tag{5.6}$$

where

$$\mathbf{u} = \begin{bmatrix} \mu_1 & \mu_2 & \dots & \mu_{N_\mu} & \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{N_V} & \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \dots & \boldsymbol{\omega}_{N_V} \end{bmatrix}^T \tag{5.7}$$

Differentiating $\mathbf{F}$ with respect to $\mathbf{u}$ yields the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \Phi}{\partial \mu} & \dfrac{\partial \Phi}{\partial \mathbf{x}} & \dfrac{\partial \Phi}{\partial \boldsymbol{\omega}} \\ \dfrac{\partial \mathbf{X}}{\partial \mu} & \dfrac{\partial \mathbf{X}}{\partial \mathbf{x}} & \dfrac{\partial \mathbf{X}}{\partial \boldsymbol{\omega}} \\ \dfrac{\partial \mathbf{W}}{\partial \mu} & \dfrac{\partial \mathbf{W}}{\partial \mathbf{x}} & \dfrac{\partial \mathbf{W}}{\partial \boldsymbol{\omega}} \end{bmatrix} \tag{5.8}$$

and the following system is solved for the update vector $\delta\mathbf{u}$ at each step of the Newton method until the L^2-norm of $\mathbf{F}$ is sufficiently small.

$$\mathbf{J}\delta\mathbf{u} = -\mathbf{F} \tag{5.9}$$

The terms of the Jacobian matrix in Equation 5.8 are somewhat tedious to derive and are therefore listed in Appendix A. Additionally, it should be noted that the Jacobian is dense and grows rapidly in size with the number of vortons since there are 6 degrees of freedom associated with each vorton.

5.3 Jacobian-Free Newton Method

The Jacobian-free Newton method is used for cases where the Jacobian cannot be easily computed, as is the case when the Fast Multipole Method is invoked, and where the analytical Jacobian is too large to store in memory. The Jacobian-free Newton method approximates the Jacobian with a finite difference along a search direction. A good search direction is obtained by a Krylov subspace solver. The Jacobian is approximated as:

$$\mathbf{J} \approx \frac{\mathbf{F}(\mathbf{u}) - \mathbf{F}(\mathbf{u} + \epsilon\mathbf{s})}{\epsilon} \tag{5.10}$$

where $\mathbf{s}$ is the direction obtained from driving a Krylov subspace solver residual to a specified tolerance α. $\epsilon = 10^{-8}$ and $\alpha = 10^{-2}$ have proven to be sufficient for the cases encountered in this work.

Chapter 6

The Fast Multipole Method

This section presents the theory of the Fast Multipole Method and it's application to the panel method presented in this work. A more detailed description of the method and associated translation operators can be found in Appendix B. The FMM is composed of three primary routines: the octree domain decomposition, the upward pass, and the downward pass. Each of the three routines are detailed below. Figure 6-2 depicts the primary steps of the FMM algorithm.

The cost of the FMM implemented in this work scales as $O(N)$ were N is the number of elements in the FEM triangulation. This has a lower cost than previously implemented aerodynamic FMM-accelerated panel methods such as [25] and [3] which scaled as $O(N \log N)$. The reduction in operation count compared to these methods is obtained by implementing the translation operators described below, at the cost of increased implementation complexity.

6.1 Octree Decomposition

A cubic domain is generated spanning all the sources, doublets, and vortons in space. The cube is recursively divided into eight boxes, until there are no more than N_{MAX} elements in a box. Boxes with no elements in them are deleted. The FMM method requires that each cell know who its *nearest neighbors* and *second nearest neighbors* are. A nearest neighbor is defined as a box which touches another box, even if they are

just touching at a point. Nearest neighbors can be easily computed by traversing down the octree, searching through the *children* of each box's *parent*. The second nearest neighbors can be computed by searching through the nearest neighbors of each box's nearest neighbors. The computational cost of the octree domain decomposition can range from $O(N \log N)$ to $O(N)$, depending on the homogeneity of the singularity distributions where N is the sum of the number of elements and vortons in the domain.

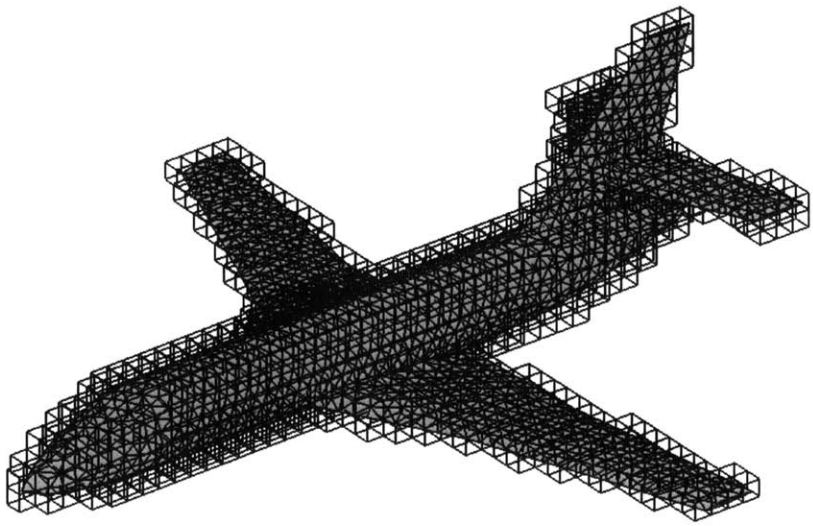

Figure 6-1: All childless boxes in an octree decomposition about a business jet mesh.

6.2 Upward Pass

The upward pass is composed of two steps: (1) Multipole Generation and (2) the Multipole-to-Multipole ($M \rightarrow M$) translation. In the Multipole Generation step, multipole expansion coefficients are computed due to all sources, dipoles, and vortons in each *childless* box. This will result in five expansion coefficients for each box, corresponding to the source distribution, doublet distribution, and three components of vorticity due to the vortons. The complexity of this step is $O(Np^2)$, where p is the order of the multipole expansion. The Multipole-to-Multipole translation is the recursive translation of the multipole coefficients up the tree, starting with the childless boxes. The computational complexity of this step is $O(Np^4)$.

6.3 Downward Pass

The downward pass is composed of four steps: (1) the Multipole to Local ($M \rightarrow L$) translation,(2) the Local to Local translation ($L \rightarrow L$), (3) the Direct Evaluation, and (4) the Local Expansion evaluation. The ($M \rightarrow L$) translation involves the conversion of all the multipole expansions of the boxes in a box's *interaction list* into a Taylor series expansion about the box's center. The interaction list can be chosen to be all the second nearest neighbors of a box, or a more complicated criteria [12]. This is the most expensive part of the FMM algorithm due the large number of boxes in the second nearest neighbor list and since this operation has a complexity that scales as $O(Np^4)$. The $L \rightarrow L$ translation is a recursive translation of local coefficients down the octree from each box to it's child. The complexity of this operation scales as $O(Np^4)$. The Direct Evaluation step computes the interaction between all sources, dipoles, and vortons in a box and that box's nearest neighbors directly without using multipole expansions. The complexity of this operation is $O(Ns)$, where s is a measure of the cost required to analytically integrate the source and dipole distribution over a element. The final step is the evaluation of the Local Expansions in each childless box. This step has a complexity of $O(Np^2)$ since it is not a translation operation, only a local evaluation.

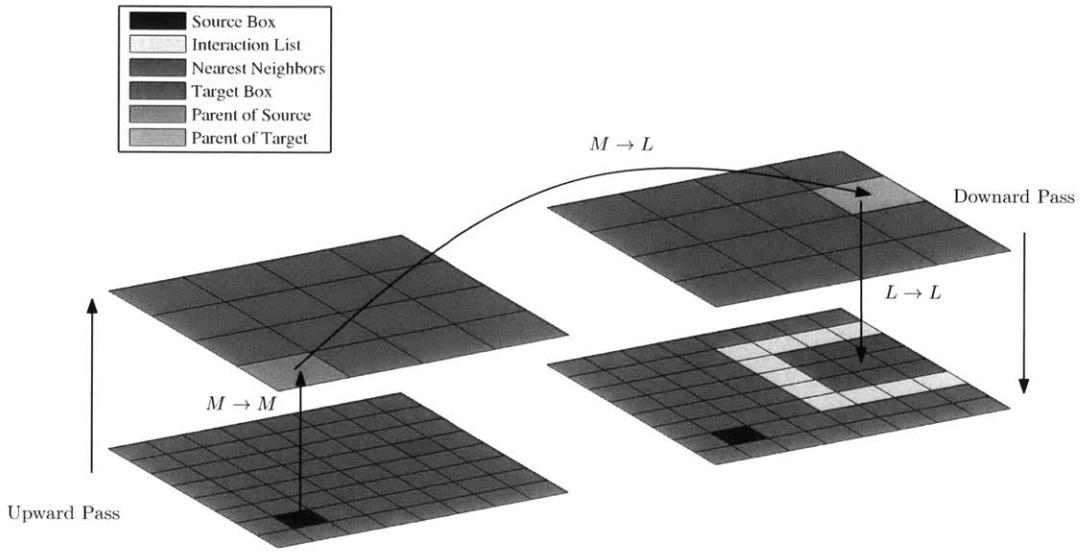

Figure 6-2: Diagram of FMM translations and interaction lists between two levels of a 2D grid. Note that the grids on the right and left are identical, and are duplicated for the purpose of visualizing the upward and downward passes.

