SANDIA REPORT

SAND2000-2504

Unlimited Release

Printed October 2000

VFLOW2D - A Vortex-Based Code for Computing Flow Over Elastically Supported Tubes and Tube Arrays

Walter P. Wolfe, James H. Strickland, Gregory F. Homicz, and Albert A. Gossler

Prepared by

Sandia National Laboratories

Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States
Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401

Facsimile: (865)576-5728

E-Mail: reports@adonis.osti.gov

Online ordering: <http://www.doe.gov/bridge>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847

Facsimile: (703)605-6900

E-Mail: orders@ntis.fedworld.gov

Online order: <http://www.ntis.gov/ordering.htm>

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SAND2000-2504
Unlimited Release
Printed October 2000

RECEIVED
NOV 02 2000
OSTI

VFLOW2D - A Vortex-Based Code for Computing Flow Over Elastically Supported Tubes and Tube Arrays

Walter P. Wolfe, James H. Strickland, Gregory F. Homicz, and Albert A. Gossler
Thermal/Fluid Computational Engineering Sciences Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0835

Abstract

A numerical flow model is developed to simulate two-dimensional fluid flow past immersed, elastically supported tube arrays. This work is motivated by the objective of predicting forces and motion associated with both deep-water drilling and production risers in the oil industry. This work has other engineering applications including simulation of flow past tubular heat exchangers or submarine-towed sensor arrays and the flow about parachute ribbons. In the present work, a vortex method is used for solving the unsteady flow field. This method demonstrates inherent advantages over more conventional grid-based computational fluid dynamics. The vortex method is non-iterative, does not require artificial viscosity for stability, displays minimal numerical diffusion, can easily treat moving boundaries, and allows a greatly reduced computational domain since vorticity occupies only a small fraction of the fluid volume. A gridless approach is used in the flow sufficiently distant from surfaces. A Lagrangian remap scheme is used near surfaces to calculate diffusion and convection of vorticity. A fast multipole technique is utilized for efficient calculation of velocity from the vorticity field. The ability of the method to correctly predict lift and drag forces on simple stationary geometries over a broad range of Reynolds numbers is presented.

(this page intentionally left blank)

Contents

Preface	9
Nomenclature	11
1 Introduction	15
2 Wall Layer	19
2.1 Overview	19
2.2 Generation of Vorticity	20
2.3 Diffusion of Wall Layer Vorticity	25
2.4 Diffusion of Added Vorticity	26
2.5 Convection of Vorticity	29
2.6 Remapping	30
2.7 Benchmark Studies of Wall Layer Model	33
3 Free Vortex Model	39
3.1 Discrete Representation	39
3.2 Convection of Vorticity	39
3.3 Diffusion of Vorticity	40
3.4 Vortex Merging	41
4 Wall Layer - Outer Flow Coupling	43
4.1 Vortex Blob Formation	43
4.2 Transition Blobs	45
4.3 Mapping Blobs Into the Wall Layer	45
4.4 Free Blobs vs. Free Vortices	46
4.5 Wall Layer Panels	46
5 Fast Multipole Method	47
5.1 Fast Multipole Method for Vortex Blobs	47
5.1.1 Method Overview	47
5.1.2 Benchmark Studies of Fast Solver	50
5.2 Fast Multipole Method Extended to Vortex Panel Sources	56
5.2.1 Generating the Vortex Panel Source Tree	57
5.2.2 Panel-Induced Field	61
5.2.3 Point Vortex Limit	64
5.2.4 Panel-Centered → Arbitrary Coordinates	65
5.2.5 Panel-Centered Origin/Multipole Expansion in Najm, <i>et al.</i> , Basis Functions	68
5.2.6 Panel-Centered Origin/Multipole Expansion in VFLOW2D Basis Functions	69
5.2.7 Arbitrary Origin/Multipole Expansion in VFLOW2D Basis Functions	71
5.2.8 Multipole Expansion vs. Exact Solution	73

5.2.9 Source Expansion Coefficients	74
5.2.10 Field Expansion Coefficients	76
5.2.11 Sample Calculations	81
6 Tube Geometry & Kinematics	85
6.1 Geometry Modules	85
6.2 Prescribed Motion Generator	87
6.3 Example/Benchmark Calculations	88
6.3.1 Circular Cylinder	88
6.3.2 Cylinder with Strakes	89
6.3.3 Rectangular Cylinder	90
6.3.4 Array of Circular Cylinders	90
7 Hydrodynamic Loads	93
7.1 Pressure Distributions	93
7.2 Integrated Loads	95
7.3 Added Mass	96
8 Results	97
8.1 Simulation Parameters	97
8.2 Body Forces	98
8.3 Boundary Layer Separation Points	100
8.4 Wake	100
8.5 Computation Time	100
9 Code Limitations and Unverified Improvements	103
9.1 Current Code Limitations	103
9.1.1 Time Integration	103
9.1.2 Fluid/Structure Coupling	103
9.1.3 Multiple Bodies	103
9.1.4 Non-Cylindrical Body Geometries	104
9.2 Unverified Code Improvements	104
9.2.1 Remapping and Accelerated Computation	104
9.2.2 Turbulence Modeling	106
9.2.3 Transition Criteria	107
9.2.4 Second-Order Time Advance	107
10 Conclusions And Recommendations	109
References	111
Appendix A: Users Manual	115
Appendix B: Example Problem	129
Appendix C: Numerical Recipes	133

FIGURES

Figure 1. Prescribed Surface Injection Points	16
Figure 2. Blanket Surface Flux (from Ref. 36)	16
Figure 3. Wall Layer Model	16
Figure 4. Wall Layer Sketch	19
Figure 5. Surface Vortex Sheet	21
Figure 6. Surface Sheet Splitting to Satisfy Tangential Boundary Conditions	23
Figure 7. Basis Function for Piecewise Linear Distribution	25
Figure 8. Diffusion Velocity Concept	26
Figure 9. Distribution of Wall Vorticity	28
Figure 10. Averaging Volume Used to Calculate the Vorticity Gradient for Cell k	32
Figure 11. Vorticity Distribution for Blasius Solution	34
Figure 12. Tangential Velocity Profile for Blasius Solution	35
Figure 13. Normal Velocity Profile for Blasius Solution	35
Figure 14. Vorticity Distributions For Periodic Solutions of Sinusoidally Oscillating Plate	36
Figure 15. Velocity Distributions For Periodic Solutions of Sinusoidally Oscillating Plate	36
Figure 16. Vorticity Distributions For Transient Solutions of Sinusoidally Oscillating Plate	37
Figure 17. Vorticity Transformation From Wall Layer to Outer Flow	43
Figure 18. Source and Field Point Domains	48
Figure 19. Adaptive Domain Mesh Scheme	49
Figure 20. Benchmark Test Cases	51
Figure 21. Error Plots for Uniform Distribution of 10,201 Vortices	52
Figure 22. CPU Times for Uniform Distribution of 10,201 Vortices	53
Figure 23. Optimum N_S and N_F for U-U Case ($N_{max} = 6$)	54
Figure 24. CPU Times for U-U Case (Direct versus Fast)	55
Figure 25. Errors Using Fast Solver for U-U Case	55
Figure 26. Vortex Panel Straddling Multiple Boxes	58
Figure 27. Panels Protruding from Lists 2 and 3 Source Boxes for $h = 2$	61
Figure 28. Vortex Panel with Linearly Varying Strength	62
Figure 29. Transformation from Panel-Centered to Arbitrary Coordinates	66
Figure 30. Symmetric and Antisymmetric Parts of Linear Strength Variation	69
Figure 31. Relative Error vs. α at $R = 4$ for Antisymmetric Loading	73
Figure 32. Relative Error vs. R at $\alpha = 0^\circ$ for Antisymmetric Loading	74
Figure 33. Shifting Expansion Centers to Center of Childless Box	75
Figure 34. Schematic of a Target Box and a List 4 Source Box	76
Figure 35. Comparison of Predictions for Flow Past a Cylinder Using FMM with $h = 2$ vs. Direct Panel-to-Point Vortex Interactions	81
Figure 36. Comparison of Predictions for Flow Past a Cylinder Using FMM with $h = 1$ vs. Direct Panel-to-Point Vortex Interactions	83
Figure 37. Predefined Tube Cross Sections	85

Figure 38. Example Geometry Construction	86
Figure 39. Rigid Body Motion	88
Figure 40. Streamlines for Starting Flow around Cylinder	89
Figure 41. Edge Velocity for Starting Flow around Cylinder	89
Figure 42. Starting Flow Around Straked Cylinder	90
Figure 43. Starting Flow Around Rectangles @ 22.5° Angle of Attack	90
Figure 44. Starting Flow Around Tube Bundles	91
Figure 45. Two Cylinders Moving Relative to Each Other	91
Figure 46. Surface Pressure Calculation	94
Figure 47. Typical Computed Streamwise Force Coefficient	98
Figure 48. Typical Calculated Lateral Force Coefficient	99
Figure 49. Comparison of Drag Coefficients for Steady Flow Over a Stationary Circular Cylinder.	99
Figure 50. Comparison of Boundary Layer Separation Points, Given as Angular Orientations Measured from the Upstream Stagnation Point.	101
Figure 51. Wake Behind Circular Cylinder in Steady Flow	101
Figure 52. Depiction of a wall layer cell with sub-cell centers at the beginning (left) and at the end (right) of a simulation time step using the alternative remapping scheme.	106

Preface

In fiscal year (FY) 1995 the Department of Energy (DOE) started the Advanced Computing Technology Initiative (ACTI) to transfer computational technology from the National Laboratories to the Nation's oil and gas industry. One of the Sandia National Laboratories (SNL) projects within this initiative was the Advanced Computational Models for Deep Water Oil & Gas Production project, CRADA* NCR 33. The industrial partner for this project was the DeepStar Consortium. Managed by Texaco, Inc., DeepStar is a consortium of oil and gas production and service companies that is engaged in identifying and developing economically viable, low risk methods to produce hydrocarbons from deepwater tracts in the Gulf of Mexico. The U.S. Gulf contains significant oil and gas reserves and remains one of the best domestic oil company development opportunities, but producing hydrocarbons in water this deep presents commercial and technical challenges. The purpose of this project was to develop a computational fluid dynamics (CFD) capability for computing the effects of ocean currents on the vortex-induced vibrations of deepwater risers and umbilicals. The computational technology being developed for this project was also applicable to DOE's Defense Programs mission.

All SNL funding for this project was provided by DOE. DeepStar provided "in kind" funding. DOE's funding for this project has an intermittent history. Although the agreement between DeepStar and SNL that initiated this project was signed in June 1995, funding was not available until April 1996. At the end of FY 1996 the project was suddenly halted. It was restarted at approximately one-third the funding level in February 1997 and was again halted at the end of FY 1997. The project was restarted in December of 1997 and continued until the end of FY 1999 when the project was officially terminated.

The computer resources available to the DeepStar participants at the start of this project necessitated that the code being developed be a two-dimensional (2D) code. The computational power required for full three-dimensional (3D) computations was not yet available. By the time the project was restarted in December 1997, we had started development of a 3D CFD code for DOE's missions. We felt that this code could also be applied for DeepStar's purposes and that computational power had increased to the point where 3D computations were practical. We therefore carried on parallel development of both the 2D and 3D codes in FY 1998 and 1999.

This report presents the results of our development of the 2D CFD code, VFLOW2D. Because of the sudden termination of the project, there are several features in the present code that have not been fully verified and validated. There are other features and improvements that need to be added to the code. Both categories of features are discussed in Section 9.

* Cooperative Research and Development Agreement

(this page intentionally left blank)

Nomenclature

A	=	area
A, B, C, D	=	coefficient matrices; coefficients
a	=	strake length; acceleration
a, b	=	coefficient matrices; point designators; rectangle length and width
C	=	damping coefficient
D, d	=	diameter
F	=	function; complex potential; force
h	=	surface panel length; transformation scale factor
i	=	$\sqrt{-1}$
i, j, k	=	indices
K	=	potential coefficient (see Eq. 104a); stiffness coefficient
L	=	length scale
l	=	length
M	=	moment; mass
m	=	number of equations
m, n	=	indices
N	=	counter
n	=	normal
P	=	complex velocity coefficients
P, p	=	point
p	=	index; pressure
q	=	curve fitting symbol (see Eq. 30 on page 28; also Eq. 56 on page 40)
R	=	distance between vortices; domain radius
Re	=	Reynolds number
r	=	radius
S	=	circumference; reference area; separation angle
s	=	arc length along body surface
t	=	time; strake thickness
U, u	=	velocity
V, v	=	velocity
W	=	complex velocity
w	=	cell width
x, y	=	cartesian coordinates
y	=	distance normal to surface
z	=	complex position coordinate

Subscripts

B, b	=	body
C	=	center
D	=	difference
d	=	diffusion
e	=	end; edge
F	=	field (target)
f	=	fluid
I	=	inner
jnt	=	joint
K	=	known fluid vorticity
n	=	normal
O	=	outer
p	=	pressure
r	=	remap
S	=	source; sum; separation; Smagoranski
s	=	stability
s, t	=	tangential
T	=	target
tr	=	transition
V	=	vortices
VP	=	vortex panels
w	=	wall
0	=	reference value
∞	=	freestream conditions

Superscripts

$+$	=	nondimensional value; outer
$-$	=	inner
$-$	=	average value
$\wedge$	=	unit vector

Greek

α	=	limiting coefficient (see Eq. 49 on page 32); angle
Γ	=	circulation (see Eq. 1 on page 19)
δ	=	Kronecker delta
γ	=	vortex sheet strength, circulation per unit length

ε	=	error
η	=	tangential coordinate
Θ	=	angle
μ	=	fluid viscosity
ν	=	fluid kinematic viscosity
ξ	=	tangential coordinate
ρ	=	nondimensional radius from vortex center (see Eq. 58 on page 40); fluid density
σ	=	core radius
τ	=	convolution parameter (see Eq. 28 on page 27)
ϕ	=	basis function
χ	=	degree of freedom matrix
Ψ	=	stream function
Ω	=	body angular velocity; domain
ω	=	vorticity (see Eq. 2 on page 19)

Subscripts

γ	=	vortex sheet
----------	---	--------------

(this page intentionally left blank)

VFLOW2D - A Vortex-Based Code for Computing Flow Over Elastically Supported Tubes and Tube Arrays

1 Introduction

In the present work we consider the two-dimensional fluid flow over tube arrays that are elastically supported. The motivation for this work arises from the need to calculate the forces on and associated motion of deepwater drilling and production risers. We note that there are several other engineering problems which require simulation of such flows. These include tubular heat exchangers, submarine-towed sensor arrays, and flow around parachute ribbons. The riser tubes used by the oil industry are in general very slender, having aspect ratios (length/diameter) on the order of several thousand. The Reynolds number range is large and in some cases may exceed several million. The three-dimensional flow over such slender bodies may be modeled to reasonable accuracy by assuming that the flow consists of a uniform flow parallel to the tube spans and a two-dimensional viscous flow in the cross-sectional planes. In the past, such flows have been simulated by the oil industry using semi-empirical approaches since conventional grid-based CFD simulation tools do not presently possess the required computational efficiency and robustness, especially at large Reynolds numbers. Our challenge is to be able to make robust calculations for a variety of tube array geometries over a large Reynolds number range in a computationally efficient manner.

We have had good success using gridless vortex methods to model unsteady, high Reynolds number flows over various engineering devices. These simulations have included the prediction of performance of and fluid forces on vertical-axis wind turbines [41, 42]; the calculation of wake development behind aircraft [12] and submarines [40]; and the prediction of flow over simple axisymmetric parachute shapes [37]. In contrast to conventional grid-based computational fluid dynamics methods, the gridless vortex method is noniterative, artificial viscosity is not required for stability, numerical diffusion is minimal, and moving boundaries can be easily treated. A salient feature of the method is that the computational domain only includes those regions where there is vorticity. For higher Reynolds numbers this domain is increasingly reduced, since vorticity occupies a smaller and smaller fraction of the fluid volume.

The vortex method is essentially a method in which one tracks the evolution of the vorticity field, including its generation, diffusion, and convection. For constant density fluids, vorticity can only be generated at the walls of immersed bodies. Vorticity is generated in just the right amount so as to satisfy the velocity boundary conditions at the wall, *i.e.*, the fluid velocity and

the velocity of the wall must be equal. As the vorticity is generated at the wall, it diffuses into the fluid under the action of viscosity and is then further convected by the local fluid velocity. The velocity field at any instant in time can be entirely constructed from the vorticity field.

Methods for transporting the vorticity from the surfaces of bodies during the computation can be grouped into a few general categories. The first is the injection of vorticity at prescribed surface injection points (Fig. 1). Examples of these methods are cited by Sarpkaya [32]. While these methods are extremely efficient since they minimize the number of vortices injected into the flowfield, the user must guess *a priori* the flow separation points, and the method cannot account for secondary flow separation.

Figure 1. Prescribed Surface Injection Points

A second category can be called blanket surface flux. In these methods large numbers of very small discrete vortex elements (also known as vortex blobs) are placed next to the surface at each time step (Fig. 2). Examples include the work of Koumoutsakos [18], Lin *et al.* [22], and Shih *et al.* [36]. While these methods can accurately compute flow separation points, they also have some disadvantages. One is that a large number of vortex elements must be used. The vortex diameters must be some fraction of the boundary layer thickness, and for high Reynolds number flows, the required number of vortex blobs becomes prohibitive. Another is that unless the element density close to the surface is very high, the details of the boundary layer are not accurately modeled.

Figure 2. Blanket Surface Flux (from Ref. 36)

A third approach being developed by Marshall and Grant [23] alleviates some of these problems by using anisotropic blobs that have thicknesses that are a fraction of the boundary layer thickness and lengths that are on the order of the surface panel length.

Figure 3. Wall Layer Model

Our proposed approach uses a wall-layer model (Fig. 3) that yields high resolution of the vorticity/velocity field near the wall and is fully compatible with discrete vortex methods. In this approach, we use infinitely thin vortex sheets placed on the walls of the bodies at the beginning of a particular time step in order to satisfy the velocity boundary conditions. We then allow this sheet to diffuse into the flow along with any other vorticity already in the flow. Near the walls we use a Lagrangian-Eulerian scheme to

calculate the diffusion and convection of vorticity. This requires the construction of a thin gridded region near the body. This Lagrangian grid is moved from its original position according to the convective and diffusive velocities at its node points. At the end of each time step, the vorticity is remapped back onto the original grid. The use of this gridded wall region is advantageous in resolving the flow near the wall without producing a large number of free vortices. As the vorticity moves across the outer boundary of the wall layer it is converted into discrete circular blobs with diameters on the order of the surface panel length, h . These discrete vortex blobs (vortices) retain their initial circulation and are convected at the local fluid velocity. Within the wall layer we have developed an efficient algorithm based on continuity, the definition of vorticity, and the wall velocity to calculate the local velocity from the vorticity between the wall and nodes of interest. In the regions away from the wall, a fast multipole technique that was developed as part of this work is utilized to allow for the efficient calculation of velocity from the vorticity field. A different implementation of the wall layer concept has also been used by Chou [8]. The present method is intended to yield good resolution of the diffusion and convection processes close to the body without generating any free vortex blobs that are smaller than $O(h)$.

The remainder of this report presents the technical details of the VFLOW2D code. Chapter 2 presents a discussion of the wall layer, including vorticity generation, diffusion, convection, and remapping. Some results of benchmark calculations for the wall layer are presented. Chapter 3 presents the details of the free vortex model, including convection, diffusion, and merging. Chapter 4 discusses the coupling between the wall layer and the outer flow region that allows vorticity to smoothly transition between the wall layer and the wake. Chapter 5 discusses the Fast Multipole Method that greatly increases the efficiency of discrete vortex calculations and makes their use tractable. Fast Multipole Methods for both free vortices and for vortex surface panels are presented. Chapter 6 discusses the tube geometry and motion options that are currently in VFLOW2D. The calculation of hydrodynamic loads is discussed in Chapter 7. Chapter 8 presents the results to date. Code limitations and recent additions to the code that have not been fully verified are discussed in Chapter 9. A summary of the work accomplished to date and recommendations for future improvements are discussed in Chapter 10. A users' manual describing the data required to run the code is presented in Appendix A. Appendix B presents an example problem output. Appendix C describes some Numerical Recipes that are used by VFLOW2D and how to acquire them.

(this page intentionally left blank)

2 Wall Layer

2.1 Overview

Vorticity, ω , is defined as the curl of the velocity field,

$$\vec{\omega} \equiv \nabla \times \vec{V}. \quad (1)$$

For two-dimensional flow, ω has only one component normal to the velocity field and can be considered a scalar value. The integral of ω over any given area, A , is the circulation, Γ , within that area,

$$\Gamma \equiv \int_A \omega \, dA. \quad (2)$$

The wall layer is a thin meshed region adjacent to the body surface within which vorticity is modeled as a continuum field using quadrilateral elements (mesh cells) that are elongated in the tangential direction as compared with their thickness normal to the wall (Fig. 4). It is composed of three distinct sublayers, $0 < y < h$, $h < y < 2h$, and $2h < y < 3h$, where y is the distance normal to the wall and h is the surface panel length. Within the first sublayer, $0 < y < h$, all vorticity is contained within the mesh cells. The second sublayer, $h < y < 2h$, is a transition region where vorticity can exist either within the mesh

cells or as discrete vortex elements. When vorticity has been convected and diffused outward from the wall into the outer layer of cells of sublayer 2, it is converted to a discrete vortex blob. This conversion conserves the total circulation and the first and second moments of vorticity. The third sublayer, $2h < y < 3h$, is an overflow region designed to catch any vorticity that "leaks" out of the top of the wall layer during a time step before a discrete vortex can be constructed. Any vorticity that enters sublayer 3 is immediately converted into a vortex blob.

The mesh within the wall layer is defined using orthogonal, curvilinear coordinates. This allows one to march outward from the wall and obtain the velocity field from the local vortic-

Figure 4. Wall Layer Sketch

ity field without having to explicitly consider the vorticity in the wake. Diffusion and convection of vorticity are accomplished using a Lagrangian-Eulerian scheme. We chose this scheme over an Eulerian scheme because it is second order and conservative and yields less numerical diffusion. At each time step the grid moves with the vorticity during diffusion and convection. The solution is then mapped back onto the original orthogonal grid. Circulation is explicitly conserved during each step in this process. After the remap step, vorticity is generated at the wall to satisfy the prescribed boundary conditions.

The linear nature of vorticity allows us to split the wall-layer vorticity into its positive and negative components. Each component is diffused and remapped independently. Both components are convected with the total convection velocity. This splitting greatly improves the numerical stability of the code when computing problems that have both signs of vorticity. The equations for vorticity generation, diffusion, convection, and remapping are developed in the following sections.

2.2 Generation of Vorticity

Vorticity is generated at the wall of each tube in order to satisfy the velocity boundary conditions. There are several ways to obtain formulations aimed at satisfying these boundary conditions. However, some methods will lead to unsatisfactory results. Kempka *et al.* [16] provide a discussion of some of the problems associated with satisfying velocity boundary conditions and present a formulation in which the “no slip” boundary condition is satisfied explicitly and the normal boundary condition is satisfied implicitly.

In the present work, we use a method that is compatible with the thin sheets used to construct the tubular geometries. In this method, the normal velocity boundary condition is satisfied explicitly by placing a vortex sheet on the thin sheet representing the body surface. The “no slip” boundary condition is used to split the calculated vortex sheet into two vortex sheets which lie on either side of the body surface sheet. This method is somewhat similar to one outlined by Strickland [37] in which a stream-function formulation was used to satisfy velocity boundary conditions for an axisymmetric flow over thin shells.

In the present method, the vortex sheet is discretized into n_{jnt} values of the sheet strength, $\gamma(s)$, that are coincident with the location of joints on the body surface as indicated in Fig. 5. For each panel on the surface, an equation can be written that specifies that the fluid velocity normal to the surface is equal to the normal velocity of the surface. The normal fluid velocity at the surface, U_{fn} , may be split into that due to the unknown vortex sheet, U_{fy} , and that due to all other known vorticity in the flow field, U_{fK} . Denoting the normal body velocity by U_{Bn} we can write

$$U_{Bn} = U_{fn} = U_{fK} + U_{fy} . \quad (3)$$

The only unknown in Eq. (3) is U_{fy} which is a function of the vortex sheet strength, $\gamma(s)$. In order to discretize this equation, we represent the vortex sheet by a number of discrete values,

Figure 5. Surface Vortex Sheet

γ_j , with linear variations between adjacent panels. As indicated in Fig. 5, the source of vorticity associated with γ_j varies from zero at joint $j-1$ to γ_j at joint j and back to zero at joint $j+1$ except when $j=1$ or $j=n_{jnt}$. We satisfy Eq. (3) in an integral sense using a Galerkin method on each body panel which yields a much smoother and well-behaved solution as compared to point collocation. The integrated normal fluid velocity due to the surface vortex sheet on panel i may be written as

$$\int_{s_i}^{s_{i+1}} U_{fV} ds = [A_{ij}] \{\gamma_j\} \quad , \quad (4)$$

where $[A_{ij}]$ is obtained from the Biot-Savart law. $[A_{ij}]$ represents the integral over the i^{th} panel of the normal velocity due to a unit value for the strength γ_j . From Eqs. (3) and (4) we may write

$$[A_{ij}] \{\gamma_j\} = \int_{s_i}^{s_{i+1}} (U_{Bn} - U_{fK}) ds \quad . \quad (5)$$

If we let this sheet represent segment m of our body, then we can modify the notation slightly in Eq. (5) to reflect that fact,

$$[A_{ij}]_m \{\gamma_j\}_m = \left\{ \int_{s_i}^{s_{i+1}} U_{Bn} - U_{fK} ds \right\}_m \quad . \quad (6)$$

An additional equation that must be satisfied is the Kelvin-Helmholtz Theorem [9] that requires that the total circulation produced by the body during a given time step remains equal to zero so as to insure that there is no circulation around the boundary of the infinite domain:

$$[B_j]_m \{\gamma_j\}_m = 0. \quad (7)$$

Here B_j is the amount of circulation in the vortex sheet on panel j and $j-1$ due to a unit value of γ_j . This may be expressed by

$$B_j = \begin{cases} \frac{s_1}{2} & \text{for } j=1 \\ \frac{s_{j+1} + s_{j-1}}{2} & \text{for } j \neq 1 \text{ \& } j \neq n_{jnt} \\ \frac{s_{n_{jnt}}}{2} & \text{for } j=n_{jnt} \end{cases} \quad (8)$$

In order to connect the various end points of the segments together, we can define a connectivity matrix by

$$[C_{kj}] \{\gamma_j\}_m = \{0\}. \quad (9)$$

Defining D_i by

$$D_i \equiv \int_{s_i}^{s_{i+1}} (U_{Bn} - U_{fK}) ds, \quad (10)$$

we can write a set of linear equations for γ_j as

$$\begin{bmatrix} [A_{ij}]_m \\ [B_j]_m \\ [C_{kj}] \end{bmatrix} \{\gamma_j\}_m = \begin{Bmatrix} \{D_i\}_m \\ 0 \\ \{0\} \end{Bmatrix} \quad (11)$$

If we let the total number of unconnected joints in the body be equal to N_{jnt} , we note from Eq. (11) that there are $N_{jnt} - m$ equations of the first form since there are that many panels. There is one equation of the second form. If the body is not closed, there are $m - 1$ equations of the third form. Thus, if the body is not closed then there are N_{jnt} unknown values of $\{\gamma_j\}_m$ and N_{jnt} equations. If, on the other hand, the body is closed, there are m equations of the third form which produces $N_{jnt} + 1$ equations and only N_{jnt} unknowns. The reason for this behavior is a study in itself and is beyond the scope of the present report. In order to treat the

multiply-connected closed body, we will use a least squares solution for the over determined set of equations. If we cast Eq. (11) in the form

$$[a]\{\gamma\} = \{b\}, \quad (12)$$

then

$$[a]^T[a]\{\gamma\} = [a]^T\{b\} \quad (13)$$

forms the least squares solution. As we will demonstrate later, this method is very robust and self consistent and provides excellent agreement with analytical results.