Chapter 7

Implementation Details and Results

This section discusses implementation and presents results including demonstration of high-order convergence, validation of the nonlinear steady-state solver, and the extension of Fast Multipole accelerated BEM to large problems.

7.1 Implementation

The high order solver is written in C++ and heavily relies on the Armadillo C++ linear algebra library [21] which provides an intuitive syntax and simple access to BLAS routines which are typically cumbersome to invoke. The Armadillo library is linked to Intel's Math Kernel Library for best performance. Additionally, the matrix assembly routine is parallelized with OpenMP to reduce runtimes. Armadillo provides a simple, MATLAB-like syntax that allows for easy prototyping. An example of a code fragment written in Armadillo is shown below:

```
timer.tic();
state.mu = solve(pmats.d,-pmats.s*sigma);
cout << "solution time: " << timer.toc() << endl;

// Compute the potential gradient
mat gradsol =  solution_gradient(master, mesh.elements,mesh.nodes, state.mu);
```

Figure 7-1: Code fragment taken from the steady-state solver routine.

7.2 High Order Solver

Convergence results for the potential flow around the unit sphere are presented using increasing orders of polynomial approximation k. The analytical solution is known, and the polynomial degree is increased from $k = 1$ to $k = 4$ while uniformly refining the mesh from $n = 32$ to $n = 512$ elements. The meshes were generated using the open-source mesh generation software GMSH [5, 1]. All errors are computed in the L^2- norm.

Table 7.1: Convergence of L^2 norm for $\tau = 10^{-10}$

Degree	Mesh	$\|\|\phi - \phi_e\|\|_{L^2}$		$\|\|C_P - C_{P_e}\|\|_{L^2}$		$\|\|S - S_e\|\|_{L^2}$		time	
k	n	Error	Order	Error	Order	Error	Order	seconds	order
1	32	4.5520e-02	-	1.0900e+00	-	3.4498e-01	-	1.9545e-01	-
1	128	6.1641e-03	2.8845	5.4974e-01	0.9875	9.8850e-02	1.8032	5.2792e-01	1.4335
1	512	1.1592e-03	2.4108	2.7628e-01	0.9926	2.5515e-02	1.9539	1.8446e+00	1.8049
2	32	1.8532e-02	-	2.2278e-01	-	2.4082e-02	-	1.5543e+00	-
2	128	2.7602e-03	2.7472	9.5314e-02	1.2248	2.9575e-03	3.0255	7.3099e+00	2.2336
2	512	3.4791e-04	2.9879	2.8276e-02	1.7531	3.6802e-04	3.0065	3.4328e+01	2.2315
3	32	6.4849e-04	-	1.4817e-02	-	8.0852e-04	-	4.8245e+00	-
3	128	3.9157e-05	4.0497	1.7070e-03	3.1177	5.2930e-05	3.9331	2.2005e+01	2.1893
3	512	2.4267e-06	4.0122	2.1460e-04	2.9917	3.3540e-06	3.9801	9.6249e+01	2.1290
4	32	5.8916e-05	-	1.7437e-03	-	6.4195e-05	-	1.6343e+01	-
4	128	1.8612e-06	4.9844	1.6132e-04	3.4342	1.8704e-06	5.1011	6.6156e+01	2.0172
4	512	5.7827e-08	5.0084	1.1044e-05	3.8685	5.8698e-08	4.9939	2.8882e+02	2.1262

Table 1 shows that the potential converges at the optimal rate of $k + 1$, while the coefficient of pressure only converges with rate k. This loss of an order is expected as the coefficient of pressure is obtained by differentiation of the surface potential. Additionally, the geometry converges with the optimal rate of $k + 1$. Note that the computational time grows approximately quadratically with the element size and linearly with the number of elements, since the cost of computing self-term influences

is far greater than computing far-field terms.

In order to test a slightly more challenging case the pressure coefficient about a finite aspect ratio wing at $\alpha = 0$ is compared to a well-resolved XFOIL [9] solution. The comparison between the current three-dimensional solver and XFOIL (which is a 2D code) should be valid provided the aspect ratio is large enough, since the angle of attack is zero and hence there are essentially no three-dimensional effects. A series of meshes of a rectangular wing with a NACA0012 airfoil and aspect ratio of eight were generated in GMSH for increasing values of polynomial approximation k. Each of the meshes are composed of 1652 elements with refinement near the leading and trailing edges. The solutions obtained with the high order BEM are then compared to a grid-resolved XFOIL solution computed using 900 linear elements.

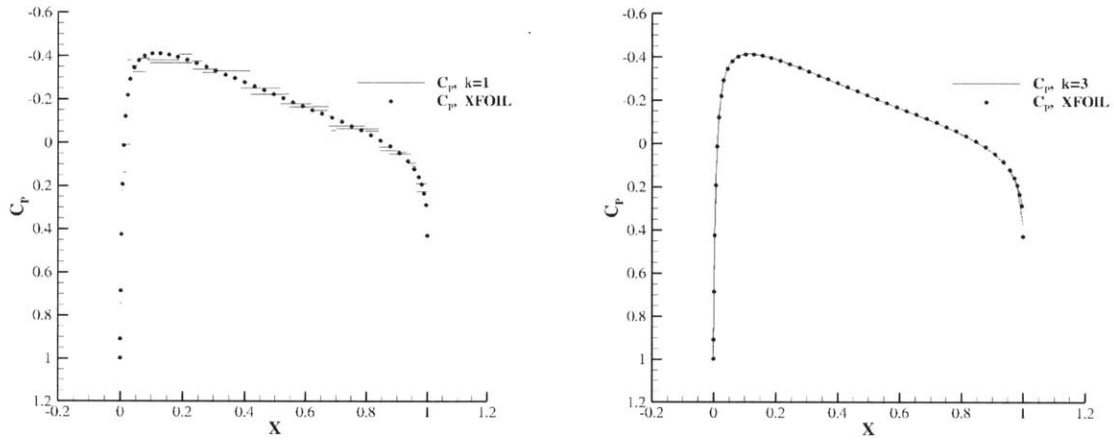

Figure 7-2: Coefficient of pressure for an AR=8 wing with 1652 elements at $\alpha = 0$ compared to XFOIL using 900 panels. $k = 1$ solution (L) and $k = 3$ solution (R).

It is clear that a significant error is committed in the coefficient of pressure for the standard linear ($k = 1$) panel method since the pressure is piecewise-constant on each element. However, the pressure distribution computed with $k = 3$ lies on top of the XFOIL solution, indicating that a third-order solution using the current mesh is k-converged.

Finally, results for the case of a lifting wing of aspect ratio 4 with a NACA0012 airfoil section are presented and compared to second-order-accurate incompressible

Euler solutions computed with FUN3D [11]. A low aspect ratio was chosen in order to test the ability of the current method to capture moderate 3D effects. The surface mesh specifications are identical to the non-lifting case presented above, except for the fact that the aspect ratio is now 4 instead of 8. For this case, a fixed, planar wake consisting of 20 stream-wise elements extending 10^3 chord lengths downstream is implemented. The FUN3D mesh consists of 4.1 million tetrahedral elements, with the wing boundary being composed of 221,862 triangular elements. The leading and trailing edge spacing for the FUN3D mesh is $c \cdot 10^{-3}$ where c is the chord length.

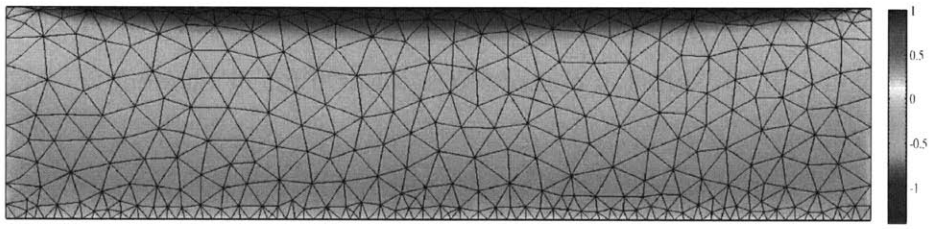

Figure 7-3: Top view of the C_P distribution at $\alpha = 5°$ computed with the current BEM solver for $k = 3$ and the corresponding mesh.