In order to satisfy the tangential boundary conditions, the surface sheet is split into an inner and outer sheet. In Fig. 6a, the surface sheet is depicted as a thin sheet with thickness ϵ . The body surface lies within this sheet and has a sheet whose strength is γ_O above it and a sheet whose strength is γ_I below it. In order to obtain the outer sheet strength, we could simply determine the value of U_O and use the relationship

$$\gamma_O = u_{bt} - U_O, \quad (14)$$

Figure 6. Surface Sheet Splitting to Satisfy Tangential Boundary Conditions

where u_{bt} is the tangential velocity of the body surface. Likewise the inner sheet strength could be determined by

$$\gamma_I = U_I - u_{bt}. \quad (15)$$

Calculation of the velocity u_{ft} , the mid-sheet tangential fluid velocity, shown in Fig. 6b is somewhat easier than calculating U_O or U_I since one can simply ignore the local sheet strength γ on the panel. In general $u_{bt} \neq u_{ft}$. The outer sheet strength γ_O can be given in terms of the mid-sheet tangential fluid velocity, u_{ft} , and the body surface velocity, u_{bt} , by the expression

$$\gamma_O = \frac{\gamma}{2} + u_{bt} - u_{ft}. \quad (16)$$

Likewise the inner sheet strength can be expressed by

$$\gamma_I = \frac{\gamma}{2} - u_{bt} + u_{ft}. \quad (17)$$

One could use Eqs (16) and (17) in conjunction with a point collocation scheme to construct piece-wise linear distributions for the two sheet strengths. However, a much more accurate scheme can be obtained by integrating over the support of the basis function for a piece-wise linear distribution. As indicated in Fig. 7, the basis function ϕ for element j (the shaded triangle) is given by

$$\phi_j = \begin{cases} 1 + \xi & \text{for interval between } j-1 \text{ and } j \\ 1 - \xi & \text{for interval between } j \text{ and } j+1 \end{cases}, \quad (18)$$

where $\xi_{j-1} = -1$, $\xi_j = 0$, and $\xi_{j+1} = 1$.

Integrating Eq. (16) over the basis function between $j-1$ and $j+1$ we obtain

$$\gamma_{Oj} = \frac{\gamma_j}{2} + \left(\frac{2}{s_{j+1} - s_{j-1}} \right) \left\{ (s_j - s_{j-1}) \int_{-1}^0 (1 + \xi)(u_{bt} - u_{ft}) d\xi \right. \\ \left. + (s_{j+1} - s_j) \int_0^1 (1 - \xi)(u_{bt} - u_{ft}) d\xi \right\} \quad (19)$$

Figure 7. Basis Function for Piecewise Linear Distribution

and

$$\gamma_{Ij} = \gamma_j - \gamma_{Oj}. \quad (20)$$

2.3 Diffusion of Wall Layer Vorticity

Diffusion of vorticity can be accomplished by one of several means. The diffusion method we use is based on the diffusion velocity concept of Kempka and Strickland [17]. In order to illustrate this diffusion velocity concept, consider the two-dimensional viscous flow field depicted in Fig. 8. The diffusion velocity is specified such that the circulation within a given boundary remains constant with time if that boundary moves at the diffusion velocity plus the local fluid velocity.

From Ref. [17], the diffusion velocity, $\vec{u}_d$, is given by

$$\vec{u}_d \times \vec{\omega} = -\nu \nabla \times \vec{\omega}. \quad (21)$$

For two-dimensional flow, this reduces to

$$\vec{u}_d = -\frac{\nu}{\omega} \nabla \omega. \quad (22)$$

In orthogonal, curvilinear coordinates, this equation becomes

$$\vec{u}_d = -\frac{\nu}{\omega} \left\{ \frac{1}{h_\xi} \frac{\partial \omega}{\partial \xi} \hat{\xi} + \frac{1}{h_\eta} \frac{\partial \omega}{\partial \eta} \hat{\eta} \right\}, \quad (23)$$

Figure 8. Diffusion Velocity Concept

where $\hat{\xi}$ and $\hat{\eta}$ are the unit vectors while h_{ξ} and h_{η} are the transformation scale factors defined as $h_{\xi} = |\partial \vec{r} / \partial \xi|$ and $h_{\eta} = |\partial \vec{r} / \partial \eta|$; $\vec{r}$ is the position vector.

2.4 Diffusion of Added Vorticity

Section 2.2 describes the method for computing the amount of vorticity that must be added during each time step to satisfy the prescribed boundary conditions. In this section we describe how this vorticity is diffused during that time step.

The diffusion velocity at any point is given by Eq. (23). Assuming that the diffusion at the wall is predominately one-dimensional and switching to cartesian coordinates for notational simplicity, Eq. (23) becomes

$$u_d|_w = -\frac{v \partial \omega}{\omega \partial y}|_w, \quad (24)$$

where y is measured normal to the wall. The flux of the circulation γ per unit length of x across the wall during a computation time step is simply the product of the diffusion velocity at the wall and the added vorticity, or

$$\left. \frac{\partial \gamma}{\partial t} \right|_w = -v \left. \frac{\partial \omega}{\partial y} \right|_w, \quad (25)$$

which means that the vorticity gradient at the wall is

$$\left. \frac{\partial \omega}{\partial y} \right|_w = -\frac{1}{v} \left. \frac{\partial \gamma}{\partial t} \right|_w. \quad (26)$$

We have examined several numerical models for diffusing the vorticity into the flow at the wall. In a paper by Koumoutsakos, Leonard, and Pepin [19] concerning vorticity generation at a wall, an algorithm for updating the particle strengths in a Particle Strength Exchange (PSE) method was presented for finite panels with curvature. In the context of the present one-dimensional model, the vorticity that would be introduced into the domain from the wall during a time Δt would have a distribution equal to

$$\Delta \omega = \omega^{t+\Delta t} - \omega^t = \frac{\gamma e^{-\frac{y^2}{4v\Delta t}}}{\sqrt{\pi v \Delta t}}, \quad (27)$$

where $\omega^{t+\Delta t}$ is the vorticity at the time $t + \Delta t$ and ω^t is the vorticity at time t . We note that Eq. (27) is identical to the exact solution for an impulsively started flat plate [10]. Adding all of the circulation to the first element at the beginning of the time step is roughly equivalent to using Eq. (27) at the end of the time step.

Since the diffusion equation is linear, we can obtain the vorticity distribution associated with the flux from the wall at the end of time Δt by superimposing the impulsive solutions via the following convolution integral,

$$\Delta \omega = \int_0^{\Delta t} \frac{\partial \gamma}{\partial \tau} \frac{\exp\left(-\frac{y^2}{4v(\Delta t - \tau)}\right)}{\sqrt{\pi v(\Delta t - \tau)}} d\tau. \quad (28)$$

Assuming that $\partial \gamma / \partial \tau$ is constant over the time interval Δt and defining $y^+ \equiv y / \sqrt{v \Delta t}$, Eq. (28) can be integrated numerically to obtain the curve labeled as "Exact" in Fig. 9. The resulting distribution can be fitted with a curve given by

$$\Delta \omega^+ \equiv \frac{\Delta \omega \sqrt{v \Delta t}}{\frac{\partial \gamma}{\partial \tau}} = 1.131 e^{-q}, \quad (29)$$

Figure 9. Distribution of Wall Vorticity

where

$$q = 0.884y^+ + 0.161y^{+2}. \quad (30)$$

This equation is plotted with the "Fit" symbols in Fig. 9. For comparison, the distribution given by a single impulse at the mid time is also plotted in Fig. 9, identified in the legend by $0.5\Delta t$. The major difference in the two curves is their distributions near the wall. From Eq. (26) it is seen that the slope of the $(\Delta\omega^+, y^+)$ curve at the wall must be equal to minus one, not zero as is given by the impulsive start representation. We note that the proper slope is obtained when using Eq. (29).

During a given time step, Δt , the quantity actually added to the wall layer is the amount of circulation per unit length, $\Delta\gamma^+$. The equation for $\Delta\gamma^+$ can be found by numerically integrating Eq. (29). The result is

$$\Delta\gamma^+ = 1 - e^{-p}, \quad (31)$$

where

$$p = 1.114y^+ + 0.160y^{+2}. \quad (32)$$

2.5 Convection of Vorticity

The equations used to compute the convection velocity within the wall layer are derived from the definition of vorticity in terms of the curl of the velocity field,

$$\vec{\omega} = \nabla \times \vec{V}, \quad (33)$$

and the continuity equation

$$\nabla \cdot \vec{V} = 0. \quad (34)$$

In 2-D orthogonal curvilinear coordinates, these equations become

$$\omega = \frac{1}{h_\xi h_\eta} \left\{ \frac{\partial}{\partial \xi} (h_\eta V_\eta) - \frac{\partial}{\partial \eta} (h_\xi V_\xi) \right\} \quad (35)$$

and

$$\frac{\partial}{\partial \xi} (h_\eta V_\xi) + \frac{\partial}{\partial \eta} (h_\xi V_\eta) = 0. \quad (36)$$

From the derivative chain rule, we have

$$\begin{aligned} dV_\xi &= \frac{\partial V_\xi}{\partial \xi} d\xi + \frac{\partial V_\xi}{\partial \eta} d\eta \\ dV_\eta &= \frac{\partial V_\eta}{\partial \xi} d\xi + \frac{\partial V_\eta}{\partial \eta} d\eta \end{aligned} \quad (37)$$

Equations (35) and (36) can be manipulated to yield the partial derivatives required in Eqs. (37), which can then be integrated to give the values of V_ξ and V_η at any point p in the wall layer. The integration is simplified if we restrict the integration to lines of constant ξ . The final equations then become

$$V_\xi = V_{\xi_w} + \int_w^p \left(-\omega h_\eta + \frac{1}{h_\xi} \frac{\partial}{\partial \xi} (h_\eta V_\eta) - \frac{V_\xi}{h_\xi} \frac{\partial h_\xi}{\partial \eta} \right) d\eta \quad (38)$$

and

$$V_\eta = V_{\eta_w} + \int_w^p \left(-\frac{1}{h_\xi} \frac{\partial}{\partial \xi} (h_\eta V_\xi) - \frac{V_\eta}{h_\xi} \frac{\partial h_\xi}{\partial \eta} \right) d\eta. \quad (39)$$

2.6 Remapping

After computing the diffusion and convection velocities at the grid nodes, each node is moved a distance equal to the sum of the local diffusion and convection velocities times the time step Δt . The total circulation within each grid cell is held constant during this movement. The scheme used to map this solution back onto the original orthogonal grid is an implementation of the second-order, conservative, remapping scheme of Dukowicz and Kodis [11]. The reader is referred to Ref. [11] for a complete derivation and discussion of the method. We present only a brief description.

The total circulation within any grid cells is given by

$$\Gamma_k = \iint_{A_k} \omega h_\xi h_\eta dA, \quad (40)$$

where Γ_k is the total circulation within cell k , A_k is the area of cell k , and ω is the vorticity within area dA . Direct implementation of this equation is difficult unless one assumes that the vorticity within each cell is uniformly distributed. However, experience has shown that such a uniform distribution yields a first-order remapping scheme that is overly diffusive and, consequently, inaccurate. A second-order-accurate remapping scheme can be obtained if one assumes that the vorticity within each cell varies linearly over the cell, i.e.,

$$\omega_k = \bar{\omega}_k + \frac{\partial \omega_k}{\partial \xi}(\xi - \bar{\xi}_k) + \frac{\partial \omega_k}{\partial \eta}(\eta - \bar{\eta}_k), \quad (41)$$

where ω_k is the vorticity at point (ξ, η) , $\bar{\omega}_k$ = average vorticity within cell $k = \Gamma_k / (h_\xi h_\eta A_k)$, $\partial \omega_k / \partial \xi$ and $\partial \omega_k / \partial \eta$ are the components of the gradient in cell k , while $\bar{\xi}_k$ and $\bar{\eta}_k$ are the coordinates of the cell's centroid.

With the addition of Eq. (41), the direct implementation of Eq. (40) becomes very difficult. A major simplification can be made by noting that, if a function $\vec{F}$ can be found such that $\omega(\vec{r}) = \nabla \cdot \vec{F}$, then the divergence theorem can be used to write Eq. (40) as

$$\Gamma_k = \iint_{A_k} \nabla \cdot \vec{F} h_\xi h_\eta dA = \oint_{S_k} \hat{n} \cdot \vec{F} h_\xi h_\eta dS. \quad (42)$$

Although $\vec{F}$ is not unique, a practical choice is that $\vec{F} = (0, P)$, giving the equation set

$$\frac{\partial P}{\partial \eta} = \omega(\xi, \eta) \quad (43)$$

$$\Gamma_k = -\oint P h_\xi h_\eta d\xi$$

Substituting Eq. (41) into Eq. (43) and performing the integrations gives

$$\Delta_1^2 \Gamma^k = h_\xi h_\eta (\xi_1 - \xi_2) \left[\frac{a_1^k}{2} (\eta_1 + \eta_2) + \frac{a_2^k}{6} \{ \eta_1 (2\xi_1 + \xi_2) + \eta_2 (\xi_1 + 2\xi_2) \} \right. \\ \left. + \frac{a_3^k}{12} \{ \eta_1 (2\eta_1 + \eta_2) + \eta_2 (\eta_1 + 2\eta_2) \} \right] \quad (44)$$

where $\Delta_1^2 \Gamma^k$ is the contribution of the line segment from point 1 to point 2 to the circulation within cell k , (ξ_1, η_1) , and (ξ_2, η_2) are the coordinates of points 1 and 2, and the a^k coefficients are given by

$$a_1^k = \bar{\omega}_k - \frac{\partial \omega_k}{\partial \xi} \bar{\xi}_k - \frac{\partial \omega_k}{\partial \eta} \bar{\eta}_k$$

$$a_2^k = \frac{\partial \omega_k}{\partial \xi} \quad (45)$$

$$a_3^k = \frac{\partial \omega_k}{\partial \eta}$$

The components of the vorticity gradient in the above equations are approximated by defining a secondary grid whose nodes are the centroids of the primary grid (see Fig. 10). The vorticity gradient is approximated as

$$\langle \nabla_k \omega \rangle = \frac{1}{h_\xi h_\eta A_j} \iint_{A_j} \nabla \omega h_\xi h_\eta dA, \quad (46)$$

where the averaging area, A_j , is the area in the secondary mesh defined by the centroids of all of the neighbors of cell k . The evaluation of the above equation is facilitated by the use of the divergence theorem, giving

$$\langle \nabla_k \omega \rangle = \frac{1}{h_\xi h_\eta A_j} \oint_{S_j} \omega h_\xi h_\eta \hat{n} dS, \quad (47)$$

Figure 10. Averaging Volume Used to Calculate the Vorticity Gradient for Cell k

where S_j is the circumference of area A_j . We assume here that ω varies linearly along the circumference. The gradient components are given by

$$\left\langle \frac{\partial \omega_k}{\partial \xi} \right\rangle = \frac{1}{h_\xi h_\eta A_j S_j} \oint \omega h_\xi h_\eta d\eta \quad (48)$$

$$\left\langle \frac{\partial \omega_k}{\partial \eta} \right\rangle = -\frac{1}{h_\xi h_\eta A_j S_j} \oint \omega h_\xi h_\eta d\xi$$

For stability reasons, the gradient computed by the above method must be limited in certain cases. The limiting method uses a two-dimensional extension of van Leer's one-dimensional method. A limiting coefficient α_k ($0 \leq \alpha_k \leq 1$) for each cell is defined such that

$$\nabla_k \omega = \alpha_k \langle \nabla_k \omega \rangle, \quad (49)$$

and α_k is determined by enforcing local monotonicity such that the vorticity within cell k does not lie outside the range of average vorticities in the neighboring cells. α_k is given by

$$\begin{aligned}
\alpha_k &= \min\{1, \alpha_{\max}, \alpha_{\min}\} \\
\alpha_{\max} &= \max\{0, (\bar{\omega}_{\max} - \bar{\omega}_k)/(\omega_{k, \max} - \bar{\omega}_k)\} \\
\alpha_{\min} &= \min\{0, (\bar{\omega}_{\min} - \bar{\omega}_k)/(\omega_{k, \min} - \bar{\omega}_k)\}
\end{aligned} \tag{50}$$

In practice, these remapping equations are implemented by making two sweeps through the wall layer: one along the grid lines of the diffused/convected mesh and a second along the grid lines of the original, orthogonal mesh. The contributions of each line segment are added to the circulation within the appropriate grid cells of the original, orthogonal mesh.

2.7 Benchmark Studies of Wall Layer Model

In order to verify the wall layer model and implementation, we compare calculated results against some known, closed-form solutions. The first solution examined is the case for flow over a finite length flat plate. In this particular calculation, the size of the wall computational space and Reynolds number were chosen to contain all of the vorticity within the wall region ($y \leq 2h$) and, hence, there was no need for any of the vorticity to be represented by free two-dimensional blobs. Rather, we treat the entire boundary layer as a wall flow in which vorticity is introduced at the wall, diffused, and convected. The plate was discretized into 50 elements with $h = 1$.

Figures 11 through 13 show the predictions from the current code compared with the Blasius solution for a flat plate [33]. These figures show very good agreement for the vorticity distribution and for both velocity components. The data in these figures were computed for a Reynolds number, based on the distance from the plate's leading edge, of 7.5×10^5 . Comparisons at other Reynolds numbers show identical results.

We next examine the case of a sinusoidally oscillating plate, sometimes referred to as Stokes Second Problem. After a few iterations, the exact solution becomes periodic, given by [10]

$$\frac{u}{U_0} = \exp\left(-\sqrt{\frac{n}{2\nu}}y\right) \cos\left(nt - \sqrt{\frac{n}{2\nu}}y\right) \tag{51}$$

and

$$\frac{\omega}{U_0} = \sqrt{\frac{n}{2\nu}} \exp\left(-\sqrt{\frac{n}{2\nu}}y\right) \left[\cos\left(nt - \sqrt{\frac{n}{2\nu}}y\right) - \sin\left(nt - \sqrt{\frac{n}{2\nu}}y\right) \right]. \tag{52}$$

Figures 14 and 15 show the comparisons of vorticity and velocity distributions, respectively, with the exact periodic solutions. Three curves are shown for three different positions in the oscillatory cycle. The position angle, Θ , is given by $\Theta = \text{acos}(nt)$. The numerical results were computed after five complete cycles. The comparison with the exact solutions is very good. Transient results during the period prior to reaching the "steady" periodic solution are

shown in Fig. 16. Wu *et. al.* [46] and Panton [28] address the issue of transient vorticity values for this problem. One method of solving the transient problem is to use the convolution integral given by Eq. (28). This method is valid in the present case since there is no net accumulation of vorticity due to convection at any stream-wise location. The “exact” solutions shown in Fig. 16 are a numerical integration of this convolution integral. The agreement is seen to be very good.

Figure 11. Vorticity Distribution for Blasius Solution

Figure 12. Tangential Velocity Profile for Blasius Solution

Figure 13. Normal Velocity Profile for Blasius Solution

Figure 14. Vorticity Distributions For Periodic Solutions of Sinusoidally Oscillating Plate

Figure 15. Velocity Distributions For Periodic Solutions of Sinusoidally Oscillating Plate

Figure 16. Vorticity Distributions For Transient Solutions of Sinusoidally Oscillating Plate

(this page intentionally left blank)

3 Free Vortex Model

3.1 Discrete Representation

In order to model the continuum vorticity field external to the wall layer, we have chosen to discretize the external vorticity field into a number of discrete vortex blobs. The term *vortex blobs* was first introduced by Chorin [7] and refers to a finite region of vorticity that is assumed to have a circular shape and a radially symmetrical distribution of vorticity. The entire blob is assumed to convect at the velocity of the blob center. The vorticity distribution within a blob does not change as a result of convection. It can be changed by diffusion. Each blob is assumed to have a Gaussian vorticity distribution,

$$\omega = \frac{\Gamma}{\pi\sigma^2} e^{-\left(\frac{|\vec{r}|}{\sigma}\right)^2}, \quad (53)$$

where $\vec{r}$ is the position vector relative to the blob's center, Γ is the blob's strength (circulation), and σ is its core radius.

3.2 Convection of Vorticity

The velocity field at any point induced by a 2-D vortex blob can be calculated using the Biot-Savart law [1, 15],

$$\vec{V} = \frac{\Gamma}{2\pi r} \hat{k} \times \hat{r}, \quad (54)$$

where r is the magnitude of the position vector relative to the blob center, $\hat{k}$ is the unit vector in the Γ direction (normal to the 2-D plane), and $\hat{r}$ is the unit vector in the $\vec{r}$ direction. The velocity induced by an array of discrete vortices is just the summation of velocities induced by the individual vortices,

$$\vec{V} = \sum_{i=1}^N \vec{V}_i, \quad (55)$$

where N is the number of free vortex blobs.

There are two problems with direct application of Eqs. (54) and (55). The first is that $r \rightarrow 0$ as the target point approaches the source point, causing Eq. (54) to become singular. In order to remove this singularity, we have implemented the high-order algebraic core function of Winckelmans and Leonard [45]. When close to a blob's center, this model computes the induced complex velocity as

$$W(z) = -\frac{i\Gamma}{2\pi z} q(\rho), \quad (56)$$

where

$$q(\rho) = \frac{\rho^2(\rho^2 + 2)}{(\rho^2 + 1)^2} \quad (57)$$

and

$$\rho^2 \equiv \frac{|z|^2}{\sigma^2}. \quad (58)$$

Here z is the complex representation of the position vector $\vec{r}$. A more detailed discussion of this implementation can be found in Ref. [38].

The second problem with the direct application of Eqs. (54) and (55) is the time required to compute the velocities. Assuming that there are an equal number of target points and source points, N , direct application of Eq. (54) is an N^2 operation that becomes computationally prohibitive when N becomes large. To make the velocity computation tractable, we have implemented a fast-multipole method. This implementation is described in detail in Chapter 5 on page 47.

3.3 Diffusion of Vorticity

As mentioned previously, there are several methods by which the viscous diffusion process may be modeled when using Lagrangian vortex elements. These methods include the diffusing core method [21] which is probably quite adequate for the outer flow field so long as the Reynolds number is moderate to large. This method is very easy to use and provides an exact solution for vortex elements with a Gaussian vorticity distribution. The diffusion process evolves by expansion of the core radius according to

$$\sigma = \sqrt{\sigma_0^2 + 4\nu(t - t_0)}, \quad (59)$$

where σ_0 is the core radius at time t_0 and ν is the fluid viscosity. Since the diffusion equation is linear, each vortex element may be diffused independently and the new vorticity field

obtained from superposition of these elements. This will exactly satisfy the diffusion equation for the field as a whole.

While the diffusing core method provides an exact means for solving the diffusion step for a number of Gaussian elements, it should be noted that one should not allow the elements to grow too large because this introduces errors in convecting the vorticity at points that are far removed from their centers. For moderate to high Reynolds numbers, the size of elements will remain within appropriate bounds.

3.4 Vortex Merging

Although the use of fast solvers greatly reduces the computational time required for evaluation of the induced velocity field, the total time required for a complete simulation is still a strong function of the number of vortices in the flow field. Since the bodies in the simulation are shedding new vortices at each time step, both the total number of vortices in the simulation and the computational time per time step increase as the simulation progresses. The purpose of vortex merging is to reduce the number of free vortices in the flowfield and, therefore, reduce the total computational time. However, vortex merging must be done in such a way as to minimize the velocity perturbations in the areas of the flowfield that are of primary interest, usually the areas in close proximity to the bodies.

For this discussion, let R represent the distance between the bodies of interest and the source vortices, and let r represent the distance between two source vortices. The velocity perturbation at the bodies of interest due to vortex merging will be insignificant if we require that

$$r \ll R. \quad (60)$$

If this condition is met, then we can replace the two vortices by a third whose strength, position, and core radius are such that we conserve the total circulation and the first and second moments of vorticity.

Let subscripts 1 and 2 denote two vortices that are to be merged and let subscript 3 denote the resulting single vortex after merging. Conservation of circulation requires that

$$\Gamma_3 = \Gamma_1 + \Gamma_2. \quad (61)$$

Let z_1 , z_2 , and z_3 be the complex coordinates of the three vortices relative to an arbitrary reference point. Conservation of the first moment of vorticity about the reference point requires that

$$z_3 = \frac{\Gamma_1 z_1 + \Gamma_2 z_2}{\Gamma_3}. \quad (62)$$

In theory the choice of reference point is completely arbitrary. However, due to the computer's finite word length, round-off errors are minimized if the reference point is located in close proximity to the two initial vortices.

The calculation for conservation of the second moment of vorticity is analogous to the calculation for mass moments of inertia of a solid body and can be found in most dynamics texts (e.g., Ref. [24]). For a discrete vortex blob, the second moment of vorticity is

$$^2M = \int r^2 d\Gamma . \quad (63)$$

For a rotationally symmetrical vorticity distribution

$$d\Gamma = \omega 2\pi r dr . \quad (64)$$

Substituting Eq. (64) into Eq. (63) and substituting the definition of ω from Eq. (53) (Gaussian distribution) gives

$$^2M = \Gamma \int_0^{\infty} e^{-\left(\frac{r}{\sigma}\right)^2} \frac{2r^3}{\sigma^2} dr . \quad (65)$$

Performing a change of variable

$$y = \left(\frac{r}{\sigma}\right)^2 \quad (66)$$

and

$$dy = \frac{2r}{\sigma^2} dr \quad (67)$$

gives

$$^2M = \Gamma \sigma^2 \int_0^{\infty} e^{-y} y dy . \quad (68)$$

From standard integral tables, the value of the integral is 1. Therefore, for a Gaussian distribution of vorticity, the second moment of vorticity of a blob relative to its own center is

$$^2M = \Gamma \sigma^2 . \quad (69)$$

Using the parallel axis theorem [24] and replacing the system of two vortex blobs with a single third one, the conservation of the second moment of vorticity requires that

$$\sigma_3 = \sqrt{\frac{^2M_1 + ^2M_2 + \Gamma_1 |z_1 - z_3|^2 + \Gamma_2 |z_2 - z_3|^2}{\Gamma_3}} . \quad (70)$$

4 Wall Layer - Outer Flow Coupling

4.1 Vortex Blob Formation

After the wall layer is remapped, the wall-layer vorticity is examined for possible formation of discrete vortex blobs. Discrete blobs are formed whenever either one of two criteria are met:

- (1) If an outer cell of the second wall sublayer (see Fig. 4 on page 19) contains circulation after the remapping step, then all of the circulation within the cells of that mesh column and within that sublayer is converted into a discrete transition blob; or
- (2) If any vorticity has moved into the third wall sublayer, then all vorticity within that mesh column and within that sublayer is also converted.

This process is more clearly illustrated with Fig. 17. For criterion (1) above, the “cut trigger layer” is the outer layer of cells within sublayer 2. The “cut line” is the boundary between sublayer 2 and sublayer 1.

The conversion process preserves the total circulation and the first and second moments of the converted vorticity. The first moment determines the blob’s location. The second moment determines the blob’s core radius. The converted vorticity is removed from the wall layer. Positive and negative vorticity are treated separately. Within each discrete vortex the vorticity is assumed to have a Gaussian distribution of the form

$$\omega = \frac{\Gamma}{\pi\sigma^2} \exp\left\{-\left(\frac{r}{\sigma}\right)^2\right\}, \quad (71)$$

where ω is the vorticity at distance r from the blob center, Γ is the total blob circulation, and σ is the core radius.

Figure 17. Vorticity Transformation From Wall Layer to Outer Flow

In order to conserve the total circulation, the circulation within a new vortex blob is the sum of the circulations within each of the wall layer cells that were used to form the new blob, *i.e.*,

$$\Gamma_b = \sum_{i=1}^n \Gamma_i, \quad (72)$$

where subscript *b* designates the new blob, *i* designates the cell index, and *n* is the number of cells used to create the blob.

The first moment of circulation, M_1 , about any reference point is given by

$$M_1 \equiv \Gamma \bar{z} = \int_A z d\Gamma, \quad (73)$$

where *z* is the complex coordinate relative to the reference point and $\bar{z}$ is the complex coordinate of the center of circulation relative to the reference point. For blob formation purposes, we assume that the cells are rectangular with evenly distributed vorticity. The first moment for the donor cells is, therefore,

$$M_{1_c} = \sum_{i=1}^n \Gamma_i z_i, \quad (74)$$

where the subscript *c* denotes the donor cells and the remaining subscripts are defined as above. The center of the newly formed blob is

$$z_b = \frac{M_{1_c}}{\Gamma_b}. \quad (75)$$

The second moment of circulation, M_2 , about some reference point is

$$M_2 \equiv \int_A |z|^2 d\Gamma = \int_A |z|^2 \omega dA. \quad (76)$$

As above, we assume that each donor cell has a rectangular geometry and a uniform vorticity distribution. The second moment of vorticity within each cell about its own cell center is

$$M_{2_c} = \frac{\Gamma_c (l^2 + w^2)}{12}, \quad (77)$$

where *l* and *w* are the cell's length and width, respectively. Transferring these moments to the center of the newly created blob and summing over the number of donor cells gives

$$M_{2_c} = \sum_{i=1}^n \Gamma_i \left\{ \frac{(l^2 + w^2)_i}{12} + |z_i - z_b|^2 \right\} \quad (78)$$

where z_b is given by Eq. (75). Applying Eq. (76) to a blob with a Gaussian vorticity distribution yields

$$M_2 = 2\Gamma \int_0^\infty \exp\left\{-\left(\frac{r}{\sigma}\right)^2\right\} \left(\frac{r^3}{\sigma^2}\right) dr. \quad (79)$$

Performing the indicated integration, Eq. (79) becomes

$$M_2 = \Gamma \sigma^2. \quad (80)$$

Thus, the core radius of the newly formed blob is

$$\sigma_b = \sqrt{\frac{M_{2_c}}{\Gamma_b}}. \quad (81)$$

4.2 Transition Blobs

A discrete vortex blob is classified as either a free or transition blob depending upon its location. If a blob's center is within wall sublayer 2 (see Fig. 4 on page 19), that blob is a transition blob. Once the blob has moved outside sublayer 2, it is a free vortex blob. Each newly formed blob is initially a transition blob. This classification is primarily for code bookkeeping. The only meaningful difference between transition and free blobs is the means for computing their convective velocity. The convective velocity for free blobs is computed from the Biot-Savart law using a Greengard-Rokhlin fast solver (see Chapter 5 on page 47). For transition blobs the convective velocity is interpolated from the wall layer convective velocities. Blobs can move freely into and out of the wall layer, and their classification is changed accordingly. If a blob moves into wall sublayer 1, it is absorbed into the wall layer and ceases to exist as a discrete blob. Each cell receives only that blob circulation that overlaps its area.

4.3 Mapping Blobs Into the Wall Layer

In order to compute convective velocities using Eqs. (38) and (39) in Section 2.5, all vorticity along the integration path must be included. For these calculations all blobs that overlap the wall layer, both transition and free, are mapped into the layer. Because a Gaussian distribution does not have finite support, the blob distribution for this mapping is assumed to be a right-circular cylinder. The total circulation and blob center are unchanged. However, the radius of the

cylinder is not the Gaussian core radius. The second moment of vorticity for a right-circular cylinder is

$$M_{2c} = \frac{\Gamma r^2}{2}. \quad (82)$$

Equating Eqs. (82) and (80) shows that the radius of the cylindrical distribution is

$$r = \sqrt{2}\sigma. \quad (83)$$

Temporary wall layer arrays are used to hold this remapped blob vorticity. Each temporary cell receives only the amount of vorticity that overlaps its area.

4.4 Free Blobs vs. Free Vortices

For bookkeeping purposes there are two classes of free vortices. All discrete vortices that lie outside of the inner two sublayers of the wall layer are classified as *free vortices*. A subset of this population, the *free blobs*, are also formed to improve computational efficiency. When performing calculations within a wall layer, all free vortices that are in close proximity to the body must be examined to see if they overlap the wall layer. If so they must be temporarily mapped into the wall layer (see previous section). In order to reduce the size of the vortex population that must be searched at each wall layer time step, the following procedure is followed. At the start of a wall layer calculation, a temporary square box with sides equal to twice the body diameter is drawn around the body of interest. All free vortices that lie within this box are also classified as free blobs. Their properties are copied into a temporary set of free blob arrays and additional properties needed for the wall layer calculations are computed. Only the free blobs are searched for wall layer overlap.

4.5 Wall Layer Panels

In computing the convective velocities of the free vortices, all vorticity within the flow field must be included. This includes the vorticity within each body's wall layer. In order to perform this calculation, the vorticity within each wall layer cell is treated as a zero thickness vortex sheet with constant strength along its length. The details of how these velocity calculations are performed are contained in Section 5.2 on page 56.

5 Fast Multipole Method

Motivation for the fast solver, developed as part of this work by Strickland and Baty [38, 39], results from a need to be able to perform fast, yet accurate, calculations of the two-dimensional flow field produced by large numbers of vortices. Large numbers, N_V , of vortices (1,000 to 100,000) are required for many simulations such as the present one. Since the flow must be convected as part of the evolution process, the velocity perturbations from all vortices on all other vortices must be calculated at each time step. This requires an amount of work that is proportional to N_V^2 for each time step using direct calculation methods. The Greengard-Rokhlin method requires an amount of work which is proportional to $N_V \ln N_V$ which saves significant CPU time in large simulations. In the solver developed as part of this work [38, 39], we have added features that allow one to efficiently evaluate field variables at target points that are not coincident with the location of the source or vortex positions. The ability to specify separate source and target fields provides one with an efficient means for calculating boundary conditions, trajectories of passive scalar quantities, and field variable plots (stream function & velocity). In the present report, we will present a brief overview of the method along with the results from a couple of benchmark studies. For more details see Refs. [38] and [39].

5.1 Fast Multipole Method for Vortex Blobs

5.1.1 Method Overview

The general method comes from work developed by Rokhlin [31], Greengard and Rokhlin [14], and Carrier, Greengard and Rokhlin [4]. In the work of Carrier, Greengard and Rokhlin [4], it was shown that the asymptotic CPU time was of order $N_V \ln N_V$ for two-dimensional problems. A break-even point on the order of 100 particles was achieved for single precision accuracy for a uniform field in a square box. For 25,600 particles in the box, their method was roughly 100 times faster than the direct method. These fast methods take advantage of multipole and local series expansions that enable one to make calculations for interactions between groups of particles that are in well-separated spatial domains rather than having to consider interactions between every pair of particles. In the present work, the spatial domains in which vortices are located are obtained in an adaptive manner that is similar to the two-dimensional scheme used by Carrier, Greengard, and Rokhlin [4]. However, unlike their work, the locations of the source particles (vortices) need not be the same as the locations of the target (evaluation) points.

Consider the two domains D_S and D_F that are depicted in Fig. 18. Domain D_S contains a set of vortices that are acting upon (producing perturbation velocities at) a set of evaluation points

Figure 18. Source and Field Point Domains

contained in domain D_F . In some cases, the set of evaluation points in domain D_F might also represent the centers of a second set of vortices. In other cases they might simply represent an arbitrary set of evaluation points. In order to implement this method, a multipole series expansion for the complex potential about the center of the source domain D_S is first obtained. This expansion includes all of the vortices that are inside D_S . The series converges for field points that are outside of D_S . In order to be somewhat conservative, the expansion about the center of the source domain is used only for field points that are separated by at least R_S from the boundary of the source domain. Thus, for field points that are within a circle of radius $2R_S$ with respect to the source domain center, the influence of vortices within domain D_S on the field point is computed directly. The center of the source domain can be shifted without actual evaluation of the multipole series, allowing one to efficiently form expansions for larger source domains that contain D_S .

A local or Taylor series expansion for the complex potential can also be obtained about the center of the field domain D_F . The series converges for field points that are inside D_F . The expansion about the center of the field domain is used only for source points that are separated by at least R_F from the boundary of the field domain. Thus, for source points that are within a circle of radius $2R_F$ with respect to the field domain center, the influence of any vortex in domain D_S on a field point is computed directly. The center of the field point domain can also be shifted without actual evaluation of the local series, allowing one to form expansions for smaller field point domains contained within D_F .

Figure 19. Adaptive Domain Mesh Scheme

As mentioned previously, the method used to divide the flow field into domains is an extension to that used by Carrier, Greengard, and Rokhlin [4]. The method requires that an adaptive domain mesh be generated that encloses all the source and target points in the flow. A square box is first constructed that encloses all of the sources in the flow as well as all of the target points. This box represents the zeroth level source box as well as the zeroth level target box. We first subdivide the source box into four boxes, if it contains more than a specified number of vortices. Each of these boxes that contains more than the specified number of vortices is further subdivided. This process is continued until all of the boxes contain less than the specified number of vortices. The grid for the target points is generated in a similar manner. Fig. 19 illustrates the generation of a source and target mesh. Each box size is associated with a box level, with the original box level being level 0. At each level, empty boxes are ignored. The ability to ignore regions of the flow-field in which vortices or evaluation points are not present is a feature of the method that provides significant time savings. Boxes that can be subdivided are parent boxes. Those that cannot are childless boxes.

A somewhat involved procedure is used to define the separation condition between a particular target box and each of the source boxes in the flow. This procedure determines the way in which vortices in a particular source box influence the complex potential at evaluation points

in a particular target box. In general, each target box at each level has five possible relationships with each source box in the mesh. In general, the five lists produce the following types of restrictions on the use of the series expansions:

- List 1: Direct calculations must be made; multipole and local series expansions cannot be used.
- List 2: Both multipole and local series expansions can be used.
- List 3: Multipole series expansions can be used, local series expansions cannot.
- List 4: Local series expansions can be used, multipole expansions cannot.
- List 5: Contributions from distant source boxes reside in the parent of the field box.

It should be pointed out that the adaptive domain meshes that are generated here are not to be confused with the adaptive meshes that are generated for conventional CFD formulations, nor with the wall layer mesh. In the present case, the meshes represent a domain decomposition that is very simple when compared to meshes generated for conventional CFD formulations. Mesh generation for conventional CFD formulations is for the purpose of writing difference or element equations that satisfy the governing differential flow equations, whereas the domain meshes generated here are for the purpose of grouping source or target points. The domain meshes in the present case do not have to conform to any flow boundaries, they extend only to regions where source or target points reside, and they are generated using a very simple algorithm. Generation of adaptive domain meshes is achieved using a very small amount of CPU time, whereas adaptive mesh generation for conventional CFD problems can be quite CPU time intensive.

5.1.2 Benchmark Studies of Fast Solver

The important dependent variables in the benchmark tests were the CPU run times and the truncation errors associated with the calculation of field variables. The independent variables were the number of vortex sources and target points in the field, the distribution of the sources and target points in the field, the number of terms used in the multipole and local series expansions, and the maximum number of source and target points allowed in any childless box in the source and target meshes. In this work, we were especially interested in how to set the maximum number of source and target points allowed in any childless box in the source and target meshes in order to minimize CPU time, irrespective of the configuration of source and target points. In view of the large number of parametric variations that might ultimately be useful, we ran a somewhat limited set of test cases to examine truncation error and CPU run times.

Four general configurations of vortex and target point placements were studied as indicated in Fig. 20. In the first set of cases, the source and target points were uniformly placed. The total number of source points chosen were sometimes different from the total number of target points. This is labeled as the uniform-uniform (U-U) case. For the second set of cases, all of the source and target points were arranged on a spiral. This case is designated as the spiral-spiral (S-S) case. For the third set of cases, the sources were on the spiral while the targets were uniformly spaced (S-U). Finally, for the fourth set, the sources were uniformly spaced while the targets were on the spiral (U-S).

Figure 20. Benchmark Test Cases

Before presenting any benchmark results, a metric for the truncation error must be defined. One such measure of the error incurred by truncation of the multipole and local series after the N^{th} order terms can be obtained for the stream function and the magnitude of the velocity vector field by means of the following equations:

$$\epsilon_{\psi} = \left[\frac{\sum_{i=1}^{N_v} (|\psi|_i - |\psi|_{ei})^2}{\sum_{i=1}^{N_v} |\psi|_{ei}^2} \right]^{\frac{1}{2}}, \quad (84)$$

$$\epsilon_U = \left[\frac{\sum_{i=1}^{N_v} (|U|_i - |U|_{ei})^2}{\sum_{i=1}^{N_v} |U|_{ei}^2} \right]^{\frac{1}{2}}.$$

Here, $|\psi|_i$ and $|U|_i$ are the magnitudes of the stream function and velocity vector at point i as calculated from the truncated series. The quantities $|\psi|_{ei}$ and $|U|_{ei}$ are the magnitudes of the stream function and velocity vector at point i as calculated exactly using the direct method. As indicated, these error calculations are obtained from summations over all of the N_v vortices in the flow. Assuming that single precision accuracy means that any number stored in the

computer will be accurate through 6 significant digits, then single precision round off errors will produce values of ϵ_ψ and ϵ_U up to about 1×10^{-6} . Therefore, errors calculated from Eqs. (84) that are equal to or less than 1×10^{-6} imply that the machine round-off errors are possibly as large as the truncation errors. For such cases, one should not continue to add terms to the series expansions for added accuracy unless the machine precision is also increased.

In this report, we first present the details of our benchmark study for the U-U case. In order to investigate the relationship between accuracy, CPU time, and optimal values for the maximum number of sources and target points in a box, a series of runs were made in which 10,202 vortices, each with unit strength, were used. A direct calculation required 356 seconds on a Sun Sparc 20 workstation. In this series, the maximum number of sources N_S and target points N_F in a box were set equal to each other and varied between 10 and 60. The number of terms used in the series expansions N_{max} was varied between 4 and 20. Results from these runs are tabulated in [38] and plotted in Figs. 21 and 22.

From Fig. 21 it can be seen that the error in the stream function ϵ_ψ is on the order of 1×10^{-6} when the number of terms in the series expansion $N_{max} \geq 6$. The error in the velocity ϵ_U for $N_{max} \geq 6$ is less than 1×10^{-4} , which is probably acceptable for most engineering calculations. The error appears to be insensitive to the values of N_S and N_F over the range of 10 to

Figure 21. Error Plots for Uniform Distribution of 10,201 Vortices

Figure 22. CPU Times for Uniform Distribution of 10,201 Vortices

60. From Fig. 22 it can be seen that a range of values for N_S and N_F between 20 and 60 will yield good results with regard to optimal CPU times.

In order to examine the effect of varying N_S , N_F , N_V , and N_T on CPU time, 150 cases were run with $N_{max} = 6$. A set of six contour plots are given in Fig. 23. The contours represent lines of constant CPU time, normalized by dividing actual CPU time by the minimum CPU time associated with a given plot. These plots suggest that in order to make optimal choices for N_S and N_F an algorithm such as the following might be used:

$$\begin{aligned} N_S &= 30 \max(1, \sqrt{N_V/N_T}) \\ N_F &= 30 \max(1, \sqrt{N_T/N_V}) \end{aligned} \quad (85)$$

It is also worth noting that if one simply lets $N_S = N_F = 30$, the CPU time will be within about 50% of the optimal value for the cases shown in Fig. 23.

We now examine the CPU times and errors associated with U-U cases when $N_{max} = 6$ with N_S and N_F being governed by Eq. (85). A set of runs were made with $N_V = N_T$ where the number of vortices and target points varied from about 100 to 30,000. In addition, four cases were run where $N_V/N_T = 1/9, 1/2, 2, 9$. All of these cases are tabulated in Ref. [38] and plot-

Figure 23. Optimum N_S and N_F for U-U Case ($N_{max} = 6$)

Figure 24. CPU Times for U-U Case (Direct versus Fast)

Figure 25. Errors Using Fast Solver for U-U Case

ted in Figs. 24 and 25. In order to plot the results for the four cases where $N_V \neq N_T$, a value of $\sqrt{N_V N_T}$ was used on the abscissa.

From Fig. 24 and the data in Ref. [38] it can be shown that the CPU times for the direct calculations are proportional to the square of N_V , or the square of N_T , when $N_V = N_T$. When the number of source and target points are not equal, the CPU time for the direct calculation is proportional to $N_V N_T$. The CPU times for the fast solver are well represented by the following equation in the indicated range of $N_V N_T$:

$$t_{cpu} = 1.05 \times 10^{-4} (\sqrt{N_V N_T} \ln \sqrt{N_V N_T}) . \quad (86)$$

It is satisfying to see that the data for different numbers of source and target points collapse onto the same CPU time curves.

From Fig. 25 we see that the error in the stream function is on the order of 10^{-6} while the error in the velocity magnitude is on the order of 10^{-4} , except for $\sqrt{N_V N_T} = 100$. For this case the solution is essentially a direct solution and the error should be very small. It is again interesting to note that the error for the cases where $N_V \neq N_T$ collapses onto the same curves where $N_V = N_T$.

Details pertaining to the S-S, U-S, and S-U benchmark cases may be found in Ref. [38]. In general, the CPU times tended to be less for these cases than for the U-U case with the same number of terms in the multipole expansions. In all cases, we found that the algorithm for values of N_S and N_F given by Eq. (85) was close to optimal.

5.2 Fast Multipole Method Extended to Vortex Panel Sources

Even with the implementation of the Fast Multipole Method (FMM) described in §5.1 to evaluate the influence of all the transition blobs on the wake vortices, and of all the wake vortices on one another, run times for the VFLOW2D code were higher than anticipated. Profiling studies were done to assess what portions of the code were consuming the most time. The biggest offender, particularly for longer simulation times, was the calculation of the velocity induced upon the wake vortices by the wall-layer vortex panels (*cf.* §4.5). For long simulation-time runs, this portion of the code was taking as much as 65% of the total CPU time.

These panel-to-point interactions were being computed in a direct interaction manner between each vortex panel/wake vortex pair. If N_V denotes the number of wake vortices in the problem, and N_{VP} the number of vortex panels, the CPU time for such a direct calculation can be expected to scale as $O(N_V N_{VP})$. Although N_V can easily number in the thousands for these simulations, N_{VP} is at most a few hundred. Our original thought was that the time for this direct calculation would be relatively small. It turned out, however, that each individual calculation is expensive, causing the time for the $N_V N_{VP}$ calculations to be prohibitive. It appeared that applying some form of FMM to this interaction could profitably reduce run times.

Rather than write an entirely separate algorithm and associated subroutines for the vortex panel/wake vortex interactions, the decision was made to adapt the existing Greengard-Rokhlin FMM to this task. A similar approach has been presented by Najm, *et al.*, in Ref. [27]. In their problem, the computational domain is two-dimensional and rectangular: straight panels with linearly varying vortical and dilatational source strengths are used to model the (straight) domain boundaries. Thus, adjacent panels are co-linear (except at corners), which simplifies the tree generation process. In the present problem, each panel is assumed to have an arbitrary orientation, which raises new questions about how to generate the source tree for such a configuration. §5.2.1 begins with a description of how the source tree generation process was modified to accommodate vortex panels. §5.2.2 and §5.2.3 present the potential and velocity fields induced by a vortex panel of linearly varying strength. This forms the basis for the multipole source expansions, that are developed in §5.2.4 through §5.2.7. Comparisons between the exact solution and the expansion for a single panel are discussed in §5.2.8. The procedures for shifting a panel's source coefficients to obtain coefficients for all the boxes in the source tree, and then obtain field expansions for all the boxes in the target tree, are described in §5.2.9 and §5.2.10. Finally, §5.2.11 concludes with some sample calculations comparing predicted force coefficients and run times using both direct interactions and the extended FMM.

5.2.1 Generating the Vortex Panel Source Tree

Originally, the Greengard-Rokhlin FMM was called twice at each time step: first with a source population consisting of the transition blobs, and then a second time with a source population consisting of the free wake vortices. In both calls the target population, *i.e.*, the points at which the induced field was being calculated, was the collection of free wake vortices. Hence, both source and target populations consisted only of point singularities. The third contribution to the wake vortex velocity field, that induced by the vortex panels in the wall layer, was computed using direct interactions between each vortex panel/wake vortex pair.

In adapting the FMM to treat the vortex panel contribution as well, the first question that arose was whether this should somehow be merged into an existing call to the FMM — *i.e.*, have a source population consisting of both zero-dimensional (point) and one-dimensional (panel) sources in the same call — or whether it should be made a separate call to the FMM, with a source population consisting only of panels. The first choice would require major modifications to the source data structure and to the FMM internal logic so that it could determine whether a given source singularity was a point or a panel. Making a separate call with just panel sources greatly reduces the necessary changes to the data structure (the panels would have their own, independent of the point sources) and internal logic, at the expense of regenerating the (same) target tree yet a third time.

As it happens, the time required to regenerate the target tree is minuscule, so it was decided to implement the vortex panel contribution as a separate call to the FMM. A scalar input containing the dimensionality of the source population, SDIM, was added to the argument list, along with additional arrays defining the location, length, and strength of the vortex panels. Thus, when SDIM = 0, the sources are all zero-dimensional points and the new panel arrays are ignored. Conversely, when SDIM = 1, the sources are all one-dimensional panels and the original point source data arrays are ignored.

Figure 26. Vortex Panel Straddling Multiple Boxes

When constructing a source tree about a vortex panel population, several unique issues arise that were not present in the case of point sources. As described in §5.1, the Greengard-Rokhlin algorithm divides parent boxes into four child boxes at the next finer level in the tree by bisecting the parent in each of the two coordinate directions. The source population in each child box is then counted. Those child boxes whose population is below the specified maximum number of sources, N_S , are deemed *childless* and are not further divided. Those whose population is above this maximum are themselves termed *parents*, and the process is repeated recursively.

It is obvious that for an arbitrary distribution of panels, sooner or later in this process a panel in the parent is going to straddle two, or possibly even three, of its child boxes as shown in Fig. 26. Unlike point vortices that have no dimension, this introduces an ambiguity into the choice of which box we should say the panel resides in. The most ambitious approach to resolving this dilemma would be to divide any panel that straddles multiple child boxes into sub-panels, such that each sub-panel is confined to a single box. Intuition suggests this would be a logistical nightmare and probably negate any advantage of using a Fast Multipole Method. Less ambitious, but still non-trivial, would be the implementation of an algorithm that would choose that box containing the greatest fraction of the panel's total length—but even this is not likely to be very efficient and would require major modifications to the tree-generation routines.

In keeping with our philosophy of reusing as much of the existing software as possible, we chose the simplest and most straightforward method to implement: The panel is said to reside in whichever box contains its center. Thus, in both cases shown in Fig. 26, the panel is assigned to the upper right-hand box, even though in Fig. 26(b) that box happens to contain the smallest fraction of its length. In cases where the center happens to lie right on a boundary, the box with the larger center coordinate value is arbitrarily chosen. The diabolical case of a panel whose center lies exactly on the parent's center would thus also be assigned to the upper right-hand child box. The implementation of this scheme simply involves replacing the input array containing the point vortex locations with another containing the vortex panel center locations.

Another related issue that first arises with vortex panels is the necessity of limiting the smallest size box into which the domain can be divided, in effect imposing an additional limit on the number of levels in the source tree. At its heart, the Greengard-Rokhlin algorithm is a means of creating a hierarchy of boxes that lump together sources/targets for the purpose of determining which sources and targets are “well-separated”, and can thus make use of the various expansions. In the case of source expansions, “well-separated” means only that the separation between source and target boxes, say Δz , is larger than some characteristic dimension of the source distribution. As long as our elemental sources are point (*i.e.*, dimensionless) vortices, the source box length at any level l , say L_l , is a suitable characteristic dimension, and thus we simply require $\Delta z > L_l$ for valid use of the source expansions.

However, if we naively carry over this criterion to a source tree built around vortex panels, we could well find ourselves applying source expansions inappropriately. As a concrete example, imagine a panel, or panels, of length 10 (arbitrary units) whose center(s) lie(s) in, and hence is presumably “contained” by, a source box of length 1. The separation criterion $\Delta z > L_l$ alone would admit use of the source expansions for a target box located only 2 units away, even though this is only one-fifth the dimension of the source distribution, and hence obviously still very much in the “near-field”. Clearly, if we wish to continue using this simple separation criterion, we must restrict the minimum size to which the source boxes can be divided, so that their size remains representative of the dimensions of the sources they contain.

We have already alluded to the fact that a given branch of the source tree terminates when the population in a box falls below a pre-specified level, N_S . This is carried over to the case with vortex panels as well. That is, if N_{panel} is the number of panels in the box, that branch will terminate at a maximum level based on population of

$$l_{max}^{pop} = \text{Level at which } N_{panel} \leq N_S. \quad (87a)$$

To this we now add the second, independent, requirement that $L_l = l_{max} = h(L_{panel})_{max}$, where $(L_{panel})_{max}$ is the maximum length of any of the vortex panels in the flow, and h is a pre-specified constant ≥ 1 . However, since $L_{l_{max}} = L_0 / 2^{l_{max}}$, where L_0 is the size of the outer-most square at Level 0, this predicts that the branch will terminate at a maximum level based on box size of

$$l_{max}^{size} = \left\lceil \log_2 \frac{L_0}{h(L_{panel})_{max}} \right\rceil, \quad (87b)$$

where $\lceil \dots \rceil$ indicates that the result is rounded down to the nearest integer. If Eq. (87b) is satisfied, the simple separation criterion $\Delta z > L_l$ automatically ensures that $\Delta z > L_{l_{max}} \geq (L_{panel})_{max}$, which was our goal.

The final level of any given branch, l_{max} , is determined by whichever of the above two criterion is triggered first, *i.e.*,

$$l_{max} = \min(l_{max}^{pop}, l_{max}^{size}). \quad (87c)$$

Equation (87a) - (87c) must be tested for each box at each level to decide whether that branch of the tree should be terminated or not. It should be noted that l_{max}^{pop} depends only on the spatial distribution of the panels, but not their size, and may vary significantly from branch to branch. l_{max}^{size} on the other hand, is determined solely by the size of the largest panel, and will have the same value for all branches of the tree. Thus it is difficult to tell *a priori* which will dictate l_{max} . In most cases this means that different branches of the tree will terminate at different levels, but this was already true of the source tree generated for point vortices and hence poses no particular difficulty.