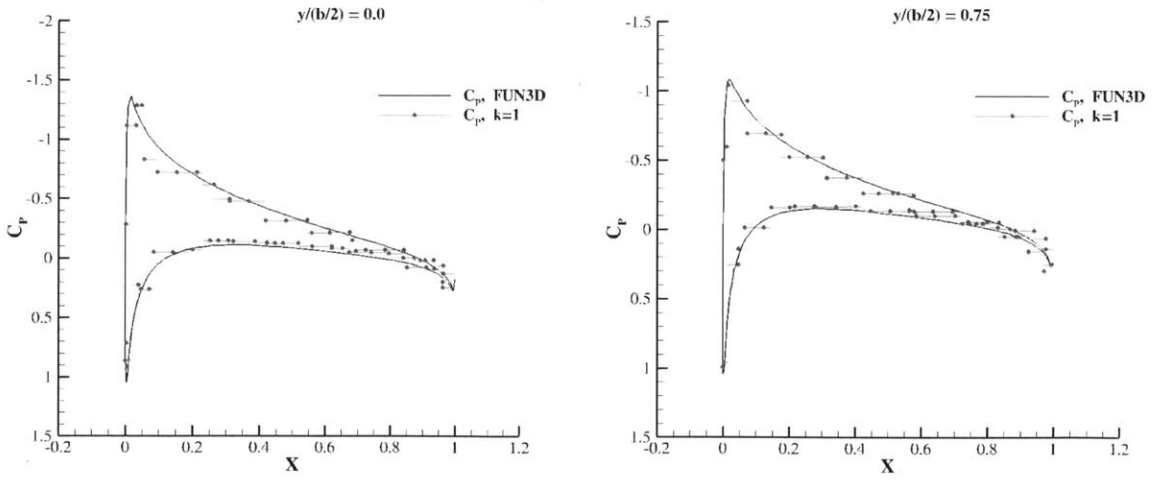

Figure 7-4: Coefficient of pressure distribution on an $AR = 4$ rectangular wing at $\alpha = 5°$ at the root (L) and 3/4 semispan (R) comparison with FUN3D computed using $k = 1$.

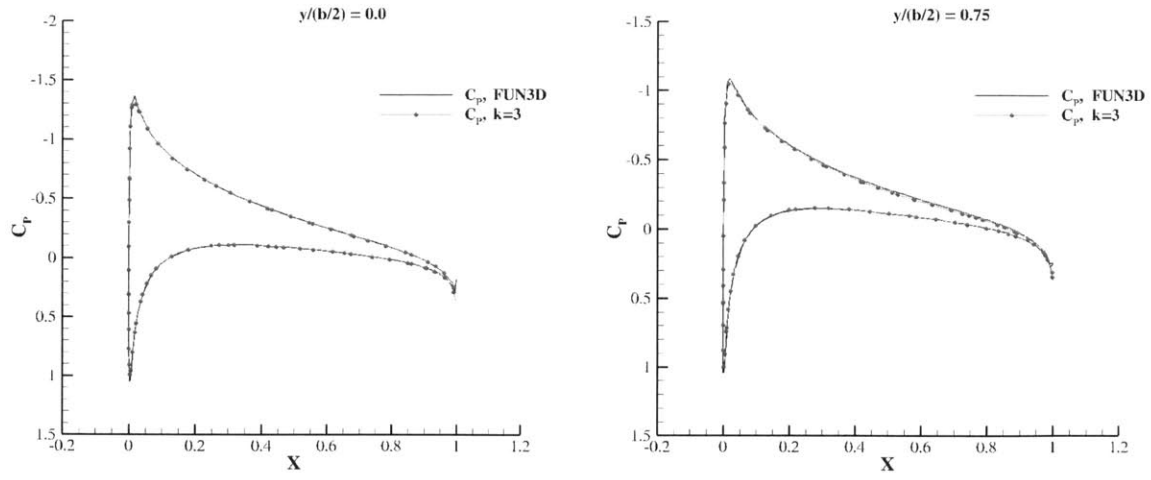

Figure 7-5: Coefficient of pressure distribution on an $AR = 4$ rectangular wing at $\alpha = 5°$ at the root (L) and 3/4 semispan (R) comparison with FUN3D computed using $k = 3$.

Figures 7-4 and 7-5 show the pressure distributions at two slice locations along the span, b, computed using $k = 1$ and $k = 3$, respectively. It can be observed that there is a significant error in the pressure distribution for the $k = 1$ case at both span-wise locations, but this error is significantly reduced by using a high polynomial degree. The pressure computed using $k = 3$ lies on top of the FUN3D solution at the wing root, and there is only a slight over-prediction of the upper surface pressure at 3/4 semi-span when compared to FUN3D. Note that a difference between the FUN3D solution and the current method should be expected since the results presented above are computed using a fixed planar wake which violates the force-free wake requirement. An angle of attack sweep is conducted in order to validate accuracy over a range of conditions. The angle of attack is increased from 0 to 12 degrees in two degree increments for $k = \{1, 2, 3\}$. The resulting lift and drag polars and shown in Figure 7-6.

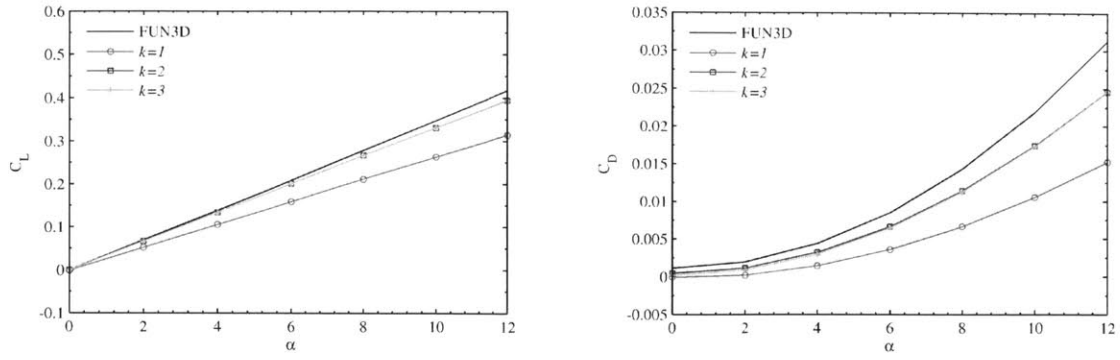

Figure 7-6: Lift polar (L) and drag polar (R) for several values of polynomial approximation k compared to the incompressible Euler solution computed with FUN3D.

The lift and drag computed with $k = 1$ differ significantly from the results obtained with FUN3D and higher order polynomial approximations. This indicates that for the current mesh, the solution is not $k-$converged with $k = 1$. The solutions computed with $k = 2$ and $k = 3$ are almost identical. For $k = 3$ the lift differs from the Euler solution by approximately 5% while the drag differs by 17% at $\alpha = 12°$.The discrepancy between the polars computed using the high order BEM and the Euler case is likely a result of using a fixed wake in place of the more expensive but also more accurate free wake. Although the drag computed with the potential solver differs from the Euler values, it exhibit the correct trend and grows with the square of the angle of attack. Note that the drag at $\alpha = 0$ obtained with FUN3D is not zero as it should be, and the $k = 2$ and $k = 3$ BEM solutions appear to be better at predicting zero lift at zero angle of attack.

7.3 Steady Solver

Two methods for obtaining a steady-state solution are presented. In the first, the wake is seeded with vortons extending in a plane from the trailing edge. A Newton solver is then invoked and the residual is driven to zero. However, this initial condition may not be sufficient for cases where the wake strongly interacts with bodies downstream. Therefore, a second method is presented whereby the wake is time-stepped for a

prescribed number of steps, and then the nonlinear steady-state solver is invoked. Additionally, either an exact or approximate Jacobian may be used depending on whether or not FMM acceleration is enabled.