This still leaves open the issue of what values to assign to N_S in Eq. (87a) and h in Eq. (87b). Time and resources did not allow us to perform a comprehensive set of benchmark calculations to determine optimal values, as was done in the case of point vortex sources, §5.1.2. However, as noted in that discussion, the CPU time in the vicinity of the optimal N_S for point sources is not a strong function of N_S — *i.e.*, the minimum in t_{CPU} is rather broad. For this reason, and the fact that the multipole expansions for a distribution of vortex panels are not significantly more complex than those for point vortices (as will be seen below), the value of N_S predicted by Eq. (85) is also used when treating vortex panels.

As for the value of h , too low a value will allow more levels in the source tree, and thus more opportunities to use the (faster) multipole expansions, but at the expense of incurring a greater error when panels protrude from the boundaries of their source box. Conversely, too high a value may greatly reduce the number of levels in the source tree, forcing the use of (slower, but more exact) direct interactions. Consider the tree sketched in Fig. 27. A target domain is labelled, along with members of its Box Lists 2 and 3, as categorized in §5.1. These are the only two lists that admit use of source expansions; hence, it is not necessary to consider Box Lists 1, 4, or 5 for the purposes of this discussion. Let us assume that the List 3 source box is at level l_{max} , as dictated by $(L_{panel})_{max}$ through Eq. (87b) using $h = 2$, and further that panels of this length reside in each of the List 2 and 3 boxes as shown. The panels have been deliberately placed in what is believed to be the most detrimental positions from the perspective of using source expansions: Their centers are just barely inside the boxes they are members of, with slightly less than half their length protruding beyond the border, and an “edge-on” orientation relative to the target domain.*

The List 2 Box (both source and local expansions may be used) is by definition the same size as the Target Box, and separated from it by a distance of at least that much. The List 3 Box (only source expansions can be used) is separated from the Target Box by a distance equal to its side, and since List 3 boxes must be smaller than the Target Box, this separation is less than in the case with the List 2 Box. Therefore, the situation depicted with the List 3 Box is the “worst case” scenario: If h is made large enough to allow the safe use of source expansions

* The reasons for this are discussed more fully in §5.2.8. Skimming that section first may make it easier to follow the present discussion.

Figure 27. Panels Protruding from Lists 2 and 3 Source Boxes for $h = 2$

with List 3 Boxes, it will admit their use with List 2 Boxes as well. The minimum separation between the center of the List 3 Box panel and any field point in the Target Box is two panel lengths. Figure 32 in §5.2.8 indicates a maximum relative error for $N_{max} = 6$ (the value used here) of $\epsilon \approx 10^{-4}$ at this separation, which should suffice for engineering applications. Note too that in “real-world” problems, the errors from multiple such scenarios will to some extent be self-cancelling, so the cumulative error may be even less. On this basis $h = 2$ was chosen for use in Eq. (87b) as a reasonable compromise between accuracy and efficiency.

This concludes the discussion of how the generation of the source tree was generalized to accommodate vortex panel sources. The generation of the target tree remains unchanged, as the targets always consist of dimensionless points. This includes use of the same estimate for N_F as given in Eq. (85). The next step in the FMM is the calculation of the multipole source expansion coefficients; but first the analytical expressions for the potential and velocity fields induced by a panel, that are the foundation for the required expansions, are derived.

5.2.2 Panel-Induced Field

We first derive the analytical expressions for the complex potential and velocity fields induced at a point by a straight vortex panel of linearly varying strength. These expressions are the basis for the “direct” interactions first used to compute the vortex panel/wake vortex interactions and also serve as the starting point for the multipole expansions needed by the FMM.

Figure 28. Vortex Panel with Linearly Varying Strength

Consider a 2D point vortex with strength Γ located at the origin in the complex z -plane. The complex potential it induces on a field point located at z is given by (Ref. [38], Eq. (1)):

$$F_{\text{point}}(z) = \frac{-i\Gamma}{2\pi} \ln(z). \quad (88)$$

Now consider a straight vortex panel of length $2|z_e|$ extending from $-z_e$ to $+z_e$ in the complex z -plane, as shown in Fig. 28. Its strength, in units of circulation per unit length (or velocity), is assumed to vary linearly from γ_a at one end to γ_b at the other. The scalar distance along the panel is denoted by s , which varies from $-|z_e|$ to $+|z_e|$. From Eq. (88), the complex potential induced at field point z by an incremental element of the panel of length ds at z' , with circulation $\gamma(s)ds$, is given by:

$$dF_{\text{panel}}(z, z') = -i \frac{\gamma[s(z')] ds}{2\pi} \ln(z - z').$$

The potential induced by the entire panel is obtained by integrating the above over s :

$$F_{\text{panel}}(z) = -\frac{i}{2\pi} \int_{-|z_e|}^{|z_e|} \gamma[s(z')] \ln(z - z') ds. \quad (89)$$

The linear variation $\gamma(s)$ that reproduces $\gamma(-|z_e|) = \gamma_a$ and $\gamma(+|z_e|) = \gamma_b$ is

$$\gamma(s) = \left(\frac{\gamma_a + \gamma_b}{2} \right) + \left(\frac{\gamma_b - \gamma_a}{2|z_e|} \right) s. \quad (90)$$

The scalar s can in turn be expressed in terms of the complex dummy variable of integration, z' , as

$$z' = s e^{i\theta} = \begin{cases} -z_e & \text{when } s = -|z_e| \\ 0 & \text{when } s = 0 \\ z_e & \text{when } s = |z_e| \end{cases} \quad s = z' e^{-i\theta} \quad ds = dz' e^{-i\theta}. \quad (91)$$

Substituting Eq. (90) and Eq. (91) into Eq. (89), and noting that $|z_e|e^{i\theta} = z_e$, the complex potential induced by the panel can now be written:

$$F_{\text{panel}}(z) = \int_{-z_e}^{z_e} (C_1 + C_2 z') \ln(z - z') dz', \quad (92)$$

where

$$C_1 = \frac{-ie^{-i\theta}(\gamma_a + \gamma_b)}{4\pi} \quad C_2 = \frac{-ie^{-i\theta}(\gamma_b - \gamma_a)}{4\pi z_e}. \quad (93)$$

Carrying out the integrations gives

$$F_{\text{panel}}(z) = C_1 \left\{ z [\ln(-z - z_e) - \ln(-z + z_e)] + z_e [-2 + \ln(z - z_e) + \ln(z + z_e)] \right\} + \quad (94)$$

$$C_2 \left\{ -zz_e + z^2 \left[\frac{1}{2} \ln(-z - z_e) - \frac{1}{2} \ln(-z + z_e) \right] + z_e^2 \left[\frac{1}{2} \ln(z - z_e) - \frac{1}{2} \ln(z + z_e) \right] \right\}$$

After some manipulation, the above result can be written in the slightly more compact form:

$$F_{\text{panel}}(z) = C_1 z \left\{ \ln \left(\frac{1 + \frac{z_e}{z}}{1 - \frac{z_e}{z}} \right) + \frac{z_e}{z} \left[\ln \left(z^2 \left(1 - \frac{z_e^2}{z^2} \right) \right) - 2 \right] \right\} + \quad (95)$$

$$C_2 \frac{z^2}{2} \left\{ \left(1 - \frac{z_e^2}{z^2} \right) \ln \left(\frac{1 + \frac{z_e}{z}}{1 - \frac{z_e}{z}} \right) - 2 \frac{z_e}{z} \right\}$$

The complex velocity is most easily obtained by differentiating the complex potential, *i.e.*,

$$W_{\text{panel}}(z) = \frac{dF_{\text{panel}}(z)}{dz}. \quad (96)$$

For this purpose it is probably easier to start from Eq. (94) rather than (95), as the former has the dependence on z more expanded. After more algebra, the end result is

$$W_{\text{panel}}(z) = C_1 \ln \left(\frac{z + z_e}{z - z_e} \right) - C_2 z_e \left[2 - \frac{z}{z_e} \ln \left(\frac{z + z_e}{z - z_e} \right) \right], \quad (97a)$$

or equivalently

$$W_{\text{panel}}(z) = C_1 \ln \left(\frac{1 + \frac{z_e}{z}}{1 - \frac{z_e}{z}} \right) + C_2 z \left[\ln \left(\frac{1 + \frac{z_e}{z}}{1 - \frac{z_e}{z}} \right) - 2 \frac{z_e}{z} \right]. \quad (97b)$$

5.2.3 Point Vortex Limit

Before proceeding further, let's make sure that the results so far make sense by performing a simple test: In the limit where the vortex panel gets very small compared to the distance to the observer, the potential should approach that of a point vortex, as given by Eq. (88). However, if we simply take the limit $z_e/z \rightarrow 0$ in Eq. (95), all the terms vanish. That's because we've inadvertently reduced the panel's *strength* as well as its size to zero. The correct way to do it is let $z_e/z \rightarrow 0$ while simultaneously holding the panel's integrated circulation,

$$\Gamma = \int_{-|z_e|}^{|z_e|} \gamma(s) ds = (\gamma_a + \gamma_b) |z_e|, \quad (98)$$

constant. This is tantamount to requiring that the combination $(z_e/z)C_1$ remain constant (cf. Eq. (93)) while $z_e/z \rightarrow 0$. The slope of the variation, C_2 , is also held constant during this process. The term proportional to C_2 still vanishes, but now we are left with

$$\lim_{\substack{z_e/z \rightarrow 0 \\ C_1 z_e, C_2 \sim \text{const}}} F_{\text{panel}}(z) = C_1 z \left\{ \ln \left(1 + \frac{z_e}{z} \right) - \ln \left(1 - \frac{z_e}{z} \right) + \frac{z_e}{z} \left[\ln z^2 + \ln \left(1 - \left(\frac{z_e}{z} \right)^2 \right) - 2 \right] \right\}$$

Now expand the logarithmic terms for small (z_e/z) using

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} + O(x^4) \dots \quad \text{for } |x| < 1. \quad (99)$$

This gives,

$$\begin{aligned} \lim_{\substack{z_e/z \rightarrow 0 \\ C_1 z_e, C_2 \sim \text{const}}} F_{\text{panel}}(z) &= C_1 z \left\{ \frac{z_e}{z} - \frac{1}{2} \left(\frac{z_e}{z} \right)^2 + \frac{z_e}{z} + \frac{1}{2} \left(\frac{z_e}{z} \right)^2 + \frac{z_e}{z} [\ln z^2 - 2] + O \left(\frac{z_e}{z} \right)^3 \dots \right\} \\ &= C_1 z \left\{ 2 \frac{z_e}{z} + 2 \frac{z_e}{z} [\ln z - 1] \right\} = 2 C_1 z_e \ln z = \frac{-i e^{-i\theta} (\gamma_a + \gamma_b) z_e}{2\pi} \ln z \end{aligned}$$

But $z_e e^{-i\theta} = |z_e|$. With this and Eq. (98), we have finally

$$\lim_{\substack{z_e/z \rightarrow 0 \\ C_1 z_e, C_2 \sim \text{const}}} F_{\text{panel}}(z) = \frac{-i(\gamma_a + \gamma_b)|z_e|}{2\pi} \ln z = \frac{-i\Gamma}{2\pi} \ln z,$$

which is precisely Eq. (88), as expected.

Similarly, we can check the behavior of the velocity field described by Eq. (97b) in the same limit:

$$\begin{aligned} \lim_{\substack{z_e/z \rightarrow 0 \\ C_1 z_e, C_2 \sim \text{const}}} W_{\text{panel}}(z) &= C_1 \left[\ln \left(1 + \frac{z_e}{z} \right) - \ln \left(1 - \frac{z_e}{z} \right) \right] \\ &\quad + C_2 z \left\{ \left[\ln \left(1 + \frac{z_e}{z} \right) - \ln \left(1 - \frac{z_e}{z} \right) \right] - 2 \frac{z_e}{z} \right\} \end{aligned}$$

Making use of Eq. (99) again, this yields,

$$\begin{aligned} \lim_{\substack{z_e/z \rightarrow 0 \\ C_1 z_e, C_2 \sim \text{const}}} W_{\text{panel}}(z) &= C_1 \left[2 \frac{z_e}{z} + O \left(\frac{z_e}{z} \right)^3 \dots \right] + C_2 z \left[2 \frac{z_e}{z} + O \left(\frac{z_e}{z} \right)^3 \dots - 2 \frac{z_e}{z} \right] \\ &= 2C_1 \frac{z_e}{z} = \frac{-ie^{-i\theta}(\gamma_a + \gamma_b)z_e}{2\pi z} = \frac{-i(\gamma_a + \gamma_b)|z_e|}{2\pi z} = \frac{-i\Gamma}{2\pi z} \end{aligned}$$

which is the correct result for the complex velocity induced by a point vortex at the origin.

5.2.4 Panel-Centered $\rightarrow$ Arbitrary Coordinates

For convenience in the derivation, the panel-induced potential and velocity fields in §5.2.2 assume that the origin is at the center of the panel. However, for the purpose of comparing them against the direct-interaction expressions already programmed in VFLOW2D, it is necessary to transform them to an arbitrary coordinate system. This transformation is shown schematically in Fig. 29 using both frames of reference. VFLOW2D assumes the origin O is at some arbitrary location as indicated, and that the endpoints of the panel are located at z_3 and z_4 , with source strengths γ_3 and γ_4 , respectively; the field point is located at z_I . θ_S is the panel inclination angle, measured CCW from the horizontal. These are all indicated in the figure, with the analogous panel-centered quantities shown in parentheses (...).

Clearly, the physics doesn't care where the absolute origin is located. The only two length scales that matter in determining the field strength are the distance from the center of the panel to the field point and the length of the panel. Thus, in transforming Eq. (97a) to arbitrary coordinates, z should not simply be interpreted as the distance from the origin to the field point, z_I , but rather as the distance from the center of the panel to the field point, *i.e.*,

Figure 29. Transformation from Panel-Centered to Arbitrary Coordinates

$$z \rightarrow z_1 - \frac{1}{2}(z_4 + z_3). \quad (100a)$$

Similarly, z_e should not be interpreted simply as the location of the right end of the panel, z_4 , but rather as the panel half-length,

$$z_e \rightarrow \frac{1}{2}(z_4 - z_3). \quad (100b)$$

Finally, it should be obvious that

$$\begin{aligned} \theta &\rightarrow \theta_S \\ \gamma_a &\rightarrow \gamma_3, \quad \gamma_b \rightarrow \gamma_4 \end{aligned} \quad (100c)$$

Making these substitutions in Eq. (97a), except in the coefficient $C_2 z_e$ for reasons that will become obvious below, yields

$$\begin{aligned} W_{\text{panel}} = C_1 \ln & \left(\frac{z_1 - \frac{1}{2}(z_4 + z_3) + \frac{1}{2}(z_4 - z_3)}{z_1 - \frac{1}{2}(z_4 + z_3) - \frac{1}{2}(z_4 - z_3)} \right) \\ & + C_2 z_e \left[-2 + \frac{z_1 - \frac{1}{2}(z_4 + z_3)}{\frac{1}{2}(z_4 - z_3)} \ln \left(\frac{z_1 - \frac{1}{2}(z_4 + z_3) + \frac{1}{2}(z_4 - z_3)}{z_1 - \frac{1}{2}(z_4 + z_3) - \frac{1}{2}(z_4 - z_3)} \right) \right] \end{aligned}$$

The coefficients C_1 and $C_2 z_e$, Eq. (93), transform as follows,

$$C_1 = \frac{-ie^{-i\theta_s}(\gamma_4 + \gamma_3)}{4\pi} = \frac{-ie^{-i\theta_s}\gamma_S}{2\pi}$$

$$C_2 z_e = \frac{-ie^{-i\theta_s}(\gamma_4 - \gamma_3)}{4\pi} = \frac{-ie^{-i\theta_s}\gamma_D}{2\pi}$$

where the following two quantities used by VFLOW2D have been introduced:

$$\gamma_S = \frac{\gamma_4 + \gamma_3}{2} \quad \gamma_D = \frac{\gamma_4 - \gamma_3}{2} \quad (101)$$

So the expression for W_{panel} now reads

$$W_{\text{panel}} = \frac{-ie^{-i\theta_s}}{2\pi} \left\{ \gamma_S \ln \left(\frac{z_1 - z_3}{z_1 - z_4} \right) + \gamma_D \left[-2 + \frac{\frac{1}{2}(z_1 - z_4 + z_1 - z_3)}{\frac{1}{2}(z_4 - z_3)} \ln \left(\frac{z_1 - z_3}{z_1 - z_4} \right) \right] \right\} \quad (102)$$

$$= \frac{-ie^{-i\theta_s}}{2\pi} \left\{ \gamma_S \ln \left(\frac{z_1 - z_3}{z_1 - z_4} \right) + \gamma_D \left[-2 + \frac{(z_1 - z_4 + z_1 - z_3)}{z_4 - z_3} \ln \left(\frac{z_1 - z_3}{z_1 - z_4} \right) \right] \right\}$$

This is precisely the expression originally programmed in VFLOW2D for the panel-to-point vortex induced velocity field (*cf.* subroutine PANPAN*). Currently, VFLOW2D does not calculate the complex potential induced at the point vortices by the vortex panels as it is not directly needed.

It is worth noting that on physical grounds the result obtained from Eq. (102) must be independent of which end is arbitrarily chosen to be "3" and which end "4" in Fig. 29. Consider the γ_D term first. Reversing z_3 and z_4 will reverse the sign of the "ln" factor and also reverse the sign of the fraction preceding it; however, γ_D will itself change sign, as we must reverse γ_3 and γ_4 to remain consistent. The net effect is that the γ_D term will change sign. Similarly, the γ_S term will also change sign because its "ln" does so, while γ_S remains unchanged. Hence, reversing "3" and "4" causes the entire factor in {...} to change sign. This leaves only the leading factor $e^{-i\theta_s}$, which the relation in the footnote below shows will also reverse sign. Thus, the final result is independent of this arbitrary choice.

To summarize the results thus far, it has been shown that: a) the complex potential and velocity fields induced by a single vortex panel, as embodied by Eq. (95) and (97a), asymptotically approach those of a point vortex as the distance from its center approaches infinity; b) when transformed to an arbitrary coordinate system, Eq. (97a) agrees exactly with the expression

* Note that the factor $e^{-i\theta_s}$ can be rewritten as $e^{-i\theta_s} = |z_4 - z_3|/(z_4 - z_3)$, which is how it is programmed in the code.

currently used to compute direct interactions between vortex panels and point vortices on a pair-wise basis. We can now proceed with confidence to develop the multipole expansions needed for the FMM starting from these expressions.

5.2.5 Panel-Centered Origin/Multipole Expansion in Najm, *et al.*, Basis Functions

To obtain the multipole expansion for $F_{\text{panel}}(z)$ about the center of the panel, we expand Eq. (95) in a Taylor series about $z_e/z = 0$:

$$F_{\text{panel}}(z) = 2C_1 z_e \left[\ln z - \frac{1}{6} \left(\frac{z_e}{z} \right)^2 - \frac{1}{20} \left(\frac{z_e}{z} \right)^4 - \frac{1}{42} \left(\frac{z_e}{z} \right)^6 - \frac{1}{72} \left(\frac{z_e}{z} \right)^8 \right] \\ - 2C_2 z_e^2 \left[\frac{1}{3} \left(\frac{z_e}{z} \right) + \frac{1}{15} \left(\frac{z_e}{z} \right)^3 + \frac{1}{35} \left(\frac{z_e}{z} \right)^5 + \frac{1}{63} \left(\frac{z_e}{z} \right)^7 \right] + O\left(\frac{z_e}{z} \right)^9 \dots$$

But from Eq. (93)

$$2C_1 z_e = \frac{-ie^{-i\theta} z_e (\gamma_a + \gamma_b)}{2\pi} = \frac{-i(\gamma_a + \gamma_b) |z_e|}{2\pi} \\ 2C_2 z_e^2 = \frac{-ie^{-i\theta} z_e (\gamma_b - \gamma_a)}{2\pi} = \frac{-i(\gamma_b - \gamma_a) |z_e|}{2\pi}$$

allowing the expansion to be expressed as

$$F_{\text{panel}}(z) = \frac{i(\gamma_a + \gamma_b) |z_e|}{2\pi} \left[-\ln z + \frac{1}{6} \left(\frac{z_e}{z} \right)^2 + \frac{1}{20} \left(\frac{z_e}{z} \right)^4 + \frac{1}{42} \left(\frac{z_e}{z} \right)^6 + \frac{1}{72} \left(\frac{z_e}{z} \right)^8 \right] + \\ \frac{i(\gamma_b - \gamma_a) |z_e|}{2\pi} \left[\frac{1}{3} \left(\frac{z_e}{z} \right) + \frac{1}{15} \left(\frac{z_e}{z} \right)^3 + \frac{1}{35} \left(\frac{z_e}{z} \right)^5 + \frac{1}{63} \left(\frac{z_e}{z} \right)^7 \right] + O\left(\frac{z_e}{z} \right)^9 \dots$$

In the interest of comparing this result to that quoted on p. 82 of the report by Najm, *et al.* [27], we rewrite it as:

$$F_{\text{panel}}(z) = a_0 \ln z + \sum_{k=1}^{\infty} a_k z^{-k} \\ a_0 = \frac{-i(\gamma_b + \gamma_a) |z_e|}{2\pi} \\ a_k = \frac{i(\gamma_b + \gamma_a) |z_e| z_e^k}{2\pi k(k+1)} \quad k > 0, \text{ even} \\ a_k = \frac{i(\gamma_b - \gamma_a) |z_e| z_e^k}{2\pi k(k+2)} \quad k > 0, \text{ odd} \quad (103)$$

Figure 30. Symmetric and Antisymmetric Parts of Linear Strength Variation

Equation (103) differs from the result quoted in Ref. [27] by the constant factor $-i/2\pi$, which can be attributed to differences in how they chose to normalize their source strength. Indeed, were it not for the factor $-i/2\pi$, the zeroth order term in the expansion,

$$\frac{-i(\gamma_a + \gamma_b)|z_e|}{2\pi} \ln z = \frac{-i\Gamma}{2\pi} \ln z,$$

would not yield the correct “point vortex” result (*cf.* §5.2.3).

It is worth noting here that the expansion represented by Eq. (103) can be viewed as the sum of two distinct interlaced expansions. The first, consisting of the zeroth and k -even terms, represents the influence of the “mean” circulation level, $(\gamma_b + \gamma_a)/2$; the other expansion consists of only the k -odd terms, and represents the “variable” portion of the circulation, $(\gamma_b - \gamma_a)/2$. In general, any linear variation can be viewed as the superposition of a constant (symmetric) part and a variable (anti-symmetric) part as sketched in Fig. 30. We will make use of this fact later in §5.2.8 when studying the numerical errors incurred by using the multipole expansions.

5.2.6 Panel-Centered Origin/Multipole Expansion in VFLOW2D Basis Functions

Complex Potential

Equation (103) is an expansion in the set of basis functions $\{\ln z, z^{-m}\}$ as used by Najm, *et al.* [27]. To minimize the changes required in the existing VFLOW2D FMM, it is necessary to rewrite this result in a form analogous to Eq. (7) of Ref. [38], that uses basis functions of the form $K_m(z) = d^m[\ln(z)]/dz^m$. The result is easily shown to be

$$F_{\text{panel}}(z) = \sum_{m=0}^{N_{\text{max}}} A_m K_m(z) \quad (104a)$$

$$K_0(z) = \ln z \quad m = 0$$

$$K_m(z) = -(m-1)! (-z)^{-m} \quad m \geq 1$$

where now

$$\begin{aligned}
 A_0 &= -\frac{i}{2\pi}(\gamma_a + \gamma_b)|z_e| & m = 0 \\
 A_m &= -\frac{i}{2\pi} \frac{(\gamma_b + \gamma_a)|z_e|(-z_e)^m}{(m+1)m!} & m > 0, \text{ even} \\
 A_m &= -\frac{i}{2\pi} \frac{(\gamma_b - \gamma_a)|z_e|(-z_e)^m}{(m+2)m!} & m > 0, \text{ odd}
 \end{aligned}$$

Note that we have changed the summation index from k to m , $\{a_k\}$ to $\{A_m\}$, and truncated the expansion at the $(N_{\max})^{th}$ term to be more consistent with the notation in Ref. [38]. The A_m coefficients can be rewritten as

$$A_m = -\frac{i}{2\pi} \left\{ \frac{\gamma_b + \gamma_a[1 - 2\text{mod}(m, 2)]}{m + 1 + \text{mod}(m, 2)} \right\} \frac{|z_e|(-z_e)^m}{m!} \quad m \geq 0. \quad (104b)$$

This form makes use of the FORTRAN-standard modulus function which returns the (integer) remainder after dividing its first argument by the second. Thus $\text{mod}(m, 2)$ yields 0 if m is even, and 1 if it is odd. Not only does it allow the A_m to be written in a more compact (though admittedly less readable) form, it should also result in more efficient code since it automatically differentiates between m even/odd without the use of IF statements. Equation (104b) is even valid for $m = 0$, since $(-z_e)^0$ and $0!$ are both defined to be one.

Complex Velocity

To obtain a multipole expansion for the velocity, we could expand the closed form result given by Eq. (97b) in a Taylor series about $z_e/z = 0$. But since we already have the expansion for the potential given above, it is easier to apply Eq. (96) directly; i.e., we simply differentiate Eq. (104a) on a term-by-term basis:

$$W_{\text{panel}}(z) = \frac{dF_{\text{panel}}(z)}{dz} = \sum_{m=0}^{N_{\max}} A_m \frac{dK_m(z)}{dz}.$$

But

$$\begin{aligned}
 \frac{dK_m(z)}{dz} &= \begin{cases} z^{-1} & m = 0 \\ (-1)^m m! z^{-(m+1)} & m \geq 1 \end{cases} \\
 &= K_{m+1}(z)
 \end{aligned} \quad (105)$$

Thus, the expansion for the complex velocity is given by

$$W_{\text{panel}}(z) = \sum_{m=0}^{N_{\text{max}}} A_m K_{m+1}(z) , \quad (106)$$

with the coefficients $\{A_m\}$ still defined by Eq. (104b).

At this point the reader may be wondering why the basis functions K_m defined in Eq. (104a) were used in VFLOW2D, rather than the z^{-m} used by Najm, *et al.*, [27], since Eq. (103) appears to be the simpler of the two — for one thing, the latter doesn't require the evaluation of any factorials. However, if Eq. (103) is rewritten as

$$F_{\text{panel}}(z) = \sum_{k=0}^{N_{\text{max}}} a_k P_k(z)$$

$$P_0(z) = \ln z \quad k = 0$$

$$P_k(z) = z^{-k} \quad k \geq 1$$

and then differentiated term-by-term to get $W_{\text{panel}}(z)$, the result is

$$W_{\text{panel}}(z) = \frac{a_0}{z} - \sum_{k=1}^{N_{\text{max}}} k a_k P_{k+1}(z) .$$

This is not as clean as the result in Eq. (106) and involves an additional multiplication. Thus with VFLOW2D's basis functions the expression for $F_{\text{panel}}(z)$ appears more complicated, but because $dK_m(z)/dz = K_{m+1}(z)$, Eq. (106) for $W_{\text{panel}}(z)$ is simplified. The rules by which expansions are "shifted" from one center to another also involve differentiation of the basis functions, and hence they too are simplified by this choice (*cf.* §5.2.9 below). With Najm, *et al.*'s basis functions, the expression for $F_{\text{panel}}(z)$ is simpler, but the expression for $W_{\text{panel}}(z)$, as well as the rules for shifting the expansion centers will be complicated by the fact that $dP_k(z)/dz \neq P_{k+1}(z)$.