7.3.1 Planar Initial Wake and Exact Jacobian

An $AR = 4$ swept wing with a leading edge sweep angle of $\Lambda = 15°$ and taper ratio of $\lambda = 0.8$ composed of 996 linear elements is used to validate the steady state solver against an unsteady solution. The wake is seeded with 100 lock-steps of vortons spaced $||\mathbf{V}_\infty||\Delta t = 0.2$ apart. A Newton solver with line search is used to drive the norm of the residual to a small tolerance, chosen to be 10^{-10}. Preconditioned GMRES [20] is used for the linear solve since it was found to be faster than a direct solver. GMRES without preconditioning is significantly slower than using a direct solver since the condition number of the system is generally between $O(10^7)$ and $O(10^9)$ (it was later discovered that the condition number could be substantially reduced by re-ordering the wake degrees of freedom). The LU decomposition of the Jacobian computed during the first Newton iteration is used as the preconditioner for the remainder of the Newton iterations. The steady-state solution and lift comparison with the unsteady solver are shown below.

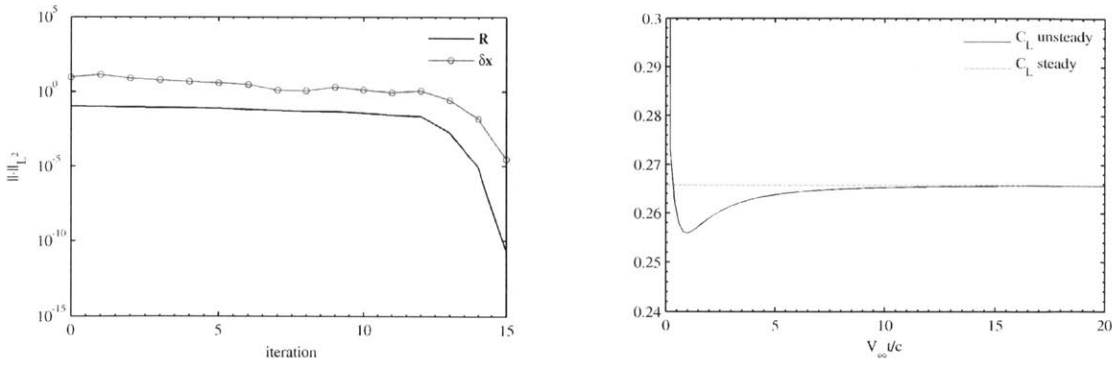

Figure 7-7: Newton solver residual convergence history (L). The unsteady lift solution compared to the steady-state value (R).

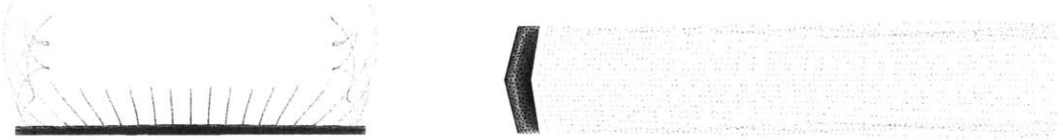

Figure 7-8: Potential distribution and wake rollup about an $AR = 4$ swept wing at $\alpha = 5°$ computed using the nonlinear steady-state code. Front view (L) and top view(R)

After $V_\infty t/c = 20$, the unsteady C_L is within 0.05% of the steady-state value, indicating that the unsteady solution is converging to the steady-state result. Figure 7-7 shows that the Newton solver only requires 15 iterations to converge to a tolerance of 10^{-10}. The solution time required for the case above was 865 seconds, 10 of which were spent computing the preconditioner and 3 of which were spent solving the linear systems with GMRES. The remainder of the time was spent in the Jacobian assembly routine, since the Jacobian must be completely re-assembled for each Newton iteration. 441MB of memory was required to store the dense Jacobian matrix.

7.3.2 Time-stepped Wake and Approximate Jacobian

Three cases are presented to demonstrate the robustness of the steady state solver. The first is for two second-order accurate wings of the same geometry as described in the previous section flying in tandem. The wings are separated by a distance of three root chord lengths. The second case is for the same second-order wing but with a sphere located at a distance of five root chord lengths downstream instead of a second wing. The last case is that of two first-order accurate Falcon business jets flying in tandem at an (unrealistically) close distance.

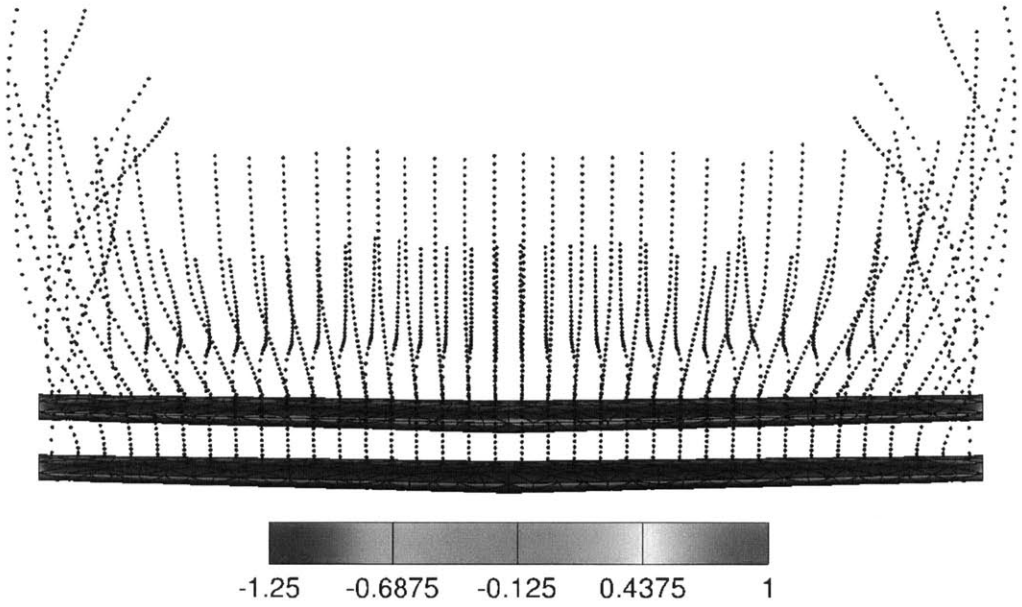

Figure 7-9: Steady-state C_P distribution and wake roll-up about two second-order wings in tandem computed after 60 time steps.

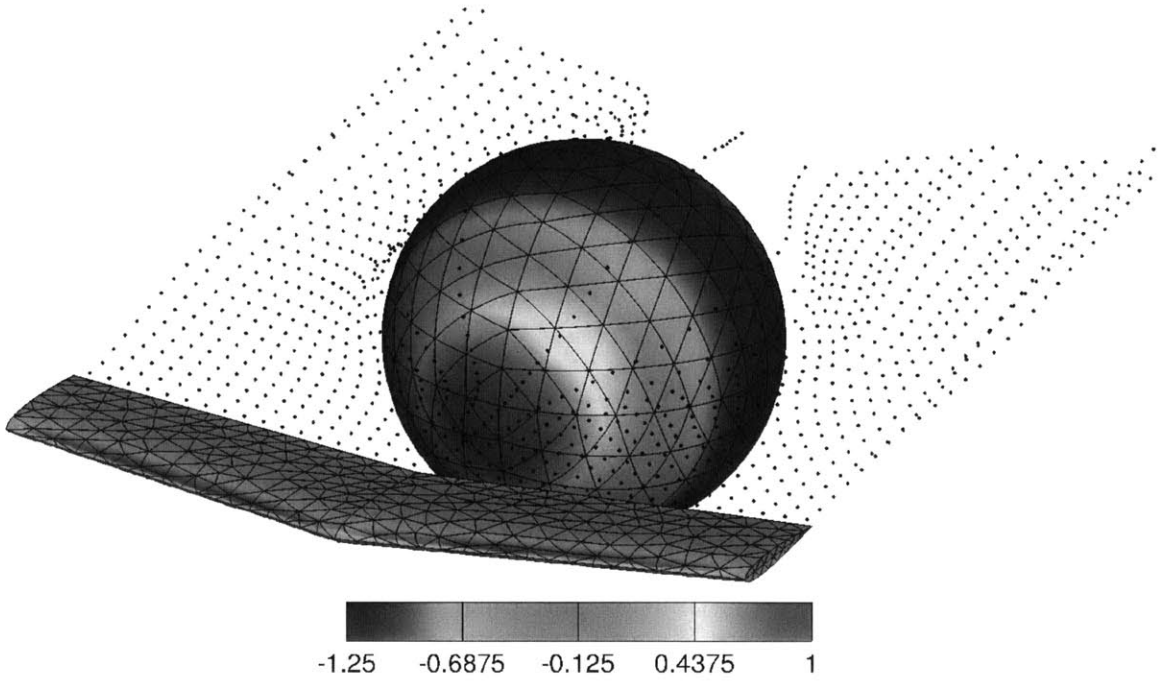

Figure 7-10: Steady-state C_P distribution and wake roll-up about a second order wing and sphere computed after 40 time steps.

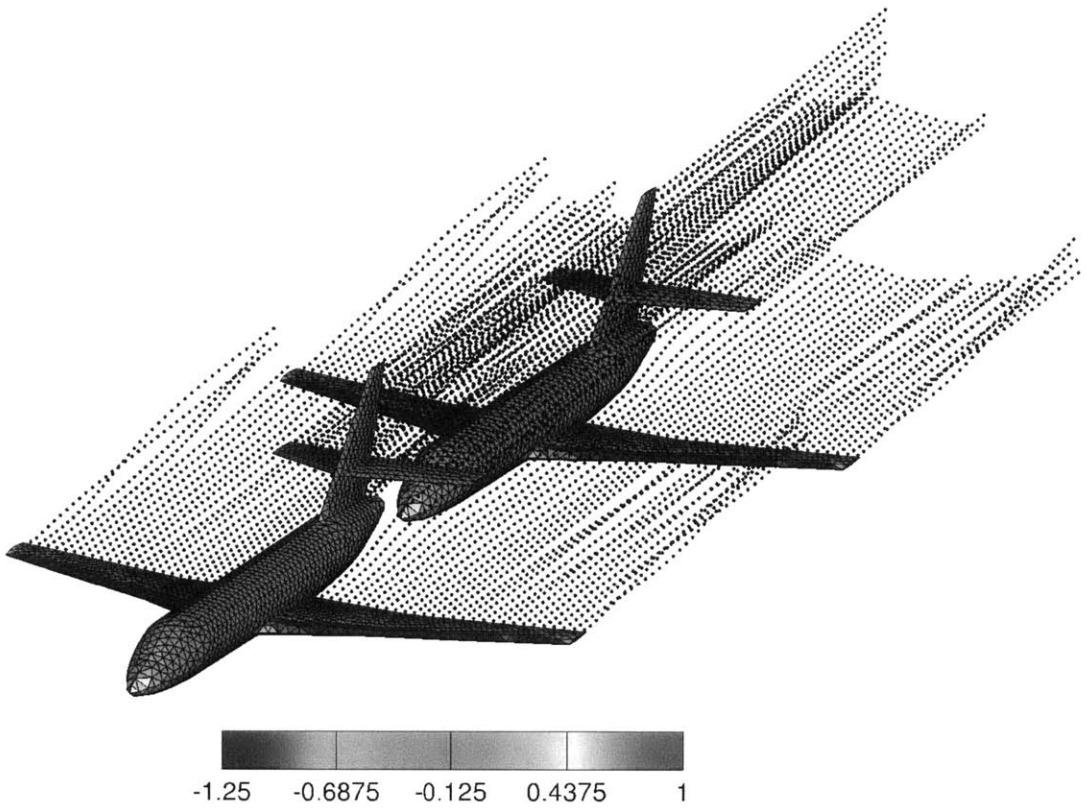

Figure 7-11: Steady-state C_P distribution and wake roll-up about two Falcon business jets flying in tandem computed after 50 time steps.

The above three cases demonstrate the robustness of the current wake model. For each of these cases, the wake is evolved for a specified number of time steps, after which the nonlinear fully-coupled solver is turned on. For the two-wing test case, the wake of the front wing travels over the aft wing, and interacts with the aft wing's wake. The wake shed from the wing in the wing-sphere case travels around the sphere as expected. Note that this case would be impossible to evaluate if linear panels were used to represent wake vorticity since the singular wake panels would intersect the sphere. The two-jet case demonstrates that the wake model is readily extendible to complex geometries, and the wake of the fore wing can be observed interacting with the second jet's surface and wake.

7.4 FMM Accelerated BEM

This section presents results for three cases of increasing complexity. The first is for flow about the unit sphere in order to demonstrate FMM convergence. The second is for non-lifting flow about a four-engine jet aircraft. The third is the case of a lifting business jet with a fixed wake.

Although the cases presented below all use linear meshes, the FMM is readably extendible to high order meshes as well. Solutions have been computed using the FMM-accelerated BEM with high-order meshes using the current solver. However, in the current implementation there is no benefit to coupling the two methods since the high-order integration techniques already scale as $O(N)$ due to the rate-limiting self-term influence integral. If a more efficient method for computing self-term integrals for high-order elements is developed in the future, then the benefits of both high order accuracy and the reduced FMM operation count could be realized at a fraction of the cost of the current high-order solver.

7.4.1 Sphere Convergence Study

The case of a unit sphere represented with 855 linear elements is chosen to demonstrate the convergence of the FMM method in multipole expansion order p. The error is quantified by computing the L^2-norm of the difference between the FMM solution and the solution generated with the direct BEM solver. Both the FMM and BEM system of equations are solved with GMRES to machine precision, and the only difference is in the approximate matrix-vector product of the FMM compared to the machine precision matrix-vector product of the direct BEM solver.

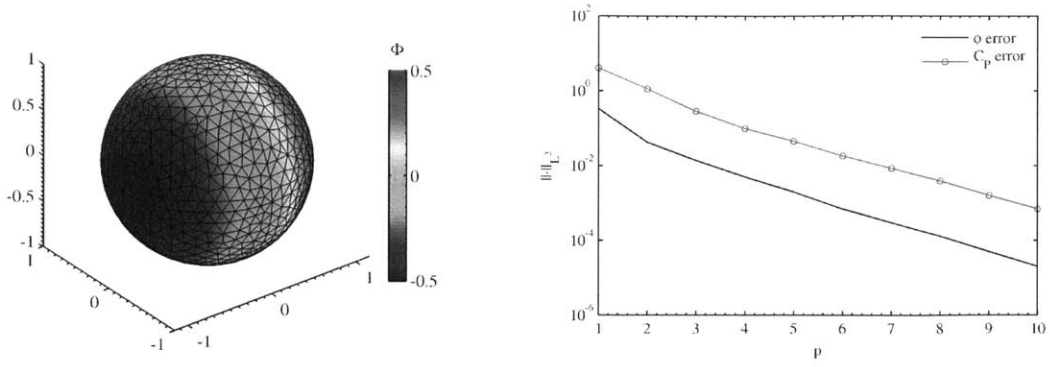

Figure 7-12: Potential over a sphere for $p = 2$ and $V_\infty = [1, 0, 0]$ (L) and error between the FMM and direct BEM (R).

Even with $p = 2$, the error in potential is small, however the error in pressure is more significant. This is due to the fact that the velocities are obtained by differentiation of the potential field, leading to a reduction in pressure accuracy. If a higher pressure accuracy is needed, the multipole order can be increased to obtain the desired fidelity. Note that the errors do not decay very fast in p. Therefore, if a high accuracy is desired then a sufficiently large value of p must be chosen. This can become expensive since the cost of the FMM presented in this work grows as $O(np^4)$.

7.4.2 Non-lifting Four-Engine Jet

In order to test a more complicated case, a four-engine jet represented with 175,712 linear elements is simulated using the FMM with a multipole expansion of order $p = 2$. The mesh was generated with the SUMO [22] mesh generator. This case does not include a wake influence, and is therefore non-lifting. The near-field interactions are computed directly, and stored in the box data structure once at the beginning of the simulation for efficiency. Additionally, a preconditioner is implemented in order to reduce the number of Krylov subspace iterations required.

Since the FMM provides a method of de-coupling near field and far-field terms, it also provides a very straightforward framework for generating a preconditioner. The preconditioner is applied in a box-by-box manner. For each childless box, the direct

influence matrix is assembled into a square matrix based on the Galerkin discretization. The resulting matrix is then inverted and stored in memory for the remainder of the computation. The FMM matrix-vector product is computed first, and the preconditioner is applied to the result by looping through the list of childless boxes.

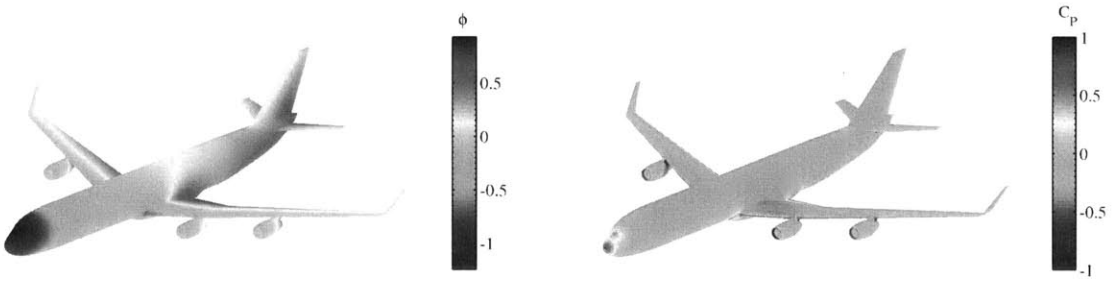

Figure 7-13: Potential solution about four-engine jet (L) and corresponding pressure distribution (R). Solution time is 240s using multipole expansion order $p = 2$.