5.2.7 Arbitrary Origin/Multipole Expansion in VFLOW2D Basis Functions

The expansions thus far have assumed that both the center of the panel and the center of the expansions coincide with the origin of the coordinate system. This was done to simplify the algebraic manipulations and to facilitate the comparison with the results of Najm, *et al.*, [27] in §5.2.5. However, for coding purposes it is more useful to have the endpoints of the panel and the field point defined relative to an arbitrary origin while retaining the center of the expansions at the panel center. This amounts to applying the same transformation as was used in §5.2.4 to compare closed-form expressions for the panel-induced field.

To simplify the notation somewhat, we introduce the quantities z_S and z_D and rewrite the transformation expressed by Eqs. (100a)–(100b) as

$$z \rightarrow z_1 - \frac{1}{2}(z_4 + z_3) = z - z_S \quad \text{where} \quad z_S \equiv \frac{1}{2}(z_4 + z_3), \quad (107a)$$

$$z_e \rightarrow \frac{1}{2}(z_4 - z_3) = z_D \quad \text{where} \quad z_D \equiv \frac{1}{2}(z_4 - z_3). \quad (107b)$$

That is, after the transformation is applied, we will still use z to refer to the field point but now with respect to the arbitrary rather than panel-centered origin. Also, in the $\{A_m\}$ coefficients we again change $\gamma_a \rightarrow \gamma_3$ and $\gamma_b \rightarrow \gamma_4$ to maintain consistency with the rest of VFLOW2D.

Complex Potential

The expansion for the potential, Eqs. (104a) – (104b), now becomes

$$F_{\text{panel}}(z) = \sum_{m=0}^{N_{\text{max}}} A_m K_m(z - z_S) \quad (108a)$$

$$K_0(z) = \ln z \quad m = 0$$

$$K_m(z) = -(m-1)! (-z)^{-m} \quad m \geq 1$$

with

$$A_m = -\frac{i}{2\pi} \left\{ \frac{\gamma_4 + \gamma_3 [1 - 2 \bmod(m, 2)]}{m + 1 + \bmod(m, 2)} \right\} \frac{|z_D| (-z_D)^m}{m!} \quad m \geq 0. \quad (108b)$$

Complex Velocity

The expansion for the velocity, Eq. (106), now reads

$$W_{\text{panel}}(z) = \sum_{m=0}^{N_{\text{max}}} A_m K_{m+1}(z - z_S), \quad (109)$$

with the $\{A_m\}$ as given by Eq. (108b). Keep in mind that while expressed in terms of global coordinates these expansions are still centered about the midpoint of the panel, z_S . Also recall that at the end of §5.2.4 it was shown how the closed-form expression for the panel-induced velocity would reduce to the same result regardless of which of the panel ends was designated as “3” and which as “4”. The same type of reasoning can be applied to Eq. (108b) to show that the $\{A_m\}$ are independent of this arbitrary choice as well.

To summarize the multipole expansion results thus far, in §5.2.5 our result for the multipole expansion of the field induced by a vortex panel with linearly-varying strength was shown to

Figure 31. Relative Error vs. α at $R = 4$ for Antisymmetric Loading

be in agreement with that derived by Najm, *et al.* [27]. Then by transforming first the basis functions (§5.2.6) and then from a local to a global coordinate system (§5.2.7), we have obtained the needed multipole expansion for a single panel in the same terms used by the FMM in VFLOW2D.

5.2.8 Multipole Expansion vs. Exact Solution

A small test code was written to evaluate the velocity field predicted by Eqs. (108b) and (109) and compare it against the exact analytical solution given by Eq. (102). It was noted in §5.2.5 that any linear variation in γ along the panel can be viewed as the sum of a constant symmetric load plus an anti-symmetric load. A unit-length panel was assumed with $\gamma_3 = \gamma_4 = +1$ for the symmetric case and $\gamma_3 = -1, \gamma_4 = +1$ for the anti-symmetric case. Surveys were performed at fixed radii from the panel center of $R = 1, 4$ and 8 and with α , the azimuthal angle between the plane of the panel and the field point, varied from 0° through 360° . Figure 31 shows the relative error, $|W_{mult} - W_{exact}| / |W_{exact}|$, as a function of α for the anti-symmetric case with $R = 4$ and various values of N_{max} . Plots at the other radii are qualitatively similar. The principal conclusions drawn from the azimuthal surveys are that the error decreases with increasing R , decreases with increasing order of the expansion, N_{max} and, other things being equal, is always less for the symmetric loading case than for the anti-symmetric case.

Although difficult to see at the logarithmic scale used in these plots, each survey also showed that the minimum error always occurs at the “head-on” positions, $\alpha = 90^\circ$ and 270° , regardless of the values of R and N_{max} or whether the load distribution was symmetric or anti-symmetric.

Figure 32. Relative Error vs. R at $\alpha = 0^\circ$ for Antisymmetric Loading

At these locations all portions of the panel are more or less equidistant from the target point, so the field is determined largely by the integrated loading that is well represented by the first few terms of the expansion. Conversely, the maximum error always occurs at the "edge-on" positions, $\alpha = 0^\circ$ and 180° . There the relative differences in distance between the target and each element of the panel are maximized so the details of the loading distribution, that are represented by the higher-order terms of the expansion, become much more important. To see how rapidly this maximum error decays, a radial survey was run with α fixed at 0° and R varying from 1 out to 8 assuming anti-symmetric loading. These results are displayed in Fig. 32. Based on these azimuthal and radial surveys of the error in the multipole expansion, it was concluded that for most engineering applications, expansions of order $N_{max} = 6$ should provide sufficient accuracy. This is the same order previously used in the FMM for point source distributions.

5.2.9 Source Expansion Coefficients

After constructing a source tree about the collection of all vortex panels, the next step in the Greengard-Rokhlin FMM is to compute the source expansion coefficients in each of the boxes. A brief outline of the steps for doing so is given here, with emphasis on how it differs from the situation when only point sources are present.

1. At the finest level, each childless box will contain one or more panels (*cf.* Fig. 33). For each panel in the box, a set of expansion coefficients $\{A_m\}$ as given by Eq. (108b) is first computed.

Figure 33. Shifting Expansion Centers to Center of Childless Box

2. The center of each expansion, currently at z_S (different for each panel), must be shifted to the common center of the childless box at z_C . The rule for shifting the center of a source expansion has already been worked out in Ref. [38], and so we simply quote the result:

$$F_{\text{panel}}(z) = \sum_{n=0}^{N_{\text{max}}} A'_n K_n(z - z_C) \quad (110)$$

$$A'_n = \sum_{p=0}^n A_{n-p} \frac{(z_C - z_S)^p}{p!}$$

This is an obvious extension of the result presented by Strickland & Baty[38] (their Eqs. (13) and (14)) for shifting a single multipole expansion from a single child box to its larger parent box. Here we have added the complication of there being multiple panels, each with its own $\{A_m\}$ and z_S that must be shifted. Let $\{A_{m,j}\}$ and $z_{S,j}$ denote the panel-centered expansion coefficients and geometric center, respectively, of the j^{th} panel in the box. It should be clear that the generalized result for the potential induced by all the panels in a given box, expanded about its center z_C , is given by

$$F_{\text{panel}}(z) = \sum_{n=0}^{N_{\text{max}}} A'_n K_n(z - z_C) \quad (111)$$

$$A'_n = \sum_{j=1}^{N_{\text{panel}}} \sum_{p=0}^n A_{n-p,j} \frac{(z_C - z_{S,j})^p}{p!}$$

where N_{panel} is the number of panels in the box. As before, the expansion for the complex velocity field is simply

$$W_{\text{panel}}(z) = \sum_{n=0}^{N_{\text{max}}} A'_n K_{n+1}(z - z_C). \quad (112)$$

3. Having obtained the expansion coefficients for the childless boxes, shift the coefficients from the centers of these boxes to their parents at the next coarsest level using Eq. (110). When used in this context, z_S and z_C should be interpreted as the centers and $\{A_n\}$ and $\{A'_n\}$ as the coefficients of the child and parent boxes, respectively.
4. Repeat Step 3 recursively for each coarser level in the tree until source coefficients for all the boxes are known.

5.2.10 Field Expansion Coefficients

With the multipole source expansions thus determined in every source box, *local* or *field* expansions can also be developed about the center of each box in the target tree. The expressions for doing so are identical to those already worked out by Strickland & Baty [38], as at this point the algorithm has no way of determining whether a given expansion was generated by point sources or panel sources.

Starting with the coarsest level in the target tree, the local expansion coefficients are computed using Eqs. (18) and (19) of Ref. [38]. The local expansion of each parent at this level is then shifted to a new expansion about the centers of each of its child boxes at the next finer level using Eqs. (23) and (24) of Ref. [38]. This process is repeated recursively for successively finer levels in the target tree until local expansions have been computed for each target box.

For any given target box, the procedures outlined in this and the preceding sections suffice to compute the contribution to the field from source boxes that lie in either the target domain's Box List 2 or Box List 3 (*cf.* §5.1). Box List 1 refers to those source boxes sufficiently close to the target domain that no expansions are justified. For such boxes all interactions between source panels and target points must be computed pair-wise using Eq. (102). Thus, the only situation not already covered pertains to List 4 boxes — those far enough removed from the given target box for local expansions to be valid, but *not* far enough to justify the use of source expansions. By inference, a List 4 source box will be larger than the target box, as shown in Fig. 34.

Each panel in the source box is characterized by its endpoints, z_3 and z_4 , with the field point denoted by z and the center of its target domain by z_C . Our starting point is Eq. (102), with z_I replaced by $z = z_C + \Delta z$:

Figure 34. Schematic of a Target Box and a List 4 Source Box

$$W_{\text{panel}} = \frac{-ie^{-i\theta_s}}{2\pi} \left\{ \gamma_S \ln \left(\frac{z_C - z_3 + \Delta z}{z_C - z_4 + \Delta z} \right) + \gamma_D \left[-2 + \left(\frac{2z_C - z_3 - z_4 + 2\Delta z}{z_4 - z_3} \right) \ln \left(\frac{z_C - z_3 + \Delta z}{z_C - z_4 + \Delta z} \right) \right] \right\}$$

Or, expanding the logarithmic factors,

$$W_{\text{panel}} = \frac{-ie^{-i\theta_s}}{2\pi} \left\{ \gamma_S [\ln(z_C - z_3 + \Delta z) - \ln(z_C - z_4 + \Delta z)] + \gamma_D \left[-2 + \left(\frac{2z_C - z_3 - z_4 + 2\Delta z}{z_4 - z_3} \right) [\ln(z_C - z_3 + \Delta z) - \ln(z_C - z_4 + \Delta z)] \right] \right\} \quad (113)$$

This expression is still exact. The desired field expansion about z_C is obtained by expanding this in a Taylor series about $\Delta z = 0$. To make the algebraic manipulations more manageable, we will treat the terms involving γ_S and γ_D separately, starting with the γ_S terms

$$W_{\text{panel}}(\text{due to } \gamma_S) \equiv W_S = \frac{-ie^{-i\theta_s}}{2\pi} \gamma_S \left[\sum_{n=0}^{N_{\max}} \frac{(\Delta z)^n}{n!} K_n(z_C - z_3 + \Delta z) \right]_{\Delta z=0} - \sum_{n=0}^{N_{\max}} \frac{(\Delta z)^n}{n!} K_n(z_C - z_4 + \Delta z) \Big|_{\Delta z=0}$$

where the $K_n = d^{(n)} \ln z / dz^n$ are still defined by Eq. (104a). This simplifies to

$$W_S = \frac{-ie^{-i\theta_s}}{2\pi} \gamma_S \sum_{n=0}^{N_{\max}} \frac{(z - z_C)^n}{n!} \left[K_n(z_C - z_3) - K_n(z_C - z_4) \right]. \quad (114)$$

Now consider the terms involving γ_D in Eq. (113). As a temporary shorthand, let $D_3 = z_C - z_3$, and $D_4 = z_C - z_4$. Then

$$W_{\text{panel}}(\text{due to } \gamma_D) \equiv W_D = \frac{-ie^{-i\theta_s}}{2\pi} \gamma_D \left\{ -2 + \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4} \right) [\ln(D_3 + \Delta z) - \ln(D_4 + \Delta z)] \right\}$$

For the expansion about $\Delta z = 0$ we must evaluate derivatives of the form

$$\frac{d^{(n)}}{d(\Delta z)^n} \left\{ \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4} \right) [\ln(D_3 + \Delta z) - \ln(D_4 + \Delta z)] \right\},$$

n = 1

$$\begin{aligned}\frac{d\{\dots\}}{d(\Delta z)} &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4}\right) \left[K_1(D_3 + \Delta z) - K_1(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{2}{D_3 - D_4}\right) \left[K_0(D_3 + \Delta z) - K_0(D_4 + \Delta z) \right]\end{aligned}$$

n = 2

$$\begin{aligned}\frac{d^2\{\dots\}}{d(\Delta z)^2} &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4}\right) \left[K_2(D_3 + \Delta z) - K_2(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{2}{D_3 - D_4}\right) \left[K_1(D_3 + \Delta z) - K_1(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{2}{D_3 - D_4}\right) \left[K_1(D_3 + \Delta z) - K_1(D_4 + \Delta z) \right] \\ &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4}\right) \left[K_2(D_3 + \Delta z) - K_2(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{4}{D_3 - D_4}\right) \left[K_1(D_3 + \Delta z) - K_1(D_4 + \Delta z) \right]\end{aligned}$$

n = 3

$$\begin{aligned}\frac{d^3\{\dots\}}{d(\Delta z)^3} &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4}\right) \left[K_3(D_3 + \Delta z) - K_3(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{2}{D_3 - D_4}\right) \left[K_2(D_3 + \Delta z) - K_2(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{4}{D_3 - D_4}\right) \left[K_2(D_3 + \Delta z) - K_2(D_4 + \Delta z) \right] \\ &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4}\right) \left[K_3(D_3 + \Delta z) - K_3(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{6}{D_3 - D_4}\right) \left[K_2(D_3 + \Delta z) - K_2(D_4 + \Delta z) \right]\end{aligned}$$

n = 4

$$\begin{aligned} \frac{d^4\{\dots\}}{d(\Delta z)^4} &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4} \right) \left[K_4(D_3 + \Delta z) - K_4(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{8}{D_3 - D_4} \right) \left[K_3(D_3 + \Delta z) - K_3(D_4 + \Delta z) \right] \end{aligned}$$

By induction, it should be evident that,

$$\begin{aligned} \frac{d^{(n)}\{\dots\}}{d(\Delta z)^n} &= \left(\frac{D_3 + D_4 + 2\Delta z}{D_3 - D_4} \right) \left[K_n(D_3 + \Delta z) - K_n(D_4 + \Delta z) \right] \\ &\quad + \left(\frac{2n}{D_3 - D_4} \right) \left[K_{n-1}(D_3 + \Delta z) - K_{n-1}(D_4 + \Delta z) \right] \quad n \geq 0 \end{aligned}$$

Thus the Taylor series for W_D about $\Delta z = 0$ has the form

$$\begin{aligned} W_D &= \frac{-ie^{-i\theta_s}}{2\pi} \gamma_D \left\{ -2 + \sum_{n=0}^{N_{max}} \frac{(\Delta z)^n}{n!} \left[\frac{d^{(n)}\{\dots\}}{d(\Delta z)^n} \right]_{\Delta z=0} \right\} \\ &= \frac{-ie^{-i\theta_s}}{2\pi} \gamma_D \left\{ -2 + \sum_{n=0}^{N_{max}} \frac{(z - z_C)^n}{n!} \left\{ \left(\frac{2z_C - z_3 - z_4}{z_4 - z_3} \right) \left[K_n(z_C - z_3) - K_n(z_C - z_4) \right] \right. \right. \\ &\quad \left. \left. + \left(\frac{2n}{z_4 - z_3} \right) \left[K_{n-1}(z_C - z_3) - K_{n-1}(z_C - z_4) \right] \right\} \right\} \quad (115) \end{aligned}$$

where we've reinserted $\Delta z = z - z_C$, $D_3 = z_C - z_3$, and $D_4 = z_C - z_4$. Combining Eqs. (114) and (115) gives the full result for W_{panel} ,

$$W_{\text{panel}}(z) = \sum_{n=0}^{N_{max}} B_n \frac{(z - z_C)^n}{n!},$$

with the field expansion coefficients, $\{B_n\}$, given by

$$B_n = \frac{-ie^{-i\theta_s}}{2\pi} \left\{ \gamma_s \left[K_n(z_C - z_3) - K_n(z_C - z_4) \right] \right. \\
+ \gamma_D \left\{ -2\delta_{n0} + \left(\frac{2z_C - z_3 - z_4}{z_4 - z_3} \right) \left[K_n(z_C - z_3) - K_n(z_C - z_4) \right] \right. \\
\left. \left. + \left(\frac{2n}{z_4 - z_3} \right) \left[K_{n-1}(z_C - z_3) - K_{n-1}(z_C - z_4) \right] \right\} \right\}$$

δ_{n0} is the Kronecker delta function, equal to 1 when $n = 0$ and 0 otherwise. This result expresses only the complex velocity induced at z by a single panel in the List 4 source domain in Fig. 34. The full effect of all N_{panel} panels in the source domain is obtained by linear superposition of an expression of this form for each of them. If we thus let $z_3, z_4 \rightarrow z_{3j}, z_{4j}$, $\theta_s \rightarrow \theta_{sj}$, and $\gamma_s, \gamma_D \rightarrow \gamma_{sj}, \gamma_{Dj}$, each represent the corresponding characteristic of the j^{th} panel in the box, and appropriately redefine $\gamma_s, \gamma_D \rightarrow \gamma_{sj}, \gamma_{Dj}$ in Eq. (101), we get finally for the velocity induced by all the panels in a List 4 Source Box,

$$W(z) = \sum_{n=0}^{N_{max}} B_n \frac{(z - z_C)^n}{n!}, \quad (116a)$$

$$B_n = \frac{-i}{2\pi} \sum_{j=1}^{N_{panel}} e^{-i\theta_{sj}} \left\{ \gamma_{sj} \left[K_n(z_C - z_{3j}) - K_n(z_C - z_{4j}) \right] \right. \\
+ \gamma_{Dj} \left\{ -2\delta_{n0} + \left(\frac{2z_C - z_{3j} - z_{4j}}{z_{4j} - z_{3j}} \right) \left[K_n(z_C - z_{3j}) - K_n(z_C - z_{4j}) \right] \right. \\
\left. \left. + \left(\frac{2n}{z_{4j} - z_{3j}} \right) \left[K_{n-1}(z_C - z_{3j}) - K_{n-1}(z_C - z_{4j}) \right] \right\} \right\} \quad (116b)$$

This completes the information needed to compute a complete set of field coefficients for each box in the target tree.

5.2.11 Sample Calculations

Extending the FMM to handle vortex panel sources does not improve the fidelity of the physical model; it is simply a way of (hopefully) speeding up the calculation. The proof of the pudding lies in being able to demonstrate a reduction in run time without significantly altering the results. As a demonstration case, the laminar flow past a stationary two-dimensional cylinder of diameter d was simulated for a Reynolds Number, Re , of 10^4 , using both the original VFLOW2D with panel-to-point vortex interactions computed directly and the modified version using the extended FMM described above. (Both runs used the FMM to compute the interactions between point vortices.) The simulations were run out to a dimensionless time of $\bar{t} = Ut/d = 20$, using a time step $\Delta \bar{t} = 0.05$. As noted earlier, the run using the modified FMM used N_S and N_F as predicted by Eq. (85) and $h = 2$ in Eq. (87b) when generating the panel source tree and $N_{max} = 6$ in the expansions.

Figure 35 compares the time histories of the streamwise and transverse force coefficients, defined by

$$C_x = \frac{F_x}{\frac{1}{2}\rho U^2 d}, \quad C_y = \frac{F_y}{\frac{1}{2}\rho U^2 d}$$

Figure 35. Comparison of Predictions for Flow Past a Cylinder Using FMM with $h = 2$ vs. Direct Panel-to-Point Vortex Interactions

for the two runs. The predictions from the run that used direct interactions are shown as the solid curves, while those using the FMM are the dashed lines. To assure that the vortex shedding from alternate sides of the cylinder was initiated at about the same time in both cases, a perturbation crossflow velocity component of $V = -0.1$ was specified from $\bar{t} = 0$ until $\bar{t} = 1.0$, after which V was identically zero. This is the cause of the sharp downward spike in C_y at $\bar{t} = 1.0$. The two sets of curves first begin to diverge around $\bar{t} \sim 4$, but the differences don't become appreciable until $\bar{t} \sim 10$ or so. Even then, the C_x curves, though noisy, clearly are fluctuating about the same mean curve. The principal difference between the two C_y curves is that the one using FMM develops a positive phase-shift with respect to that using direct interactions. However, the amplitude and frequency of the oscillations in both curves remain comparable. It should be pointed out that C_y oscillates about a mean value of zero as the result of vortices being shed from alternate sides of the cylinder. Thus, it represents the sum total of the effects of hundreds of thousands, even millions, of interactions between individual point vortices and between vortex panels and point vortices. As such, it is very sensitive to the cumulative effects of round-off error. It is believed that the level of agreement displayed in Fig. 35 is as good as can be expected under the circumstances, particularly since accurately predicting the phase of such oscillations is beyond the capability of any currently available method.

The reduction in run time was even more dramatic than had been expected. The run using direct panel-to-point vortex interactions used 170 mins. (2.8 hrs.) of CPU time, while that using the FMM took only 34 mins. (0.57 hrs.) — very nearly a factor of 5 reduction. These figures are for the entire code, not simply the panel-to-point interactions, suggesting that if the latter portion had been timed separately, the drop in that fraction of the overall CPU time would have been even more precipitous. It is also expected that the longer the simulation time covered in a given run, the greater will be the reduction in CPU time. This is because the number of target points (wake vortices) in the simulation grows with time, so the payoff at later times is ever increasing.

Recall from §5.2.1 that the source tree is constructed in such a way that any vortex panel can extend outside the smallest box in which it resides by at most half its length. In the "worst case" scenario drawn in Fig. 27 for the case $h = 2$, the minimum separation between a target and a panel in a List 3 source box would be two panel-lengths. If h were reduced to 1, some branches of the tree could now extend another level, to boxes half as small as the smallest in Fig. 27. The minimum separation between a target and a panel in a List 3 source box (not necessarily the same as that drawn) would then be only one panel-length. This increases the maximum relative error (*cf.* Fig. 32) in the multipole expansions for $N_{max} = 6$ from approximately 10^{-4} to 6×10^{-3} . The only reason for considering such a change is that adding another level to the tree should increase the fraction of interactions for which multipole expansions can be employed, thus further reducing run time.

To quantify the effect of varying h , the simulation using the extended FMM was repeated with $h = 1$. The results are shown in Fig. 36, where again the direct interaction calculation is used as a benchmark for comparison. As was the case for $h = 2$, both C_x curves clearly oscillate about the same mean, although one could argue that the agreement is not quite as good as before. The changes in the FMM curve for C_y are more noticeable. It begins to diverge from

Figure 36. Comparison of Predictions for Flow Past a Cylinder Using FMM with $h = 1$ vs. Direct Panel-to-Point Vortex Interactions

the direct predictions at $\bar{t} \sim 3.5$, and the differences become appreciable starting at $\bar{t} \sim 6$, *i.e.*, earlier than was the case with $h = 2$. The phase difference between the two calculations is also greater, although the amplitude and frequency remain comparable. However, the CPU time for this run was 33 mins., a reduction of only $\sim 3\%$. For this problem at least, the small reduction in run time hardly seems worth the increased error, and so $h = 2$ is still recommended. A more systematic study would be needed to determine whether this is true in general.

This extension of the Greengard-Rokhlin FMM to the treatment of vortex panels makes VFLOW2D a more practical tool for riser applications, though further reductions in run time are certainly still desirable. A systematic study of the effects of varying N_{max} , h , and N_S is recommended to provide the data on which informed decisions can be made regarding the trade-offs between accuracy and run time. These encouraging results suggest that the extension of FMMs to other higher-order distributions could be a fruitful avenue of research as well.

(this page intentionally left blank)

6 Tube Geometry & Kinematics

6.1 Geometry Modules

As a means of specifying tube geometry, a set of geometry modules have been made available which contain predefined cross sections. The three basic geometries shown in Fig. 37 constitute initial configurations which will be considered. These geometries were chosen so as to be compatible with our objectives of modeling deep-water oil production and drilling risers while using a relatively simple wall layer grid generation algorithm. The circle is specified by its diameter d , the rectangle by the lengths of its sides a and b , and the circle with strakes by its diameter d , the length of the strakes a , their thickness b , and the number of equally spaced strakes n . The position of the circle is given by its center location in terms of a complex number $z_c = x_c + iy_c$. The positions of the rectangle as well as that of the straked cylinder are given in terms of their centers z_c and their angular orientations θ .

The surfaces are generated by specifying a number of segments n_s each of which contains one or more panels placed between n_{jnt} joints. The first joint in the second segment is joined to the last joint in the first segment and so on. If the body is closed, the first joint in the first segment will be joined to the last joint in the last segment. An example of this methodology is depicted in Fig. 38 for a circle with strakes.

We now proceed to develop the surface geometry for each of the predefined tube cross sections. Starting with the circular cross section, it is clearly convenient to use a single segment. The position on the surface may be given by

$$z_s = z_c + \frac{d}{2} e^{i\left(\frac{2s}{d} + \theta\right)} \quad (117)$$

Figure 37. Predefined Tube Cross Sections

Figure 38. Example Geometry Construction

where θ is a reference angle measured counterclockwise from the positive x axis and s is the distance along the surface whose value varies from 0 to π/d . The surface distance s is measured from the reference point in the counterclockwise direction.

We also construct the rectangle using a single segment although we could have used two or four segments about as conveniently. The position on the surface for the rectangle is given by

$$z_s = z_c + e^{i\theta} \begin{cases} -a - ib + s_1 & \text{for side 1} \\ a - ib + is_2 & \text{for side 2} \\ a + ib - s_3 & \text{for side 3} \\ -a + ib - s_4 & \text{for side 4} \end{cases} \quad (118)$$

Here, s_1, s_2, s_3 , and s_4 represent the distances along each side from their respective corners. Increasing values of s_1, s_2, s_3 , and s_4 occur as one moves around the body surface in the counterclockwise direction. The limits on the variables are

$$\begin{aligned} 0 &\leq s_1 \leq a \\ 0 &\leq s_2 \leq b \\ 0 &\leq s_3 \leq a \\ 0 &\leq s_4 \leq b \end{aligned} \quad (119)$$

For the circular cylinder with n strakes, we divide the surface into n segments. The position on the m^{th} segment is given by

$$z_s = z_c + e^{i(\theta + \theta_m)} \begin{cases} \frac{d}{2} e^{-i\theta_B} + s_1 & \text{for strake side 1} \\ \frac{d}{2} + a - i\frac{t}{2} + is_t & \text{for the strake tip} \\ \frac{d}{2} + a + i\frac{t}{2} - s_2 & \text{for strake side 2} \\ \frac{d}{2} e^{i\left(\theta_B + 2\frac{s_c}{d}\right)} & \text{for the circle} \end{cases} \quad (120)$$

Here, θ_m is the angle which the center line of the m^{th} strake makes with respect to the reference angle θ . The angle $\theta_B = \arcsin(t/d)$ is the half angle on the circle associated with the intersection of the strake base. From Eq. (120) it can be noted that this geometry is constructed by defining the first segment and then successively rotating that segment to form the other $n - 1$ segments. Limits on the surface coordinates along each surface are given by

$$\begin{aligned} 0 \leq s_1 &\leq a + \frac{d}{2}(1 - \cos\theta_B) \\ 0 \leq s_2 &\leq a + \frac{d}{2}(1 - \cos\theta_B) \\ 0 \leq s_t &\leq t \\ 0 \leq s_c &\leq d\left(\frac{\pi}{n} - \theta_B\right) \end{aligned} \quad (121)$$

6.2 Prescribed Motion Generator

For purposes of the fluid flow simulation, we need only know the relative motion of the tubes with respect to the undisturbed fluid. This results from the fact that for incompressible flows, the fluid mechanics (including fluid forces on the body) are identical for unsteady flow over a stationary body as opposed to unsteady motion of the body in an otherwise still fluid. In the present work, this allows us to view the fluid as always being stationary for purposes of simplicity. We therefore only need specify the motion of the rigid body. We denote the complex velocity of the body surface by $W_b \equiv u_b - iv_b$. By using the notation of Fig. 39, we can write the complex velocity of the surface in terms of the velocity W_c at some point c fixed in the body and the angular velocity of the body Ω_c as

$$W_b = W_c - i\Omega_c \bar{z}_b, \quad (122)$$

Figure 39. Rigid Body Motion

where $\bar{z}_b$ is the complex conjugate of z_b . The surface velocity W_b is therefore defined by the parameters u_c , v_c , Ω_c , and z_b .