A total of 39 GMRES iterations were required to reach a residual tolerance of 10^{-6}. The solution time for this rather complicated test case was 240 seconds using four processors. The memory required to store the preconditioner was 141MB, and the memory required to store the multipole coefficients and octree structure was 278MB. If this case were to be solved with the standard dense BEM, 62GB of memory would be required.

7.4.3 Lifting Business Jet

Lastly, case of lifting potential flow about a fine business jet mesh is demonstrated. A fixed panelled wake is implemented, extending to infinity. Since the wake extends to infinity, the octree decomposition of the domain required by the FMM cannot be generated. Therefore, the wake influence is computed directly. Note that this should not be problematic, since the wake influence matrix is much smaller than the full aerodynamic influence matrix would be. The mesh was generated in GMSH and consists of 31,398 linear elements. The process of generating the mesh from a .stp file was essentially hands-free, and only required specifying one parameter controlling

the mesh fineness. The pressure distribution at $\alpha = 5°$ is shown below.

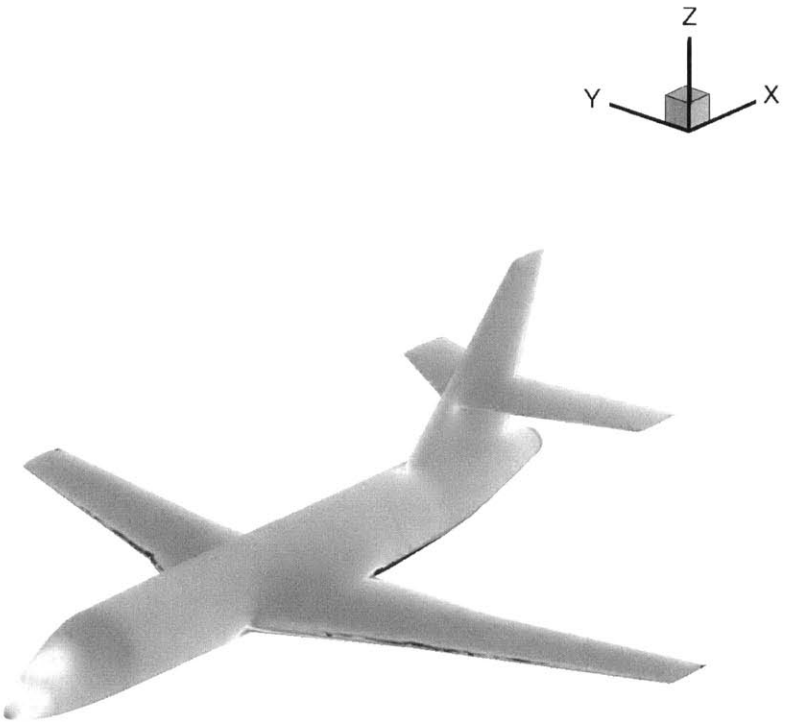

Figure 7-14: Interpolated pressure coefficient for a business jet operating at $\alpha = 5°$.

Note that there is a band of low pressure aft of the wing trailing edge. This is due to the edge of the wake dipole sheet being close to the fuselage surface. In the future, the wake sheet will intersect the fuselage elements and a jump in potential on the element will be permitted using discontinuous basis functions as described in [24]. Also note that the pressure distribution in figure 7-14 is continuous on the surface, and not piecewise-constant as in the previous results. This is a result of a post-processing step where the pressure is interpolated at each node from the surrounding elements. This is common practice for aerodynamic panel methods since it creates a more visually appealing solution, although it does not increase the pressure accuracy.

Also note that the maximum pressure coefficient is only approximately 0.9, and this only occurs at the wing root. The pressure coefficient should be 1.0 at some point along all leading edges, since there should always be a stagnation point near the leading edge. This effect is more clearly observed in the figure below.

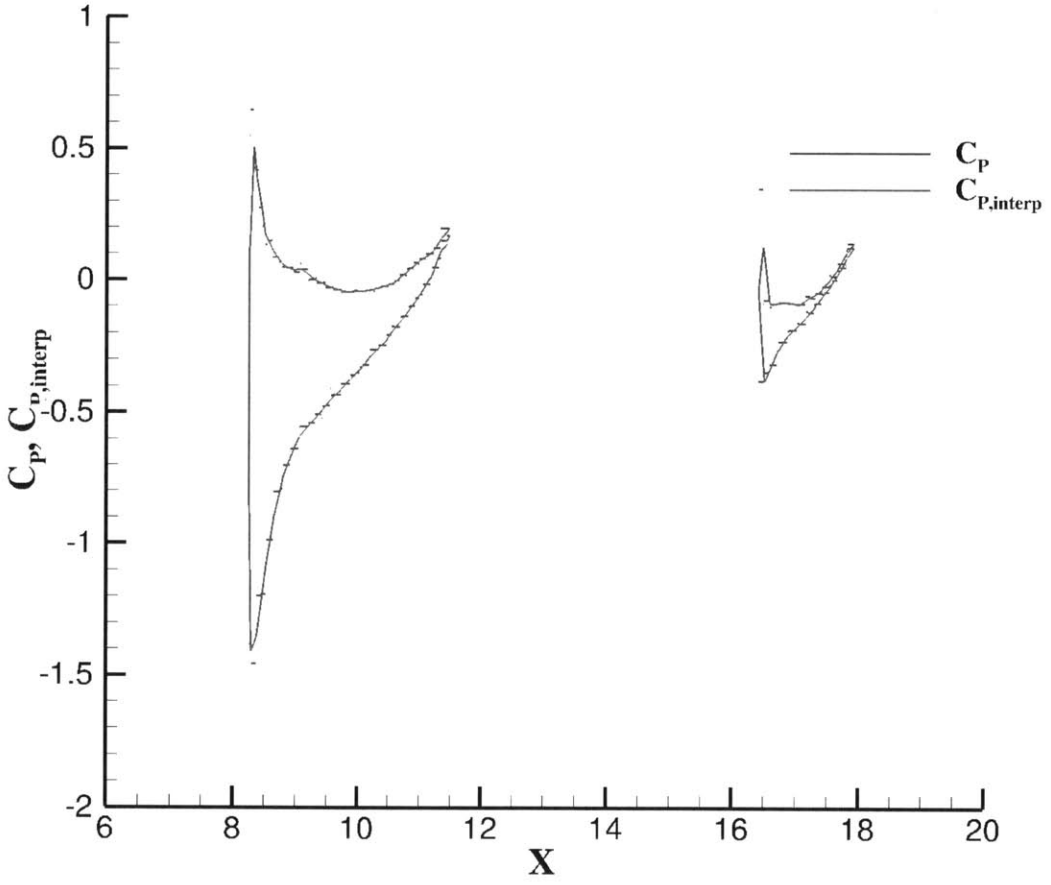

Figure 7-15: Comparison of the interpolated and non-interpolated pressure coefficient at a slice plain of $Y = 2.5$. The distribution on the left is the C_P over the wing, and the distribution on the right is the tail C_P.

The maximum interpolated pressure coefficient at the leading edge of the wing at a slice location of $Y = 2.5$ is 0.5, and is 0.2 at the trailing edge. Note that this is lower than the non-interpolated pressure coefficient and varies considerably from the true maximum C_P of 1.0. This behavior is observed in commercially available

panel codes as well [19]. Note that the interpolated pressure is less accurate than the non-interpolated pressure. Therefore, integrated quantities (forces, moments) should always be generated from the raw non-interpolated C_P. The error in pressure is due to the low-order surface discretization and relatively coarse leading edge mesh resolution. A more accurate leading edge pressure could obviously obtained by refining the mesh near the leading edge.

Currently the process from geometric representation via a .stp file to solution is fully automated, but the meshing step is not highly refined. Additional work is needed in order to develop an automated method for generating accurate (perhaps anisotropic or adapted) meshes from geometry files.

Chapter 8

Conclusions and Future Work

A fully unstructured, high-order potential flow solver based on the Boundary Element Method has been developed and validated. The solver has the following capabilities:

1. Arbitrarily high-order spatial accuracy due to a new method of integrating Green's function for Laplace's equation over high-order curved elements.

2. An automated method for both steady-state and unsteady wake treatment due to using vortons in the wake and casting the wake terms and BEM into a fully coupled nonliner system.

3. A Fast Multipole Method that scales as $O(N)$ for accelerating far-field potential interactions.

The solver is wrapped into a single, stand-alone code written in C++. The entire solution process is automated and scripted, from geometry definition via an .igs or .stp file all the way to obtaining pressure distributions and forces.

There are still many areas for improvement and continuing work, some of which include:

1. The Development of a hybrid doublet-vorton wake model. The current work uses vortons in the wake which act as a velocity influence on the surface. However, when the vortons approach surfaces, the velocity influence becomes nearly

singular and is not capable of being accurately represented by the test functions. A new hybrid method is proposed where wake-surface interactions are computed as potential influences (which are less singular than velocity influences) due to point doublets. The wake velocities would be computed using the current vorton model.

2. Coupling of the current inviscid code to a viscous boundary layer solver such as the one developed by Drela in [10]. This would enable the computation of viscous drag and separation effects.

3. The development of a more robust method of going from a parametric geometry file to an accurate surface mesh. It may be worthwhile to investigate adjoint-based grid adaptation using anisotropic meshes or a more robust high-order mesh generator for this purpose.

4. The implementation of a method for dealing with panelled wake-body intersections via discontinuous basis functions.

Ultimately, the goal is to produce a solver capable of accurately predicting both steady and unsteady inviscid and viscous forces while only requiring specification of a surface geometry and freestream conditions.

Appendix A

Steady-State Jacobian Derivation

In order to compute the first term of the Jacobian, the surface potential residual in Equation 3.12 is differentiated with respect to μ:

$$\frac{\partial \Phi}{\partial \mu} = \left\langle \frac{\partial \phi_K}{\partial \mu}, w \right\rangle_{\mathcal{T}_h} \tag{A.1}$$

where

$$\frac{\partial \phi_K}{\partial \mu} = \int_{S \cup S_W} \left(G_d(v_h, r) + G_s(\frac{\partial \sigma_s(\mathbf{r})}{\partial \mu}, r) \right) \tag{A.2}$$

and $v_h \in W_h$ is the basis-function representation of μ and $\dfrac{\partial \sigma(\mathbf{r})}{\partial \mu}$ is evaluated as:

$$\frac{\partial \sigma_s(\mathbf{r})}{\partial \mu} = \left(\frac{\partial \mathbf{V}_\Psi(\mathbf{r})}{\partial \boldsymbol{\omega}} \frac{\partial \boldsymbol{\omega}}{\partial \mu} \right) \cdot \hat{n} \tag{A.3}$$

The $\dfrac{\partial \boldsymbol{\omega}}{\partial \mu}$ term above is computed from the conversion of wake doublet sheets used to enforce the Kutta condition into volume vorticity. The second and third terms of the Jacobian are similarly obtained by differentiating the potential with respect to the vorton positions $\mathbf{x}$ and vorticity $\boldsymbol{\omega}$, respectively:

$$\frac{\partial \Phi}{\partial \mathbf{x}, \omega} = \left\langle \frac{\partial \phi_K}{\partial \mathbf{x}, \omega}, w \right\rangle_{\mathcal{T}_h} \tag{A.4}$$

where

$$\frac{\partial \phi_K}{\partial \mathbf{x}, \omega} = \int_{S \cup S_W} \frac{\partial G_s(\sigma_s(\mathbf{r}), r)}{\partial \sigma_s(\mathbf{r})} \cdot \frac{\partial \sigma_s(\mathbf{r})}{\partial \mathbf{x}, \omega} \tag{A.5}$$

and

$$\frac{\partial \sigma_s(\mathbf{r})}{\partial \mathbf{x}, \omega} = \frac{\partial \mathbf{V}_\Psi(\mathbf{r})}{\partial \mathbf{x}, \omega} \cdot \hat{n}_s \tag{A.6}$$

Next, the sensitivity of the vorton position residual $\mathbf{X}$ with respect to the doublet strength μ, the vorton positions, and the wake vorticity $\boldsymbol{\omega}$ must be obtained. The derivative of the vorton positions with respect to the doublet strength is obtained by differentiating Equation 5.4:

$$\frac{\partial \mathbf{X}_j}{\partial \mu} = -\frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mu}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.7}$$

where

$$\frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mu} = \frac{\partial \nabla \phi(x_{j-N_s})}{\partial \mu} + \frac{\partial V_\Psi(\mathbf{r})}{\partial \boldsymbol{\omega}} \frac{\partial \boldsymbol{\omega}}{\partial \mu} \tag{A.8}$$

The first term is computed as detailed in Equation A.2 and the $\frac{\partial \boldsymbol{\omega}}{\partial \mu}$ term is computed from the wake conversion as mentioned above. The sensitivity of the vorton positions with respect to themselves is obtained by differentiating Equation 5.4 with respect to $\mathbf{x_j}$.

$$\frac{\partial \mathbf{X}_j}{\partial \mathbf{x}_j} = \frac{1}{\Delta t}\mathbf{I} - \frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mathbf{x}_j}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.9}$$

where $\mathbf{I}$ is the identity matrix and

$$\frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mathbf{x}_j} = \frac{\partial \nabla \phi(\mathbf{x}_{j-N_s})}{\partial \mathbf{x}_j} + \frac{\partial \mathbf{V}_\Psi(\mathbf{r})}{\partial \mathbf{x}_j} \tag{A.10}$$

Note that the first term is a tensor derivative and is computed in a similar manner to Equation A.5 and the second was already computed in Equation A.3. The derivative of the vorton positions with respect to vorticity is:

$$\frac{\partial \mathbf{X}_j}{\partial \boldsymbol{\omega}} = -\frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \boldsymbol{\omega}}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.11}$$

where

$$\frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \boldsymbol{\omega}} = \frac{\partial \nabla \phi(x_{j-N_s})}{\partial \boldsymbol{\omega}} + \frac{\partial \mathbf{V}_\Psi(\mathbf{r})}{\partial \omega} \tag{A.12}$$

and the first term again requires a tensor derivative and the second term is computed as in A.6

The sensitivity of the vorticity residual with respect to the doublet strength μ, the vorton positions $\mathbf{x}$, and the vorton strengths $\boldsymbol{\omega}$ is obtained almost identically to the sensitivities of the wake positions with respect to the same variables. The derivative of vorticity with respect to the double strength is:

$$\frac{\partial \mathbf{W}_j}{\partial \mu} = \left(\frac{\partial \boldsymbol{\omega}}{\partial \mu} \cdot \nabla \right) \mathbf{U}(\mathbf{x}_{j-N_s}) + (\boldsymbol{\omega} \cdot \nabla) \frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mu}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.13}$$

where all of the terms above have been described previously. The derivative of vorticity with respect to vorton position is simply

$$\frac{\partial \boldsymbol{\omega}}{\partial \mathbf{x}_j} = (\boldsymbol{\omega} \cdot \nabla) \frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \mathbf{x}_j}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.14}$$

The final component of the Jacobian is the sensitivity of the vorticity residual with respect to vorticity and is computed as:

$$\frac{\partial \mathbf{W}_j}{\partial \boldsymbol{\omega}_j} = \frac{1}{\Delta t} \mathbf{I} - (\boldsymbol{\omega} \cdot \nabla) \frac{\partial \mathbf{U}(\mathbf{x}_{j-N_s})}{\partial \boldsymbol{\omega}_j}, \quad j \in \{N_s + 1, \ldots, N_V\} \tag{A.15}$$

Appendix B

Multipole Operators

The equations presented below closely follow those developed by Greengard [12] for use in particle simulations. In three dimensions, multipole expansions are expressed in terms of spherical harmonics. The spherical harmonics have a degree n and order m and are defined by the following formula[12]:

$$Y_n^m = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|} cos(\theta) e^{im\phi} \tag{B.1}$$

where P_n^m are associated Legendre functions. A truncated multipole expansion of order p about a box origin representing a collection of K sources of strength q_s located at spherical coordinates $(\rho_1, \alpha_1, \beta_1)...(\rho_K, \alpha_K, \beta_K)$ with respect to the box centroid can be written as:

$$\Phi(r,\theta,\phi) = \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} Y_n^m(\theta,\phi), \quad M_n^m = \sum_{s=1}^{K} \sigma_s \rho_s^n Y_n^{-m}(\alpha_s, \beta_s) \tag{B.2}$$

where (r,θ,ϕ) are the spherical coordinates of the point where the multipole expansion is evaluated. Similarly, the multipole coefficients M_n^m of the multipole expansion for a collection of K dipoles with strength μ_s and orientation $\hat{n}_s$ is

$$M_n^m = \sum_{s=1}^{K} \mu_s \frac{\partial}{\partial n_s} \left(\rho_s^n Y_n^{-m}(\alpha_s, \beta_s) \right) \tag{B.3}$$