In Table 1 we indicate a set of prescribed motion cases which should be very useful in testing the behavior of the fluid flow solver. We primarily intend to use these for single tube numerical experiments but will also include a couple of multiple tube simulations.

Case	u_c	v_c	Ω_c
1	$-U_\infty$	0	0
2	0	$-U_\infty$	0
3	$-U_\infty$	0	ω_b
4	$-U_\infty$	$A \sin(t/\tau)$	0
5	$-U_\infty + A \sin(t/\tau)$	0	0
6	$-U_\infty$	0	$A \sin(t/\tau)$

Table 1: Prescribed Motions for Single Tubes

6.3 Example/Benchmark Calculations

In the sections that follow, we present starting flow simulations, *i.e.*, one iteration, for the circular cylinder, cylinder with strakes, rectangle, and cylinder arrays.

6.3.1 Circular Cylinder

Figure 40 shows VFLOW2D calculations for a single cylinder. The streamlines in Figure 40a are seen by an observer that is moving with the velocity of the body while the streamlines shown in Fig. 40b are those that would be seen by an observer moving with the fluid.

Figure 40. Streamlines for Starting Flow around Cylinder

Figure 41. Edge Velocity for Starting Flow around Cylinder

Figure 41 shows the excellent agreement between the computed velocity at the boundary layer edge as compared to the exact solution.

6.3.2 Cylinder with Strakes

In Fig. 42 we show the starting flow around a triple-start straked cylinder that is simply translating in Figure 42a, while translating and rotating in Fig. 42b. In Fig. 42b it can be noted that the streamlines do not lie parallel to the strakes at their surfaces. This is due to the fact that these streamlines are not being viewed in a body-fixed reference since the reference frame is translating but not rotating with the body.

Figure 42. Starting Flow Around Straked Cylinder

Figure 43. Starting Flow Around Rectangles @ 22.5° Angle of Attack

6.3.3 Rectangular Cylinder

In Fig. 43, rectangles with two different aspect ratios are shown. We have demonstrated that rectangles with aspect ratios as high as 1000:1 still yield consistent results for the starting flow over flat plates.

6.3.4 Array of Circular Cylinders

In Fig. 44 we have computed the starting flow over arrays of circular cylinders that are all moving at the same velocity. We have been able to move the cylinders very close to one

Figure 44. Starting Flow Around Tube Bundles

Figure 45. Two Cylinders Moving Relative to Each Other

another where the gap between cylinders is on the order of one thousandth of a diameter and still calculate reasonable results. As the vorticity diffuses away from the surface the relatively high velocities in such narrow gaps will rapidly diminish.

In Fig. 45 calculations for the starting flow for two cylinders which have motion relative to each other are shown. Streamlines are observed in a fluid-fixed reference frame. In Fig. 45a both cylinders are moving relative to the fluid but in opposite directions. In Fig. 45b one cylinder is fixed with respect to the fluid while the other is moving with respect to the fluid.

(this page intentionally left blank)

7 Hydrodynamic Loads

7.1 Pressure Distributions

The pressure gradient along the wall of a body may be obtained by evaluating the Navier-Stokes equation at the wall:

$$\nabla p = \mu \nabla^2 \vec{u} - \rho \frac{d\vec{u}}{dt} = -\mu(\nabla \times \vec{\omega})_w - \rho \vec{a}_w, \quad (123)$$

where $\vec{a}_w$ is the acceleration of the wall and the w subscript on the quantity $\nabla \times \vec{\omega}$ indicates that it is to be evaluated at the wall. Denoting the coordinate that is tangent to the wall by s and the coordinate normal to the wall as n , Eq. (123) may be written as follows for the s direction,

$$\frac{\partial p}{\partial s} = -\mu \left. \frac{\partial \omega}{\partial n} \right|_w - \rho a_s. \quad (124)$$

Here, the coordinates s and n form a right hand system. For a left hand system, the sign on the first term on the right hand side is positive. For convenience, we repeat Eq. (26) as

$$\left. \frac{\partial \omega}{\partial n} \right|_w = -\frac{1}{v} \left. \frac{\partial \gamma}{\partial t} \right|_w, \quad (125)$$

where $(\partial \gamma / \partial t)_w$ represents the time rate of production of circulation per unit length of wall. Combining Eqs. (124) and (125) we obtain

$$\frac{\partial p}{\partial s} = \rho \left. \frac{\partial \gamma}{\partial t} \right|_w - \rho a_s, \quad (126)$$

and for a left hand coordinate system

$$\frac{\partial p}{\partial s} = -\rho \left. \frac{\partial \gamma}{\partial t} \right|_w - \rho a_s. \quad (127)$$

Referring to Fig. 46, we note that the pressure gradient on the top + and bottom - surfaces are calculated using right hand and left hand coordinate systems respectively:

Figure 46. Surface Pressure Calculation

$$\frac{\partial p^+}{\partial s} = \rho \frac{\partial}{\partial t}(u_s - u_s^+) - \rho a_s, \quad \frac{\partial p^-}{\partial s} = -\rho \frac{\partial}{\partial t}(u_s^- - u_s) - \rho a_s. \quad (128)$$

For cases where the pressure jump across the surface sheet is of interest, the gradient of the jump along the surface is given by

$$\frac{\partial}{\partial s}(p^- - p^+) = \rho \frac{\partial}{\partial t}(u_s^- - u_s^+). \quad (129)$$

A pressure coefficient for the pressure jump across the body surface sheet segment can be defined by

$$\Delta C_p \equiv \frac{p^- - p^+}{\frac{1}{2} \rho U_o^2}, \quad (130)$$

where U_o is an arbitrary reference velocity. Defining $\gamma_s \equiv u_s^- - u_s^+$ and integrating from one end of the sheet segment to a position s on the sheet yields

$$\Delta C_p \equiv \Delta C_{p0} + \frac{2}{U_o^2} \frac{\partial}{\partial t} \left(\int_0^s \gamma_s ds \right). \quad (131)$$

For open bodies, the value of ΔC_p is equal to zero at the edges which means that $\Delta C_{p0} = 0$.

If the body is closed, the value of ΔC_{p0} is arbitrary since it represents the pressure jump between the inside and outside of the shell at some point. It should also be noted that for closed bodies, Eq. (131) yields the pressure jump for both the interior and exterior fluid. If only the pressure distribution due to the exterior fluid is desired, then Eq. (126) may be integrated along the surface to yield

$$C_p^+ \equiv C_{p0}^+ + \frac{2}{U_o^2} \frac{\partial}{\partial t} \left(\int_0^s \gamma_s ds \right) - \frac{2}{U_o^2} \int_0^s a_s ds, \quad (132)$$

where $\gamma_s = u_s - u_s^+$ and the acceleration of the surface a_s is in the direction of s .

7.2 Integrated Loads

The loads on each surface sheet segment can be obtained by integrating the local x or y force components along the segment from one end to the other. The moments on each surface segment are likewise obtained by integrating the products of the local force components and the appropriate distances from some reference point along the segment from one end to the other.

$$\begin{aligned} F_x &= - \int_0^l (p^- - p^+) \sin \alpha ds, \\ F_y &= \int_0^l (p^- - p^+) \cos \alpha ds, \\ M_o &= \int_0^l (p^- - p^+) (x_s \cos \alpha - y_s \sin \alpha) ds. \end{aligned} \quad (133)$$

Force and moment coefficients can be defined as

$$C_x = \frac{F_x}{\frac{1}{2} \rho L_o U_o^2}, \quad C_y = \frac{F_y}{\frac{1}{2} \rho L_o U_o^2}, \quad C_M = \frac{M_o}{\frac{1}{2} \rho L_o^2 U_o^2}, \quad (134)$$

where L_o is an arbitrary length scale. In terms of the pressure coefficient ΔC_p , C_x , C_y , and C_M can be written as

$$\begin{aligned} C_x &= - \int_0^{l^+} \Delta C_p \sin \alpha ds^+, \\ C_y &= \int_0^{l^+} \Delta C_p \cos \alpha ds^+, \\ C_M &= \int_0^{l^+} \Delta C_p (x_s^+ \cos \alpha - y_s^+ \sin \alpha) ds^+, \end{aligned} \quad (135)$$

where $l^+ = l/L_o$, $s^+ = s/L_o$, $x_s^+ = x/L_o$, and $y_s^+ = y/L_o$. If only the external force is desired then ΔC_p is replaced by C_p^+ in the above equation.

An alternative way of expressing the effects of moments on the body is to calculate the point x_c^+, y_c^+ about which there is no moment due to the fluid forces:

$$x_c^+ = \frac{1}{C_y} \int_0^r \Delta C_p x_s^+ \cos \alpha ds^+,$$

$$y_c^+ = -\frac{1}{C_x} \int_0^r \Delta C_p y_s^+ \sin \alpha ds^+.$$
(136)

7.3 Added Mass

Because the surface pressures and integrated forces and moments discussed in the previous two sections are obtained from an unsteady fluid calculation that accounts for the effects of body acceleration, the fluid forces associated with added mass (also called apparent mass and virtual mass) are included by the above equations and do not need to be explicitly included.

8 Results

As part of our code verification and validation efforts, we compared the code predictions with experimental measurements. Several experimental data sets had been acquired for this purpose. However, at the time of project termination, the only comparison that had been completed using the latest version of the code was steady flow over a stationary circular cylinder. The quantities used for comparisons are the steady-state drag coefficient and the boundary layer separation point. The results of those comparisons are shown below.

8.1 Simulation Parameters

Flow past an immersed circular cylinder was calculated over the Reynolds number range $1 \times 10^3 \leq Re \leq 1 \times 10^6$. This canonical flow condition was selected because experimental results are known and can be compared with simulation results. Examination of the cylinder boundary layer velocity profiles presented by Schlichting [34] shows that for Reynolds numbers of $O(10^6)$ the boundary layer thickness is of $O(10^{-3} D)$, where D is the cylinder diameter. To perform accurate simulations for these conditions, the minimum boundary layer resolution must be at least an order of magnitude smaller.

For our cylinder flowfield calculations, we set the resolved length normal to the body, l_n , equal to $l_n = y/D = 0.2 \times (2Re)^{-1/2}$, where y is the length measured normal to the body surface. Parametric simulations conducted over the Reynolds number range of interest indicated that no significant grid dependence existed at this resolution. The resolution in the wall layer normal to the body varies as the reciprocal of the square root of the Reynolds number which is expected as the thickness of a steady attached boundary layer is known to vary as $(Re)^{-1/2}$ [2].

The required resolved wall layer length tangent to the body, l_t , is also expected to be a function of Reynolds number. Preliminary parametric simulations at a Reynolds number of 1×10^6 indicate that a tangential resolution of approximately 1° , $l_t = x/D = 0.00873$, where x is the length measure tangent to the body, predicts the separation point with high accuracy and with no significant grid dependence. The tangential resolutions used in the wall layer for this set of runs were: 2° , $l_t = 0.0175$, at $Re \leq 1 \times 10^4$; 1.5° , $l_t = 0.0131$, at $Re = 1 \times 10^5$; and 1° , $l_t = 0.00873$, at $Re = 1 \times 10^6$.

8.2 Body Forces

Figures 47 and 48 show an example of the streamwise (C_x) and lateral (C_y) force coefficients from these calculations. Both curves show the effects of wake unsteadiness. To obtain an average drag coefficient, we took a time average of the streamwise force coefficients over the last half of the simulation ($5 \leq t \leq 10$). Figure 48 shows that the von Karman vortex street forms at approximately $t = 5$. It also shows that the mean lateral force is displaced upward, probably due to the manner in which the asymmetric shedding was started by having a lateral velocity for $t \leq 1$. The wake is still displaced laterally and has not yet returned to a zero mean lateral displacement. A sinusoidal curve fit of the lateral force coefficient in Fig. 48 over the last half of the simulation ($5 \leq t \leq 10$) gives a lateral offset of 0.54 from zero, a sinusoidal amplitude of 0.59, and Strouhal number of 0.29.

In Fig. 49, drag coefficients from our simulations are compared with experimental data of Schlichting [35] and Blevins [3]. Close agreement between the predicted and experimental results is seen. The "error" bars on the calculated results represent the approximate variations in the drag coefficient over time in the simulation. These variations are due to physical wake fluctuations and numerical discretization of the wake vorticity. Using strictly a laminar model at the resolutions given above, the simulations are seen to approximate the sudden drop in drag coefficient during the transition from a laminar to a turbulent boundary layer at approximately $Re = 5 \times 10^5$. The laminar model underpredicts the drag coefficient reduction (over pre-

Figure 47. Typical Computed Streamwise Force Coefficient

Figure 48. Typical Calculated Lateral Force Coefficient

Figure 49. Comparison of Drag Coefficients for Steady Flow Over a Stationary Circular Cylinder.

dicts the actual drag coefficient) in comparison with the experimental results. Application of a turbulence model (the Smagorinsky turbulence model with a Smagorinsky coefficient value of $C_s = 0.9$) is seen to overpredict the drag coefficient drop (underpredicts the actual drag coefficient). The turbulence model is discussed in more detail in Section 9.2.2 on page 106.

Although the drag coefficients in Fig. 49 are close to the experimental values, they consistently overpredict the drag. This is consistent with the work of Mittal and Balachandar [25, 26] in which they analyzed the results of both 2-D and 3-D direct numerical simulations for flow of 2-D elliptical and circular cylinders. They observed that there were substantial differences between the forces (both streamwise and lateral) computed from 2-D and 3-D simulations (2-D simulations overpredict the forces), and that these differences were due to the 2-D simulations overpredicting the in-plane Reynolds stresses. They also observed that these differences are more significant for bluffer cylinders. Comparison of our lateral force coefficients with experimental data, e.g., Gopalkrishnan [13], shows that they are also overpredicted. Mittal and Balachandar do not discuss Strouhal number, so it is unknown whether our higher-than-normal Strouhal numbers are due to this effect or not.

8.3 Boundary Layer Separation Points

The boundary layer separation points were also calculated in the same set of simulations described above. The locations of the separation points are given as the angular orientations along the cylinder surface measured from the upstream stagnation point. In Fig. 50, our simulation results are compared with experimental data given by White [43]. Very good agreement is seen over the laminar boundary layer range $Re \leq 5 \times 10^5$. In the turbulent boundary layer region at $Re = 1 \times 10^6$, the laminar model simulation is seen to significantly underpredict the separation angle. Applying the Smagorinsky turbulence model with a Smagorinsky coefficient value of $C_s = 0.9$ yields a separation angle in close agreement with experimental values. Application of this turbulence model with $C_s = 0.9$ improves the separation angle prediction but also changes the nature of the wake in a non-physical manner as is discussed in Section 9.2.2.

8.4 Wake

Figure 51 shows a computed wake for steady flow over a circular cylinder. Each dot in the wake represents one shed vortex. One can easily see the von Karman vortex street that has developed in the wake.

8.5 Computation Time

The solution time for the example computation described above was approximately 40 hours on a 400 MHz Sparc Ultra. This is obviously too long for a practical engineering tool. Ways to reduce the computation time were being addressed at project termination.

Figure 50. Comparison of Boundary Layer Separation Points, Given as Angular Orientations Measured from the Upstream Stagnation Point.

Figure 51. Wake Behind Circular Cylinder in Steady Flow

(this page intentionally left blank)

9 Code Limitations and Unverified Improvements

This chapter contains discussions of current code limitations and needed improvements. We also discuss new additions to the code that were not fully verified and validated due to the premature termination of the project. They are presented for completeness, but the reader and/or user of the code is cautioned that they need further verification and validation.

9.1 Current Code Limitations

9.1.1 Time Integration

As it currently exists, the code performs simple explicit Euler time integration. We were unable to investigate higher order or implicit integration schemes. Currently, the maximum allowable time step is determined by the remapping scheme. We had begun investigation of different remapping schemes in order to increase the minimum time step, but were unable to complete the investigation. The progress that was made is documented in Section 9.2.1.

9.1.2 Fluid/Structure Coupling

One of our un-accomplished goals was to couple VFLOW2D with a simple structural code in order to investigate the numerical stability of fluid/structure coupling. The surface pressures, forces, and moments are passed through the load common block. The motions of the body surfaces are communicated through the bvel common block. Since we were unable to investigate the coupling issues, it is unclear whether simply passing forces and moments to the structural code and body velocities to the fluids code is sufficient. Other fluid/structure coupling problems where the mass of the affected fluid (added mass) is large relative to the mass of the body (*e.g.*, parachute inflation) require more sophisticated coupling schemes. The coupling problem arises because the fluid forces on the body are a function of body acceleration and the body acceleration is a function of fluid forces. A successful coupling scheme must achieve consistency between forces and accelerations at each time step.

9.1.3 Multiple Bodies

VFLOW2D was constructed to compute the interactions between multiple bodies, as long as they do not physically touch. However, this capability was never verified, except for prescribed body motion.

9.1.4 Non-Cylindrical Body Geometries

Although the body surface generators built into VFLOW2D contain a number of different geometries, all of our investigations used plain cylinders. The effects on non-cylindrical geometries on the code has not been investigated.

9.2 Unverified Code Improvements

9.2.1 Remapping and Accelerated Computation

The computational time step is intrinsically related to the boundary layer resolution. Therefore, this discussion of remapping acceleration begins with a brief discussion of the boundary layer resolution.

The boundary layer resolution is key to both the accuracy and the practicality of computation of unsteady flow about a body. Accurate computation is possible only if the vorticity within the boundary layer is sufficiently well resolved, while practical computation dictates that only the minimum number of computation points be used to resolve the boundary layer. The wall layer cell size is determined by achieving a balance between these two opposing constraints.

Resolution of the boundary layer tangential to the body, $l_t = x/L$, where x is measured tangent to the body surface and L is a reference length, must be sufficient to allow accurate prediction of the separation point from bluff bodies at high Reynolds numbers, Re . Resolution of the boundary layer normal to the body, $l_n = y/L$, where y is measured normal to the body surface, must be sufficient to reproduce the relatively steep velocity gradient within the boundary layer at high Reynolds numbers. Accuracy of the boundary layer resolution was investigated using validation simulations conducted for flow past a canonical cylinder geometry. The reference length is equated to the cylinder diameter, D . Preliminary parametric simulations at a Reynolds number of 1.0×10^6 indicate that boundary layer resolution of approximately 1° ($l_t = x/D = 0.00873$) predicts the separation point with high accuracy and with no significant grid dependence. This resolution requirement is expected to increase with increasing Reynolds number. Required resolution normal to the body is also expected to increase with increasing Reynolds number as the thickness of a steady, attached boundary layer is known to vary as $(Re)^{-1/2}$ [2]. Further parametric simulations over the range $1 \times 10^3 \leq Re \leq 1 \times 10^6$ indicate that no significant grid dependence is present for $l_n = y/D = 0.2 \times (2Re)^{-1/2}$, consistent with the cylinder boundary layer velocity distribution profiles presented by Schlichting [34]. Based on these simulations, it can be seen that the wall layer cell resolution requirements are consistent with boundary layer length scales, $l_n \ll l_t$.

Practical implementation of the described remapping scheme of Dukowicz and Kodis [11] restricts the allowable distortion of the wall layer cells during a simulation time step. As a result, this scheme is subject to a wall layer resolution limitation. In order to achieve the remapping from the wall layer cells, movement of each node during the computation time step

Δt_r is restricted to a characteristic length $l_w = \min(l_p, l_n) = l_n$. Using a non-dimensional reference free-stream velocity $U_f = 1$, the time step is calculated to be

$$\Delta t_r \leq l_n / U_f = 0.2 \times (2Re)^{-1/2}. \quad (137)$$

Computation of diffusion in the wall layer is also subject to a stability limitation on the time step Δt_s , similar to that for the explicit heat conduction equation with centered finite-difference solution [29] that can be written in the form

$$\Delta t_s = 0.5 \times (l_w)^2 Re = 0.01 \quad (138)$$

and is independent of Reynolds number. Hence, for large Reynolds numbers, $Re > 2.0 \times 10^4$, the remapping time restriction is much more limiting than is the stability time restriction, $\Delta t_s \gg \Delta t_r$. For example at $Re = 1.0 \times 10^6$, $\Delta t_s = 70\Delta t_r$.

An alternative remapping scheme that does not restrict the distortion of the wall layer cells during a simulation time step was developed. As a result, this scheme is not subject to the Δt_r computation time step restriction and the inherent time step advantage $\Delta t_s \gg \Delta t_r$ at large Reynolds numbers can be exploited. Conservation of circulation is explicitly enforced in this remapping. At the time of this writing, the alternative scheme was partially implemented. The alternative scheme distributes the circulation within the wall layer cells onto prescribed sub-cells at the start of each time step. Each cell is evenly divided into a relatively large number of sub-cells with centers aligned in columns normal to the body and rows tangent to the body. The centers of each subcell column or row are equally spaced. The subcell center locations are determined relative to the node locations. A schematic of a cell with subcell centers is shown in Fig. 52, showing the cell distortion in a single time step that is allowed with this alternative remapping scheme. The vorticity at each node is bilinearly interpolated onto the centers of the subcells within connected cells. The vorticity within a subcell is held constant at this value.

This process defines a C^0 vorticity distribution function over the wall layer ($C^0(\Omega) \equiv \{f(x) \in F(\Omega) \rightarrow f(x) \text{ that is continuous on the wall layer domain } \Omega\}$) with piecewise smooth first derivatives and then uses the sub-cells to discretize this C^0 function into a piecewise smooth function space. This remapping approach is expected to approach second-order accuracy as the number of subcells becomes sufficiently large. Subcell areas are defined by hypothetical lines equally dividing the distance between neighboring subcell centers. The subcell vorticities and areas are used to calculate the subcell circulation.

The cell nodes are then moved a distance equal to the sum of the local diffusion and convection velocities multiplied by the time step Δt , as described in Chapter 2 on page 19. The new positions of the subcells' centers are determined relative to the new node locations. This is an efficient method of moving each subcell center with a velocity equivalent to that bilinearly

Figure 52. Depiction of a wall layer cell with sub-cell centers at the beginning (left) and at the end (right) of a simulation time step using the alternative remapping scheme.

interpolated from the calculated grid node velocities. The total circulation within each subcell is held constant during this movement. The solution remapping is then completed by summing the subcell circulations associated with the new subcell center locations into the appropriate original orthogonal grid cells.

9.2.2 Turbulence Modeling

A simple turbulence model that is consistent with the wall layer model will be selected from among those currently used by the CFD community. There are several possibilities including the Baldwin-Lomax, Johnson-King, Baldwin-Barth, and Spalart-Allmarus models. A recent book by Wilcox [44] discusses these models as well as other turbulence modeling issues. Lall *et al.* [20] recently evaluated several turbulence models for attached and separated airfoil flows. Simple models for the enhanced turbulent mixing produced by local three-dimensional instabilities in the near wake will also be sought.

Parametric simulations of the Smagorinsky turbulence model and the Baldwin-Lomax turbulence model have been completed. At lower Smagorinsky coefficient values, $C_s < 0.2$, the model introduces sufficient diffusion to mimic low Reynolds number flows with accompanying small separation angles and large drag coefficients. At large C_s values, the correct body forces and separation angles for known canonical cylinder flows can be achieved, but the nature of the wake is greatly altered, reducing the vortex street oscillations. Introduction of the Smagorinsky turbulence model into a simulation in which grid independence has been established has been observed to reintroduce grid dependence. The Baldwin-Lomax model does not include empirically adjustable coefficients. Application of this model yields the artificially low Re appearance.

9.2.3 Transition Criteria

In general, the transition problem is an unsolved one, especially when one considers real-world flows in which the effects of surface roughness, wall curvature, free-stream turbulence, temperature gradients, etc. are present. The code currently has no transition model. We had planned to adopt a simple transition model that is consistent with the wall layer model and that will be selected from among those currently available to the CFD community.

One possible model is one developed by Cebeci and Smith [5] based on a combination of Michel's method and Smith and Gamberoni's e^9 correlation. This correlation connects the momentum thickness Reynolds number at transition, $R_{\theta_{tr}}$, with a Reynolds number, $Re_{x_{tr}}$, that is based on the distance along the surface from the start of the boundary layer. The correlation is given by

$$R_{\theta_{tr}} = 1.174 \left(1 + \frac{22,400}{Re_{x_{tr}}} \right) Re_{x_{tr}}^{0.46} \quad (139)$$

In order to calculate the momentum thickness, the wall layer will have to be sufficiently thick so as to include the region contributing to it. We must also be able to determine the value of the so-called "edge velocity." In addition, this correlation will require the identification of the surface location where the boundary layer begins. Thus even the application of this simple criteria to our wall layer model could prove to be challenging.

9.2.4 Second-Order Time Advance

Each free vortex blob is convected at the local fluid velocity. The distance that a particular vortex moves between $t_0 = t - \Delta t$ and time $t_1 = t$ is currently obtained using the simple Eulerian first-order integration formula

$$\Delta z_c = \bar{W}_c|_{t_0} \Delta t. \quad (140)$$

Our intension was to add a second-order integration formula, such as the second order Adams-Bashforth open integration formula [6]

$$\Delta z_c = \left[\frac{3}{2} \bar{W}_c|_{t_1} - \frac{1}{2} \bar{W}_c|_{t_0} \right] \Delta t. \quad (141)$$

Here, Δz_c is the change in the complex position of the center of the vortex. $W_c|_{t_1}$ and $W_c|_{t_0}$ are the conjugate complex velocities of the vortex center at the current and previous time steps, respectively. A simple Euler equation will still be used for the first time step.

(this page intentionally left blank)

10 Conclusions And Recommendations

We have made significant progress towards development of a vortex-based flow solver that can be used to compute the vortex-induced vibrations on deep ocean risers and umbilicals. Comparisons with known analytical solutions show that the current implementation of the wall layer model yields accurate flowfield solutions within the boundary layer. Implementation of a multipole fast solver for both discrete vortices and vortex panels shows speedup over direct calculations and reduction in total run times while maintaining computational accuracy. Our vortex merging algorithm also provides run time reduction while preserving solution accuracy. Comparison of calculated and experimental drag coefficients and flow separation points for circular cylinders at Reynolds numbers from 10^3 to 10^6 show good agreement below the drag crisis that occurs between $Re = 10^5$ and 10^6 . At a Reynolds number of 10^6 , the agreement is somewhat degraded, but still shows the effects of laminar to turbulent transition.