Each surface panel is represented by a collection of point sources and dipoles, and their strength and location on each panel is given by the weights and Gauss points of a numerical quadrature rule. Similarly, the vector of multipole expansion coefficients representing a collection of vortons with vorticities $\boldsymbol{\omega}_s$ can be computed as:

$$\mathbf{M}_n^m = \sum_{s=1}^{K} \boldsymbol{\omega}_s \frac{\partial}{\partial n_s} \left(\rho_s^n Y_n^{-m}(\alpha_s, \beta_s) \right) \tag{B.4}$$

In the Multipole to Multipole step of the algorithm, multipole expansions of child boxes must be transferred to their parent box. Let (ρ, α, β) be spherical coordinates of the parent box's centroid with respect to the child box's centroid. Then the multipole coeffecients MT of the parent box are given by the following multipole translation formula

$$MT_j^k = \begin{cases} \sum_{n=0}^{j} \sum_{m=-n}^{n} \frac{M_{j-n}^{k-m} \cdot i^{|k|-|m|-|k-m|} \cdot A_n^m \cdot A_{j-n}^{k-m} \cdot Y_n^{-m}(\alpha,\beta)}{A_j^k} x & |k-m| <= p \\ 0 & |k-m| > p \end{cases} \tag{B.5}$$

where

$$A_n^m = \frac{(-1)^n}{\sqrt{|n-m|! \cdot |n+m|!}} \tag{B.6}$$

The Multipole to Local translation is detailed below. Let (ρ, α, β) be the spherical coordinates of a centroid of a box in a box's interaction list with respect to the centroid of the box itself. Let M_n^m be the coefficients of the multipole expansion of a box in the interaction list of this same box. Then the corresponding local expansion coefficients about the center of the box is:

$$L_j^k = \sum_{n=0}^{p} \sum_{m=-n}^{n} \frac{M_n^m \cdot i^{|k-m|-|k|-|m|} \cdot A_n^m \cdot A_j^k \cdot Y_{j+n}^{m-k}(\alpha, \beta)}{(-1)^n A_{j+n}^{m-k} \cdot \rho^{j+n+1}} \tag{B.7}$$

The Local to Local translation is detailed below. Let (ρ, α, β) be the spherical coordinates of a child box's centroid with respect to its parent's centroid. If L_n^m are the coefficients of the parent box's local expansion, then the corresponding coefficients of

the local expansion about the child box's centroid are:

$$LT_j^k = \sum_{n=j}^{p} \sum_{m=-n}^{n} \frac{L_n^m \cdot A_{n-j}^{m-k} \cdot A_j^k \cdot Y_{n-j} m - k(\alpha, \beta) \cdot \rho^{n-j}}{(-1)^{n+j} \cdot A_n^m} \tag{B.8}$$

The Local Evaluation is detailed below. The potential resulting from the evaluation of the local expansion at spherical coordinates (r, θ, ϕ) with respect to a box centroid is computed as follows:

$$\Phi(r, \theta, \phi) = \sum_{j=0}^{p} \sum_{k=-j}^{j} LT_j^k \cdot Y_j^k(\theta, \phi) \cdot r^j \tag{B.9}$$

where LT_j^k are the coefficients of the local expansion representing the potentials of all boxes which are not nearest neighbors of the box. Note that there may be as many as five sets of LT_j^k coefficients per childless box, corresponding to the contributions from the doublet elements, source elements and three components of the vector potential.

Bibliography

[1] J.F. Remacle A. Johnen and C. Ceuzaine. Geometric validity of curvilinear finite elements. *Journal of Computational Physics*, 2012.

[2] T.R. Quackenbush A.H. Boschitsch, T.B. Burbishley and M.E. Teske. A fast method for potential flows about complex geometries. *34th AIAA Aerospace Sciences Meeting*, 1996.

[3] A.H. Boschitsch and R.J. Epstein. Fast lifting panel method. *14th Computational Fluid Dynamics Conference*, 1999.

[4] R. Brown and A. Line. Efficient high resolution wake modeling using the vorticity transport equation. *AIAA Journal*, 47:1434–1443, 2010.

[5] C. Ceuzaine and F.F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal of Numerical Methods in Engineering*, 79:1309–1331, 2009.

[6] J.E. Kerwin C.Y. Hsin and J.N. Newman. A high order panel method based on b-splines. *Proceedings of the Sixth International COnference on Numerical Ship Hydrodynamics*, pages 133–151, 1993.

[7] J. Peraire D.J. Willis and J.K. White. A combined pfft-multipole tree code, unsteady panel method with vortex particle wake. *International Journal of Numerical Methods in Fluids*, 53:1339–1422, 2007.

[8] M.R. Dudley D.L. Ashby and S.K. Iguchi. Development and validation of an advanced low order panel method. NASA Technical Memorandum 101024, NASA Ames Research Center, Moffett Field, CA, October 1988.

[9] M. Drela. Xfoil: An analysis and design system for low reynolds number airfoils. *Lecture Notes In Engineering*, pages 1–12, 1989.

[10] M. Drela. Three-dimensional integral boundary layer formulation for general configurations. *21st AIAA Computational Fluid Dynamics Conference*, 2013.

[11] E. Nielsen S. Pirzadeh E. Lee-Rausch, D. Hammond and C. Rumsey. Application of the fun3d unstructured-grid navier-stokes solver to the 4th aiaa drag prediction workshop cases. *4th AIAA Drag Prediction Workshop*, 2010.

[12] L. Greengard H. Gheng and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[13] J.L. Hess and A.M.O. Smith. Calculation of potential flow about arbitrary bodies. *Progress in Aeronautical Sciences*, 8, 1967.

[14] Joseph Katz and Allen Plotkin. *Low-Speed Aerodynamics.* Cambridge University Press, 2nd edition, 2001.

[15] A.E. Magnus and .A. Epton. Pan air- a computer program for predicting subsonic or supersonic linear potential flows about arbitrary configurations using a high order panel method. Theory document, nasa cr-3251, 1980.

[16] Hiren D. Maniar. *A Three Dimensional High Order Panel Method Based on B-Splines, PhD Thesis.* PhD thesis, Massachusetts Institute of Technology, Massachusetts, 2006.

[17] L. Morino and C.C. Kuo. Subsonic potential aerodynamics for complex configurations. a general theory. *AIAA Journal*, 12:191–197, 1974.

[18] K. Nabors and J. White. Fastcap: A multipole accelerated 3-d capacitance extraction program. *IEEE Transactions on Computer-Aided Design*, 10(11):1447–1459, 1991.

[19] J. Nathman and M. Matarrese. Hybrid grid (structured and unstructured) calculations with a potential-based panel method. *22nd Applied Aerodynamic Conference and Exhibit*, 2004.

[20] Y. Saad and M.H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing*, 9:856–869, 1986.

[21] C. Sanderson. Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. *Technical Report, NICTA*, 2010.

[22] M. Tomac and D. Eller. From geometry to cfd-based aerodynamic derivatives-an automated approach. *27th Inernational Congress on Aeronautical Sciences*, 2010.

[23] John C. Vassberg. A fast surface-panel method capable of solving million-element problems. *35th AIAA Aerospace Sciences Meeting*, 1997.

[24] D. Willis. Implementing automatic wake-body intersections for full aircraft configurations in dirichlet panel methods. *21st AIAA Computational Fluid Dynamics Conference*, 2013.

[25] David J. Willis. *An unsteady, accelerated, high order panel method with vortex particle wakes.* PhD thesis, Massachusetts Institute of Technology, Massachusetts, 2006.

[26] D.J. Willis. A quadratic basis function, quadratic geometry, high order panel method. *4th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

[27] D.J. Willis. Enriched basis functions for automatically handling wake-body intersections in source-doublet-potential panel methods. *50th AIAA Aerospace Sciences Meeting*, 2012.

[28] G.S. Winckelsmans and A. Leonard. Contributions to vortex particle methods for the computation of three-dimensional incompressible unsteady flows. *Journal of Computational Physics*, 109:247–273, 1993.

[29] J.N. Newman X. Wang and J.K. White. Robust algorithms for boundary element integrals on curved surfaces. *International Conference on Modeling and Simulation of Micro Systems*, 2000.