A number of improvements still need to be made to make VFLOW2D a viable engineering tool. At the present time, run times required to resolve the boundary layer at high Reynolds numbers ($> 10^5$) are too large (approximately 40 hours for the example problem shown in Section 8). Improvements in the wall layer remapping scheme should reduce this time. More work needs to be done select, implement, verify and validate an appropriate turbulence model with this formulation. The code needs to be coupled with a simple structural model for coupled fluid/structure calculations. Finally, the code needs to be fully verified and validated.

(this page intentionally left blank)

References

- [1] Batchelor, G. K., *An Introduction to Fluid Mechanics*, Cambridge University Press, Great Britain, §2.6, 1967.
- [2] *ibid*, §5.8-5.10.
- [3] Blevins, R. D., *Flow-Induced Vibration*, 2nd ed., Van Nostrand Reinhold, New York, p. 213, 1990.
- [4] Carrier, J., Greengard, L., and Rokhlin, V., "A Fast Adaptive Multipole Algorithm for Particle Simulations," *SIAM Journal on Scientific and Statistical Computing*, Vol.9, No.4, pp.669-696, July 1988.
- [5] Cebeci, T., and Smith, A. M. O., *Analysis of Turbulent Boundary Layers*, Academic Press, New York, 1974.
- [6] Chapra, S. C., and Canale, R. P., "Numerical Methods For Engineers With Personal Computer Applications," p. 526, McGraw-Hill, 1985.
- [7] Chorin, A. J., "Numerical Study of Slightly Viscous Flows," *J. Fluid Mech.*, vol. 75, pp. 785-796, 1973.
- [8] Chou, M., "Simulation of Slightly Viscous External Flow by a Grid-Particle Domain Decomposition Method," *Computers & Fluids*, Vol. 24, No. 3, pp. 333-347, 1995.
- [9] Currie, I. G., *Fundamental Mechanics of Fluids*, McGraw-Hill, p. 46, 1974.
- [10] *ibid.*, pp. 231-237.
- [11] Dukowicz, J. K., and Kodis, J. W., "Accurate Conservative Remapping (Rezoning) for Arbitrary Lagrangian-Eulerian Computations," *SIAM J. Sci. Stat. Comput.*, Vol. 8, No. 3, May 1987, pp. 305-320.
- [12] Fullerton, T. L., Strickland, J. H., and Sundberg, W. D., "A Vortex Panel method for Calculating Aircraft Downwash on Parachute Trajectories," AIAA paper 91-0875, presented at the AIAA 11th Aerodynamic Decelerator Systems Technology Conference, April 1991.
- [13] Gopalkrishnan, R., "Vortex-Induced Forces on Oscillating Bluff Cylinders," D.Sc. Dissertation, Massachusetts institute of Technology, February 1993.
- [14] Greengard, L. and Rokhlin, V. "A Fast Algorithm for Particle Simulations," *J. Comput. Phys.*, Vol. 73, pp.325-348, 1987.
- [15] Karamcheti, K., *Principles of Ideal-Fluid Aerodynamics*, John Wiley and Sons, New York, pp. 526-530, 1966.
- [16] Kempka, S. N., Glass, M. W., Perry, J. S., and Strickland, J. H., "Accuracy Considerations for Implementing Velocity Boundary Conditions in Vorticity Formulations," Sandia National Laboratory Report SAND96-0583, March 1996.

- [17] Kempka, S. N., and Strickland, J. H., "A Method to Simulate Viscous Diffusion of Vorticity by Convective Transport of Vortices at a Non-Solenoidal Velocity," SAND93-1763, Sandia National Laboratories, Aug. 1993.
- [18] Koumoutsakos, P. D., "Direct Numerical Simulations of Unsteady Separated Flows Using Vortex Methods," Ph.D. Dissertation, California Institute of Technology, 1993.
- [19] Koumoutsakos, P., Leonard, A., and Pepin, F., "Boundary Conditions for Viscous Vortex Methods," *J. Comput. Phy.*, vol. 113, pp. 52-61, 1994.
- [20] Lall, V., Papadakis, M., and Hoffman, K., "Evaluation of Turbulence Models for Attached and Separated Airfoil Flows," AIAA paper 95-0052, presented at the 33rd Aerospace Sciences Meeting and Exhibit, January, 1995.
- [21] Leonard, A., "Vortex Methods for Flow Simulation," *J. Comput. Phy.*, vol. 37, pp. 289-335, 1980.
- [22] Lin, H., Vezza, M., and Galbriath, R. A. McD., "Discrete Vortex Method for Simulating Unsteady Flow Around Pitching Airfoils," *AIAA Journal*, Vol. 35, No. 3, pp. 494-499, March 1997.
- [23] Marshall, J. S., and Grant, J. R., "A Method for Determining the Velocity Induced by Highly Anisotropic Vorticity Blobs," *J. Comput. Phy.*, 126, pp. 286-298, 1996.
- [24] Meriam, J. L., *Dynamics*, John Wiley & Sons, New York, pp. 228-231, 1975.
- [25] Mittal, R., and Balachandar, S., "Vortical Structures in Bluff Body Wakes," AIAA 95-0867, 33rd Aerospace Sciences Meeting, January, 1995.
- [26] Mittal, R., and Balachandar, S., "Effect of Three-Dimensionality on the Lift and Drag of Nominally Two-Dimensional Cylinders," *Physics of Fluids*, Vol. 7, No. 8, August 1995.
- [27] Najm, H. N., Schefer, R. W., Milne, C. J., Devine, K. D., and Kempka, S. N., *Numerical and Experimental Investigation of Vortical Flow-Flame Interaction*, SAND98-8232, Sandia National Laboratories, February 1998.
- [28] Panton, R., "The Transient for Stokes's Oscillating Plate: A Solution in Terms of Tabulated Functions," *J. Fluid Mech.*, vol. 31, pp. 819-825, 1968.
- [29] Peyret, R., Taylor, T. D., *Computational Methods for Fluid Flow*, Springer-Verlag, New York, p. 40, 1983.
- [30] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN*, 2nd ed., Cambridge University Press, New York, 1992.
- [31] Rokhlin, V., "Rapid Solution of Integral Equations of Classical Potential Theory," *J. Comput. Phy.*, Vol. 60, pp. 187-207, 1983.
- [32] Sarpkaya, T., "Vortex Element Methods for Flow Simulation," *Advances in Applied Mechanics*, Vol. 31, pp. 113-247, 1994.
- [33] Schlichting, H., *Boundary Layer Theory*, McGraw-Hill, 6th ed., 1968, pp. 125-133.
- [34] *ibid.*, pp. 158-162.
- [35] *ibid.*, p. 17.

- [36] Shih, C., Lourenco, L., Van Dommelen, L., Krothapalli, A., "Unsteady Flow Past an Airfoil Pitching at a Constant Rate," *AIAA Journal*, Vol. 30, No 5, pp. 1153-1161, May 1992.
- [37] Strickland, J. H., "A Prediction Method For Unsteady Axisymmetric Flow Over Parachutes," AIAA paper 93-1217, presented at the RAeS/AIAA 12th Aerodynamic Decelerator Systems Technology Conference, London, England, May10-13, 1993. Also, *AIAA Journal of Aircraft*, Vol. 31, No. 3, pp. 637-643, May-June 1994.
- [38] Strickland, J. H., and Baty, R. S., "A Two-Dimensional Fast Solver for Arbitrary Vortex Distributions," Sandia National Laboratories Report SAND97-0880, April 1997.
- [39] Strickland, J. H., and Baty, R. S., "Modification of the Carrier, Greengard, and Rokhlin FMM for Independent Source and Target Fields," *J. Comput. Phy.*, Vol. 142, pp. 123-128, 1998.
- [40] Strickland, J. H., and Kempka, S. N., "Modeling Wake Evolution Behind a Submarine: Two Dimensional Maneuvers," Sandia National Laboratories Report SAND93-0040, April 1993.
- [41] Strickland, J. H., Webster, B. T., and Nguyen, T., "A Vortex Model of the Darrieus Turbine: An Analytical and Experimental Study," *Journal of Fluids Engineering*, Vol. 101, No. 4, pp. 500-505, 1979.
- [42] Touryan, K. J., Strickland, J. H. and Berg, D. E., "Electric Power From Vertical-Axis Wind Turbines (A Survey)," *AIAA Journal of Propulsion and Power*, Vol. 3, No. 6, pp. 481-493, Nov-Dec 1987.
- [43] White, F. M., *Fluid Mechanics*, 2nd ed., McGraw-Hill, Inc., New York, p. 415, 1986.
- [44] Wilcox, D. C., *Turbulence Modeling for CFD*, Griffin Printing, Glendale CA, 1993.
- [45] Winckelmans, G. S., and Leonard, A., "Contributions to Vortex Particle Methods for the Computation of Three-Dimensional Incompressible Unsteady Flows," *J. Comput. Phys.*, No. 109, pp. 247-273, 1993.
- [46] Wu, J. Z., Wu, X. H., Ma, H. Y., Wu, J. M., "Dynamic Vorticity Condition: Theoretical Analysis and Numerical Implementation," *International Journal for Numerical Methods in Fluids*, vol. 19, pp. 905-938, 1994.

(this page intentionally left blank)

Appendix A: Users Manual

A.1 Driver Program

VFLOW2D is constructed as a subroutine to be called by a driver program. The driver program presented here and contained with the initial source code distribution is test_vflow2d. The current test_vflow2d is contained below.

Example Driver Program

```
program test_vflow2d

c-----This code predicts the unsteady flow around a set of two-dimensional
c rigid closed bodies which are undergoing translational and or
c rotational motion in an otherwise still fluid.

c UNDER CONSTRUCTION!!!!!!!!!!

implicit none
include 'pdim.h'
include 'fblob.h'
include 'initm.h'
include 'tblob.h'
include 'vort.h'

integer inb, nts, ntf, nt, nnprnt, jrun, min_subs,
& t_model
real*8 delt, t, xnu, U0, XL0, min_sbox_fac,
& gratio, yratio, y2, y3,
& min_nfv, sepn_dist, body_dist
logical perturb_vel

c-----Call subroutine initial to specify number of bodies, geometry type and
c-----dimensions, initial location of each body, etc.

nnprnt = 0

call initial(nt, nts, ntf, t, delt, inb, xnu, U0, XL0,
$ min_sbox_fac, min_subs,
$ min_nfv, sepn_dist, body_dist,
$ t_model, gratio, yratio, y2, y3,
$ perturb_vel)

c-----Loop through time steps starting at nts and ending at ntf.
do 10 nt=nts,ntf
  if(perturb_vel) then
    if(t.gt. 1.d0) then
      av(1) = 0.d0
```

```

        else
            av(1) = -0.1d0
        end if
    end if

    write(*,*)'time cycle no. ',nnprnt
    call vflow2d(nt, nts, ntf, t, delt, inb, xnu, U0, XL0,
$           min_sbox_fac, min_nfv, body_dist, sepn_dist,
$           min_subs, t_model, gratio, yratio, y2, y3)

    nnprnt = nnprnt + 1
    t=t+delt

    open (unit=999, file='run.dat', status='unknown')
    read (999,*) jrun
ccc    read (999,*) delt
ccc    read (999,*) perturb_vel
ccc    read (999,*) kprnt
ccc    read (999,*) sepn_dist
    close (unit=999)
    if (jrun .ne. 1) then
        write(*,*)'RUN STOPPED BECAUSE run.dat VALUE IS NOT EQUAL TO 1'
        stop
    end if

10    continue

    print *, 'nfv = ', nfv
    print *, 'nfb = ', num_fblobs
    print *, 'ntv = ', num_tblobs

    stop
end

```

A.2 Initial Data Input

All initial data required for a simulation are currently input into the code through subroutine `initial`, that is called once by the driving program (see section A.1 on page 115). An example input is contained below. Discussion of the input variables is presented in section A.3 on page 124.

Example Subroutine `initial`

```

subroutine initial(nt,nts,ntf,t,delt,iinb,xnu,U0,XL0,
$           min_sbox_fac, min_subs,
$           min_nfv, sepn_dist, body_dist,
$           t_model, gratio, yratio, y2, y3,
$           perturb_vel)
-----
C    ... THIS SUBROUTINE SPECIFIES ALL SIMULATION INPUT VARIABLES
C    called by:  test_vflow2d

```

C external calls to: new_vid

C-----

```
implicit none
include 'common.h'
include 'initg.h'
include 'initm.h'
include 'vort.h'
```

C ... OUTPUT VARIABLES
integer nt, nts, ntf, iinb, t_model
real*8 t, delt, xnu, U0, XL0, min_sbox_fac
real*8 min_nfv, sepn_dist, body_dist, gratio, yratio, y2, y3

C ... LOCAL VARIABLES
integer ibn, min_subs
real*8 pi, reynolds_no

C ... LOGICAL VARIABLES
logical perturb_vortex, perturb_vel, merge

C ... FUNCTIONS
integer new_vid
external new_vid

C ... CONSTANTS
pi=4.d0*atan(1.d0)
c dynamic viscosity of salt water @70 F in ft^2/sec:
c xnu = 1.10568d-5

C-----

C ***** REFERENCE LENGTH AND TIME SCALES *****

```
reynolds_no = 1.d5!REYNOLDS NUMBER
U0 = 1.d0!REFERENCE VELOCITY
XL0 = 1.d0!REFERENCE LENGTH SCALE
nts = 1!TIME STEP LOOP STARTING INDEX
ntf = 1000 !TIME STEP LOOP FINAL INDEX
delt = 0.01d0 !INVISCID TIME STEP
min_subs = 4 !MINIMUM # OF VISCOUS SUBSTEPS PER TIME STEP
t = 0.d0 !INITIAL TIME
xnu = 1.d0/reynolds_no !DIMENSIONLESS VISCOSITY
min_sbox_fac = 2.d0 !(MIN. SIZE SOURCE BOX)/(LONGEST VORTEX PANEL)
!SHOULD BE .GE. 1.d0
```

C-----

C ***** SPECIFY BODY VARIABLES *****

c 1. CIRCLE: diameter (d), center (zc), reference point on circle is at
c angle measured clockwise from horizontal x axis (theto).

c 2. RECTANGLE: width (a), thickness (b), center (zc), clockwise angle from
c x axis to side "a" (theto), reference point (zs) is at corner designated
c by zs=zc+0.5*exp(i*theto)*(-a-i*b)

c 3. CIRCLE WITH STRAKES: diameter (d), center (zc), number of strakes

```

c      (nst), height of strakes (a), reference point is at base of
c      first strake which
c      is at an angle measured clockwise from horizontal x axis (theto).
c      Available prescribed motions are of the form:
c      Velocity of the body center (zc) in the x direction:
c      Uc=Au+Bu*sin(t*fu+Cu)
c      Velocity of the body center (zc) in the y direction:
c      Vc=Av+Bv*sin(t*fv+Cv)
c      Rotational velocity of the of the body: Om=Ao+Bo*sin(t*fo+Co)

c      4. NACA(xxxx) AIRFOIL: chord (c), position of leading edge (zc),
c      reference
c      point is at the leading edge. The angle (theto) represents the angle of
c      a straight line connecting the leading edge with the trailing edge with
c      respect to the horizontal x axis. The NACA(xxxx) designator is given by:
c      First x: maximum mean line ordinate in percent of chord.
c      Second x: distance from leading edge to max camber in tenths of chord.
c      Third & fourth x's: Airfoil thickness in percent of chord.

c      BODY CENTER VELOCITY VARIABLES:
c      Body translational and rotational velocities calculated in bodvel.f as
c      uc=au(ibn)+bu(ibn)*sin(t*fu(ibn)+cu(ibn))
c      vc=av(ibn)+bv(ibn)*sin(t*fv(ibn)+cv(ibn))
c      wc(i)=dcmplx(uc,-vc)
c      om(i)=ao(i)+bo(i)*sin(t*fo(i)+co(i))

      inb = 1!TOTAL # OF BODIES .le. nb in pdim.h

      do ibn = 1,inb

C      ... BODY #1 GEOMETRY .....

      if (ibn .eq. 1) then
      itb(ibn) = 1!SELECT BODY TYPE FROM LIST ABOVE

      if
      $(itb(ibn) .eq. 1) then !CIRCLE
      d(ibn)=1.d0!CIRCLE DIAMETER
      zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
      theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
      ngpts(ibn)=270 !# OF PANELS IN DISCRETE CIRCLE
      nst(ibn)=0 !NO STRAKES -- DO NOT CHANGE

      else if
      $(itb(ibn) .eq. 2) then!RECTANGLE
      a(ibn)=1.d0!RECTANGLE WIDTH
      b(ibn)=1.d0!RECTANGLE THICKNESS
      zc(ibn)=dcmplx(0,0)!RECTANGLE CENTER
      theto(ibn)=0.d0*pi/4.d0!RECTANGLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS

      else if
      $(itb(ibn) .eq. 3) then!CIRCLE WITH STRAKES
      d(ibn)=1.d0!CIRCLE DIAMETER
      zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
      theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
      ngpts(ibn)=180 !# OF PANELS IN DISCRETE CIRCLE
      nst(ibn)=3 !# OF STRAKES
      a(ibn)=0.25d0!HEIGHT OF STRAKES
      hs(ibn)=.1d0!THICKNESS OF STRAKES

```

```

    else if
$(itb(ibn) .eq. 4) then!NACA AIRFOIL
  n4des(ibn)=0021      !DESIGNATE (NACA(0021) or NACA(9420)
  c(ibn)=1.d0!NACA AIRFOIL CHORD LENGTH
  zc(ibn)=dcmplx(0,0)!NACA AIRFOIL LEADING EDGE
  theto(ibn)=0.d0*pi/4.d0!NACA AIRFOIL REF. PT. MEAS. C.W. FROM X-AXIS
  end if      !for 'if (itb(ibn) .eq. 1)'

C    ... BODY #1 MOTION

  au(ibn)= -1.d0 !SEE COMMENTS ABOVE
  bu(ibn)=  0.d0
  cu(ibn)=  0.d0
  fu(ibn)=  0.d0
  av(ibn)=  0.d0
  bv(ibn)=  0.d0
  cv(ibn)=  0.d0
  fv(ibn)=  0.d0
  ao(ibn)=  0.d0
  bo(ibn)=  0.d0
  co(ibn)=  0.d0
  fo(ibn)=  0.d0

C    ... BODY #2 GEOMETRY .....

    else if      !for 'if (ibn .eq. 1)'
$(ibn .eq. 2) then
  itb(ibn) = 1!SELECT BODY TYPE FROM LIST ABOVE

  if
$(itb(ibn) .eq. 1) then !CIRCLE
    d(ibn)=1.d0!CIRCLE DIAMETER
    zc(ibn)=dcmplx(0,0)      !CIRCLE CENTER
    theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
    ngpts(ibn)=180          !# OF PANELS IN DISCRETE CIRCLE
    nst(ibn)=0              !NO STRAKES -- DO NOT CHANGE

  else if
$(itb(ibn) .eq. 2) then!RECTANGLE
    a(ibn)=1.d0!RECTANGLE WIDTH
    b(ibn)=1.d0!RECTANGLE THICKNESS
    zc(ibn)=dcmplx(0,0)      !RECTANGLE CENTER
    theto(ibn)=0.d0*pi/4.d0!RECTANGLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS

  else if
$(itb(ibn) .eq. 3) then!CIRCLE WITH STRAKES
    d(ibn)=1.d0!CIRCLE DIAMETER
    zc(ibn)=dcmplx(0,0)      !CIRCLE CENTER
    theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
    ngpts(ibn)=180          !# OF PANELS IN DISCRETE CIRCLE
    nst(ibn)=3              !# OF STRAKES
    a(ibn)=0.25d0!HEIGHT OF STRAKES
    hs(ibn)=.1d0!THICKNESS OF STRAKES

  else if
$(itb(ibn) .eq. 4) then!NACA AIRFOIL
    n4des(ibn)=0021      !DESIGNATE (NACA(0021) or NACA(9420)
    c(ibn)=1.d0!NACA AIRFOIL CHORD LENGTH

```



```

zc(ibn)=dcmplx(0,0)!NACA AIRFOIL LEADING EDGE
theto(ibn)=0.d0*pi/4.d0!NACA AIRFOIL REF. PT. MEAS. C.W. FROM X-AXIS
end if !for 'if ((itb(ibn) .eq. 1)'
```

C ... BODY #2 MOTION

```

au(ibn)= -1.d0 !SEE COMMENTS ABOVE
bu(ibn)= 0.d0
cu(ibn)= 0.d0
fu(ibn)= 0.d0
av(ibn)= 0.d0
bv(ibn)= 0.d0
cv(ibn)= 0.d0
fv(ibn)= 0.d0
ao(ibn)= 0.d0
bo(ibn)= 0.d0
co(ibn)= 0.d0
fo(ibn)= 0.d0
```

C ... BODY #3 GEOMETRY

```

else if !for 'if (ibn .eq. 1)'
$(ibn .eq. 3) then
itb(ibn) = 1!SELECT BODY TYPE FROM LIST ABOVE

if
$(itb(ibn) .eq. 1) then !CIRCLE
d(ibn)=1.d0!CIRCLE DIAMETER
zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
ngpts(ibn)=180 !# OF PANELS IN DISCRETE CIRCLE
nst(ibn)=0 !NO STRAKES -- DO NOT CHANGE

else if
$(itb(ibn) .eq. 2) then!RECTANGLE
a(ibn)=1.d0!RECTANGLE WIDTH
b(ibn)=1.d0!RECTANGLE THICKNESS
zc(ibn)=dcmplx(0,0) !RECTANGLE CENTER
theto(ibn)=0.d0*pi/4.d0!RECTANGLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS

else if
$(itb(ibn) .eq. 3) then!CIRCLE WITH STRAKES
d(ibn)=1.d0!CIRCLE DIAMETER
zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
ngpts(ibn)=180 !# OF PANELS IN DISCRETE CIRCLE
nst(ibn)=3 !# OF STRAKES
a(ibn)=0.25d0!HEIGHT OF STRAKES
hs(ibn)=.1d0!THICKNESS OF STRAKES

else if
$(itb(ibn) .eq. 4) then!NACA AIRFOIL
n4des(ibn)=0021 !DESIGNATE (NACA(0021) or NACA(9420)
c(ibn)=1.d0!NACA AIRFOIL CHORD LENGTH
zc(ibn)=dcmplx(0,0)!NACA AIRFOIL LEADING EDGE
theto(ibn)=0.d0*pi/4.d0!NACA AIRFOIL REF. PT. MEAS. C.W. FROM X-AXIS
end if !for 'if ((itb(ibn) .eq. 1)'
```

C ... BODY #3 MOTION

```

au(ibn)= -1.d0!SEE COMMENTS ABOVE
bu(ibn)= 0.d0
cu(ibn)= 0.d0
fu(ibn)= 0.d0
av(ibn)= 0.d0
bv(ibn)= 0.d0
cv(ibn)= 0.d0
fv(ibn)= 0.d0
ao(ibn)= 0.d0
bo(ibn)= 0.d0
co(ibn)= 0.d0
fo(ibn)= 0.d0

```

C ... BODY #4 GEOMETRY

```

else if !for 'if (ibn .eq. 1)'
$(ibn .eq. 4) then
itb(ibn) = 1!SELECT BODY TYPE FROM LIST ABOVE

if
$(itb(ibn) .eq. 1) then !CIRCLE
d(ibn)=1.d0!CIRCLE DIAMETER
zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
ngpts(ibn)=180 !# OF PANELS IN DISCRETE CIRCLE
nst(ibn)=0 !NO STRAKES -- DO NOT CHANGE

else if
$(itb(ibn) .eq. 2) then!RECTANGLE
a(ibn)=1.d0!RECTANGLE WIDTH
b(ibn)=1.d0!RECTANGLE THICKNESS
zc(ibn)=dcmplx(0,0) !RECTANGLE CENTER
theto(ibn)=0.d0*pi/4.d0!RECTANGLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS

else if
$(itb(ibn) .eq. 3) then!CIRCLE WITH STRAKES
d(ibn)=1.d0!CIRCLE DIAMETER
zc(ibn)=dcmplx(0,0) !CIRCLE CENTER
theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
ngpts(ibn)=180 !# OF PANELS IN DISCRETE CIRCLE
nst(ibn)=3 !# OF STRAKES
a(ibn)=0.25d0!HEIGHT OF STRAKES
hs(ibn)=.1d0!THICKNESS OF STRAKES

else if
$(itb(ibn) .eq. 4) then!NACA AIRFOIL
n4des(ibn)=0021 !DESIGNATE (NACA(0021) or NACA(9420)
c(ibn)=1.d0!NACA AIRFOIL CHORD LENGTH
zc(ibn)=dcmplx(0,0)!NACA AIRFOIL LEADING EDGE
theto(ibn)=0.d0*pi/4.d0!NACA AIRFOIL REF. PT. MEAS. C.W. FROM X-AXIS
end if !for 'if ((itb(ibn) .eq. 1)'
```

C ... BODY #4 MOTION

```

au(ibn)= -1.d0!SEE COMMENTS ABOVE
bu(ibn)= 0.d0
cu(ibn)= 0.d0
fu(ibn)= 0.d0

```

```

av(ibn)= 0.d0
bv(ibn)= 0.d0
cv(ibn)= 0.d0
fv(ibn)= 0.d0
ao(ibn)= 0.d0
bo(ibn)= 0.d0
co(ibn)= 0.d0
fo(ibn)= 0.d0

```

C ... BODY #5 GEOMETRY

```

else if      !for 'if (ibn .eq. 1)'
$(ibn .eq. 5) then
  itb(ibn) = 1!SELECT BODY TYPE FROM LIST ABOVE

  if
$(itb(ibn) .eq. 1) then !CIRCLE
  d(ibn)=1.d0!CIRCLE DIAMETER
  zc(ibn)=dcmplx(0,0)      !CIRCLE CENTER
  theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
  ngpts(ibn)=180           !# OF PANELS IN DISCRETE CIRCLE
  nst(ibn)=0               !NO STRAKES -- DO NOT CHANGE

  else if
$(itb(ibn) .eq. 2) then!RECTANGLE
  a(ibn)=1.d0!RECTANGLE WIDTH
  b(ibn)=1.d0!RECTANGLE THICKNESS
  zc(ibn)=dcmplx(0,0)      !RECTANGLE CENTER
  theto(ibn)=0.d0*pi/4.d0!RECTANGLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS

  else if
$(itb(ibn) .eq. 3) then!CIRCLE WITH STRAKES
  d(ibn)=1.d0!CIRCLE DIAMETER
  zc(ibn)=dcmplx(0,0)      !CIRCLE CENTER
  theto(ibn)=0.d0*pi/4.d0!CIRCLE REF. PT. MEAS. CLOCKWISE FROM X-AXIS
  ngpts(ibn)=180           !# OF PANELS IN DISCRETE CIRCLE
  nst(ibn)=3              !# OF STRAKES
  a(ibn)=0.25d0!HEIGHT OF STRAKES
  hs(ibn)=.1d0!THICKNESS OF STRAKES

  else if
$(itb(ibn) .eq. 4) then!NACA AIRFOIL
  n4des(ibn)=0021          !DESIGNATE (NACA(0021) or NACA(9420)
  c(ibn)=1.d0!NACA AIRFOIL CHORD LENGTH
  zc(ibn)=dcmplx(0,0)!NACA AIRFOIL LEADING EDGE
  theto(ibn)=0.d0*pi/4.d0!NACA AIRFOIL REF. PT. MEAS. C.W. FROM X-AXIS
end if      !for 'if ((itb(ibn) .eq. 1)'

```

C ... BODY #5 MOTION

```

au(ibn)= -1.d0!SEE COMMENTS ABOVE
bu(ibn)= 0.d0
cu(ibn)= 0.d0
fu(ibn)= 0.d0
av(ibn)= 0.d0
bv(ibn)= 0.d0
cv(ibn)= 0.d0
fv(ibn)= 0.d0
ao(ibn)= 0.d0

```

```

      bo(ibn)= 0.d0
      co(ibn)= 0.d0
      fo(ibn)= 0.d0

c      copy and modify above inputs here for additional bodies

      end if      !for 'if (ibn .eq. 1)'
      end do      !for 'do ibn = 1,inb'

C-----

C      ***** STARTING FLOW PERTURBATION *****

      perturb_vel = .true.      !TRUE TO ADD PERTURBING FLOW IN TEST_VFLOW2d
      perturb_vortex = .false. !TRUE TO INTRODUCE THE VORTEX

      if (perturb_vortex) then
        zfv(nfv) = (0.d0,-1.d0)!VORTEX POSITION
        wfv(nfv) = (0.d0,0.d0)!VORTEX VELOCITY
        acf(nfv) = 0.05d0      !VORTEX RADIUS
        gaf(nfv) = 0.1d0!VORTEX STRENGTH
        vidf(nfv) = new_vid(1) !VORTEX NUMBER
        nvf = 1!DO NOT CHANGE--INITIAL FREE BLOB COUNT
        fblob_index(nfv) = 0    !DO NOT CHANGE--FREE BLOB INDEX
      end if      !for 'if (perturb_vortex)'

C-----

C      ***** TURBULENCE MODEL *****

      turbulent = .false. !TRUE TO INCLUDE TURBULENCE MODEL
      t_model = 3 !1 = Baldwin-Lomax turbulence model
      !2 = Prandtl's mixing length model
      !3 = Smagorinsky model

C-----

C      ***** WALL LAYER VARIABLES *****

                                !SET nj > ngpts() IN pdim.h
                                !SET ny IN common.h (# OF GRID CELLS NORMAL
!TO BODY)
      yratio = 3.d0 !WALL LAYER THICKNESS/ngpts SEGMENT LENGTH
      gratio = 1.035d0!LENGTH INCREASE OF SUCCESSIVE ny CELLS
      !1.d0 < gratio < 1.1d0
      !WALL LAYER 1 BEGINS ON THE WALL
      y2 = 1.d0/3.d0!FRACTION OF WALL LAYER WHERE LAYER 2 BEGINS
      y3 = 2.d0/3.d0!FRACTION OF WALL LAYER WHERE LAYER 3 BEGINS

C-----

C      ***** FREE VORTEX MERGING VARIABLES *****

      merge = .true.      !TRUE TO MERGE NEAR VORTICES
ccc      sepn_dist = 1.3d0*pi/ngpts(1)
      sepn_dist = 1.3d0*(y3-y2)*yratio*pi*XL0/ngpts(1)
      sepn_dist = 1.5d-2
!MIN. DIST. BETWEEN VORTICES
      body_dist = 1.d0!MIN. DIST. FROM BODY BEFORE MERGING

```

```

min_nfv = 1.d3!MIN. # OF FREE VORTICES BEFORE MERGING

if(.not. merge)
$ min_nfv=1.d33 !DO NOT CHANGE

```

```

C-----

iinb = inb

return
end

```

A.3 Discussion of Input Variables

Subroutine `initial` is extensively commented to describe each of the input variables. While perhaps somewhat repetitive, this section contains more detailed discussion of the input data and variables.

A.3.1 Reference Values and Time Scale

The Reynolds number for the simulation is input via variable `reynolds_no`. VLFOW2D was written assuming nondimensional variables. However, any consistent set of units can be used for the data. The length and velocity scales used for the nondimensionalization and the Reynolds number are `XL0` and `U0`, respectively. The beginning and final time steps are `nts` and `ntf`. The time step taken by the inviscid portions of the code is `delt`. The minimum number of sub-time steps taken by the viscous wall layer calculations for each inviscid time step is `min_subs`. The wall layer section of the code has its own minimum time step routine to ensure numerical stability, so the wall layer can take many more (and smaller) time steps than `min_subs`, but it cannot take less. The time at the start of the simulation is `t`. The minimum size box used in the panel-to-point induced velocity calculations is `min_sbox_fac`. This is the ratio of the minimum box size to the largest panel length. A minimum value of 2.0 is recommended.

A.3.2 Body Geometry

The number of bodies in the simulation is `inb`. Although the code was written for multiple bodies, the code has not been checked for greater than one body. There are four geometry packages in the code: circle, rectangle, circle with strakes, and NACA 4-digit airfoil series. The only one that has been thoroughly used is the circle option. For a circular body, the input data are diameter, `d`; body center complex coordinates, `zc`; body angular orientation in radians, `theto`; number of panels defining the circle, `ngpts`; number of strakes, `nst` (`= 0` for circle). Input data required for the other geometry options are described in `initial`.

A.3.3 Body Motion

VFLOW2D contains a prescribed motion generator of the form $u = a + b \sin(ft + c)$, where a , b , and c are constants, f is frequency and t is time. There is one motion equation for the x direction and one for y . For the x direction, the input variables are `au`, `bu`, `cu`, and `fu`. For the y direction, they are `av`, `bv`, `cv`, and `fv`. There also is a rotational motion generator of the

same form, $\omega = a + b \sin(ft + c)$. The rotational motion input variables are `ao`, `bo`, `co`, and `fo`.

A.3.4 Initial Flow Perturbation

When a cylinder begins steady motion from an impulsive start, the wake is initially symmetric. This is an unstable condition, and the wake will soon begin periodic shedding, forming the characteristic von Karman vortex street. The numerical simulation of an impulsively started cylinder exhibits similar behavior. Since the asymmetries are started by numerical roundoff errors, the start of the periodic shedding can vary greatly with small changes in input data or the code itself. In order to more accurately control the start of the periodic shedding, we have provided a mechanism to input an initial flow perturbation.

Two perturbation flags are provided, `perturb_vel` and `perturb_vortex`. Both can be set to `.false`. Only one can be set to `.true`. If `perturb_vel` is `.true`., the y-component of velocity is set equal to -0.1 for time < 1.0. This is currently set in the driver program and can be changed there (see section A.1). If `perturb_vortex` is `.true`., then a single vortex is inserted into the flowfield at the start of the simulation. The parameters for this initial vortex are set within `initial` with variables `zfv`, `wfv`, `acf`, and `gaf`, for position, velocity, size, and strength, respectively. The remainder of the variables within this section of `initial` must not be changed.

A.3.5 Turbulence Model

[CAUTION: The turbulence model implementation has not been fully tested.]

Three turbulence model options have been implemented within the wall layer. No turbulence model has been implemented outside the wall layer. The logical flag that activates the turbulence option is `turbulent`. The flag that selects the desired model is `t_model`. A value of 1 = Baldwin-Lomax model, 2 = Prandtl's mixing length model, and 3 = Smagoransky model.

A.3.6 Wall Layer Variables

The wall layer is a relatively thin meshed region adjacent to a body's surface (see Chapter 2 on page 19). The variables used to define the wall layer are as follows. `y_ratio` is the thickness of the wall layer in terms of the maximum length of segments used to define the body's surface. `g_ratio` is the ratio of adjacent cell thicknesses normal to the wall. `y2` defines the starting location of the second wall sublayer as a fraction of the wall layer thickness. `y3` defines the starting location of the third wall sublayer as a fraction of the wall layer thickness.

A.3.7 Vortex Merging

The logical flag `merge` activates the vortex merging option. Only free vortices are eligible for merging, not transition vortices. The variable `sepn_dist` specifies the merge criterion distance. Vortices closer than `sepn_dist` are merged. `body_dist` specifies the minimum distance from the body's surface before merging is allowed. Vortices closer to the body than `body_dist` are not merged regardless of their `sepn_dist`. `min_nfvs` specifies the minimum number of free vortices that must be present in the simulation before merging is allowed.

A.4 Common Blocks

Data are passed between different sections of the code using common blocks. Each common block is defined only once in a separate file identified with a ".h" extension. They are included into each subroutine/function that needs them using `include` statements. The ones that are of most interest to the user are described below. The others can be found in the source code distribution.

A.4.1 bvel.h

Common block `bvel` contains body velocities. It can be used to pass body velocities from a structural code to the fluids code. If the bodies are rigid, only arrays `om` and `wc` are required to be updated. Each array has one value per body.

```
c ***** file bvel.h *****
c
c This common block contains arrays used to store body velocities
c
c   gam      = amount of circulation that must be added at each time
c              step on each surface joint to impose correct boundary
c              conditions
c   gamold   = same as "gam" but at previous time step
c   om       = rotational velocity of each body
c   wc       = complex velocity (u-iv) of each body center
c   wflupan  = complex velocity (u-iv) induced by fluid at each surface
c              panel
c   wjnt     = complex velocity (u-iv) of each surface joint
c   wpan     = complex velocity (u-iv) of each surface panel
c   wpanold  = complex velocity (u-iv) of each surface panel from previous
c              time step
c
c
c   real*8 om, gam, gamold
c   complex*16 wc, wjnt, wpan, wflupan, wpanold
c   common/bvel/wc(nb), om(nb), wjnt(nj,ns,nb), gam(nj,ns,nb),
c   $ gamold(nj,ns,nb), wpan(nj-1,ns,nb), wflupan(nj-1,ns,nb),
c   $ wpanold(nj-1,ns,nb)
```

A.4.2 load.h

The common block `load` contains surface pressures and integrated forces and moments. It can be used to pass force and moment information to a structure code. The arrays `cx`, `cy`, and `cm` contain the data of interest. Each array has one value per body.

```
c ***** file 'load.h' *****
c
c This block holds data for computing body pressures and loads
c
c   cpjnt     = pressure coefficients at body joints
c   cpmid     = pressure coefficients at surface panel midpoints
c   cx        = streamwise force coefficient due to pressure
c   cy        = lateral force coefficient due to pressure
c   xm        = moment reference point x coordinate
c   ym        = moment reference point y coordinate
c   cm        = moment coefficient
c   omega_jnt = vorticity values at body joints
```

```

c      cxf      = streamwise force coefficient due to skin friction
c      cyf      = sideways force coefficient due to skin friction
c      yplusmin = minimum value of boundary layer parameter y+
c      yplusmax = maximum value of boundary layer parameter y+

      real*8 cpjnt, cpmid, cx, cy, xm, ym, cm, omega_jnt, cxf, cyf,
$         yplusmin, yplusmax
      common/load/cpjnt(nj,ns,nb), cpmid(nj,ns,nb),
$         cx(nb), cy(nb), xm(nb), ym(nb), cm(nb),
$         omega_jnt(nj,ns,nb), cxf(nb), cyf(nb), yplusmin,
$         yplusmax

```

A.5 Code Output

VFLOW2D output is described in Appendix B. We have not provided any routines to output files for graphical viewing of the flowfield because they are too site specific. Also, if one tries to write out all vortices at all time steps, the file can become huge. The common blocks contained in file vort.h hold the flowfield variables. Data can be extracted from these blocks and written to files that are compatible with the user's computer system.

(this page intentionally left blank)

Appendix B: Example Problem

B.1 Example Output

This appendix presents examples of the tabular and screen output produced by VFLOW2D. The example problem used to produce this output is that shown in Appendix A. We have truncated the actual output files here to conserve space. Listings of the entire output would take many pages.

B.2 Output File - loads.out

The following is an example of the loads.out output file from VFLOW2D. Only the first portion of the file, through time = 0.5, is shown. In this example, the motion is in the negative x direction. The first column is nondimensional time. The next two columns are the x and y force coefficients due to pressure. The next two are the x and y force coefficients due to skin friction. The last two columns are the total x and y force coefficients.

```
#reference velocity = 1.0
#reference length = 1.0
#Reynolds number = 0.100E+06
#body id = 1
```

#	time	cxp	cyp	cxp	cyp	cx	cy
0.000000E+00	0.157037E+03	0.157037E+02	0.000000E+00	0.000000E+00	0.157037E+03	0.157037E+02	
0.100000E-01	0.392619E+00	0.393015E-01	0.418155E-01	0.418169E-02	0.434435E+00	0.434832E-01	
0.200000E-01	0.140505E+00	0.140894E-01	0.334249E-01	0.334254E-02	0.173930E+00	0.174320E-01	
0.300000E-01	0.124456E+00	0.124553E-01	0.286813E-01	0.286814E-02	0.153138E+00	0.153235E-01	
0.400000E-01	0.115237E+00	0.115353E-01	0.255737E-01	0.255735E-02	0.140811E+00	0.140926E-01	
0.500000E-01	0.109685E+00	0.109873E-01	0.233246E-01	0.233244E-02	0.133009E+00	0.133198E-01	
0.600000E-01	0.105986E+00	0.106273E-01	0.215920E-01	0.215917E-02	0.127578E+00	0.127865E-01	
0.700000E-01	0.103652E+00	0.104011E-01	0.201991E-01	0.201988E-02	0.123851E+00	0.124210E-01	
0.800000E-01	0.101653E+00	0.102055E-01	0.190455E-01	0.190452E-02	0.120698E+00	0.121100E-01	
0.900000E-01	0.100001E+00	0.100393E-01	0.180685E-01	0.180682E-02	0.118070E+00	0.118461E-01	
0.100000E+00	0.981744E-01	0.986907E-02	0.172262E-01	0.172259E-02	0.115401E+00	0.115917E-01	
0.110000E+00	0.972715E-01	0.978097E-02	0.164901E-01	0.164899E-02	0.113762E+00	0.114300E-01	
0.120000E+00	0.965742E-01	0.971119E-02	0.158393E-01	0.158391E-02	0.112413E+00	0.112951E-01	
0.130000E+00	0.959116E-01	0.964464E-02	0.152584E-01	0.152582E-02	0.111170E+00	0.111705E-01	
0.140000E+00	0.956167E-01	0.961808E-02	0.147356E-01	0.147354E-02	0.110352E+00	0.110916E-01	
0.150000E+00	0.952288E-01	0.957417E-02	0.142617E-01	0.142615E-02	0.109490E+00	0.110003E-01	
0.160000E+00	0.944569E-01	0.950193E-02	0.138290E-01	0.138287E-02	0.108286E+00	0.108848E-01	
0.170000E+00	0.942648E-01	0.948441E-02	0.134319E-01	0.134317E-02	0.107697E+00	0.108276E-01	
0.180000E+00	0.940142E-01	0.946122E-02	0.130658E-01	0.130655E-02	0.107080E+00	0.107678E-01	
0.190000E+00	0.939918E-01	0.944842E-02	0.127266E-01	0.127264E-02	0.106718E+00	0.107211E-01	
0.200000E+00	0.939123E-01	0.943318E-02	0.124109E-01	0.124107E-02	0.106323E+00	0.106743E-01	
0.210000E+00	0.945282E-01	0.948342E-02	0.121164E-01	0.121162E-02	0.106645E+00	0.106950E-01	
0.220000E+00	0.956671E-01	0.960113E-02	0.118408E-01	0.118405E-02	0.107508E+00	0.107852E-01	
0.230000E+00	0.959379E-01	0.962246E-02	0.115820E-01	0.115816E-02	0.107520E+00	0.107806E-01	
0.240000E+00	0.956516E-01	0.959198E-02	0.113377E-01	0.113373E-02	0.106989E+00	0.107257E-01	
0.250000E+00	0.960279E-01	0.962175E-02	0.111068E-01	0.111063E-02	0.107135E+00	0.107324E-01	
0.260000E+00	0.969745E-01	0.971358E-02	0.108881E-01	0.108876E-02	0.107863E+00	0.108023E-01	
0.270000E+00	0.975069E-01	0.976661E-02	0.106805E-01	0.106800E-02	0.108187E+00	0.108346E-01	
0.280000E+00	0.973780E-01	0.974208E-02	0.104827E-01	0.104821E-02	0.107861E+00	0.107903E-01	

0.290000E+00	0.972165E-01	0.972229E-02	0.102935E-01	0.102929E-02	0.107510E+00	0.107516E-01
0.300000E+00	0.985034E-01	0.983956E-02	0.101128E-01	0.101121E-02	0.108616E+00	0.108508E-01
0.310000E+00	0.991494E-01	0.990136E-02	0.993978E-02	0.993904E-03	0.109089E+00	0.108953E-01
0.320000E+00	0.990771E-01	0.988434E-02	0.977343E-02	0.977263E-03	0.108850E+00	0.108616E-01
0.330000E+00	0.100603E+00	0.100306E-01	0.961368E-02	0.961281E-03	0.110216E+00	0.109919E-01
0.340000E+00	0.100592E+00	0.100267E-01	0.945973E-02	0.945878E-03	0.110052E+00	0.109726E-01
0.350000E+00	0.100519E+00	0.100041E-01	0.931081E-02	0.930979E-03	0.109830E+00	0.109351E-01
0.360000E+00	0.102273E+00	0.101787E-01	0.916709E-02	0.916598E-03	0.111440E+00	0.110953E-01
0.370000E+00	0.103193E+00	0.102629E-01	0.902813E-02	0.902693E-03	0.112221E+00	0.111656E-01
0.380000E+00	0.104141E+00	0.103588E-01	0.889341E-02	0.889211E-03	0.113034E+00	0.112481E-01
0.390000E+00	0.105202E+00	0.104536E-01	0.876281E-02	0.876144E-03	0.113965E+00	0.113297E-01
0.400000E+00	0.106219E+00	0.105447E-01	0.863589E-02	0.863447E-03	0.114855E+00	0.114081E-01
0.410000E+00	0.107303E+00	0.106465E-01	0.851239E-02	0.851090E-03	0.115815E+00	0.114976E-01
0.420000E+00	0.108450E+00	0.107472E-01	0.839208E-02	0.839050E-03	0.116843E+00	0.115862E-01
0.430000E+00	0.109610E+00	0.108569E-01	0.827473E-02	0.827302E-03	0.117885E+00	0.116842E-01
0.440000E+00	0.110858E+00	0.109698E-01	0.816016E-02	0.815827E-03	0.119019E+00	0.117856E-01
0.450000E+00	0.112087E+00	0.109806E-01	0.804816E-02	0.804612E-03	0.120135E+00	0.117852E-01
0.460000E+00	0.113239E+00	0.111775E-01	0.793858E-02	0.793626E-03	0.121178E+00	0.119711E-01
0.470000E+00	0.114566E+00	0.113553E-01	0.783123E-02	0.782879E-03	0.122397E+00	0.121381E-01
0.480000E+00	0.116176E+00	0.114684E-01	0.772597E-02	0.772349E-03	0.123902E+00	0.122408E-01
0.490000E+00	0.117506E+00	0.115942E-01	0.762271E-02	0.762006E-03	0.125129E+00	0.123562E-01
0.500000E+00	0.119160E+00	0.117192E-01	0.752127E-02	0.751836E-03	0.126681E+00	0.124710E-01

B.3 Screen Output

The following is an example of the screen output produced by VFLOW2D while running. Each block labeled "time cycle" represents one inviscid time step. Each line within the blocks represents one viscous (wall layer) time step. Only the first five inviscid time steps are shown here. The first column is the total time within the simulation. The second column is the viscous time step. The last two columns give the conservation of circulation within the wall layer remap step and are printed to monitor the status of the calculation. A value of 1.0 would be perfect conservation of circulation. These two columns are for the positive and negative circulation layers, respectively.

time cycle no. 1			
8.00669E-04	8.00669E-04	0.999784	0.999784
2.30125E-03	1.50058E-03	0.999780	0.999780
3.89893E-03	1.59768E-03	0.999780	0.999780
5.58930E-03	1.69037E-03	0.999780	0.999780
7.31021E-03	1.72091E-03	0.999780	0.999780
9.02811E-03	1.71790E-03	0.999780	0.999780
1.00000E-02	9.71888E-04	0.999784	0.999784
time cycle no. 2			
1.04788E-02	4.78815E-04	0.999786	0.999786
1.22271E-02	1.74828E-03	0.999780	0.999780
1.38795E-02	1.65244E-03	0.999781	0.999781
1.55156E-02	1.63602E-03	0.999781	0.999781
1.69316E-02	1.41607E-03	0.999782	0.999782
1.83083E-02	1.37668E-03	0.999782	0.999782
1.95620E-02	1.25371E-03	0.999783	0.999783
2.00000E-02	4.37981E-04	0.999785	0.999785
time cycle no. 3			
2.10836E-02	1.08355E-03	0.999783	0.999783
2.21761E-02	1.09258E-03	0.999783	0.999783
2.30514E-02	8.75231E-04	0.999783	0.999783
2.39224E-02	8.70990E-04	0.999783	0.999783
2.47870E-02	8.64606E-04	0.999783	0.999783
2.54933E-02	7.06367E-04	0.999784	0.999784
2.63325E-02	8.39143E-04	0.999783	0.999783

2.70209E-02	6.88457E-04	0.999783	0.999784
2.78870E-02	8.66045E-04	0.999783	0.999783
2.85400E-02	6.53051E-04	0.999783	0.999783
2.93407E-02	8.00674E-04	0.999783	0.999783
2.99912E-02	6.50460E-04	0.999783	0.999783
3.00000E-02	8.83840E-06	0.999784	0.999784
time cycle no. 4			
3.07863E-02	7.86291E-04	0.999783	0.999783
3.14613E-02	6.74993E-04	0.999783	0.999783
3.22428E-02	7.81502E-04	0.999783	0.999783
3.29361E-02	6.93294E-04	0.999783	0.999783
3.37248E-02	7.88702E-04	0.999783	0.999783
3.44407E-02	7.15949E-04	0.999783	0.999783
3.51486E-02	7.07840E-04	0.999783	0.999783
3.59019E-02	7.53311E-04	0.999783	0.999783
3.66725E-02	7.70632E-04	0.999783	0.999783
3.74872E-02	8.14675E-04	0.999782	0.999782
3.83592E-02	8.72026E-04	0.999782	0.999782
3.89840E-02	6.24816E-04	0.999782	0.999782
3.97086E-02	7.24617E-04	0.999782	0.999782
4.00000E-02	2.91355E-04	0.999783	0.999783
time cycle no. 5			
4.05951E-02	5.95102E-04	0.999782	0.999782
4.13129E-02	7.17789E-04	0.999782	0.999782
4.19594E-02	6.46559E-04	0.999782	0.999782
4.25504E-02	5.90964E-04	0.999782	0.999782
4.32755E-02	7.25053E-04	0.999782	0.999782
4.38964E-02	6.20898E-04	0.999782	0.999782
4.46079E-02	7.11568E-04	0.999782	0.999782
4.52770E-02	6.69097E-04	0.999782	0.999782
4.59387E-02	6.61648E-04	0.999782	0.999782
4.66402E-02	7.01494E-04	0.999782	0.999782
4.74170E-02	7.76816E-04	0.999781	0.999781
4.80080E-02	5.91034E-04	0.999782	0.999782
4.88128E-02	8.04814E-04	0.999781	0.999781
4.95273E-02	7.14480E-04	0.999781	0.999781
5.00000E-02	4.72683E-04	0.999782	0.999782

(this page intentionally left blank)

Appendix C: Numerical Recipes

VFLOW2D uses five subroutines and functions from *Numerical Recipes in FORTRAN* [30]. This was judged to be more efficient than developing our own routines from scratch. The source code for this software is copyrighted by Numerical Recipes Software and is therefore not included with the VFLOW2D source code. The Numerical Recipes source code can be purchased at a reasonable fee from Numerical Recipes Software at <http://www.nr.com>.

We modified the original Numerical Recipes routines to make all real variables double precision. The Numerical Recipes routines that are called from VFLOW2D are:

subroutine nsort(n,arr) - Sorts an integer array *arr(1:n)* into ascending numerical order using the Quicksort algorithm. This is a modification of the Numerical Recipe sort routine in that all variables within the routine are declared to be integer.

subroutine lubksb(am,n,np,indx,bm) - Solves the set of n linear equations $AX=B$. A is input not as the original matrix but rather as the LU decomposition of A from subroutine *ludcmp*.

subroutine ludcmp(am,n,np,indx,d) - Computes the LU decomposition of matrix A to be used with *lubksb* to solve the linear equation set $AX=B$.

subroutine mprove(a,alud,n,np,indx,b,x) - Improves the solution vector X of the linear equation set $AX=B$. Used in conjunction with *lubksb* and *ludcmp*.

subroutine svbksb(u,w,v,m,n,np,np,b,x) - Solves $AX = B$ for a vector X , where A is specified by the arrays u , w , v as returned by *svdcmp*.

subroutine svdcmp(a,m,n,np,np,w,v) - Computes singular value decomposition (SVD) of matrix A .

function pythag(a,b) - Computes $\sqrt{a^2 + b^2}$ without destructive underflow or overflow.

(this page intentionally left blank)

Distribution

<p>1 Prof. Peter S. Bernard
Dept. of Mech. Eng.
University of Maryland
College Park, MD 20742-3035</p>	<p>2 University of New Mexico
Dept. of Mech. Eng.
Albuquerque, NM 87131
Attn: Dr. M. S. Ingber
Dr. C. R. Truman</p>
<p>1 Paul Devlin
Texaco, Inc.
P.O. Box 430
Bellaire, TX 77402-0430</p>	<p>1 Ernesto Valenzuela
EV Consulting, Inc.
801 N. Eldridge
Houston, TX 77079</p>
<p>1 Dr. John R. Grant
Naval Undersea Center
1176 Howell St.
Bldg. 108, Code 8233
Newport, RI 02841-1708</p>	
<p>1 Prof. C. E. Hailey
Mech. & Aero. Eng.
Utah State Univ.
UMC 4130
Logan, UT 84322-4130</p>	<p>1 MS 1380 1315 F. A. Barker
1 MS 0825 9115 W. H. Rutledge
1 MS 0826 9113 W. L. Hermina
1 MS 0827 9117 R. Griffith
1 MS 0834 9112 A. C. Ratzel
1 MS 0836 9100 C. W. Peterson
5 MS 0835 9111 S. N. Kempka
10 MS 0835 9111 A. A. Gossler
5 MS 0835 9111 G. F. Homicz
2 MS 0835 9111 J. H. Strickland
5 MS 0835 9111 W. P. Wolfe
1 MS 0835 9121 J. S. Peery
1 MS 0841 9100 T. C. Bickel
1 MS 9018 8945-1 Central Technical
Files
2 MS 0899 9616 Technical Library
1 MS 0612 9612 Review & Approval
Desk for DOE/OSTI
1 MS 1380 1321 Technology Transfer</p>
<p>1 Dr. Stephen Huyer
Naval Undersea Weapon Center
Hydrodynamics Branch, Code 8233
Building 1302/1
Newport, RI 02841</p>	
<p>2 Prof. Anthony Leonard
Graduate Aeronautics Lab.
California Institute of Technology
Pasadena, CA 91125
Attn: Lorena Barba</p>	
<p>1 Prof. Jeffery S. Marshall
Iowa Institute of Hydraulic Research
University of Iowa
300 S. Riverside Dr.
Iowa City, IA 52242-1585</p>	
<p>1 Dr. M. Nitsche
Dept. of Math. & Stat.
University of New Mexico
Albuquerque, NM 87131</p>	